Análise, Projeto e Avaliação de Aplicação em Rede sobre HTTP: Protocolo QUIC

Líder: Thaís dos Santos

Daniel Oliveira de Freitas[1], **Joao Emilio Antonio Villa**[1], **Júlio Silveira Ortiz Rocha**[1], **Mateus Balda Mota**[1], **Thaís dos Santos**[1]

[1]Universidade Federal do Pampa (UNIPAMPA)
Caixa Postal 97546-550 – Alegrete – RS – Brazil

{joaovilla,julioortiz,danieloliveira,mateusmota,thaisds2}.aluno@unipampa.edu.br

1. Objetivo

O QUIC (*Quick UDP Internet Connection, ou 'Conexão UDP Rápida com a Internet'*) é um protocolo de transporte desenvolvido pelo Google. Ele reúne o que há de melhor no protocolo TCP (*Transmission Control Protocol*), e melhora o desempenho em cenários de comunicação em rede. Unido a isso, ele incorpora a criptografia de transporte TLS (*Transport Layer Security*), que cuida da parte da segurança integrada nas comunicações, e assim, o QUIC consegue combinar a eficiência e confiabilidade do TCP com a segurança do TLS, ao mesmo tempo que introduz melhorias significativas no desempenho para transferência de dados em rede [Agnė Augustėnė 2023]. Projetado para melhorar a segurança, eficiência e velocidade do tráfego HTTP - *Hypertext Transfer Protocol*, combina TCP e UDP oferecendo segurança e velocidade.

O objetivo dos autores é praticar atividades relacionadas à análise, projeto, desenvolvimento e avaliação de aplicações em rede sobre HTTP versão 3, especificamente focando na implementação e tentativa avaliação do desempenho de uma aplicação rodando com a biblioteca *QuicGo* da *GoLang*, linguagem de programação criada pela Google.

2. Metodologia

2.1. Proposta Inicial

Nossa primeira proposta era fazer o sistema na linguagem Java-Spring, utilizar a biblioteca "Netty-Incubator-codec-quic", que é baseada na biblioteca "quiche"e utilizar o IntelliJ como IDE (*Integrated Development Environment*). A ideia era abranger a compreensão do protocolo QUIC, a configuração do ambiente de desenvolvimento, a adição de dependências e bibliotecas, a configuração do servidor QUIC, a implementação dos endpoints QUIC, a lógica da aplicação, levando em conta como seria feito na linguagem Java, a execução da aplicação, as métricas de desempenho e uma explicação do uso do GitHub para controle de versão, e a medição das métricas de desempenho fariamos em Jmeter, que é uma ferramenta usada para medir métricas de desempenho do protocolo QUIC.

As métricas incluem latência, taxa de transferência, taxa de erro e tempo de resposta do servidor. . Em suma, abrangiriamos os passos essenciais para implementar o QUIC em Java, desde a compreensão do protocolo até a medição de desempenho e gerenciamento de versão.

2.2. Proposta Secundária

A segunda proposta era fazer o sistema na linguagem Python, utilizar a biblioteca ”aioquic”, que é utilizada para realizar um serviço de mensagens instantâneas utilizando o protocolo QUIC, ele é um módulo Python que implementa tanto o cliente quanto o servidor QUIC, e por fim, utilizar o VSCode como IDE (*Integrated Development Environment*). A ideia era, novamente, abranger a compreensão do protocolo QUIC, a configuração do ambiente de desenvolvimento, a adição de dependências e bibliotecas, a configuração do servidor QUIC, a lógica da aplicação, execução da aplicação levando em conta como seria feito em Python, e o controle de versão usando o GitHub. Isso não foi possível, por problemas com SSL quando tentavamos rodar o servidor, e esse foi o motivo para tentarmos na linguagem GoLang. Em suma, abrangeríamos os passos essenciais para implementar o QUIC em Python, desde a compreensão do protocolo até a medição de desempenho.

2.3. Proposta Final

2.3.1. Entendimento do Protocolo QUIC

Antes de iniciar a implementação, é essencial compreender os conceitos e funcionamentos do protocolo QUIC. É recomendado estudar a especificação do protocolo e seus recursos.

2.3.2. Configuração do Ambiente de Desenvolvimento

Certifique-se de ter um ambiente de desenvolvimento na linguagem Go (GoLang) configurado corretamente. Isso inclui a instalação dos pacotes necessários para rodar a linguagem, uma IDE (*Integrated Development Environment*) como o VSCode, e a criação de um projeto da forma adequada.

2.3.3. Dependências e Bibliotecas

A linguagem Go (GoLang), consiste em um simples cliente e servidor em puro protocolo rápido, e que utiliza da estrutura Gin Web, que fornece uma série de recursos e ferramentas que simplificam o desenvolvimento de aplicativos web em Go. Para utilizar o protocolo QUIC em uma aplicação com Go (GoLang), é necessário adicionar as dependências corretas ao projeto. Uma biblioteca utilizada é a ”QuicGo”, que fornece uma implementação do QUIC em GoLang.

2.3.4. Configuração do Servidor QUIC

Após adicionar as dependências, é necessário configurar o servidor QUIC na sua aplicação GoLang. Isso envolve a configuração do servidor QUIC e a definição de

parâmetros como portas, certificados e opções de segurança, para que tudo funcione como deve.

2.3.5. Implementar a lógica da aplicação

A lógica da aplicação acontece por meio de 2 pacotes, ”client”e ”server”, onde dentro fica a lógica responsavel pela conversação dos clientes com o servidor, além disso, contém a implementação da lógica da aplicação, manipulação dos dados recebidos e o retorno das respostas apropriadas.

2.3.6. Execução da aplicação

Executar a aplicação GoLang, testar a comunicação com o servidor QUIC e verificar se o servidor fica ativo esperando pelas mensagens. Usar ferramentas como o curl com suporte ao QUIC (por exemplo, o QuicGo - https://github.com/quic-go/quic-go) ou bibliotecas de cliente QUIC para testar a comunicação com o servidor.

2.3.7. Gerenciador de Versionamento

Para o controle de versão, usaremos o GITHUB, pois é um dos mais conhecidos e porque permite que os programadores, usuários ou qualquer membro que for consiga contribuir em projetos privados e/ou Open Source de qualquer lugar do mundo.

3. Resultados esperados

Como sabemos, o QUIC é uma junção de tudo que ha de bom no TCP (*Transmission Control Protocol*), unido a segurança da criptografia TLS (*Transport Layer Security*), porém com o muito mais velocidade para transferência de dados. Sabendo disso, analisaremos alguns pontos fortes do QUIC, como velocidade da resposta, e descreveremos na seção 3.1 como cada tópico do QUIC irá auxiliar em nossa aplicação [Agnė Augustėnė 2023].

Maior velocidade de transferência de dados: o QUIC é projetado para ser mais rápido e eficiente do que o protocolo TCP, o que significa que a transferência de mensagens instantâneas seria mais rápida e mais suave.

Maior segurança: o QUIC é projetado para ter medidas de segurança mais robustas do que o TCP, o que significa que as comunicações entre usuários seriam mais seguras e menos propensas a ataques cibernéticos.

Melhor a conexões instáveis ou de baixa qualidade: o QUIC é projetado com a capacidade de lidar com conexões de rede instáveis ou de baixa qualidade, o que significa que os usuários ainda poderiam enviar e receber mensagens instantâneas mesmo em condições de rede desafiadoras.

Maior escalabilidade: o QUIC é projetado para ser mais escalável do que o protocolo TCP, o que significa que o serviço de mensagens instantâneas desenvolvido com

o protocolo QUIC pode lidar com um grande número de usuários sem comprometer a velocidade ou a segurança.

Em suma, o serviço de mensagens instantâneas a ser desenvolvido com o protocolo QUIC pode fornecer uma experiência de usuário mais rápida, segura, confiável e escalável em comparação a outros serviços que não utilizam esse protocolo.

3.1. Desenvolvendo com QUIC

Após analisarmos as características do QUIC e seus pontos positivos, resolvemos formular os resultados que esperamos ao utilizar o QUIC na aplicação, e em um primeiro momento, concluímos que provavelmente alcançaríamos resultados de melhoras com relação ao desempenho da aplicação, eficiência e a experiência do usuário.

Levando isso em conta, abaixo estão alguns dos pontos fortes do QUIC que julgamos importantes e que farão diferença na nossa futura aplicação:

Alto Desempenho: Esperamos que nossa aplicação tenha um alto desempenho, devido ao fato do QUIC, que usa o protocolo UDP *(User Datagram Protocol)* para as conexões, ser projetado para oferecer conexões mais rápidas e com menor latência em comparação com o TCP (*Transmission Control Protocol*), ou seja, sabendo disso, esperamos que a aplicação tenha uma transferência de dados mais eficiente e um resultado de um tempo de carregamento mais rapidos para as paginas web.

Maior Confiabilidade: Em comparação com o TCP, o QUIC lida de forma mais eficaz com relação a perda de pacotes e retransmissões, ou seja, as conexões QUIC são mais resistentes a interrupções e instabilidades de rede, devido a isso, esperamos que nossa aplicação entregue uma experiência mais confiável e com qualidade para os usuários, por exemplo, contendo disponibilidade de serviço, ou seja a aplicação estará disponível a qualquer momento que os usuários precisarem, dispondo de integridade de dados e recuperação de erros, no caso de algo der errado, a aplicação deveria lidar com esses erros e usar de mecanismos de recuperação, evitando assim, a perda de dados dos usuários ou funcionalidades do sistema. Além disso, que todas essas funções estivessem disponíveis até mesmo para as redes sem fio.

Segurança: O protocolo do QUIC contem uma criptografia incorporada, que utiliza TLS (*Transport Layer Security*) para proteger as comunicações, e é ela que garante a segurança e a privacidade dos dados transmitidos, ou seja, esperamos que a aplicação tenha uma garantia da integridade dos dados, evitando corromper dados durante transferências e que sejam mantidos intactos e confiáveis a todo momento.

4. Referências

Referências

Agnė Augustėnė (2023). Tudo que você precisa saber sobre o protocolo quic. Disponível em: `https://nordvpn.com/pt-br/blog/quic-o-que-e/`. Acesso em: 14 de maio 2023.