UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

PEDRO AFONSO CARRARO

UM ESTUDO COMPARATIVO DE DESEMPENHO ENTRE AS DIFERENTES VERSÕES DO PROTOCOLO HTTP

GUARAPUAVA

2025

PEDRO AFONSO CARRARO

UM ESTUDO COMPARATIVO DE DESEMPENHO ENTRE AS DIFERENTES VERSÕES DO PROTOCOLO HTTP

A comparative study of performance between the different HTTP protocol versions

Mongrafia de Trabalho de Conclusão de Curso de Graduação apresentado como requisito para obtenção do título de Tecnólogo em Tecnologia em Sistemas para Internet do Curso Superior de Tecnologia em Sistemas para Internet da Universidade Tecnológica Federal do Paraná.

Orientador: Prof. Dra. Sediane Carmem Lunardi Hernandes

Coorientador: Prof. Dr. Hermano Pereira

GUARAPUAVA

2025

PEDRO AFONSO CARRARO

UM ESTUDO COMPARATIVO DE DESEMPENHO ENTRE AS DIFERENTES VERSÕES DO PROTOCOLO HTTP

Mongrafia de Trabalho de Conclusão de Curso de Graduação apresentado como requisito para obtenção do título de Tecnólogo em Tecnologia em Sistemas para Internet do Curso Superior de Tecnologia em Sistemas para Internet da Universidade Tecnológica Federal do Paraná.

Data de aprovação: 26/novembro/2025

Sediane Carmem Lunardi Hernandes
Doutora
Universidade Tecnológica Federal do Paraná - Campus Guarapuava

---

Hermano Pereira
Doutor
Universidade Tecnológica Federal do Paraná - Campus Guarapuava

---

William Alberto Cruz Castaneda
Doutor
Universidade Tecnológica Federal do Paraná - Campus Guarapuava

---

Dênis Lucas Silva
Mestre
Universidade Tecnológica Federal do Paraná - Campus Guarapuava

GUARAPUAVA

2025

RESUMO

Este trabalho apresenta uma análise comparativa entre as versões do protocolo *Hyper Transfer Protocol* (HTTP): HTTP/1.1, HTTP/2 e HTTP/3, motivada pela necessidade crescente de compreender como esses protocolos se comportam em aplicações web modernas, especialmente diante de requisitos de desempenho e eficiência em diferentes condições de rede. O objetivo foi avaliar, de maneira sistemática, como cada versão responde a variações de latência, perda de pacotes e concorrência, identificando suas vantagens e limitações práticas. A metodologia consistiu na implementação de um ambiente padronizado com o servidor NGINX, no qual foram executados testes controlados e reais utilizando métricas como *throughput*, tempo total de transferência e tempo até o primeiro byte. Os experimentos foram conduzidos em três cenários distintos: Local, LAN e Nuvem, permitindo observar tanto o comportamento em redes ideais quanto em ambientes degradados e de uso real. Os resultados indicam que os protocolos apresentam características complementares: o HTTP/1.1 demonstra limitações estruturais, o HTTP/2 mantém desempenho estável sob alta carga e o HTTP/3 oferece benefícios significativos em cenários com latência e em cenários específicos. Conclui-se que a escolha da versão mais adequada depende do contexto de aplicação e do ambiente de operação, reforçando a importância de compreender as particularidades de cada protocolo.

Palavras-chave: protocolo; http; desempenho; redes; latência.

ABSTRACT

This work presents a comparative analysis of HTTP protocol: HTTP/1.1, HTTP/2, and HTTP/3, motivated by the growing need to understand how these protocols behave in modern web applications, especially under increasing demands for performance and efficiency in different network conditions. The objective is to systematically evaluate how each version responds to variations in latency, packet loss, and concurrency, identifying their practical advantages and limitations. The methodology consists of implementing a standardized environment using the NGINX server, in which controlled and real experiments were conducted employing metrics such as throughput, total transfer time, and time to first byte. The experiments were carried out in three distinct scenarios—Local, LAN, and Cloud, allowing the observation of protocol behavior both in ideal networks and in degraded or real-world environments. The results indicate that the protocols exhibit complementary characteristics: HTTP/1.1 shows structural limitations, HTTP/2 maintains stable performance under high load, and HTTP/3 provides significant benefits in scenarios with latency and in specific conditions. It is concluded that the most appropriate version depends on the application context and the operating environment, highlighting the importance of understanding the particularities of each protocol.

Keywords: protocol; http; performance; networks; latency.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

Abreviaturas

art.	Artigo
cap.	Capítulo
sec.	Seção

Siglas

ABNT	Associação Brasileira de Normas Técnicas
ARPANET	Advanced Research Projects Agency Network
CNPq	Conselho Nacional de Desenvolvimento Científico e Tecnológico
EPS	*Encapsulated PostScript*
HTTP	Protocolo de Transferência de Hipertexto, do inglês *Hypertext Transfer Protocol*
IETF	Internet Engineering Task Force
IP	Protocolo de Internet, do inglês *Internet Protocol*
KB	*Kilo Byte*
PDF	Formato de Documento Portátil, do inglês *Portable Document Format*
PS	*PostScript*
QUIC	Conexão Rápida UDP de Internet, do inglês *Quick UDP Internet Connections*
TCP	Protocolo de Controle de Transmissão, do inglês *Transmission Control Protocol*
TTFB	*Time To First Byte*
UDP	Protocolo de datagrama do usuário, do inglês *User Datagram Protocol*
UTFPR	Universidade Tecnológica Federal do Paraná

SUMÁRIO

1 INTRODUÇÃO

Embora muitas vezes tratada como uma inovação recente e disruptiva, a Internet é, em certo sentido, a materialização técnica de ideias muito mais antigas. A noção de uma rede interligando inteligências humanas remonta a séculos anteriores à invenção dos computadores. No século XVII, por exemplo, Gottfried Wilhelm Leibniz idealizou uma máquina capaz de organizar todo o conhecimento humano por meio de uma linguagem universal. Athanasius Kircher, por sua vez, concebia sistemas que conectassem mentes por meio de tubos e sons — visões que, embora rudimentares, antecipavam a ideia de uma comunicação global automatizada. A Internet é fruto de uma longa tradição de sonhos e especulações sobre como expandir a capacidade da mente humana por meio de redes artificiais (SMITH, 2022).

Com o avanço tecnológico e a digitalização da sociedade, essas ideias escalaram e tomaram forma. Atualmente, a Internet é uma infraestrutura essencial para o funcionamento da vida contemporânea. Ela está presente em praticamente todos os aspectos do cotidiano, sendo alguns deles a comunicação pessoal, os serviços bancários, a educação, o trabalho remoto, o comércio eletrônico, o lazer e o acesso a informações. Segundo o Pew Research Center (2024), mais de 90% dos adultos em países desenvolvidos utilizam a Internet diariamente. No Brasil, a pesquisa TIC Domicílios do Cetic.br (2024) aponta que 84% dos lares têm acesso à rede, sendo o celular o principal meio de conexão. Essa presença definitiva e constante revela uma dependência crescente da sociedade em relação às tecnologias digitais e à infraestrutura que as sustenta.

O crescimento da Internet veio acompanhado de diversos desafios relacionados ao seu desempenho e à confiabilidade. Problemas como lentidão no carregamento de páginas, interrupções em serviços online, congestionamento de tráfego e alto consumo de dados impactam diretamente a experiência do usuário e a eficiência de sistemas que dependem da rede. Tais falhas técnicas não são triviais; elas afetam desde a produtividade no ambiente de trabalho até a qualidade do ensino remoto, o funcionamento de serviços públicos digitais e a competitividade de negócios on-line. Isso evidencia a importância de compreender não apenas as estruturas subjacentes que sustentam o funcionamento da Internet, mas também os protocolos de alto nível.

Dentre os diversos protocolos existentes, destaca-se o *Hyper Transfer Protocol* (HTTP), responsável por definir as regras para a comunicação entre servidores e clientes (i.e., navegadores) na *World Wide Web* (Web) (FOROUZAN; MOSHARRAF, 2013).[1] Mesmo sendo invisível para a maior parte dos usuários, o HTTP é um dos pilares da Web, sendo responsável por 82% do tráfego da Internet (CLOUDFLARE, 2024a). Nesse contexto, é compreensível que sua eficiência esteja diretamente ligada aos problemas citados anteriormente. Com isso, foram de-

[1] A Web é um serviço cliente-servidor distribuído que permite a recuperação de documentos por meio de páginas Web (FOROUZAN; FEGAN, 2009).

senvolvidas diferentes versões do protocolo, buscando superar limitações técnicas e oferecer mais velocidade, segurança e confiabilidade na comunicação.

Por fim, este trabalho teve como objetivo confrontar o desempenho das diferentes versões do HTTP, com ênfase nas versões HTTP/1.1, HTTP/2 e HTTP/3. Buscou-se, através de testes, identificar as vantagens e limitações de cada versão em diferentes contextos de uso, quanto ao carregamento de páginas em redes com estabilidade distinta, verificando o tempo de resposta e o consumo de dados. Com isso, pretendeu-se fornecer subsídios para profissionais e para o meio acadêmico da área na tomada de decisões mais coerentes sobre a adoção de cada versão conforme as necessidades específicas de seus projetos, sistemas ou aplicações.

1.1 Objetivos

1.1.1 Objetivo geral

Analisar o desempenho das versões HTTP/1.1, HTTP/2 e HTTP/3 do protocolo HTTP.

1.1.2 Objetivos específicos

Os objetivos específicos deste trabalho foram:

- Analisar as especificações técnicas das versões HTTP/1.1, HTTP/2 e HTTP/3, com base nas respectivas *Request for Comments* (RFCs), destacando suas principais características, inovações introduzidas e limitações observadas em cada uma.
- Examinar a complexidade de configuração das diferentes versões do protocolo HTTP em um ambiente real, utilizando o servidor web NGINX, a fim de compreender os requisitos práticos para sua adoção.
- Avaliar o comportamento das versões HTTP/1.1, HTTP/2 e HTTP/3 quanto ao desempenho, considerando métricas como o tempo de resposta, por exemplo.

1.2 Justificativa

O protocolo HTTP, utilizado na Web, é responsável por grande parte do tráfego global da Internet. Essa grande dominância acaba subdividida entre suas versões, sendo que em 2024 apenas 21% do tráfego foi representado pelo HTTP/3, o tráfego HTTP/2 correspondeu a aproximadamente 50% e as versões HTTP/1.0 e HTTP/1.1 equivaleram a cerca de 30% do tráfego (CLOUDFLARE, 2024b).

Essa diferença na distribuição do tráfego do protocolo pode ser explicada pelo fato do HTTP/3 ser relativamente recente e em processo de adoção. Isso resulta em menos documen-

tação e experiência consolidada sobre seu uso em diferentes contextos, causando certa incerteza de sua eficiência. Paralelamente, versões anteriores continuam sendo usadas porque, em muitos casos, ainda atendem de forma satisfatória às necessidades, além de não exigirem modificações nas aplicações e servidores já existentes.

Nesse contexto, torna-se relevante analisar e comparar o desempenho dessas diferentes versões, levando em conta que cada uma pode apresentar vantagens ou desvantagens dependendo do ambiente em que se encontra. A velocidade de carregamento de um site, por exemplo, é um dos principais fatores que impactam a usabilidade e a percepção de qualidade por parte do usuário. Segundo Nielsen (2010), atrasos acima de 1 segundo já interrompem o fluxo de pensamento do usuário e acima de 10 segundos geram frustração e abandono. Dados mais recentes também mostram que essa tolerância tem diminuído. Uma pesquisa citada pela Dycoders (2023) aponta que 47% dos usuários esperam que a página carregue em até 2 segundos e 40% abandonam o site se levar mais de 3 segundos. Esses fatores contribuem diretamente para altas taxas de rejeição, frustração do usuário e perda de oportunidades de negócio. Em contrapartida, um site rápido e com bom desempenho pode melhorar a satisfação do usuário. Essa comparação pode auxiliar profissionais e instituições a tomarem decisões mais embasadas quanto à adoção ou manutenção de determinadas versões do protocolo, conforme suas demandas específicas.

Assim, a finalidade deste trabalho provém da necessidade de uma melhor compreensão do comportamento das versões do protocolo HTTP na prática, contribuindo para um uso mais consciente e adequado de acordo com cada situação. Os resultados deste estudo podem beneficiar tanto a comunidade acadêmica, ao ampliar o conhecimento técnico sobre o tema, quanto o setor de tecnologia, ao apresentar um estudo prático do protocolo e seu desempenho. O estudo busca contribuir para um campo ainda em consolidação na literatura técnica e acadêmica, especialmente no que se refere à comparação prática entre as versões do protocolo HTTP e sua configuração em ambientes reais. Embora existam documentos normativos, como as RFCs, e relatórios técnicos de desempenho, há uma escassez de trabalhos que integrem essas abordagens de forma aplicada, o que reforça o potencial deste estudo como base para futuras pesquisas e aplicações.

Por fim, a escolha do tema também reflete um interesse pessoal do autor na área de redes e protocolos, reforçado por experiências acadêmicas relacionadas à infraestrutura da Web. A viabilidade do estudo foi garantida pelo acesso a ambientes de teste e ferramentas adequadas, os quais permitiram a realização de análises práticas e objetivas de desempenho das diferentes versões do protocolo HTTP.

1.3 Estrutura do trabalho

O trabalho é estruturado como segue. O Capítulo 1 apresenta a introdução do trabalho, os objetivos e a justificativa. No Capítulo 2, o referencial teórico do trabalho é descrito. No Capí-

tulo 3, os materiais e métodos utilizados durante o desenvolvimento do trabalho são detalhados. Os resultados alcançados são apresentados no Capítulo 4, que é organizado de modo a responder a cada objetivo específico definido no Capítulo 1. As considerações finais são descritas no Capítulo 5 e, por fim, as referências utilizadas são especificadas.

2 REFERENCIAL TEÓRICO

A Web teve origem como uma plataforma voltada à disseminação de informações científicas, mas rapidamente evoluiu para uma das principais estruturas de comunicação, comércio e entretenimento do mundo contemporâneo. Essa transformação foi possível graças à arquitetura em camadas da Internet, que permite a interoperabilidade entre dispositivos e aplicações distribuídas globalmente (FOROUZAN; FEGAN, 2009).

No centro dessa arquitetura estão os protocolos fundamentais Internet Protocol (IP) e Transmission Control Protocol (TCP). O IP é encarregado do endereçamento e do roteamento dos pacotes de dados, garantindo que as informações cheguem ao seu destino correto. Já o TCP provê um serviço de transporte confiável, orientado à conexão, com controle de fluxo e mecanismos de retransmissão para lidar com perdas de pacotes (FOROUZAN; FEGAN, 2009).

Esses protocolos formam a base da chamada pilha TCP/IP, composta por camadas como a de enlace, de rede, de transporte e de aplicação. Na camada de transporte, além do TCP, destaca-se o User Datagram Protocol (UDP), um protocolo mais simples, sem verificação de estado e que não exige o estabelecimento de conexão entre emissor e receptor. O UDP é amplamente utilizado em aplicações sensíveis à latência, como chamadas de voz e transmissões ao vivo, justamente por evitar os atrasos inerentes ao controle de confiabilidade do TCP (FOROUZAN; FEGAN, 2009).

A camada de aplicação, por sua vez, é onde atua o protocolo *Hypertext Transfer Protocol - HTTP*, responsável por estruturar a comunicação entre navegadores e servidores. Ao longo do tempo, o HTTP passou por diferentes versões, cada uma buscando resolver limitações da anterior no que tange à performance, paralelismo e segurança. A seção a seguir detalha o protocolo HTTP e as principais características das versões HTTP/1.1, HTTP/2 e HTTP/3, com base nas respectivas especificações publicadas pela Internet Engineering Task Force (IETF) (FIELDING; NOTTINGHAM; RESCHKE, 2022; BELSHE; PEON; THOMSON, 2015; BISHOP, 2022).

2.1 O protocolo HTTP

O protocolo HTTP é um protocolo da camada de aplicação do tipo requisição/resposta que opera sobre um protocolo de transporte confiável, o TCP, sendo utilizado para definir como os programas cliente e servidor podem ser implementados para recuperar páginas web (FIELDING; NOTTINGHAM; RESCHKE, 2022). Logo, um cliente HTTP é um programa que estabelece uma conexão com um servidor com o objetivo de enviar uma ou mais requisições HTTP, enquanto um servidor HTTP é um programa que aceita conexões com o propósito de aceitar requisições HTTP e enviar respostas HTTP (FIELDING; NOTTINGHAM; RESCHKE, 2022).

2.2 HTTP/1.1

A versão HTTP/1.1, definida em 1999 e padronizada em 2022 (Standard 94 - STD 94 (FIELDING; NOTTINGHAM; RESCHKE, 2022)), é amplamente utilizada nos dias atuais. Essa versão introduziu melhorias em relação ao HTTP/1.0, como (FIELDING; NOTTINGHAM; RESCHKE, 2022):

- mecanismos adicional de cache, o qual é especificado no cabeçalho HTTP por meio dos campos: *cache control* - especifica diretivas para mecanismos de cache tanto em requisições quanto em respostas; *e-tag* - determinar se a cópia de um recurso em cache de um cliente está atualizada; e, *expires* – indica a data e a hora em que uma resposta é considerada velha).
- conexões persistentes (*keep-alive*): o servidor deixa a conexão aberta para outras solicitações após o envio de uma resposta e pode fechar a conexão a pedido de um cliente ou se um tempo limite for atingido.
- *pipelining* de solicitação, em que múltiplas solicitações HTTP são enviadas em uma única conexão TCP.
- codificação de transferência (*TransferEncoding*) especifica a forma de codificação usada para transferir seguramente o corpo da mensagem, permitindo compressão de dados.

No entanto, apresenta limitações significativas quanto à multiplexação: por padrão, apenas uma requisição pode ser processada por conexão TCP. Isso leva ao fenômeno conhecido como *head-of-line blocking* - *HOL-Blocking* (em português, o bloqueio de cabeçote de linha). O *HOL-Blocking* acontece quando pacotes de uma fila são retidos pelo primeiro pacote, que se perdido ou sofrer uma condição de erro, precisa ser retransmitido. Assim, somente depois desse pacote ser retransmitido, os outros podem ser enviados. Neste caso, o primeiro pacote impede o envio dos próximos pela mesma conexão impactando no desempenho da rede e na latência.

Para contornar essa limitação, os navegadores passaram a abrir múltiplas conexões paralelas para o mesmo servidor, frequentemente de quatro a seis por domínio, a fim de possibilitar o carregamento simultâneo de recursos como *scripts*, imagens e folhas de estilo (KUROSE; ROSS, 2021). No entanto, esse modelo aumenta o consumo de recursos tanto no cliente quanto no servidor, além de gerar maior sobrecarga de gerenciamento de conexões. Essas restrições motivaram o desenvolvimento de versões mais eficientes do protocolo, como o HTTP/2 e o HTTP/3.

2.3 HTTP/2

O HTTP/2, definido em 2015 pela IETF, representa uma evolução significativa do HTTP/1.1. Ele introduz um modelo binário de comunicação e permite a multiplexação de múltiplos *streams*[1] em uma única conexão TCP, eliminando o *head-of-line blocking* a nível de aplicação. Além disso, traz compressão de cabeçalho e permite ao servidor realizar *server push*, antecipando respostas antes mesmo que o cliente as solicite. O *server push* é um mecanismo pelo qual o servidor pode enviar proativamente recursos ao cliente, como scripts ou folhas de estilo, assim que detecta que eles provavelmente serão necessários, sem esperar por requisições explícitas. Isso visa reduzir o tempo de carregamento de páginas web (BELSHE; PEON; THOMSON, 2015).

Apesar dos avanços, o HTTP/2 ainda herda algumas limitações do TCP. Como o protocolo TCP é orientado à conexão, qualquer perda de pacote em uma conexão HTTP/2 pode impactar todos os *streams* multiplexados, pois o TCP exige a entrega ordenada dos dados. Esse problema é especialmente visível em redes instáveis, como as móveis (BELSHE; PEON; THOMSON, 2015; KUROSE; ROSS, 2021).

2.4 HTTP/3

O HTTP/3 surge como uma resposta às limitações estruturais do HTTP/2 ao transferir sua camada de transporte do TCP para o protocolo Quick UDP Internet Connections (QUIC), desenvolvido pelo Google e especificado por meio da RFC 9000 (IYENGAR; THOMSON, 2021b), o qual é implementado sobre o UDP. O QUIC é um protocolo orientado a conexão que cria uma interação sem estado (*stateful*) entre um cliente e um servidor (IYENGAR; THOMSON, 2021b).

Uma das principais inovações do QUIC é a integração nativa com o Transport Layer Security (TLS)1.3, permitindo que a negociação criptográfica e o estabelecimento de conexão ocorram simultaneamente, em um único *handshake*. Essa abordagem reduz significativamente a latência, especialmente em conexões seguras, ao agilizar o tempo até a primeira resposta do servidor *(Time to First Byte – TTFB)*.

Além disso, o QUIC oferece suporte à multiplexação de múltiplos *streams* independentes dentro de uma mesma conexão. Diferentemente do TCP, que impõe um único fluxo ordenado de *bytes*, o QUIC permite que cada *stream* opere de forma isolada. Isso elimina o *head-of-line blocking* na camada de transporte. Assim, mesmo um *stream* precisando ser reenviado, o envio dos outros *streams* acontece normalmente, não bloqueando os demais. Logo, a perda de pacotes em um *stream* específico não impacta o processamento dos demais, promovendo maior robustez e desempenho em redes com perdas ou alta variação de latência.

Entre suas principais vantagens, o QUIC oferece:

[1] *Stream* é um fluxo bidirecional de frames (a menor unidade de comunicação dentro de uma conexão HTTP/2) dentro de uma conexão HTTP/2

- **Redução na latência de conexão**: combina o *handshake* criptográfico do TLS com o estabelecimento da conexão de transporte, eliminando rodadas adicionais de comunicação;
- **Multiplexação nativa e eficiente**: os *streams* são independentes, de forma que a perda de pacotes em um *stream* não afeta os demais;
- **Migração de conexão**: permite que uma sessão seja retomada após mudanças de IP, característica útil em dispositivos móveis;
- **Segurança integrada**: ao contrário do TCP, que depende de camadas adicionais para criptografia (como o TLS), o QUIC incorpora segurança por padrão.

Com essas características, o HTTP/3 busca atender às exigências de desempenho e confiabilidade da Web moderna, especialmente em dispositivos móveis e redes com maior variabilidade. Estudos recentes apontam ganhos relevantes de desempenho em diversas implementações e contextos (TREVISAN *et al.*, 2021; LIU *et al.*, 2024; BALEJ; SOCHOR, 2023).

As diferenças arquiteturais discutidas nesta seção, especialmente a transição do TCP para o QUIC e a eliminação do Head-of-Line Blocking, resultam em comportamentos distintos de performance e configuração. Para uma visão consolidada das especificações técnicas e das diferenças estruturais entre as três versões abordadas, consulte a Tabela 2, apresentada na Seção 4.1.1.

2.5 Trabalhos relacionados

Diversos trabalhos recentes têm se dedicado à análise do protocolo HTTP, com foco em aspectos específicos como desempenho, adoção e impacto em diferentes ambientes. Trevisan *et al.* (2021), por exemplo, realizaram uma análise em larga escala da adoção e performance do HTTP/3 em domínios populares, revelando melhorias significativas na latência e no tempo de resposta, especialmente em cenários com alta perda de pacotes. No entanto, o estudo limita-se à versão HTTP/3, sem uma comparação prática com versões anteriores, nem abordagem sobre a configuração de servidores web em ambientes locais.

Já Liu *et al.* (2024) exploram o comportamento do HTTP/2 e HTTP/3 em ambientes com e sem proxy, investigando o impacto de *middleboxes* e CDNs (*Content Delivery Networks*) sobre a eficiência do protocolo. Embora o estudo aprofunde-se nas interações entre protocolos modernos e infraestrutura de rede, ele não contempla o HTTP/1.1 e sua análise se concentra em ambiente de produção com componentes adicionais de rede.

Outro estudo relevante é o de Balej e Sochor (2023), que compara diferentes implementações de servidores HTTP/3, como Caddy, OpenLiteSpeed e NGINX. O foco está na forma como cada servidor lida com a nova versão do protocolo, mas não há comparação direta entre as versões HTTP/1.1, HTTP/2 e HTTP/3 em um mesmo ambiente.

Em contraste, este projeto de Trabalho de Conclusão de Curso propõe uma análise comparativa entre as versões HTTP/1.1 (FIELDING; NOTTINGHAM; RESCHKE, 2022), HTTP/2 (BELSHE; PEON; THOMSON, 2015) e HTTP/3 (BISHOP, 2022), com foco tanto nas especificações técnicas apresentadas pelas RFCs quanto no comportamento prático das versões em um mesmo servidor (NGINX). Além disso, busca-se examinar a complexidade de configuração, o impacto da performance no uso real da Web e identificar quais versões oferecem maior eficiência em diferentes cenários.

Assim, compreender essas três versões, seus mecanismos internos, vantagens e limitações é essencial para avaliar o impacto do HTTP na experiência do usuário, no desempenho da Web e nas decisões técnicas adotadas pelas organizações. Essa análise se torna ainda mais relevante diante de um cenário em que cada milissegundo de atraso pode significar perda de receita, engajamento ou confiabilidade.

3 MATERIAIS E MÉTODOS

Esta seção descreve os materiais que foram utilizados para o desenvolvimento do trabalho e a metodologia adotada para a realização dos experimentos propostos neste estudo. O objetivo foi conduzir uma análise comparativa entre as versões HTTP/1.1, HTTP/2 e HTTP/3, observando aspectos de desempenho, comportamento de rede e impacto na infraestrutura.

3.1 Documentação teórica

As especificações técnicas de cada versão do protocolo foram obtidas diretamente dos documentos normativos publicados pela IETF:

- RFC 9110 – HTTP/1.1 Semantics (FIELDING; NOTTINGHAM; RESCHKE, 2022)
- RFC 7540 – HTTP/2 (BELSHE; PEON; THOMSON, 2015)
- RFC 9114 – HTTP/3 (BISHOP, 2022)
- RFC 9000 - QUIC (IYENGAR; THOMSON, 2021b)

Esses documentos serviram não somente como base conceitual e técnica para a implementação e análise dos testes, mas também para o estudo técnico empregado sobre cada versão e para definição de alguns parâmetros de teste.

3.2 Servidor

O servidor web utilizado nos testes foi o NGINX, em sua versão 1.28.0, com suporte habilitado para HTTP/1.1, HTTP/2 e HTTP/3. O NGINX foi escolhido por sua ampla adoção na indústria, flexibilidade de configuração e suporte ativo às últimas versões do protocolo. Além disso, segundo Balej e Sochor (2023), entre diferentes servidores com suporte a HTTP/3, o NGINX apresentou o melhor desempenho geral nos testes realizados, destacando-se em métricas como estabilidade, tempo de resposta e uso de recursos.

A configuração de servidores web foi diretamente impactada pela evolução dos protocolos HTTP. No caso do HTTP/1.1, a configuração típica envolve a ativação do suporte a SSL/TLS na porta 443, com a definição dos caminhos do certificado digital e da chave privada do servidor. Além disso, recomenda-se restringir o uso do TLS às versões mais modernas e seguras, conforme orientado pela documentação do NGINX (NGINX, Inc., 2024), bem como selecionar cifras robustas para garantir a confidencialidade e a integridade da comunicação.

Com a adoção do HTTP/2, o processo de configuração no NGINX mantém praticamente a mesma estrutura da configuração do HTTP/1.1, exigindo apenas a inclusão da diretiva `http2 on;` no arquivo de configuração do servidor, juntamente com a diretiva `listen 443 ssl;`.

Essa característica demonstra a compatibilidade incremental do protocolo, permitindo que aplicações migrem do HTTP/1.1 para o HTTP/2 sem mudanças estruturais significativas no servidor, mas obtendo melhorias em paralelismo, compressão de cabeçalhos e eficiência no uso da conexão (NGINX, INC., 2024a).

Já para a configuração do HTTP/3 no NGINX, tem-se um cenário distinto. Por utilizar o protocolo QUIC, sua implementação requer a compilação personalizada do NGINX com bibliotecas adicionais, como o *quicTLS* (PROJECT, 2024). Além disso, o suporte é restrito ao TLS 1.3, utilizando cifras específicas como *AES-GCM* e *ChaCha20-Poly1305*. Outras diretivas adicionais incluem o uso de `listen 443 quic reuseport` para habilitar o QUIC, bem como o envio de cabeçalhos *Alt-Svc* para anunciar a disponibilidade do HTTP/3 aos navegadores. Esses aspectos evidenciam que a transição para o HTTP/3 vai além da simples alteração de parâmetros, representando um salto arquitetural com impactos diretos na segurança, desempenho e compatibilidade (NGINX, INC., 2024b; BISHOP, 2022).

3.3 Ferramentas de análise

Para os testes realizados no trabalho foram necessárias algumas ferramentas, dentre elas:

1. **H2Load**: Ferramenta de *benchmark* desenvolvida como parte do projeto nghttp2, utilizada para testar o desempenho de servidores configurados para HTTP/2, HTTP/3 e também HTTP/1.1. Permite simular múltiplas conexões e requisições concorrentes, além de diferentes requisições por segundo. Fornece métricas como tempo de resposta e taxa de transferência. É útil para avaliar a eficiência e escalabilidade de implementações modernas do protocolo HTTP (nghttp2.org, 2025).

2. **TC (Traffic Control)**: Ferramenta do Linux que permite manipular o tráfego de rede, simulando condições como latência, perda de pacotes e limitação de banda. A ferramenta é frequentemente utilizada para criar cenários de rede adversos em ambientes de teste, permitindo verificar o comportamento de protocolos como TCP, UDP e QUIC em situações realistas de instabilidade ou congestionamento.

3. **Multipass**: Ferramenta de virtualização desenvolvida pela Canonical para a emulação de máquinas virtuais Ubuntu em diferentes sistemas operacionais. O multipass é frequentemente utilizado para testes e desenvolvimento local, proporcionando uma maneira simplificada de configuração e remoção de ambientes (Canonical, 2025).

3.4 Métodos

A metodologia adotada neste trabalho combinou abordagem experimental com revisão bibliográfica, de modo a proporcionar uma compreensão abrangente sobre a evolução do protocolo HTTP. A revisão teórica fundamenta-se na análise das principais RFCs de cada versão do protocolo, bem como em artigos científicos recentes que investigam aspectos técnicos e práticos relacionados ao desempenho, adoção e implementação do HTTP/1.1, HTTP/2 e HTTP/3. Essa etapa serviu de base para definir os critérios de comparação e estruturar os testes experimentais.

Complementando a análise teórica, foram realizados experimentos controlados e replicáveis em diferentes ambientes de rede e sob múltiplas condições de operação. Os ambientes definidos para os experimentos foram:

- **Local (Multipass):** Cliente e servidor simulados via maquinas virtuais Multipass na mesma máquina. Ambiente de latência mínima, ideal para observar o comportamento puro dos protocolos.
- **Rede LAN:** Cliente e servidor em máquinas distintas conectadas via *switch* ou roteador. Permite simular uma rede interna com condições mais realistas e controle sobre os dispositivos de rede.
- **Nuvem (provedor externo):** O servidor foi hospedado em um ambiente de nuvem pública da DigitalOcean[1]. Esse cenário simula um ambiente real de Internet, com latência variável e fatores externos de rede.

Para cada versão do protocolo, os testes foram executados nas mesmas condições de hardware, rede e carga, garantindo isonomia nos resultados. Desta forma, foram montados diferentes cenários de comunicação entre cliente(s) e servidor, considerando os três ambientes principais definidos: **local (Multipass)**, **rede interna (LAN)** e **nuvem (externo)**. Cabe salientar que em cada cenário foram testadas as versões HTTP/1.1, HTTP/2 e HTTP/3. A quantidade de clientes foi variável entre um único cliente e múltiplos clientes simultâneos. A Tabela 1 apresenta os cenários descritos.

Tabela 1 – Cenários definidos para os testes.

Cenário	Ambiente	Topologia	Protocolos	Sessões
1a	Local (Multipass)	1 cliente $\rightarrow$ 1 servidor	HTTP/1.1, HTTP/2, HTTP/3	1 ... n
2a	Local (Multipass)	n clientes $\rightarrow$ 1 servidor	HTTP/1.1, HTTP/2, HTTP/3	1 ... n
1b	Rede LAN	1 cliente $\rightarrow$ 1 servidor	HTTP/1.1, HTTP/2, HTTP/3	1 ... n
2b	Rede LAN	n clientes $\rightarrow$ 1 servidor	HTTP/1.1, HTTP/2, HTTP/3	1 ... n
1c	Nuvem (externo)	1 cliente $\rightarrow$ 1 servidor	HTTP/1.1, HTTP/2, HTTP/3	1 ... n
2c	Nuvem (externo)	n clientes $\rightarrow$ 1 servidor	HTTP/1.1, HTTP/2, HTTP/3	1 ... n

[1] https://www.digitalocean.com

Para a execução dos cenários descritos acima, foram definidos parâmetros para a configuração do servidor e do cliente, a carga de trabalho do servidor utilizando o número de conexões concorrentes e *streams* simultâneos, além das condições de rede.

3.4.1 Parâmetros Experimentais

Foram definidos dois cenários de configuração para o servidor (NGINX) e para o cliente (h2load). O cliente permaneceu configurado em uma máquina virtual configurada como descrito abaixo, presente no mesmo computador em todos os cenários. O servidor por sua vez, foi instalado em diferentes computadores com as configurações descritas, respeitando a topologia específica de cada cenário de teste.

- **Cliente:** Foi utilizada para o computador cliente uma configuração com 4 processadores (i.e., CPUs) e uma memória principal (i.e., memória RAM) com 4GB, com uma variação entre os cenários: uma conexão e múltiplas conexões simultâneas devido a uma limitação da ferramenta utilizada nos testes. O h2load não permite a utilização de um número de núcleos maior que o número de conexões. Logo, a configuração efetiva utilizada pela ferramenta nos cenários 1a, 1b e 1c foi de 1 processador e uma memória principal com 4GB para o computador cliente.
- **Servidor:** Para a máquina servidor foi utilizada uma configuração com 4 processadores e uma memória principal com 8GB em todos os cenários.

Visando simular o carregamento de páginas com diferentes graus de complexidade e normalizar a variância de peso de cada componente presente em páginas modernas (imagens, arquivos de estilização, vídeos, entre outros), foram utilizados os percentis 25, 50 e 75 de peso em *Kilo Byte* (KB) e número de requisições de páginas para computadores do dia 1 de outubro de 2025, presentes no relatório *Page Weight* do Archive (2025). Nesse contexto, para o tamanho do *payload* utilizado em cada requisição foi calculada a média dos valores de cada percentil, dividindo o peso da página em KB pelo número de requisições, resultando nos seguintes valores:

- **Página pesada:** Simula o percentil 75 dos dados com 124 requisições de 47KB por conexão. Para o escopo deste trabalho, essa configuração foi definida como uma página pesada.
- **Página média:** Simula o percentil 50, isto é, a mediana dos dados com 76 requisições de 38KB por conexão. Tratando-se da mediana dos dados, essa foi definida como uma página média.
- **Página leve:** Simula o percentil 25 com 43 requisições de 34KB por conexão. Essa foi definida como uma página leve de acordo com os outros valores acima.

Para cada cenário (Tabela 1) foram utilizadas diferentes combinações de parâmetros passados à ferramenta de testes, visando simular diferentes cenários de carga e comportamentos de acesso. Estes cenários foram divididos em dois grupos principais, focados em analisar a sobrecarga fundamental dos protocolos e o desempenho em cenários de concorrência:

- **Usuário único sequencial:** Estes valores foram escolhidos visando simular um único usuário realizando requisições de forma sequencial. Isso para comparar diretamente as três versões do protocolo pelo custo de estabelecimento de conexões e processamento da requisição deixando de lado a vantagem da multiplexação do HTTP/2 e HTTP/3 em relação ao HTTP/1.1.
  - **Parâmetros:**
    - Conexões (-c): 1
    - *Streams* simultâneos (-m): 1
- **Múltiplos Usuários e *Streams*:** Estes valores simulam um cenário mais realista utilizados para avaliar o comportamento das versões com diferentes níveis de carga e concorrência.
  - **Parâmetros:**
    - Conexões (-c): 10, 100, 200
      - Justificativa: Simula níveis de carga baixo (10), médio (100) e alto (200) no servidor. A configuração do NGINX foi ajustada para *worker_processes* auto (resultando em 4 processos de trabalho, um para cada núcleo da CPU servidora) e *worker_connections* elevado para 10.000, suportado por um *worker_rlimit_nofile* de 65.536. Isso estabelece um limite teórico de 40.000 conexões concorrentes (4 * 10.000). Manter os níveis de teste abaixo desse teto é uma decisão deliberada para garantir que qualquer gargalo observado seja decorrente do processamento de cada protocolo, e não de uma saturação artificial do limite de conexões do servidor.
    - *Streams* simultâneos (-m): 10, 50, 100
      - Justificativa: Mede a eficiência e o ganho da multiplexação. Os valores de 10 e 50 simulam ganhos moderados, enquanto 100 simula um navegador agressivo carregando vários ativos, alinhando-se ao padrão recomendado para **streams** simultâneos no HTTP/2 (BELSHE; PEON; THOMSON, 2015).

Para avaliar o desempenho dos protocolos em diferentes cenários de implantação, os testes foram executados em três topologias de rede distintas, cada uma com seus próprios parâmetros e condições:

- **Cenário 1: Ambiente LOCAL**
  - **Descrição:** Cliente e servidor executados em máquinas virtuais no mesmo hospedeiro físico. Esta topologia visa medir o desempenho com latência de rede negligenciável, focando na capacidade de processamento de cada versão.
  - **Condições Simuladas:** A banda foi limitada sinteticamente via `cake` para estabelecer duas linhas de base:
    - 1000 Mbps
    - 100 Mbps
- **Cenário 2: Ambiente LAN (Sintético)**
  - **Descrição:** Cliente e servidor em máquinas físicas distintas conectados por uma rede local (LAN). Este foi o cenário principal para testes de resiliência, onde as condições de rede foram manipuladas sinteticamente com `tc` e `netem`.
  - **Condições Simuladas:** A seleção destes cenários foi baseada nos perfis "Mobile"(50ms RTT, 4% de perda) e "Wi-Fi"(50ms RTT, 0% de perda) apresentados no trabalho de Liu *et al.* (2024, Fig. 7), complementados por uma linha de base ideal:
    - **Rede Ideal (Baseline LAN):** 1000 Mbps, 0 ms delay, 0% loss.
    - **Rede com Latência ("Wi-Fi"):** 1000 Mbps, 50 ms delay, 0% loss.
    - **Rede com Latência e Perda ("Mobile"):** 1000 Mbps, 50 ms delay, 4% loss.
- **Cenário 3: Ambiente CLOUD (Internet Real)**
  - **Descrição:** Testes executados entre o cliente (em rede local) e o servidor NGINX instanciado em um provedor de nuvem pública, simulando um acesso real de usuário pela Internet.
  - **Condições da Rede (Não-sintéticas):** Nenhuma modelação de tráfego foi aplicada. Os testes utilizaram as condições reais da rede, caracterizadas por:
    - Banda: $\approx$ 300 Mbps (largura de banda nominal da instância).
    - Latência: $\approx$ 250 ms (RTT médio observado).
    - Perda/Jitter: Variação natural da rede pública de Internet durante a execução dos testes.

3.4.2 Automação e Coleta de Dados

Para garantir a repetibilidade e a execução controlada dos testes, foi desenvolvido um código de automação em `bash`, presente na Listagem A.3 (Apendice A). Este algoritmo foi responsável pela execução de todos os cenários de teste, iterando sobre as combinações de parâmetros predefinidas.

O processo de automação executava os seguintes passos:

- **Iteração da Matriz de Teste:** O *script* itera sobre todas as combinações de: (1) Perfis de Página (P25, P50, P75); (2) Protocolos (HTTP/1.1, HTTP/2, HTTP/3); e (3) Grupos de Carga (Cenário 1a com `-c=1`, `-m=1` e; Cenário 2a com `-c={10, 100, 200}` e `-m={10, 50, 100}`).
- **Repetições:** Para cada combinação de parâmetros única, o teste foi executado **cinco vezes** (`NUM_TESTS=5`) para permitir a análise estatística e a identificação de desvios atípicos.
- **Execução do `h2load`:** O *script* invocou a ferramenta `h2load` (compilada localmente em `$HOME/nghttp2/src/h2load`) com os parâmetros apropriados. O número total de requisições (`-n`) foi calculado dinamicamente, multiplicando o número de conexões (`-c`) pelo número de requisições por cliente (`rpc`) definido em cada perfil de página, visando manter o número de requisições por cliente fixo. O número de *threads* do cliente (`-t`) também foi ajustado para adequar-se ao número de clientes.
- **Tratamento Específico de Protocolo:** A lógica do *script* adapta os parâmetros para cada protocolo. Para testes em HTTP/1.1, o número de *streams* foi forçado para `-m=1`, refletindo a ausência de multiplexação. Além disso, execuções redundantes foram descartadas. Para HTTP/2 e HTTP/3, os valores de `-m` definidos foram utilizados.
- **Coleta e Agregação de Dados:** Ao final de cada execução, a saída de texto do `h2load` foi salva em um arquivo `.txt` individual. Uma função de captura dos dados (`parse_results`) foi acionada para extrair as métricas (como *throughput*, latência, *Time To First Byte* (TTFB) e contagem de status) e agregá-las em um arquivo `results.csv`.

Este processo assegurou que cada teste fosse executado sob condições idênticas e produziu um conjunto de dados estruturado e pronto para a análise.

3.4.3 Métricas de Desempenho Coletadas

Para a análise comparativa dos protocolos, foram coletadas três métricas de desempenho principais em cada execução de teste fornecidas pela ferramenta `h2load`.

- **Tempo de Chegada do Primeiro Byte (TTFB):** O TTFB mede o intervalo de tempo desde o envio da requisição HTTP pelo cliente até o recebimento do primeiro byte da resposta do servidor. Esta métrica é fundamental para medir a responsividade combinada da rede e do servidor. Ela engloba a latência da rede (RTT), o tempo de estabelecimento da conexão (TCP e TLS *handshake*) e, crucialmente, o tempo de processamento do servidor. Um TTFB alto é frequentemente um indicador de gargalos no *back-end* ou latência de rede elevada (W3C, 2021).

- **Tempo Total do Teste:** Esta é a métrica *total time* reportada pelo `h2load`. Ela mede a duração total do teste, desde o início da primeira requisição até a conclusão da última requisição do total de requisições `-n` definido no início do teste. O tempo **Total do Teste** avalia o desempenho agregado do sistema para processar a carga de trabalho completa. É a medida mais abrangente do experimento, refletindo a capacidade de ponta a ponta do sistema em processar um volume de trabalho. Um tempo total menor indica um desempenho geral superior.

- **Throughput (Vazão de Aplicação):** O *Throughput* reportado pelo `h2load` mede a vazão efetiva na camada de aplicação. Esta taxa é calculada dividindo o tamanho total dos arquivos transferidos com sucesso pelo tempo total do teste. Esta métrica representa o "goodput", pois ignora a sobrecarga dos cabeçalhos de protocolo (TCP, UDP, QUIC, HTTP) e o tráfego gerado por retransmissões. É a medida mais precisa da taxa de entrega de dados do ponto de vista da aplicação, refletindo diretamente a velocidade com que o conteúdo foi recebido pelo cliente (KUROSE; ROSS, 2021).

Desta forma, com o servidor definido e as ferramentas de análise especificadas, os experimentos foram realizados e os resultados são apresentados no próximo capítulo.

4 RESULTADOS ALCANÇADOS

Este trabalho realizou uma análise comparativa entre as versões HTTP/1.1, HTTP/2 e HTTP/3, com foco em suas especificações técnicas, complexidade de configuração e desempenho em diferentes contextos. O problema central abordado foi a carência de estudos atualizados que integrem análise teórica, testes práticos com servidor Web e interpretação de dados reais de tráfego de forma analítica.

A abordagem adotada foi estruturada em três eixos principais:

- **Estudo técnico** das RFCs que definem cada versão do protocolo, destacando as mudanças mais relevantes, como multiplexação, compressão de cabeçalho, eliminação do *handshake* TCP e uso do QUIC.
- **Configuração prática** de cada versão utilizando o servidor NGINX, identificando as diferenças, dificuldades e exigências específicas de cada uma.
- **Avaliação de desempenho** por meio de testes controlados e análise de relatórios existentes, os quais permitiram identificar os contextos em que cada versão apresenta o melhor desempenho.

Esses eixos foram definidos visando atender aos objetivos específicos do trabalho.

4.1 Estudo técnico

O estudo técnico envolveu a análise das diferentes versões do protocolo HTTP (isto é, as versões do HTTP/1.1, do HTTP/2 e do HTTP/3), com base nas respectivas *Request for Comments* (RFCs).

4.1.1 Comparação entre as diferentes versões do HTTP

A versão HTTP/1.1 foi amplamente adotada, permanecendo como o padrão dominante por mais de duas décadas. Ela introduziu melhorias significativas em relação ao HTTP/1.0, como o suporte a conexões persistentes, mecanismo adicional de cache, *pipeling* de solicitação e transferência de codificação fragmentada (FIELDING; NOTTINGHAM; RESCHKE, 2022).

No entanto, com o crescimento da Web e o aumento da complexidade das páginas e aplicações, surgiram desafios relacionados à latência e à eficiência do protocolo, sobretudo devido ao seu modelo de comunicação sequencial sobre o TCP (BELSHE; PEON; THOMSON, 2015).

Para superar essas limitações, foi desenvolvido o HTTP/2, que trouxe avanços expressivos na forma como os dados são transmitidos. Seu principal diferencial é a multiplexação de

streams em uma única conexão TCP, eliminando o bloqueio por ordem de requisição e permitindo maior paralelismo. Adicionalmente, o protocolo incorporou a compressão de cabeçalhos por meio do algoritmo HPACK, reduzindo a sobrecarga causada pela repetição de informações como *cookies* e campos de autenticação (BELSHE; PEON; THOMSON, 2015).

Apesar dessas melhorias, o HTTP/2 ainda herdava as limitações do TCP, em especial os efeitos do (*Head-Of-Line Blocking - HOL-Blocking*). Cabe salientar, que esse problema ocorre quando um pacote de dados na frente de uma fila impede que outros pacotes sejam processados, no caso de uma perda, mesmo que estes estejam prontos para serem transmitidos (no envio). Se um pacote se perde, o TCP espera o reenvio antes de entregar os pacotes seguintes. Isso significa que todos os *streams* multiplexados no HTTP/2 ficam bloqueados até que o pacote perdido seja retransmitido. Mesmo que só um *stream* seja afetado, todos os outros que compartilham a mesma conexão TCP sofrem atraso (BELSHE; PEON; THOMSON, 2015). Consequentemente, como o TCP garante a entrega ordenada dos pacotes, a perda de um único segmento pode atrasar a entrega de todos os demais *streams* multiplexados naquela conexão, comprometendo a performance em ambientes de rede com alta latência ou perda de pacotes.

Com o HTTP/3, o protocolo passa por uma mudança mais profunda ao adotar o QUIC como novo transporte subjacente, substituindo o TCP. O QUIC, por sua vez, é implementado sobre o UDP e foi projetado para oferecer conexões mais rápidas, seguras e resilientes. Uma das inovações centrais do QUIC está em seu processo de *handshake*, que combina o estabelecimento de conexão e a negociação criptográfica em uma única etapa, reduzindo o tempo até a primeira resposta TTFB (BISHOP, 2022).

Além disso, o QUIC permite a multiplexação de *streams* independentes, o que significa que perdas de pacotes em um fluxo não afetam os demais, eliminando o problema do *Head-Of-Line Blocking - HOL-Blocking*. Essa característica resulta teoricamente em melhorias substanciais de desempenho, especialmente em redes móveis ou instáveis, em que a latência e a perda de pacotes são mais frequentes (BISHOP, 2022). A Figura 1 mostra a pilha de protocolos TCP/IP considerando as diferentes versões do protocolo HTTP.

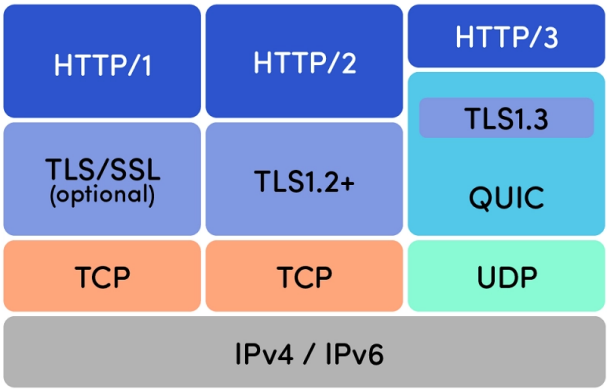

Figura 1 – Pilha de Protocolos considerando as diferentes versões do protocolo HTTP. Fonte: (COMMUNICATIONS, 2023)

As diferentes versões do protocolo HTTP foram projetadas para superar limitações de desempenho das versões anteriores, principalmente no que diz respeito à latência, eficiência na transmissão e robustez em ambientes de rede instáveis. A Tabela 2 apresenta um resumo comparativo das principais características entre o HTTP/1.1, HTTP/2 e HTTP/3, conforme especificado nas RFCs 9110, 7540 e 9114.

Tabela 2 – Comparação entre HTTP/1.1, HTTP/2 e HTTP/3 em aspectos de desempenho

Aspecto	**HTTP/1.1 (RFC 9110)**	**HTTP/2 (RFC 7540)**	**HTTP/3 (RFC 9114)**
Transporte	TCP	TCP	QUIC (UDP)
Formatação das mensagens	Texto (ASCII)	Binário (*frames*)	Binário (*frames* sobre QUIC)
Multiplexação	Não possui; concorrência exige múltiplas conexões	Multiplexação em uma única conexão TCP	Multiplexação em uma única conexão QUIC
Head-of-Line Blocking (HOL)	Presente no nível de aplicação e TCP	Mitigado na aplicação, mas ainda no TCP	Eliminado no QUIC (*streams* independentes)
Compressão de cabeçalhos	Não possui	HPACK (compressão eficiente)	QPACK (adaptado ao QUIC)
Manutenção de conexão	*Keep-Alive* limitado	Conexão persistente com múltiplos *streams*	Conexão persistente otimizada com QUIC
Prioridade de *streams*	Não possui	Suporte a prioridade e dependências	Suporte a prioridade no QUIC
Segurança/TLS	TLS opcional	TLS 1.2/1.3 obrigatório	TLS 1.3 integrado ao QUIC
Latência de conexão	Alta (*handshakes* TCP + TLS separados)	Menor, mas dependente do TCP	Muito baixa (0-RTT/1-RTT[1] com QUIC)

4.2 Configuração Prática do Servidor

A configuração prática do ambiente de testes exigiu que o mesmo servidor NGINX fosse capaz de responder a requisições HTTP/1.1, HTTP/2 e HTTP/3. Para tal, foi utilizada uma versão do NGINX com os módulos necessários para o suporte ao protocolo QUIC, instalada utilizando os pacotes binários para Ubuntu, de acordo com o tutorial oficial do NGINX (NGINX, INC., 2024b).

A configuração foi dividida em dois níveis: parâmetros globais do Nginx para otimização de desempenho e a configuração específica do *virtual host* para habilitar os três protocolos.

[1] No contexto do QUIC (RFC 9000), o **1-RTT** corresponde ao estabelecimento de uma nova conexão com apenas um ciclo de ida e volta (Round-Trip Time). O **0-RTT**, possibilitado pelo TLS 1.3, permite que em retomadas o cliente envie dados imediatamente antes do término de um novo *handshake*, reduzindo de forma significativa a latência inicial (IYENGAR; THOMSON, 2021b; BISHOP, 2022).

4.2.1 Parâmetros Globais e de Desempenho

No arquivo principal de configuração (`nginx.conf`), os parâmetros de desempenho foram ajustados para:

- **Processos de Trabalho (Workers):** A diretiva `worker_processes` foi definida como `auto`, instruindo o NGINX a iniciar um processo de trabalho por núcleo de CPU disponível (quatro, no caso do servidor de testes).
- **Limites de Conexão:** Para suportar os testes de carga, o limite de descritores de arquivo por processo foi elevado para `65536` (`worker_rlimit_nofile`). O número de conexões por *worker* foi definido para `10000` (`worker_connections`), resultando em uma capacidade teórica total de 40.000 conexões concorrentes, garantindo que o limite do servidor não fosse um gargalo.
- **Recursos do Sistema:** A diretiva `sendfile on` foi utilizada para permitir que o kernel copiasse dados de arquivos diretamente para o *socket*, otimizando a entrega de *payloads* estáticos ao reduzir a sobrecarga de cópias entre o *kernelspace* e o *userspace* (KUROSE; ROSS, 2021).

A configuração completa destes parâmetros globais, incluindo os ajustes de workers e limites de descritores de arquivo visando evitar gargalos no sistema operacional, pode ser consultada na Listagem A.1 (Apêndice A).

4.2.2 Configuração Unificada de Protocolos

Como pré-requisito para a criptografia (mandatória no HTTP/2 e HTTP/3), um certificado digital autoassinado do tipo RSA (4096 bits, SHA-256) foi gerado via OpenSSL:

```
$ sudo openssl req -x509 -newkey rsa:4096 -sha256 -days 365 \
-nodes -keyout cert.key -out cert.crt
```

O bloco de configuração do servidor `server-block`, responsável por servir o conteúdo, foi configurado para habilitar os três protocolos de forma unificada na porta 443:

- **Habilitação de H1.1 e H2:** A diretiva `listen 443 ssl http2;` instruiu o NGINX a escutar na porta 443 sobre TCP, utilizando SSL, e a negociar tanto HTTP/1.1 quanto HTTP/2.
- **Habilitação de H3:** A diretiva `listen 443 quic reuseport;` instruiu o NGINX a escutar na *mesma porta* 443, mas sobre o protocolo UDP, para aceitar conexões QUIC (HTTP/3). A opção `reuseport` é crucial para permitir que múltiplos processos *worker* se vinculem ao mesmo *socket* UDP, melhorando o desempenho.

- **Configuração TLS:** O servidor foi configurado para usar **TLSv1.3** (`ssl_protocols TLSv1.3;`), um requisito para o QUIC (NGINX, Inc., 2024). As *ciphersuites* foram definidas para garantir o uso de cifras modernas. Além disso, a diretiva `ssl_early_data on` foi habilitada para suportar o 0-RTT, uma das principais vantagens de desempenho do TLS 1.3 (NGINX, Inc., 2024).

Finalmente, o bloco `location` foi configurado para servir os arquivos de *payload* estáticos (P25, P50, P75) a partir do diretório `/etc/nginx/html`.

O bloco de servidor resultante, que consolida as diretivas necessárias para suportar simultaneamente HTTP/1.1, HTTP/2 e HTTP/3 (QUIC) na mesma porta 443 com segurança TLS 1.3, está detalhado na Listagem A.2 (Apêndice A).

4.3 Avaliação de desempenho

Esta seção apresenta a avaliação comparativa entre HTTP/1.1, HTTP/2 e HTTP/3 a partir dos experimentos executados. A análise segue a metodologia descrita no capítulo anterior, considerando as métricas de *Throughput* (MB/s), Tempo Total de Carregamento (ms) e TTFB (ms). Os resultados são agrupados por ambiente (Local, LAN e Nuvem), permitindo observar como largura de banda, latência e perda influenciam o comportamento de cada protocolo.

A avaliação também contempla diferentes tamanhos de página (Leve, Média e Pesada) e cargas variáveis de requisições, controladas pelo número de conexões simultâneas (`-c`) e, para os protocolos multiplexados, pelo número de *streams* simultâneos (`-m`).

4.3.1 Cenário LOCAL

O cenário LOCAL representa um ambiente idealizado, com largura de banda elevada (1000 Mbps), latência mínima e ausência total de perda de pacotes. O objetivo não é simular um caso real, mas estabelecer um ponto de referência para comparação com os demais cenários.

Com apenas uma conexão sequencial, os resultados obtidos são mostrados nas Figuras 2 e 3. Observa-se que, em ambiente ideal, HTTP/1.1 e HTTP/2 apresentam desempenho praticamente idêntico em termos de vazão e tempo total. Ambos saturam a largura de banda disponível e completam os testes simultaneamente. Já o HTTP/3 apresenta desempenho inferior nos dois parâmetros. Essa diferença está alinhada ao que é discutido por Rath *et al.* (2023): o QUIC, por operar no *user-space*, não aproveita as otimizações maduras do *kernel* TCP/IP, resultando em maior custo computacional. Em ambientes de baixa latência e alta largura de banda, esse custo passa a ser um limitador perceptível do desempenho. Entretanto, quando a banda é artificialmente limitada (100 Mbps), o gargalo passa a ser o link, e não mais o servidor. Nesse caso, as três versões apresentam resultados praticamente idênticos.

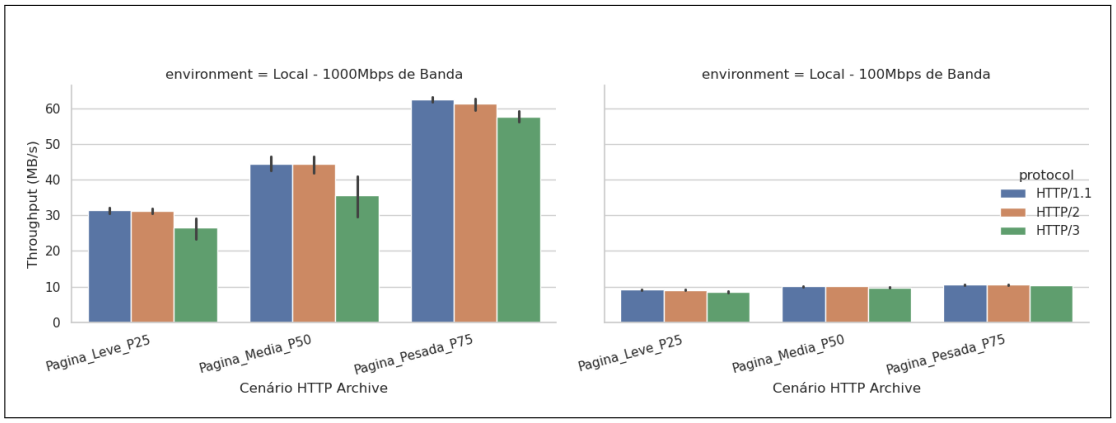

Figura 2 – Throughput vs. Complexidade da Página em Cenários LOCAL (1a).

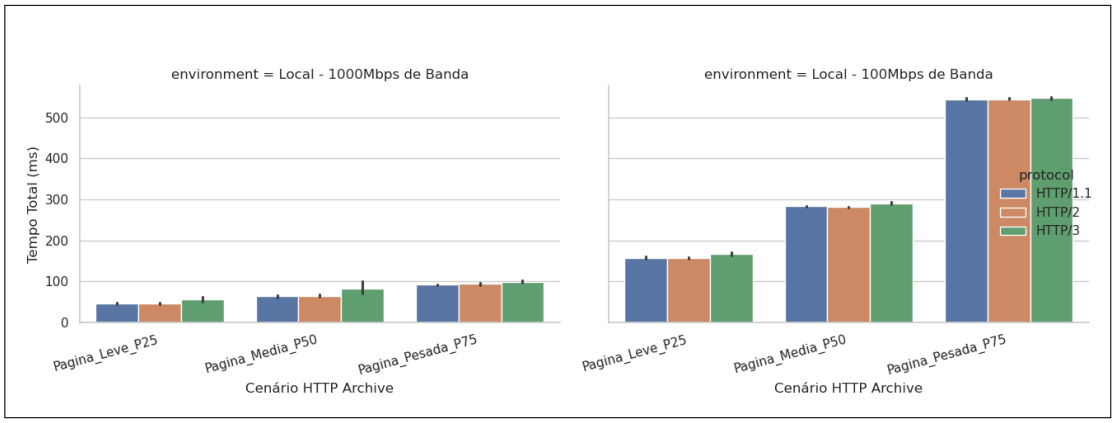

Figura 3 – Tempo Total (s) vs. Complexidade da Página em Cenários LOCAL (1a).

Nos cenários com dez e cem conexões, o comportamento se mantém: HTTP/1.1 e HTTP/2 permanecem equivalentes, enquanto o HTTP/3 continua atrás na rede ideal. Com duzentas conexões simultâneas (Figuras 4 e 5), observa-se uma instabilidade pontual da implementação do HTTP/3 utilizada, com flutuações na vazão e aumento do tempo total.

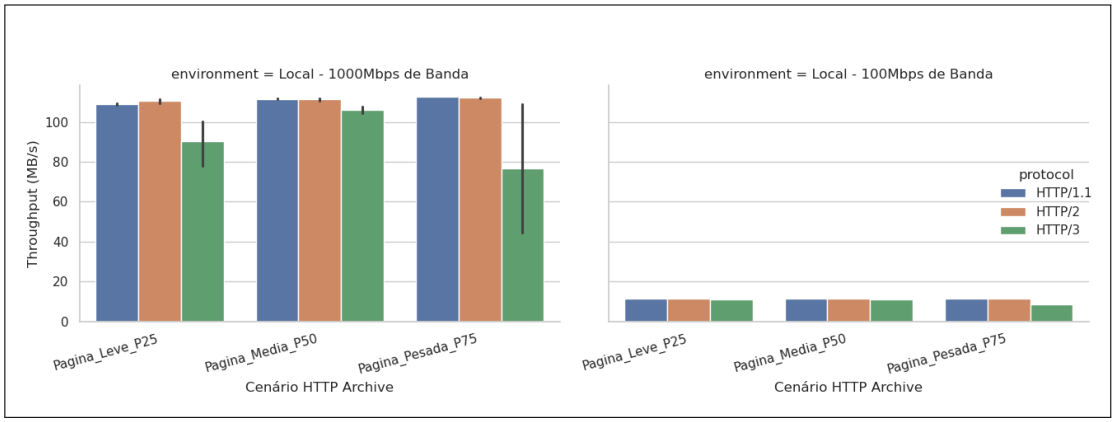

Figura 4 – Throughput vs Complexidade da Página em Cenários LOCAL (2a).

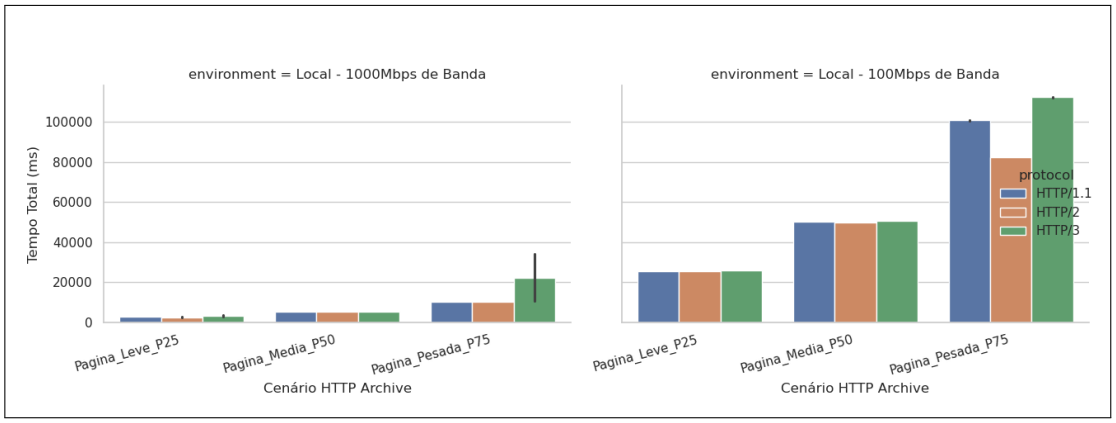

Figura 5 – Tempo Total(s) vs. Complexidade da Página em Cenários LOCAL (2a).

Esse comportamento, repetido em outras cargas, indica uma possível limitação de maturidade da implementação do QUIC em condições de alto estresse.

4.3.2 Cenário LAN

O cenário LAN introduz latência e perda controladas sobre a mesma largura de banda do cenário LOCAL (Figura (6) e Figura 7), simulando o ambiente de uma rede corporativa. Com uma única conexão, o comportamento é semelhante ao do cenário LOCAL, já que a latência aplicada ainda é pequena para causar diferenças marcantes. Porém, ao adicionar perda, surge uma vantagem clara para o HTTP/3 no tempo total. Essa vantagem se explica pelo QUIC eliminar o *HOL-Blocking* em nível de transporte, permitindo que perdas afetem apenas o *stream* correspondente.

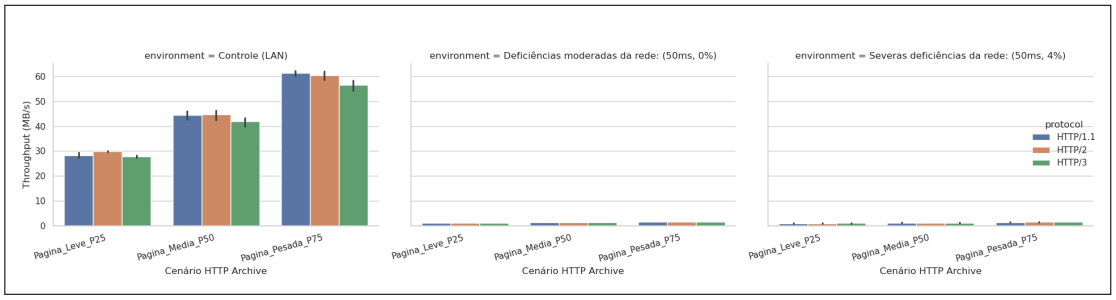

Figura 6 – Throughput vs. Complexidade da Página em Cenários LAN (1b).

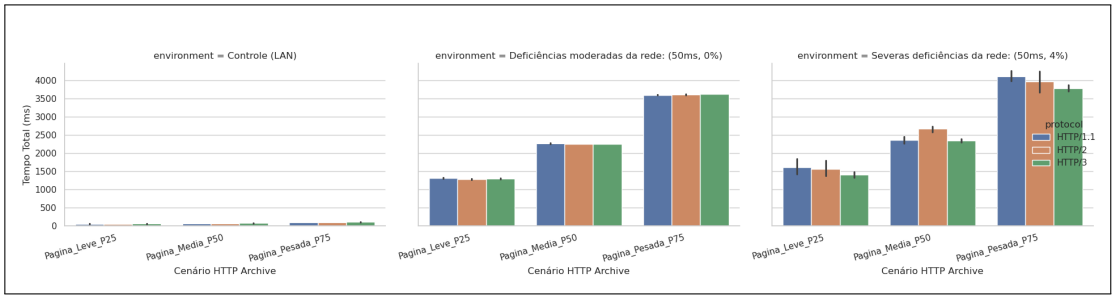

Figura 7 – Tempo Total (s) vs. Complexidade da Página em Cenários LAN (1b).

Ao aumentar a carga, as Figuras 8 e 9 mostram que o HTTP/1.1 passa a ser fortemente penalizado pela latência, já que cada requisição exige uma conexão independente. O HTTP/2 e o HTTP/3 permanecem próximos, com o HTTP/3 mantendo vantagem nos cenários degradados.

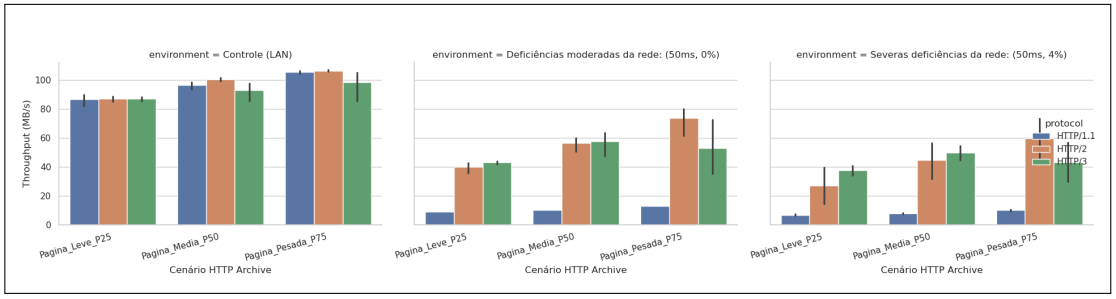

Figura 8 – Throughput vs. Complexidade da Página em Cenários LAN (2b).

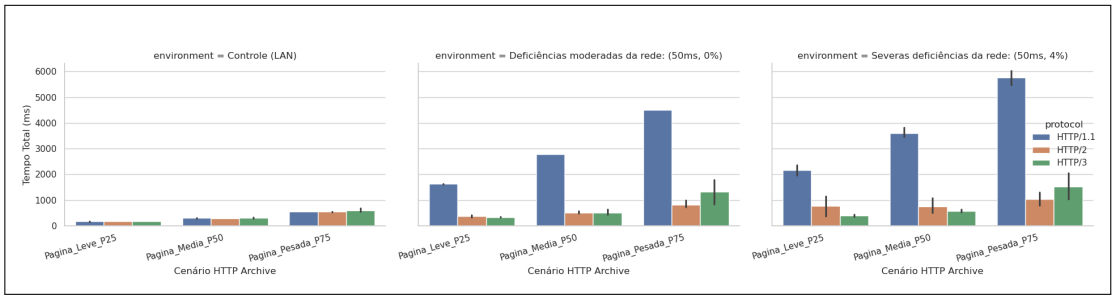

Figura 9 – Tempo Total (s) vs. Complexidade da Página em Cenários LAN (2b).

Com cem e duzentas conexões, evidencia-se novamente a instabilidade do HTTP/3 nesta implementação (Figuras 10 a 13), enquanto HTTP/2 permanece estável. O HTTP/1.1 melhora levemente a vazão por abrir muitas conexões, mas continua inferior em tempo total.

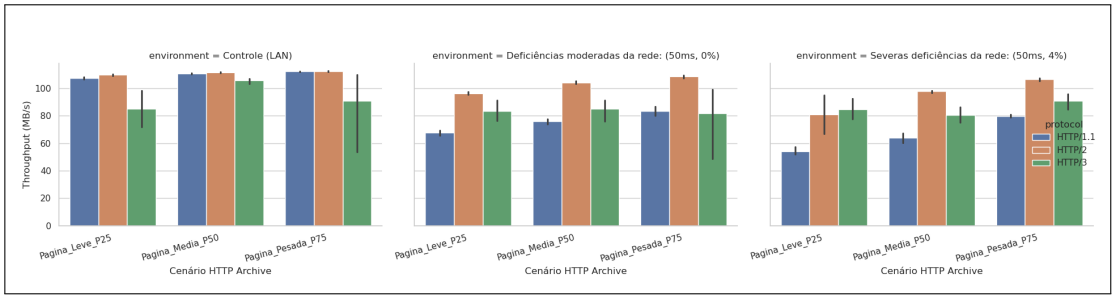

Figura 10 – Throughput vs. Complexidade da Página em Cenários LAN (2b).

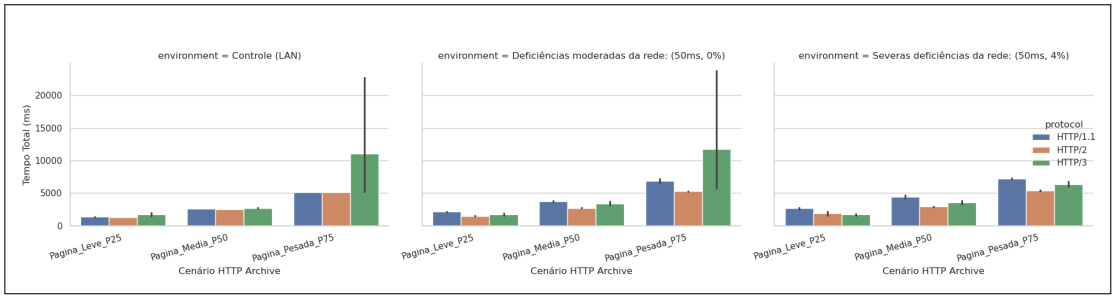

Figura 11 – Tempo Total (s) vs. Complexidade da Página em Cenários LAN (2b).

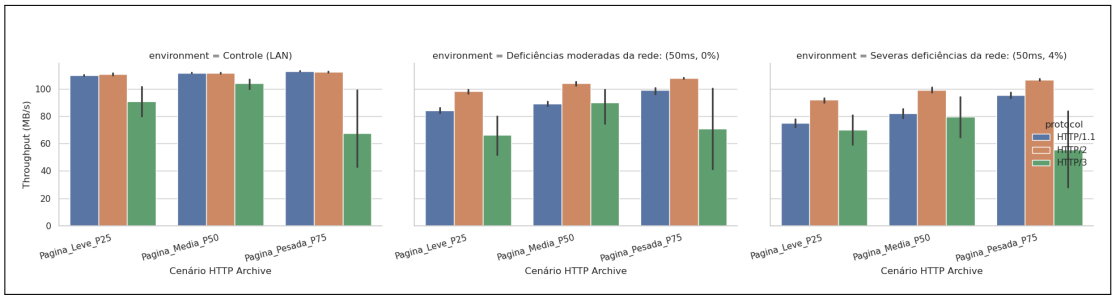

Figura 12 – Throughput vs. Complexidade da Página em Cenários LAN (2b).

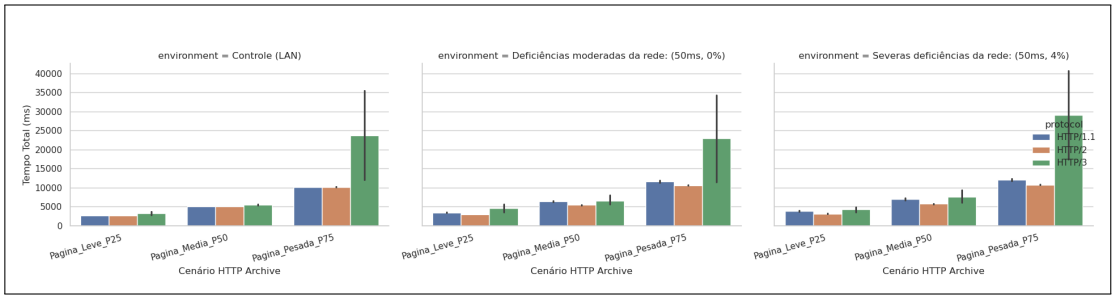

Figura 13 – Tempo Total (s) vs. Complexidade da Página em Cenários LAN (2b).

O grau de multiplexação (`-m`), assim como no cenário LOCAL, não alterou significativamente os resultados, reforçando que a variável determinante para carga no servidor foi o número de conexões simultâneas.

4.3.3 Cenário NUVEM

O cenário Nuvem representa um ambiente sem interferências artificiais, onde a variação natural de latência e possíveis perdas impactam o desempenho.

Com uma única conexão sequencial, o comportamento é semelhante entre as versões, já que a carga é baixa e a latência natural do ambiente não impacta significativamente o desempenho (Figura 14 e Figura 15).

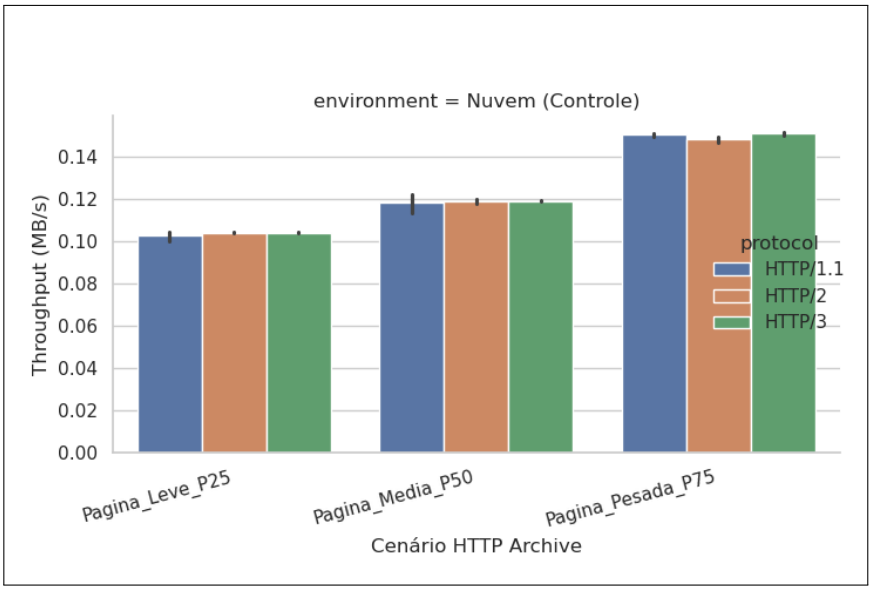

Figura 14 – Throughput vs. Complexidade da Página em Cenários Nuvem (1c).

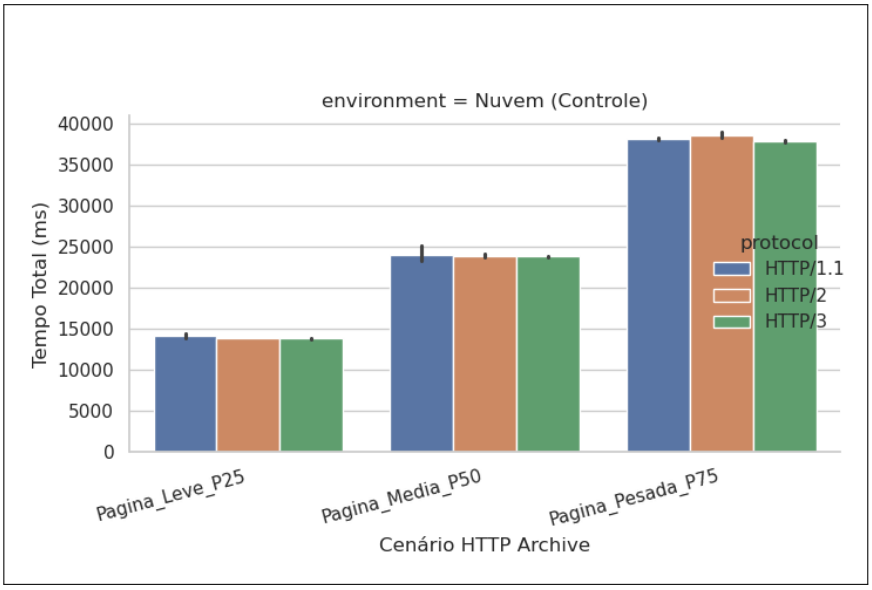

Figura 15 – Tempo Total (s) vs. Complexidade da Página em Cenários Nuvem (1c).

Ao adicionar conexões e *streams*, as Figuras 16 e 17 revelam uma vantagem notável do HTTP/3 em praticamente todos os cenários com dez *streams*. Em redes não ideais, o QUIC tende a se sobressair por mitigar o *HOL-Blocking* e lidar melhor com microperdas, conforme enfatizado pela RFC 9114 (BISHOP, 2022).

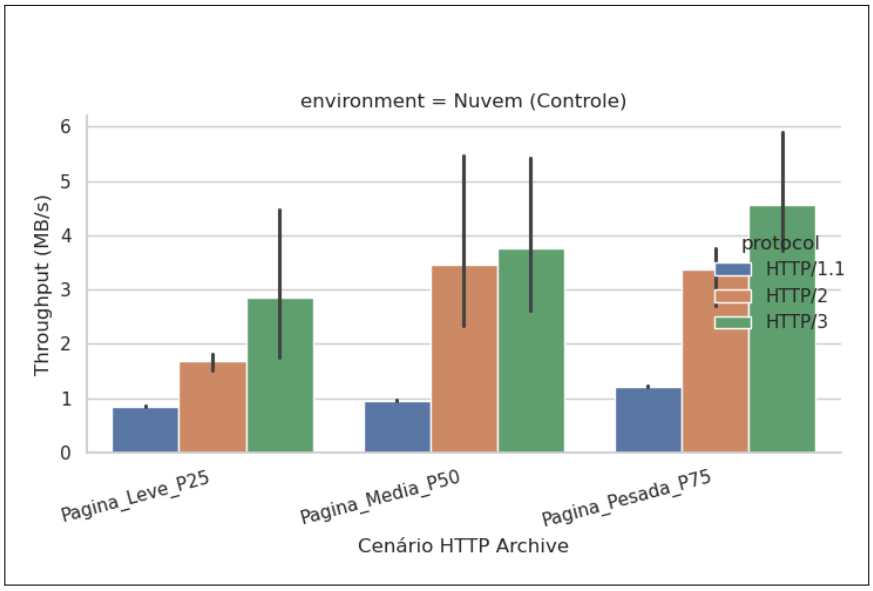

Figura 16 – Throughput vs. Complexidade da Página em Cenários NUVEM (2c).

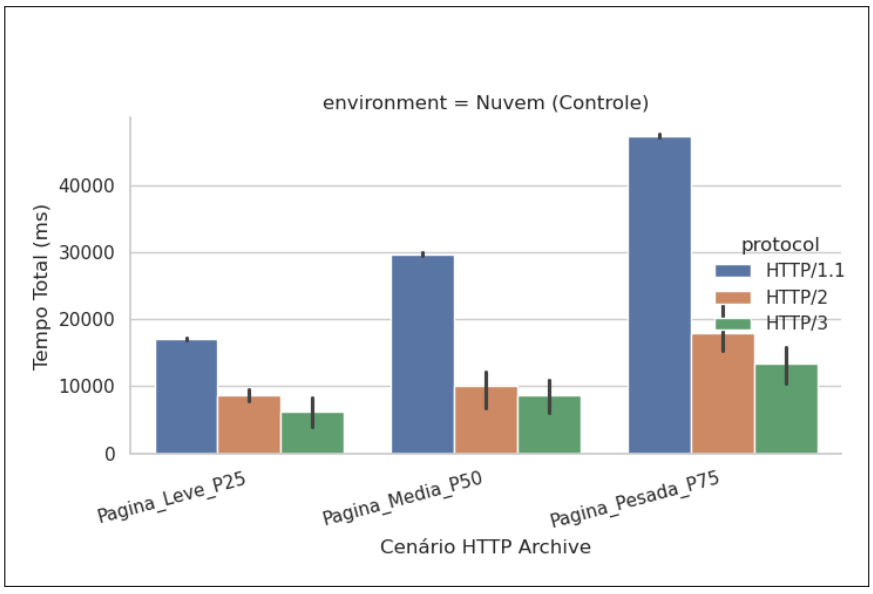

Figura 17 – Tempo Total (s) vs. Complexidade da Página em Cenários NUVEM (2c).

Essa vantagem persiste mesmo com mais carga, como ilustrado nas Figuras 18 e 19.

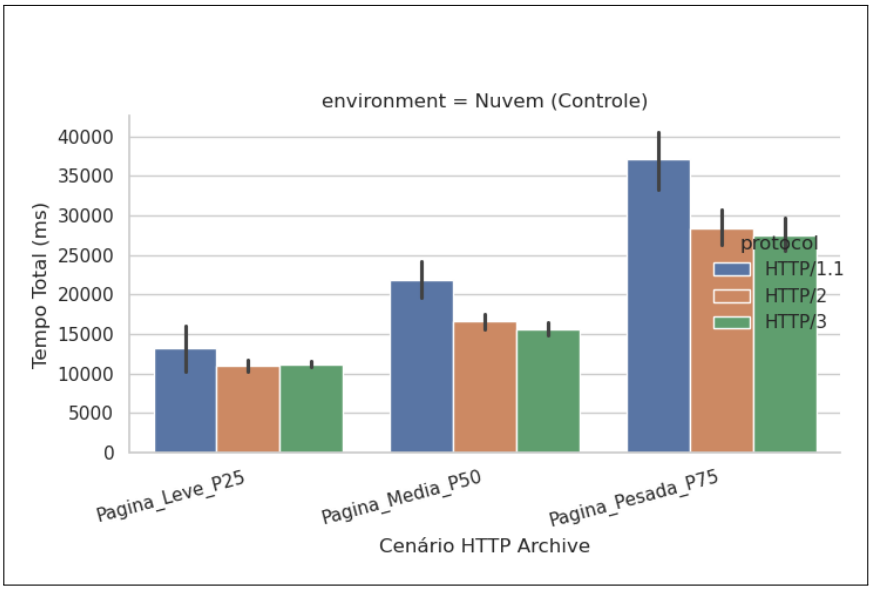

Figura 18 – Tempo Total (s) vs. Complexidade da Página em Cenários NUVEM (2c)

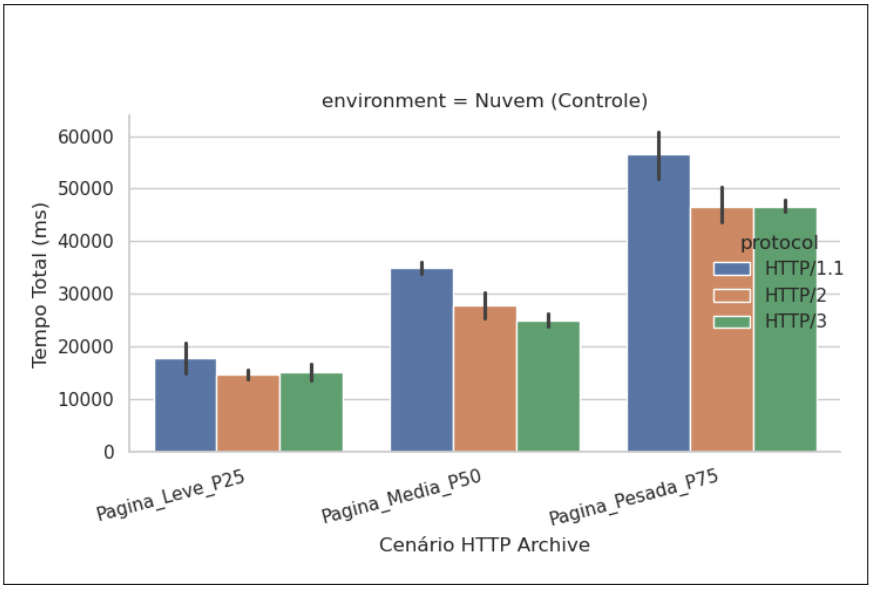

Figura 19 – Tempo Total (s) vs. Complexidade da Página em Cenários NUVEM (2c)

Entretanto, quando o número de *streams* passa de dez e a concorrência aumenta, observa-se que o HTTP/3 volta a ser superado pelo HTTP/2 (Figuras 20 e 21). Esse comportamento reforça a hipótese levantada no cenário LOCAL: o custo computacional do HTTP/3, por operar em *user-space*, torna-se o novo gargalo em cargas elevadas, enquanto o HTTP/2 se beneficia da pilha TCP/IP otimizada no *kernel* (KUROSE; ROSS, 2021).

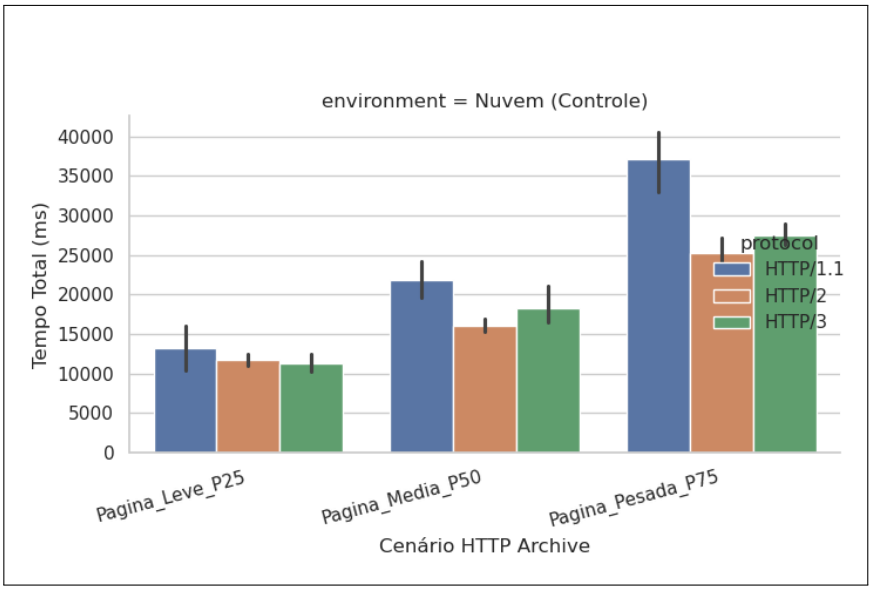

Figura 20 – Tempo Total (s) vs. Complexidade da Página em Cenários NUVEM (2c).

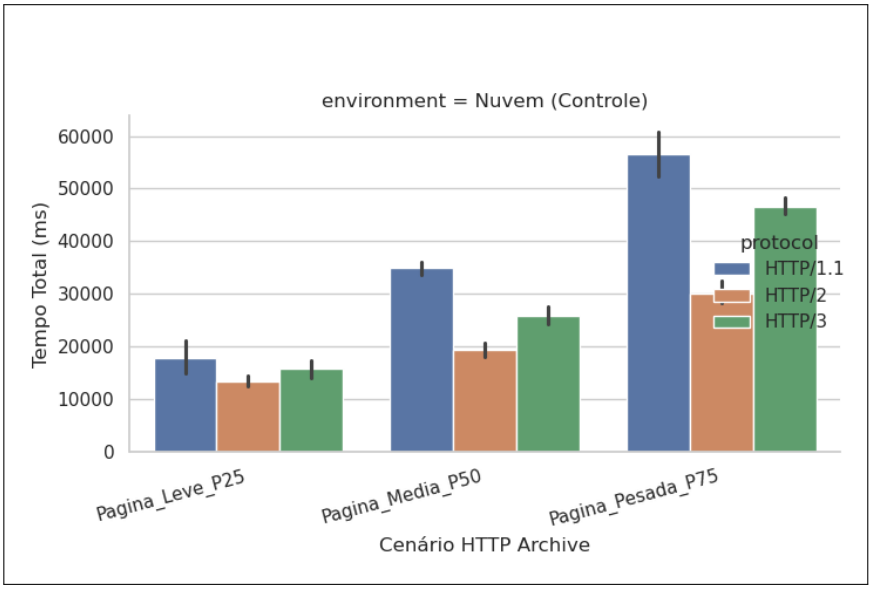

Figura 21 – Tempo Total (s) vs. Complexidade da Página em Cenários NUVEM (2c).

4.3.4 Análise de TTFB

A Figura 22 apresenta a comparação de TTFB entre os protocolos. A vantagem do HTTP/3 é evidente nos três cenários. O motivo está no *handshake* otimizado do QUIC (IYENGAR; THOMSON, 2021a): o estabelecimento da conexão e a negociação do TLS 1.3 ocorrem em 1-RTT, com a possibilidade de 0-RTT na retomada de sessão. Já o HTTP/2 sobre TCP exige 1-RTT para o *handshake* TCP e mais 1-RTT para o *handshake* do TLS, totalizando pelo menos 2-RTTs. Esse ganho se torna ainda mais relevante em redes com maior latência, como a LAN e a Nuvem.

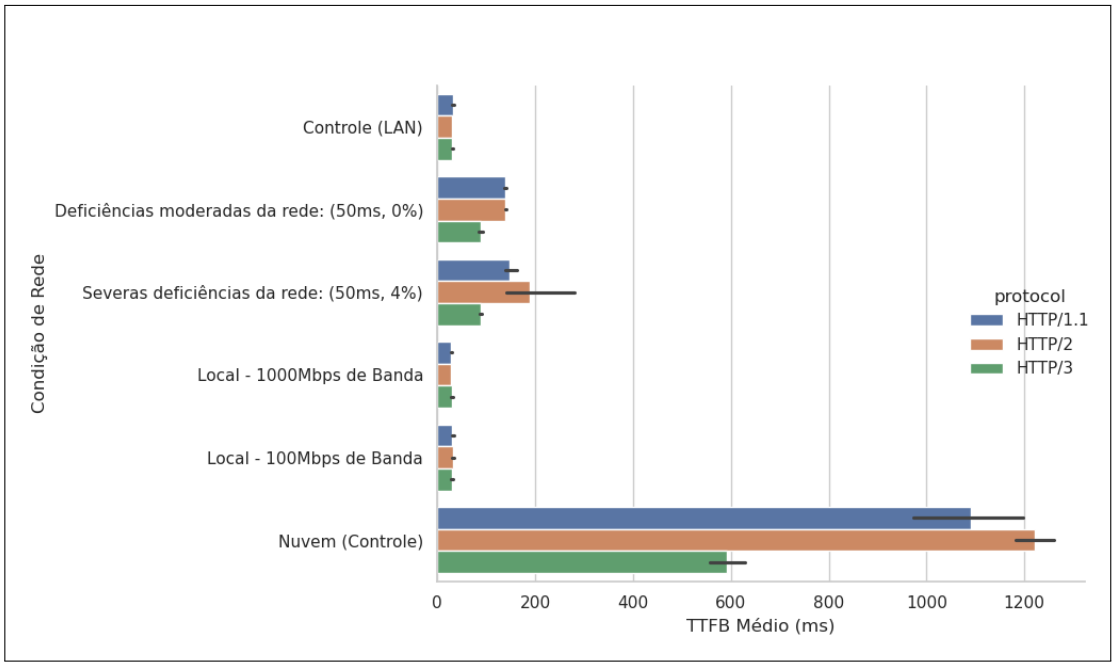

Figura 22 – TTFB vs. Ambientes.

4.3.5 Considerações Finais

A análise comparativa realizada nesta monografia demonstra que os resultados experimentais obtidos estão amplamente alinhados com a literatura sobre HTTP/2 e HTTP/3, mas também revelam nuances importantes que os trabalhos existentes não abordaram simultaneamente — sobretudo por integrarem ambientes LOCAL, LAN e NUVEM sob uma mesma metodologia e sob o mesmo servidor (NGINX).

Os cenários com degradação de rede confirmaram a vantagem do HTTP/3 em métricas sensíveis à latência, como o TTFB e o tempo total de transferência. Assim como observado por Trevisan et al. (2021) e Liu et al. (2024), o HTTP/3 se mostrou mais resiliente em condições adversas, comportamento explicado pela ausência de *Head-of-Line Blocking* no transporte QUIC e pela redução do custo de estabelecimento de conexão (1-RTT e 0-RTT). Esses fatores sustentam a conclusão de que o HTTP/3 oferece benefícios concretos ao usuário final, especialmente em redes típicas da Internet real, onde perdas intermitentes e latência elevada são comuns.

Entretanto, um aspecto pouco discutido nos trabalhos relacionados — e evidenciado de forma clara nos testes desta monografia — é o impacto computacional do HTTP/3 em cenários de carga muito elevada. Em redes de alta capacidade e baixa latência, como o Cenário LOCAL (1000 Mbps), o HTTP/2 apresentou desempenho superior em throughput e estabilidade. Esse comportamento está alinhado com observações mais recentes sobre o custo operacional do QUIC, que, por funcionar no *user space*, demanda mais processamento do que o TCP otimizado no kernel. Assim, embora o HTTP/3 seja tecnologicamente mais avançado, sua implementação ainda apresenta limitações práticas quando o gargalo principal não é a rede, mas sim o processamento no servidor.

Dessa forma, a comparação entre as versões do protocolo permite uma visão mais equilibrada: (i) o HTTP/3 deve ser preferido para aplicações voltadas ao usuário final, serviços distribuídos globalmente e ambientes com latência imprevisível; (ii) o HTTP/2 permanece altamente competitivo — e em alguns cenários superior — para sistemas internos, ambientes de datacenter, microsserviços e aplicações que priorizam throughput bruto; (iii) o HTTP/1.1, apesar de tecnicamente superado, mantém relevância histórica e compatibilidade ampla, mas suas limitações estruturais são evidentes quando comparado aos demais.

Em síntese, este trabalho amplia o entendimento prático sobre o comportamento das versões do HTTP ao demonstrar que a adoção do HTTP/3 não deve ser pautada apenas por seu potencial teórico, mas por uma análise contextual das características da aplicação, do ambiente de rede e dos custos computacionais envolvidos. Os resultados obtidos fortalecem a literatura existente, ao mesmo tempo em que introduzem evidências empíricas inéditas dentro de um cenário abrangente e metodologicamente unificado.

4.4 Comparação com os trabalhos relacionados

A literatura sobre HTTP/2 e HTTP/3 destaca avanços importantes em latência, resiliência e desempenho. No entanto, poucos estudos avaliam simultaneamente as três versões do protocolo em diversos ambientes de rede utilizando um mesmo servidor. Essa lacuna é parcialmente preenchida pelo presente trabalho. Alguns apontamentos:

- **Trevisan et al. (2021) — Adoção e desempenho do HTTP/3 em larga escala.**
  O estudo de Trevisan *et al.* (2021) demonstrou que o HTTP/3 reduz significativamente a latência, especialmente em cenários com perda de pacotes, graças à eliminação do *HOL-Blocking*. **Comparação:** Nos cenários LAN e Nuvem desta monografia, essa vantagem foi confirmada. O HTTP/3 apresentou menor tempo total e menor TTFB, refletindo exatamente os benefícios associados ao QUIC observados por esses autores.

- **Balej et al. (2023) — Comparação entre implementações HTTP/3.**
  Em Balej e Sochor (2023), o NGINX apresentou melhor desempenho em relação a outras implementações. **Comparação:** Essa conclusão fundamentou a escolha do servidor neste trabalho. No entanto, os resultados aqui obtidos ampliam a discussão ao mostrar que, apesar do desempenho superior do NGINX para HTTP/3, o protocolo apresenta sobrecarga computacional perceptível sob alta taxa de transferência e alta concorrência, permitindo que o HTTP/2 o supere em cenários específicos.

- **Liu et al. (2024) — Resiliência do HTTP/3 em redes degradadas.**
  O estudo de Liu *et al.* (2024) mostrou que o HTTP/3 mantém desempenho consistente em alta latência e perda, superando o HTTP/2 em até 80% dos casos. **Comparação:** Os resultados desta monografia corroboram com essa resiliência. Em cenários LAN

e Nuvem, o HTTP/3 lidou melhor com microperdas e apresentou os menores tempos totais sem necessitar de otimizações adicionais, reforçando a robustez do QUIC.

- **Contribuições deste estudo.**

  Diferente dos estudos anteriores, esta monografia:

  - compara simultaneamente HTTP/1.1, HTTP/2 e HTTP/3;
  - utiliza o mesmo servidor NGINX para todas as versões;
  - avalia ambientes distintos (Local, LAN e Nuvem);

  Essa abordagem mais ampla permite observar não apenas os benefícios teóricos do HTTP/3, mas também limitações práticas, como sobrecarga computacional em cenários de alta concorrência.

Em síntese, enquanto estudos anteriores analisam aspectos isolados, esta monografia oferece uma comparação empírica completa entre as três versões do HTTP, contribuindo para orientar escolhas técnicas tanto na academia quanto no mercado.

5 CONCLUSÃO

Este trabalho apresentou uma investigação abrangente sobre a evolução dos protocolos HTTP, analisando comparativamente as versões HTTP/1.1, HTTP/2 e HTTP/3 em cenários distintos de rede e carga. A motivação central foi compreender como cada versão responde às demandas atuais da web, em que latência, variabilidade de rede e escalabilidade desempenham um papel fundamental na experiência do usuário e no desempenho de aplicações distribuídas. A construção de um ambiente controlado, aliada à utilização de ferramentas de medição consolidadas, permitiu realizar testes capazes de expor com clareza tanto os avanços quanto as limitações práticas de cada protocolo.

Os resultados revelaram que o HTTP/1.1, embora historicamente relevante, não atende mais às necessidades das aplicações modernas, sobretudo em cenários com latência ou múltiplas requisições simultâneas. A ausência de multiplexação e o excesso de conexões paralelas tornam seu desempenho limitado, confirmando sua inadequação para ambientes modernos. O HTTP/2 demonstrou ser a opção mais estável entre as avaliadas, beneficiando-se da maturidade do TCP e oferecendo desempenho consistente mesmo sob alta carga, o que o coloca como alternativa robusta para infraestruturas internas e sistemas que exigem previsibilidade. O HTTP/3, por sua vez, apresentou vantagens notáveis em redes degradadas e cenários com perda de pacotes, evidenciando os benefícios estruturais do QUIC. Entretanto, a análise identificou limitações associadas ao processamento em *user-space*, que impactam negativamente seu desempenho em condições de alta taxa de transferência ou grande concorrência.

A comparação entre as três versões reforça que não existe um protocolo universalmente superior, mas sim tecnologias otimizadas para contextos distintos. A escolha adequada depende diretamente da natureza da aplicação, das características da rede e do perfil de acesso esperado. Além disso, esta monografia contribui ao oferecer uma avaliação integrada e empírica, conduzida sobre uma base uniforme de configuração, permitindo observar interações que raramente são abordadas em estudos isolados.

Por fim, o estudo aponta para a necessidade de continuidade na evolução dos protocolos de transporte e aplicação, assim como para o amadurecimento das implementações de HTTP/3. À medida que o ecossistema web avança, ajustes em arquiteturas de servidores, estratégias de cache e camadas de transporte se tornam cada vez mais essenciais. Assim, espera-se que os resultados apresentados auxiliem tanto pesquisadores quanto profissionais na tomada de decisões informadas sobre o emprego das versões modernas do HTTP, contribuindo para sistemas mais eficientes, resilientes e responsivos.

REFERÊNCIAS

ARCHIVE, H. **Report: Page Weight**. [*S.l.*], 2025. Acessado em: 08 nov 2025. Disponível em: https://httparchive.org/reports/page-weight.

BALEJ, M.; SOCHOR, T. Performance comparison of http/3 server implementations. **Mendel University Repository**, 2023. Disponível em: https://doi.mendelu.cz/artkey/doi-990007-2000_PERFORMANCE-COMPARISON-OF-HTTP-3-SERVER-IMPLEMENTATIONS.php.

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.

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

Canonical. **Multipass documentation**. 2025. Acesso em: 13 nov. 2025. Disponível em: https://documentation.ubuntu.com/multipass/latest/.

CETIC.BR. **TIC Domicílios 2023: Pesquisa sobre o uso das tecnologias de informação e comunicação nos domicílios brasileiros**. 2024. Site Cetic.br. Disponível em: https://cetic.br/pt/pesquisa/domicilios/. Acesso em: 06 abr. 2025.

CLOUDFLARE, I. **Cloudflare Radar**. 2024. Site Cloudflare Radar. Disponível em: https://radar.cloudflare.com/pt-br#protocols. Acesso em: 10 abr. 2025.

CLOUDFLARE, I. **Cloudflare Radar 2024 Year**. 2024. Site Cloudflare Radar 2024. Disponível em: https://radar.cloudflare.com/year-in-review/2024. Acesso em: 06 abr. 2025.

COMMUNICATIONS, S. **Building the Next-Generation Web with HTTP/3**. 2023. Spirent Blog. Disponível em: https://www.spirent.com/blogs/building-next-generation-web-with-http-3. Acesso em: 25 maio 2025.

DYCODERS. **The Role of Website Speed in User Experience**. 2023. Blog Dycoders. Disponível em: https://dycoders.com/blog/the-role-of-website-speed-in-user-experience/. Acesso em: 16 abr. 2025.

FIELDING, R. T.; NOTTINGHAM, M.; RESCHKE, J. **HTTP Semantics**. RFC Editor, 2022. RFC 9110. (Request for Comments, 9110). Disponível em: https://www.rfc-editor.org/info/rfc9110.

FOROUZAN, B.; FEGAN, S. **Protocolo TCP/IP - 3.ed.** McGraw Hill Brasil, 2009. ISBN 9788563308689. Disponível em: https://books.google.com.br/books?id=fNvIgp3kkyQC. Acesso em: 16 abr. 2025.

FOROUZAN, B.; MOSHARRAF, F. **Redes de Computadores: Uma Abordagem Top-Down**. AMGH, 2013. ISBN 9788580551693. Disponível em: https://books.google.com.br/books?id=57BIAgAAQBAJ. Acesso em: 16 abr. 2025.

IYENGAR, J.; THOMSON, M. **QUIC: A UDP-Based Multiplexed and Secure Transport**. RFC Editor, 2021. RFC 9000. (Request for Comments, 9000). Disponível em: https://www.rfc-editor.org/info/rfc9000.

IYENGAR, J.; THOMSON, M. **RFC 9000: QUIC: A UDP-Based Multiplexed and Secure Transport**. [*S.l.*], 2021. Disponível em: https://www.rfc-editor.org/info/rfc9000. Acesso em: 5 set. 2025.

KUROSE, J. F.; ROSS, K. W. **Computer Networking: A Top-Down Approach**. 8th. ed. [*S.l.*]: Pearson, 2021. ISBN 978-0136681557.

LIU, F. *et al.* Performance comparison of http/3 and http/2: Proxy vs. non-proxy environments. **arXiv preprint arXiv:2409.16267**, 2024. Disponível em: https://arxiv.org/abs/2409.16267.

nghttp2.org. **h2load - HTTP/2 benchmarking tool**. 2025. Acesso em: 13 nov. 2025. Disponível em: https://nghttp2.org/documentation/h2load-howto.html.

NGINX, Inc. **NGINX: Module ngx_http_ssl_module**. 2024. Acesso em: 5 set. 2025. Disponível em: https://nginx.org/en/docs/http/ngx_http_ssl_module.html.

NGINX, INC. **NGINX: Module ngx_http_v2_module**. [*S.l.*], 2024. Disponível em: https://nginx.org/en/docs/http/ngx_http_v2_module.html. Acesso em: 5 set. 2025.

NGINX, INC. **NGINX QUIC/HTTP3 Development Branch**. [*S.l.*], 2024. Disponível em: https://hg.nginx.org/nginx-quic/. Acesso em: 5 set. 2025.

NIELSEN, J. **Website Response Times: The 3 Important Limits**. 2010. Site Nielsen Norman Group. Disponível em: https://www.nngroup.com/articles/response-times-3-important-limits/. Acesso em: 16 abr. 2025.

Pew Research Center. **Internet, Broadband Fact Sheet**. 2024. Site Pew Research Center. Disponível em: https://www.pewresearch.org/internet/fact-sheet/internet-broadband/. Acesso em: 06 abr. 2025.

PROJECT, Q. T. **OpenSSL with QUIC support**. 2024. GitHub repository. Disponível em: https://github.com/quictls/openssl. Acesso em: 5 set. 2025.

RATH, B. *et al.* Quic on the highway: Evaluating performance on high-rate links. **arXiv preprint arXiv:2309.16395**, 2023. Disponível em: https://arxiv.org/abs/2309.16395.

SMITH, J. E. H. **The Internet Is Not What You Think It Is: A History, a Philosophy, a Warning**. Princeton University Press, 2022. ISBN 9780691212326. Disponível em: https://books.google.com.br/books?id=6Z1EEAAAQBAJ. Acesso em: 06 abr. 2025.

TREVISAN, M. *et al.* Measuring http/3: Adoption and performance. *In*: **2021 19th Mediterranean Communication and Computer Networking Conference (MedComNet)**. IEEE, 2021. p. 1–8. Disponível em: https://ieeexplore.ieee.org/document/9501274.

W3C. **Navigation Timing Level 2**. 2021. W3C Recommendation. URL: https://www.w3.org/TR/navigation-timing-2/.

APÊNDICE A – Algoritmo de Coleta e Configuração do Servidor

Listing A.1 – Configuração do Nginx utilizada nos experimentos

```
user  nginx;
worker_processes  auto;

worker_rlimit_nofile 65536;

error_log  /var/log/nginx/error.log notice;
pid        /run/nginx.pid;

events {
    worker_connections  10000;
}

http {
    include       /etc/nginx/mime.types;
    default_type  application/octet-stream;

    log_format  main  '$remote_addr - $remote_user [$time_local] "
   $request" '
                      '$status $body_bytes_sent "$http_referer" '
                      '"$http_user_agent" "$http_x_forwarded_for"';

    access_log  /var/log/nginx/access.log  main;

    sendfile        on;
    keepalive_timeout  65;

    include /etc/nginx/conf.d/*.conf;
}
```

Listing A.2 – Configuração do servidor Nginx com HTTP/2 e HTTP/3 (QUIC)

```
server {
    listen 443 ssl;
    http2 on;
    listen 443 quic reuseport;

    ssl_certificate /etc/nginx/certs/cert.crt;
    ssl_certificate_key /etc/nginx/certs/cert.key;

    ssl_protocols TLSv1.3;

    ssl_conf_command Ciphersuites TLS_AES_256_GCM_SHA384:
   TLS_CHACHA20_POLY1305_SHA256:TLS_AES_128_GCM_SHA256;

    ssl_early_data on;

    add_header Alt-Svc 'h3=":443"; ma=86400';

    root /etc/nginx/html;
    index index.html;

    location / {
        try_files $uri $uri/ =404;
    }
}
```

Listing A.3 – Algoritmo utilizado para automatizar os testes com HTTP/1.1, HTTP/2 e HTTP/3

```
#!/bin/bash

SERVER="https://157.245.233.1"
OUTPUT_DIR="results"
CSV_FILE="$OUTPUT_DIR/results.csv"
META_FILE="$OUTPUT_DIR/timestamps.csv"

H2LOAD_BIN="$HOME/nghttp2/src/h2load"

if [ ! -x "$H2LOAD_BIN" ]; then
  echo "Error: h2load not found at $H2LOAD_BIN"
  exit 1
fi

SCENARIOS_HTTP_ARCHIVE=(
    "Light_Page_P25,43,payload-p25.bin"
    "Medium_Page_P50,76,payload-p50.bin"
    "Heavy_Page_P75,124,payload-p75.bin"
)

PROTOCOLS=("h1" "h2" "h3")
NUM_TESTS=5
PAUSE=0.5

C1A_STREAMS=(1)
C1A_CONCURRENCIES=(1)

C2A_STREAMS=(10 50 100)
C2A_CONCURRENCIES=(10 100 200)

mkdir -p "$OUTPUT_DIR"

```

```
33 echo "scenario_http,protocol,payload_file,connections,streams,rpc,
      test_id,start_human,start_ts,end_human,end_ts,duration_seconds,
      total_reqs,total_time,succeeded_reqs,failed_reqs,errored_reqs,
      status_2xx,status_4xx,status_5xx,req_per_sec,throughput,
      avg_latency,min_latency,max_latency,stdev_latency,avg_ttfb,
      connection_time" > "$CSV_FILE"
34 echo "scenario_http,protocol,payload_file,connections,streams,rpc,
      test_id,start_human,start_ts,end_human,end_ts,duration_seconds" >
      "$META_FILE"
35
36 parse_results() {
37   local file=$1
38   local scenario=$2
39   local protocol=$3
40   local payload=$4
41   local connections=$5
42   local streams=$6
43   local reqs_per_client=$7
44   local total_requests=$8
45   local test_id=$9
46   local start_human=${10}
47   local start_ts=${11}
48   local end_human=${12}
49   local end_ts=${13}
50   local duration=${14}
51
52   local req_line=$(grep "requests:" "$file")
53   local status_line=$(grep "status codes:" "$file")
54   local perf_line=$(grep "req/s" "$file" | head -1)
55   local timing_line=$(grep "time for request:" "$file")
56   local ttfb_line=$(grep "time to 1st byte:" "$file")
57   local connect_time_line=$(grep "time for connect:" "$file")
58
59   if [[ -z "$perf_line" ]]; then
```

```
    local errored_reqs=$(echo "$req_line" | awk '{print $12}')
    errored_reqs=${errored_reqs:-$total_requests}
    echo "$scenario,$protocol,$payload,$connections,$streams,$reqs_per_client,$test_id,$start_human,$start_ts,$end_human,$end_ts,$duration,$total_requests,0,0,0,$errored_reqs,0,0,0,0,0,0,0,0,0,0,0" >> "$CSV_FILE"
    return
  fi

  local succeeded_reqs=$(echo "$req_line" | awk '{print $8}')
  local failed_reqs=$(echo "$req_line" | awk '{print $10}')
  local errored_reqs=$(echo "$req_line" | awk '{print $12}')
  local total_time=$(echo "$perf_line" | awk '{print $3}' | tr -d ',')
  local status_2xx=0; local status_4xx=0; local status_5xx=0
  local status_parts=($status_line)
  for i in "${!status_parts[@]}"; do
    case "${status_parts[$i]}" in
      "2xx,") status_2xx=${status_parts[i-1]} ;;
      "4xx,") status_4xx=${status_parts[i-1]} ;;
      "5xx,") status_5xx=${status_parts[i-1]} ;;
    esac
  done
  local req_per_sec=$(echo "$perf_line" | awk '{print $4}' | tr -d ',')
  local throughput=$(echo "$perf_line" | awk '{print $6}')
  local min_latency=$(echo "$timing_line" | awk '{print $4}')
  local max_latency=$(echo "$timing_line" | awk '{print $5}')
  local avg_latency=$(echo "$timing_line" | awk '{print $6}')
  local stdev_latency=$(echo "$timing_line" | awk '{print $7}')
  local avg_ttfb=$(echo "$ttfb_line" | awk '{print $6}')
  local connection_time=$(echo "$connect_time_line" | awk '{print $6}')

```

```
88    echo "$scenario,$protocol,$payload,$connections,$streams,
       $reqs_per_client,$test_id,$start_human,$start_ts,$end_human,
       $end_ts,$duration,$total_requests,$total_time,$succeeded_reqs,
       $failed_reqs,$errored_reqs,$status_2xx,$status_4xx,$status_5xx,
       $req_per_sec,$throughput,$avg_latency,$min_latency,$max_latency,
       $stdev_latency,$avg_ttfb,$connection_time" >> "$CSV_FILE"
89  }
90
91  run_tests() {
92    local scenario_prefix=$1
93    local concurrencies_ref=$2
94    local streams_ref=$3
95    local thread_flag=$4
96
97    local concurrencies=("${!concurrencies_ref}")
98    local streams_list=("${!streams_ref}")
99
100   for scenario_csv in "${SCENARIOS_HTTP_ARCHIVE[@]}"; do
101     IFS=',' read -r label rpc payload_file <<< "$scenario_csv"
102
103     local scenario_name="${scenario_prefix}_${label}"
104
105     for m_orig in "${streams_list[@]}"; do
106
107       for proto in "${PROTOCOLS[@]}"; do
108
109         local effective_m=$m_orig
110         local m_for_log=$m_orig
111         case "$proto" in
112           "h1")
113             ALPN_FLAG="http/1.1"
114             effective_m=1
115             m_for_log=1
116
```

```
117             if [[ "$m_orig" != "${streams_list[0]}" ]]; then
118               continue
119             fi
120             ;;
121           "h2")
122             ALPN_FLAG="h2"
123             ;;
124           "h3")
125             ALPN_FLAG="h3"
126             ;;
127         esac
128
129         for c in "${concurrencies[@]}"; do
130
131           local n_total=$((rpc * c))
132           local n_for_log=$rpc
133
134           for i in $(seq 1 $NUM_TESTS); do
135             local outfile="$OUTPUT_DIR/${scenario_name}-${proto}-c${c
    }-m${m_for_log}-run${i}.txt"
136
137             echo "Running ${scenario_name} | Proto=${proto} | Payload
    =${payload_file} | N_per_client=${n_for_log} | Connections=${c} |
    Streams=${m_for_log} | Total_N=${n_total} | Run #${i}"
138
139             START_HUMAN=$(date -u +"%Y-%m-%d %H:%M:%S")
140             START_TS=$(date +%s)
141
142             $H2LOAD_BIN -n $n_total -c $c -m $effective_m -t
    $thread_flag --alpn-list $ALPN_FLAG "$SERVER/$payload_file" > "
    $outfile" 2>/dev/null
143
144             END_HUMAN=$(date -u +"%Y-%m-%d %H:%M:%S")
145             END_TS=$(date +%s)
```

```
146             DURATION=$((END_TS - START_TS))
147
148             echo "$scenario_name,$proto,$payload_file,$c,$m_for_log,
    $n_for_log,$i,$START_HUMAN,$START_TS,$END_HUMAN,$END_TS,$DURATION"
     >> "$META_FILE"
149             parse_results "$outfile" "$scenario_name" "$proto" "
    $payload_file" "$c" "$m_for_log" "$n_for_log" "$n_total" "$i" "
    $START_HUMAN" "$START_TS" "$END_HUMAN" "$END_TS" "$DURATION"
150
151             echo "Test $i finished (${DURATION}s). Waiting ${PAUSE}s
    ..."
152             sleep $PAUSE
153           done
154         done
155       done
156     done
157   done
158 }
159
160 echo "Starting Scenario 1a (Single-User, Serial)"
161 run_tests "1a" "C1A_CONCURRENCIES[@]" "C1A_STREAMS[@]" "1"
162
163 echo "Starting Scenario 2a (Multi-User, Parallel)"
164 run_tests "2a" "C2A_CONCURRENCIES[@]" "C2A_STREAMS[@]" "4"
165
166 echo "Results saved at:"
167 echo " - $CSV_FILE"
168 echo " - $META_FILE"
```