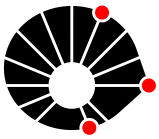

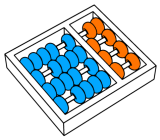

Explorando Estratégias de Content Steering para Transmissões de Vídeo Adaptativas

Lucas Jacinto Gonçalves

Relatório Técnico - IC-PFG-24-25
Projeto Final de Graduação
2024 - Dezembro

UNIVERSIDADE ESTADUAL DE CAMPINAS
INSTITUTO DE COMPUTAÇÃO

Explorando Estratégias de Content Steering para Transmissões de Vídeo Adaptativas

Lucas Jacinto Gonçalves*

Resumo

Este trabalho apresenta o desenvolvimento de um sistema adaptativo de *content steering* para otimização do streaming de vídeo em ambientes *edge-cloud*. O sistema foi projetado para enfrentar os desafios da entrega eficiente de conteúdo audiovisual em cenários caracterizados por heterogeneidade nas condições de rede, dispositivos e demandas dos usuários. Utilizando uma abordagem híbrida que combina métodos tradicionais de gerenciamento de rede e estratégias baseadas em Inteligência Artificial, o projeto implementa uma solução que ajusta dinamicamente os parâmetros de rede e a alocação de servidores de cache para maximizar a Qualidade de Experiência (QoE).

A arquitetura do sistema integra um painel de controle interativo para monitoramento e ajuste de condições de rede em tempo real, simulando diferentes cenários de conectividade (como 2G, 3G, 4G, 5G, 6G e fibra óptica). A seleção de servidores foi implementada de forma dual: utilizando métricas tradicionais de rede e um modelo de aprendizado de máquina baseado em *Random Forest*, treinado para prever o servidor ideal com base em condições de rede e carga dos servidores. A solução incorpora ferramentas de monitoramento de tráfego, controle adaptativo e geração de gráficos para análise detalhada do desempenho.

Este trabalho oferece uma contribuição significativa ao campo de streaming adaptativo, com uma abordagem modular e extensível que pode ser aplicada em cenários reais de distribuição de conteúdo audiovisual. Além disso, fornece informações relevantes para o desenvolvimento de estratégias futuras que explorem o potencial das arquiteturas *edge-cloud* e do *content steering* adaptativo.

1 Introdução

O streaming de vídeo tornou-se um elemento central da experiência digital contemporânea, transformando profundamente a maneira como consumimos conteúdo audiovisual. [1] Com a crescente demanda por streaming de alta qualidade, impulsionada pela popularização de serviços como Netflix, YouTube e Amazon Prime, surgem desafios significativos relacionados à distribuição eficiente desse conteúdo em escala global.

*l240013@dac.unicamp.br

[1] Fonte: Jornal da USP. Disponível em: https://jornal.usp.br/atualidades/crescimento-do-streaming-modifica-o-consumo-de-producoes-audiovisuais/

Os provedores de streaming enfrentam o desafio de oferecer uma experiência de visualização uniforme e de alta qualidade em um cenário marcado por uma diversidade significativa. [2] Essa diversidade envolve fatores como a ampla gama de dispositivos utilizados pelos usuários, que vão de smartphones a televisores inteligentes, diferentes tipos de conexão à internet, como redes móveis e conexões de fibra ou cabo, além de condições de rede que podem variar ao longo de uma sessão de streaming.

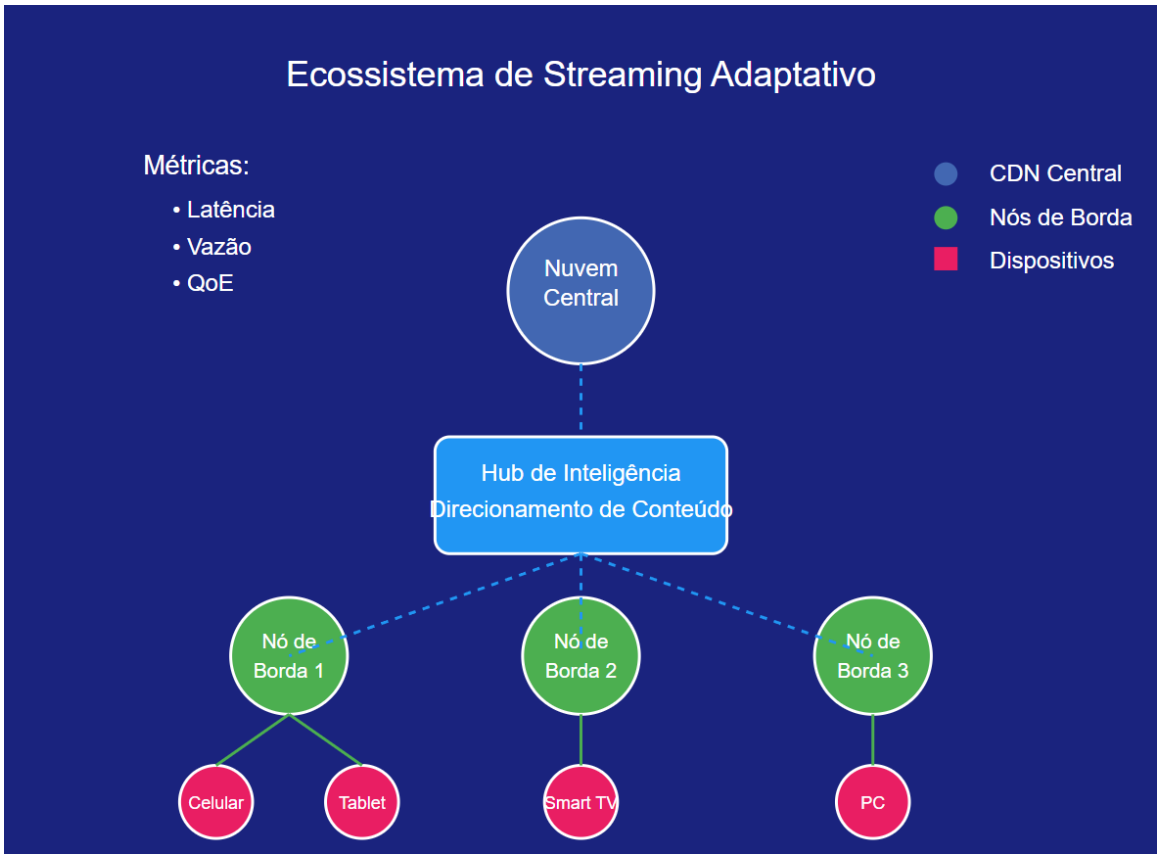

Figura 1: Ecossistema de Streaming Adaptativo

Outro desafio relevante é o aumento contínuo do tráfego de vídeo na Internet, que tem gerado uma pressão crescente sobre a infraestrutura de rede tradicional. Estima-se que, no Brasil, o tráfego de vídeo na internet crescerá três vezes entre 2016 e 2021, com uma taxa de crescimento anual composta de 23% [3] . Essa tendência é intensificada pela crescente popularidade de conteúdos em alta resolução, como 4K e 8K, reforçando a necessidade de soluções inovadoras para otimizar a distribuição de conteúdo e reduzir a latência percebida pelos usuários.

[2]Fonte: CDN Star. Disponível em: https://cdnstar.com.br/o-crescimento-exponencial-do-mercado-de-streaming-no-brasil-e-os-desafios-dos-isps/

[3]Fonte: Núcleo do Conhecimento. Disponível em: `https://www.nucleodoconhecimento.com.br/ciencia-da-computacao/over-the-top?`

A emergência do paradigma de *edge computing* apresenta novas oportunidades para enfrentar esses desafios. [4]Ao aproximar o conteúdo dos usuários através de servidores de cache distribuídos geograficamente, é possível reduzir significativamente a latência e melhorar a qualidade geral do streaming. No entanto, gerenciar eficientemente essa infraestrutura distribuída e decidir dinamicamente qual servidor deve atender cada requisição não é uma tarefa trivial.

É neste contexto que se insere o conceito de *content steering*, uma abordagem que visa otimizar a distribuição de conteúdo através do direcionamento inteligente das requisições dos usuários para os servidores mais apropriados. Esta otimização precisa considerar múltiplos fatores em tempo real, como a qualidade da conexão, a carga dos servidores e a disponibilidade do conteúdo em cache.

O presente trabalho aborda estes desafios através do desenvolvimento de um sistema adaptativo de *content steering* para *streaming* de vídeo em ambientes *edge-cloud*. O sistema proposto combina técnicas tradicionais de gerenciamento de rede com abordagens inovadoras baseadas em Inteligência Artificial, buscando otimizar a experiência do usuário em diferentes cenários de conectividade.

O projeto tem como objetivo principal desenvolver uma solução que possa adaptar-se dinamicamente às condições variáveis de rede, mantendo a melhor qualidade possível de streaming mesmo em situações desafiadoras. Para isso, implementa um conjunto de funcionalidades que permitem não apenas o monitoramento em tempo real das condições de rede, mas também a simulação de diferentes cenários de conectividade, facilitando o teste e a validação de estratégias de otimização.

A contribuição deste trabalho transcende o ambiente acadêmico, oferecendo insights valiosos para a indústria de streaming e telecomunicações. Para provedores de conteúdo, apresenta estratégias eficientes de distribuição que podem resultar em economia significativa de recursos e melhoria na satisfação dos usuários. Para operadoras de rede, fornece uma compreensão mais profunda dos padrões de tráfego e requisitos de infraestrutura em diferentes cenários de streaming. Para desenvolvedores, oferece um framework extensível que pode servir como base para futuras inovações em sistemas adaptativos de distribuição de conteúdo. O projeto também contribui para o avanço do estado da arte em streaming adaptativo, propondo novas abordagens para a otimização da entrega de conteúdo em ambientes *edge-cloud*.

2 Fundamentação Teórica

Nesta seção, são apresentados os conceitos fundamentais relacionados ao *streaming* adaptativo, *QoE*, *content steering* e tecnologias correlatas. Esses conceitos servem como base para o desenvolvimento do sistema proposto.

[4]Fonte: Red Hat. Disponível em: https://www.redhat.com/pt-br/topics/edge-computing/what-is-edge-computing?

2.1 Streaming Adaptativo HTTP

O protocolo (*Dynamic Adaptive Streaming over HTTP - DASH*) organiza o conteúdo de vídeo em segmentos curtos, geralmente entre dois e dez segundos, codificados em múltiplas qualidades. [5] Durante a reprodução, o cliente seleciona dinamicamente a qualidade mais adequada para cada segmento, considerando as condições da rede e a capacidade do dispositivo. Essa escolha é guiada pelo manifesto *DASH*, denominado (*Media Presentation Description - MPD*), um documento XML que descreve as representações disponíveis do vídeo, URLs dos segmentos, informações temporais, sincronização e metadados de qualidade e codificação.

A capacidade de adaptação do *DASH* permite ajustar a reprodução para minimizar interrupções e manter uma experiência de alta qualidade, especialmente em cenários de redes instáveis ou com variações na largura de banda.

2.2 Qualidade de Experiência (QoE)

O *QoE* no *streaming* adaptativo é um conceito essencial para avaliar a percepção do usuário em relação à qualidade do serviço. O *QoE* é avaliado utilizando uma combinação de métricas objetivas e subjetivas, que medem tanto os aspectos técnicos da entrega do conteúdo quanto a experiência percebida pelo usuário:

- **Métricas objetivas:** incluem resolução do vídeo, taxa de bits (*bitrate*), tempo de inicialização da reprodução, frequência de rebufferizações e mudanças na qualidade durante a reprodução. Essas métricas são diretamente relacionadas ao desempenho da rede e aos recursos do dispositivo.

- **Métricas subjetivas:** consideram a percepção do usuário, como satisfação geral, engajamento com o conteúdo e tolerância a problemas técnicos, fatores que não podem ser diretamente quantificados, mas são cruciais para medir a qualidade percebida.

Fatores como as condições da rede (latência, largura de banda e perda de pacotes), a capacidade do dispositivo (poder de processamento e compatibilidade de codecs) e as expectativas individuais dos usuários influenciam fortemente o *QoE*. Estudos demonstram que métricas objetivas, como a redução de rebufferizações e o aumento da resolução, têm uma correlação direta com a satisfação do usuário, enquanto métricas subjetivas são críticas para avaliar a experiência geral em contextos variados. [6]

A compreensão e a otimização do *QoE* são fundamentais para o sucesso de sistemas de *streaming* adaptativo, uma vez que melhoram o engajamento do usuário e reduzem a taxa de cancelamento de serviços.

[5] Fonte: CloudFlare. Disponível em: https://www.cloudflare.com/pt-br/learning/video/what-is-mpeg-dash/

[6] Fonte: IEEE Communications Magazine. Disponível em: https://ieeexplore.ieee.org/document/6913491

2.3 Content Steering

O *content steering* é uma técnica que direciona dinamicamente as requisições dos usuários para os servidores mais adequados, considerando critérios como proximidade geográfica, carga atual dos servidores, disponibilidade do conteúdo em cache e condições da rede. Essa abordagem utiliza algoritmos para tomar decisões em tempo real, maximizando a eficiência da infraestrutura de rede e aprimorando a experiência do usuário. Por exemplo, ao responder a uma solicitação de conteúdo da web com uma versão em cache mais próxima fisicamente do usuário final, uma CDN alivia o tráfego nos servidores de origem e melhora a experiência na web. [7]

Essa técnica utiliza algoritmos para tomar decisões em tempo real, maximizando a eficiência da infraestrutura de rede e otimizando a experiência do usuário.

2.4 Tecnologias Correlatas

A escolha de codecs de vídeo desempenha um papel crucial nos sistemas de *streaming* adaptativo. Codecs amplamente utilizados, como H.264 (AVC), H.265 (HEVC), VP9 e AV1, diferem em termos de eficiência de compressão, qualidade visual, requisitos computacionais e compatibilidade com dispositivos. O H.265, por exemplo, proporciona uma compressão de dados mais eficiente em relação ao H.264, reduzindo a largura de banda necessária para transmitir vídeos de alta qualidade. Apesar disso, o H.264 ainda é amplamente utilizado devido à sua compatibilidade com uma grande variedade de dispositivos e plataformas.

A seleção apropriada de codecs é essencial para alcançar um equilíbrio entre qualidade de vídeo e eficiência de transmissão, especialmente em cenários onde a largura de banda é limitada ou onde há necessidade de compatibilidade com dispositivos legados. Novos codecs, como o AV1, têm se destacado pela sua eficiência em compressão e crescente adoção por empresas importantes do setor. [8]

Além dos codecs, padrões de tráfego também desempenham um papel relevante. O consumo de vídeo pode apresentar picos em horários específicos ou variar sazonalmente, o que exige estratégias de gestão de recursos para evitar gargalos e garantir uma entrega eficiente. Técnicas de previsão e ajustes proativos na rede são fundamentais para manter o desempenho em cenários de alta demanda. A análise detalhada do comportamento do tráfego permite a alocação otimizada de recursos, reduzindo o impacto de flutuações na demanda.

3 Arquitetura do Sistema

O sistema desenvolvido apresenta uma arquitetura que integra diferentes componentes para otimizar a entrega de conteúdo de vídeo através do streaming adaptativo. Esta arquitetura foi projetada levando em consideração os desafios modernos de distribuição de conteúdo, incluindo variações nas condições de rede, mobilidade dos usuários e necessidade de escalabilidade. A arquitetura pode ser compreendida em duas perspectivas principais: a infra-

[7] Fonte: Akamai. Disponível em: https://www.akamai.com/pt/glossary/what-is-a-cdn?
[8] Fonte: Wikipedia. Disponível em: https://en.wikipedia.org/wiki/AV1

estrutura física edge-cloud, ilustrada na Figura 2, e a arquitetura de software do sistema, detalhada na Figura 3.

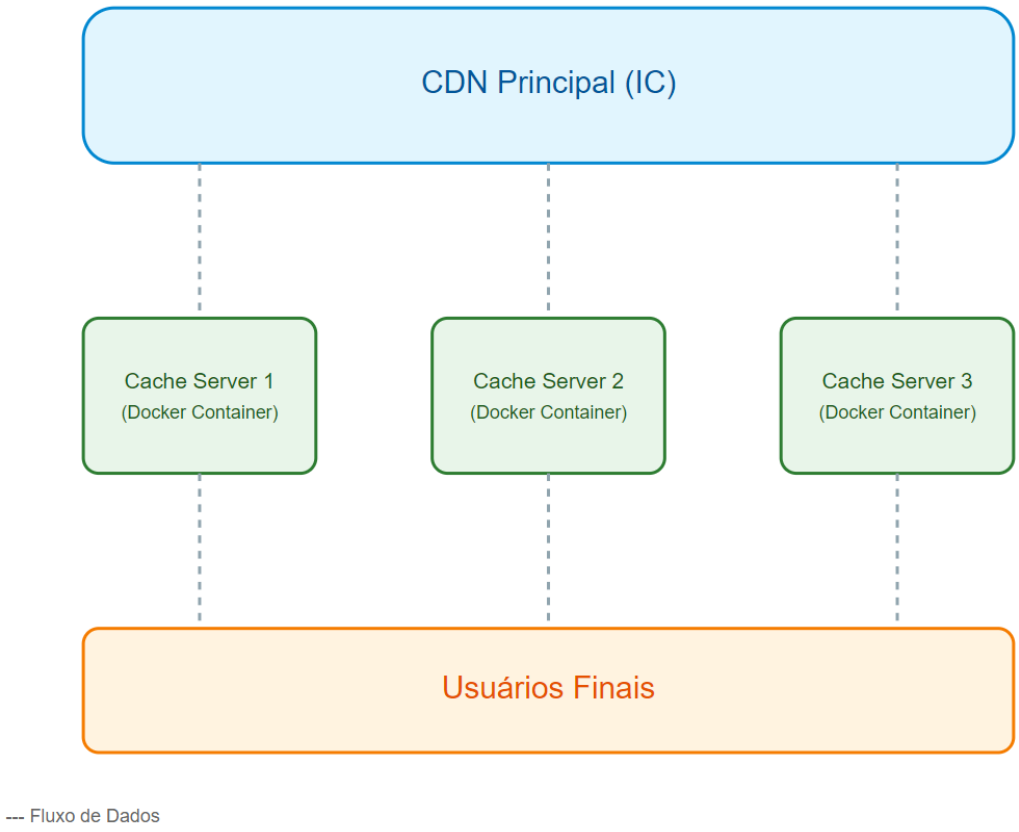

Figura 2: Infraestrutura Física Edge-Cloud

A infraestrutura *edge-cloud*, apresentada na Figura 2, é constituída por três camadas distintas e hierárquicas. No topo da hierarquia, encontra-se a Rede de Distribuição de Conteúdo (Content Delivery Network - CDN) principal. Este servidor é responsável pelo armazenamento dos vídeos em diferentes qualidades e pelo gerenciamento inicial das conexões dos clientes.

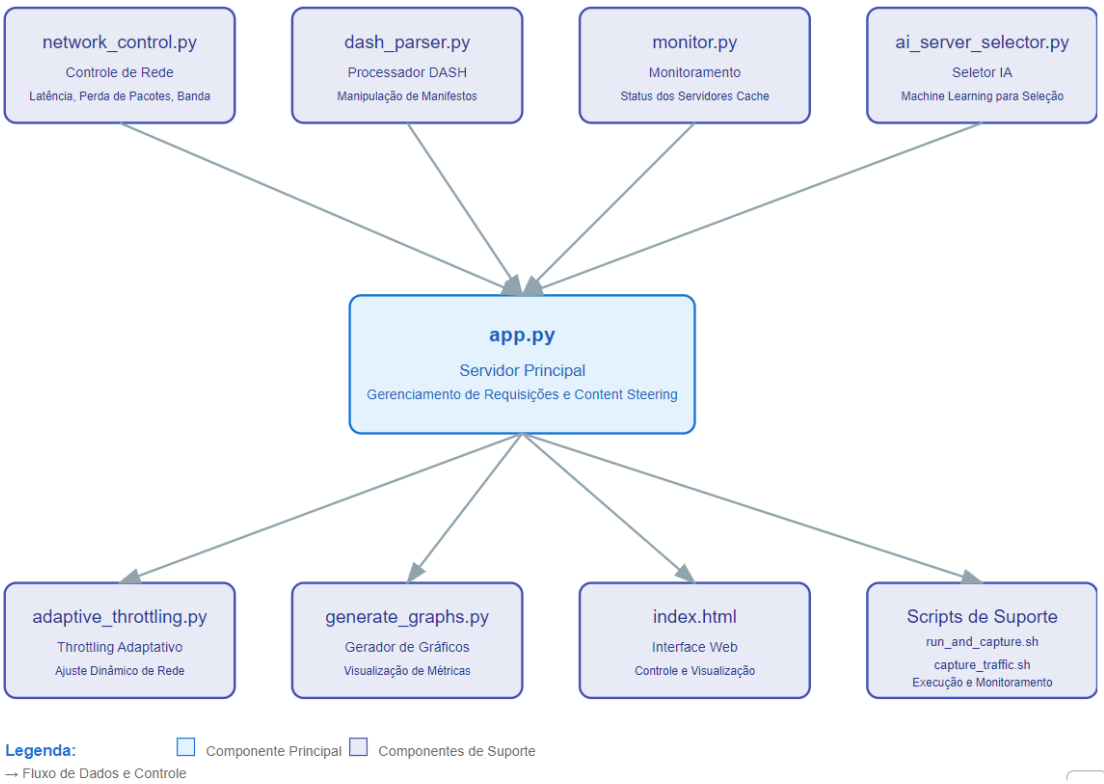

Figura 3: Arquitetura de Software do Sistema

Na camada intermediária, encontram-se três servidores de cache (Cache Server 1, Cache Server 2 e Cache Server 3), implementados como containers Docker que são executados localmente na máquina do usuário. Como mostrado na Figura 2, estes servidores atuam como pontos de distribuição de borda, simulando efetivamente uma infraestrutura *edge-cloud* real. A escolha da tecnologia Docker para implementação destes servidores possibilita um rápido deploy, isolamento eficiente de recursos e alta portabilidade. As linhas tracejadas na figura representam o fluxo de dados entre as diferentes camadas, ilustrando como o conteúdo é distribuído do CDN Principal até os usuários finais através dos servidores de cache.

Na base da infraestrutura, encontram-se os usuários finais, que representam os consumidores do conteúdo de *streaming*. As conexões tracejadas entre os servidores de cache e os usuários finais demonstram os múltiplos caminhos possíveis para a entrega do conteúdo, permitindo que o sistema escolha dinamicamente o melhor servidor baseado nas condições de rede e disponibilidade dos servidores.

A arquitetura de software do sistema, ilustrada na Figura 3, apresenta uma organização centralizada em torno do componente principal `app.py`, que atua como o núcleo do sistema, gerenciando requisições e coordenando o *content steering*. Este componente central interage com diversos módulos especializados, tanto recebendo dados e comandos dos componentes

superiores quanto controlando e coordenando os componentes inferiores.

Na parte superior do diagrama de software, encontram-se quatro componentes essenciais: `network_control.py`, responsável pelo controle das condições de rede (latência, perda de pacotes e banda); `dash_parser.py`, que processa e manipula os manifestos *DASH*; `monitor.py`, que monitora o status dos servidores de cache; e `ai_server_selector.py`, que implementa a seleção inteligente de servidores utilizando machine learning.

Na parte inferior, o sistema conta com componentes de suporte igualmente importantes: `adaptive_throttling.py`, que realiza o ajuste dinâmico das condições de rede; `generate_graphs.py`, responsável pela geração de visualizações e métricas; `index.html`, que implementa a interface web para controle e visualização; e os scripts de suporte de nomes (`run_and_capture.sh` e `capture_traffic.sh`), que auxiliam na execução e monitoramento do sistema.

O fluxo de dados e controle, representado pelas setas no diagrama de software, demonstra como as informações e comandos fluem entre os diferentes componentes. Todas as interações são coordenadas pelo componente central `app.py`, que recebe informações dos módulos de monitoramento e controle, processa estas informações e coordena as ações dos componentes de execução e visualização.

A arquitetura implementa um sistema abrangente de logging e geração de gráficos, que permite o monitoramento em tempo real e análise posterior do desempenho do sistema. Isso é particularmente importante para a avaliação da qualidade da entrega de conteúdo e métricas de QoE, facilitando a identificação de pontos de melhoria e otimização.

A natureza modular da arquitetura, evidenciada em ambos os diagramas, permite que o sistema seja facilmente expandido ou modificado. Novos servidores de cache podem ser adicionados à infraestrutura *edge-cloud*, e novos componentes de software podem ser integrados ao sistema sem comprometer sua integridade. Esta flexibilidade, combinada com os mecanismos de recuperação automática de falhas e balanceamento dinâmico de carga, resulta em um sistema resiliente e adaptável às mudanças nas condições de rede e demandas dos usuários.

4 Implementação Técnica

A implementação do sistema de *content steering* adaptativo para streaming de vídeo foi realizada de forma modular, permitindo o desenvolvimento, teste e integração independentes de seus componentes. A seguir, descrevem-se os principais módulos e suas funcionalidades:

4.1 Controle de Rede Adaptativo

Este módulo utiliza a ferramenta *tc* (*Traffic Control*) do Linux para configurar dinamicamente parâmetros como latência, largura de banda e perda de pacotes, simulando cenários variados de conectividade. A lógica é encapsulada na classe `NetworkControl`, que:

- Detecta automaticamente a interface de rede ativa, ignorando interfaces de *loopback*, utilizando a biblioteca `netifaces`.

- Permite configurar condições de rede padrão, como latência de 35 ms, largura de banda de 10.000 kbit/s e perda de pacotes de 0,5%.
- Implementa o método `update_conditions`, que aplica novas regras de rede somente após verificar mudanças significativas nos valores configurados.
- Inclui métodos para monitorar o estado atual da rede (`get_current_conditions`) e diagnosticar as regras aplicadas (`get_tc_rules`).

Esse módulo é fundamental para testar o desempenho do sistema sob diferentes condições de rede e garantir a estabilidade da aplicação.

4.2 Throttling Adaptativo

O `adaptive_throttling.py` ajusta automaticamente as condições de rede com base em métricas de desempenho em tempo real. A classe `AdaptiveThrottling` implementa:

- Coleta de métricas de rede e cálculo de médias ponderadas para priorizar informações recentes.
- Ajuste automático de parâmetros de latência, largura de banda e perda de pacotes com base em um fator de correção.
- Integração com o módulo `network_control.py` para aplicar configurações dinâmicas.
- Suporte a atualizações manuais, interrompendo ajustes automáticos quando necessário para testes específicos.

Este componente garante que o sistema mantenha a *QoE* em níveis aceitáveis, mesmo em condições de rede adversas.

4.3 Seleção de Servidores com IA

Este módulo implementa a lógica de *content steering* baseada em aprendizado de máquina, utilizando um modelo de *Random Forest* para prever o servidor ideal em tempo real. A classe `AIServerSelector` oferece:

- Treinamento do modelo com dados sintéticos que simulam diferentes condições de rede, incluindo latência, perda de pacotes, largura de banda, uso de CPU e memória.
- Normalização de métricas com `StandardScaler`, garantindo maior precisão nas previsões.
- Método `predict_best_server`, que avalia os servidores disponíveis e seleciona aquele com maior *QoE* prevista.
- Integração com o módulo `monitor.py` para coletar métricas atualizadas de servidores.

Esse módulo permite decisões precisas e adaptativas, otimizando a experiência do usuário em tempo real.

4.4 Manipulação de Manifestos DASH

Responsável por processar e ajustar manifestos *DASH* (*MPD*), o módulo `dash_parser.py` reflete as condições de rede e decisões de *content steering*. As funcionalidades incluem:

- Configuração de pesos (`alpha`, `beta` e `delta`) para priorizar latência, largura de banda e perda de pacotes na seleção de servidores.
- Ajuste dinâmico dos URLs dos segmentos no manifesto, priorizando servidores adequados.
- Método `build`, que gera manifestos adaptados com base nas condições atuais de rede e no servidor selecionado.
- Suporte à atualização de pesos e limites de largura de banda em tempo real.

Esse módulo é essencial para garantir que as decisões de *content steering* sejam refletidas diretamente na reprodução de vídeos.

4.5 Interface Web do Dashboard

O painel de controle do sistema foi desenvolvido em HTML, CSS e JavaScript, com as seguintes funcionalidades:

- Exibição de métricas de rede (*QoE*, latência, largura de banda) e do servidor ativo.
- Controles para alternar entre métodos de *steering* (tradicional e IA) e aplicar *presets* de rede como 2G, 3G, 4G, 5G, 6G e fibra óptica.
- Player de vídeo configurado para reprodução adaptativa utilizando manifestos *DASH*.
- Atualização em tempo real de informações da interface, comunicando-se com o *backend* via requisições HTTP.

A interface é responsiva e foi projetada para facilitar o monitoramento e controle do sistema.

4.6 Monitoramento de Contêineres

Este módulo monitora continuamente os servidores de cache e gerencia seu estado, implementando:

- Verificação periódica da saúde dos servidores via requisições HTTP.
- Reinicialização automática de contêineres inativos.
- Registro de eventos em `monitor.log`, incluindo alterações no estado dos servidores.
- Métodos para ajustar manualmente a lista de servidores ativos.

O `monitor.py` é crucial para garantir a robustez do sistema e a continuidade do serviço.

4.7 Aplicação Principal

O `app.py` é o núcleo do sistema, integrando os módulos mencionados e fornecendo rotas HTTP para controle e monitoramento. As funcionalidades incluem:

- Inicialização do sistema, configurando variáveis de ambiente, módulos e logs.
- Exposição de rotas para configurar condições de rede, gerenciar servidores e ajustar manifestos *DASH*.
- Registro detalhado de eventos críticos, como mudanças nos métodos de *steering* e ajustes de rede.
- Integração com o `monitor.py` para monitorar servidores e gerenciar falhas.

Este componente central coordena todas as operações do sistema, garantindo sua funcionalidade e consistência.

4.8 Script de Inicialização e Captura de Tráfego

Automatiza o processo de inicialização do sistema e captura de tráfego para análise. Suas tarefas incluem:

- Limpeza de logs e encerramento de processos conflitantes.
- Captura de tráfego com `tcpdump`, armazenando dados em `capture.pcap`.
- Execução da aplicação principal e monitoramento de seu estado.
- Geração de gráficos ao final da execução, utilizando `generate_graphs.py`.

O script garante um fluxo de trabalho consistente e facilita a coleta de dados para análise de desempenho.

5 Configuração e Inicialização da Aplicação

A configuração e inicialização do sistema de *content steering* para streaming de vídeo foram projetadas para serem simples e eficientes, garantindo o funcionamento correto de todos os componentes necessários. Este processo é dividido em etapas sequenciais que cobrem desde a preparação dos servidores de cache até a execução da aplicação principal e o monitoramento dos dados coletados.

O repositório oficial do projeto está disponível em: `https://github.com/LucasLJG/Content-Steering-no-Edge-Cloud.git`.

Ele contém o arquivo `.ova` para importar a máquina virtual, além de um tutorial detalhado para auxiliar na configuração inicial e no funcionamento da aplicação.

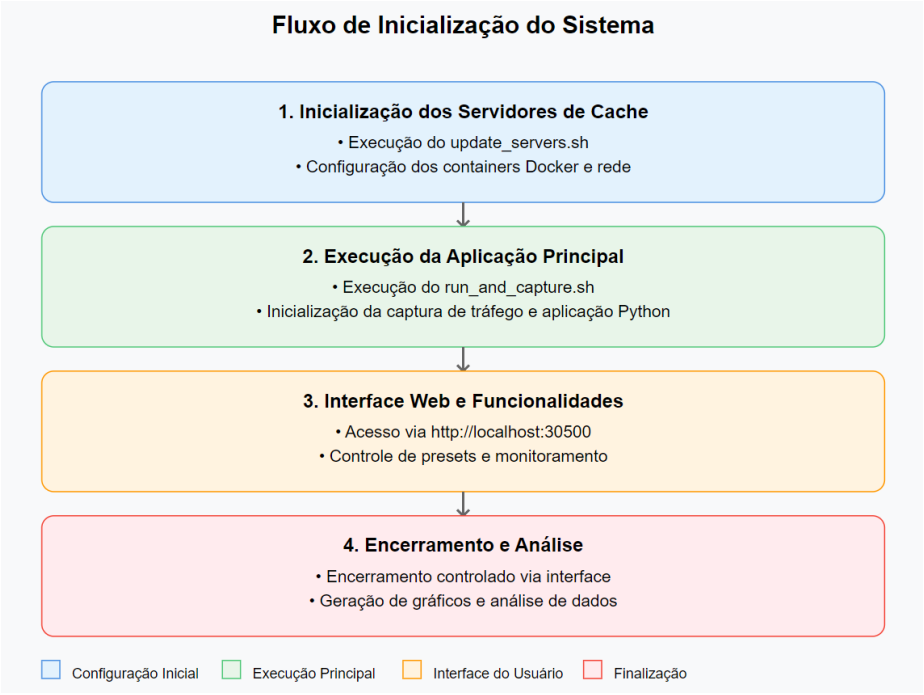

Figura 4: Fluxo de Inicialização do Sistema

5.1 Inicialização dos Servidores de Cache

A primeira etapa consiste em configurar os servidores de cache, responsáveis pela distribuição do conteúdo de vídeo de forma descentralizada. Para isso, acesse o diretório:

```
/home/tutorial/Documents/content-steering-tutorial/streaming-service
```

e execute o script `update_servers.sh` com privilégios de administrador. Utilize o seguinte comando no terminal:

```
sudo ./update_servers.sh
```

Observação: Sempre que solicitado, insira a senha padrão: `tutorial`.

O script executado realiza uma série de operações fundamentais, conforme ilustrado na Figura 5:

- Remove contêineres existentes para garantir um ambiente limpo.
- Constrói imagens Docker necessárias para os servidores de cache.
- Inicia três contêineres de cache (`video-streaming-cache-1`, `video-streaming-cache-2` e `video-streaming-cache-3`).
- Configura a rede Docker para assegurar a comunicação entre os componentes.
- Atualiza automaticamente o arquivo `/etc/hosts` com os endereços IP dos servidores de cache, facilitando o acesso por nomes de domínio.

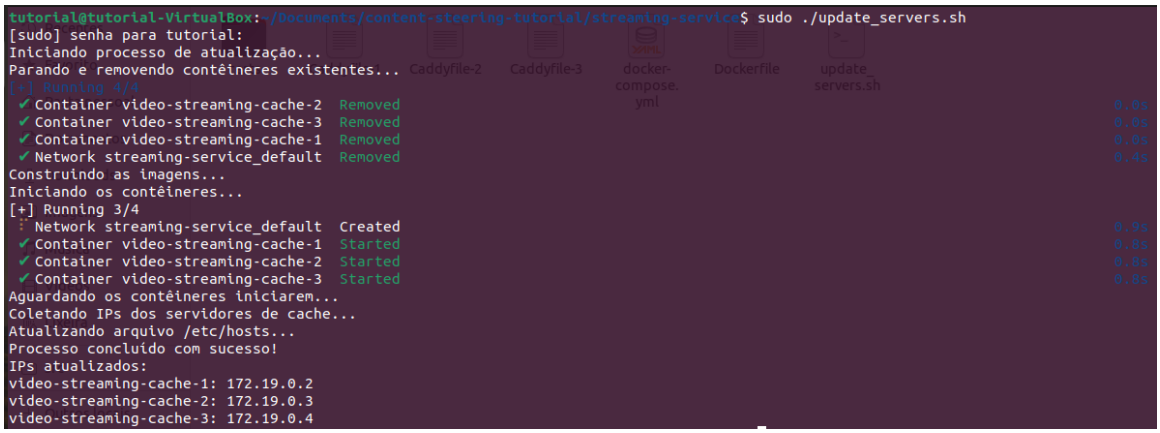

Figura 5: Execução do script para inicializar os servidores de cache.

5.2 Execução da Aplicação Principal

Com os servidores de cache ativos, o próximo passo é executar a aplicação principal. Navegue até o diretório:

```
/home/tutorial/Documents/content-steering-tutorial/steering-service/src
```

e execute o script `run_and_capture.sh` com privilégios de administrador:

```
sudo ./run_and_capture.sh
```

Este script realiza uma série de tarefas críticas:

- Limpa logs antigos para evitar conflitos de execução.
- Libera a porta 30500, caso esteja ocupada, garantindo a inicialização da aplicação.
- Inicia a captura de tráfego de rede (`tcpdump`) para análise posterior.
- Executa a aplicação Python principal, que inclui a lógica de *content steering*.

Durante a execução, o script exibe mensagens informativas, indicando o progresso da inicialização e fornecendo a URL para acessar a interface web da aplicação (`http://localhost:30500`).

5.3 Interface Web e Funcionalidades

Após a inicialização bem-sucedida, a interface web estará acessível no endereço `http://localhost:30500`. Este painel interativo oferece uma série de funcionalidades, permitindo ao usuário monitorar e controlar o sistema de maneira prática e eficiente:

- Carregar vídeos externos utilizando URLs de manifestos MPD.
- Visualizar e controlar o estado dos servidores de cache.
- Alternar entre diferentes *presets* de condições de rede (2G, 3G, 4G, 5G, 6G e *Fiber*).
- Monitorar métricas em tempo real, como QoE, latência e throughput.
- Alternar entre os métodos de *content steering* (Padrão e IA).

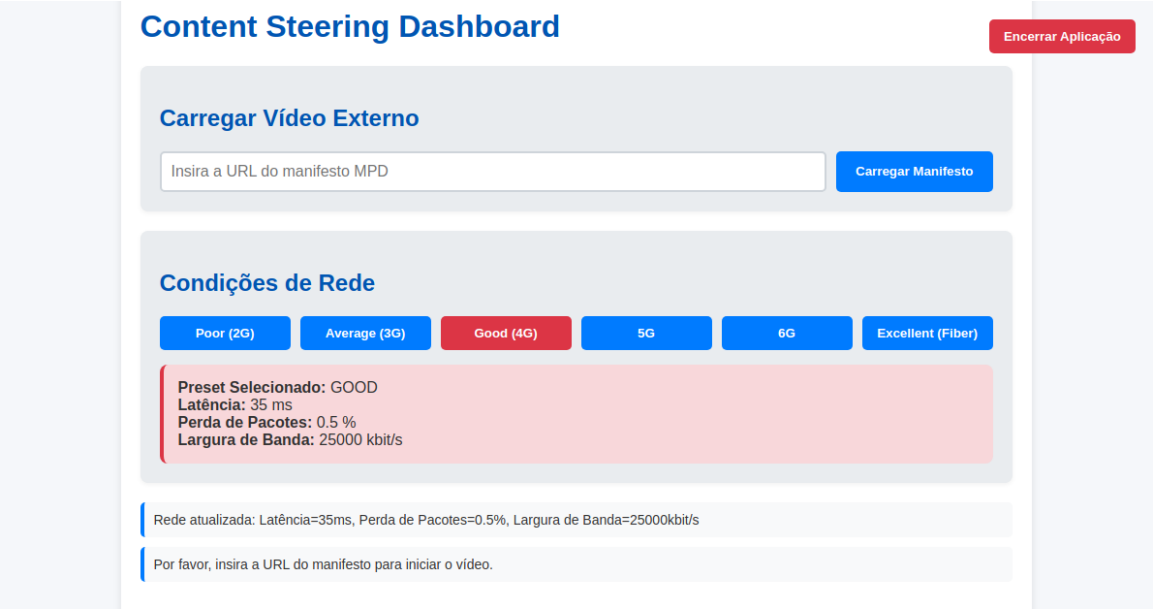

Figura 6: Painel principal da interface web para inserção de URLs e ajuste de condições de rede.

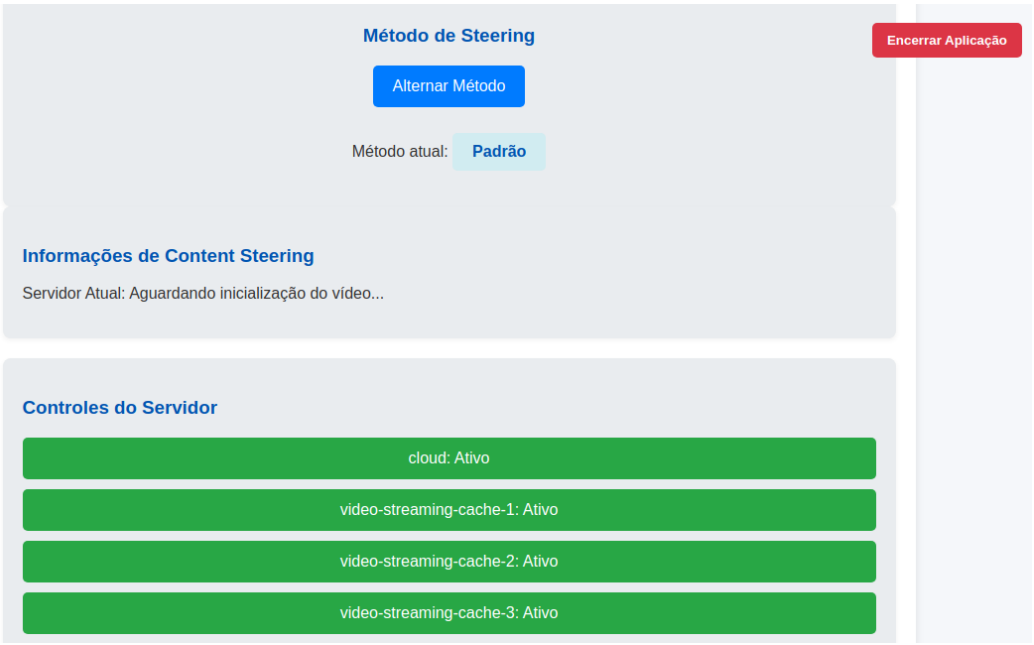

Figura 7: Controles da interface web para gerenciamento do sistema.

5.4 Encerramento e Análise de Dados

Para encerrar a aplicação de maneira adequada, clique no botão "Encerrar Aplicação", localizado no canto superior direito da interface web. Ao realizar este comando, o sistema executa os seguintes passos automaticamente:

- Finaliza a captura de tráfego de rede.
- Gera gráficos com as métricas coletadas, como latência, perda de pacotes, largura de banda e QoE.
- Limpa logs temporários e encerra todos os processos relacionados.

Os gráficos e logs gerados são armazenados no diretório de execução da aplicação, proporcionando dados detalhados para análise posterior. Estes artefatos incluem informações visuais que ajudam a compreender o comportamento do sistema sob diferentes condições de rede e configurações de *content steering*.

A integração clara e automatizada de todos os componentes garante que o sistema funcione de maneira coesa, permitindo uma experiência robusta para experimentos e validação de estratégias de *content steering*.

6 Metodologia

Esta seção descreve os métodos utilizados para desenvolver e validar o sistema adaptativo de *content steering*, com o objetivo de garantir reprodutibilidade e facilitar o entendimento das etapas seguidas.

6.1 Configuração do Ambiente

O sistema foi configurado para rodar em uma máquina virtual baseada em Linux Ubuntu. Scripts automatizados foram desenvolvidos para simplificar o processo de inicialização dos servidores e da aplicação principal. As etapas incluem:

1. Inicialização dos servidores de cache com o script `update_servers.sh`, que:
    - Remove contêineres antigos.
    - Constrói novas imagens Docker.
    - Inicia contêineres configurados para simular servidores de cache distribuídos.
2. Execução da aplicação principal com o script `run_and_capture.sh`, que:
    - Configura variáveis de ambiente.
    - Inicia a aplicação Flask na porta 30500.
    - Captura tráfego de rede para análise posterior.

6.2 Simulação de Cenários de Rede

Para avaliar o sistema, foram definidos seis *presets* de rede que simulam diferentes condições de conectividade:

- **2G:** Alta latência, baixa largura de banda e alta perda de pacotes.
- **3G:** Latência e largura de banda moderadas, com perdas controladas.
- **4G:** Latência reduzida e maior largura de banda.
- **5G:** Largura de banda muito alta e baixa latência.
- **6G:** Latência mínima e largura de banda extremamente alta.
- **Fibra Óptica:** Conexão estável com a menor latência e alta largura de banda.

Os *presets* foram implementados no `network_control.py` e ativados por meio da interface web, possibilitando a simulação e replicação de condições reais de rede.

6.3 Teste com Manifesto DASH

Os experimentos foram realizados utilizando o manifesto disponível em `https://ftp.itec.aau.at/datasets/mmsys22/Eldorado/4sec/av1/manifest.mpd`. Durante os testes, os seguintes *presets* de rede foram aplicados na ordem:

1. **4G:** Inicialmente, simulou-se uma conexão com latência reduzida e maior largura de banda.
2. **3G:** Posteriormente, as condições de rede foram degradadas para uma latência e largura de banda moderadas.
3. **2G:** Em seguida, aplicou-se uma conexão com alta latência, baixa largura de banda e perda de pacotes significativa.
4. **5G:** Após o período de baixa qualidade, o sistema foi testado em condições de largura de banda muito alta e baixa latência.
5. **6G:** Condições de latência mínima e largura de banda extremamente alta foram simuladas.
6. **Fibra Óptica:** Por fim, utilizou-se uma conexão estável com a menor latência e alta largura de banda.

Este procedimento garantiu que o sistema fosse avaliado em condições variadas, replicando cenários reais de conectividade.

6.4 Coleta e Processamento de Dados

Durante os testes, métricas de desempenho foram coletadas continuamente, incluindo:

- Latência, largura de banda e perda de pacotes.
- QoE.
- Seleção e estado dos servidores.

Os dados capturados foram processados com o script `generate_graphs.py`, que gerou gráficos para análise visual do comportamento do sistema sob diferentes condições de rede.

6.5 Validação Experimental

Os experimentos foram conduzidos em um ambiente controlado, e o desempenho foi avaliado comparando dois métodos de *content steering*:

1. **Método Tradicional:** Baseado em métricas fixas como latência, largura de banda e perda de pacotes.
2. **Método Baseado em IA:** Utilizando aprendizado de máquina para prever a melhor opção de servidor.

Os resultados foram analisados em termos de latência, largura de banda, perda de pacotes e QoE, demonstrando a capacidade do sistema de se adaptar às condições de rede e otimizar a experiência do usuário.

6.6 Reprodutibilidade

Todos os scripts, códigos-fonte e instruções detalhadas para configuração e execução do sistema estão disponíveis no repositório oficial do projeto: `https://github.com/LucasLJG/Content-Steering-no-Edge-Cloud.git`. O repositório inclui:

- Máquina virtual (`.ova`) pré-configurada.
- Scripts de inicialização e automação.
- Dados de exemplo e logs gerados nos experimentos.

7 Resultados e Análise

A escolha do método de *content steering* é uma decisão fundamental no contexto de sistemas adaptativos para streaming de vídeo. No presente trabalho, dois métodos foram implementados e analisados: o método tradicional, baseado em métricas fixas de rede, e o método utilizando Inteligência Artificial (IA). Ambos possuem características, vantagens e desvantagens que são detalhadamente exploradas nesta seção.

7.1 Método Tradicional

O método tradicional baseia-se na avaliação de métricas de rede, como latência, largura de banda e perda de pacotes. Esses parâmetros são combinados com pesos predefinidos para determinar a prioridade dos servidores disponíveis. A lógica deste método é simples e direta: o servidor com as melhores condições de rede no momento é selecionado para atender a requisição.

Vantagens:

- **Baixa complexidade computacional:** A lógica de decisão envolve apenas cálculos diretos, resultando em baixa sobrecarga no sistema.

- **Estabilidade:** Como os pesos são fixos e as decisões são baseadas em regras estáticas, o método apresenta comportamento previsível e consistente.
- **Fácil implementação:** Sua simplicidade torna a integração com o sistema mais rápida e menos propensa a erros.

Desvantagens:

- **Limitada capacidade adaptativa:** Mudanças rápidas nas condições de rede podem não ser refletidas imediatamente, o que pode levar a decisões subótimas.
- **Falta de aprendizado:** O método não considera padrões históricos ou tendências nos dados, utilizando apenas o estado atual da rede.
- **Ineficiência em cenários complexos:** Em condições com múltiplos fatores interdependentes (como congestionamento de servidores), o método pode não capturar adequadamente a complexidade do problema.

7.2 Método com Inteligência Artificial (IA)

O método baseado em IA utiliza um modelo de aprendizado de máquina, especificamente um regressor *Random Forest*, para prever a Qualidade de Experiência (QoE) de cada servidor disponível. O modelo é treinado com dados históricos, que incluem métricas de rede (latência, largura de banda e perda de pacotes), uso de recursos dos servidores (CPU e memória) e informações de QoE obtidas em testes anteriores.

Vantagens:

- **Alta capacidade adaptativa:** O modelo pode se ajustar rapidamente às mudanças nas condições de rede, já que utiliza previsões em tempo real baseadas em dados atualizados.
- **Utilização de múltiplas variáveis:** O aprendizado de máquina permite considerar interações complexas entre múltiplas métricas, o que aumenta a precisão na seleção do servidor.
- **Evolução contínua:** O sistema pode ser reentreinado com novos dados, melhorando seu desempenho ao longo do tempo.

Desvantagens:

- **Complexidade computacional:** A execução de modelos de aprendizado de máquina requer maior capacidade de processamento, o que pode impactar o desempenho em sistemas com recursos limitados.
- **Dependência de dados:** O desempenho do modelo depende da qualidade e da representatividade dos dados utilizados para treinamento.
- **Curva de aprendizado:** O desenvolvimento e a integração de sistemas baseados em IA requerem conhecimento técnico especializado, aumentando o custo inicial de implementação.

7.3 Comparação Teórica entre os Métodos

Enquanto o método tradicional se destaca pela simplicidade e estabilidade, ele é limitado em sua capacidade de adaptação a cenários mais dinâmicos e complexos. Por outro lado, o método baseado em IA oferece maior precisão e flexibilidade, mas apresenta um custo computacional e técnico mais elevado.

Figura 8: Métodos de Steering

Em termos de desempenho, o método tradicional é mais adequado para cenários com condições de rede previsíveis e consistentes, onde alterações são raras e os requisitos de QoE são moderados. Em contrapartida, o método com IA se mostra mais eficaz em cenários com alta variabilidade, onde é crucial prever e responder rapidamente a mudanças na rede e no comportamento do usuário.

7.4 Resultados Obtidos

A análise dos resultados obtidos através dos experimentos realizados com o sistema de *content steering* demonstra o bom funcionamento da solução implementada em diferentes cenários de rede. Durante os testes, foram avaliados diversos aspectos do sistema utilizando diferentes presets de rede, que simulam condições reais desde conexões 2G até conexões de fibra óptica.

Os resultados são apresentados através de quatro gráficos principais que mostram o comportamento do sistema ao longo do tempo: largura de banda, latência, perda de pacotes e *QoE*. Os testes foram realizados em um período de aproximadamente 2 minutos e 15 segundos, durante o qual diferentes presets de rede foram aplicados para avaliar a adaptabilidade do sistema.

Analisando o gráfico de largura de banda (Figura 9), podemos observar as variações

conforme os diferentes presets foram aplicados. Inicialmente, com o preset "Good (4G)", a largura de banda se manteve estável em aproximadamente 25.000 kbit/s. Ao mudar para o preset "Average (3G)", observamos uma queda significativa para cerca de 1.000 kbit/s. O momento mais crítico ocorreu durante o preset "Poor (2G)", onde a largura de banda caiu para aproximadamente 500 kbit/s. Em contrapartida, quando aplicados os presets "5G", "6G"e "Excellent (Fiber)", a largura de banda aumentou expressivamente, atingindo valores próximos a 1.000.000 kbit/s, demonstrando a capacidade do sistema em se adaptar a condições de rede mais favoráveis.

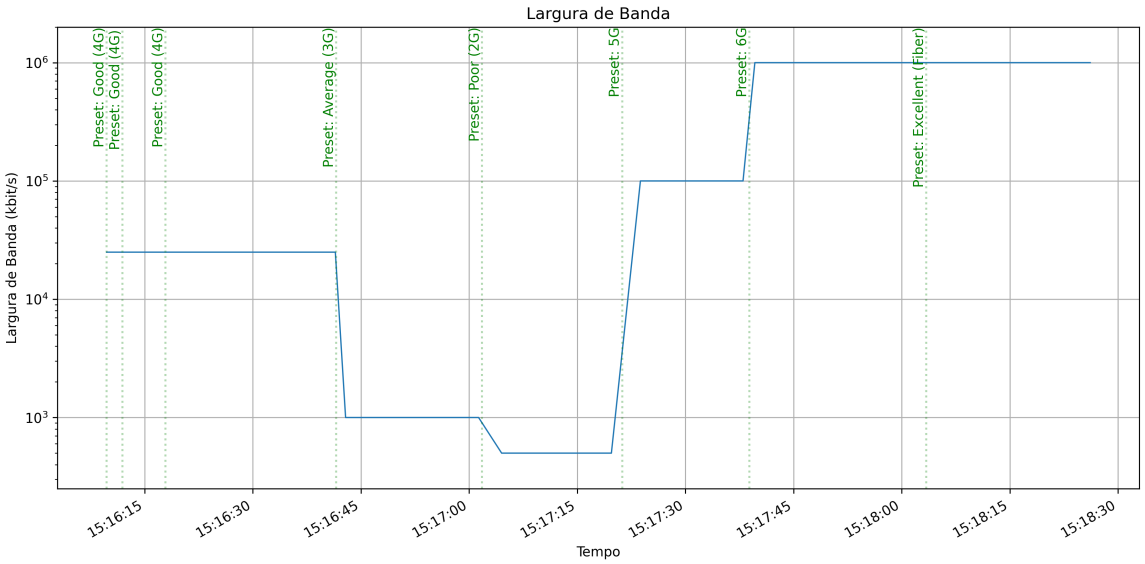

Figura 9: Largura de Banda

O gráfico de latência (Figura 10) revela um comportamento igualmente interessante. Com o preset "Good (4G)", a latência se manteve em torno de 35ms, um valor típico para redes 4G. Durante o preset "Average (3G)", houve um aumento para 100ms, e no preset "Poor (2G)", a latência atingiu seu pico em 250ms. É notável a melhoria significativa quando os presets de maior qualidade foram aplicados, com a latência caindo para valores próximos a 1ms nos presets "6G"e "Excellent (Fiber)", demonstrando como diferentes tecnologias de rede impactam diretamente o tempo de resposta do sistema.

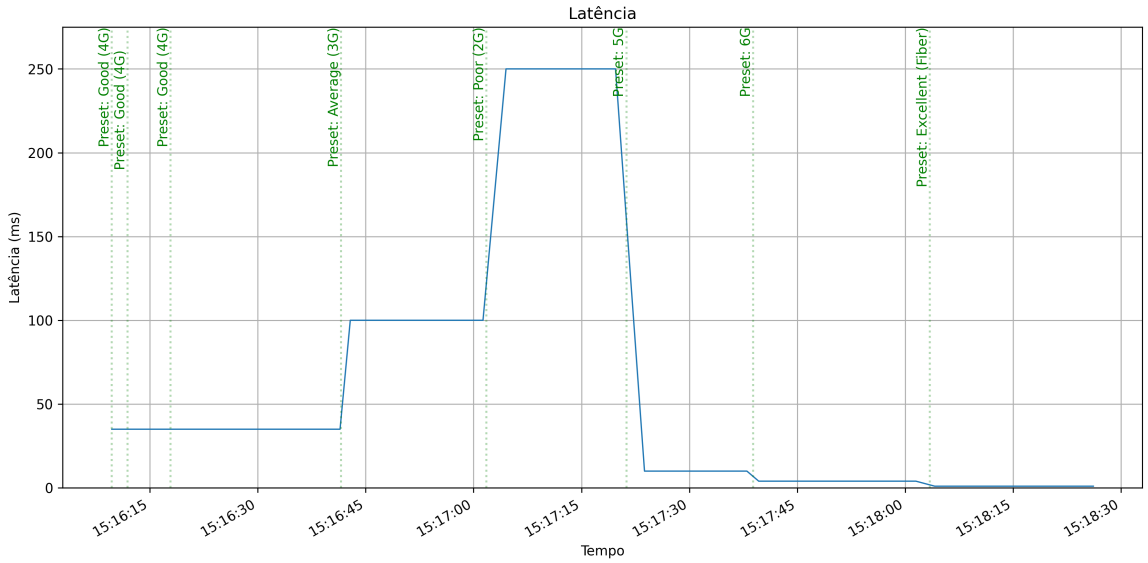

Figura 10: Latência

Quanto à perda de pacotes, o gráfico (Figura 11) mostra uma variação significativa entre os diferentes presets. Nos presets de melhor qualidade (4G, 5G, 6G e Fiber), a perda de pacotes se manteve em níveis muito baixos, próximos a 0.001%. No entanto, durante o preset "Poor (2G)", observamos um aumento considerável, chegando a aproximadamente 2% de perda de pacotes.

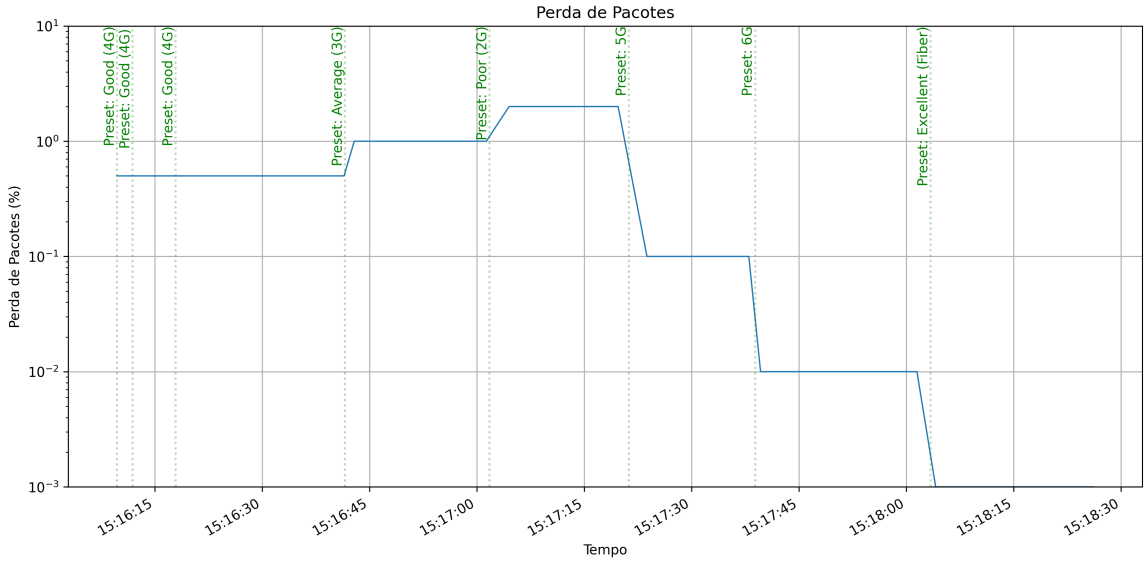

Figura 11: Perda de Pacotes

O gráfico de *QoE* (Figura 12) reflete o impacto combinado de todos os parâmetros

anteriores na experiência do usuário. Com o preset "Good (4G)", o QoE se manteve em um nível bom, próximo a 4.0. Durante a transição para presets de menor qualidade, observamos uma queda gradual no QoE, atingindo seu valor mais baixo (aproximadamente 1.0) durante o preset "Poor (2G)". É notável a recuperação rápida do QoE quando o sistema transitou para os presets de maior qualidade, chegando a valores próximos a 5.0 nos presets "6G"e "Excellent (Fiber)".

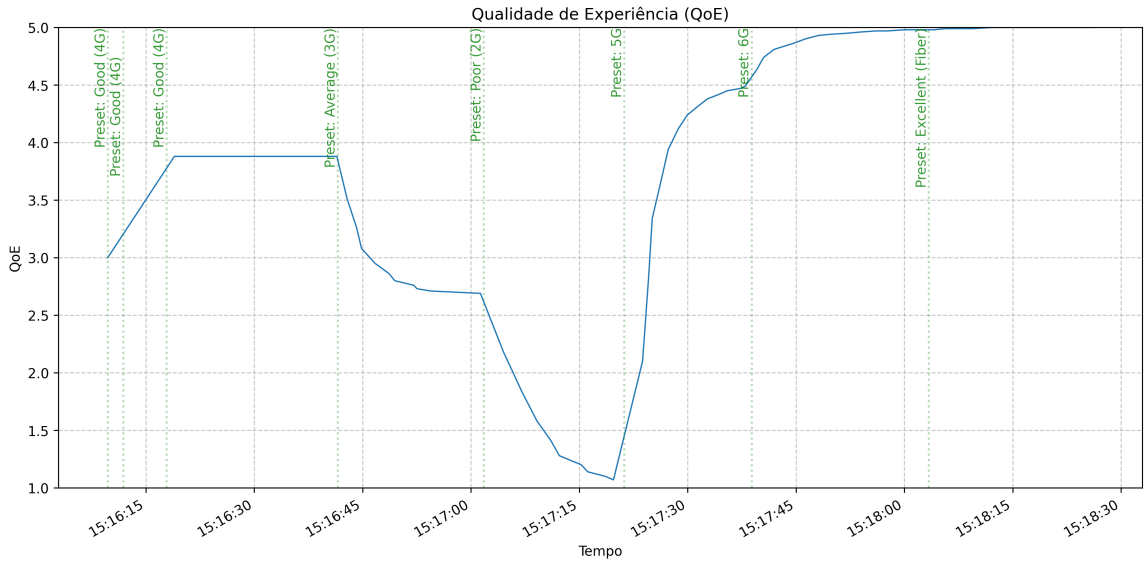

Figura 12: Qualidade de Experiência (QoE)

Estes resultados demonstram a capacidade do sistema em se adaptar a diferentes condições de rede, mantendo a melhor qualidade possível de streaming dentro das limitações impostas por cada cenário. A transição entre os diferentes presets evidencia como o sistema responde às mudanças nas condições de rede, ajustando automaticamente os parâmetros de streaming para otimizar a experiência do usuário.

8 Conclusão e Trabalhos Futuros

O desenvolvimento e implementação do sistema adaptativo de content steering para streaming de vídeo apresentou resultados significativos que demonstram sua eficácia em otimizar a entrega de conteúdo em diferentes cenários de rede.

Os resultados obtidos através dos experimentos realizados demonstraram a capacidade do sistema em adaptar-se dinamicamente a diferentes condições de rede, desde cenários altamente limitados como conexões 2G até ambientes de alta performance como redes de fibra óptica.

Um aspecto importante do projeto foi a implementação do sistema de *presets* de rede, que permitiu simular de forma realista diferentes condições de conectividade. Esta funcionalidade não apenas facilitou os testes e validação do sistema, mas também demonstrou como

o content steering pode se adaptar a mudanças bruscas nas condições de rede mantendo a melhor qualidade de experiência possível para o usuário.

O monitoramento das métricas de desempenho, incluindo latência, perda de pacotes e largura de banda, forneceu insights valiosos sobre o comportamento do sistema em diferentes cenários. A geração automática de gráficos e análise desses gráficos permite uma compreensão do impacto das diferentes condições de rede na qualidade do streaming, facilitando a identificação de pontos de melhoria e otimização.

Como pontos de melhorias para trabalhos futuros, sugere-se:

1. Desenvolvimento de algoritmos mais sofisticados de predição de qualidade de rede utilizando técnicas avançadas de machine learning
2. Implementação de mecanismos de cache mais inteligentes que considerem padrões de consumo de conteúdo
3. Expansão do sistema para suportar diferentes protocolos de streaming além do DASH
4. Implementação de mecanismos mais robustos de recuperação de falhas
5. Otimização do uso de recursos em ambientes com limitações de hardware

Este sistema de content steering adaptativo mostrou que é possível otimizar a entrega de streaming de vídeo em cenários de rede variados, garantindo uma experiência de qualidade para o usuário. Com a rápida evolução das tecnologias de rede e a crescente demanda por vídeos de alta qualidade, este projeto aponta caminhos promissores para continuar avançando no campo do streaming adaptativo, com foco em soluções que acompanhem as necessidades de um público cada vez mais exigente.

Referências

[1] R. Rodrigues-Filho, E. S. Gama, M. M. Assis, R. Immich, E. Madeira, L. F. Bittencourt. Content Steering: Leveraging the Computing Continuum to Support Adaptive Video Streaming. Universidade Estadual de Campinas e Universidade Federal de Santa Catarina, 2024.

[2] Reznik, Y., Cabrera, G., Silhavy, D., Pham, S., Giladi, A., Balk, A., Begen, A., & Law, W. (2024). "Content Steering: a Standard for Multi-CDN Streaming"

[3] Gama, E., Silva, L., & Curado, M. (2024). "Content Steering for Adaptive Video Streaming over the Edge-Cloud Continuum"

[4] Feng, Y., Shen, S., Xu, M., Zhang, C., Wang, X., Wang, X., Wang, W., & Leung, V. C. M. (2023). "A large-scale holistic measurement of crowdsourced edge cloud platform".