**Avanços obtidos na semana**

Tive uma rápida conversa com o professor Ranieri,no qual mostrei a ele a implementação do protocolo QUIC, e mesmo não conhecendo muito sobre o protocolo QUIC(pois eu não expliquei muito), o mesmo me fez pensar sobre uma outra abordagem para entender melhor o protocolo. A abordagem é ver o protocolo de baixo para cima(abordagem Down-TOP), estudando e implementando o udp. Eu pesquisei por um tempo(equivalente a 1 dia em horas líquidas), e eu tive um insight de que o QUIC é um UDP “organizado”. O UDP tem somente poucos parâmetros de segurança(ou seja, nenhum), com somente o envio de dados(semelhante à camada de rede). O QUIC está “acima” do UDP(multiplexado), mas com algoritmos que implementam confiabilidade e segurança. Esse insight é importante, pois agora eu entendo melhor alguns conceitos da camada de transporte e entendo melhor como o protocolo QUIC funciona. Essa é a base de como o QUIC possui um RTT de até 1(igual ao UDP). Portanto, o conceito mais importante que eu terei que estudar(depois de entender outros aspectos intrínsecos ao protocolo), é como subdividir o pacote QUIC em datagramas UDP. Tenho que observar outras abordagens(MSQUIC e QUICHE).

Avanços no código:

* Implementei um algoritmo simples de codificação e decodificação de bytes no campo tamanho(Length) do pacote longo(Long Header).
* Implementei uma biblioteca simples de envio de dados udp. Fiz a mesma com ajuda do chat gpt, pois eu queria a menor utilização possível de bibliotecas. Posteriormente irei procurar implementações já feitas para mitigar um trabalho desnecessário(“reinventar a roda”).
* Implementei o setup do zeroRTT e HandshakePacket. Fiz um arquivo de testes que através da compilação do arquivo , eu posso construir a estrutura necessária. Por enquanto, somente a construção da estrutura é correta, ainda falta muita coisa n o código para estar minimamente aceitável
* OBSERVAÇÃO: O nome do projeto que eu criei no Github **Dquic** é para associar o meu nome a ele(Davi) ou a palavra **Dynamic,** dando assim dois sentidos interessantes.

Avanços na apresentação(ainda não mostrados ao professor):

* Quando eu apresento para pessoas que não conhecem muito sobre o tema, mas tem uma certa familiaridade(alunos por exemplo), eu consigo explicitar o problema com certa facilidade. Faço isso através de diversas analogias e através de um artefato da comunicação que permite receber o feedback(através da pergunta “entendeu ?”). Com esse feedback, eu consigo adaptar a apresentação. Entretanto, através de dois conhecidos professores da língua Portuguesa, foi feita uma observação acerca do fato de que numa apresentação expositiva(como as que eu estou me preparando para fazer), não se deve perguntar durante o espaço de exposição(em que não é permitido perguntas), somente em outro momento ou num espaço de perguntas é adequado fazer isso. Num ambiente de sala de aula seria adequado isso(procurar por feedbacks para ajustar a apresentação).
* Consigo organizar uma ordem de pensamentos melhor para exposição(provavelmente era a falta de conhecimento de caso). Com as conversas e contínuos insights eu consegui avançar um pouco mais nisso.

**Controle de Fluxo:**

O controle do fluxo é feito a partir de uma análise do rtt(tempo de ida e volta do pacote), e essas informações são salvas no quadro MAX\_STREAM\_DATA. Esse quadro é atualizado com o parâmetro máximo de troca de dados frequentemente, caso o fluxo seja maior que o quadro, irá ocorrer um erro de protocolo. Os limites de fluxos são aumentados através de quadros MAX\_STREAMS(que podem ser ignorados).

**\*\* Estudo sobre implementação do QUIC(RFC 9369-QUIC Versão 2 ) \*\***

======================================================================

Estudei também algumas comparações acerca da comparação de desempenho do protocolo QUIC e TCP(alguns artigos que estão no fim do documento), e algumas soluções para os problemas do TCP(para eu comparar com o QUIC). A RFC 2001(<https://www.rfc-editor.org/rfc/rfc2001> ) que fala de alguns algoritmos que melhoram a eficiência do protocolo TCP, foi dito que existem algumas maneiras de ter o controle de congestionamento ocasionado pelo envio de pacotes. As três maneiras são boas práticas:

1-Enviar os pacotes aos poucos até atingir o limite(caso exista uma perda de pacote o transmissor diminui o fluxo pela metade).

2-Se ele receber 3 ou mais ACKs então ele retransmite imediatamente o pacote que parece ser o perdido(sem esperar o temporizador de retransmissão).

3-O pacote de retransmissão não é feito a inicialização lenta(passo 1) novamente, evitando atrasos, tendo assim uma recuperação rápida dos pacotes.

Essas boas práticas foram melhorias que foram implementadas no TCP, através do avanço da tecnologia, entretanto, mesmo com isso, ainda há alguns problemas referentes ao funcionamento das aplicações(retirados do artigo: <https://www.net.in.tum.de/fileadmin/TUM/NET/NET-2016-09-1/NET-2016-09-1_06.pdf> )como por exemplo:

* O fluxo de dados(stream) no TCP é mantido pelo número da porta. Portanto para realizar uma conexão full duplex, uma porta cliente se conecta a uma porta servidor, criando dois fluxos de dados independentes que podem funcionar ao mesmo tempo. Se você quer se conectar num servidor HTTP não persistente utilizando o TCP, você precisará criar várias conexões a cada acesso(diminuindo muito o desempenho).
* Um outro exemplo, é se você precisar enviar vários tipos de dados de diferentes aplicações(de diferentes portas). O TCP irá se conectar a cada porta, mantendo várias conexões em paralelo(já que a instância da conexão é o número da porta).
* Caso você use o HTTP/2 que possui suporte ao TLS, há uma latência maior por causa do protocolo TLS suportado(que utiliza o handshake TCP ), e há o estabelecimento da comunicação em TCP também(mesmo sendo melhorado em relação à versão anterior)
* No HTTP/2 existe o problema de bloqueio de cabeçalho(Head-of-Line blocking), em que caso uma solicitação ou resposta for perdida, todos os pacotes subsequentes que compartilham a mesma conexão TCP terão que esperar que o pacote que foi perdido sejam enviados(mesmo eles podendo ser enviados). Assim causando um atraso. Para fins de informação, caso queira pesquisar sobre o que seria essa técnica de compactação de cabeçalho do HTTP/2, pesquise por HPACK: <https://blog.cloudflare.com/hpack-the-silent-killer-feature-of-http-2/> ou a rfc que fala dessa técnica, a rfc 7451 (<https://httpwg.org/specs/rfc7541.html> ou <https://datatracker.ietf.org/doc/html/rfc7541> ). Existe um artigo que fala sobre isso(<http://129.69.170.1/en/Content/Publications/Archive/Sf_GLOBECOM2006_36516.pdf> )

O QUIC resolve esses problemas com a funcionalidade de ID’s de conexão, permitindo que uma stream não afete outra e que cada stream tenha o próprio estado de fluxo, tendo assim uma multiplexação mais eficiente.

======================================================================

Um novo protocolo que eu aprendi sobre(inicialmente) orientado à conexão é o SCTP:

* <https://www.inf.ufrgs.br/~cechin/Net/sctp/sctp.html>
* <https://www.ietf.org/rfc/rfc3257.txt>
* <https://www.ietf.org/rfc/rfc3286.txt>

======================================================================

Também baixei a nova edição de Sistemas distribuídos de Tanenbaum e Maarten van Steen para entender melhor sobre a área de IOT. Eu consegui baixar o livro, pois a versão digital foi disponibilizada pelo professor Maarten(autor) na internet, no qual foi somente necessária o cadastro do GMAIL(o livro vem personalizado com o GMAIL)

<https://www.distributed-systems.net/>

<https://www.distributed-systems.net/index.php/books/>

======================================================================

Implementação do QUIC em rust(para estudo): <https://github.com/cloudflare/quiche/tree/7ab6a55cfe471267d61e4d28ba43d41defcd87e0/quiche>

A implementação é feita em rust, mas é importante para fins comparativos(na implementação que estou construindo)

======================================================================

Fui apresentado a um site pelo professor Ranieri, que contém várias informações pertinentes no que se refere a alguns protocolos da WEB: <https://hpbn.co/>