Resumo - A QUIC Implementation for ns-3

Renê Cardozo
rene.cardozo@usp.br

1 Motivação

Pesquisas recentes tem focado em tópicos relacionados a camada de transporte, em especial gerando novos protocols ou extensões para o TCP, como o SCTP (Stream Control Transmission Protocol), QUIC (Quick UDP Internet Connections), e MPTCP (Multi-path TCP). Ainda, outros tipo de algoritmos para controle de congestionamento forem elaborados, visando adaptar tecnologias existentes para novas formas de comunicação, como RANs (Radio Access Networks). Neste artigo, foi integrada uma implementação nativa do QUIC para o ns-3, o qual é compatível com a versão 13 do rascunho de padronização pela IETF (Internet Engineering Task Force). Esta implementação é compatível com outras aplicações da pilha de protocolos TCP/IP do ns-3.

Apesar de existirem diversas implementações reais do protocolo QUIC, sua integração com o ns-3 possui diversos problemas. As funcionalidades de segurança com o TLS (Transport Layer Security) exigem a geração e configuração de certificados de segurança. A integração de implementações reais com o ns-3 podem gerar incompatibilidades com certos sistemas operacionais. Ainda, uma integração direta não permitiria aos pesquisadores alterar parâmetros do protocolo ou testar novas funcionalidades. Por fim, aplicações do ns-3 para a simulação de algoritmos de congestionamento são baseadas na estrutura de sockets TCP do ns-3, as quais não podem ser utilizadas sem uma implementação nativa. O módulo proposto foi elaborado de maneira a permitir o uso de diversas implementações de algoritmos de controle de congestionamento junto ao QUIC, o que torna mais fácil a comparação destes algoritmos, enquanto implementações reais apenas permitem certas adaptações do NewReno e CUBIC.

2 Descrição do QUIC

O QUIC, assim como o TCP, utiliza ACKs (acknowledgments) para inferir o estado da conexão e reagir adaptando sua janela de congestionamento ou retransmitindo os pacotes necessários. Porém, o algoritmo de recuperação em caso de perda de pacotes é melhores: ACKs são diferenciados entre fora de ordem ou por perda e suportam nativamente SACKs (Selective Acknowledgments) com um numero maior de blocos do que o TCP. Além disso, o QUIC provê mais

informações para o algoritmo de controle de congestionamento, aumentando a precisão de suas estimativas.

QUIC foi projetado para ser um protocolo multiplexador, no qual os pacotes da dados podem pertencer a diferentes streams de dados, tendo cada qual um controle de fluxo diferente. Esta função combina perfeitamente com a interface do HTTP/2, evitando o fenômeno do head-of-line blocking presente em conexões TCP. Ainda, o protocolo possui integração direta com funcionalidades do TLS 1.3, tornando possível o estabelecimento de conexões seguras em apenas 1 RTT (Round-Time Trip) e, para conexões posteriores, é possível reduzir para 0 RTT. Por fim, o QUIC também possui identificadores para cada conexão, os quais previnem engasgos na conexão em casos de mudanças de IP causadas por mudanças em NATs (Network Address Translators) ou movimentação em redes sem fio.

3 Implementação do Módulo

O protocolo foi implementado baseando-se em outro anterior: ns-3 internet, o qual já possui outros modelos relacionados à camada de transporte como UDP e TCP. O código assemelha-se a implementação do TCP, uma ve que separa o controle de congestionamento da lógica de sockets principal.

3.1 Sockets

QuicSocketBase (que estende QuicSocket) é a classe principal da implementação, modelando sua lógica de sockets. Cada cliente possuirá um único objeto QuicSocketBase, enquanto o servidor terá uma para cada conexão recebida. Estes objetos recebem e transmitem pacotes e ACKs, realizam o controle de congestionamento, realizam o handshake e troca de parâmetros, detectam e executam retransmissões, bem como controla a máquina de estados do protocolo. O objeto também possui ponteiros para os buffers de transmissão e recepção (QuicSocketTxBuffer e QuicSocketRxBuffer), para a máquina de estados (QuicSocketState) e para TcpCongestionOps, classe que realiza operações de compatibilidade e controle de congestionamento com algoritmos TCP.

Os sockets do QUIC são atrelados a um socket UDP por meio do objeto QuicL4Protocol, que também realiza a criação do objeto QuicSocketBase, sinaliza ao socket UDP para atrelar-se a conexão, e realiza a entrega dos pacotes do socket UDP para QuicSocketBase.

3.2 Streams e Pacotes

Uma stream de dados é modelada pela classe QuicStreamBase (que estende QuicStream), sendo responsável por armazenar um buffer de dados da aplicação, controlar o fluxo da stream, e trocar dados com a aplicação. De maneira similar a QuicSocketBase, QuicStreamBase possui ponteiros para buffers de tranmissão e recepção (QuicStreamTxBuffer e QuicStreamRxBuffer). A conexão entre

streams e seu socket é realizada pela classe QuicL5Protocol, contendo um vetor de ponteiros para cada uma das streams. Esta classe cria e configura cada uma das streams, realizando a troca de pacotes entre elas e seu socket.

Os headers dos pacotes são modelados pela classe QuicHeader, que estende a classe do ns-3 Header. Os headers do QUIC possuem dois formatos diferentes: long ou short, de acordo com a quantidade de informação no header.

Os dados dos pacotes são carregados por frames, os quais são mapeados para streams. Cada frame pode possuir um subheader, o qual é descrito pela classe QuicSubheader, responsável por especificar o tipo do frame.

3.3 Buffers

Os buffers são divididos entre sockets e streams:

- QuicSocketTxBuffer: armazena streams e frames de controle que serão enviados pelo QuicSocketBase; retorna pacotes, compostos por um ou mais frames, para o respectivo socket. Este buffer possui uma lista de pacotes para os ainda não transmitidos e outra para os já transmitidos mas que ainda não receberam ACK, caso precisem de retransmissão. Cada item armazenado nestas listas está encapsulado em uma classe QuicSocketTxItem. Ainda há um sistema de prioridade pra certos frames da stream 0, que deverão ser transmitidos na primeira oportunidade.

- QuicStreamTBuffer: armazena pacotes da aplicação até que recebam ACK, porém não é responsável por efetuar retransmissões, as quais, segundo o draft da IETF são apenas executadas em nível de socket. Cada pacote enviado possui uma marcação do deslocamento de bytes com relação ao início da stream.

- QuicStreamRxBuffer: Os pacotes recebidos são lidos e particionados em subframes individuais, sendo armazenados na classe. Se os bytes estiverem fora de ordem, espera-se até receber os elementos faltantes para colocalos na lista.

- QuicSocketRxBuffer: Os pacotes são então transmitidos de QuicStreamRxBuffer para QuicSocketRxBuffer, que armazena os pacotes até ser requisitado pela aplicação. Ao receber uma flag FIN, o buffer retornará todos os pacotes recebidos na stream.

4 Fluxo de Dados

4.1 Envio

QuicSocketBase implementa a API da classe Socket do ns-3, portanto o método Send é responsável pelo envio de pacotes através do socket. Para que o esquema de streams seja implementado sem que a classe Socket seja alterada, utiliza-se uma flag já disponível no método Send para identificar a stream para qual

deve ser enviado o respectivo pacote por meio da classe QuicL5Protocol. Caso seja o primeiro pacote, será criado o objeto QuicStreamBase, o qual, criará um QuicSubheader para enviar o frame, caso o controle de fluxo permita.

O pacote sairá da stream utilizando o objeto QuicL5Protocol, que o encaminhará para o socket (QuicSocketBase), sendo então adicionado ao buffer de transmissão (QuicSocketTxBuffer). O frame é então agregado com os demais e enviado para o socket UDP através do objeto QuicL4Protocol.

4.2 Recepção

Os pacotes são recebidos no socket UDP, o qual é responsável por invocar o método ForwardUp, o qual encaminhará o pacote para o método ReceivedData da classe QuicSocketBase. O pacote será diferenciado entre controle e outras formas, sendo então encaminhado para o método DispatchRecv do objeto QuicL5Protocol, que ciclará pelos frames do pacote acionando as funções pertinentes dos objetos QuicSocketBase e QuicStreamBase.

4.3 Retransmissão

O processo de retransmissão do QUIC ocorre a nível de socket, retransmitindo pacotes completos (formados por um ou mais frames). A classe QuicSocketBase é responsável por receber os ACKs por meio do método OnRecevedAckFrame, o qual é ativado quando um pacote contem um frame ACK. Um frame ACK possui o maior número de sequencia reconhecido e possui diversos blocos informando quais pacotes foram perdidos.

5 Compatibilidade com Algoritmos de Controle de Congestionamento TCP

A modularização dos algoritmos de controle de congestionamento TCP no ns-3 é realizada pela classe TcpCongestionOps. QuicSocketBase pode ser utilizado em modo legado, podendo utilizar estes algoritmos já produzidos para o TCP, além disso é possível produzir novos métodos de controle específicos para o QUIC e independentes do módulo.

A compatibilidade é alcançada utilizando uma instancia TcpCongestionOps como o objeto de controle de congestionamento básico do QuicSocketBase. O algoritmo padrão QuicCongestionControl é uma classe que estende TcpNewReno, adicionando especificidades dos Drafts do QUIC. A classe SetCongestionControlAlgorithm checa se o algoritmo estende a classe QuicCongestionControl, modificando a flag m_quicCongestionControlLegacy para falso se sim. Caso contrário, a flag será verdadeira e o modo legado será ativado.

O estabelecimento da conexão pode ser feito com 2, 1 ou 0 RTT a depender dos parâmetros indicados e do histórico de comunicação entre cliente e servidor.

6 Funcionalidades Faltantes

A implementação do módulo tem como base na versão 13 do draft IETF do protocolo. Contudo, não foram implementadas algumas funcionalidades presentes neste documento. A pilha do protocolo TLS não foi implementada e não são utilizadas bibliotecas de cryptografia externas. Essa diferença não afeta a avaliação da performance, pois o estabelecimento da conexão como se fosse segura pode ser simulado, mesmo que não não exista objetivamente.

Há também uma diferença na troca de parâmetros de transporte entre cliente e servidor. De acordo com o draft do protocolo, o servidor troca parâmetros com o cliente durante o handshake inicial, e o cliente é reponsável por aplicá-los. Como a QuicSocketBase é uma classe que representa tanto cliente como servidor, há apenas um conjunto de atributos para ambos, portanto, definindo os atributos para o servidor também aplica-os automaticamente ao cliente.

Apesar destas diferenças, não há implicações diretas na avaliação de performance do protocolo.

7 Referências

1 - Alvise De Biasio, Federico Chiariotti, Michele Polese, Andrea Zanella, and Michele Zorzi. 2019. A QUIC Implementation for ns-3. In Proceedings of the 2019 Workshop on ns-3 (WNS3 2019). Association for Computing Machinery, New York, NY, USA, 1–8. DOI:https://doi.org/10.1145/3321349.3321351