INSTITUTO FEDERAL DE SANTA CATARINA

ARTHUR ANASTOPULOS DOS SANTOS

Análise de possibilidades de balanceamento de carga em WebRTC e SIP

São José - SC

dezembro/2023

ANÁLISE DE POSSIBILIDADES DE BALANCEAMENTO DE CARGA EM WEBRTC E SIP

Projeto de Trabalho de conclusão de curso apresentado à Coordenadoria do Curso de Engenharia de Telecomunicações do campus São José do Instituto Federal de Santa Catarina

Orientador: Prof. Ederson Torresini, Msc.

São José - SC

dezembro/2023

RESUMO

Este estudo investigará os efeitos das técnicas de balanceamento de carga em ambientes SIP e WebRTC, visando otimizar a qualidade das comunicações em tempo real. Para tal, serão definidos objetivos claros e selecionados ambientes de testes representativos. Será escolhida uma das diversas técnicas para ser implementada e avaliada em cenários realistas de implementação. Os dados coletados abrangerão métricas cruciais, como latência, perda de pacotes e utilização de recursos. A análise estatística revelará insights significativos, demonstrando a eficácia das estratégias de balanceamento de carga. Os resultados validarão a importância dessas técnicas para garantir a confiabilidade das comunicações em tempo real em ambientes dinâmicos e congestionados, contribuindo para o avanço dessa área vital da tecnologia de rede. Este estudo oferecerá uma contribuição original e valiosa para o campo, destacando a relevância do balanceamento de carga em contextos SIP e WebRTC, fundamentais em um mundo cada vez mais conectado.

Palavras-chave: Balanceamento de carga, SIP, WebRTC

ABSTRACT

This study will investigate the effects of load balancing techniques in SIP and WebRTC environments, aiming to optimize the quality of real-time communications. To this end, clear objectives will be defined and representative test environments will be selected. One of several techniques will be chosen to be implemented and evaluated in realistic implementation scenarios. The data collected will cover crucial metrics such as latency, packet loss, and resource utilization. Statistical analysis will reveal significant insights, demonstrating the effectiveness of load balancing strategies. The results will validate the importance of these techniques in ensuring the reliability of real-time communications in dynamic and congested environments, contributing to the advancement of this vital area of network technology. This study will offer an original and valuable contribution to the field, highlighting the relevance of load balancing in SIP and WebRTC contexts, fundamental in an increasingly connected world.

Keywords: Load balancing, SIP, WebRTC

LISTA DE ILUSTRAÇÕES

LISTA DE ABREVIATURAS E SIGLAS

API Interface de programação de aplicativo.

DDoS Negação de serviço distribuída.

DNS Sistema de Nomes de Domínio.

DNSSec *DNS Security*.

HTTP Protocolo de Transferência de Hipertexto.

IP Protocolo de Internet.

QOS Qualidade de Serviço.

QUIC Conexão rápida com a Internet UDP.

RFC Solicitação de comentários.

SIP *Protocolo de Iniciação de Sessão*.

SPOF Ponto único de falha.

TCP protocolo de Controle de Transmissão.

TLD Domínio de nível superior.

UDP Protocolo de datagrama do usuário.

WebRTC Comunicações na Web em tempo real.

SUMÁRIO

1 INTRODUÇÃO

A crescente demanda por comunicações em tempo real e a evolução das tecnologias de rede vêm estimulando a utilização de protocolos, como o *Protocolo de Iniciação de Sessão* (SIP) e Comunicações na Web em tempo real (WebRTC) para estabelecer comunicações de voz, vídeo e dados pela Internet. Segundo dados da CNN Brasil (Denise Ribeiro; Anthony Wells, 2023), uso de aplicativos móveis de videoconferência aumentou em 150% durante o primeiro semestre de 2021, refletindo a importância desses protocolos para diversas aplicações. Contudo, assegurar uma experiência de comunicação contínua e confiável por meios desses protocolos torna-se um desafio em cenários de redes extremamente dinâmicas e sobrecarregadas.

O SIP, que é o padrão para estabelecer, modificar e encerrar sessões de comunicação, desde chamadas de voz até videoconferências, oferece uma infraestrutura versátil para conectar indivíduos e sistemas. Já o WebRTC, uma tecnologia mais recente, estende essa capacidade diretamente para os navegadores web, permitindo a comunicação em tempo real por meio de Interfaces de programação de aplicativo (APIs) incorporadas nos próprios navegadores. Assim, tornou-se possível não apenas estabelecer chamadas e conferências por voz e vídeo, mas também compartilhar dados e colaborar em aplicações web sem a necessidade de instalação de software adicional.

No cenário atual, onde testemunhamos um crescimento exponencial e verdadeiramente gritante no número de usuários ativos na internet. De acordo com o estudo de Kemp (2022), o número de indivíduos que acessam a rede regularmente se aproximou da marca de 5 bilhões de pessoas em janeiro de 2022. Esse marco representa quase 63% da população global. Esse crescimento meteórico na conectividade tem sido impulsionado por avanços tecnológicos e pela disponibilidade generalizada da internet, permitindo que as pessoas em todo o mundo se comuniquem, colaborem e acessem informações em tempo real.

Apesar dos avanços tecnológicos, o desafio de preservar a qualidade e eficiência dessas comunicações em um cenário de rede dinâmico e frequentemente congestionado através desses protocolos não é uma tarefa trivial. A complexidade existente às comunicações em tempo real se manifesta através de uma série de desafios. Como visto em Loureiro et al. (2023), a diversidade de dispositivos, variando desde *smartphones* a computadores, introduz variações na capacidade de processamento e na qualidade de conexão. Além disso, a oscilação das condições de rede, incluindo atrasos, perda de pacotes e flutuações de largura de banda, pode prejudicar a qualidade das chamadas e comprometer a experiência ao usuário.

Nesse contexto de redes altamente dinâmicas e frequentemente sobrecarregadas, a necessidade de abordagens que garantam a qualidade e a escalabilidade das comunicações se torna ainda mais urgentes. Surge então a exploração de estratégias de balanceamento de carga. Este estudo se concentra, segundo Neghabi et al. (2018), em distribuir eficientemente a carga de trabalho entre diversos recursos, otimizando a utilização dos mesmos disponíveis e minimizando os gargalos de desempenho em qualquer um dos recursos.

O escopo deste trabalho se aplica à avaliação da qualidade de desempenho das técnicas de balanceamento de carga aplicadas a sistemas baseados nos protocolos SIP e WebRTC. Através da análise crítica dessas técnicas, buscas-se entender como diferentes estratégias podem influenciar diretamente na qualidade das comunicações e na capacidade de expansão desses sistemas em ambientes altamente dinâmicos e potencialmente congestionados.

1.1 Justificativa do tema

A comunicação em tempo real por meio de protocolos como SIP e WebRTC está se tornando cada vez mais essencial para uma variedade de aplicações, incluindo chamadas de voz, videoconferências e até compartilhamento de dados em tempo real. À medida que o mundo se torna cada vez interconectado, a demanda por essas formas de comunicação só tende a crescer, tornando-as fundamentais para a maneira como as pessoas trabalham, interagem e se comunicam.

No entanto, a qualidade dessas comunicações é frequentemente comprometida por desafios inerentes a ambientes de rede dinâmicos e congestionados, como latência, perda de pacotes e congestionamento de rede. Além disso, é crucial considerar os pontos de gargalo e Pontos únicos de falha (SPOFs), que podem surgir em infraestruturas de comunicação complexas.

A justificativa para este estudo sobre o balanceamento de carga em ambientes SIP e WebRTC se baseia na importância crítica de otimizar a experiência do usuário e garantir a confiabilidade contínua desses sistemas vitais. Ao entender profundamente os efeitos das diferentes técnicas de balanceamento de carga, podemos tomar medidas concretas para melhorar a qualidade das comunicações em tempo real, mesmo em condições adversas de rede.

Portanto, o estudo proposto visa ampliar o conhecimento da área, contribuindo para o avanço das estratégias de balanceamento de carga em ambientes SIP e WebRTC. Ao fazê-lo, busca-se garantir que essas tecnologias continuem a oferecer uma experiência de comunicação de alta qualidade, promovendo a eficácia das comunicações em tempo real em um mundo cada vez mais conectado e dependente delas.

1.2 Objetivos

Este estudo tem como objetivo geral analisar o impacto das técnicas de balanceamento de carga no desempenho de sistemas que utilizam os protocolos SIP e WebRTC. Pretende-se compreender como essas técnicas influenciam a qualidade das comunicações em tempo real e a capacidade de expansão desses sistemas em um ambiente de rede dinâmico.

1.2.1 Objetivos específicos

1. Detalhar as técnicas de balanceamento de carga mais utilizadas em sistemas SIP e WebRTC, identificando suas características e modos de implementação.

2. Avaliar o desempenho de uma técnica específica de balanceamento de carga escolhida em termos de latência, perda de pacotes e capacidade de escalabilidade.

3. Comparar os resultados obtidos com as estratégias de balanceamento de carga com base em métricas de qualidade de comunicação e eficiência operacional.

4. Investigar como as variações na carga de tráfego e na topologia da rede podem afetar a eficácia das estratégias de balanceamento de carga.

1.3 Metodologia do trabalho

O presente estudo emprega uma abordagem meticulosa e abrangente, visando obter uma melhor compreensão dos efeitos das técnicas de balanceamento de carga em ambientes SIP e WebRTC. Para atingir esse objetivo, a metodologia adotada se desdobra em várias etapas distintas, cada uma projetada para fornecer uma perspectiva única sobre o desempenho e eficácia das estratégias de balanceamento de carga.

1. Definição dos Objetivos de Pesquisa: O primeiro passo foi definir objetivos claros e específicos, incluindo a identificação de questões centrais e a formulação de hipóteses, para orientar a coleta e análise de dados, estabelecendo uma estrutura clara para o estudo.

2. Seleção de Ambientes de Teste Representativos: Para garantir a validade e relevância dos resultados, selecionamos ambientes de teste que representam um cenário real especifico de comunicações em tempo real com SIP e WebRTC, considerando a heterogeneidade de dispositivos, variações de rede e sobrecarga potencial.

3. Implementação da Técnica de Balanceamento de Carga: A seguir, selecionamos uma técnica de balanceamento de carga especifica, configurando servidores e distri-

buindo o tráfego. Avaliamos a eficácia da técnica na otimização do desempenho das comunicações em tempo real.

4. Coleta de Dados e Monitoramento do Desempenho: Durante os testes, medimos latência, perda de pacotes, uso de recursos e tempos de resposta. O monitoramento contínuo permitiu comparações entre técnicas de balanceamento de carga.

5. Análise Estatística e Comparativa: Os dados coletados passaram por análise estatística detalhada, identificando padrões e diferenças entre técnicas de balanceamento de carga.

6. Validação e Interpretação dos Resultados: Os resultados foram validados em relação às hipóteses iniciais e interpretados à luz do contexto teórico e práticas recomendadas. Implicações e direções futuras também foram discutidas.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo, abordaremos a fundamentação teórica que serve como base para a compreensão do tema central deste trabalho. Para uma compreensão mais aprofundada, faz-se necessário explorar conceitos fundamentais relacionados como: compreensão dos protocolos e tecnologias principais utilizadas, bem como a interconexão entre essas tecnologias, considerando o contexto em que são utilizadas e a necessidade de balanceamento de carga.

2.1 Balanceamento de Carga

A prática do balanceamento de carga desempenha um papel crucial no campo das redes de comunicação, incluindo aplicações WebRTC e SIP. Conforme definido por Belgaum et al. (2020), o balanceamento de carga é uma metodologia que visa alocar eficientemente a carga de rede entre diferentes componentes, resultando na otimização do desempenho da rede e na melhoria da Qualidade de Serviço (QOS) para os usuários finais. Isso é alcançado por meio da aplicação de diversas técnicas, estratégias e algoritmos de balanceamento de carga. Tais métodos auxiliam tanto os usuários finais quanto os provedores de serviços na alocação e redistribuição da carga, com o intuito de otimizar a eficiência da rede.

Uma das vantagens notáveis do balanceamento de carga, conforme destacado por Belgaum et al. (2020), reside na sua capacidade de antecipar e prever gargalos no tráfego de rede antes que ocorram, agindo como "polícia de controle de tráfego de rede". Por meio dessa estratégia, a rede pode ser gerenciada de maneira mais eficaz e assegurar que o tráfego seja distribuído de maneira equitativa e eficiente.

De acordo com Neghabi et al. (2018), o balanceamento de carga é uma técnica que divide a carga de trabalho entre vários recursos da rede, impedindo que qualquer recurso específico seja sobrecarregado. Esse equilíbrio na alocação de carga auxilia na manutenção da estabilidade da rede e na garantia de que os recursos permaneçam operacionais, contribuindo, assim para a confiabilidade e a alta disponibilidade do sistema.

Portanto, é evidente que o balanceamento de carga desempenha um papel crucial na otimização do desempenho das redes de comunicação, garantindo a eficiência a qualidade dos serviços oferecidos.

2.1.1 Importância do Balanceamento de Carga

A importância do balanceamento de carga é evidente ao considerarmos os diversos parâmetros de medição que podem ser utilizados para avaliar a eficácia das técnicas e algoritmos envolvidos, conforme ressaltado por Sajjan e Biradar (2017). Esses parâmetros desempenham um papel crucial na determinação de quão eficaz o balanceamento de carga está desempenhando seu papel na rede. Abaixo, destacamos esses parâmetros e como eles influenciam a importância do balanceamento de carga:

- Taxa de Transferência: Definida por Sajjan e Biradar (2017) como, a quantidade de trabalho que pode ser realizada em um determinado período de tempo. O balanceamento de carga é vital para garantir que a rede mantenha uma taxa de transferência ideal, permitindo que o sistema lide eficientemente com as demandas de tráfego variáveis. O aumento da taxa de transferência resulta em um desempenho aprimorado.

- Tolerância a Falhas: A tolerâncias a falhas, como mencionado por Sajjan e Biradar (2017), refere-se à capacidade do algoritmo de balanceamento de carga de permitir que o sistema funcione mesmo em condições falha. Isso garante alta disponibilidade e confiabilidade, pois o balanceamento de carga pode redirecionar o tráfego para servidores funcionais, mesmo quando ocorrem falhas em componentes da rede.

- Escalabilidade: Está é crucial para sistemas que enfrentam flutuações na demanda de tráfego. Um algoritmo de balanceamento de carga eficaz, como observado por Sajjan e Biradar (2017), é capaz de dimensionar-se conforme necessário, acomodando um aumento ou diminuição na carga de trabalho. Isso permite que a rede mantenha um desempenho consistente independentemente das mudanças nas condições.

- Desempenho: O desempenho geral da rede, considerando precisão, custo e velocidade, é diretamente impactado pelo balanceamento de carga. Um algoritmo eficaz equilibra a carga de maneira que resulte em um desempenho otimizado, garantindo que os recursos da rede sejam utilizados de forma eficiente e econômica.

- Utilização de Recursos: O controle de utilização de recursos é fundamental para otimizar a eficiência da rede. Conforme apontado por Sajjan e Biradar (2017), o balanceamento de carga ajuda a distribuir a carga de trabalho entre os recursos disponíveis de maneira uniforme, impedindo que qualquer recurso específico seja sobrecarregado. Isso garante a utilização eficaz dos recursos e evita desperdícios.

Dessa forma, a importância do balanceamento de carga na rede é ressaltada ao considerar sua capacidade de otimizar a taxa de transferência, reduzir o tempo de resposta, garantir tolerância a falhas escalar conforme necessário, melhorar o desempenho

e controlar a utilização de recursos. Esse parâmetros de medição demonstram que o balanceamento de carga desempenha um papel crítico na manutenção da eficiência e da qualidade da rede, proporcionando uma experiência superior aos usuários.

2.1.2 Estratégias de Balanceamento de Carga

As estratégias de balanceamento de carga constituem uma etapa crucial para otimizar o desempenho das redes. Elas são responsáveis pela alocação eficiente de tarefas e recursos, garantindo um uso equitativo e eficaz dos componentes de rede disponíveis.

De acordo com Sajjan e Biradar (2017), os algoritmos de balanceamento de carga podem ser divididos em duas categorias distintas, levando em consideração o estado atual do sistema. Esta classificação é essencial para determinar qual algoritmo é mais apropriado para determinado contexto.

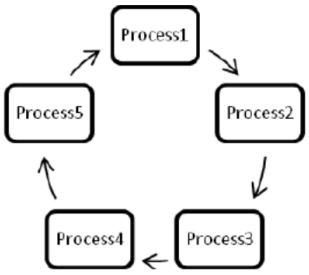
2.1.2.1 Algoritmos Estáticos

Os algoritmos estáticos demonstram eficácia em ambientes homogêneos e estáveis. No entanto, é importante destacar que eles podem apresentar limitações na adaptação a mudanças dinâmicas nos atributos do sistema. Conforme observado por Sajjan e Biradar (2017), algoritmos estáticos não avaliam o histórico das tarefas ao atribuí-las a nós da rede.

Para contextualizar, é relevante mencionar que em protocolos de comunicação como o protocolo de Controle de Transmissão (TCP) e o Protocolo de datagrama do usuário (UDP), o balanceamento de carga estático pode ter implicações no tráfego de dados. Enquanto o TCP oferece um controle mais rigoroso de fluxo e uma transmissão confiável, o UDP, por sua vez, é mais leve e direto, sendo especialmente útil em aplicações que requerem baixa latência, mas aceitam alguma perda de pacotes.

- Algoritmo Round Robin: O algoritmo Round Robin, conforme apresentado por Sajjan e Biradar (2017), adota uma abordagem onde um tempo quântico fixo é alocado para cada tarefa. Essa técnica distribui as tarefas de forma circular entre os nós da rede, garantindo que todos os processadores sejam atribuídos em uma ordem cíclica. Dessa forma, evita-se o problema de inanição, onde nenhum nó permanece inativo por longos períodos de tempo, proporcionando uma resposta mais rápida em cenários de distribuição equitativa de carga de trabalho entre os processos. Contudo, é importante ressaltar que esse método pode resultar em alguns nós sobrecarregados, enquanto outros podem permanecer ociosos e subutilizados.

Figura 1 – Algoritmo Round Robin



Fonte: Sajjan e Biradar (2017)

Além disso, de acordo com Jiang et al. (2012), o algoritmo Round Robin se destaca por garantir uma distribuição mais equitativa de chamadas para servidores em um cluster, em comparação com algoritmos baseados em hash. Para exemplificar, se a chamada anterior foi atribuída ao servidor M, a próxima chamada será direcionada ao servidor $(M + 1) \mathrm{mod} N$, onde N representa o número total de servidores no cluster. Essa característica é crucial para manter um equilíbrio efetivo entre os servidores, otimizando assim o desempenho do sistema de comunicação.

- Algoritmo Min-Min: O Algoritmo Min-Min de Balanceamento de Carga, conforme elucidado por Sajjan e Biradar (2017), opera por meio da manutenção de uma lista de tarefas e do cálculo do tempo mínimo de conclusão para todos os nós disponíveis. A abordagem adotada por esse algoritmo consiste em atribuir a tarefa ao nó que apresenta o menor tempo de conclusão, justificando assim o nome "min-min". A cada atribuição de tarefa, a lista e o tempo de execução da máquina são atualizados, proporcionando resultados satisfatórios, especialmente quando se lida com tarefas de menor magnitude. Esse método demonstra eficácia na otimização da alocação de recursos e, quando aplicado a sistemas de comunicação, pode influenciar diretamente na taxa de transferência e no tempo de resposta das solicitações.

- Algoritmo Min-Max: O Algoritmo Min-Max de Balanceamento de Carga, conforme foi explicado por Sajjan e Biradar (2017), opera mantendo uma lista de tarefas e calculando o tempo mínimo de conclusão para todos os nós disponíveis. No entanto, ao contrário do Algoritmo Min-Min, esse método atribui a tarefa ao nó que apresenta o tempo máximo de conclusão, justificando o nome "min-max". A cada alocação de tarefa, a lista e o tempo de execução da máquina são atualizados. Essa abordagem é particularmente útil quando se deseja otimizar a utilização de recursos em sistemas de comunicação, uma vez que pode influenciar a taxa de transferência e o desempenho geral do sistema, considerando fatores como carga de trabalho e disponibilidade dos nós.

2.1.2.2 Algoritmos Dinâmicos

Os algoritmos dinâmicos apresentam vantagens significativas em ambientes heterogêneos e dinâmicos. Eles são mais flexíveis e capazes de se adaptar a mudanças nos atributos do sistema. Contudo, é importante mencionar que sua implementação pode ser mais complexa.

Neste contexto, ao considerar protocolos de comunicação como UDP e o TCP, os algoritmos dinâmicos pode ser particularmente relevantes para lidar com variações no tráfego de dados. O UDP, por exemplo, pode ser utilizado em conjunto com algoritmos dinâmicos para garantir que os recursos de rede sejam alocados de forma eficiente, mesmo diante de mudança na demanda.

Esses algoritmos pode ser implementados de duas formas distintas:

- Sistema Distribuído: Um sistema distribuído refere-se a uma infraestrutura de computação na qual vários dispositivos ou componentes de *hardware* interagem e colaboram entre si para realizar tarefas e fornecer serviços de forma conjunta. Esses sistemas têm a capacidade de processar uma maior escalabilidade, confiabilidade e eficiência. No contexto dos sistemas distribuídos, a alocação equitativa de tarefas e recursos entre os diferentes nós ou componentes é fundamental para otimizar o desempenho e garantir a utilização eficaz dos recursos disponíveis.

  Compreender as diferentes abordagens de implementação dos algoritmos de balanceamento de carga é crucial para alcançar esses objetivos. Portanto, é possível distinguir essas abordagens com base no contexto em que são aplicadas, seguindo as orientações de Sajjan e Biradar (2017). Em um sistema distribuído, existem duas modalidades de operação:

  - Sistema Distribuído Cooperativos: No contexto dos sistemas distribuídos cooperativos, a colaboração entre todos os nós é a essência do balanceamento de carga. Os nós interagem de maneira colaborativa para garantir que as tarefas sejam alocadas de forma equilibrada e eficiente em toda a rede. Esta cooperação entre os nós pode resultar em uma alocação de recursos mais justa e eficaz.
  - Sistemas Distribuídos Não Cooperativos: Em sistemas distribuídos não cooperativos, cada nó age de forma independente no que diz respeito ao balanceamento de carga. Cada nó toma decisões independentes sobre como alocar tarefas e recursos, sem coordenação centralizada. Essa autonomia individual pode ter implicações no equilíbrio da carga, uma vez que a colaboração não é uma prioridade.

- Sistema não Distribuído: Em sistemas centralizados, um nó central é responsável por coordenar o balanceamento de carga de todo o sistema. Os demais nós interagem com esse nó central para receber tarefas e recursos. No entanto, em caso de falha do nó central, a funcionalidade de balanceamento de carga é interrompida, o que pode dificultar a recuperação do sistema.

  Já nos sistemas semi-distribuídos, os nós são agrupados em conjuntos, formando *clusters*. Um nó central em cada *cluster* é responsável por equilibrar a carga internamente nesse conjunto. Vários nós centrais podem gerenciar o balanceamento de carga de forma independente em diferentes *clusters*, o que permite uma distribuição mais precisa da carga. A falha de um nó central afetará apenas o *cluster* específico, mantendo a funcionalidade nos demais.

2.2 TCP (Transmission Control Protocol)

O protocolo de Controle de Transmissão (TCP), conforme definido na Solicitação de comentários (RFC) 9293 (EDDY, 2022), é projetado para ser utilizado como um protocolo altamente confiável de *host* para *host* em redes de comunicação de computadores comutados por pacotes e em sistemas interconectados por meio dessas redes. Essa definição estabelece o TCP como uma peça fundamental na arquitetura de comunicação de sistemas distribuídos, oferecendo confiabilidade em ambientes de redes heterogêneas.

Ao considerar a implementação de estratégias de Balanceamento de Carga em ambientes que envolvem protocolos como SIP e WebRTC, a confiabilidade do TCP torna-se essencial para assegurar uma comunicação estável e eficiente.

No contexto de SIP, que é amplamente utilizado para estabelecer e encerrar sessões de comunicação em tempo real, o TCP pode ser empregado para garantir a entrega precisa e ordenada de mensagens, contribuindo para a estabilidade das chamadas e a continuidade das sessões.

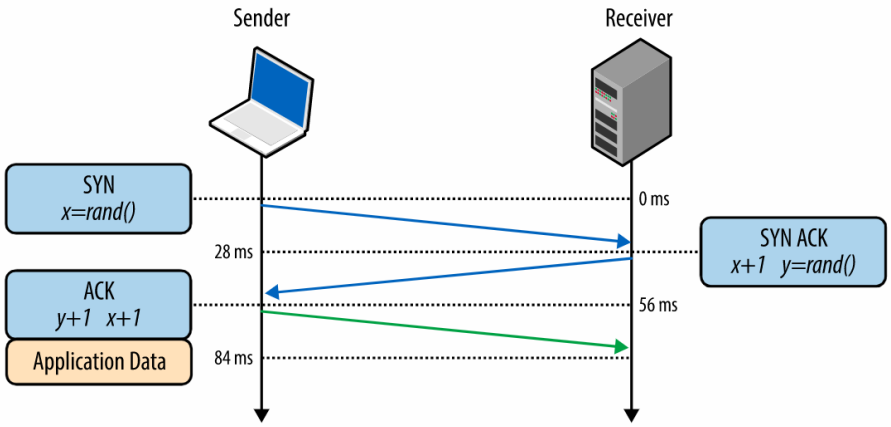
No caso do WebRTC, que permite a comunicação em tempo real diretamente nos navegadores web, o TCP desempenha um papel vital na transmissão de dados de áudio e vídeo de forma confiável, fundamental para proporcionar uma experiência de usuário consistente.

A relação entre o TCP e o Balanceamento de Carga reside na necessidade de distribuir eficientemente as cargas de trabalho entre os diferentes nós do sistema, garantindo que o tráfego TCP seja gerenciado de maneira equitativa. Estratégias de Balanceamento de Carga podem ser aplicadas para otimizar a utilização dos recursos disponíveis, melhorando o desempenho global do sistema e assegurando uma comunicação estável mesmo em cenários de alta demanda.

O TCP opera como um protocolo de transporte, fornecendo uma comunicação confiável e orientada à conexão entre *hosts*. Sua confiabilidade é fundamental para garantir que os dados sejam entregues corretamente e em ordem ao destino, mesmo em condições adversas de rede. Essa características é crucial para aplicações que exigem uma entrega precisa de informações, como transferência de arquivos, *streaming* de mídia e transações online.

A natureza orientada à conexão do glsxtrshorttcp implica a estabelecimento de uma conexão antes da transferência de dados. Esse processo, como definido por Grigorik (2013) inclui um aperto de mão triplo entre os hosts envolvidos, como podemos ver na Figura 2, antes que o cliente ou o servidor possam trocar quaisquer dados da aplicação, eles devem concordar em inciar os números de sequência dos pacotes, bem como uma série de outras variáveis específicas da conexão, de ambos os lados. Estes números de sequência são escolhidos aleatoriamente em ambos os lados por razões de segurança.

Figura 2 – Aperto de mão Triplo



Fonte: Grigorik (2013)

Após o estabelecimento da conexão, o TCP monitora ativamente a entrega de pacotes, retransmitindo-os se necessário e garantindo a ordenação correta na chegada ao destino.

Além disso, o TCP integra mecanismos de controle de congestionamento para gerenciar eficientemente o tráfego na rede. Esses mecanismos visam evitar sobrecargas e garantir uma operação suave, adaptando a taxa de transmissão conforme as condições da rede.

A importância do TCP na comunicação de sistemas distribuídos está diretamente relacionada à sua capacidade de proporcionar confiabilidade, integridade e ordenação na entrega de dados. Essas características fazem do TCP, uma escolha fundamental para uma variedade de aplicações críticas, contribuindo para a estabilidade e eficácia das operações em ambientes distribuídos.

Apesar das vantagens do TCP em garantir a integridade e ordenação dos dados, sua abordagem orientada à conexão pode introduzir atrasos significativos no estabelecimento de conexão. Em cenários como SIP e WebRTC, onde a latência é crítica para comunicação em tempo real, esses atrasos podem afetar adversamente a experiência do usuário. O TCP, por sua natureza mais pesada em termos de *overhead*, pode ser menos eficiente do que o UDP em situações onde a velocidade de transmissão é prioritária sobre garantias rigorosas de entrega. A escolha entre TCP e UDP deve levar em consideração os requisitos específicos da aplicação e a priorização entre confiabilidade e latência desejadas.

2.3 UDP (User Datagram Protocol)

O Protocolo de datagrama do usuário (UDP), conforme definido na RFC 768 (POSTEL, 1980), desempenha um papel fundamental na comunicação eficiente em ambientes de rede de computadores comutadas por pacotes. Projetado para operar em conjunto com o Protocolo de Internet (IP), o UDP oferece um modo de comunicação de datagrama, caracterizado por sua abordagem não orientada para conexão e pela ausência de garantias estritas de entrega e ordenação de mensagens.

Ao contrário do TCP, o UDP é uma escolha apropriada para cenários em que a latência e a eficiência são críticas, enquanto a perda ocasional de dados não compromete a aplicação. Esta características torna o UDP particularmente adequado para aplicações SIP e WebRTC, onde a comunicação em tempo real, como chamadas de voz e vídeo, exige respostas rápidas e uma experiência de usuário fluída.

No contexto do Balanceamento de Carga, a escolha do UDP pode ser estratégica, uma vez que seu design leve resulta em menor sobrecarga, e consequentemente, menor latência quando comparado ao TCP. Essa eficiência é especialmente valiosa em situações onde a prioridade é dada à transmissão rápida de dados, garantindo uma interação mais imediata e satisfatória para o usuário.

É importante ressaltar que, embora o UDP ofereça vantagens em termos de eficiência e baixa latência, não proporciona garantidas de entrega, ordenação ou proteção contra duplicação de mensagens. Portanto, ao implementar o UDP em aplicações que exigem comunicação em tempo real, é crucial considerar as necessidades específicas da aplicação, ponderando o equilíbrio entre latência e confiabilidade desejados. O UDP destaca-se como uma alternativa eficaz quando a agilidade na transmissão de dados é prioritária sobre a necessidade estrita de garantias de entrega.

2.4 DNS (Domain Name System)

O Sistema de Nomes de Domínio (DNS), regulamentado pelas RFCs 1034 (MOCKAPETRIS, 1987a) e 1035 (MOCKAPETRIS, 1987b) , desempenha um papel fundamental na infraestrutura da internet, possibilitando a tradução de nomes de domínio legíveis por humanos em endereços IP associados aos servidores correspondentes. Essa tradução é essencial para comunicação entre dispositivos na internet, uma vez que os computadores utilizam endereços IP para identificar e localizar uns aos outros.

O DNS opera em um modelo hieráquico, onde servidores são organizados em uma estrutura de árvore, com a raiz no topo, seguida pelos Domínios de nível superior (TLDs), subdomínios e assim por diante. Esse design permite uma distribuição eficiente das responsabilidades entre os servidores, otimizando a resolução de nomes de domínio.

A estrutura distribuída do DNS permite a adição de novos servidores e domínios sem prejudicar a operação global. Além disso, a redundância é assegurada pela presença de servidores secundários, que podem ser consultados caso um servidor primário esteja inacessível.

Entretanto, o DNS também apresenta desafios. A possibilidade de ataques, como envenenamento de cache e ataques de Negação de serviço distribuída (DDoS), representa preocupações de segurança. Além disso, a dependência do DNS para resolução de nomes pode torná-lo um ponto único de falha em determinados cenários.

Seu papel no balanceamento de carga, está em distribuir solicitações entre vários servidores de um mesmo serviço. Essa abordagem, conhecida como balanceamento de carga baseado em DNS, permite otimizar a distribuição de tráfego e melhorar a disponibilidade do serviço.

2.4.1 DNSSec (DNS Security)

O *DNS Security* (DNSSec) é uma extensão do DNS projetada para reforçar a segurança das informações de resolução de nomes de domínio. Implementado conforme as especificações delineadas na RFC 4033 (ROSE et al., 2005), o DNSSec adiciona uma camada adicional de proteção ao DNS, mitigando possíveis vulnerabilidades e ataques que visam comprometer a integridade e autenticidade das informações de resolução de nomes.

O principal objetivo do DNSSec é garantir a autenticidade dos dados de resposta do DNS, certificando-se de que as informações fornecidas pelo servidor DNS são legítimas e não foram alteradas durante a transmissão. Para alcançar isso, o DNSSec incorpora mecanismos de assinatura digital, possibilitando a verificação da autenticidade dos registros DNS.

Este tendo uma vantagem significativa em seu uso, a proteção contra ataques de

envenenamento de cache, onde um invasor tenta fornecer informações DNS falsas para um servidor DNS, induzindo-o a armazenar dados incorretos. Ao verificar a autenticidade dos registros DNS por meio de assinaturas digitais, o DNSSec reduz a eficácia desses ataques.

Contudo, a implementação do DNSSec também apresenta desafios. O aumento do tamanho das respostas DNS devido à inclusão de assinaturas digitais pode impactar o desempenho da resolução de nomes. Além disso, a complexidade operacional e a necessidade de manter as chaves de assinatura podem tornar a administração do DNSSec mais desafiadora.

o DNSSec não desempenha um papel direto no contexto do balanceamento de carga, pois sua função principal está relacionada à segurança e integridade dos dados de resolução de nomes. Mas, a segurança aprimorada proporcionada pelo DNSSec contribui para um ambiente DNS mais confiável, o que, por sua vez, é benéfico para qualquer aplicação que dependa de balanceamento de carga baseado em DNS.

2.5 HTTP/1.1 (Hypertext Transfer Protocol)

O Protocolo de Transferência de Hipertexto (HTTP) é protocolo de comunicação de dados que atua como a base da *World Wide Web*, permitindo a transferência de informações entre clientes e servidores de maneira sem estado. O HTTP/1.1, conforme definido na RFC 9112 (FIELDING; NOTTINGHAM; RESCHKE, 2022), é protocolo de solicitação/resposta de aplicação, caracterizado por semântica extensível e cargas de mensagem autoexplicativas para interações flexíveis com sistemas de informação baseados em hipertexto na rede.

Como dito anteriormente, este protocolo opera de maneira sem estado, o que significa que cada solicitação do cliente é tratada independentemente, sem retenção de informações sobre solicitações anteriores. A especificação do HTTP/1.1 está dividida em várias partes, cobrindo desde a sintaxe das mensagens até aspectos como semântica, autenticação, condicionais, cache e intervalo de solicitações.

O HTTP/1.1 proporciona uma comunicação eficaz entre clientes e servidores, sendo a base para a navegação e interação na Web. No entanto, a padronização inicial foi desafiada pela proliferação de diferentes implementações, o que resultou em uma falta de uniformidade.

Ao considerar o Balanceamento de Carga em aplicações SIP e WebRTC, o HTTP/1.1 pode desempenhar um papel crucial na distribuição eficiente de solicitações entre servidores. O HTTP sem estado torna possível distribuir cargas de trabalho sem depender do estado de conexões anteriores, favorecendo estratégias de balanceamento de carga dinâmicas e adaptáveis.

No entanto, vale ressaltar que o HTTP, por sua natureza sem estado, não é otimizado para ambientes que exigem conexões persistentes, como em algumas aplicações WebRTC. Entretanto, é importante destacar que o HTTP/1.1 introduziu melhorias nesse aspecto com a implementação da conexão persistente, permitindo várias requisições numa mesma conexão TCP, embora com respostas em sequência. Dessa forma, a versatilidade do protocolo HTTP/1.1 possibilita sua implementação sobre os protocolos UDP ou TCP, sendo essencial avaliar as características específicas de cada cenário para determinar a melhor escolha.

2.5.1 HTTP/2

A segunda versão do Protocolo de Transferência de Hipertexto (HTTP) representa uma evolução significativa em relação ao seu antecessor, o HTTP/1.1, visando otimizar a eficiência da comunicação entre clientes e servidores. O HTTP/1.1, embora bem-sucedido, apresentava algumas características que impactavam negativamente o desempenho das aplicações, especialmente no que diz respeito à utilização da conexão subjacente.

O HTTP/2, conforme definido na RFC 7540 (BELSHE; PEON; THOMSON, 2015), aborda algumas limitações do HTTP/1.1. Por exemplo, o HTTP/1.1 permitia apenas uma solicitação pendente por vez em uma conexão TCP, o que levava a estratégias como a utilização de várias conexões para atingir concorrência e reduzir a latência. Além disso, os campos de cabeçalho HTTP eram frequentemente repetitivos e extenso, resultando em tráfego de rede desnecessário e preenchimento rápido da janela de congestão do TCP, causando latência excessiva.

Nesta iteração do protocolo HTTP, é introduzido um mapeamento otimizado das semânticas específicas do HTTP para a conexão subjacente. Isso viabiliza a *interleaving* de mensagens de solicitação e resposta na mesma conexão, proporcionando uma abordagem eficiente na codificação dos campos de cabeçalho HTTP. Essa eficiência permite a priorização de solicitações, resultando em aprimoramento de desempenho. Além disso, é importante ressaltar que as requisições e respostas não têm relação entre si, o que contribui para a redução do atraso por espera das respostas, uma questão presente no HTTP/1.1. Adicionalmente, a capacidade de multiplexação possibilita que várias solicitações HTTP ocorram simultaneamente em uma única conexão, mitigando, em parte, o desafio de "bloqueio na linha de frente"presente no HTTP/1.1.

Ao considerar o Balanceamento de Carga em aplicações SIP e WebRTC, a eficiência da multiplexação do HTTP/2 pode ser estratégica. A capacidade de realizar várias solicitações simultaneamente em uma única conexão pode simplificar o gerenciamento de conexões e melhorar a distribuição em uma única conexão pode simplificar o gerenciamento de conexões e melhorar a distribuição de carga entre servidores.

Como destacado no http.dev (2022), o HTTP/2 difere de seu antecessor de várias manerias, sendo um protocolo binário, introduzindo compressão de campos e cabeçalho HTTP e suportando trocas de dados simultâneas em uma única conexão HTTP. A multiplexação possibilita que várias solicitações HTTP ocorram ao mesmo tempo em uma única conexão, melhorando a utilização dos recursos de rede.

No entanto, é importante observar que, embora o HTTP/2 traga melhorias substanciais, como compressão de cabeçalhos e suporte a fluxos de dados multiplexados, não existe uma abordagem única para todos os cenários. A escolha entre HTTP/1.1 e HTTP/2 depende das características específicas da aplicação e das necessidades de desempenho.

2.5.2 HTTP/3

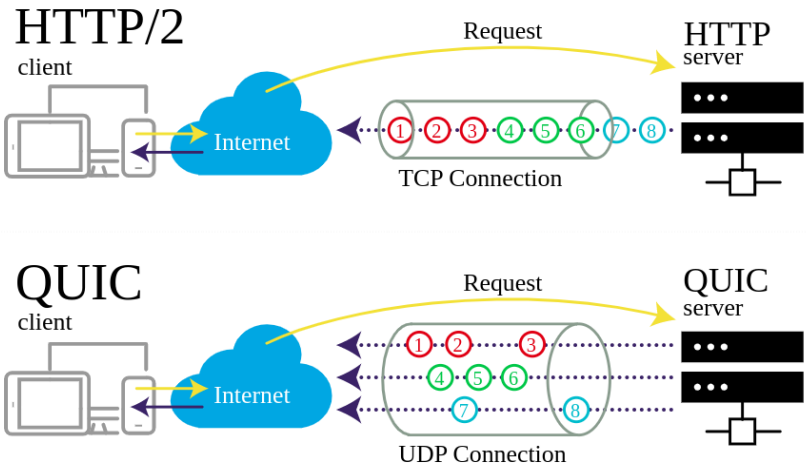
A terceira versão do Protocolo de Transferência de Hipertexto (HTTP) representa uma evolução no cenário das comunicações web, estendendo as capacidades dos seus predecessores, HTTP/1.1 e HTTP/2. Introduzido pela RFC 9114 (BISHOP, 2022), o HTTP/3 mantém as semânticas do HTTP sobre um novo protocolo de transporte chamado Conexão rápida com a Internet UDP (QUIC).

O HTTP/1.1 e o HTTP/2, apresentavam limitações que o HTTP/3 visa superar. O HTTP/1.1 utilizava campos de texto delimitados por espaços para transmitir mensagens HTTP, resultando em complexidade de análise e comportamento variante. Além disso, a falta de uma camada de multiplexação no HTTP/1.1 levava à utilização de múltiplas conexões TCP para atender solicitações em paralelo, impactando negativamente no controle de congestionamento e eficiência de rede.

O HTTP/2 introduziu uma camada binária de formatação e multiplexação para melhorar a latência sem modificar a camada de transporte. No entanto, a natureza paralela da multiplexação do HTTP/2, invisível aos mecanismos de recuperação de perda do TCP, causava paralisação em todas as transações ativas em caso de perda ou reordenação de pacotes. No HTTP/1.1, identificou-se que o problema residia na camada de aplicação. Com a evolução para o HTTP/2.0, observou-se uma melhoria parcial, pois algumas funcionalidades ainda dependiam da camada de transporte, herdando assim seus vícios. Isso evidencia que, embora o HTTP/2 tenha solucionado parte do problema inicial, persistiram desafios relacionados às dependências com a camada de transporte, ressaltando a complexidade inerente à otimização do protocolo.

Já o HTTP/3 resolve esses desafios ao adotar o protocolo QUIC como seu meio de transporte subjacente. QUIC, este desenvolvido pela Google, é um algoritmos de comunicação multiplexado construído sobre o protocolo UDP. Diferentemente do TCP, que exige uma viagem de ida e volta completa para iniciar uma conexão HTTP, QUIC minimiza a latência, resultando em tempos de transmissão mais rápidos.

Figura 3 – Diferença no transporte do HTTP/2 e QUIC



Fonte: http.dev (2022)

2.6 WebSockets

O Protocolo *WebSocket*, definido pela RFC 6455 (MELNIKOV; FETTE, 2011), proporciona uma forma eficiente de comunicação bidirecional entre um cliente e servidor. Sua principal característica é permitir a troca de dados de uma forma contínua e orientada a mensagens, superando as limitações de abordagens tradicionais que dependem da abertura de múltiplas conexões HTTP.

O *WebSocket* opera sobre o TCP e inicia com um aperto de mão de abertura, seguido por um formato básico de mensagem. Essa tecnologia visa atender às necessidades de aplicações baseadas em navegador que requerem comunicação bidirecional com servidores, sem depender da abertura de várias conexões HTTP. O modelo de segurança adotado é baseado na origem, comumente utilizado por navegadores web.

Como explicado por Grigorik (2013), o *WebSocket* permite a transmissão bidirecional de dados de texto e binários entre cliente e servidor. Ele é considerado a API mais próxima de um *socket* de rede bruto no navegador, oferecendo uma abstração de complexidade por meio de uma API simples e serviços adicionais, como negociação de conexão, aplicação da política de mesma origem, interoperabilidade com a infraestrutura HTTP existente, comunicação orientada a mensagens com eficiente estruturação de mensagens, negociação de sub-protocolo e extensibilidade.

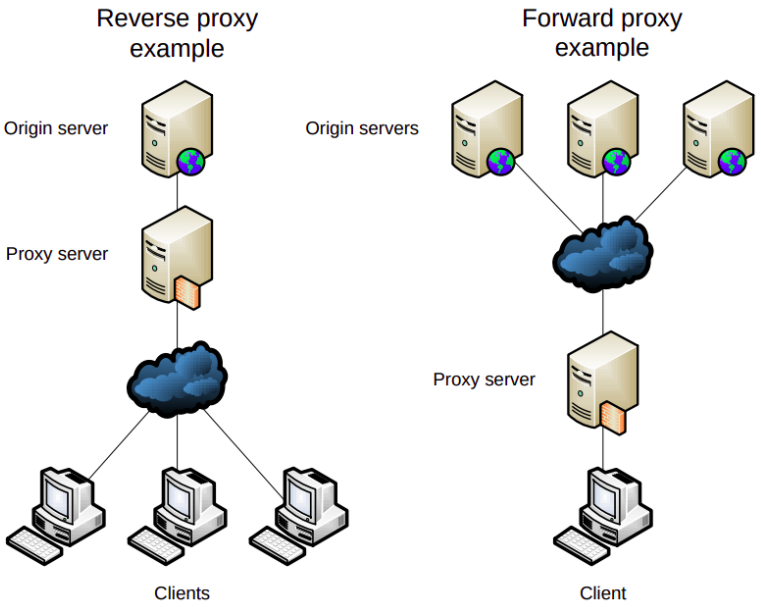
Embora este seja uma opção versátil e flexível no navegador, é essencial reconhecer que não substitui o HTTP. Cada Transporte possui suas próprias forças e melhores casos de uso. Aplicações que utilizam protocolos personalizados devem gerenciar aspectos como controle de estados compressão e cache, normalmente providos pelos navegadores.

2.7 Proxy Reverso

O *Proxy* Reverso desempenha um papel crucial na arquitetura de redes, atuando como intermediário entre clientes e servidores de destino. Conforme explicado por Dély (2014), um servidor *proxy* pode ser tanto um serviço em si quanto um *host* físico que fornece esse serviço. Existem dois principais de servidores *proxy*: o *proxy* de encaminhamento e o *proxy* reverso.

O *proxy* de encaminhamento intervém no tráfego entre o cliente e o destino escolhido pelo cliente. Em contraste, o *proxy* reverso adota uma abordagem diferente, apresentando ser o servidor de origem para o s clientes conectados. Ele encaminha as solicitações dos clientes para o servidor de origem e, em seguida, repassa o conteúdo ou informações do servidor de volta aos clientes.

Figura 4 – Exemplos mostrando as diferenças entre um *proxy* reverso (à esquerda) que aparecerão ser a origem do conteúdo e um *proxy* de encaminhamento (à direita)



Fonte: Dély (2014)

No contexto do balanceamento de carga, o *proxy* Reverso desempenha um papel significativo. Ao contrário do *proxy* de encaminhamento, que age como um intermediário direto entre o cliente e o servidor de destino, o *proxy* reverso tem a capacidade de distribuir a carga de maneira eficiente entre vários servidores de origem. Ele gerencia as solicitações dos clientes encaminhando-as para diferentes servidores, otimizando assim o desempenho e garantindo a disponibilidade do serviço.

As vantagens do *proxy* reverso incluem a melhoria da segurança, pois oculta a estrutura interna dos servidores de origem, e a capacidade de realizar o cache de conteúdo reduzindo assim a carga nos servidores e melhorando a velocidade de resposta. No entanto,

é importante considerar que o uso inadequado do *proxy* reverso pode resultar em desafios de configuração e complexidade adicional na gestão da infraestrutura.

2.8 Cache Web

O uso de cache web é uma estratégia eficaz para aprimorar o desempenho na web, conforme destacado por Nottingham (2013). O cache web atua como intermediário entre um servidor de origem e um ou mais clientes, armazenando temporariamente as respostas enviadas pelo servidor. Esse armazenamento temporário permite que as respostas sejam servidas diretamente quando solicitadas novamente, sem a necessidade de envolver o servidor de origem.

Nottingham (2013) menciona que os navegadores frequentemente realizam o cache de conteúdo baixado durante a navegação de usuários na web. Essa prática evita o download repetido do mesmo material, uma vez que o navegador salva os arquivos em cache. Além disso, o cache web pode ser implementado por meio de *proxies* reversos, os quais são capazes de armazenar em cache as respostas dos servidores de origem. Essa abordagem melhora o tempo de resposta do site e a quantidade de solicitações que podem ser tratadas simultaneamente (NOTTINGHAM, 2013).

A utilização eficiente do cache web desempenha um papel crucial no balanceamento de carga. Ao armazenar temporariamente as respostas frequentemente solicitadas, o cache reduz a carga nos servidores de origem, otimizando assim o tempo de resposta e a capacidade de lidar com um maior volume de solicitações simultâneas. Essa prática contribui para a eficiência do sistema, melhorando a experiencia do usuários e garantido a disponibilidade de conteúdo.

Apesar das vantagens proporcionadas pelo cache web, é importante considerar que ele pode resultar em desafios, como a necessidade de gerenciar o tempo de vida útil do cache e lidar com questões relacionadas à consistência dos dados armazenados. No entanto, quando implementado corretamente, o cache web é uma ferramenta valiosa para otimizar o desempenho e a eficiência operacional em ambientes online.

2.9 WebRTC (Web Real-Time Communication)

O Comunicações na Web em tempo real (WebRTC) , como destacado por Grigorik (2013), representa uma evolução significativa na comunicação em tempo real na web, oferecendo um conjunto de padrões, protocolos e APIs *JavaScript* que possibilitam o compartilhamento ponto a ponto de áudio, vídeo e dados entre navegadores.

Para criar experiências ricas em m aplicativos em tempo real, como teleconferências de áudio e vídeo e troca de dados ponto a ponto, o WebRTC introduz novos recursos

nos navegadores. Essas funcionalidades, como explicado por Grigorik (2013), incluem processamento de áudio e vídeo, APIs de aplicativos inovadoras e suporte para novos protocolos de rede. Sendo as três APIs principais:

- *MediaStream*: para aquisição de streams de áudio e vídeo;
- *RTCPeerConnection*: para comunicação de dados de áudio e vídeo;
- *RTCDataChannel*: para comunicação de dados de aplicativos.

Estas APIs encapsulam a complexidade subjacente e possibilitam a implementação desses recursos com apenas algumas linhas de código *JavaScript* (GRIGORIK, 2013).

É importante destacar que, embora o WebRTC ofereça uma solução eficiente para comunicação ponto a ponto, ele não inclui um mecanismo de sinalização padrão. A falta de um protocolo formal de sinalização pode ser considerada uma limitação, pois é necessário um canal externo para coordenar a comunicação entre os pares. Esta ausência destaca a importância de soluções externas, como o SIP, para realizar a sinalização necessária à configuração e estabelecimento de chamadas em tempo real.

No entanto, a implementação efetiva do WebRTC vai além dessas APIs. As etapas de sinalização descoberta de pares, negociação de conexão e segurança desempenham papéis crucias. A arquitetura e os protocolos subjacentes ao WebRTC também influenciam diretamente seu desempenho, incluindo aspectos como latência na configuração da conexão, sobrecarga de protocolo e semãntica de entrega. Notavelmente, o WebRTC utilza o UDP para transportar dados, destando a enfase na minimilização da latência (GRIGORIK, 2013).

Quando considerarmos o papel do WebRTC no papel do balanceamento de carga, sua capacidade de permitir comunicação ponto a ponto direta entre navegadores pode impactar positivamente a distribuição eficiente de cargas de trabalho. A flexibilidade e a simplicidade proporcionadas pelo WebRTC podem ser exploradas para otimizar a alocação de recursos e melhorar a experiência do usuário em cenários de carga variável.

2.10 SIP (Session Initiation Protocol)

O *Protocolo de Iniciação de Sessão* (SIP), conforme definido pelo RFC 3261 (SCHOOLER et al., 2002), desempenha um papel crucial na facilitação de aplicações na internet que demandam a criação e gestão de sessões entre participantes. Uma sessão, nesse contexto, refere-se a uma troca de dados entre uma associação de participantes, podendo envolver diversas formas de dados em tempo real, como voz, vídeo ou mensagens de texto.

Este atua em conjunto com vários protocolos que transportam dados de sessão multimídia em tempo real, permitindo que pontos finais da internet, conhecidos como agentes de usuários, descubram uns aos outros e concordem com a caracterização de uma sessão que desejam compartilhar. Isso é particularmente útil em cenários onde os usuários podem se movimentar entre diferentes pontos finais, ter múltiplos endereços e se comunicar em diversos meios simultaneamente.

Sua sinalização, como visto pela RFC 3261 (SCHOOLER et al., 2002), envolve a troca de mensagens entre agentes de usuário e servidores SIP, é essencial para coordenar e negociar parâmetros de sessões. Essa capacidade de sinalização contribui para a integração eficaz do SIP em cenários diversos, incluindo seu papel no contexto do WebRTC, onde desempenha uma função fundamental na descoberta de participantes e na negociação de parâmetros para sessões compartilhadas.

A flexibilidade do SIP é evidente na sua capacidade de criar, modificar e encerrar sessões de forma independente dos protocolos de transporte subjacentes e sem depender do tipo específico de sessão que está sendo estabelecida. Para localizar participantes em potencial de uma sessão e realizar outras funções, o SIP possibilita a criação de uma infraestrutura de *hosts* de rede, chamados de servidores *proxy*, para os quais os agentes de usuário podem enviar registros, convites para sessões e outras solicitações.

Seu papel no contexto do balanceamento de carga, o SIP desempenha um papel crucial ao permitir a distribuição eficiente de solicitações entre servidores, garantindo uma utilização equitativa dos recursos disponíveis. Além disso, o SIP se integra de maneira significativa ao WebRTC, uma vez que fornece os meios descobrir participantes de sessões e acordar sobre os parâmetros da sessão compartilhada.

3 PROPOSTA

A proposta para este projeto consiste em Implementar e analisar um dos algoritmos de balanceamento de carga definidos no Capítulo 2. Os passos estão descritos nos tópicos abaixo.

3.1 Definição do Projeto

A definição do projeto será estabelecida com base nos estudos conduzidos durante o Capítulo 2, culminando na seleção de um algoritmo específico para a implementação do balanceamento de carga.

Além disso, será necessário definir as ferramentas a serem empregadas para a implementação e avaliação do desempenho do algoritmo.

3.2 Implementação do Software

A implementação do algoritmo para o balanceamento de carga será conduzida de maneira abrangente, contemplando diversos cenários para fornecer uma análise mais robusta e adaptável.

Inicialmente, o foco estará na integração do algoritmo com o servidor, para alcançar a funcionalidade desejada. Este ponto de partida será enriquecido por variações que abordam diferentes cenários de implementação.

- Uso de DNS: O projeto considerará cenários tanto com quanto sem o uso do DNS para resolução de nomes. Avaliaremos como a implementação se comporta em ambas as situações, analisando impactos na eficiência e flexibilidade.
- Versão do HTTP: Diferentes versões do protocolo HTTP serão consideradas na implementação, explorando casos que envolvem o HTTP/1.1 e o HTTP/2. Isso permitirá uma análise aprofundada do desempenho do algoritmo em contextos variados.
- TCP e UDP: A implementação será adaptada para suportar tanto o protocolo TCP quanto o UDP. Investigaremos como cada protocolo influencia o desempenho do algoritmo, especialmente em situações de alta carga e comunicações em tempo real.

- Cache Web: A integração de técnicas de cache web será incorporada ao projeto. Isso incluirá o estudo de estratégias de cache para otimizar o desempenho, considerando a eficácia do algoritmo em ambientes com armazenamento em cache.
- *Proxy* Reverso: Será explorada a implementação do algoritmo em cenários que envolvem o uso de *proxy* reverso. Essa abordagem permitirá analisar como o balanceamento de carga opera em ambientes nos quais o tráfego passa por um intermediário antes de atingir os servidores finais.

Essas adaptações na implementação visam enriquecer a análise de desempenho, proporcionando uma compreensão mais abrangente das capacidades e limitações do algoritmo em diferentes contextos. Ao considerar uma gama diversificada de cenários, o estudo se tornará mais abrangente, contribuindo para uma avaliação mais completa do impacto do balanceamento de carga.

3.3 Avaliação de Desempenho

A Avaliação de Desempenho será conduzida empregando ferramentas especializadas, para obter uma visão abrangente do comportamento do sistema durante a implementação do algoritmo para balanceamento de carga.

3.4 Análise dos resultados

A análise dos resultados obtidos na avaliação de desempenho será realizada de maneira abrangente, utilizando as métricas capturadas por ferramentas de análise de desempenho e outros dados relevantes. A interpretação das métricas, incluindo latência, tempos de resposta, taxas de transferência e eficiência na entrega de pacotes, será conduzida com atenção especial para variações e tendências observadas durante os testes.

A comparação dos resultados entre diferentes cenários e variações implementadas permitirá destacar as diferenças de desempenho em condições específicas, identificando pontos fortes e áreas que demandam melhorias.

3.5 Cronograma

Ao longo das quatro primeiras semanas de fevereiro, será dedicado tempo ao estudo aprofundado das ferramentas de software necessárias para a implementação do projeto. O período subsequente, abrangendo quatro semanas de março e a primeira de abril, será destinado ao desenvolvimento e implementação do projeto, concentrando esforços na integração do algoritmo e na configuração do ambiente com as adaptações necessárias.

Posteriormente, ao longo de três semanas de abril, realizar-se-á a avaliação do desempenho, conduzindo testes em ambientes controlados e registrando métricas fundamentais. Nas três semanas de maio, a análise detalhada dos resultados será conduzida, utilizando ferramentas de análise de desempenho para interpretar as métricas e identificar padrões de comportamento.

Na última semana de maio e nas quatro semanas de junho, concentrarão os esforços na documentação completa do processo e resultados obtidos. Esta fase incluirá a elaboração de relatórios detalhados, a descrição do código implementado, configurações do servidor utilizado, procedimentos adotados e conclusões derivadas da análise. Essa abordagem sequencial e estruturada visa garantir a execução eficiente e organizada de cada fase do projeto, culminando em uma documentação abrangente.

Tabela 1 – Cronograma de atividades

Etapas	Fev				Mar				Abr				Maio				Junho			
	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
Estudo	X	X	X	X	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Desenvolvimento	–	–	–	–	X	X	X	X	X	–	–	–	–	–	–	–	–	–	–	–
Avaliação	–	–	–	–	–	–	–	–	–	X	X	X	–	–	–	–	–	–	–	–
Análise	–	–	–	–	–	–	–	–	–	–	–	–	X	X	X	–	–	–	–	–
Documentação	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	X	X	X	X	X

Fonte: Elaborada pelo autor.

4 CONSIDERAÇÕES PARCIAIS

O objetivo principal deste projeto de pesquisa é analisar o impacto das técnicas de balanceamento de carga em ambientes SIP e WebRTC, com o propósito de aprimorar a qualidade das comunicações em tempo real. Diante da crescente demanda por comunicações em tempo real e do aumento do uso de protocolos como SIP e WebRTC, é essencial abordar os desafios de garantir uma experiência confiável em redes dinâmicas e congestionadas. Este estudo utilizará uma metodologia abrangente, incluindo a definição de objetivos, testes em ambientes representativos, implementação de técnicas de balanceamento de carga, coleta de dados de desempenho e análise estatística, com o objetivo de fornecer *insights* valiosos para o avanço das comunicações em tempo real.

REFERÊNCIAS

BELGAUM, M. R. et al. A systematic review of load balancing techniques in software-defined networking. *IEEE Access*, v. 8, p. 98612–98636, 2020. Disponível em: https://ieeexplore.ieee.org/document/9097181. 11

BELSHE, M.; PEON, R.; THOMSON, M. *Hypertext Transfer Protocol Version 2 (HTTP/2)*. RFC Editor, 2015. RFC 7540. (Request for Comments, 7540). Disponível em: https://www.rfc-editor.org/info/rfc7540. 21

BISHOP, M. *HTTP/3*. RFC Editor, 2022. RFC 9114. (Request for Comments, 9114). Disponível em: https://www.rfc-editor.org/info/rfc9114. 22

Denise Ribeiro; Anthony Wells. *Com pandemia, demanda por videoconferências dispara em empresas brasileiras*. 2023. Acessado em 12 de setembro de 2023. Disponível em: https://www.cnnbrasil.com.br/economia/com-pandemia-demanda-por-videoconferencias-dispara-em-empresas-brasileiras/. 7

DéLY, T. L. Caching HTTP: A comparative study of caching reverse proxies Varnish and Nginx. *Journal of Web Performance and Optimization*, June 2014. Disponível em: https://www.diva-portal.org/smash/get/diva2:734117/FULLTEXT01.pdf. 24

EDDY, W. *Transmission Control Protocol (TCP)*. RFC Editor, 2022. RFC 9293. (Request for Comments, 9293). Disponível em: https://www.rfc-editor.org/info/rfc9293. 16

FIELDING, R. T.; NOTTINGHAM, M.; RESCHKE, J. *HTTP/1.1*. RFC Editor, 2022. RFC 9112. (Request for Comments, 9112). Disponível em: https://www.rfc-editor.org/info/rfc9112. 20

GRIGORIK, I. *High Performance Browser Networking*. O'Reilly Media, Inc., 2013. ISBN 9781449344764. Disponível em: https://hpbn.co/. 17, 23, 25, 26

HTTP.DEV. *HTTP/2*. 2022. https://http.dev/2. 22, 23

JIANG, H. et al. Design, implementation, and performance of a load balancer for sip server clusters. *IEEE/ACM Transactions on Networking*, v. 20, n. 4, August 2012. Disponível em: https://dl-acm-org.ez130.periodicos.capes.gov.br/doi/pdf/10.1109/TNET.2012.2183612. 14

KEMP, S. *DIGITAL 2022: GLOBAL OVERVIEW REPORT*. 2022. Acessado em 12 de setembro de 2023. Disponível em: https://datareportal.com/reports/digital-2022-global-overview-report. 7

LOUREIRO, A. A. et al. Comunicação sem fio e computação móvel: Tecnologias, desafios e oportunidades. 2023. Acessado em 12 de setembro de 2023. Disponível em: https://homepages.dcc.ufmg.br/~loureiro/cm/docs/jai03.pdf. 7

MELNIKOV, A.; FETTE, I. *The WebSocket Protocol*. RFC Editor, 2011. RFC 6455. (Request for Comments, 6455). Disponível em: https://www.rfc-editor.org/info/rfc6455. 23

MOCKAPETRIS, P. *Domain names - concepts and facilities*. RFC Editor, 1987. RFC 1034. (Request for Comments, 1034). Disponível em: https://www.rfc-editor.org/info/rfc1034. 19

MOCKAPETRIS, P. *Domain names - implementation and specification*. RFC Editor, 1987. RFC 1035. (Request for Comments, 1035). Disponível em: https://www.rfc-editor.org/info/rfc1035. 19

NEGHABI, A. A. et al. Load balancing mechanisms in the software defined networks: A systematic and comprehensive review of the literature. *IEEE Access*, v. 6, p. 14159–14178, 2018. 8, 11

NOTTINGHAM, M. *Caching Tutorial*. 2013. Disponível em: http://www.mnot.net/cache_docs/. 25

POSTEL, J. *User Datagram Protocol*. RFC Editor, 1980. RFC 768. (Request for Comments, 768). Disponível em: https://www.rfc-editor.org/info/rfc768. 18

ROSE, S. et al. *DNS Security Introduction and Requirements*. RFC Editor, 2005. RFC 4033. (Request for Comments, 4033). Disponível em: https://www.rfc-editor.org/info/rfc4033. 19

SAJJAN, R. S.; BIRADAR, R. Y. Load balancing and its algorithms in cloud computing: A survey. *Survey Paper*, E-ISSN: 2347-2693, v. 5, n. 1, 2017. Available online at: http://www.ijcseonline.org. Disponível em: https://www.researchgate.net/profile/Rajani-Sajjan-2/publication/313766818_Load_Balancing_and_its_Algorithms_in_Cloud_Computing_A_Survey/links/58a554d44585150402cc4376/Load-Balancing-and-its-Algorithms-in-Cloud-Computing-A-Survey.pdf. 12, 13, 14, 15

SCHOOLER, E. et al. *SIP: Session Initiation Protocol*. RFC Editor, 2002. RFC 3261. (Request for Comments, 3261). Disponível em: https://www.rfc-editor.org/info/rfc3261. 26, 27