INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
DE SANTA CATARINA

CAMPUS SÃO JOSÉ

ANA PAULA HOOG

ANÁLISE COMPARATIVA DE PROTOCOLOS DE TRANSMISSÃO EM TEMPO REAL: UM ESTUDO SOBRE RTMP E SRT EM CONDIÇÕES DE MOBILIDADE

SÃO JOSÉ

2025

Ana Paula Hoog

Análise Comparativa de Protocolos de Transmissão em Tempo Real: Um Estudo sobre RTMP e SRT em condições de Mobilidade

Monografia apresentada ao Curso de Engenharia de Telecomunicações do Instituto Federal de Santa Catarina, para a obtenção do título de bacharel em Engenharia de Telecomunicações.

Área de concentração: Telecomunicações

Orientador: Prof. Eraldo Silveira e Silva, Dr.

São José

2025

Ana Paula Hoog

Análise Comparativa de Protocolos de Transmissão em Tempo Real: Um Estudo sobre RTMP e SRT em condições de Mobilidade

Monografia apresentada ao Curso de Engenharia de Telecomunicações do Instituto Federal de Santa Catarina, para a obtenção do título de bacharel em Engenharia de Telecomunicações.

São José, 30 de Julho de 2025.

gov.br
Documento assinado digitalmente
ERALDO SILVEIRA E SILVA
Data: 13/08/2025 08:24:17-0300
Verifique em https://validar.iti.gov.br

Prof. Eraldo Silveira e Silva, Dr.
Instituto Federal de Santa Catarina

gov.br
Documento assinado digitalmente
EDERSON TORRESINI
Data: 13/08/2025 17:49:37-0300
Verifique em https://validar.iti.gov.br

Prof. Ederson Torresini, Me.
Instituto Federal de Santa Catarina

gov.br
Documento assinado digitalmente
MARCELO MAIA SOBRAL
Data: 13/08/2025 20:22:04-0300
Verifique em https://validar.iti.gov.br

Prof. Marcelo Maia Sobral, Dr.
Instituto Federal de Santa Catarina

AGRADECIMENTOS

Sou eternamente grato aos meus pais, Eliane e João Carlos, pelo amor incondicional e pelos valores que me ensinaram, fundamentais ao longo da minha jornada acadêmica e profissional.

Ao meu querido esposo, Anderson, cuja paciência, compreensão e suporte foram meu porto seguro em meio às tempestades, agradeço por ser a luz que me motiva a seguir em frente mesmo nos momentos mais desafiadores.

Agradeço profundamente ao meu orientador, professor Eraldo. Sua dedicação e conhecimento foram essenciais para que este projeto se concretizasse, especialmente nos momentos em que eu duvidava de mim mesmo.

À minha liderança na Intelbras, Vanessa e Suzane, sou grata pela confiança e pelas oportunidades valiosas de crescimento que me proporcionaram. Vocês foram fundamentais para meu desenvolvimento profissional, muito obrigada por permitirem que eu conciliasse o trabalho com a faculdade durante todos esses anos.

Aos meus colegas de trabalho e de faculdade, agradeço por serem estrelas brilhantes em minha jornada, iluminando meu caminho com apoio e amizade. Cada um de vocês deixou uma marca indelével na minha trajetória.

A todos vocês, minha gratidão eterna por acreditarem em mim.

“A vida é feita de escolhas e desafios; é nas dificuldades que encontramos a força para seguir em frente. Após dez anos de dedicação e aprendizado, celebro não apenas a conquista do diploma, mas cada lição que essa jornada me proporcionou. — Reflexão pessoal

RESUMO

O presente trabalho tem como objetivo realizar uma análise comparativa entre os protocolos de transmissão de vídeo em tempo real Real-Time Messaging Protocol (RTMP) e Secure Reliable Transport (SRT). O estudo foca em avaliar o desempenho desses protocolos em termos de confiabilidade, abordando aspectos como taxa de perda de pacotes, frequência e duração de interrupções na transmissão (*buffering*), e erros na entrega de pacotes, considerando a mobilidade dos usuários. A metodologia do estudo envolve a simulação de cenários no software Objective Modular Network Testbed in C++ (OMNeT++), com coleta de dados sobre o comportamento dos protocolos em cenário híbrido de aplicação real com uma rede simulada. Com essa análise, espera-se evidenciar a eficiência do protocolo SRT, destacando seu potencial em aplicações de transmissão de vídeo em tempo real, especialmente para sugerir melhorias em câmeras de vigilância. A utilização desse protocolo pode proporcionar maior estabilidade, qualidade de imagem e menor latência em cenários críticos, como monitoramento remoto e vigilância de perímetros, além de otimizar o uso da largura de banda e garantir a integridade das transmissões, mesmo em condições de rede adversas.

Palavras-chave: *Streaming* de vídeo, RTMP, SRT, análise de desempenho.

ABSTRACT

This work aims to conduct a comparative analysis between the real-time video transmission protocols RTMP and SRT. The study focuses on evaluating the performance of these protocols in terms of reliability, addressing aspects such as packet loss rate, frequency and duration of transmission interruptions (buffering), and packet delivery errors, considering user mobility.

The methodology involves simulating scenarios using the OMNeT++ software, with data collection on protocol behavior in a hybrid environment combining real application with a simulated network. This analysis is expected to highlight the efficiency of the SRT protocol, emphasizing its potential in real-time video transmission applications—particularly for suggesting improvements in surveillance cameras.

The use of this protocol can provide greater stability, image quality, and lower latency in critical scenarios, such as remote monitoring and perimeter surveillance, as well as optimize bandwidth usage and ensure transmission integrity, even under adverse network conditions.

Keywords: video streaming, RTMP, SRT, perfomance analysis.

LISTA DE ILUSTRAÇÕES

Figura 1 – Áudio/vídeo na Internet 23
Figura 2 – Comparação entre os desempenhos do SSIM 31
Figura 3 – Fluxo Real-Time Protocol (RTP) 35
Figura 4 – Formato do cabeçalho de pacotes RTP 37
Figura 5 – Handshake RTMP 38
Figura 6 – Conexão RTMP 39
Figura 7 – Stream RTMP 40
Figura 8 – Visão geral do protocolo SRT 43
Figura 9 – Handshake Caller-Listener SRT 44
Figura 10 – Handshake Rendezvous 45
Figura 11 – Cenário de rede cabeada com dispositivos estáticos 51
Figura 12 – Fluxo de pacotes e traduções de endereços (NAT) entre Host1 e Host2 atravessando Router1, infraestrutura Wi-Fi (Ap1/Ap2) e RouterMobile no cenário de mobilidade. 54
Figura 13 – Cenário com mobilidade e *handover* entre dois pontos de acesso 56
Figura 14 – Cenário com mobilidade e tráfego de fundo 59
Figura 15 – Comparação de VMAF no Cenário 1. 72
Figura 16 – Comparação de VMAF no Cenário 1 - adição de delay e perda de pacotes. 73
Figura 17 – Comparação de VMAF no Cenário 1. 75
Figura 18 – Comparação de VMAF no Cenário 1, compressão em H.265 e bitrate de 4Mbps 76
Figura 19 – Comparação de VMAF no Cenário 1 - adição de delay e perda de pacotes. 77
Figura 20 – Comparação de VMAF no Cenário com um móvel e compressão H.264 e bitrate variável 78
Figura 21 – Comparação de VMAF no Cenário com um móvel e compressão H.264 e bitrate 2Mbps 79
Figura 22 – Comparação de VMAF no Cenário com um móvel e compressão H.264 e bitrate 4Mbps 81
Figura 23 – Comparação de VMAF no Cenário com um móvel e compressão H.265 e *bitrate* variável 82
Figura 24 – Comparação de VMAF no cenário UmMovel e compressão H.265 e bitrate 2Mbps 84
Figura 25 – Comparação de VMAF no cenário UmMovel e compressão H.265 e bitrate 4Mbps 85
Figura 26 – Comparação de VMAF no cenário UmMovel 86

Figura 27 – Comparação de VMAF no cenário *umMovel* e tráfego de fundo - Compressão H.264 e bitrate variável 87
Figura 28 – Comparação de VMAF no cenário doisMoveis - Compressão H.264 e *bitrate* 2 Mbps 89
Figura 29 – Comparação de VMAF no cenário *doisMoveis*. Compressão H.264 e bitrate 4Mbps 90
Figura 30 – Comparação de VMAF no cenário *doisMoveis*. Compressão H.265 e bitrate variável 91
Figura 31 – Comparação de VMAF no cenário *doisMoveis*. Compressão H.265 e bitrate 2Mbps 93
Figura 32 – Comparação de VMAF no cenário *umMovel* e tráfego de fundo. Compressão H.265 e bitrate 4Mbps 94
Figura 33 – Comparação de VMAF no cenário *umMovel* e tráfego de fundo 95

LISTA DE QUADROS

Quadro 1 – Interpretação dos valores de PSNR 30
Quadro 2 – Classificação da Qualidade de Vídeo com base na pontuação VMAF . 32
Quadro 3 – Comparação entre SRT e RTMP. 46

LISTA DE TABELAS

Tabela 1 – Interpretação dos valores de SSIM 32
Tabela 2 – Regras de NAT configuradas para permitir a comunicação bidirecional entre transmissor (host1) e receptor (host2) 53
Tabela 3 – Regras de Network Address Translation (NAT) configuradas para permitir a comunicação bidirecional entre transmissor (host1) e receptor (host2) no cenário com mobilidade. Endereços "antes" representam o cabeçalho IP ao chegar na interface indicada; "depois" representam o cabeçalho reescrito ao sair após a regra de NAT. 54
Tabela 4 – Fatores experimentais e seus níveis de variação 59
Tabela 5 – Cenário 1 – Rede cabeada (sem mobilidade e sem tráfego de fundo) . . 60
Tabela 6 – Cenário 2 – Rede com mobilidade (sem tráfego de fundo) 61
Tabela 7 – Cenário 3 – Rede com mobilidade e tráfego de fundo 61
Tabela 8 – Comparação entre os arquivos H.264 e H.265 (mesma resolução) 62
Tabela 9 – Métricas de qualidade de vídeo para o Cenário 1 – H.264 a 4 Mbps . . 72
Tabela 10 – Métricas de desempenho de rede e QoE para o Cenário 1 – H.264 a 4 Mbps 72
Tabela 11 – Métricas de qualidade de vídeo para o Cenário 1 – adição de delay e perda de pacotes 73
Tabela 12 – Métricas de desempenho de rede e Cenário – H.264 a 4 Mbps com delay 74
Tabela 13 – Métricas de qualidade de vídeo para o Cenário 1 – H.265 a 2 Mbps . . 75
Tabela 14 – Métricas de desempenho de rede para o Cenário 1 – H.265 a 2 Mbps . 75
Tabela 15 – Métricas de qualidade de vídeo para o Cenário 1 – H.265 a 4 Mbps . . 76
Tabela 16 – Métricas de desempenho de rede e QoE para o Cenário 1 – H.265 a 4 Mbps 76
Tabela 17 – Comparação de qualidade de vídeo para o cenário *UmMovel* – H.264 . 78
Tabela 18 – Métricas de transmissão para o cenário *UmMovel* – H.264 78
Tabela 19 – Métricas de qualidade de vídeo para o Cenário UmMovel – H.264 a 2 Mbps 79
Tabela 20 – Métricas de desempenho de rede para o Cenário UmMovel – H.264 a 2 Mbps 80
Tabela 21 – Métricas de qualidade de vídeo para os protocolos no cenário *UmMovel* com H.264 a 4 Mbps 81
Tabela 22 – Métricas de desempenho de rede para os protocolos no cenário *UmMovel* com H.264 a 4 Mbps 81
Tabela 23 – Métricas de qualidade de vídeo para os protocolos no cenário *UmMovel* com compressão H.265 82

Tabela 24 – Métricas de desempenho de rede para os protocolos no cenário *UmMovel* com compressão H.265 83
Tabela 25 – Métricas de qualidade de vídeo para o cenário UmMovel – H.265 a 2 Mbps 84
Tabela 26 – Métricas de desempenho de rede cenário UmMovel – H.265 a 2 Mbps . 84
Tabela 27 – Métricas de qualidade de vídeo para o cenário UmMovel – H.265 a 4 Mbps 85
Tabela 28 – Métricas de desempenho de rede para o cenário UmMovel – H.265 a 4 Mbps 85
Tabela 29 – Métricas de qualidade de vídeo para o cenário *umMovel* e tráfego de fundo – H.264 87
Tabela 30 – Métricas de desempenho de rede com cenário *umMovel* e tráfego de fundo – H.264 87
Tabela 31 – Métricas de qualidade de vídeo para o cenário doisMoveis – H.264 a 2 Mbps 89
Tabela 32 – Métricas de desempenho de rede cenário *doisMoveis* – H.264 a 2 Mbps 89
Tabela 33 – Métricas de qualidade de vídeo para o cenário *doisMoveis* – H.264 a 4 Mbps 90
Tabela 34 – Métricas de desempenho de rede cenário *doisMoveis* – H.264 a 4 Mbps 90
Tabela 35 – Métricas de qualidade de vídeo para o cenário *doisMoveis*. Compressão H.265 e bitrate variável 92
Tabela 36 – Métricas de desempenho de rede cenário *doisMoveis*. Compressão H.265 e bitrate variável 92
Tabela 37 – Métricas de qualidade de vídeo para o cenário *doisMoveis*. Compressão H.265 e bitrate 2Mbps 93
Tabela 38 – Métricas de desempenho de rede no cenário *doisMoveis*. Compressão H.265 e bitrate 2Mbps 93
Tabela 39 – Métricas de qualidade de vídeo para o cenário *umMovel* e tráfego de fundo – H.265 a 4 Mbps 94
Tabela 40 – Métricas de desempenho de rede cenário *umMovel* e tráfego de fundo – H.265 a 4 Mbps 94
Tabela 41 – Resumo comparativo qualitativo entre os protocolos 98
Tabela 42 – Resultados para o Cenário 1 – H.264 a 2 Mbps 118
Tabela 43 – Resultados para o Cenário 1 – H.264 a 4 Mbps 118
Tabela 44 – Métricas de qualidade de vídeo e desempenho de rede para o Cenário 1 – H.264 a 4 Mbps - com delay e perdas 118
Tabela 45 – Resultados para o Cenário 1 – H.265 a 2 Mbps 118
Tabela 46 – Resultados para o Cenário 1 – H.265 a 4 Mbps 118
Tabela 47 – Métricas de desempenho para o cenário *UmMovel* com compressão H.264 119

Tabela 48 – Métricas de desempenho para o cenário *UmMovel* com compressão H.264 a 2 Mbps 119
Tabela 49 – Métricas do cenário *UmMovel* – H.264 a 4 Mbps 119
Tabela 50 – Métricas do cenário *UmMovel* – H.265 119
Tabela 51 – Métricas do cenário *UmMovel* – H.265 a 2 Mbps 120
Tabela 52 – Métricas do cenário *UmMovel* – H.265 a 4 Mbps 120
Tabela 53 – Métricas para o cenário DoisMoveis – H.264 120
Tabela 54 – Métricas para o cenário DoisMoveis – H.264 – 2 Mbps 120
Tabela 55 – Métricas do cenário DoisMoveis – H.264 a 4 Mbps 121
Tabela 56 – Métricas para o cenário DoisMoveis – H.265 121
Tabela 57 – Métricas para o cenário DoisMoveis – H.265 – 2MB 121
Tabela 58 – Métricas para o cenário DoisMoveis – H.265 – 4 Mbps 121

LISTA DE CÓDIGOS

Código 3.1 – Configuração das interfaces TAP 51
Código 3.2 – Configuração das interfaces eth0 e eth1 do Router 51
Código 3.3 – Configuração da tabela de roteamento do router 52
Código 3.4 – Configuração da tabela de roteamento do router e routerMobile . . . 55
Código 3.5 – Configuração do routerMobile e a mobilidade 57
Código 3.6 – Execução do cenário OMNeT++ 63
Código 3.7 – Chamada do script de transmissão/recepção 64
Código 3.8 – Fluxo resumido do script de transmissão/recepção de vídeo 65

LISTA DE ABREVIATURAS E SIGLAS

ABR Adaptive Bitrate Streaming.

ACK acknowledgemen.

ARQ Automatic Repeat reQuest.

AVC Advanced Video Coding.

FEC Forward Error Correction.

HEVC High Efficiency Video Coding.

HLS HTTP Live Streaming.

ICMP Internet Control Message Protocol.

IoT Internet of Things.

ITU União Internacional de Telecomunicações.

MOS Mean Opinion Score.

NAK Negative Acknowledgement.

NAT Network Address Translation.

OMNeT++ Objective Modular Network Testbed in C++.

PCM Pulse Code Modulation.

PSNR Peak Signal-to-Noise Ratio.

QoE Quality of Experience.

QoS Quality of Service.

RIST Reliable Internet Stream Transport.

RTCP Real-Time Control Protocol.

RTMP Real-Time Messaging Protocol.

RTP Real-Time Protocol.

RTSP Real Time Streaming Protoco.

SRT Secure Reliable Transport.

SSIM Structural Similarity Index.

TCP Transmission Control Protocol.

UDP User Datagram Protocol.

UDT UDP-based Data Transfer Protocol.

VLAN Virtual Local Area Network.

VMAF Video Multi-Method Assessment Fusion.

VoIP Voice Over Internet Protocol.

WLAN Wireless Local Area Network.

SUMÁRIO

1 INTRODUÇÃO 19
1.1 MOTIVAÇÃO 19
1.2 CONTEXTO PESSOAL E PROFISSIONAL 20
1.3 OBJETIVO GERAL 21
1.4 OBJETIVOS ESPECÍFICOS 21
1.5 DELIMITAÇÃO DO ESTUDO 21

2 FUNDAMENTAÇÃO TEÓRICA 23
2.1 VISÃO GERAL DE APLICAÇÕES E PROTOCOLOS MULTIMÍDIA . . . 23
2.2 CODECS 24
2.2.1 Escolha dos Codecs 25
2.2.2 H.264 26
2.2.3 H.265 26
2.3 QOS / QOE 27
2.3.1 QoS em Redes 27
2.3.2 Métricas de Qualidade de Experiência (QoE) para Transmissão de Vídeo 28
2.3.3 Modelos de Avaliação de QoE 29
2.3.4 PSNR 29
2.3.5 SSIM 30
2.3.6 VMAF 32
2.3.7 Cálculo das Métricas Avaliadas 32
2.3.8 Retransmissão 33
2.3.9 Tempo total de Sessão 34
2.3.10 Jitter 34
2.3.11 Buffering 35
2.4 RTP 35
2.5 PROTOCOLO RTMP 37
2.5.1 Estabelecimento de Conexão no RTMP 38
2.5.2 Handshake 38
2.5.3 Conexão 39
2.5.4 Stream 40
2.6 PROTOCOLO SRT 41
2.6.1 Estabelecimento de conexão no SRT 42
2.6.2 *Buffer* 45
2.7 ANÁLISE COMPARATIVA ENTRE SRT E RTMP 46

2.8 REVISÃO DE TRABALHOS SOBRE RTMP E SRT COM FOCO EM AVALIAÇÃO DE DESEMPENHO . . . 46

3 DESCRIÇÃO DE CENÁRIOS E ASPECTOS METODOLÓGICOS 49
3.1 SIMULAÇÃO DE MOBILIDADE COM OMNET++ . . . 49
3.2 CENÁRIO DE REDE CABEADA (ESTÁTICO) . . . 50
3.3 CENÁRIO DE MOBILIDADE . . . 53
3.4 CONSIDERAÇÕES SOBRE A EXECUÇÃO DOS EXPERIMENTOS . . . 59
3.4.1 Fatores e Métricas Consideradas . . . 59
3.4.2 Execução . . . 63
3.4.3 Uso das ferramentas de vídeo e coleta de dados . . . 66
3.4.4 Extração das métricas de qualidade e desempenho . . . 66
3.4.4.1 Cálculo do Jitter . . . 67

4 ANÁLISE . . . 71
4.1 ANÁLISE SEM MOBILIDADE . . . 71
4.1.1 Compressão de vídeo H.264 e bitrate de 4Mbps . . . 71
4.1.2 Compressão de vídeo H.265 e bitrate de 2Mbps . . . 74
4.1.3 Compressão de vídeo H.265 e bitrate de 4Mbps . . . 75
4.1.4 Comparação de VMAF no Cenário 1 . . . 76
4.2 ANÁLISE COM MOBILIDADE . . . 77
4.2.1 Mobilidade de um único host . . . 77
4.2.1.1 Cenário umMovel - Compressão de vídeo H.264 e bitrate variável . . . 77
4.2.1.2 Cenário umMovel - Compressão de vídeo H.264 e bitrate 2Mbps . . . 79
4.2.1.3 Cenário umMovel - Compressão de vídeo H.264 e bitrate 4 Mbps . . . 80
4.2.1.4 Cenário umMovel - Compressão de vídeo H.265 e bitrate variável . . . 81
4.2.1.5 Cenário umMovel - Compressão de vídeo H.265 e bitrate 2Mbps . . . 83
4.2.1.6 Cenário umMovel - Compressão de vídeo H.265 e bitrate 4Mbps . . . 84
4.2.2 Comparação de VMAF no Cenário *umMovel* . . . 85
4.2.3 Mobilidade de um único host com tráfego de fundo . . . 86
4.2.3.1 Cenário doisMoveis - Compressão de vídeo H.264 e bitrate variável . . . 86
4.2.3.2 Cenário doisMoveis - Compressão de vídeo H.264 e bitrate 2Mbps . . . 88
4.2.3.3 Cenário doisMoveis - Compressão de vídeo H.264 e bitrate 4Mbps . . . 89
4.2.3.4 Cenário doisMoveis - Compressão de vídeo H.265 e bitrate variável . . . 91
4.2.3.5 Cenário doisMoveis - Compressão de vídeo H.265 e bitrate 2Mbps . . . 92
4.2.3.6 Cenário doisMoveis - Compressão de vídeo H.265 e bitrate 4Mbps . . . 93
4.2.4 Comparação de VMAF no Cenário umMovel e tráfego de fundo . 95

5 CONCLUSÕES . . . 96
5.1 ANÁLISE DOS RESULTADOS . . . 96

5.1.1 Impacto da compressão de vídeo e taxa de transferência 97
5.2 EFEITOS DA MOBILIDADE E DO TRÁFEGO DE FUNDO 97
5.3 ANÁLISE COMPARATIVA 98
5.4 CONCLUSÃO GERAL 99
5.5 AVALIAÇÃO DO ATENDIMENTO AOS OBJETIVOS 99
AVALIAÇÃO DO ATENDIMENTO AOS OBJETIVOS 99

6 TRABALHOS FUTUROS 101
6.1 PADRONIZAÇÃO NOS TESTES DE MOBILIDADE 101
6.2 ANÁLISE CRUZADA COM LOGS DO OMNET++ 101
6.3 DIAGNÓSTICO DO TRAVAMENTO DO RTMP APÓS EXECUÇÕES REPETIDAS 102
6.4 EXPANSÃO PARA NOVOS PROTOCOLOS E REDES HETEROGÊNEAS 102
6.5 AVALIAÇÃO DA QUALIDADE DE ÁUDIO 103
6.6 CONCLUSÃO 103

Referências 104

Apêndice A: Execução dos Testes de Simulação no OMNeT++ . 108

Apêndice B: Scripts de Transmissão e Análise 114

Apêndice C: Tabela de Resultados dos Testes 118

1 INTRODUÇÃO

O avanço da tecnologia de transmissão de vídeo e a crescente demanda por conteúdos em tempo real têm impulsionado o desenvolvimento de protocolos especializados para a transmissão de áudio e vídeo em redes de computadores. Protocolos como RTMP e SRT são fundamentais para garantir uma entrega eficiente e estável de áudio e vídeo em diferentes tipos de rede. Cada um desses protocolos possui características próprias que os tornam mais adequados para diferentes usos, como transmissões ao vivo de baixa latência e vídeos sob demanda em redes mais instáveis.

A transmissão de vídeo em tempo real apresenta desafios em termos de latência, qualidade e confiabilidade. Protocolos tradicionais como o RTMP, amplamente utilizados em plataformas de *streaming* ao vivo, enfrentam limitações em redes instáveis. Mais recentemente, o protocolo SRT surgiu como uma solução robusta para lidar com as inconsistências da internet, oferecendo alta qualidade de transmissão com mecanismos avançados de correção de erros.

1.1 MOTIVAÇÃO

No Brasil, a demanda por câmeras de segurança tem aumentado devido à preocupação com a segurança residencial, empresarial e pública. As cidades brasileiras estão cada vez mais adotando sistemas de vigilância em resposta ao aumento da criminalidade e à necessidade de gestão de tráfego e infraestrutura urbanas. Isso tem impulsionado não apenas o aumento nas vendas de câmeras, mas também o desenvolvimento de tecnologias de vigilância mais sofisticadas a preços competitivos(MORDOR INTELLIGENCE, 2023).

Conforme Mordor Intelligence (2023) o investimento em sistemas de vigilância é visto cada vez mais como uma necessidade em muitos setores, e as tendências indicam que o mercado brasileiro continuará a crescer, com tecnologias cada vez mais avançadas se tornando acessíveis para um espectro mais amplo de consumidores.

Ao instalar uma câmera de vigilância com necessidade de monitoramento remoto, o usuário espera obter imagem em tempo real, sem interrupções ou travamentos, além de áudio de boa qualidade, capaz de captar claramente os sons do ambiente.

Essa demanda crescente impõe a necessidade do uso de protocolos eficientes, capazes de gerenciar a transmissão de áudio e vídeo, mantendo a qualidade, a sincronia e a estabilidade das informações para os usuários. O acesso a vídeos por meio de dispositivos móveis representa um desafio adicional para esses sistemas, o que motiva o desenvolvimento deste trabalho. Neste estudo, considera-se inicialmente a mobilidade em redes Wireless Local Area Network (WLAN), com foco em cenários de transição entre

pontos de acesso (*handover*). Essa escolha se justifica pela ampla utilização do Wi-Fi em aplicações de vídeo sob demanda e transmissões ao vivo, além da viabilidade prática de emulação e simulação em ambientes controlados. Dessa forma, busca-se caracterizar o impacto da mobilidade sobre protocolos de transmissão de vídeo em tempo real em condições realistas, mas ainda reprodutíveis em laboratório.

Outra demanda em ascensão é o serviço de armazenamento em nuvem para sistemas de vigilância, que oferece diversas vantagens significativas, conforme destacado em (SISTEMA IRIS, 2023). Entre os benefícios, estão a mobilidade de acesso e o compartilhamento seguro, permitindo que usuários acessem as imagens de qualquer lugar com acesso à internet. Isso aumenta a flexibilidade e a segurança na gestão de sistemas de vigilância. Essas características fazem do armazenamento em nuvem uma solução mais eficiente para empresas que buscam modernizar seus sistemas de vigilância, garantindo a integridade e a disponibilidade dos dados de forma contínua e segura.

A análise comparativa entre os protocolos RTMP e SRT é pertinente, considerando que o RTMP ainda é amplamente utilizado em diversas plataformas de grande alcance, como *YouTube* e *Facebook* devido à sua consolidação no mercado. Por outro lado, o SRT é uma tecnologia mais recente, que vem ganhando destaque por oferecer soluções mais robustas contra problemas de instabilidade e segurança, especialmente em redes públicas. Esta pesquisa tem como objetivo investigar as vantagens e limitações de cada protocolo, com ênfase na capacidade de ambos em manter a qualidade da transmissão de vídeo em contextos de mobilidade — adequado para aplicações nas áreas de entretenimento, segurança e comunicação corporativa.

1.2 CONTEXTO PESSOAL E PROFISSIONAL

A escolha do tema deste trabalho está diretamente relacionada à minha experiência profissional na área de CFTV[1] e soluções de transmissão de vídeo em tempo real. Durante minha atuação no setor de pós-venda em uma empresa de tecnologia de segurança, tive contato frequente com cenários práticos de instalação, configuração e suporte de sistemas de monitoramento baseados em câmeras IP, DVRs[2] e NVRs[3] .

Nesse contexto, tornou-se evidente a importância dos protocolos de aplicação de vídeo para garantir a qualidade de experiência do usuário, especialmente em situações de

[1] CFTV (Circuito Fechado de Televisão) refere-se a sistemas de videomonitoramento nos quais as câmeras transmitem imagens para dispositivos de gravação e visualização restritos a um conjunto definido de usuários, diferindo de transmissões abertas ou públicas.

[2] DVR (*Digital Video Recorder*) é um dispositivo dedicado à gravação digital de imagens provenientes de câmeras de videomonitoramento, geralmente utilizando a tecnologia analógica. Ele converte os sinais recebidos em formato digital, permitindo armazenamento em disco rígido, reprodução posterior e acesso remoto às imagens.

[3] NVR (*Network Video Recorder*) é um dispositivo de gravação voltado para câmeras IP, que recebem e armazenam diretamente os fluxos digitais transmitidos pela rede.

mobilidade, instabilidade de rede e necessidade de transmissão em tempo real. Problemas relacionados à latência, interrupções na reprodução e degradação da qualidade do vídeo foram recorrentes, evidenciando a relevância de estudos comparativos entre protocolos amplamente utilizados no mercado, como o RTMP e o SRT.

Assim, este trabalho surge como uma oportunidade de aproximar a prática profissional do ambiente acadêmico, permitindo não apenas a análise técnica dos protocolos, mas também a reflexão sobre sua aplicabilidade em contextos reais de segurança, entretenimento e comunicação.

1.3 OBJETIVO GERAL

O objetivo geral desta pesquisa é comparar os protocolos RTMP e SRT em condições de mobilidade, utilizando métricas associadas às interrupções na reprodução da transmissão e à qualidade de vídeo.

1.4 OBJETIVOS ESPECÍFICOS

Para alcançar o objetivo geral, os seguintes objetivos específicos foram definidos:

1. caracterizar cenários representativos de transmissão de áudio e vídeo, considerando a capacidade variável da rede e a mobilidade do consumidor do fluxo;
2. desenvolver um cenário híbrido que combine aplicação real com rede simulada, permitindo a exploração de modelos de mobilidade no ambiente de transmissão;
3. identificar os principais fatores que impactam o desempenho dos protocolos em contextos de mobilidade, como variações de latência, perda de pacotes e *jitter*;
4. comparar a latência e a qualidade de vídeo oferecidas por cada protocolo em diferentes condições de mobilidade, utilizando métricas objetivas de avaliação;
5. avaliar a viabilidade de utilização de cada protocolo em diferentes cenários de transmissão, como *streaming* ao vivo e vigilância.

1.5 DELIMITAÇÃO DO ESTUDO

É importante destacar que esta pesquisa considera transmissões de vídeo unidirecionais, típicas de aplicações de *live streaming* e vídeo sob demanda, sem contemplar cenários de interação bidirecional. Além disso, a mobilidade é analisada apenas no lado consumidor da transmissão, refletindo o comportamento de usuários finais em redes sem fio, enquanto o servidor permanece fixo, de forma compatível com infraestruturas reais de distribuição de vídeo.

Embora existam outros protocolos que poderiam ser explorados em transmissões de vídeo em tempo real, como o SCTP[4], o QUIC[5] e o WebRTC[6], este trabalho concentrou-se nos protocolos RTMP, SRT e RTP por três razões principais: (i) são amplamente utilizados em cenários práticos de transmissão de vídeo, incluindo plataformas de *streaming* e sistemas de videovigilância; (ii) possuem suporte nativo em ferramentas como o FFmpeg[7], viabilizando experimentação em ambientes de emulação e simulação; e (iii) representam abordagens distintas que permitem análises comparativas: RTMP, baseado em TCP, apresenta restrições no uso de *codecs* modernos como o H.265; SRT, baseado em User Datagram Protocol (UDP), emprega mecanismos de correção de erros para maior resiliência; e RTP, também baseado em UDP, oferece suporte direto ao H.265 e é amplamente adotado em transmissões de baixa latência, servindo como contraponto adicional na avaliação da relação entre Quality of Service (QoS) e Quality of Experience (QoE) em condições de mobilidade.

[4] O SCTP (*Stream Control Transmission Protocol*) (STEWART, 2007) é um protocolo de transporte da camada 4 projetado para aplicações que exigem transmissão confiável de mensagens. Ele combina características do TCP, como confiabilidade e controle de congestionamento, com recursos adicionais como *multistreaming* (múltiplos fluxos independentes em uma mesma conexão) e *multihoming* (suporte a múltiplos endereços IP em um único *endpoint*). Embora promissor para aplicações multimídia, sua adoção é mais restrita em comparação ao TCP e UDP.

[5] O QUIC (*Quick UDP Internet Connections*) (LANGLEY et al., 2017) é um protocolo de transporte desenvolvido pelo Google que opera sobre UDP, oferecendo confiabilidade, controle de congestionamento e segurança nativa (TLS 1.3). Ele foi projetado para reduzir a latência de conexões em aplicações de *streaming* e navegação web, sendo atualmente a base para o HTTP/3. Apesar de seu potencial em transmissões multimídia, sua utilização ainda é mais comum em serviços web e menos frequente em plataformas de videovigilância.

[6] O WebRTC (*Web Real-Time Communication*) (BERGKVIST et al., 2012) é um conjunto de protocolos e APIs projetado para comunicação multimídia em tempo real diretamente entre navegadores e aplicações. Baseia-se em UDP, mas pode utilizar Transmission Control Protocol (TCP) quando necessário, e inclui mecanismos de controle de congestionamento, NAT traversal (ICE, STUN, TURN) e segurança ponta a ponta (DTLS-SRTP). Embora amplamente adotado em videoconferências e aplicações interativas, seu foco difere das transmissões unidirecionais contínuas típicas de sistemas de *streaming* e videovigilância.

[7] O FFmpeg é um *framework* de processamento de áudio, vídeo e outros formatos de mídia, com amplo suporte a codecs (ABOUT..., s.d.).

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo apresenta-se a fundamentação teórica do estudo, abordando em detalhes os protocolos utilizados no cenário proposto, bem como uma descrição das ferramentas empregadas para a análise. Inicialmente, é apresentada uma visão geral sobre aplicações multimídia. Em seguida, descrevem-se os *codecs* utilizados, os protocolos investigados e as métricas adotadas para avaliar a Qualidade de Experiência (QoE). Por fim, apresenta-se um breve levantamento de trabalhos relacionados, de modo a situar a pesquisa no contexto acadêmico e tecnológico existente.

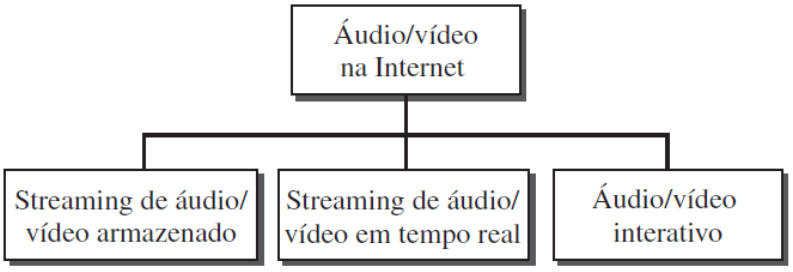
2.1 VISÃO GERAL DE APLICAÇÕES E PROTOCOLOS MULTIMÍDIA

A transmissão de mídia em tempo real via *internet* tornou-se uma aplicação crucial em vários contextos, como entretenimento, videoconferência, educação a distância e, principalmente, segurança. Este cenário é intensificado pelo crescente uso de dispositivos Internet of Things (IoT), como câmeras de vídeo, que geram grandes quantidades de dados de *streaming* que estão sendo produzidos em alta velocidade (ALI et al., 2020).

Segundo Kurose e Ross (2021), um vídeo é uma sequência de imagens, em geral exibidas a uma velocidade constante, com 24 ou 30 imagens por segundo. Além disso, o vídeo pode ser compactado para qualquer taxa de bits desejada; quanto mais alta a taxa de bits, melhor a qualidade da imagem e, em geral, melhor a experiência de exibição do usuário.

Os serviços de áudio e vídeo podem ser categorizados em três tipos principais: *streaming* de conteúdo armazenado, *streaming* em tempo real e *streaming* interativo, conforme ilustrado na Figura 1, segundo (KUROSE; ROSS, 2021).

Figura 1 – Áudio/vídeo na Internet



Fonte: (KUROSE; ROSS, 2021).

No *streaming* de conteúdo armazenado, os arquivos são pré-gravados, comprimidos e armazenados em um servidor para que os usuários possam começar a assistir ou ouvir

quase imediatamente após iniciar o download, conhecido também como áudio/vídeo sob demanda. Exemplos incluem músicas, filmes, séries e outros conteúdos que são acessados frequentemente por demanda. Na segunda categoria, o *streaming* de áudio/vídeo em tempo real permite aos usuários acessar transmissões ao vivo pela internet. Na terceira categoria encontram-se o áudio/vídeo interativo, que são os casos de ligações de voz e videochamadas via *Whatsapp*.

Conforme Comer (2015), muitas aplicações de áudio e vídeo são consideradas em tempo real, pois requerem que a transmissão e reprodução ocorram dentro de prazos específicos, ainda que não explícitos ao usuário. Em situações como chamadas telefônicas e vídeos interativos, é essencial que áudios e vídeos sejam entregues com baixa latência, pois atrasos podem comprometer significativamente a qualidade da experiência. Além disso, é necessário que a transmissão mantenha a ordem correta dos pacotes e preserve o sincronismo original dos sinais, garantindo a inteligibilidade e a fidelidade da informação transmitida. Por exemplo, se um sinal é amostrado a cada 125 µs, o receptor deve reproduzi-lo na mesma taxa, respeitando o prazo característico das aplicações em tempo real.

Como elementos fundamentais para aplicações multimídia, podemos destacar a digitalização e os *codecs*, que transformam e otimizam sinais para uso digital; compressores, que reduzem o tamanho dos dados mantendo a qualidade; protocolos, que regulam a transmissão de dados; e os mecanismos de QoS e QoE, que asseguram a eficiência da entrega e a satisfação do usuário, respectivamente. Cada um desses componentes será explorado mais detalhadamente a seguir.

2.2 CODECS

Antes de transmitir voz ou vídeo por uma rede de dados, é essencial utilizar um dispositivo conhecido como codificador/decodificador (codec), que converte sinais analógicos em formatos digitais (COMER, 2015). O tipo mais comum, o codificador de forma de onda, avalia a amplitude do sinal em intervalos regulares, transformando cada ponto medido em um valor digital (um número inteiro). Do outro lado, no receptor, o *codec* recebe uma série de inteiros e os transforma novamente em um sinal analógico contínuo que reflete os valores digitais.

Diversos padrões de codificação digital existem, com compromissos entre a qualidade do som e o tamanho dos arquivos digitais. Por exemplo, o sistema telefônico padrão utiliza a Modulação por Código de Pulso (Pulse Code Modulation (PCM)) , que captura 8 bits a cada 125 microssegundos, resultando em uma taxa de 64 Kbps. Essa taxa de amostragem gera uma quantidade considerável de dados, onde um arquivo de áudio de 128 segundos pode ocupar até um megabyte de espaço.

Para reduzir o volume de dados produzidos pela codificação digital, Comer (2015) explica que pode-se diminuir a frequência de amostragem, reduzir o número de bits por amostra, ou aplicar esquemas de compressão digital que diminuem o tamanho do arquivo final. Cada uma dessas técnicas tem suas desvantagens: reduzir a frequência de amostragem ou o número de bits pode comprometer a qualidade do áudio, limitando a capacidade de reproduzir uma gama completa de sons. Já a compressão, apesar de eficaz na redução do tamanho do arquivo, pode introduzir atrasos no processamento, exigindo CPUs mais potentes para uma compactação mais eficiente ou resultando em maiores latências, o que pode ser problemático para comunicações em tempo real, mas menos crítico para arquivos armazenados.

Os *codecs*, como H.264 e H.265, desempenham um papel crítico na compressão de vídeos, permitindo a transmissão de imagens de alta definição com uso otimizado da largura de banda disponível. A importância da compressão de dados e das taxas de bits ajustáveis é destacada pelo fato de que o *streaming* de vídeo pode variar de 100 kbits/s para conteúdo de baixa qualidade até mais de 4 Mbits/s para filmes em alta definição, chegando a mais de 10 Mbits/s para vídeos em qualidade 4K, conforme mencionado por Kurose e Ross (2021).

A compressão de vídeo permite a criação de múltiplas versões do mesmo conteúdo, adaptando-se a diversas condições de rede e larguras de banda, otimizando o uso dos recursos disponíveis e melhorando a experiência do usuário em ambientes com conexões variáveis Laghari et al. (2023). Uma das técnicas utilizadas para a redução do tamanho dos arquivos é a compressão intraquadro e interquadro, que diminui significativamente a quantidade de dados ao armazenar e transmitir apenas as alterações entre os quadros, em vez de cada quadro completo. Entre os padrões mais populares de compressão de vídeo estão o H.264/AVC e o H.265/HEVC. Neste trabalho, esses *codecs* serão utilizados e, portanto, serão detalhados a seguir.

2.2.1 Escolha dos Codecs

A seleção dos *codecs* de vídeo utilizados nos experimentos foi orientada por sua ampla adoção em aplicações de transmissão em tempo real e pelo equilíbrio entre eficiência de compressão e compatibilidade prática. Foram escolhidos o H.264/AVC e o H.265/HEVC, ambos suportados nativamente pelo *framework* FFmpeg e amplamente utilizados em soluções comerciais de videovigilância, plataformas de *streaming* e dispositivos de consumo.

O *codec* H.264 é o mais consolidado no setor de CFTV, estando presente na grande maioria das câmeras de vigilância atualmente em operação, devido ao suporte universal em *softwares* de gerenciamento e DVRs/NVRs. Já o H.265 representa a evolução tecnológica desse padrão, oferecendo ganhos de eficiência de compressão de 30–50% em relação ao H.264 para a mesma qualidade perceptual, o que o torna particularmente atrativo em

cenários de mobilidade ou de largura de banda restrita.

Essa escolha garante que os resultados obtidos reflitam situações práticas do mercado de monitoramento por vídeo e de transmissões ao vivo, permitindo avaliar o impacto dos protocolos em *codecs* efetivamente utilizados em ambientes reais.

2.2.2 H.264

O H.264 Advanced Video Coding (AVC) é um *codec* de compressão de vídeo amplamente usado, conhecido por sua alta eficiência e qualidade em uma variedade de aplicações de *streaming* e gravação. Ele foi desenvolvido para oferecer uma compressão eficaz, mantendo uma alta qualidade de imagem e suportando resoluções de até 4K. Um dos principais benefícios do H.264 é sua capacidade de reduzir significativamente o tamanho do arquivo de vídeo, o que é essencial para *streaming* de vídeo em redes com largura de banda limitada. Além disso, o H.264 é compatível com uma vasta gama de dispositivos e plataformas, tornando-o indicado para aplicações de vídeo digitais onde a interoperabilidade é crucial (ADDU; POTUVARDANAM, 2014).

2.2.3 H.265

De acordo com o Xu et al. (2018), High Efficiency Video Coding (HEVC), também conhecido como H.265, é um padrão de codificação de vídeo avançado que oferece melhorias significativas em relação ao seu antecessor, o H.264. Essas melhorias incluem maior eficiência de codificação, possibilitando vídeos de alta qualidade com aproximadamente a metade da taxa de bits, suporte robusto para resoluções Ultra HD (4K e 8K) e técnicas de predição intra-quadro e inter-quadro aprimoradas. Além disso, o HEVC adapta-se a diferentes ambientes de rede, tornando-o uma solução adequada para a crescente demanda por vídeos de alta resolução.

Além da eficiência de compressão, tanto o H.264 quanto o H.265 incorporam mecanismos voltados à resiliência frente a erros de transmissão. Entre eles destacam-se a detecção de erros e a ocultação de erros (*error concealment*), técnica em que o decodificador tenta reconstruir trechos corrompidos ou perdidos a partir de macroblocos vizinhos ou quadros anteriores. Essa funcionalidade permite manter a continuidade da reprodução, ainda que com degradações visuais momentâneas. Em cenários mais críticos, pode-se ainda empregar Forward Error Correction (FEC), adicionando redundância ao fluxo de vídeo para possibilitar a recuperação de dados sem necessidade de retransmissão, embora ao custo de maior sobrecarga de rede. Dessa forma, os codecs diferem não apenas em termos de eficiência de compressão, mas também em sua tolerância a perdas, o que influencia diretamente a qualidade percebida em transmissões sujeitas a *jitter*, atrasos ou congestionamento.

2.3 QOS / QOE

Segundo Juluri, Tamarapalli e Medhi (2015), a QoE representa a percepção do usuário quanto à qualidade do serviço prestado, sendo influenciada por fatores como desempenho do sistema, contexto de uso, características individuais do usuário e conteúdo transmitido. Por outro lado, a QoS refere-se a parâmetros técnicos de rede — como vazão, atraso, *jitter* e perda de pacotes — que impactam diretamente o desempenho das aplicações. Enquanto a QoS oferece uma avaliação objetiva baseada em métricas da infraestrutura, a QoE traduz esses resultados para a perspectiva subjetiva do usuário, refletindo sua satisfação efetiva com a transmissão de áudio e vídeo.

2.3.1 QoS em Redes

QoS em redes refere-se à capacidade de garantir a performance de tráfego prioritário em uma rede, controlando e gerenciando o fluxo de dados para evitar a degradação de qualidade em aplicações críticas (FORTINET, 2023). A QoS permite que administradores de rede priorizem certos tipos de tráfego, o que é crucial para aplicações que requerem altos níveis de desempenho e baixa latência, como chamadas Voice Over Internet Protocol (VoIP) e conferências de vídeo. A ausência de QoS pode levar a uma redução na qualidade das aplicações sensíveis a atrasos, pois os dados serão tratados igualmente.

Para se ter um fluxo de mídia de qualidade, são atribuídas 4 características:

1. Confiabilidade: garantir que os dados sejam entregues corretamente e sem erros.
2. Latência: refere-se ao tempo que leva para os dados viajarem da origem ao destino.
3. *Jitter*: é a variação no atraso de pacotes recebidos em uma transmissão de dados.
4. Taxa de transmissão mínima: valor mínimo de capacidade da rede necessário para suportar a quantidade de dados transmitidos sem degradação perceptível de qualidade.

Conforme Comer (2015), o *jitter* é uma variabilidade no tempo de trânsito que pode afetar adversamente a qualidade da transmissão de dados, especialmente em aplicações sensíveis ao tempo, como transmissões de áudio e vídeo. Esse fenômeno é causado por atrasos variáveis nos pacotes de dados ao serem transmitidos pela rede, podendo resultar em desequilíbrio e atrasos na chegada das informações ao destino. Com atrasos variáveis, pode haver, inclusive, a recepção de pacotes fora de ordem, o que, dependendo do *codec*, impacta significativamente o processamento da mídia. Trata-se de um problema particularmente crítico em comunicações de voz sobre IP e em *streaming* de vídeo, nas quais a qualidade da transmissão pode ser comprometida devido a essas variações.

Para uma transmissão eficaz de sinais digitalizados em redes com protocolo IP, é crucial adicionar suporte de protocolo que maneje questões como duplicação de *datagramas* e entrega desordenada. Isso é feito incluindo números sequenciais em cada transmissão. Além disso, para compensar o *jitter*, cada pacote deve conter um carimbo de data/hora indicando o momento exato de reprodução dos dados. Essa sincronização ajuda o receptor a restaurar corretamente o sinal, mesmo com a perda de pacotes ou interrupções na codificação (COMER, 2015).

Para gerenciar o *jitter*, o receptor pode implementar um *buffer* de reprodução, cuja função é acumular dados antes de iniciar a reprodução, garantindo uma reserva suficiente para lidar com variações no tempo de chegada dos pacotes. O ponto de reprodução é iniciado quando o *buffer* atinge um limite predefinido, permitindo uma experiência contínua de visualização ou audição, mesmo que alguns pacotes cheguem com atraso. Entretanto, essa abordagem não se aplica de forma ideal a todos os cenários: em transmissões em tempo real, o uso de *buffers* extensos pode aumentar a latência inicial e comprometer a interatividade, prejudicando a experiência do usuário.

2.3.2 Métricas de Qualidade de Experiência (QoE) para Transmissão de Vídeo

A QoE refere-se à percepção subjetiva do usuário sobre a qualidade geral de um serviço ou aplicação, especialmente em contextos multimídia, como *streaming* de vídeo e voz sobre IP. Diferente do QoS, que mede parâmetros técnicos e objetivos (como largura de banda, latência e taxa de perda de pacotes), o QoE avalia a experiência do usuário em relação à utilidade, eficiência e satisfação durante o uso do serviço (KANAI et al., 2019). A avaliação da QoE no contexto de protocolos de *streaming* pode ser realizada utilizando métricas objetivas e subjetivas. Entre as principais métricas, destacam-se:

Mean Opinion Score (MOS): é uma escala subjetiva que mede a percepção de qualidade do usuário, variando de 1 (péssimo) a 5 (excelente). Essa métrica pode ser obtida por meio de testes com usuários ou calculada por algoritmos que simulam o comportamento humano.

Peak Signal-to-Noise Ratio (PSNR): mede a qualidade da imagem comparando o vídeo original com o vídeo recebido. Valores mais altos indicam melhor qualidade visual.

Structural Similarity Index (SSIM): avalia a similaridade estrutural entre o vídeo transmitido e o recebido, considerando características como luminância, contraste e estrutura.

Rebuffering Ratio: mede o tempo total de pausas para carregamento em relação ao tempo total de reprodução do vídeo. Altos valores dessa métrica indicam experiências negativas para o usuário, pois refletem interrupções recorrentes ou prolongadas durante a exibição. O impacto do *rebuffering* na QoE tem sido amplamente estudado na litera-

tura. Segundo os experimentos conduzidos por Pessemier et al. (2015), tanto a frequência quanto a duração dos eventos de *rebuffering* influenciam negativamente a avaliação subjetiva da qualidade. Os resultados indicam que *rebufferings* mais longos ou mais frequentes degradam significativamente a experiência do usuário, especialmente quando ocorrem no meio ou no final do vídeo.

Initial Startup Delay: mede o tempo necessário para o início da reprodução do vídeo após o pedido de reprodução.

Video Quality Adaptation: avalia a frequência e a magnitude das mudanças na qualidade do vídeo devido a mecanismos de adaptação dinâmica Adaptive Bitrate Streaming (ABR).

2.3.3 Modelos de Avaliação de QoE

Existem diversos modelos desenvolvidos para avaliar o QoE de serviços de *streaming*, que combinam métricas objetivas e subjetivas. Alguns dos principais são:

ITU-T P.1203: Especificação padronizada pela União Internacional de Telecomunicações (ITU) para avaliar a qualidade de vídeo, áudio e interatividade em serviços de *streaming*.

Video Multi-Method Assessment Fusion (VMAF): modelo desenvolvido pela empresa *Netflix*, que combina diversas métricas, como PSNR, SSIM e outros parâmetros, para fornecer uma avaliação precisa da qualidade percebida pelo usuário.

Além das métricas mencionadas, realizou-se a análise de métricas objetivas de qualidade de vídeo, com o intuito de avaliar a degradação visual causada pela transmissão em diferentes condições de rede. Para isso, adotou-se o uso de bibliotecas do *FFmpeg*, uma vez que se trata de uma ferramenta consolidada, de código aberto e amplamente utilizada em pesquisas acadêmicas e aplicações práticas, oferecendo métodos padronizados para a análise comparativa entre o vídeo original e o vídeo recebido (GONÇALVES; CAVALCANTI; FIGUEIREDO, 2020).

2.3.4 PSNR

O PSNR é uma métrica objetiva utilizada para avaliar a qualidade de vídeos e imagens comprimidas em relação a uma versão original de vídeo. Ela é calculada através da razão entre o valor máximo possível de um sinal e o erro introduzido por técnicas de compressão ou transmissão. O cálculo é baseado na diferença média quadrática (MSE) entre os quadros da imagem original e do vídeo recebido. A fórmula é dada por:

$$\text{PSNR} = 10 \cdot \log_{10} \left(\frac{\text{MAX}^2}{\text{MSE}} \right) \tag{2.1}$$

onde *MAX* é o valor máximo possível do pixel (255 para vídeos em 8 bits), e o *MSE* é o erro quadrático médio entre os dois vídeos. Os valores mais altos de PSNR indicam uma menor distorção e, portanto, melhor qualidade de vídeo. No entanto, essa métrica não reflete necessariamente a percepção humana de qualidade de vídeo, sendo sensível apenas a erros de intensidade e não de estrutura visual (ITU-T, 2008).

Quadro 1 – Interpretação dos valores de PSNR

Faixa de PSNR (dB)	**Interpretação da Qualidade do Vídeo**
PSNR > 50	Qualidade excelente (praticamente sem distorção perceptível)
$40 <$ PSNR ≤ 50	Qualidade muito boa (distorções imperceptíveis ou mínimas)
$30 <$ PSNR ≤ 40	Qualidade boa (distorções perceptíveis, mas aceitáveis)
$20 <$ PSNR ≤ 30	Qualidade regular (distorções visíveis)
PSNR ≤ 20	Qualidade baixa (distorções severas perceptíveis)

Fonte: Adaptada de (HUYNH-THU; GHANBARI, 2008).

A interpretação dos valores de PSNR adotada neste trabalho está apresentada no Quadro 1, a qual relaciona faixas de valores em decibéis (dB) com níveis qualitativos de percepção visual. As classificações foram estabelecidas com base em convenções utilizadas nas referências, sendo adaptadas principalmente de Huynh-Thu e Ghanbari (HUYNH-THU; GHANBARI, 2008). Embora o *FFmpeg* não forneça diretamente interpretações subjetivas para os valores obtidos, os resultados gerados por meio do filtro `psnr` foram analisados à luz dessas faixas, permitindo estimar a qualidade objetiva dos vídeos recebidos em cada cenário.

2.3.5 SSIM

O SSIM foi proposto como uma alternativa ao PSNR, com o objetivo de se aproximar da percepção humana sobre a qualidade de imagens. Essa métrica considera as mudanças estruturais entre os quadros, analisando componentes de luminância, contraste e estrutura. O valor de SSIM varia entre 0 e 1, sendo que os valores próximos de 1 indicam alta similaridade entre o conteúdo original e o recebido. A fórmula do SSIM é definida como:

$$\text{SSIM}(x, y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)} \tag{2.2}$$

Onde:

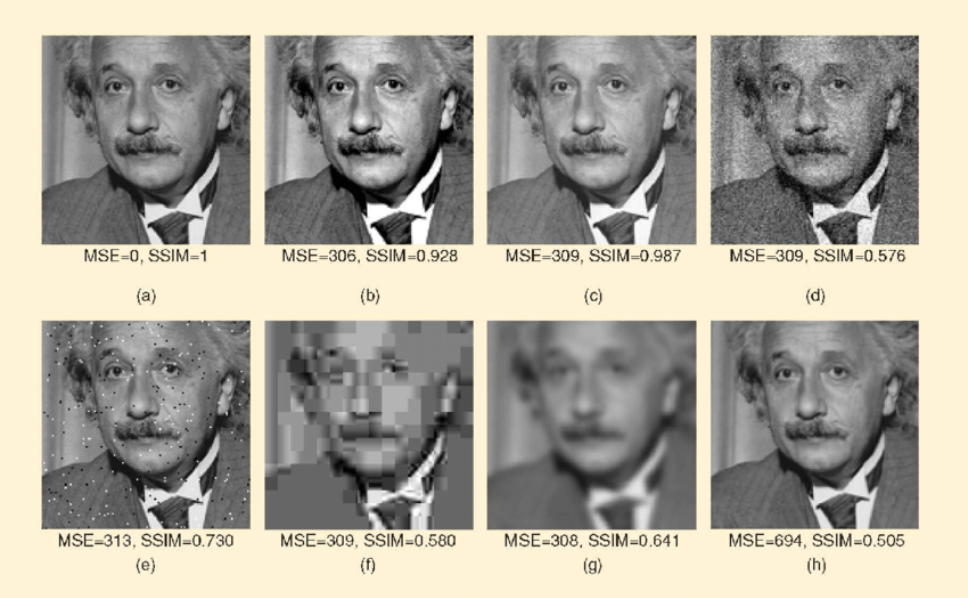
- μ_x, μ_y são as médias de luminância;
- σ_x^2, σ_y^2 são as variâncias;

- σ_{xy} é a covariância;
- C_1, C_2 são constantes para evitar divisão por zero.

O SSIM é utilizado na avaliação de artefatos de compressão ou perdas que preservam a estrutura da imagem, mas alteram sua intensidade. Ele foi proposto por Wang et al. (WANG et al., 2004) como um modelo percentual mais coerente com a visão humana.

A Figura 2 demonstra como o índice SSIM é utilizado na avaliação perceptual da qualidade de imagens. Visualmente, essa diferença é clara — a imagem (c) mantém quase toda a estrutura da original, enquanto (d) está fortemente degradada. O SSIM é capaz de capturar alterações estruturais que impactam a qualidade percebida, como perda de nitidez, ruído ou distorções locais, sendo mais confiável que métricas puramente baseadas em erro pixel a pixel.

Figura 2 – Comparação entre os desempenhos do SSIM



Fonte: (HORÉ; ZIOU, 2010).

A interpretação dos valores de SSIM está apresentada na Tabela 1. Assim como ocorre com o PSNR, foram definidas faixas de valores associadas a níveis qualitativos de percepção visual. Valores mais próximos de 1 indicam maior similaridade estrutural entre o vídeo original e o recebido, refletindo uma qualidade percebida mais alta. Já valores abaixo de 0,70 representam degradações severas, perceptíveis ao usuário final. Essas faixas foram adaptadas de Horé e Ziou (2010), que discutem a relação entre métricas objetivas e a percepção humana de qualidade de imagem.

Tabela 1 – Interpretação dos valores de SSIM

Faixa de SSIM	Interpretação da Qualidade do Vídeo
SSIM $\geq$ 0,95	Qualidade excelente (quase idêntico ao original)
0,90 $\leq$ SSIM $<$ 0,95	Qualidade muito boa (diferenças mínimas)
0,80 $\leq$ SSIM $<$ 0,90	Qualidade boa (diferenças visíveis, mas aceitáveis)
0,70 $\leq$ SSIM $<$ 0,80	Qualidade regular (diferenças perceptíveis)
SSIM $<$ 0,70	Qualidade baixa (distorções severas)

Fonte: Adaptado de (HORÉ; ZIOU, 2010).

2.3.6 VMAF

O VMAF é uma métrica desenvolvida pela *Netflix*[1] que combina múltiplos modelos de avaliação da qualidade de vídeo com base em aprendizado de máquina para refletir com mais precisão a percepção do usuário. Ao contrário de PSNR e SSIM, o VMAF integra múltiplas métricas, (como *Detail Loss Metric*, *Visual Information Fidelity* e outros atributos) em uma pontuação única entre 0 e 100, onde valores mais altos indicam melhor qualidade percebida (NETFLIX TECHNOLOGY BLOG, 2016).

Essa métrica foi criada com base em avaliações humanas e considerando fatores como nitidez, ruído, desfoque e outros artefatos comuns em compressão e transmissão (NETFLIX TECHNOLOGY BLOG, 2016). O VMAF é especialmente indicado em estudos de QoE, sendo cada vez mais adotado na indústria de *streaming*. No presente trabalho, o VMAF foi calculado utilizando ferramentas automatizadas após a recepção dos vídeos, permitindo comparações precisas entre os protocolos testados.

Quadro 2 – Classificação da Qualidade de Vídeo com base na pontuação VMAF

Pontuação VMAF	Nível de Qualidade Percebida
90 – 100	Excelente (sem perdas visíveis)
75 – 89	Boa (alta qualidade com poucas perdas perceptíveis)
50 – 74	Regular (artefatos visíveis, mas ainda assistível)
20 – 49	Baixa (qualidade comprometida, perdas perceptíveis)
0 – 19	Muito baixa (quase inassistível)

Fonte: Adaptado de Netflix Technology Blog (2016).

2.3.7 Cálculo das Métricas Avaliadas

A fim de garantir a fidelidade dos testes e capturar os impactos reais sobre a experiência do usuário, este trabalho adota uma abordagem de instrumentação no lado do cliente, com análise ativa automatizada, conforme classificado pelos autores Juluri, Tamarapalli e Medhi (2015). Esse tipo de metodologia permite avaliar a QoE em cenários realistas, utilizando aplicações reais (como o FFmpeg) integradas a um ambiente de rede

[1] Empresa norte-americana de streaming que fornece conteúdo multimídia sob demanda via internet.

simulado, com controle sobre variáveis como *jitter*, perda de pacotes e mobilidade. Assim, a estratégia adotada neste experimento está alinhada com as práticas recomendadas na literatura para estudos de desempenho de serviços multimídia.

Embora a QoE esteja centrada na percepção do usuário, ela é diretamente impactada por métricas objetivas de QoS, como latência, *jitter*, perda de pacotes e vazão. A avaliação da QoS permite compreender os fatores técnicos que afetam a entrega de vídeo e serve como base para interpretar os impactos observados na QoE. Este ponto é fundamental, pois a análise integrada entre QoS e QoE possibilita identificar não apenas o desempenho bruto da rede, mas também a forma como esse desempenho se reflete na experiência prática do usuário final, o que orienta a comparação entre os protocolos avaliados.

As principais métricas utilizadas neste trabalho, conforme apontado por Juluri, Tamarapalli e Medhi (2015), são: retransmissão, tempo total de sessão, *jitter* e *buffering*. Essas métricas foram extraídas a partir de arquivos de captura (.pcap) e logs gerados durante a execução dos testes com tráfego real entre os nós da simulação OMNeT++/INET, incluindo variações controladas, como perda, *delay* e mobilidade.

2.3.8 Retransmissão

A **taxa de retransmissão** é uma métrica para avaliação da QoS, pois indica a confiabilidade da transmissão. Pacotes retransmitidos indicam que houve perda ou erro durante o envio, exigindo que o transmissor reenvie os dados.

A fórmula para calcular a taxa de retransmissão é dada por:

$$\text{Taxa de Retransmissão (\%)} = \frac{N_{\text{retransmitidos}}}{N_{\text{total}}} \times 100 \tag{2.3}$$

onde:

- $N_{\text{retransmitidos}}$ é o número de pacotes reenviados (detectados por duplicação do número de sequência TCP);
- N_{total} é o número total de pacotes transmitidos.

Na análise de tráfego RTMP, as retransmissões no protocolo TCP são identificadas pela repetição dos mesmos números de sequência (*Seq*) para o mesmo par origem/destino. Na análise do SRT cada pacote de dados do tipo `DATA` inclui um `seqno` (número de sequência SRT) e um `msgno` (identificador da mensagem).

2.3.9 Tempo total de Sessão

Essa métrica representa a duração completa de uma transmissão, calculada através da diferença entre o *timestamp* do primeiro pacote e o *timestamp* do último pacote pertencente ao fluxo de vídeo.

$$\text{Tempo Total da Sessão} = T_{\text{último}} - T_{\text{primeiro}} \tag{2.4}$$

onde $T_{\text{último}}$ é o tempo de captura do último pacote da sessão, e T_{primeiro} é o do primeiro. Essa métrica não representa a *latência* entre origem e destino, mas essencial para normalizar métricas como *jitter*, taxa de retransmissão por segundo e vazão média.

2.3.10 Jitter

O *jitter*, definido como a variação no atraso entre pacotes sucessivos, foi calculado através da análise do intervalo de tempo entre pacotes consecutivos capturados pelo *Wireshark*. Para o protocolo SRT, que é baseado em UDP, o valor de *jitter* foi obtido utilizando a seguinte fórmula adaptada da RFC 3550 RTP:

$$J = J + \frac{|D(i-1,i)| - J}{16} \tag{2.5}$$

Onde J representa o *jitter* atual, e $D(i-1,i)$ é a variação no atraso entre dois pacotes consecutivos. Essa variação é calculada por:

$$D(i-1,i) = (R_i - S_i) - (R_{i-1} - S_{i-1}) \tag{2.6}$$

Nesta equação:

- R_i: tempo de chegada do pacote i
- S_i: *timestamp* do envio do pacote i
- R_{i-1}: tempo de chegada do pacote anterior
- S_{i-1}: *timestamp* do envio do pacote anterior

Já no caso do protocolo RTMP, baseado em TCP, os mecanismos de retransmissão e reordenação da pilha de transporte mascaram o *jitter* em nível de aplicação, entregando os dados de forma ordenada. No entanto, essa ocultação é obtida à custa de maior latência, pois a entrega ao reprodutor é atrasada até que os pacotes ausentes sejam recebidos. A métrica ainda pode ser inferida indiretamente pela variação no tempo de chegada dos dados ao reprodutor.

2.3.11 Buffering

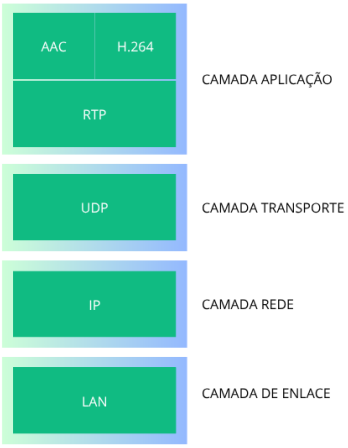
Podem ser identificadas de duas maneiras: (i) por meio de cronometragem manual baseada nas pausas perceptíveis na reprodução do vídeo; e (ii) por análise automática dos intervalos entre pacotes capturados, onde *gaps* superiores a 500 ms entre pacotes sucessivos foram considerados como indicativos de interrupções. Essa análise foi realizada com o auxílio do software *Wireshark*, a partir dos arquivos *.pcap* gerados nas simulações.

2.4 RTP

A RFC 3550 (SCHULZRINNE et al., 2003) especifica o RTP, um protocolo responsável por fornecer serviços de entrega em tempo real de ponta a ponta para dados multimídia, como áudio e vídeo interativos. Em termos funcionais, o RTP é descrito como um protocolo de transporte especializado, por oferecer mecanismos adicionais de suporte à transmissão contínua. Geralmente, é executado sobre o UDP, aproveitando serviços como multiplexação e soma de verificação.

No entanto, dentro da arquitetura do modelo TCP/IP, o RTP é classificado como um protocolo da camada de aplicação, já que opera sobre o UDP e é utilizado diretamente por aplicações multimídia. Essa distinção é relevante para compreender seu papel na pilha de protocolos e sua integração em sistemas como VoIP[2] , videoconferência e *streaming* ao vivo.

Figura 3 – Fluxo RTP



Fonte: Elaborada pelo autor.

Segundo Forouzan (2010), o RTP se situa entre o UDP e o programa aplicativo,

[2] VoIP (*Voice over Internet Protocol*) é uma tecnologia que permite a transmissão de voz em tempo real utilizando redes de pacotes baseadas em IP, em vez das redes telefônicas tradicionais (PSTN). Essa abordagem converte a voz analógica em pacotes digitais, que são transmitidos pela rede de dados, proporcionando redução de custos, integração com outros serviços multimídia e maior flexibilidade na comunicação.

conforme apresentado na Figura 3. O protocolo não garante a entrega de dados nem fornece outras garantias de qualidade de serviço, dependendo dos serviços de camadas inferiores para isso — como é o caso de utilizar o TCP na camada de transporte, o DiffServ[3] na camada de rede, e o Virtual Local Area Network (VLAN) tagging[4] na camada de enlace. O RTP é utilizado principalmente em aplicações como conferências multimídia, mas não se limita a elas, sendo também aplicável em simulação distribuída interativa.

O RTP inclui uma parte para transporte de dados (o próprio RTP) e uma para controle do transporte (Real-Time Control Protocol (RTCP)), que monitora a qualidade do serviço e transmite informações sobre os participantes durante uma sessão.

As funcionalidades do RTP são:

Numeração sequencial: os pacotes são numerados sequencialmente, facilitando a verificação de perdas e o sequenciamento correto dos pacotes;

Estampa de tempo (Time-stamp): responsável por manter a correta sincronização dos pacotes de áudio e vídeo;

Envio de pacotes sem retransmissão: importante para transmissões multimídia em que pequenas perdas são aceitáveis, sendo que a ausência de retransmissão contribui para reduzir a latência e aumentar a robustez do sistema. A depender do *codec* ou do envelopador (*container*) utilizado, essas perdas podem ter impacto mínimo — como no caso de *codecs* com compressão interquadro e mecanismos de tolerância a erros — ou comprometer significativamente a qualidade perceptual, especialmente quando afetam quadros-chave ou metadados essenciais para a decodificação.

Identificação de origem e de conteúdo: fundamental em cenários como conferências *multicast*, permitindo a identificação clara de quem enviou o pacote e o tipo de conteúdo;

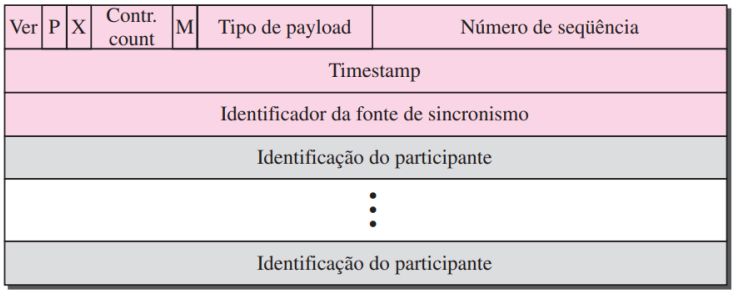
Sincronismo: gerenciamento de *jitter* e atrasos variáveis entre pacotes, usando *buffers* para ajustar diferenças de temporização;

A Figura 4 ilustra o formato do cabeçalho de um pacote RTP, sendo este simples e genérico para atender a uma ampla gama de aplicações de tempo real (FOROUZAN, 2010). Não há necessidade de aprofundamento nos itens que compõem o cabeçalho, pois não é o objetivo deste trabalho.

Para complementar o protocolo RTP, foi desenvolvido o RTCP, utilizado principalmente em aplicações de áudio e vídeo. Enquanto o RTP carrega os dados da mídia propriamente ditos, o RTCP é responsável por monitorar a qualidade da transmissão, fornecendo *feedback* sobre a entrega dos pacotes, o que inclui métricas como a perda de pacotes, *jitter* e o tempo de trânsito. Além disso, o RTCP é responsável pela sincronização

[3] Arquitetura de rede que classifica e gerencia o tráfego de rede (WIKIPEDIA CONTRIBUTORS, 2023)
[4] Método de identificação de pacotes que pertencem a diferentes LANs virtuais (NORDVPN, 2023).

Figura 4 – Formato do cabeçalho de pacotes RTP



Fonte: (FOROUZAN, 2010).

de áudio e vídeo durante a reprodução. Este protocolo opera periodicamente, enviando pacotes de controle para todos os participantes em uma sessão de comunicação, o que facilita a adaptação das taxas de transmissão.

2.5 PROTOCOLO RTMP

O RTMP é um protocolo proprietário inicialmente desenvolvido pela Macromedia e posteriormente adquirido pela Adobe, sendo amplamente utilizado para a transmissão de vídeo em tempo real. Ele opera sobre o TCP, o que garante uma entrega mais confiável dos pacotes de dados em comparação ao UDP, evitando perdas e desordenação. Essa maior confiabilidade pode contribuir para uma reprodução mais completa do conteúdo, porém à custa de maior latência, devido aos mecanismos de retransmissão de pacotes perdidos (KAUFFMANN, 2023).

Historicamente, sua principal aplicação foi no *Adobe Flash Player*, onde facilitava a transmissão de áudio, vídeo e outros dados entre servidores e clientes. Com a descontinuação do *Flash*, o RTMP passou a ser utilizado principalmente para o consumo de transmissões ao vivo em plataformas como *YouTube* e *Facebook Live*, onde os servidores convertem o fluxo RTMP para protocolos mais modernos como HTTP Live Streaming (HLS)[5] (ALOMAN et al., 2015).

O RTMP suporta a comunicação bidirecional, permitindo não apenas a transmissão de dados, mas também a comunicação de comandos entre o cliente e o servidor, o que é essencial para aplicações interativas como jogos online. Além disso, o protocolo permite ajustes dinâmicos de *bitrate*, adaptando a qualidade da transmissão às mudanças nas condições da rede para manter a melhor experiência visual possível (PARMAR; THORNBURGH, 2012).

[5] HLS (*HTTP Live Streaming*) é um protocolo de transmissão de vídeo desenvolvido pela Apple, baseado na divisão do fluxo contínuo em pequenos segmentos de mídia transportados sobre HTTP. Essa técnica permite adaptação dinâmica da taxa de bits (*adaptive bitrate streaming*), garantindo melhor experiência do usuário em redes com variação de largura de banda. O HLS é amplamente utilizado em plataformas de *streaming* devido à sua compatibilidade com navegadores e dispositivos móveis.

Uma das desvantagens do RTMP é a sua sensibilidade a redes congestionadas. Como o protocolo utiliza TCP, qualquer perda de pacotes aciona o mecanismo de retransmissão, o que pode causar atrasos e *buffering* perceptíveis para o usuário final. Esse fator torna o RTMP menos indicado para aplicações que exigem baixa latência, como transmissões esportivas e eventos ao vivo (NUÑEZ; TOASA, 2020).

2.5.1 Estabelecimento de Conexão no RTMP

O processo de estabelecimento de sessão do protocolo RTMP ocorre em 3 etapas (PARMAR; THORNBURGH, 2012). São elas:

2.5.2 Handshake

Esta é a etapa inicial no estabelecimento de conexão entre o cliente e o servidor.

Figura 5 – Handshake RTMP

Fonte: Elaborada pelo autor.

Fase 1: O cliente envia um pacote C0, que contém a versão do protocolo RTMP que ele suporta. Após o C0, ele envia um pacote C1, que inclui um carimbo de tempo e outros dados aleatórios.

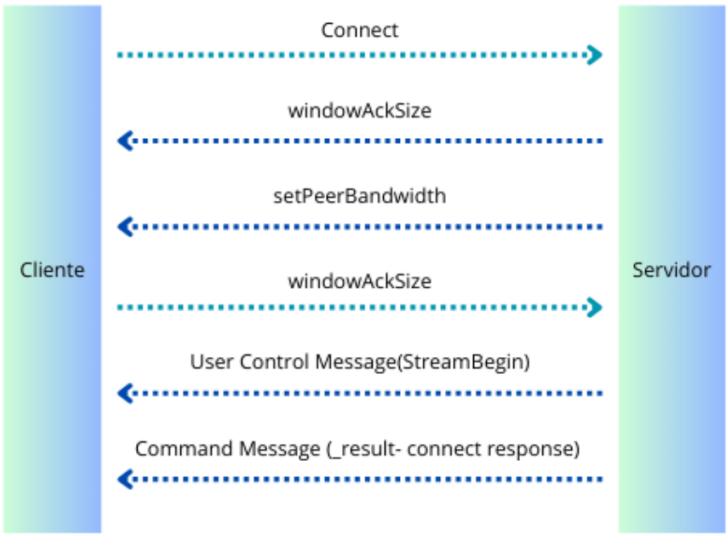
Fase 2: O servidor responde com um pacote S0, confirmando a versão do protocolo, seguido por um pacote S1, que ecoa o carimbo de tempo e os dados aleatórios. O servidor também envia um pacote S2, que é uma cópia do pacote C1 recebido do cliente.

Fase 3: O cliente completa o *handshake* enviando um pacote C2, que é uma cópia do pacote S1.

2.5.3 Conexão

Após o *handshake* bem-sucedido, o protocolo inicia o processo de conexão, onde o cliente e o servidor concordaram com as especificações da sessão de *streaming*. O cliente envia uma solicitação ao servidor que inclui detalhes como a URL da conexão e os tipos de *codecs* de áudio e vídeo que serão utilizados. O servidor analisa essas informações e, se tudo estiver em ordem, a conexão é estabelecida. Na Figura 6 é apresentado o fluxograma dessa comunicação.

Figura 6 – Conexão RTMP



Fonte: Elaborada pelo autor.

1. O cliente envia o comando *connect* ao servidor para solicitar a conexão com a instância do aplicativo do servidor;

2. Após receber o comando *connect*, o servidor envia a mensagem de protocolo *'windowAckSize'*, que define o tamanho da janela ao cliente. O servidor também se conecta ao aplicativo mencionado no comando *connect*;

3. O servidor envia a mensagem de protocolo *'setPeerBandWidth'*, que define a largura de banda ao cliente;

4. O cliente envia a mensagem de protocolo *'windowAckSize'* ao servidor após processar a mensagem de protocolo *'setPeerBandWidth'*;

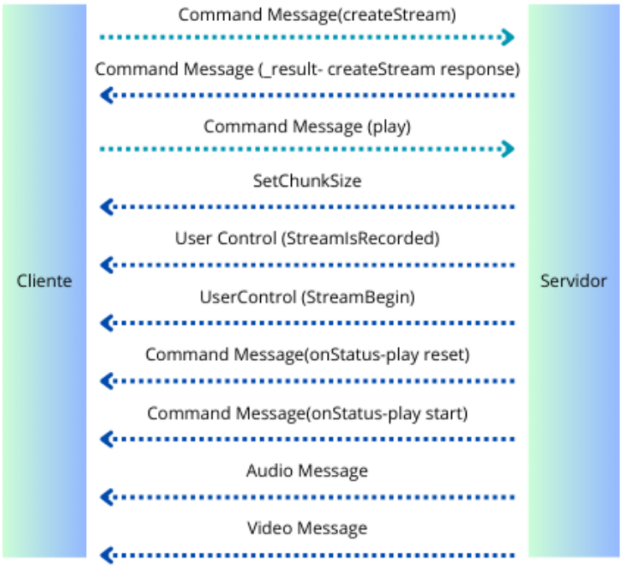
5. O servidor envia outra mensagem de protocolo do tipo *User Control Message(StreamBegin)* ao cliente;

6. O servidor envia a mensagem de comando *result* informando ao cliente o status da conexão (sucesso/falha).

2.5.4 Stream

Após as etapas anteriores terem sido finalizadas, o cliente terá acesso ao *streaming* de vídeo. O RTMP permite comandos como *createStream*, *play*, *seek* e pause. Esses comandos ajudam a gerenciar o fluxo dos dados de *streaming*. Eles dão ao transmissor controle sobre o *stream*, tornando possível iniciar ou parar o vídeo, retroceder ou pular para uma parte diferente do conteúdo.

Figura 7 – Stream RTMP



Fonte: Elaborada pelo autor.

O fluxo de mensagens durante a execução do comando é:

1. O cliente envia "*command message*"com a solicitação "*createStream*";
2. O servidor responde com "*command message*"contendo o resultado da solicitação do cliente;
3. O cliente envia o comando play após receber o resultado do comando *createStream* como sucesso do servidor;
4. Ao receber o comando *play*, o servidor envia uma mensagem *protocol* para definir o tamanho do bloco;
5. O servidor envia outra mensagem "*user control*"especificando o evento '*StreamIsRecorded*' e o ID do fluxo nessa mensagem;
6. O servidor envia outra mensagem "*user control*"especificando o evento '*StreamBegin*', para indicar o início do *streaming* para o cliente;

7. O servidor envia mensagens de comando *onStatus NetStream.Play.Start* e *NetStream.Play.Reset* se o comando *play* enviado pelo cliente for bem-sucedido. O *NetStream.Play.Reset* é enviado pelo servidor somente se o comando *play* enviado pelo cliente tiver definido o sinalizador *reset*. Se o fluxo a ser reproduzido não for encontrado, o servidor envia a mensagem *onStatus NetStream.Play.StreamNotFound*. Depois disso, o servidor envia dados de áudio e vídeo que o cliente reproduz.

Durante a transmissão, o servidor monitora a qualidade da conexão e pode ajustar dinamicamente a largura de banda e a qualidade do *stream*. Isso é realizado através do envio de mensagens *onMetaData* atualizadas com novos valores de taxa de bits e resolução. O cliente, por sua vez, pode enviar *feedback* ao servidor se enfrentar problemas como *bufferização* excessiva ou qualidade de vídeo inadequada, solicitando ajustes na transmissão. Se necessário, o servidor pode reduzir a resolução ou a taxa de bits do vídeo para se adaptar a uma redução na largura de banda disponível, garantindo que o *stream* continue fluindo sem interrupções significativas.

Em alguns casos, se a largura de banda se tornar suficientemente estável ou aumentar, o servidor pode aumentar a qualidade do *stream*, melhorando a experiência do usuário sem necessidade de reiniciar a sessão. Todo esse processo de ajuste e renegociação é mantido ao longo da sessão RTMP, com o servidor e o cliente em constante comunicação para garantir que os parâmetros de *streaming* sejam ideais para as condições atuais da rede. A sessão pode ser encerrada pelo cliente ou servidor enviando um comando *deleteStream* ou *close*, que encerra o *stream* e termina a comunicação (PARMAR; THORNBURGH, 2012).

2.6 PROTOCOLO SRT

O SRT é um protocolo de transmissão de vídeo desenvolvido pela Haivision, projetado para oferecer baixa latência, segurança e confiabilidade em redes não gerenciadas, como a internet pública. O SRT é uma solução de código aberto, disponível gratuitamente sob a licença Mozilla Public License (MPL 2.0)[6] , permitindo sua integração em diversas plataformas sem custos de licenciamento (HAIVISION, 2023).

Segundo Sharabayko e Kim (2020), o SRT é um protocolo de nível de usuário que retém a maioria dos conceitos básicos do protocolo UDP-based Data Transfer Protocol (UDT) e mecanismos, ao mesmo tempo que introduz vários refinamentos e melhorias, incluindo modificações de pacotes de controle, controle de fluxo aprimorado para mani-

[6] MPL 2.0 (*Mozilla Public License 2.0*) é uma licença de software livre e de código aberto desenvolvida pela Mozilla Foundation. Ela permite que o código licenciado seja usado, modificado e distribuído, inclusive em projetos proprietários, desde que os arquivos modificados permaneçam sob a mesma licença. Diferencia-se de licenças mais restritivas, como a GPL, por adotar um modelo de copyleft mais permissivo, favorecendo a integração entre software aberto e fechado.

pulação de *streaming* ao vivo, controle de congestionamento aprimorado e um mecanismo para criptografar pacotes.

Diferente do RTMP, que depende do TCP e tende a apresentar maior latência em redes congestionadas, o SRT utiliza o UDP como base, aplicando técnicas avançadas de correção de erros para mitigar perdas de pacotes e manter a estabilidade da transmissão. Durante a transmissão, o SRT monitora continuamente as condições da rede e adapta-se quase em tempo real, ajustando parâmetros como taxa de envio e mecanismos de recuperação para lidar com instabilidades e variações de largura de banda causadas pelo congestionamento. Esses mecanismos permitem reduzir significativamente a perda de pacotes, comum em conexões de Internet.

O SRT utiliza o Automatic Repeat reQuest (ARQ), um sistema de retransmissão de pacotes perdidos, garantindo qualidade superior mesmo em redes instáveis. Além disso, ele permite a criptografia dos dados, tornando a transmissão mais segura contra ataques e interceptações (ISAKA, 2020).

Devido à sua eficiência em redes com alta latência e perdas de pacotes, o SRT tem sido amplamente adotado para *streaming* profissional, especialmente em transmissões de eventos ao vivo e aplicações *broadcast* (HAIVISION, 2019). O protocolo monitora continuamente as condições da rede e adapta-se quase em tempo real, utilizando técnicas como ARQ e FEC, individualmente ou combinadas, para contornar perdas e manter baixa latência (SHARABAYKO, 2021)(HAIVISION SYSTEMS INC., 2024). Não há um limite fixo de tolerância a perdas — o desempenho depende da configuração: ARQ pode recuperar perdas moderadas com latência controlada; FEC é útil em cenários de alta RTT onde retransmissão seria onerosa; e a combinação de ARQ e FEC oferece o equilíbrio entre confiabilidade e eficiência (HAIVISION SYSTEMS INC., 2024).

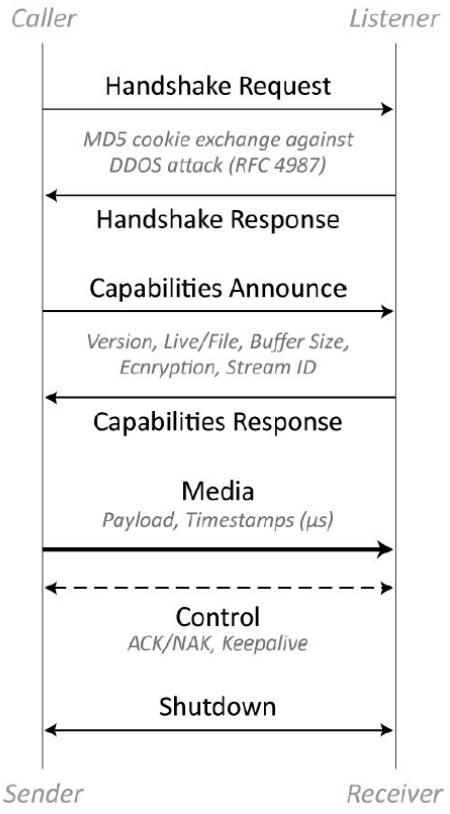
A Figura 8 ilustra uma visão de alto nível da troca de dados, incluindo dados de controle, entre dois pares em uma sessão SRT ponto a ponto. Durante a sessão, as funções dos pares podem mudar: inicialmente podem começar como Chamador e Ouvinte durante o processo de *handshake* e, subsequentemente, mudam para Remetente e Receptor durante a fase de transmissão de dados.

2.6.1 Estabelecimento de conexão no SRT

Uma conexão SRT é caracterizada pelo fato de que ela é (SHARABAYKO; KIM, 2020):

- iniciada por um processo de *handshake*;
- mantida enquanto quaisquer pacotes estiverem sendo trocados em tempo hábil;

Figura 8 – Visão geral do protocolo SRT



Fonte: (HAIVISION SYSTEMS INC., 2018).

- considerada fechada quando uma parte recebe o comando de fechamento apropriado de seu par (conexão fechada pelo *host* estrangeiro), ou quando não recebe nenhum pacote por algum tempo predefinido (conexão interrompida no tempo limite).

O SRT suporta duas configurações de conexão: *Caller-Listener* e *Rendezvous*.

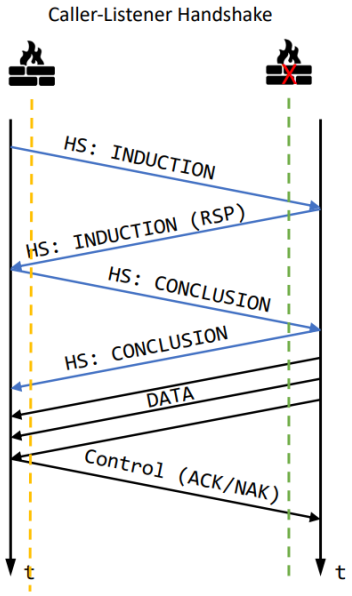
Caller-Listener

Neste processo, o cliente inicia com uma chamada de estabelecimento de conexão, conforme a Figura 9.

1. Cliente inicia o processo enviando um pacote de *handshake* (*caller to listener*) ao servidor. Esse pacote contém informações críticas como a versão do protocolo, configurações desejadas para a sessão (como latência máxima permitida), e se a criptografia está ativada. Ao receber o pacote de *handshake* do cliente, o servidor verifica se pode suportar as configurações solicitadas.

2. O servidor envolve a resposta ao pacote de *handshake* do cliente. Essa resposta pode confirmar a aceitação das configurações propostas pelo cliente ou sugerir ajustes com base nas capacidades do servidor ou nas condições da rede.

Figura 9 – Handshake Caller-Listener SRT



Fonte: (HAIVISION SYSTEMS INC., 2018).

Rendezvous

Neste processo de conexão, ambos os lados tentam iniciar uma conexão. A troca de *handshake Rendezvous* tem a seguinte ordem de Tipos de *Handshake*:

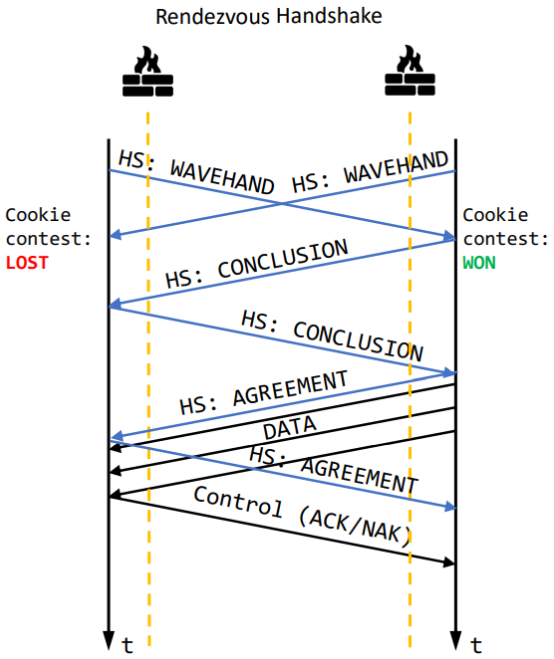
O processo *Rendezvous* usa uma máquina de estado. Ambas as partes começam com *WAVEAHAND*, onde ambas as partes criam um *cookie* para um processo chamado de "*Cookie contest*". Isso é necessário para a atribuição de funções de iniciador e atendedor. Cada parte gera um valor de *cookie* (um número de 32 bits) com base no *host*, porta e hora atual com precisão de 1 minuto. Este valor é embaralhado usando um cálculo de soma MD5 [7]. Os valores dos *cookies* são então comparados entre si. Quando o valor do *cookie* de uma das partes é maior que o do seu par, ela ganha o concurso de *cookies* e se torna o iniciador (a outra parte se torna o atendedor).

Neste ponto, existem dois possíveis "fluxos de *handshake*": serial e paralelo. No fluxo de *handshake* serial, uma parte é sempre a primeira e a outra segue. As chances de fluxo de *handshake* paralelo são muito baixas, mas ainda assim podem ocorrer se as mensagens de *handshake* com *WAVEAHAND* forem enviadas e recebidas por ambos os pares precisamente ao mesmo tempo.

O protocolo SRT trata a perda de pacotes, normalmente, por meio de um sistema de *feedback* ativo, no qual o receptor envia pacotes de Negative Acknowledgement (NAK) ao transmissor para notificar sobre dados não recebidos, permitindo a retransmissão sele-

[7] É uma função *hash* criptográfica desenvolvida por Ronald Rivest em 1991 como uma melhoria em relação às funções *hash* anteriores. Ele recebe uma entrada de qualquer comprimento e produz uma saída de comprimento fixo de 128 bits, normalmente representada como um número hexadecimal de 32 caracteres. (SPASOJEVIC, 2024)

Figura 10 – Handshake Rendezvous



Fonte: (HAIVISION SYSTEMS INC., 2018)

tiva quando o modo ARQ está habilitado. Embora eficaz na recuperação de perdas, esse mecanismo pode aumentar a latência em redes com alto RTT, razão pela qual, em alguns cenários, opta-se pelo uso de FEC ou pela desativação do ARQ para priorizar a entrega em tempo real.

2.6.2 *Buffer*

Em ambos os casos de uso, os usuários da comunicação possuem *buffers* definidos no código SRT, que não são expostos externamente. No remetente, a latência é o tempo que o SRT retém um pacote para aumentar a chance de uma entrega bem-sucedida, mantendo uma taxa de transmissão consistente até o destinatário. O impacto da latência é mínimo no remetente, onde é usada principalmente para decidir sobre o descarte de pacotes se um acknowledgemen (ACK) está ausente ou atrasado.

Os *buffers* de latência no remetente e no receptor armazenam temporariamente os pacotes, criando uma janela de tempo durante a qual ajustes na transmissão são feitos para maximizar a probabilidade de entrega bem-sucedida. Esta janela de latência funciona como uma deslizante que avança com o tempo, controlando a liberação de pacotes para o aplicativo no receptor. Se um pacote alcança o final dessa janela sem ser confirmado pelo receptor, ele é considerado perdido e pode ser retransmitido se ainda estiver dentro da janela.

A janela de latência também facilita a sincronização entre remetente e receptor, garantindo que ambos tenham uma visão consistente do estado da transmissão. Isso é

crucial, pois se os pacotes não chegarem no tempo esperado devido ao *jitter* ou outros atrasos, a qualidade da transmissão pode ser severamente afetada.

Além disso, a gestão dos *buffers* de latência pelo SRT permite lidar com altas taxas de *bitrate* e transmissões de alta definição. Em aplicações de *streaming* de vídeo ao vivo, esse recurso é permite assegurar qualidade e continuidade da visualização, ainda que se trate de mídia em tempo real. Nesses casos, o tamanho do *buffer* deve ser ajustado de acordo com o tipo de aplicação: maior para priorizar a estabilidade e a correção de perdas, e menor quando a interatividade e a baixa latência forem prioritárias.

2.7 ANÁLISE COMPARATIVA ENTRE SRT E RTMP

Nesta seção, será realizada uma análise comparativa entre os protocolos de *streaming* SRT e RTMP, com base em experimentos já realizados na literatura. A seguir, o Quadro 3 apresenta um resumo das principais pesquisas e testes comparativos realizados.

Os estudos analisados demonstram que o SRT oferece vantagens significativas sobre o RTMP, especialmente em redes instáveis e de alta latência. O uso do ARQ e criptografia fim a fim fazem do SRT uma solução mais moderna e robusta para transmissões ao vivo (HAIVISION, 2023).

Quadro 3 – Comparação entre SRT e RTMP.

Característica	**SRT**	**RTMP**
Latência	Baixa	Média a alta
Correção de erros	Sim (ARQ)	Sim (TCP)
Suporte à criptografia	Sim	Não nativo
Adoção moderna	Crescente	Em declínio
Plataformas compatíveis	CDN, Broadcast	Facebook, YouTube

Fonte: Elaborada pelo autor.

2.8 REVISÃO DE ALGUNS TRABALHOS SOBRE O RTMP E SRT COM FOCO EM AVALIAÇÃO DE DESEMPENHO

A transmissão de vídeo sobre redes de computadores é um tema amplamente estudado devido à crescente demanda por conteúdo de *streaming* em tempo real. Diversos estudos analisam os desafios relacionados à latência, confiabilidade e qualidade de transmissão dos principais protocolos utilizados para esse fim. Essa seção apresenta uma análise comentada das pesquisas mais relevantes sobre o tema, focando nos protocolos SRT e RTMP, destacando os pontos de vista convergentes e divergentes entre os autores.

Nos últimos anos, protocolos como RTMP e SRT têm sido avaliados em diversos contextos. Segundo (ISAKA, 2020), embora o RTMP ainda seja amplamente utilizado, ele enfrenta desafios de latência devido à sua dependência do TCP. Isaka destaca que o

SRT, baseado em UDP e com correção de erros dinâmica, surge como alternativa para melhorar a eficiência e confiabilidade da transmissão, especialmente em redes instáveis. Essa característica, aliada à capacidade de adaptação a redes variáveis, tem impulsionado a adoção do SRT em transmissões profissionais, conforme evidenciado por (HAIVISION, 2019).

Outro estudo relevante foi conduzido por (SONONO, 2019), que avaliou o desempenho dos protocolos Reliable Internet Stream Transport (RIST) e SRT em comparação a uma solução proprietária. Os resultados demonstraram que o SRT obteve desempenho superior na maioria dos cenários testados, especialmente em ambientes com taxas de perda de pacotes de até 2%. O SRT foi o único protocolo que, em algumas execuções, conseguiu evitar totalmente a perda de pacotes ao longo de uma hora de medições. O RIST apresentou boa estabilidade, mas seu desempenho foi inferior ao SRT em cenários de alta perda. O estudo não aborda o RTMP, focando exclusivamente na comparação entre RIST e SRT.

O estudo de (NURROHMAN; ABDUROHMAN, 2018) avaliou o desempenho do protocolo RTMP combinado com o codec H.264 em comparação ao sistema Real Time Streaming Protoco (RTSP)/H.263, destacando melhorias significativas em atraso (137,48ms a 146,02ms), *jitter* (22,917ms a 27,695ms) e perda de pacotes (0% a 0,4%), atendendo aos padrões do ITU-T para comunicação em tempo real. Embora o foco deste TCC seja a comparação entre os protocolos RTMP e SRT, vale mencionar que o estudo não aborda o protocolo SRT, limitando-se ao desempenho do RTMP/H.264 em dispositivos Android. Essa análise reforça a relevância de investigar protocolos de aplicação modernos, como SRT, para ambientes de streaming de vídeo com requisitos de baixa latência e em mobilidade.

Com base no estudo de (LAGHARI et al., 2023), que apresenta uma revisão abrangente sobre *streaming* de vídeo, reconhece-se que protocolos como o RTMP, embora ainda presentes em algumas aplicações, enfrentam limitações em cenários que exigem baixa latência. Em contrapartida, o SRT, baseado no UDP, é apontado como uma solução promissora, especialmente adequada para transmissões em redes instáveis, oferecendo mecanismos de correção de erros que contribuem para uma transmissão eficiente (LAGHARI et al., 2023). Além disso, o estudo aborda os desafios inerentes à transmissão de vídeos com resoluções 4K e 8K, como a crescente demanda por largura de banda e os custos associados, ressaltando a importância de soluções eficientes para garantir a QoS e a QoE.

Outro ponto relevante discutido é o processamento de vídeo em nuvem, que pode otimizar a transmissão e reduzir a latência, o que se alinha com as capacidades do SRT em ambientes de rede complexos (LAGHARI et al., 2023).

A metodologia empregada por Laghari et al. (2023), baseada em uma Revisão Sistemática da Literatura (SLR) que abrangeu diversas bases de dados, permitiu compilar

e organizar evidências para a análise comparativa entre diferentes protocolos de *streaming*, incluindo RTMP e SRT.

Assim, este capítulo fornece os fundamentos necessários para a compreensão dos experimentos realizados e das análises apresentadas nos capítulos subsequentes. Ao abordar conceitos de aplicações multimídia, técnicas de compressão, protocolos de transmissão e métricas de avaliação de qualidade, estabelece-se a base teórica que sustenta a comparação entre os protocolos RTMP e SRT em cenários de mobilidade e condições de rede adversas.

3 DESCRIÇÃO DE CENÁRIOS E ASPECTOS METODOLÓGICOS

Este capítulo descreve os cenários criados para as simulações, as ferramentas utilizadas e os métodos empregados para a análise dos resultados obtidos. Foram desenvolvidas duas infraestruturas simuladas: uma rede cabeada, com dispositivos estáticos, e uma rede híbrida, que combina elementos com fio e sem fio, incluindo mobilidade do receptor entre pontos de acesso. Sobre estas infraestruturas foram usadas aplicações reais para a avaliação de desempenho dos protocolos RTMP e SRT.

3.1 SIMULAÇÃO DE MOBILIDADE COM OMNET++

Este trabalho investiga os efeitos da mobilidade sobre os protocolos RTMP e SRT em transmissões de vídeo. Para tanto, combina-se o uso de aplicações reais de *streaming* com esses protocolos em um cenário de simulação controlado que incorpora mobilidade. O simulador adotado é o OMNeT++, e as aplicações de transmissão foram implementadas com o *FFmpeg*. A seguir apresenta-se uma visão geral do OMNeT++.

O OMNeT++ é um simulador baseado em eventos discretos usado na modelagem de redes de computadores e sistemas distribuídos. Nesses sistemas, os fenômenos de interesse mudam de estado apenas em instantes específicos: cada evento provoca alterações no estado do sistema e, entre eventos, o sistema permanece inalterado (ALMEIDA, s.d.). Como discutido por Mayer e Gamer (MAYER; GAMER, 2008), o tempo em simuladores de eventos discretos é representado como uma sequência ordenada de eventos, o que facilita a análise de redes distribuídas em condições reprodutíveis.

Integrar aplicações reais em um ambiente simulado oferece controle sobre topologia, parâmetros de enlace e padrões de tráfego, ao mesmo tempo em que reduz custos e complexidade frente à implantação de *testbeds*[1] físicos. Mayer e Gamer (MAYER; GAMER, 2008) destacam que essa abordagem permite avaliar o comportamento de aplicações em redes de grande escala, com diferentes topologias e cargas, de maneira econômica e flexível.

Para viabilizar essa integração, o OMNeT++ (via INET) disponibiliza um *modo de emulação*, no qual o simulador troca pacotes com aplicações reais por meio de interfaces TAP[2]. Nesse modo, o simulador precisa sincronizar sua linha do tempo interna com o

[1] Um *testbed* refere-se a uma plataforma experimental composta por hardware, software e configurações de rede, utilizada para testar, validar e avaliar o desempenho de protocolos, aplicações ou sistemas em condições controladas, aproximando-se de cenários reais.

[2] *Terminal Access Point* são dispositivos virtuais de rede em nível de enlace que permitem a injeção e a captura de pacotes Ethernet entre o sistema operacional hospedeiro e ambientes de simulação/emulação.

relógio do sistema (*wall-clock time*), o que impõe desafios de precisão temporal e pode introduzir perdas caso o processamento não acompanhe o ritmo do tráfego real.

Scussel, Moreira e Gaspary (2015) avaliaram o desempenho do modo de emulação da biblioteca INET e identificaram limitações, incluindo atrasos de até 22 ms para pacotes Internet Control Message Protocol (ICMP) simples. Os autores relatam que os eventos nesse modo não são processados imediatamente, mas sincronizados com o relógio real do sistema. Após modificações no código do agendador em tempo real — incluindo a ativação do modo `immediate` de captura de pacotes via `libpcap` — observou-se melhoria significativa da precisão temporal.

Apesar disso, Scussel et al. alertam que a captura imediata pode aumentar a perda de pacotes em cenários de tráfego intenso (SCUSSEL; MOREIRA; GASPARY, 2015). As melhorias propostas foram incorporadas à versão 3.0 da INET, tornando o modo de emulação mais robusto para aplicações sensíveis à latência, como transmissão de vídeo em tempo real.

Para instrumentação e análise, o *Wireshark* pode ser utilizado como analisador de protocolos para capturar e inspecionar pacotes quase em tempo real durante as execuções. A integração dos *showcases* do OMNeT++ com o *Wireshark* viabiliza análises mais realistas. Por exemplo, o *showcase Mobility Models* (INET FRAMEWORK, 2023b) permite simular o movimento de dispositivos a fim de examinar o impacto da mobilidade no desempenho de rede; o *showcase IEEE 802.11 Handover* (INET FRAMEWORK, 2023a) demonstra o processo de *handover* em redes WLAN, essencial para manter conectividade e qualidade de serviço em ambientes sem fio; e o *showcase Using Real Applications* (INET FRAMEWORK TEAM, 2024) integra aplicações reais de *streaming* de vídeo a uma rede simulada para avaliar o impacto de diferentes condições de rede (incluindo mobilidade) sobre métricas de desempenho. Esses *showcases* serviram de referência para a construção do cenário investigado neste trabalho.

Com base nessas evidências, este trabalho utiliza o modelo de emulação de rede do OMNeT++/INET para interligar aplicações reais (*FFmpeg*) ao cenário simulado por meio de interfaces TAP. O agendador `RealTimeScheduler` foi configurado para manter a sincronização com o tempo real e permitir a observação de métricas objetivas como atraso, perda, *jitter* e *rebuffering*. A ativação do modo de emulação no arquivo `omnetpp.ini` é feita pela diretiva:

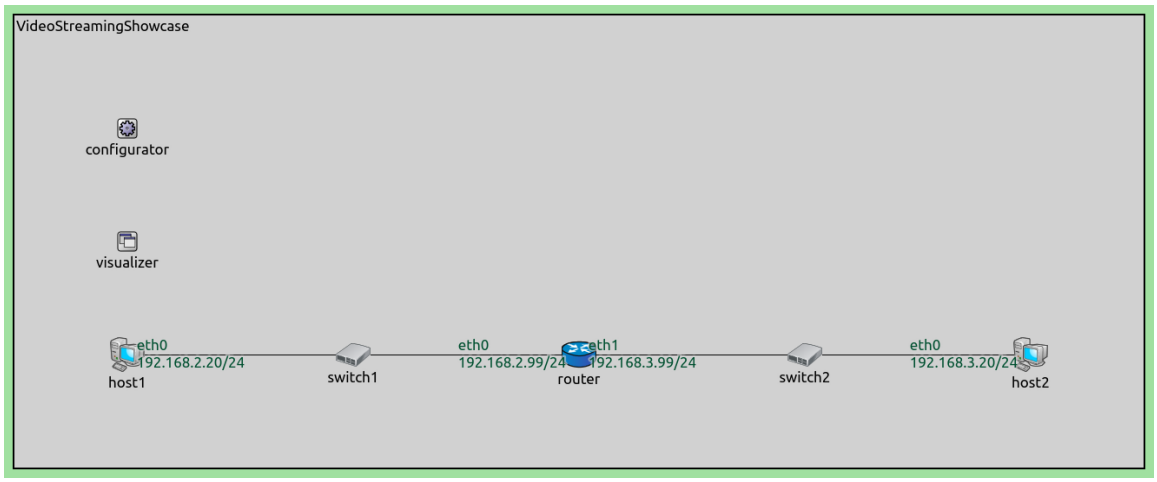
```
scheduler-class = "inet::RealTimeScheduler"
```

3.2 CENÁRIO DE REDE CABEADA (ESTÁTICO)

Neste cenário, foi construída uma infraestrutura com comunicação com fio e sem mobilidade. Todos os dispositivos permanecem estáticos, conectados por meio de enlaces

Ethernet com capacidade de 100 Mbps. O tráfego de vídeo e a captura de pacotes foram realizados por aplicações reais (como *FFmpeg* e *tcpdump*), utilizando interfaces TAP conectadas aos nós simulados. Na Figura 11 é apresentado este cenário, que é uma referência estável para comparação de desempenho.

Figura 11 – Cenário de rede cabeada com dispositivos estáticos



Fonte: Elaborada pelo autor.

No cenário com rede cabeada, foi utilizada uma topologia baseada no exemplo (INET FRAMEWORK TEAM, 2024), contendo dois nós (`host1` e `host2`) conectados por um roteador. A conexão entre a simulação e as aplicações é realizada por meio das interfaces TAP nomeadas `tapa` e `tapb`, respectivamente. A sincronização com o tempo real é garantida pelo agendador `RealTimeScheduler`. O trecho do arquivo `omnetpp.ini` abaixo apresenta a configuração utilizada para esse cenário:

Código 3.1 – Configuração das interfaces TAP

```
1 *.host1.eth[0].typename = "ExtUpperEthernetInterface"
2 *.host1.eth[0].device = "tapa"
3 *.host1.eth[0].copyConfiguration = "copyFromExt"
4
5 *.host2.eth[0].typename = "ExtUpperEthernetInterface"
6 *.host2.eth[0].device = "tapb"
7 *.host2.eth[0].copyConfiguration = "copyFromExt"
```

Além da definição das interfaces externas TAP, foi necessário configurar manualmente os endereços IP das interfaces do roteador por meio do módulo `IPv4NetworkConfigurator`, assegurando o roteamento correto entre os domínios de IP distintos. O trecho a seguir define os endereços IP estáticos das interfaces `eth0` e `eth1` do roteador.

Código 3.2 – Configuração das interfaces eth0 e eth1 do Router

```
1 *.configurator.config = xml("<config> \
```

```
2     <interface hosts='router' names='eth0' address='192.168.2.99' netmask
      ='255.255.255.0'/> \
3     <interface hosts='router' names='eth1' address='192.168.3.99' netmask
      ='255.255.255.0'/> \
4     </config>")
```

No exemplo original da topologia `VideoStreamingShowcase`, a comunicação era unidirecional: o `host1` envia pacotes ao `host2`, mas não havia retorno desses pacotes. Deve ser observado que qualquer fluxo gerado a partir do tapA para o tapB não seria injetado na rede emulada, pois seria uma entrega direta. O artifício proposto no exemplo, é a injeção do tráfego a partir do tapA (192.168.2.20) em direção ao endereço 192.168.2.99 (eth0) do roteador emulado, obrigando o tráfego a passar pela rede emulada. Uma regra NAT pré roteamento deste roteador, mapeia novos endereços fonte 192.168.3.99 (eth1) e destino do pacote (tapB - 192.168.3.20). Desta forma fica garantido o fluxo na direção tapA tapB passando pela rede emulada. A aplicação transmissora deve usar o endereço de destino 192.168.2.99 para atingir a aplicação receptora que espera em 192.168.3.20. Ver primeira regra apresentada na Tabela 2

Para protocolos como RTMP e SRT, que exigem negociação de conexão bidirecional, isso inviabilizava o funcionamento adequado. Por esse motivo, foi necessário adicionar regras de NAT na tabela de roteamento, de forma que os pacotes de resposta emitidos pelo `host2` pudessem ser roteados de volta ao `host1`. A configuração abaixo ilustra essa modificação:

Código 3.3 – Configuração da tabela de roteamento do router

```
1 *.router.ipv4.natTable.config = xml("<config> \
2     <entry type='prerouting' packetFilter='has(Ipv4Header) &amp;&amp; Ipv4Header.
      protocolId != 1 &amp;&amp; Ipv4Header.destAddress.str() == \"192.168.2.99\"'
      srcAddress='192.168.3.99' destAddress='192.168.3.20'/> \
3     <entry type='prerouting' packetFilter='has(Ipv4Header) &amp;&amp; Ipv4Header.
      protocolId != 1 &amp;&amp; Ipv4Header.destAddress.str() == \"192.168.3.99\"'
      srcAddress='192.168.2.99' destAddress='192.168.2.20'/> \
4     </config>")
```

A segunda regra de pré-roteamento acrescentada, transforma pacotes destinados ao IP 192,168.3.99 vindos do host2 em endereços fonte 192.168.2.99 (eth0 do roteador) e destino 192.168.20 (tapA). Desta forma, fica garantido fluxos em ambos os sentidos através da rede emulada. Deve ser observado que a aplicação no host2 deve direcionar seu tráfego para 192.168.3.99. Com essas regras, a simulação passou a suportar comunicação bidirecional completa entre os nós, viabilizando a execução de testes com protocolos de transporte que dependem de *handshake* e troca de mensagens de controle. Na tabela 2 são apresentadas essas regras de forma mais clara.

Tabela 2 – Regras de NAT configuradas para permitir a comunicação bidirecional entre transmissor (host1) e receptor (host2)

Recebido em (Iface/IP)	**Src (antes)**	**Dst (antes)**	**Src (depois)**	**Dst (depois)**
eth0 - 192.168.2.99	192.168.2.20	192.168.2.99	192.168.3.99	192.168.3.20
eth1 - 192.168.3.99	192.168.3.20	192.168.3.99	192.168.2.99	192.168.2.20

Fonte: Elaborada pelo autor.

As perdas de pacotes foram parametrizadas no nível de enlace, por meio do atributo `per` (*packet error rate*) dos canais Ethernet no INET. Esse parâmetro define a probabilidade de descarte de cada quadro transmitido no enlace, de forma independente. Nos experimentos, foram configurados valores de 0% (sem perda) e 1%, de modo a representar condições controladas de rede.

3.3 CENÁRIO DE MOBILIDADE

Este cenário representa uma rede heterogênea, na qual o receptor realiza o processo de *handover*[3] entre eles. Para a criação da comunicação *wireless*, foi necessária a utilização de um módulo `routerMobile` conectado em um *switch* e, posteriormente, ao `host2` via cabo. Essa adaptação foi necessária pois não foi identificado como utilizar as mesmas configurações nos *hosts*. Desta forma, a mobilidade foi implementada internamente na topologia da rede emulada.

O módulo `routerMobile` permite a configuração simultânea de duas interfaces de rede distintas: uma cabeada e uma sem fio (`wlan0`), com suporte ao protocolo IEEE 802.11, escaneamento ativo, associação e troca automática de ponto de acesso, fundamentais para que o móvel possa alternar sua conexão entre os diferentes APs durante a simulação.

O roteador fixo (nomeado `router`) utiliza as interfaces `eth0` e `eth1`, com os endereços `192.168.2.99` e `192.168.4.99`, respectivamente. O roteador móvel (`routerMobile`) recebe os endereços `192.168.4.98` na interface sem fio e `192.168.3.99` na interface cabeada, permitindo a comunicação entre a rede sem fio e o *host2*. À medida que o `routerMobile` se movimenta entre os pontos de acesso, sua conexão depende da associação Wi-Fi e também das rotas.

Foram configuradas regras de NAT em ambos os roteadores. No roteador fixo, o tráfego destinado ao IP `192.168.2.99` (interface cabeada do `router`) é redirecionado para o endereço `192.168.4.98` (interface Wi-Fi do `routerMobile`), permitindo que os pacotes originados no *host1* alcancem o nó móvel. O caminho inverso — da rede móvel em direção à rede onde se encontra o *host1* — é tratado por uma segunda regra, que substitui

[3] Refere-se à troca da conexão ativa de um dispositivo móvel entre dois pontos de acesso (APs), com o objetivo de manter a conectividade contínua durante o deslocamento.

o endereço de origem `192.168.2.99` por `192.168.2.20`, garantindo a transparência da comunicação bidirecional.

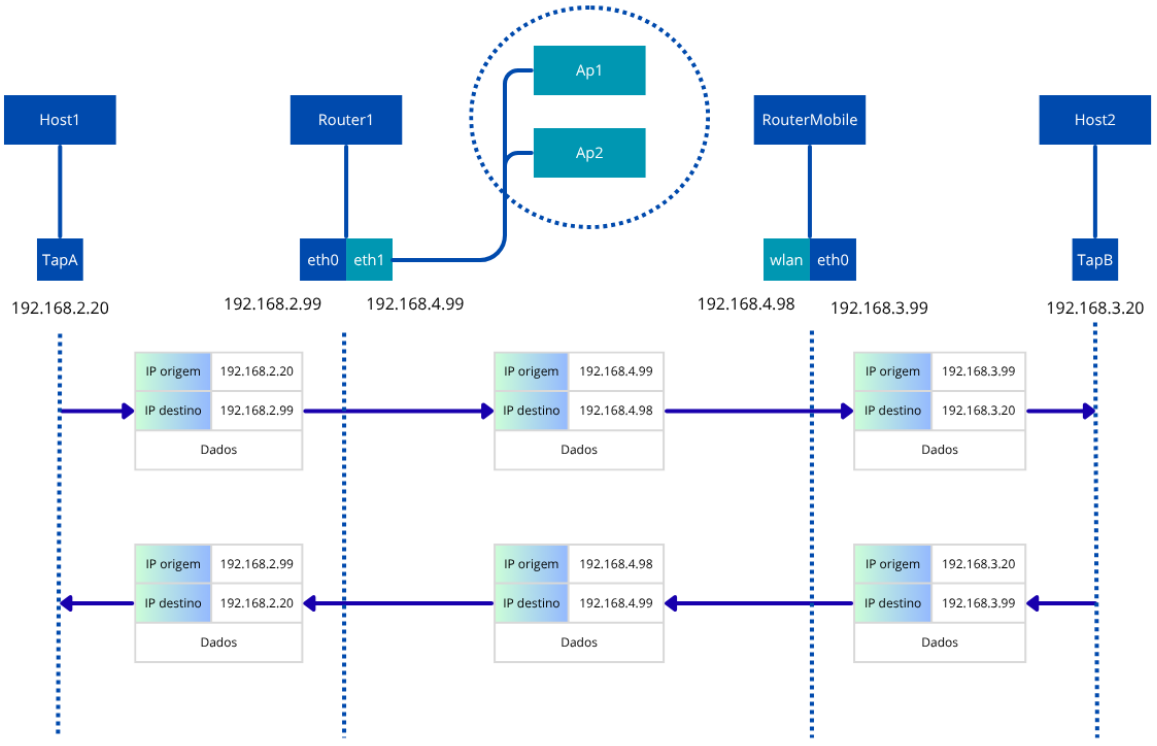
Já no `routerMobile`, foram inseridas regras para redirecionar o tráfego entre sua interface sem fio (`wlan0`) e a interface cabeada conectada ao `host2`, com endereços `192.168.4.98` e `192.168.3.99`, respectivamente. Isso garante que pacotes destinados ao receptor sejam entregues corretamente e que as respostas sejam encaminhadas de volta ao transmissor.

Tabela 3 – Regras de NAT configuradas para permitir a comunicação bidirecional entre transmissor (host1) e receptor (host2) no cenário com mobilidade. Endereços "antes" representam o cabeçalho IP ao chegar na interface indicada; "depois" representam o cabeçalho reescrito ao sair após a regra de NAT.

Recebido em (Iface/IP)	**Src (antes)**	**Dst (antes)**	**Src (depois)**	**Dst (depois)**
Router - eth0 (192.168.2.99) routerMobile - wlan0 (192.168.4.98)	192.168.2.20 192.168.4.99	192.168.2.99 192.168.4.98	192.168.4.99 192.168.3.99	192.168.4.98 192.168.3.20
routerMobile - eth0 (192.168.3.99) Router - eth1 (192.168.4.99)	192.168.3.20 192.168.4.98	192.168.3.99 192.168.4.99	192.168.4.98 192.168.2.99	192.168.4.99 192.168.2.20

Fonte: Elaborada pelo autor.

Figura 12 – Fluxo de pacotes e traduções de endereços (NAT) entre Host1 e Host2 atravessando Router1, infraestrutura Wi-Fi (Ap1/Ap2) e RouterMobile no cenário de mobilidade.



Fonte: Elaborada pelo autor.

As regras de roteamento foram implementadas de forma a garantir a comunicação entre os nós da rede, como resumido na Tabela 3 e na Figura 12. Cada entrada define o redirecionamento de tráfego a uma interface específica, substituindo endereços de origem ou destino conforme necessário para que os pacotes sejam entregues corretamente entre transmissor e receptor, mesmo em diferentes sub-redes.

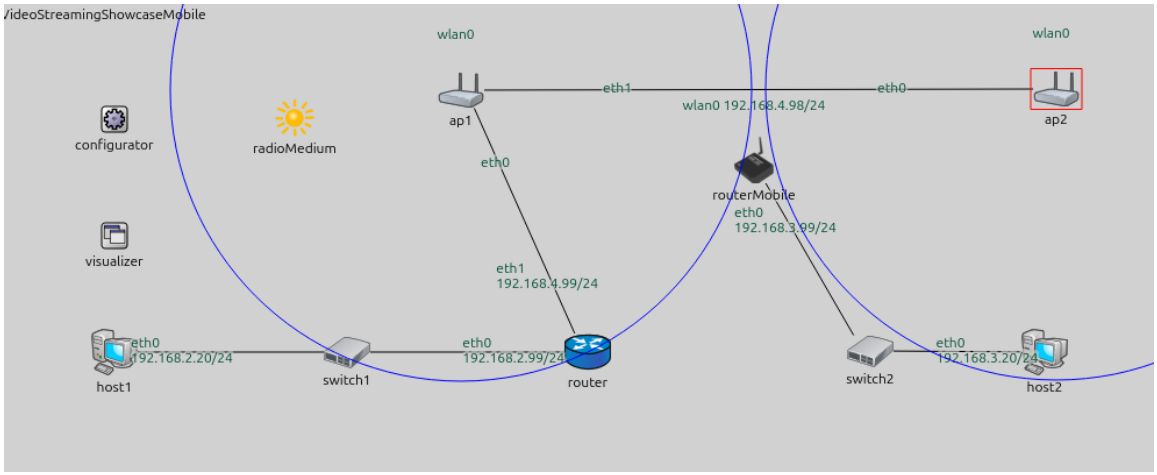
Código 3.4 – Configuração da tabela de roteamento do router e routerMobile

```
1 *.router.ipv4.natTable.config = xml("<config> \
2     <entry type='prerouting' packetFilter='has(Ipv4Header) &amp;&amp; Ipv4Header.
      protocolId != 1 &amp;&amp; Ipv4Header.destAddress.str() == \"192.168.2.99\"'
      srcAddress='192.168.4.99' destAddress='192.168.4.98'/> \
3     <entry type='prerouting' packetFilter='has(Ipv4Header) &amp;&amp; Ipv4Header.
      protocolId != 1 &amp;&amp; Ipv4Header.destAddress.str() == \"192.168.4.99\"'
      srcAddress='192.168.2.99' destAddress='192.168.2.20'/> \
4     </config>")
5
6 *.routerMobile.ipv4.natTable.config = xml("<config> \
7     <entry type='prerouting' packetFilter='has(Ipv4Header) &amp;&amp; Ipv4Header.
      protocolId != 1 &amp;&amp; Ipv4Header.destAddress.str() == \"192.168.4.98\"'
      srcAddress='192.168.3.99' destAddress='192.168.3.20'/> \
8     <entry type='prerouting' packetFilter='has(Ipv4Header) &amp;&amp; Ipv4Header.
      protocolId != 1 &amp;&amp; Ipv4Header.destAddress.str() == \"192.168.3.99\"'
      srcAddress='192.168.4.98' destAddress='192.168.4.99'/> \
9     </config>")
```

A Figura 13 ilustra como ficaram disponibilizados os membros na rede, com seus respectivos endereços.

O tipo de mobilidade utilizado foi o `CircleMobility`, onde este define um padrão de deslocamento circular, no qual o nó móvel percorre uma trajetória contínua ao redor de um ponto central previamente definido. O `routerMobile` realiza esse movimento com centro em (650 m, 50 m), raio de 85 metros e velocidade constante de 40 m/s. A escolha desse padrão de mobilidade permite simular, de forma controlada, a alternância de conexão entre os dois diferentes pontos de acesso, resultando em múltiplos eventos de *handover* durante a simulação. O movimento circular permite que o dispositivo se aproxime e se afaste dos APs de maneira periódica, promovendo variações na qualidade do sinal e possibilitando a análise do impacto da mobilidade na qualidade da transmissão de vídeo.

Figura 13 – Cenário com mobilidade e *handover* entre dois pontos de acesso



Fonte: Elaborada pelo autor.

A rede sem fio foi configurada com taxa de transmissão fixa de 54,Mbps, valor máximo suportado pelo padrão IEEE802.11g. Dois pontos de acesso foram definidos: `ap1` e `ap2`, cada um com identificadores SSID distintos e canais de operação diferentes (2 e 3, respectivamente). Embora esses canais apresentem sobreposição no espectro de 2,4,GHz em redes reais — o que poderia gerar interferência —, no ambiente de simulação foram considerados independentes, de modo a isolar os efeitos da mobilidade sem a influência de fatores externos. Ambos os APs utilizam um intervalo de *beacon*[4] de 100,ms e operam com endereços MAC únicos. O nó móvel (`routerMobile`) realiza varreduras sobre todos os canais disponíveis, utilizando tempos de escuta mínimos e máximos de 150,ms e 300,ms por canal, com atraso de sonda de 0,1,s. Os tempos máximos para autenticação e associação foram configurados em 5,s — valor superior ao observado em redes reais (tipicamente na ordem de centenas de milissegundos) —, mas adotado para garantir robustez no ambiente de simulação e evitar falhas decorrentes de tempos limite curtos durante o processo de *handover*[5].

A transição entre pontos de acesso não utilizou mecanismos de aceleração. O processo foi desencadeado quando o nó móvel entrou na área do sinal mais favorável de outro AP. A partir desse gatilho, o nó executou varredura ativa em todos os canais disponíveis (com *probeDelay* de 0,1 s e tempos de escuta por canal entre 150 ms e 300 ms), seguida das etapas de autenticação e associação. Considerando a configuração adotada

[4] Define o tempo entre transmissões periódicas de quadros de gerenciamento (*beacons*) enviados por um ponto de acesso (AP). Esses quadros anunciam a presença da rede e são essenciais para o processo de escaneamento e associação de dispositivos móveis. Um intervalo menor permite detecção mais rápida, mas pode aumentar o *overhead* de controle na rede.

[5] O padrão IEEE802.11g, utilizado nesta simulação, representa redes Wi-Fi legadas. Padrões mais recentes, como o Wi-Fi6/6E e o Wi-Fi7, oferecem melhorias significativas em taxa de transmissão (centenas de Mbps a múltiplos Gbps), eficiência espectral, latência e gerenciamento de múltiplos dispositivos. Tais avanços poderiam reduzir perdas e atrasos, alterando substancialmente os resultados aqui apresentados.

(dois APs nos canais 2 e 3), a interrupção observada na entrega de tráfego de aplicação situou-se tipicamente na ordem de 0,5–1,0 s, podendo aumentar em casos de contenção do meio ou retransmissões na camada de enlace. Esse intervalo explica picos de *rebuffering* e oscilações pontuais de qualidade (VMAF/SSIM/PSNR) próximos aos instantes de troca de BSS.

No nível de enlace, além da taxa fixa de 54 Mbps, a potência de transmissão foi ajustada para 2 mW[6], permitindo que variações no sinal fossem perceptíveis ao longo do deslocamento. Para compor um ambiente de propagação mais realista, foi adotado o modelo de perda de sinal `LogNormalShadowing`, que introduz pequenas flutuações aleatórias no nível de sinal, típicas de cenários urbanos e com obstáculos. A topografia foi modelada como plana (`FlatGround`), representando um terreno uniforme para o movimento do nó móvel. Não foram aplicados mecanismos de diferenciação de tráfego, como WMM (*Wireless Multimedia Extensions*), de modo que todos os fluxos compartilharam os mesmos recursos do enlace sem fio.

Na camada de rede, utilizou-se a pilha IPv4 padrão do INET, com roteamento estático e regras de NAT para permitir a comunicação entre os nós. As aplicações de vídeo (FFmpeg) operaram sobre as pilhas TCP e UDP do sistema operacional hospedeiro, conectadas à simulação via interfaces TAP.

Código 3.5 – Configuração do routerMobile e a mobilidade

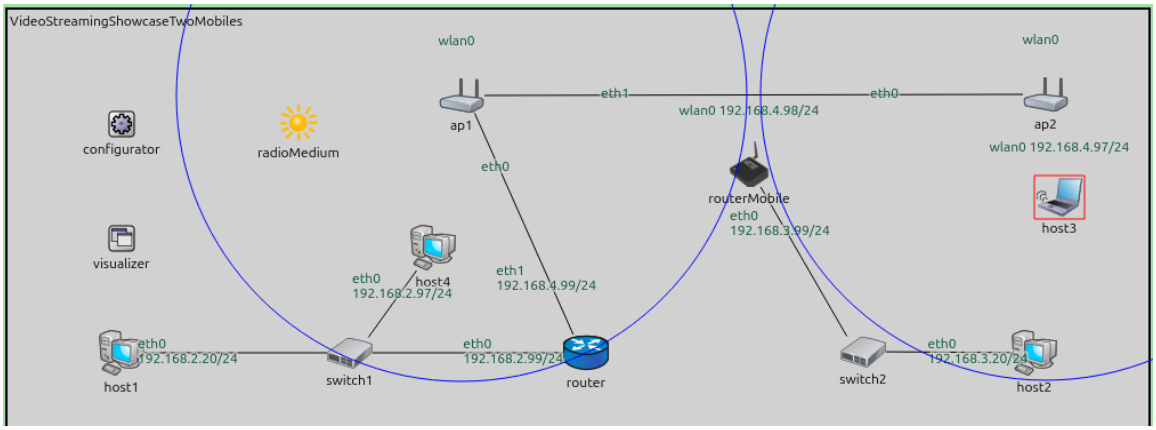
```
*.radioMedium.analogModel.ignorePartialInterference = true
**.mgmt.numChannels = 5

**.ap1.wlan[*].mgmt.ssid = "ap1"
**.ap2.wlan[*].mgmt.ssid = "ap2"
**.ap*.wlan[*].mgmt.beaconInterval = 100ms

**.ap1.wlan[*].address = "10:00:00:00:00:00"
**.ap2.wlan[*].address = "20:00:00:00:00:00"
**.ap1.wlan[*].radio.channelNumber = 2
**.ap2.wlan[*].radio.channelNumber = 3
**.routerMobile.wlan[*].radio.channelNumber = 0
**.wlan[*].mgmt.numAuthSteps = 4

**.radio.transmitter.power = 2mW

**.wlan[*].agent.activeScan = true
**.wlan[*].agent.channelsToScan = ""
**.wlan[*].agent.probeDelay = 0.1s
```

[6] A potência de 2 mW foi definida para limitar o alcance do sinal. Em redes Wi-Fi baseadas no padrão IEEE 802.11g, potências variam entre 15 mW e 100 mW. Em ambientes simulados, ela foi reduzida para intensificar os efeitos da atenuação e permitir uma análise mais evidente da mobilidade e da qualidade do sinal.

```
**.wlan[*].agent.minChannelTime = 0.15s
**.wlan[*].agent.maxChannelTime = 0.3s
**.wlan[*].agent.defaultSsid = ""
**.wlan[*].agent.authenticationTimeout = 5s
**.wlan[*].agent.associationTimeout = 5s

*.*.wlan*.opMode = "g(erp)"
*.*.wlan*.bitrate = 54Mbps

**.constraintAreaMinX = 0m
**.constraintAreaMaxX = 1000m
**.constraintAreaMinY = 0m
**.constraintAreaMaxY = 400m

*.routerMobile.mobility.typename = "CircleMobility"
*.routerMobile.mobility.cx = 650m
*.routerMobile.mobility.cy = 50m
*.routerMobile.mobility.r = 85m
*.routerMobile.mobility.speed = 40mps
*.routerMobile.mobility.startAngle = 90deg

*.physicalEnvironment.ground.typename = "FlatGround"
*.physicalEnvironment.ground.elevation = 0m
*.radioMedium.pathLoss.typename = "LogNormalShadowing"
```

O terceiro cenário, representado na Figura 14, introduz um fluxo de tráfego de fundo adicional ao ambiente com mobilidade. Nesse arranjo, dois novos nós são adicionados: o `host4`, responsável por gerar pacotes UDP periódicos, e o `host3`, que atua como receptor desse tráfego por meio de uma aplicação `UdpEchoApp`. O `host4` utiliza a aplicação `UdpBasicApp`, configurada para transmitir pacotes de 1000 bytes a cada 1s, durante toda a execução da simulação (250s). Essa comunicação paralela não interfere diretamente no fluxo principal de vídeo entre `host1` e `host2`, mas compartilha os recursos da rede sem fio, permitindo analisar o impacto do tráfego concorrente sobre métricas como *jitter*, vazão e *rebuffering*. A parametrização escolhida, equivalente a uma carga aproximada de 8 kbps, representa um tráfego leve e controlado, emulando aplicações de caráter periódico, como sensores IoT ou mensagens de controle, de modo a avaliar isoladamente o efeito mínimo da concorrência antes de considerar cenários mais intensos.

Figura 14 – Cenário com mobilidade e tráfego de fundo



Fonte: Elaborada pelo autor.

3.4 CONSIDERAÇÕES SOBRE A EXECUÇÃO DOS EXPERIMENTOS

Nesta seção são descritos os procedimentos de execução dos experimentos realizados, os cenários de simulação utilizados, os parâmetros configurados e as ferramentas empregadas para coleta e análise dos dados. Os testes foram realizados com variação controlada de fatores que impactam diretamente a qualidade da transmissão, incluindo perdas, atrasos, taxas de compressão.

3.4.1 Fatores e Métricas Consideradas

Os experimentos foram organizados de forma a isolar o impacto de diferentes fatores sobre a qualidade do vídeo transmitido, utilizando múltiplas combinações de protocolos, taxas de bits e cenários de rede. A Tabela 4 resume os principais fatores considerados.

Tabela 4 – Fatores experimentais e seus níveis de variação

Fator	**Níveis de variação**
Protocolo	SRT, RTP, RTMP
Codec de vídeo	H.264, H.265
Taxa de bits (*bitrate*)	VBR, 2 Mbps, 4 Mbps
Perda de pacotes	0%, 1%
Delay de rede	0 ms, 50 ms
Mobilidade	Sem mobilidade, Um Móvel
Tráfego de fundo	Ausente, Presente (alta intensidade)

Fonte: Elaborada pelo autor.

Embora o foco deste trabalho seja a comparação entre RTMP e SRT, optou-se também por incluir o RTP nos experimentos. O protocolo RTP foi utilizado como linha de base para representar o transporte de vídeo sobre UDP sem mecanismos adicionais de confiabilidade ou adaptação. Dessa forma, sua inclusão permitiu avaliar de forma mais

clara as vantagens e limitações do SRT em relação a um fluxo UDP simples, evidenciando o impacto dos mecanismos de correção de erros, retransmissão seletiva e controle de congestionamento implementados no SRT. É importante destacar que o protocolo RTMP não possui suporte nativo ao codec H.265/HEVC, sendo compatível apenas com fluxos H.264/AAC encapsulados em FLV. Dessa forma, para viabilizar os testes com H.265, adotou-se o protocolo RTP como alternativa. A inclusão do RTP, portanto, não teve como objetivo compará-lo diretamente ao RTMP ou ao SRT em termos de funcionalidades, mas sim permitir a avaliação do impacto do codec H.265 sobre as métricas de qualidade e desempenho. Como consequência, o RTP passou a representar uma linha de base para fluxos em UDP, servindo também como referência para destacar os mecanismos adicionais de confiabilidade e correção de erros presentes no SRT.

As Tabelas 5, 6 e 7 detalham os conjuntos de experimentos realizados neste trabalho, organizados em três cenários distintos. O Cenário 1 corresponde à rede cabeada, sem mobilidade e sem tráfego de fundo, incluindo testes adicionais com atraso de 50 ms e perda de 1% para avaliar o impacto de condições adversas de rede. O Cenário 2 representa a rede com mobilidade, mas sem tráfego de fundo, possibilitando observar o efeito isolado da movimentação do nó móvel sobre a qualidade do vídeo. Por fim, o Cenário 3 introduz tráfego de fundo em um ambiente com mobilidade, simulando situações de maior concorrência por recursos da rede. Em todos os cenários, foram avaliadas múltiplas combinações de protocolos (SRT, RTP e RTMP), codecs de vídeo (H.264 e H.265) e taxas de bits (VBR, 2Mbps e 4 Mbps), de forma a permitir uma análise comparativa do desempenho.

Tabela 5 – Cenário 1 – Rede cabeada (sem mobilidade e sem tráfego de fundo)

Protocolo	**Codec**	**Bitrate**	**Delay**	**Perda**	**Mobilidade**	**Tráfego**
SRT	H.264	2 Mbps	0 ms	Não	Não	Não
RTP	H.264	2 Mbps	0 ms	Não	Não	Não
RTMP	H.264	2 Mbps	0 ms	Não	Não	Não
SRT	H.264	4 Mbps	0 ms	Não	Não	Não
RTP	H.264	4 Mbps	0 ms	Não	Não	Não
RTMP	H.264	4 Mbps	0 ms	Não	Não	Não
SRT	H.264	4 Mbps	50 ms	1%	Não	Não
RTP	H.264	4 Mbps	50 ms	1%	Não	Não
RTMP	H.264	4 Mbps	50 ms	1%	Não	Não
SRT	H.265	2 Mbps	0 ms	Não	Não	Não
RTP	H.265	2 Mbps	0 ms	Não	Não	Não
SRT	H.265	4 Mbps	0 ms	Não	Não	Não
RTP	H.265	4 Mbps	0 ms	Não	Não	Não

Fonte: Elaborada pelo autor.

Tabela 6 – Cenário 2 – Rede com mobilidade (sem tráfego de fundo)

Protocolo	**Codec**	**Bitrate**	**Delay**	**Perda**	**Mobilidade**	**Tráfego**
SRT	H.264	VBR	0 ms	Não	Sim	Não
RTP	H.264	VBR	0 ms	Não	Sim	Não
RTMP	H.264	VBR	0 ms	Não	Sim	Não
SRT	H.264	2 Mbps	0 ms	Não	Sim	Não
RTP	H.264	2 Mbps	0 ms	Não	Sim	Não
RTMP	H.264	2 Mbps	0 ms	Não	Sim	Não
SRT	H.264	4 Mbps	0 ms	Não	Sim	Não
RTP	H.264	4 Mbps	0 ms	Não	Sim	Não
RTMP	H.264	4 Mbps	0 ms	Não	Sim	Não
SRT	H.265	2 Mbps	0 ms	Não	Sim	Não
RTP	H.265	2 Mbps	0 ms	Não	Sim	Não
SRT	H.265	4 Mbps	0 ms	Não	Sim	Não
RTP	H.265	4 Mbps	0 ms	Não	Sim	Não

Fonte: Elaborada pelo autor.

Tabela 7 – Cenário 3 – Rede com mobilidade e tráfego de fundo

Protocolo	**Codec**	**Bitrate**	**Delay**	**Perda**	**Mobilidade**	**Tráfego**
SRT	H.264	VBR	0 ms	Não	Sim	Sim
RTP	H.264	VBR	0 ms	Não	Sim	Sim
RTMP	H.264	VBR	0 ms	Não	Sim	Sim
SRT	H.264	2 Mbps	0 ms	Não	Sim	Sim
RTP	H.264	2 Mbps	0 ms	Não	Sim	Sim
RTMP	H.264	2 Mbps	0 ms	Não	Sim	Sim
SRT	H.264	4 Mbps	0 ms	Não	Sim	Sim
RTP	H.264	4 Mbps	0 ms	Não	Sim	Sim
RTMP	H.264	4 Mbps	0 ms	Não	Sim	Sim
SRT	H.265	2 Mbps	0 ms	Não	Sim	Sim
RTP	H.265	2 Mbps	0 ms	Não	Sim	Sim
SRT	H.265	4 Mbps	0 ms	Não	Sim	Sim
RTP	H.265	4 Mbps	0 ms	Não	Sim	Sim

Fonte: Elaborada pelo autor.

Dois vídeos de referência foram utilizados como entrada nas transmissões, variando apenas quanto ao *codec* de vídeo empregado. O primeiro vídeo utiliza o *codec* H.264/AVC, enquanto o segundo utiliza o H.265/HEVC. Trata-se do mesmo arquivo de vídeo previamente gravado, codificado em diferentes padrões de compressão, preservando as demais características técnicas. Essa escolha permitiu garantir a repetibilidade dos experimentos, assegurando que todos os protocolos fossem avaliados sob as mesmas condições. O conteúdo do vídeo é dinâmico, contendo movimentação e variação de cena, de modo a explorar os mecanismos de compressão de ambos os *codecs* e evidenciar os efeitos da rede sobre a qualidade de experiência percebida. As principais características técnicas dos arquivos estão apresentadas na Tabela 8. Ambos possuem resolução 640×360 e duração aproximada de 126,5 segundos.

A resolução de 640×360 pixels foi adotada com o objetivo de equilibrar a qualidade visual do conteúdo de referência e a viabilidade computacional dos experimentos. Esse formato apresenta menor carga de processamento para codificação e decodificação, reduzindo a possibilidade de que gargalos de hardware interfiram nas métricas avaliadas.

Além disso, a escolha facilita o controle sobre variáveis como *bitrate* e complexidade de compressão, permitindo isolar com maior precisão o impacto dos protocolos e das condições de rede (atraso, perdas e *jitter*) nos resultados. Em cenários reais de *streaming* sob redes instáveis ou com largura de banda limitada, resoluções reduzidas como 640×360 são frequentemente utilizadas para preservar a continuidade da reprodução. Outro fator relevante é a compatibilidade dessa resolução com os codecs H.264 e H.265, assegurando que as métricas objetivas de qualidade — como VMAF, PSNR e SSIM — possam ser calculadas com precisão, mantendo ao mesmo tempo tamanhos de arquivo adequados para armazenamento e análise repetida.

Nos experimentos, a taxa de quadros foi mantida fixa em 29,97 fps, conforme os vídeos de referência. Já a taxa de bits foi configurada em três modos distintos — VBR, 2 Mbps e 4 Mbps — de forma a comparar cenários com compressão adaptativa e taxas fixas. Enquanto o modo fixo trouxe maior previsibilidade de vazão, o VBR buscou eficiência de compressão ao custo de oscilações de qualidade e maior risco de *rebuffering*.

Tabela 8 – Comparação entre os arquivos H.264 e H.265 (mesma resolução)

Parâmetro	**v2.mp4 (H.264)**	**v2-265.mp4 (H.265)**
Codec de vídeo	H.264 (Main)	H.265 / HEVC (Main)
Resolução	640×360	640×360
Duração	126,5 s	126,5 s
Frames	3792	3789
Taxa de quadros	29,97 fps	29,97 fps
Codec de áudio	AAC LC	AAC LC
Taxa de amostragem	44,1 kHz	44,1 kHz
Canais de áudio	Estéreo (2)	Estéreo (2)
Bitrate do áudio	128 kbps	127 kbps
Tamanho do arquivo	6,1 MB	9,4 MB

Fonte: Elaborada pelo autor.

É importante destacar que, nos experimentos realizados, não houve mecanismos de renegociação de mídia durante a transmissão. Todos os fluxos foram iniciados com parâmetros fixos de codec, bitrate e resolução, sem ajustes dinâmicos em tempo de execução. Da mesma forma, não ocorreu troca de mídia ou alteração de codecs durante os testes. Este comportamento reflete o uso típico de transmissões em tempo real em sistemas de CFTV e *streaming* direto, mas difere de abordagens adaptativas (como DASH, HLS ou WebRTC), que podem renegociar parâmetros dinamicamente em resposta às condições de rede.

As métricas empregadas para avaliação foram agrupadas em dois conjuntos: (i) métricas de qualidade de vídeo — PSNR, SSIM e VMAF — e (ii) métricas de desempenho de rede — vazão, perda, *jitter* e *rebuffering*. A coleta foi realizada de forma automatizada

por meio de scripts em Python e ferramentas como *FFmpeg*, *tcpdump*, *tshark* e analisadores de vídeo.

Optou-se pelas métricas PSNR, SSIM e VMAF por sua complementaridade: enquanto PSNR e SSIM fornecem indicadores objetivos baseados em propriedades matemáticas e estruturais do sinal, o VMAF incorpora aspectos perceptuais através de modelos de aprendizado de máquina, oferecendo uma estimativa mais próxima da QoE percebida pelo usuário (JULURI; TAMARAPALLI; MEDHI, 2015; NETFLIX TECHNOLOGY BLOG, 2016).

Nos experimentos, o protocolo SRT foi utilizado no *live mode*[7] , configuração padrão do FFmpeg para transmissões em tempo real. Nesse modo, o buffer de recepção é reduzido ao mínimo necessário, priorizando baixa latência em detrimento da entrega absolutamente confiável de todos os pacotes. Essa escolha é coerente com o objetivo do trabalho, que busca avaliar os impactos da mobilidade, do jitter e das perdas sobre a qualidade de experiência em transmissões de vídeo em tempo real.

3.4.2 Execução

A execução dos experimentos de mobilidade segue dois passos: (i) inicialização do cenário de rede no OMNeT++ e (ii) transmissão do vídeo de referência via *FFmpeg* usando um dos protocolos avaliados (SRT, RTP ou RTMP). Durante cada execução são coletadas capturas de tráfego (`tcpdump`), logs de transmissão e recepção, e métricas objetivas de qualidade de vídeo (PSNR e SSIM) calculadas a partir do arquivo recebido..

O cenário de rede é carregado no OMNeT++ com a configuração `General-01`, que corresponde ao caso “sem perdas de pacotes e sem delay”. O comando a seguir deve ser executado no diretório do projeto (ou via IDE configurada) após a compilação do *showcase*:

Código 3.6 – Execução do cenário OMNeT++

```
1 # Cenário baseline: sem injeção de perda ou atraso adicional
2 ./VideoStreamingShowcase -u Cmdenv -f omnetpp.ini -c General-01
```

O uso de `-u Cmdenv` força execução em modo não-gráfico (terminal).

Para cada experimento, um *script* de automação aciona o *FFmpeg* tanto no transmissor quanto no receptor, configurando codificação (H.264 ou H.265, e variantes de taxa alvo) e protocolo. A escolha do protocolo é feita pelo primeiro argumento do *script*:

[7] O protocolo SRT (*Secure Reliable Transport*) pode operar em dois modos principais: (i) *live mode*, no qual o buffer de recepção é mantido no mínimo necessário para reduzir a latência, aceitando pequenas perdas; e (ii) *file mode*, que prioriza a entrega completa e ordenada dos pacotes, utilizando buffers maiores ao custo de maior atraso. Neste trabalho, foi utilizado o *live mode*, configuração padrão do FFmpeg, por ser mais adequado a cenários de transmissão em tempo real.

Código 3.7 – Chamada do script de transmissão/recepção

```
# Vídeo com compressão H.264; escolha do protocolo no argumento final
# Opções de script: ffmpeg_264.sh, ffmpeg_264_2m.sh, ffmpeg_264_4m.sh,
#                   ffmpeg_265.sh, ffmpeg_265_2m.sh, ffmpeg_265_4m.sh
# Uso:
#   ./ffmpeg_264.sh <srt|rtp|rtmp>

./ffmpeg_264.sh srt
```

As variantes `_2m`, `_4m` etc. possuem os mesmos códigos, alterando somente o tipo do codec e a taxa de transferência para os cenários. Ou seja, o arquivo intitulado "ffmpeg_264_.sh"foi configurado com o codec H.264 e bitrate variável. Já o arquivo "ffmpeg_264_2m.sh"foi configurado com o codec H.264 e bitrate variável em torno de 2Mbps. O Código 3.8 apresenta o fluxo lógico do *script* . A versão completa, com todos os comandos e verificações, encontra-se no Apêndice 6.6.

Cada execução gera um diretório exclusivo nomeado com o nome do protocolo utilizado no teste, seguido do *timestamp*. Os caminhos de todos os arquivos produzidos são preparados em seguida: captura de pacotes em PCAP[8] através do *tcpdump*, versão CSV da captura, arquivo do fluxo recebido e arquivos de estatísticas de PSNR/SSIM. Essa organização evita sobrescrever dados entre rodadas e simplifica a análise posterior.

O *script* seleciona o fluxo de execução com base na variável `PROTO` e, para cada protocolo, lança em segundo plano um transmissor e um receptor `ffmpeg` (com `-re` para enviar em tempo real), registrando seus PIDs e gravando logs separados de transmissão (ffmpeg_tx_.log) e recepção (ffmpeg_rx_.log).

As URLs utilizadas nos *scripts* de transmissão variam conforme o protocolo adotado. No protocolo SRT, a URL adota a sintaxe *srt://<ip>:<porta>?mode=...*, onde se define o modo de operação (*listener* para o transmissor e *caller* para o receptor) e o parâmetro pkt_size, que ajusta o tamanho dos pacotes transportados via UDP.

Já no protocolo RTP, a URL segue o formato *rtp://<ip>:<porta>?pkt_size=...*, com o formato mpegts encapsulado via RTP (-f rtp_mpegts), ajustando-se o pkt_size para evitar fragmentações IP.

Por fim, o protocolo RTMP utiliza URLs no padrão *rtmp://<ip>/live/stream*. Neste caso, o fluxo de vídeo é encapsulado em contêiner FLV[9], sendo publicado em um servidor (como o Nginx com módulo RTMP), o qual posteriormente serve o conteúdo aos

[8] A extensão `.pcap` refere-se a arquivos de captura de pacotes (*packet capture*), utilizados para armazenar dados de tráfego de rede coletados por ferramentas como *tcpdump* ou *Wireshark*. Esses arquivos registram os pacotes transmitidos e recebidos em uma interface de rede, permitindo posterior análise detalhada do comportamento da comunicação.

[9] FLV (Flash Video) é um contêiner de mídia desenvolvido pela Adobe para transmissão de áudio e vídeo por meio do protocolo RTMP.

receptores conectados ao mesmo ponto de publicação.

As URLs utilizadas no receptor (comando ffmpeg -i ...) variam conforme o protocolo adotado, refletindo a forma como a conexão é estabelecida:

O receptor SRT se conecta ao transmissor, que está em modo listener. Por isso, o receptor precisa especificar mode=caller na URL, indicando que ele iniciará a conexão.

```
srt://$CLI_SRT:$PORTA_SRT?mode=caller
```

No receptor RTP, o cliente escuta a porta indicada, sem necessidade de parâmetros adicionais.

```
rtp://$CLI_RTP:$PORTA_RTP
```

Já o protocolo RTMP, o receptor atua como cliente de um servidor de *streaming* (que neste caso é o Nginx). A URL especifica o endereço do servidor e o ponto de publicação (/live/stream).

```
rtmp://$CLI_RTMP:1935/live/stream
```

Código 3.8 – Fluxo resumido do script de transmissão/recepção de vídeo

```
case "$PROTO" in
  srt)
    # transmissor (listener); receptor (caller)
    ffmpeg -re -i "$VIDEO" -c:v libx264 -c:a aac -f mpegts \
      "srt://$SERV_SRT:$PORTA_SRT?mode=listener&pkt_size=1316" \
      >"$DIR/ffmpeg_tx_${TS}.log" 2>&1 &
    TX_PID=$!

    sleep 1
    ffmpeg -i "srt://$CLI_SRT:$PORTA_SRT?mode=caller" -c copy "$RECEBIDO" \
      >"$DIR/ffmpeg_rx_${TS}.log" 2>&1 &
    RX_PID=$!
    ;;
  rtp)
    # transmissor envia MPEG-TS encapsulado em RTP unicast
    ffmpeg -re -i "$VIDEO" -c:v libx264 -c:a aac -f rtp_mpegts \
      "rtp://${SERV_RTP}:$PORTA_RTP?pkt_size=1300" \
      >"$DIR/ffmpeg_tx_${TS}.log" 2>&1 &
    TX_PID=$!
    sleep 1
```

```
21      ffmpeg -i "rtp://${CLI_RTP}:$PORTA_RTP" -c copy "$RECEBIDO" \
22        >"$DIR/ffmpeg_rx_${TS}.log" 2>&1 &
23      RX_PID=$!
24      ;;
25    rtmp)
26      # reinicia Nginx c/ módulo RTMP e publica/consome fluxo FLV
27      sudo nginx -s stop &>/dev/null; sleep 5; sudo nginx
28      ffmpeg -re -i "$VIDEO" -c:v libx264 -c:a aac -f flv \
29        "rtmp://$SERV_RTMP:1935/live/stream" \
30        >"$DIR/ffmpeg_tx_${TS}.log" 2>&1 &
31      TX_PID=$!
32      sleep 1
33      ffmpeg -i "rtmp://$CLI_RTMP:1935/live/stream" -c copy "$RECEBIDO" \
34        >"$DIR/ffmpeg_rx_${TS}.log" 2>&1 &
35      RX_PID=$!
36      ;;
37  esac
```

3.4.3 Uso das ferramentas de vídeo e coleta de dados

A coleta e avaliação da qualidade dos vídeos transmitidos nos experimentos foram realizadas com auxílio do software *FFmpeg*, com suporte às bibliotecas `libx264`, `libx265` e `libvmaf`. Para cada experimento, os vídeos de entrada e saída foram comparados utilizando os filtros PSNR, SSIM e `libvmaf`, extraindo os respectivos valores quadro a quadro. As métricas objetivas de qualidade — PSNR e SSIM — foram armazenadas em arquivos separados e, posteriormente, processadas para obtenção dos valores médios.

Além disso, informações relacionadas ao comportamento da transmissão foram obtidas a partir dos logs do `FFmpeg` (tanto do transmissor quanto do receptor), bem como dos arquivos de captura de pacotes gerados pelo `tcpdump` e convertidos para o formato `CSV` com o uso do `tshark`. Esses dados foram utilizados para analisar a vazão média da transmissão e porcentagem de perda de pacotes.

3.4.4 Extração das métricas de qualidade e desempenho

O *script* `obter_metricas-v2.py`, desenvolvido em `Python`, percorre os diretórios de cada experimento para localizar e processar os arquivos de saída das transmissões de vídeo. A seguir, detalha-se o método de extração adotado para cada métrica:

- PSNR, SSIM e VMAF médios: os valores quadro a quadro são extraídos dos arquivos gerados pelo `ffmpeg` com os filtros correspondentes. O PSNR é extraído por meio de expressões regulares nos arquivos `psnr.stats`, o SSIM por meio de valores do tipo `All:0.XXXX`, e o VMAF é lido de arquivos `.json`, onde os valores de VMAF por quadro são obtidos e depois calculada sua média.

- **Porcentagem de perda de pacotes:** calculada pela diferença entre os pacotes transmitidos e recebidos, com base na seguinte fórmula:

$$\text{Perda (\%)} = \frac{\text{Transmitidos} - \text{Recebidos}}{\text{Transmitidos}} \times 100$$

  O valor é exibido com duas casas decimais de precisão.

  O número de pacotes transmitidos é estimado a partir do log do transmissor por meio do campo `pkts_sent=X` (ou, alternativamente, pelo maior valor de quadro codificado). No receptor, é utilizado o maior valor de quadro decodificado reportado no log do `ffmpeg` para determinar o total de pacotes recebidos.

- **jitter médio:** extraído por diferentes métodos adaptativos

- **Número de eventos de *rebuffering*:** detectados ao contar lacunas superiores a 100 ms entre dois pacotes sucessivos, a partir da análise dos timestamps de recepção (extraídos do arquivo `.csv` gerado pelo `tshark`). Em protocolos baseados em TCP, como RTMP, também são considerados eventos com mensagens explícitas de `buffering` ou `rebuffer` no log. No caso do RTP, é importante destacar que o protocolo em si não possui mecanismos que provoquem *rebuffering*. Os eventos contabilizados resultam do funcionamento do reprodutor de mídia utilizado, que implementa um *buffer* para compensar perdas e *jitter* durante a reprodução.

3.4.4.1 Cálculo do Jitter

O *jitter* foi calculado de forma adaptativa conforme a disponibilidade dos dados gerados na simulação:

- **log do receptor (`ffmpeg`)**: quando o receptor imprimia mensagens no formato `delayed for XX ms`, foi possível extrair diretamente os valores de atraso e calcular a média desses valores. Utilizado com o protocolo RTMP.

- **campo `jitter` no CSV do `tshark`**: no caso de transmissões RTP, o campo `rtp.interarrival_jitter` foi exportado junto à captura em formato `.csv`.

- **estimativa a partir de *timestamps***: na ausência dos métodos anteriores, o *jitter* foi estimado com base no desvio padrão dos intervalos de tempo entre pacotes sucessivos, calculado a partir dos *timestamps* relativos presentes na captura. O resultado foi convertido para milissegundos e assumido como aproximação do *jitter* médio. Este método foi utilizado no protocolo SRT, pois este não fornece diretamente um campo de *interarrival jitter*.

É importante destacar que, no caso do protocolo RTMP, baseado em TCP, o valor reportado pelo receptor não representa o *jitter* percebido pela aplicação em tempo real, mas

sim variações de atraso que foram compensadas pela retransmissão e ordenação de pacotes. Dessa forma, os resultados de *jitter* no RTMP devem ser interpretados apenas como um indicador indireto da instabilidade da rede, estando seu impacto prático refletido mais diretamente em métricas de latência e de *rebuffering*.

Os valores extraídos são apresentados no terminal e exportados para arquivos `CSV`, que posteriormente foram utilizados para a construção dos gráficos comparativos discutidos no Capítulo 5.

A automação da coleta e análise foi realizada por meio de um *script* auxiliar em *Python*, *obter_metricas.py*. Esse *script* percorre os diretórios dos experimentos e localiza automaticamente os arquivos de log do *ffmpeg*, os relatórios de qualidade objetiva (PSNR, SSIM, VMAF), e os arquivos de captura de pacotes exportados em `CSV` pelo `tshark`. Ele aplica expressões regulares para extrair dados dos logs, interpreta o campo `rtp.interarrival_jitter` quando disponível e, nos demais casos, calcula o *jitter* com base no desvio padrão dos intervalos entre pacotes sucessivos. O valor resultante é convertido para milissegundos e assumido como aproximação do *jitter* médio.

Além disso, o *script* identifica lacunas temporais superiores a 100 ms entre pacotes consecutivos para contabilizar eventos de *rebuffering*. Os resultados são exibidos no terminal de forma resumida e exportados para arquivos `.csv`, organizados por experimento.

A execução do *script* pode ser feita diretamente no terminal com o seguinte comando:

```
$ python3 obter_metricas.py capturas/srt_20250214_18-05-20
```

Esse comando irá processar os arquivos do experimento correspondente e gerar uma planilha consolidada com as métricas de qualidade e desempenho da transmissão de vídeo.

- Arquivos de métricas visuais: `psnr.txt`, `ssim.txt` e `vmaf.json`;
- Arquivos de log do transmissor e receptor: `ffmpeg_tx.log` e `ffmpeg_rx.log`;
- Arquivo CSV contendo as capturas do tcpdump, gerado pelo `tshark`.

A lógica do script pode ser resumida nas seguintes etapas:

1. **Localização de arquivos**: busca arquivos com nomes compatíveis com os padrões esperados.
2. **Extração de métricas visuais**: os valores médios de PSNR, SSIM e VMAF são calculados a partir dos arquivos de texto e JSON.

3. **Análise dos logs de transmissão**: o script estima a vazão média (*throughput*) com base nas amostras de *bitrate* extraídas dos logs do FFmpeg.
4. **Análise dos logs de recepção**: os logs do receptor são utilizados para determinar o número de quadros recebidos, o *jitter* médio extraído de mensagens de atraso (‘delayed for ... ms‘) e o número de eventos de *rebuffering* identificados por palavras-chave nos logs.
5. **Análise via CSV**: o arquivo CSV com estatísticas de rede é analisado para estimar o *jitter* com base na variação de *timestamps* consecutivos. O mesmo arquivo também pode ser usado para estimar o número de eventos de *rebuffering* com base em interrupções de tempo superiores a um limiar (tipicamente 100 ms).
6. **Cálculo de perdas**: caso o número de pacotes transmitidos e recebidos seja conhecido, o *script* calcula a quantidade e a porcentagem de pacotes perdidos.
7. **Apresentação dos resultados**: ao final da execução, o *script* exibe um resumo com todas as métricas extraídas.

```
Arquivo VMAF encontrado: capturas/rtp_20250720_23-14-23/vmaf.json
Total de frames carregados: 3068
Arquivo salvo com 3068 linhas em: capturas/rtp_20250720_23-14-23/vmaf_por_frame.txt
PSNR médio (dB): 17.67
SSIM médio: 0.6982
VMAF médio: 24.31

== Estatísticas de Transmissão/Recepção ==
Pacotes transmitidos: 3792
Pacotes recebidos: 3014
Pacotes perdidos: 778
Perda percentual: 20.52%
Jitter médio (ms): 32.71
Rebuffering (eventos): 254
```

Para medir a *vazão média de rede* de forma consistente entre os diferentes cenários experimentais, foi utilizado um *script* auxiliar denominado `calcular_vazao.py`, que opera sobre capturas de pacotes previamente processadas pelo `tshark`. Esse script espera como entrada um arquivo CSV contendo, obrigatoriamente, os campos `frame.time_relative` (*timestamp* relativo de cada quadro em segundos) e `frame.len` (tamanho de cada quadro em bytes).

A duração total da transmissão é obtida pela diferença entre o maior e o menor valor presente na coluna `frame.time_relative`, correspondendo ao tempo entre o primeiro e o último pacote capturado. Em seguida, calcula-se o total de dados transmitidos,

a partir da soma dos valores da coluna `frame.len`, que representa o tamanho de cada quadro em bytes.

Com essas duas informações, é possível estimar a vazão média da transmissão por meio da seguinte expressão:

$$\text{Vazão (Mbps)} = \frac{\text{Total de bytes} \times 8}{\text{Tempo total (s)} \times 10^6}$$

```
python3 calcular_vazao.py capturas/srt_20250723_19-52-03/srt_capture_20250723_19
    -52-03.csv
```

```
Arquivo analisado: capturas/srt_20250723_19-52-03/srt_capture_20250723_19-52-03.csv
Tempo total de transmissão: 105.90 segundos
Total de dados recebidos: 15474700 bytes
Vazão média: 1.169 Mbps
```

Esse valor representa a taxa efetiva de transferência de dados ao longo da transmissão, levando em conta todos os pacotes capturados.

4 ANÁLISE

Este capítulo apresenta a análise dos resultados obtidos a partir dos experimentos realizados, com base nas métricas de QoE e QoS definidas na metodologia. O objetivo é avaliar de forma comparativa o desempenho dos protocolos em diferentes condições de rede, incluindo cenários com mobilidade, perdas e tráfego de fundo. Para cada combinação experimental, foram executadas três repetições por protocolo, adotando-se na análise o resultado com maior valor de VMAF, de modo a garantir consistência e reduzir o impacto de variações pontuais nas medições.

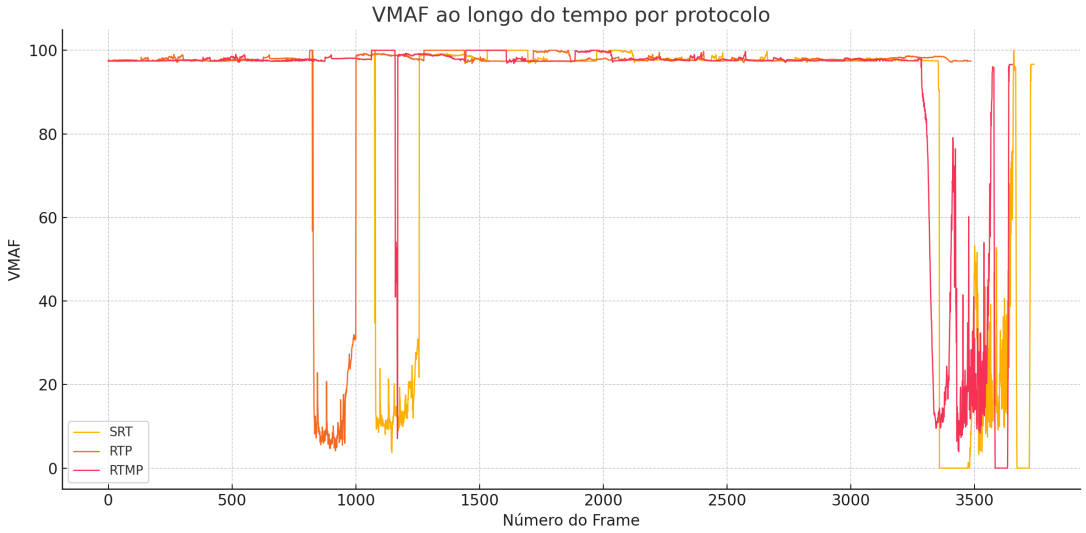
4.1 ANÁLISE SEM MOBILIDADE

Nos testes realizados no cenário estático, com usuários conectados por cabo (sem utilização de rede sem fio), foram desenvolvidos 3 conjuntos de experimentos, variando parâmetro de bitrate e compressão de vídeo, a fim de avaliar o comportamento dos protocolos RTMP, RTP e SRT com diferentes configurações.

4.1.1 Compressão de vídeo H.264 e bitrate de 4Mbps

A Figura 15 mostra a variação dos valores da métrica VMAF ao longo do tempo, quadro a quadro, para os três protocolos avaliados: RTP, RTMP e SRT. O protocolo RTP obteve o maior valor médio de qualidade (93,83), mas também apresentou uma taxa significativa de perda de pacotes (3,78%) e o maior número de eventos de *rebuffering* (202), conforme indicado na Tabela 10. O SRT, por sua vez, destacou-se por sua maior estabilidade ao longo da transmissão, registrando a menor taxa de perdas (1,55%) e uma curva de VMAF mais uniforme. Já o RTMP apresentou um valor médio de VMAF ligeiramente inferior ao do RTP (91,22), com perdas um pouco mais elevadas, mas conseguiu reduzir os eventos de *rebuffering* para 173. Embora o RTP tenha alcançado a melhor média de VMAF, essa vantagem pode ter sido ofuscada pela instabilidade observada durante a reprodução. O protocolo SRT, por outro lado, demonstrou um desempenho mais equilibrado entre qualidade de imagem e fluidez, oferecendo uma experiência potencialmente mais consistente para o usuário.

Figura 15 – Comparação de VMAF no Cenário 1.



Fonte: Elaborada pelo autor.

Tabela 9 – Métricas de qualidade de vídeo para o Cenário 1 – H.264 a 4 Mbps

Protocolo	**PSNR (dB)**	**SSIM**	**VMAF**
SRT	50,10	0,9508	85,94
RTP	41,58	0,9534	93,83
RTMP	49,94	0,9580	91,22

Fonte: Elaborada pelo autor.

Tabela 10 – Métricas de desempenho de rede e QoE para o Cenário 1 – H.264 a 4 Mbps

Protocolo	**Vazão Média (Mbps)**	**Perda (%)**	**Jitter (ms)**	**Rebuffering**
SRT	6,379	1,55%	12,51	174
RTP	6,416	3,78%	13,43	202
RTMP	8,590	3,95%	19,26	173

Fonte: Elaborada pelo autor.

Um segundo teste foi realizado com as mesmas configurações de vídeo do experimento anterior, mantendo a compressão H.264 e a taxa de bits de 4Mbps. A principal diferença neste cenário de simulação é a introdução de um atraso fixo de 50ms e uma taxa de perda de pacotes de 1%, com o objetivo de simular uma rede com maior latência e perdas moderadas.

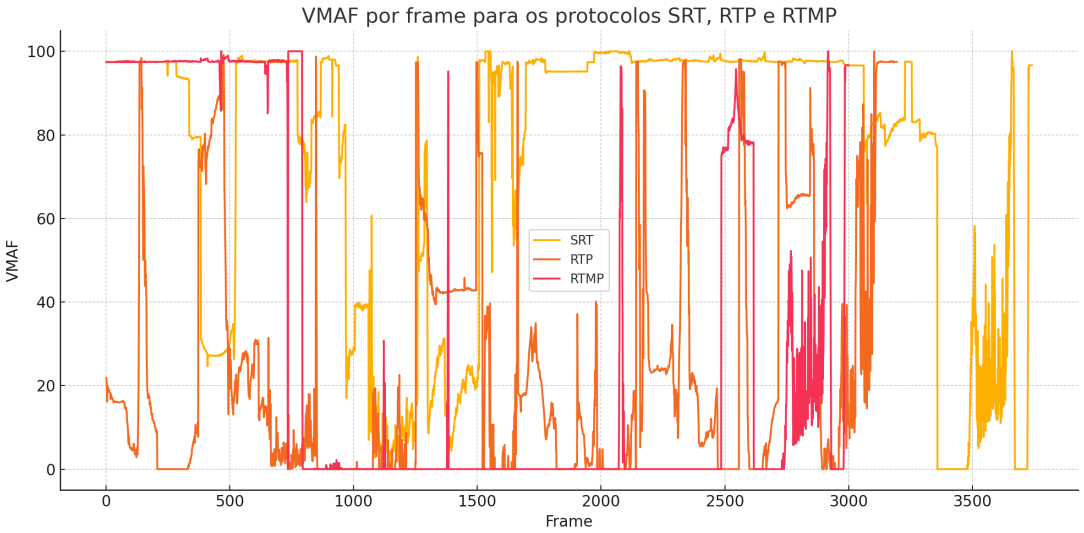
O protocolo SRT apresentou o melhor equilíbrio entre qualidade de vídeo e desempenho de rede. Obteve os maiores valores de PSNR (41,08dB), SSIM (0,9214) e VMAF (73,36), com vazão média de (6,995Mbps), baixa taxa de perda (1,73%) e eventos de rebuffering de 68.

O protocolo RTMP teve desempenho inferior ao SRT, com valores mais baixos de VMAF (32,24), quantidade de rebuffering de 23 eventos, e uma taxa de perda extrema-

mente elevada (62,98%), evidenciando vulnerabilidade em redes instáveis. Ainda assim, sua vazão foi razoável (5,294 Mbps).

Já o protocolo **RTP** apresentou o VMAF inferior aos demais, apenas 23,96, SSIM de 0,6521 e PSNR de 16,64dB, além de maior jitter (11,72ms) e o maior número de eventos de rebuffering (143).

Figura 16 – Comparação de VMAF no Cenário 1 - adição de delay e perda de pacotes.



Fonte: Elaborada pelo autor.

Tabela 11 – Métricas de qualidade de vídeo para o Cenário 1 – adição de delay e perda de pacotes

Protocolo	**PSNR (dB)**	**SSIM**	**VMAF**
SRT	41,08	0,9214	73,36
RTP	16,64	0,6521	23,96
RTMP	25,72	0,7726	32,24

A diferença acentuada nas métricas observadas neste cenário decorre principalmente da forma como cada protocolo lida com perdas e atrasos na rede, bem como das particularidades na medição das métricas.

No caso do RTP, a ausência de mecanismos nativos de correção de erros faz com que perdas mesmo moderadas (1%) provoquem a corrupção de quadros de referência no fluxo H.264, degradando significativamente métricas como PSNR, SSIM e VMAF. Essa degradação é amplificada pelo uso de estruturas de *GOP* longas, pois a perda de um único quadro-chave pode comprometer vários segundos de vídeo.

O SRT, por sua vez, utiliza retransmissão seletiva (*ARQ*) e controle de congestionamento para recuperar pacotes perdidos, preservando a qualidade visual e resultando em

métricas objetivas elevadas. Entretanto, o uso de um *latency buffer* relativamente curto frente ao atraso fixo de 50 ms e ao *jitter* residual provoca diversos eventos de *rebuffering* de curta duração, aumentando a contagem total de interrupções, embora o impacto visual seja pequeno. Além disso, as retransmissões elevam a vazão média medida.

Já no RTMP, baseado em TCP, as perdas na rede são mascaradas pela pilha de transporte, que garante a entrega de todos os pacotes. No entanto, o controle de retransmissão e reordenação aumenta a latência e pode ocasionar interrupções prolongadas no reprodutor, levando ao descarte de *buffers* e à contabilização de "perda" pela metodologia empregada, mesmo sem perda efetiva no enlace. Isso explica o valor de perda registrado (62,98%), que reflete mais o impacto do atraso na reprodução do que perdas de rede propriamente ditas.

Dessa forma, os valores observados não indicam apenas desempenho bruto, mas também revelam como cada protocolo equilibra qualidade visual, resiliência a perdas e continuidade da reprodução em condições adversas.

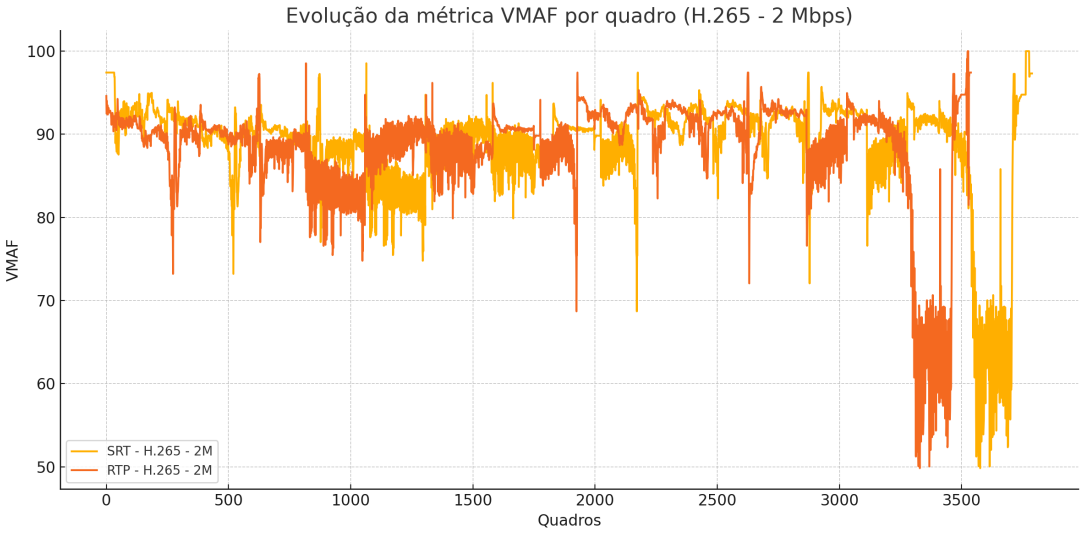
Tabela 12 – Métricas de desempenho de rede e Cenário – H.264 a 4 Mbps com delay

Protocolo	**Vazão Média (Mbps)**	**Perda (%)**	**Jitter (ms)**	**Rebuffering**
SRT	6,995	1,73	8,77	68
RTP	6,589	4,89	11,72	143
RTMP	5,294	62,98	10,16	23

4.1.2 Compressão de vídeo H.265 e bitrate de 2Mbps

A Figura 17 apresenta o gráfico apenas dos protocolos RTP e SRT devido à compressão de vídeo H.265 que não é suportada nativamente pelo protocolo RTMP. Ambos os protocolos mantiveram um bom desempenho no que se refere a médias de VMAF, que ficou em torno de 88. Os dados das Tabela 13 e Tabela 14 conseguem destacar os desempenhos individualmente: o SRT obteve valores mais altos de PSNR (40,62dB) e SSIM (0,9799), além de registrar uma perda de pacotes menor (1,13%) e um *jitter* médio mais baixo (8,19ms), resultando em apenas dois eventos de *rebuffering*. Já o RTP apresentou maior instabilidade na rede, com perda de 6,65%, *jitter* de 19,71 ms e 28 eventos de *rebuffering*.

Figura 17 – Comparação de VMAF no Cenário 1.



Fonte: Elaborada pelo autor.

Tabela 13 – Métricas de qualidade de vídeo para o Cenário 1 – H.265 a 2 Mbps

Protocolo	**PSNR (dB)**	**SSIM**	**VMAF**
SRT	40,62	0,9799	88,47
RTP	36,78	0,9619	88,13

Fonte: Elaborada pelo autor.

Tabela 14 – Métricas de desempenho de rede para o Cenário 1 – H.265 a 2 Mbps

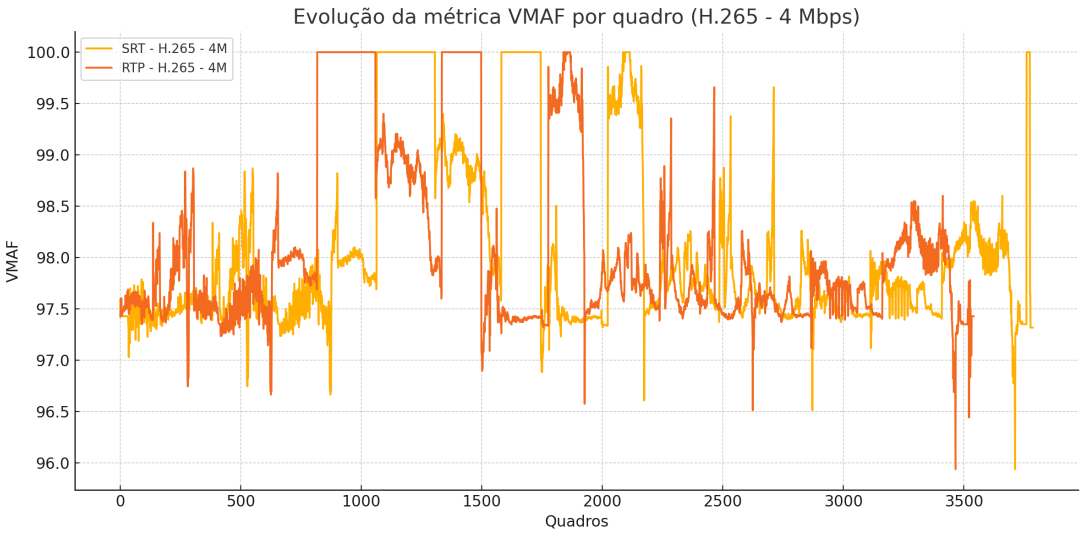
Protocolo	**Vazão Média (Mbps)**	**Perda (%)**	**Jitter (ms)**	**Rebuffering**
SRT	1,197	1,13%	8,19	2
RTP	1,101	6,65%	19,71	28

Fonte: Elaborada pelo autor.

4.1.3 Compressão de vídeo H.265 e bitrate de 4Mbps

Na Figura 18, observa-se a evolução da qualidade dos vídeos transmitidos com codificação H.265 a 4 Mbps. Tanto o SRT quanto o RTP mantiveram níveis elevados de VMAF ao longo da transmissão, com média acima de 98. Porém, o gráfico apresenta o RTP com oscilações mais visíveis ao longo do tempo, enquanto o SRT manteve os valores praticamente constante. Os demais indicadores, presente nas Tabela 15 e Tabela 16 também ajudam a explicar essa diferença: o SRT apresentou um PSNR superior (63,20 dB) e um SSIM de (0,9997), além de taxa de perda de pacotes (1,08%) e apenas três eventos de *rebuffering*. O RTP, por outro lado, teve mais perdas (6,59%) e mais *rebuffering* (10 eventos), o que pode justificar as quedas momentâneas na qualidade perceptiva observadas no gráfico.

Figura 18 – Comparação de VMAF no Cenário 1, compressão em H.265 e bitrate de 4Mbps



Fonte: Elaborada pelo autor.

Tabela 15 – Métricas de qualidade de vídeo para o Cenário 1 – H.265 a 4 Mbps

Protocolo	**PSNR (dB)**	**SSIM**	**VMAF**
SRT	63,20	0,9997	98,08
RTP	42,16	0,9749	98,11

Fonte: Elaborada pelo autor.

Tabela 16 – Métricas de desempenho de rede e QoE para o Cenário 1 – H.265 a 4 Mbps

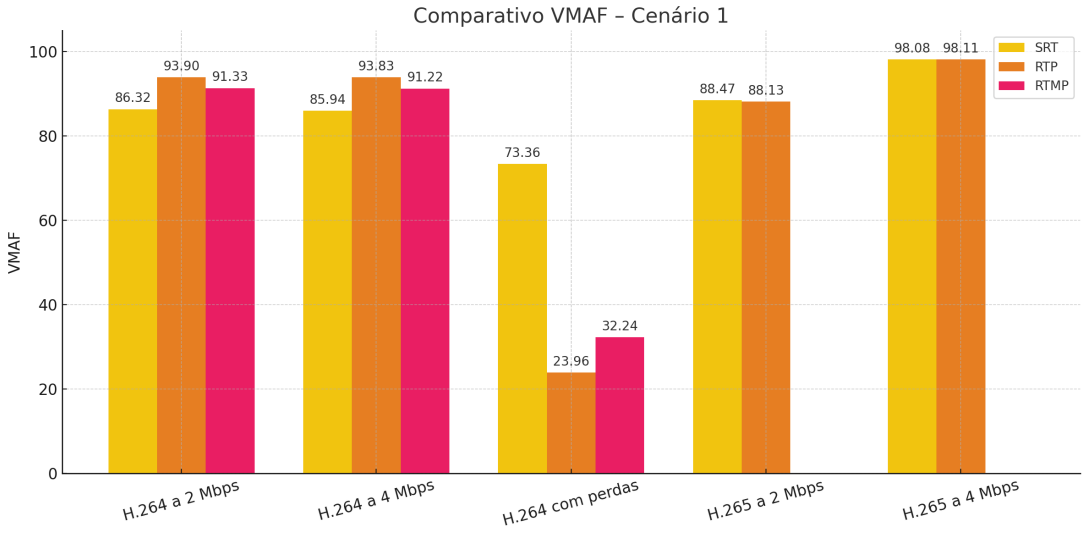
Protocolo	**Vazão Média (Mbps)**	**Perda (%)**	**Jitter (ms)**	**Rebuffering**
SRT	8,647	1,08%	4,14	3
RTP	8,629	6,59%	4,94	10

Fonte: Elaborada pelo autor.

4.1.4 Comparação de VMAF no Cenário 1

Em resumo, a Figura 19 apresenta uma comparação dos valores de VMAF obtidos nos diferentes experimentos. Observa-se que, em condições ideais com H.264, o protocolo RTP apresentou os maiores valores de médio de VMAF, seguido de perto pelo RTMP e SRT. No entanto, com a adição de perdas e atraso, o desempenho do RTP caiu drasticamente, enquanto o SRT manteve uma qualidade mais estável. Com a mudança para o codec H.265, tanto SRT quanto RTP atingiram valores de VMAF elevados, superiores a 98, mesmo em taxas de bits distintas, demonstrando a eficácia da compressão H.265 na preservação da qualidade visual.

Figura 19 – Comparação de VMAF no Cenário 1 - adição de delay e perda de pacotes.



Fonte: Elaborada pelo autor.

4.2 ANÁLISE COM MOBILIDADE

Os testes com mobilidade tiveram como objetivo avaliar o impacto do deslocamento dos usuários e das variações na qualidade do canal sobre o desempenho dos protocolos RTMP e SRT. Para essa análise, utilizou-se a simulação com mobilidade representada na Figura 13.

4.2.1 Mobilidade de um único host

Nesta configuração, apenas um dos *hosts* envolvidos na transmissão de vídeo permanece em movimento, simulando um cenário no qual o usuário se desloca durante o consumo de conteúdo audiovisual. Ao longo do trajeto, ocorrem eventos de *handover* entre os roteadores, os quais podem comprometer a estabilidade da conexão. Para a análise dos impactos da compressão e da taxa de *bitrate* na qualidade da transmissão, foram conduzidos experimentos com diferentes configurações de codificação de vídeo. Os resultados obtidos com compressão H.264 e taxa de *bitrate* variável são apresentados a seguir.

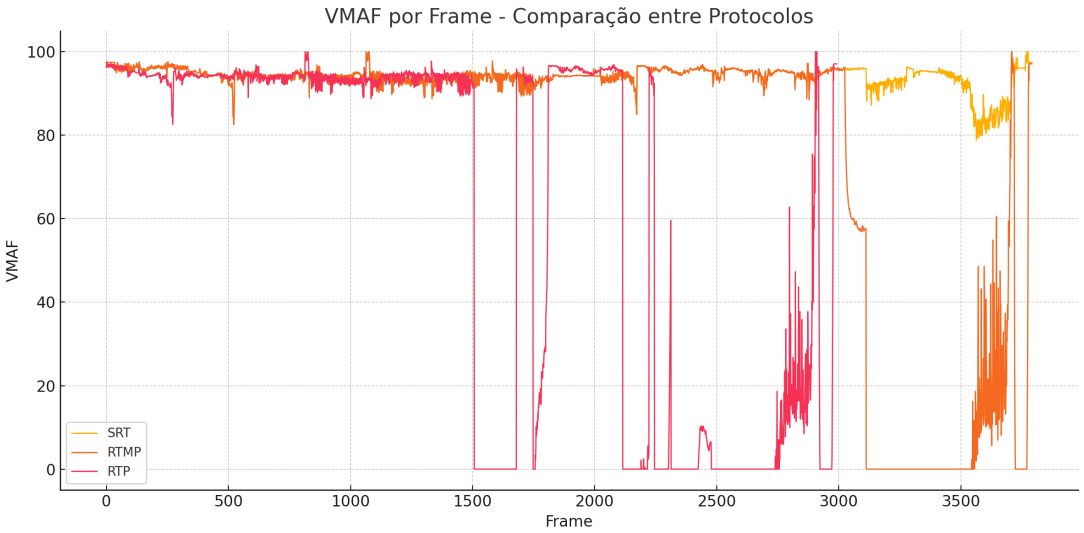
4.2.1.1 Cenário umMovel - Compressão de vídeo H.264 e bitrate variável

A Figura 20 apresenta as métricas de VMAF ao longo do tempo para os três protocolos, permitindo uma análise comparativa da qualidade perceptual dos vídeos transmitidos. O protocolo SRT obteve os valores mais elevados de VMAF, com quedas pontuais rapidamente compensadas, indicando capacidade de recuperação frente a perdas e *jitter*, sem comprometimento significativo da qualidade. A Tabela 17 confirma esse desempenho, com valor médio de VMAF igual a 94,01 e taxa de perda de 0%, em conformidade com o comportamento observado no gráfico.

O protocolo RTMP apresentou desempenho intermediário, com valores iniciais de VMAF elevados, seguidos por quedas perceptíveis após os quadros 1500 e 2000. Apesar de manter níveis de qualidade considerados aceitáveis, a maior frequência de oscilações, especialmente após o quadro 3000, indica menor estabilidade frente às variações da rede. Esse comportamento é compatível com os valores médios registrados em tabela, nos quais o VMAF para o RTMP permaneceu em torno de 78.

O protocolo RTP apresentou as maiores oscilações e as quedas mais acentuadas nos valores de VMAF entre os três protocolos analisados, incluindo trechos com ausência total de quadros. Esse comportamento decorre da falta de mecanismos nativos de retransmissão e correção de erros, resultando em uma taxa de perda de pacotes de 34,97%.

Figura 20 – Comparação de VMAF no Cenário com um móvel e compressão H.264 e bitrate variável



Fonte: Elaborada pelo autor.

Tabela 17 – Comparação de qualidade de vídeo para o cenário *UmMovel* – H.264

Protocolo	**PSNR (dB)**	**SSIM**	**VMAF**
SRT	42,73	0,9833	94,01
RTP	28,99	0,8843	62,68
RTMP	37,80	0,9443	78,52

Fonte: Elaborada pelo autor.

Tabela 18 – Métricas de transmissão para o cenário *UmMovel* – H.264

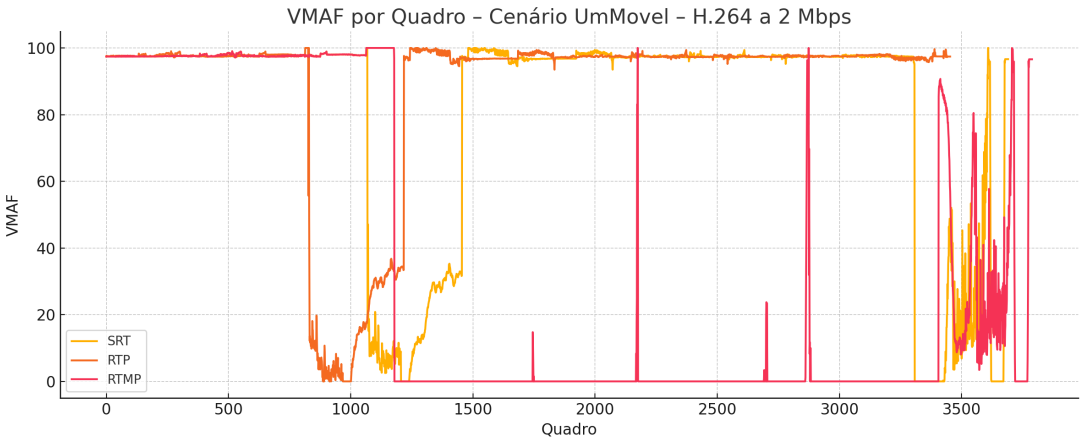
Protocolo	**Vazão Média (Mbps)**	**Perda**	**Jitter (ms)**	**Rebuffering**
SRT	1,313	0,00%	21,32	219
RTP	1,107	34,97%	33,92	249
RTMP	1,367	20,23%	77,09	223

Fonte: Elaborada pelo autor.

4.2.1.2 *Cenário umMovel - Compressão de vídeo H.264 e bitrate 2Mbps*

O gráfico de VMAF para este cenário, com compressão H.264 a 2Mbps, apresentado na Figura 21, evidencia diferenças significativas no comportamento dos três protocolos analisados. Conforme os dados da Tabela 19, o protocolo SRT manteve qualidade relativamente estável, com VMAF médio de 80,89, mesmo diante de uma taxa de perda de 3,05%, demonstrando resiliência frente às variações da rede. O protocolo RTP obteve a maior média de VMAF (88,42) e a menor taxa de perda (2,80%), porém apresentou oscilações perceptíveis ao longo da transmissão. Tais flutuações podem estar associadas ao maior *jitter* registrado (15,76ms) e ao elevado número de eventos de *rebuffering* (179), os quais comprometem a fluidez da reprodução. Já o RTMP demonstrou desempenho inferior, com VMAF médio de apenas 34,32 e alta taxa de perda (16,45%), resultando em qualidade visual degradada e instável. Esses resultados indicam que, embora o RTP apresente a melhor média de qualidade visual, o SRT proporciona uma experiência mais equilibrada e fluida, enquanto o RTMP se revela o menos adequado para cenários com mobilidade e compressão H.264 a 2 Mbps.

Figura 21 – Comparação de VMAF no Cenário com um móvel e compressão H.264 e bitrate 2Mbps



Fonte: Elaborada pelo autor.

Tabela 19 – Métricas de qualidade de vídeo para o Cenário UmMovel – H.264 a 2 Mbps

Protocolo	**PSNR (dB)**	**SSIM**	**VMAF**
SRT	44,71	0,9344	80,89
RTP	39,52	0,9418	88,42
RTMP	28,98	0,7967	34,32

Fonte: Elaborada pelo autor.

Tabela 20 – Métricas de desempenho de rede para o Cenário UmMovel – H.264 a 2 Mbps

Protocolo	**Vazão Média (Mbps)**	**Perda (%)**	**Jitter (ms)**	**Rebuffering**
SRT	3,672	3,05%	11,15	89
RTP	3,736	2,80%	15,76	179
RTMP	4,944	16,45%	12,95	111

Fonte: Elaborada pelo autor.

A análise deste cenário tem como objetivo verificar o impacto da mobilidade na qualidade percebida do vídeo (QoE) e na estabilidade da rede (QoS) para diferentes protocolos de transmissão, mantendo as mesmas condições de compressão e *bitrate*. Ao comparar SRT, RTP e RTMP sob essas condições, busca-se identificar qual protocolo oferece o melhor compromisso entre qualidade visual e continuidade da reprodução quando a rede está sujeita a variações causadas pelo deslocamento do nó móvel. Os resultados permitem compreender não apenas o valor médio das métricas, mas também a forma como cada protocolo reage às flutuações de atraso, perda e *jitter*, fornecendo subsídios para recomendar seu uso em cenários práticos de *streaming* ao vivo.

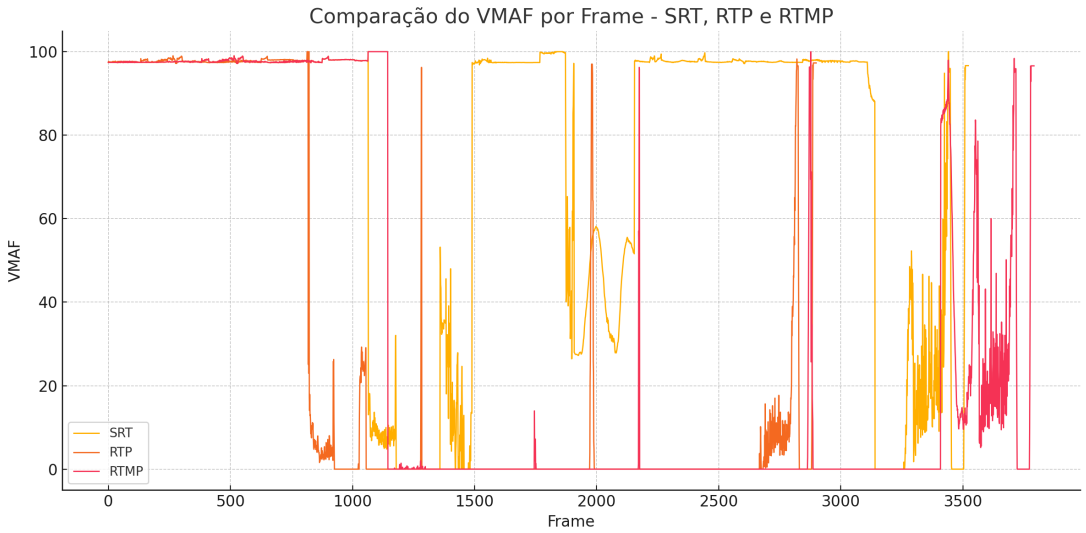
4.2.1.3 Cenário umMovel - Compressão de vídeo H.264 e bitrate 4 Mbps

Os resultados obtidos com compressão H.264 a 4Mbps no cenário com um host em movimento, apresentados na Figura 22, revelam diferenças marcantes entre os protocolos analisados. O protocolo SRT manteve a maior qualidade de vídeo, com média de VMAF de 73,91, PSNR de 43,28dB e SSIM de 0,907, conforme evidenciado na Tabela 21, mesmo diante de uma taxa de perda de 8,80% e 50 eventos de *rebuffering*, conforme a Tabela 22. Esse desempenho reforça a resiliência do protocolo frente às variações impostas pela mobilidade.

Em contrapartida, o protocolo RTP apresentou o menor desempenho entre os três, com VMAF médio de apenas 30,12, perda de 65,21% e 177 eventos de *rebuffering*, apesar de sua vazão ser comparável à do SRT. A elevada taxa de perda e o número expressivo de interrupções indicam que a mobilidade afeta severamente a estabilidade do RTP, comprometendo a experiência de reprodução.

Já o protocolo RTMP demonstrou desempenho intermediário. Embora tenha alcançado VMAF médio superior ao do RTP (33,49), permaneceu consideravelmente abaixo do valor obtido com SRT. Com taxa de perda de 35,66%, *jitter* de 17,5 ms e 47 eventos de *rebuffering*, o protocolo apresentou instabilidades perceptíveis durante a reprodução, também refletidas nas quedas abruptas de qualidade visíveis na Figura 22.

Figura 22 – Comparação de VMAF no Cenário com um móvel e compressão H.264 e bitrate 4Mbps



Fonte: Elaborada pelo autor.

Tabela 21 – Métricas de qualidade de vídeo para os protocolos no cenário *UmMovel* com H.264 a 4 Mbps

Protocolo	**PSNR (dB)**	**SSIM**	**VMAF**
SRT	43,28	0,907	73,91
RTP	24,88	0,7718	30,12
RTMP	28,71	0,7953	33,49

Fonte: Elaborada pelo autor.

Tabela 22 – Métricas de desempenho de rede para os protocolos no cenário *UmMovel* com H.264 a 4 Mbps

Protocolo	**Vazão (Mbps)**	**Perda (%)**	**Jitter (ms)**	**Rebuffering**
SRT	5,619	8,80	9,2	50
RTP	4,712	65,21	14,21	177
RTMP	6,812	35,66	17,5	47

Fonte: Elaborada pelo autor.

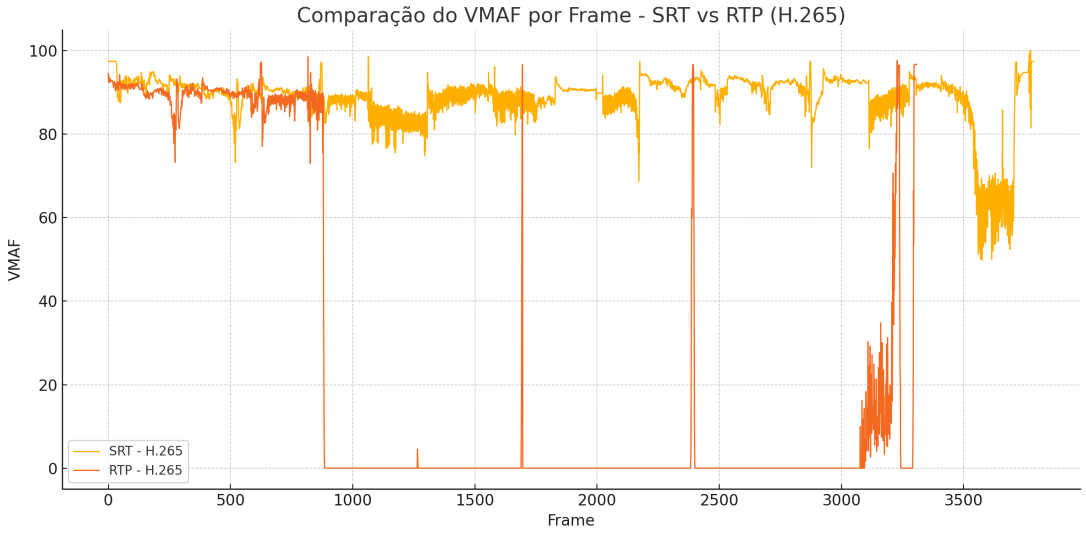
4.2.1.4 Cenário umMovel - Compressão de vídeo H.265 e bitrate variável

A avaliação do desempenho dos protocolos SRT e RTP no cenário com compressão H.265 evidenciou diferenças significativas em termos de qualidade de vídeo e adaptação às condições da rede. Conforme indicado na Tabela 23, o protocolo SRT apresentou os melhores resultados, com valores elevados de PSNR (40,77 dB), SSIM (0,98) e VMAF (88,45), indicando uma transmissão com qualidade visual próxima à referência.

Por outro lado, o protocolo RTP obteve desempenho consideravelmente inferior, com PSNR de 20,73 dB, SSIM de 0,733 e VMAF médio de apenas 25,75. Esses resultados podem ser explicados pelas métricas de rede observadas na Tabela 24. O protocolo SRT demonstrou estabilidade, com vazão média de 0,495 Mbps, perda de apenas 0,29%, *jitter* de 7,02 ms e apenas dois eventos de *buffering* ao longo da transmissão.

Em contraste, o protocolo RTP, apesar de manter vazão similar (0,837 Mbps), apresentou elevada taxa de perdas (61,74%), *jitter* médio de 21,49 ms e um total de 107 eventos de *buffering*, comprometendo significativamente a fluidez da reprodução. A Figura 23 ilustra essas diferenças, evidenciando uma curva de VMAF estável e alta no caso do SRT, e oscilações abruptas com quedas acentuadas no caso do RTP, em consonância com os dados das tabelas.

Figura 23 – Comparação de VMAF no Cenário com um móvel e compressão H.265 e *bitrate* variável



Fonte: Elaborada pelo autor.

Tabela 23 – Métricas de qualidade de vídeo para os protocolos no cenário *UmMovel* com compressão H.265

Protocolo	**PSNR (dB)**	**SSIM**	**VMAF**
SRT	40,77	0,980	88,45
RTP	20,73	0,733	25,75

Fonte: Elaborada pelo autor.

Tabela 24 – Métricas de desempenho de rede para os protocolos no cenário *UmMovel* com compressão H.265

Protocolo	**Vazão (Mbps)**	**Perda (%)**	**Jitter (ms)**	**Rebuffering**
SRT	1,206	0,29	7,02	2
RTP	0,837	61,74	21,49	107

Fonte: Elaborada pelo autor.

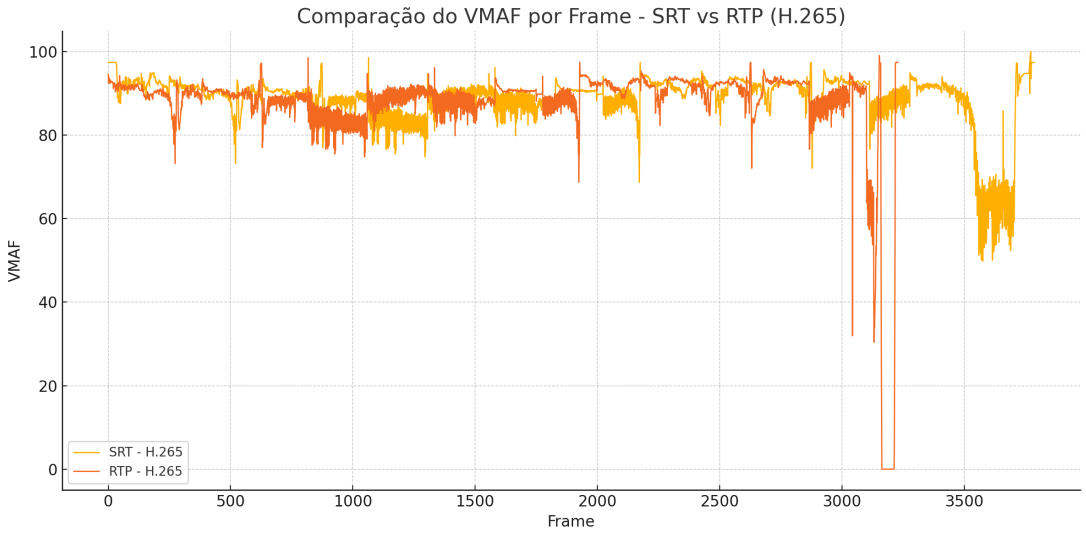
4.2.1.5 Cenário umMovel - Compressão de vídeo H.265 e bitrate 2Mbps

A Tabela 25 apresenta diferenças relevantes no comportamento dos protocolos SRT e RTP sob compressão H.265 a 2 Mbps. Ambos alcançaram valores elevados de qualidade perceptual: o protocolo SRT registrou PSNR de 40,71 dB, SSIM de 0,9800 e VMAF de 88,46, enquanto o RTP apresentou valores ligeiramente inferiores, com PSNR de 34,73 dB, SSIM de 0,9476 e VMAF de 87,27. Apesar da proximidade nos valores médios de VMAF, a estabilidade das transmissões foi distinta entre os protocolos.

Conforme os dados da Tabela 26, o SRT apresentou desempenho superior em termos de robustez à degradação da rede, com taxa de perda de 0,42%, *jitter* médio de 7,07 ms e apenas dois eventos de *rebuffering*, mantendo uma vazão média de 0,495 Mbps. Já o protocolo RTP registrou maior instabilidade, com 11,58% de perdas, *jitter* médio de 17,86 ms e 37 eventos de *rebuffering*, indicando menor tolerância a variações na qualidade do canal.

O gráfico da Figura 24 corrobora essa análise: embora ambos os protocolos tenham iniciado a transmissão com valores elevados de VMAF, o RTP apresentou oscilações mais frequentes ao longo dos *frames*, ao passo que o SRT manteve uma qualidade visual mais estável, demonstrando maior eficiência nos mecanismos de correção de erros e controle de transmissão em ambientes com mobilidade.

Figura 24 – Comparação de VMAF no cenário UmMovel e compressão H.265 e bitrate 2Mbps



Fonte: Elaborada pelo autor.

Tabela 25 – Métricas de qualidade de vídeo para o cenário UmMovel – H.265 a 2 Mbps

Protocolo	**PSNR (dB)**	**SSIM**	**VMAF**
SRT	40,71	0,9800	88,46
RTP	34,73	0,9476	87,27

Fonte: Elaborada pelo autor.

Tabela 26 – Métricas de desempenho de rede cenário UmMovel – H.265 a 2 Mbps

Protocolo	**Vazão Média (Mbps)**	**Perda (%)**	**Jitter (ms)**	**Rebuffering**
SRT	1,206	0,42%	7,07	2
RTP	1,088	11,58%	17,86	37

Fonte: Elaborada pelo autor.

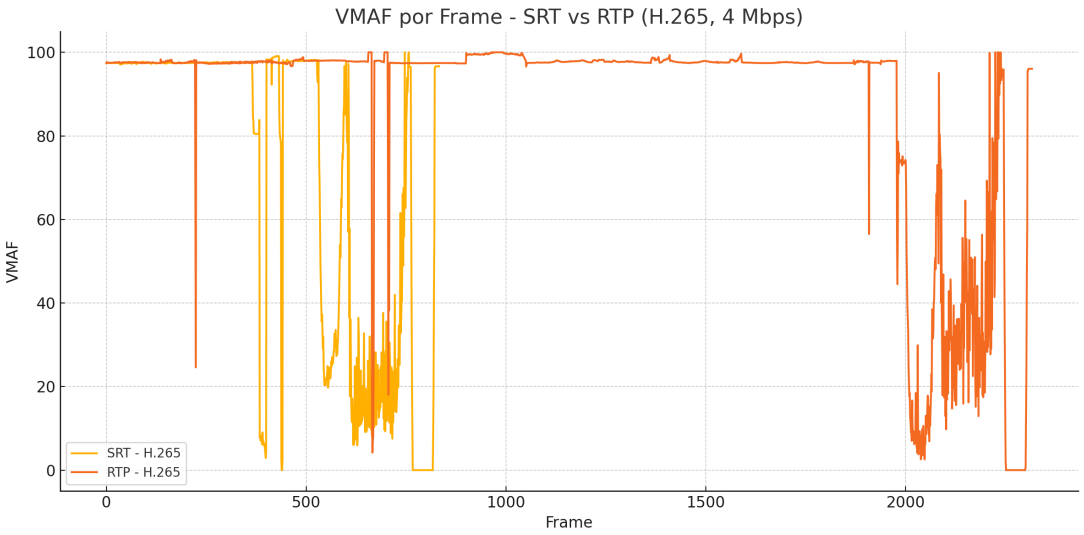
4.2.1.6 Cenário umMovel - Compressão de vídeo H.265 e bitrate 4Mbps

A Tabela 27 apresenta os resultados obtidos com compressão H.265 a 4 Mbps no cenário *UmMovel*, nos quais se verifica uma inversão no desempenho dos protocolos em relação a testes anteriores. O protocolo RTP registrou a melhor qualidade média de vídeo, com VMAF de 88,35, PSNR de 32,98 dB e SSIM de 0,8402.

Os dados da Tabela 28 indicam que o SRT apresentou uma taxa de perda mais elevada (51,94%), enquanto o RTP obteve 23,76%. Apesar da menor perda, o protocolo RTP acumulou mais eventos de *rebuffering* (6 contra 2), o que impacta a fluidez da reprodução. O *jitter* médio manteve-se em torno de 5 ms para ambos os protocolos, sugerindo que as diferenças de desempenho estão mais relacionadas à resiliência de cada protocolo às perdas e instabilidades do canal do que à largura de banda efetiva.

A Figura 25 reforça essa análise ao mostrar instabilidade acentuada na transmissão por SRT, com quedas bruscas de VMAF nos primeiros 1000 *frames*. Em contraste, o protocolo RTP apresentou uma curva de qualidade mais uniforme ao longo da maior parte do vídeo, com degradação mais pronunciada apenas na parte final da transmissão.

Figura 25 – Comparação de VMAF no cenário UmMovel e compressão H.265 e bitrate 4Mbps



Fonte: Elaborada pelo autor.

Tabela 27 – Métricas de qualidade de vídeo para o cenário UmMovel – H.265 a 4 Mbps

Protocolo	**PSNR (dB)**	**SSIM**	**VMAF**
SRT	20,84	0,6843	70,69
RTP	32,98	0,8402	88,35

Fonte: Elaborada pelo autor.

Tabela 28 – Métricas de desempenho de rede para o cenário UmMovel – H.265 a 4 Mbps

Protocolo	**Vazão Média (Mbps)**	**Perda (%)**	**Jitter (ms)**	**Rebuffering**
SRT	5,716	51,94	5,63	2
RTP	6,968	23,76	5,17	6

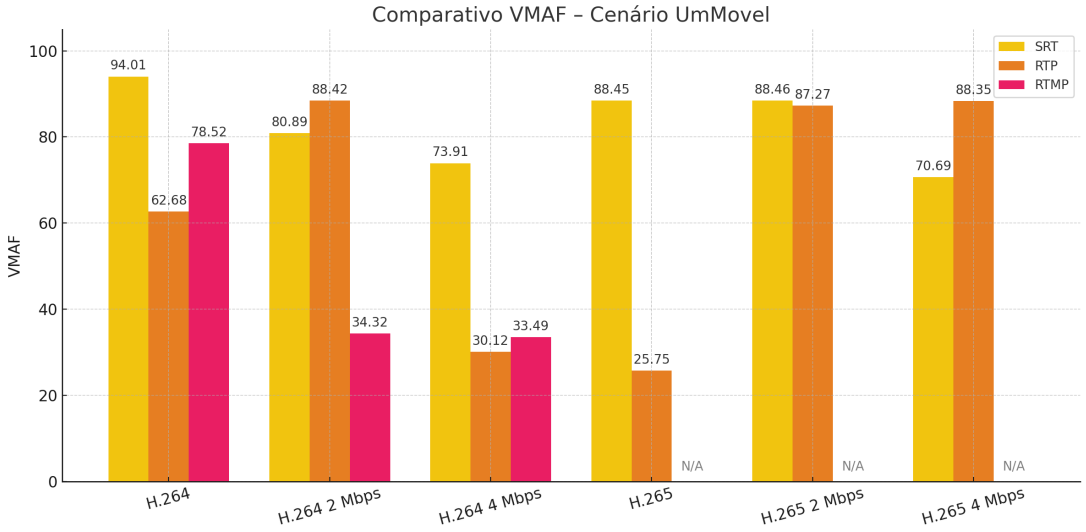
Fonte: Elaborada pelo autor.

4.2.2 Comparação de VMAF no Cenário *umMovel*

A Figura 26 apresenta os resultados comparativos entre os protocolos, considerando diferentes taxas de compressão e codificações de vídeo. O protocolo SRT demonstrou desempenho consistente, mantendo valores elevados de VMAF em todos os testes, com destaque para a codificação H.265 a 2 Mbps, na qual obteve VMAF de 88,46. Em

contrapartida, o protocolo RTP apresentou grande variabilidade: obteve VMAF de 88,42 com H.264 a 2 Mbps, porém o valor caiu significativamente para 30,12 na configuração com H.264 a 4 Mbps, evidenciando maior sensibilidade a variações no cenário. O protocolo RTMP, por sua vez, apresentou desempenho intermediário, com resultados razoáveis especialmente nas configurações baseadas em H.264.

Figura 26 – Comparação de VMAF no cenário UmMovel



Fonte: Elaborada pelo autor.

4.2.3 Mobilidade de um único host com tráfego de fundo

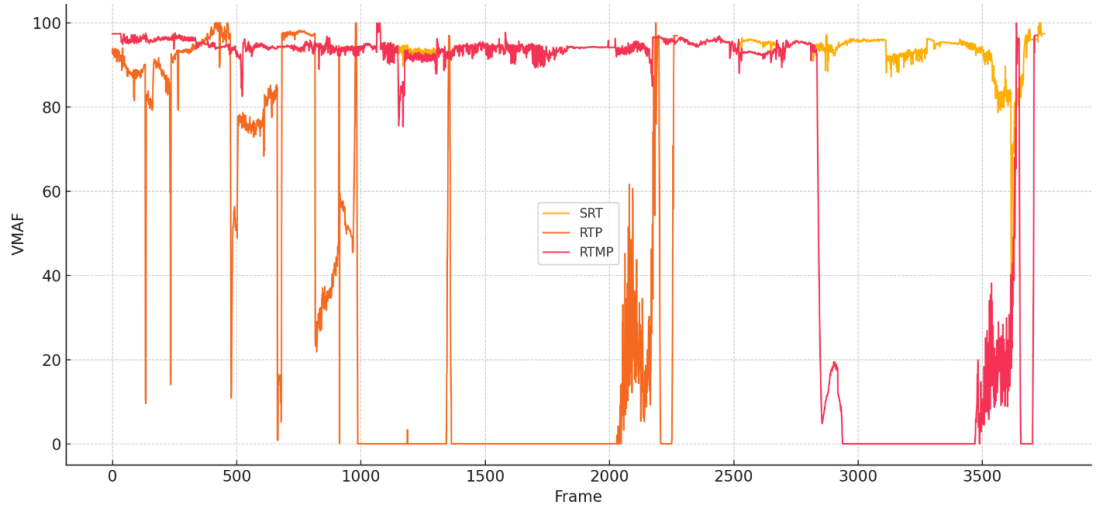
Nesta etapa dos experimentos, além da mobilidade de um único host, foi introduzido tráfego de fundo na rede, com o intuito de avaliar o impacto da concorrência por largura de banda no desempenho dos protocolos SRT e RTP. A inclusão desse tráfego visa simular condições mais realistas de operação, em que múltiplas aplicações coexistem e competem por largura de banda, resultando em variações adicionais de latência, perdas e *jitter*. Dessa forma, pretende-se analisar a resiliência e a eficiência dos protocolos em cenários com maior grau de degradação. Os testes mantiveram o modelo de mobilidade circular utilizado nas etapas anteriores, com variações nos parâmetros de compressão de vídeo e taxa de transmissão.

4.2.3.1 Cenário doisMoveis - Compressão de vídeo H.264 e bitrate variável

Neste cenário, o gráfico apresentado na Figura 27, juntamente com os dados das Tabela 29 e Tabela 30, revela diferenças significativas entre os protocolos analisados. O protocolo SRT apresentou os melhores resultados de qualidade de vídeo, com VMAF de 93,98, SSIM de 0,9828 e PSNR de 42,66 dB, acompanhado de uma vazão média de 1,320 Mbps e elevado número de eventos de *rebuffering* (198). A curva exibida no gráfico indica variações entre 80 e 100 ao longo da transmissão, sem quedas significativas.

O protocolo RTP, embora tenha atingido a maior vazão (6,127 Mbps) e número reduzido de eventos de *rebuffering* (5), registrou perdas expressivas (52%), o que resultou em uma qualidade perceptual inferior (VMAF de 37,88), conforme evidenciado pelas quedas abruptas observadas na representação gráfica. Já o protocolo RTMP apresentou desempenho intermediário, com VMAF de 73,82, SSIM de 0,9348, perda de 25,24%, *jitter* elevado (109,48ms) e 225 eventos de *rebuffering*. Ressalta-se uma interrupção prolongada na qualidade visual próxima ao *frame* 3000, que se estendeu até aproximadamente o *frame* 3500.

Figura 27 – Comparação de VMAF no cenário *umMovel* e tráfego de fundo - Compressão H.264 e bitrate variável



Fonte: Elaborada pelo autor.

Tabela 29 – Métricas de qualidade de vídeo para o cenário *umMovel* e tráfego de fundo – H.264

Protocolo	**PSNR (dB)**	**SSIM**	**VMAF**
SRT	42,66	0,9828	93,98
RTP	23,96	0,7541	37,88
RTMP	36,39	0,9348	73,82

Fonte: Elaborada pelo autor.

Tabela 30 – Métricas de desempenho de rede com cenário *umMovel* e tráfego de fundo – H.264

Protocolo	**Vazão Média (Mbps)**	**Perda (%)**	**Jitter (ms)**	**Rebuffering**
SRT	1,320	1,08	20,61	198
RTP	6,127	52,00	5,30	5
RTMP	1,278	25,24	109,48	225

Fonte: Elaborada pelo autor.

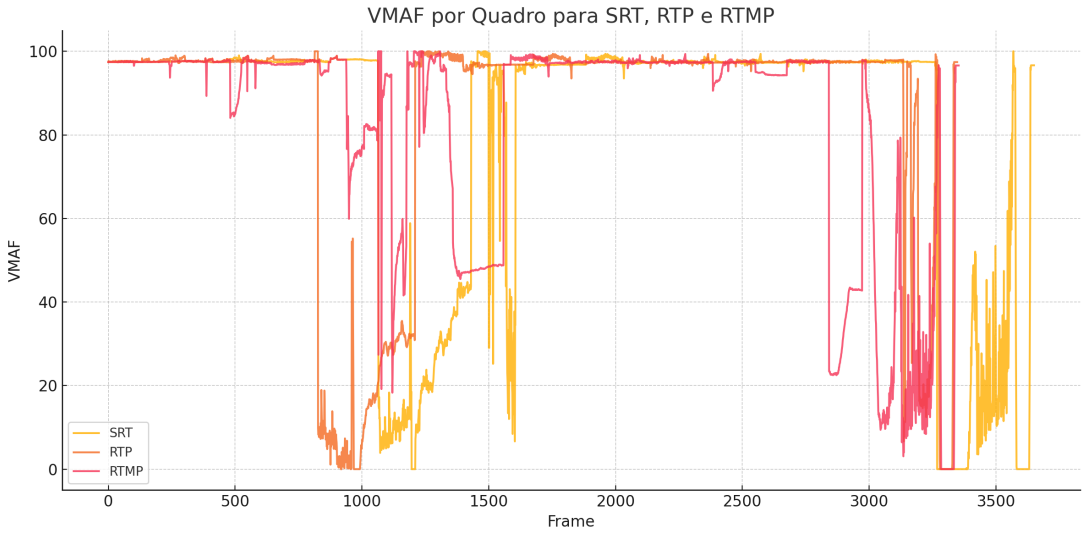
As degradações observadas neste cenário podem ser explicadas pelas diferenças entre os protocolos. O RTP, por não dispor de mecanismos de confiabilidade, apresentou elevada taxa de perdas (52%), o que resultou em quedas de qualidade visual e baixo VMAF, apesar da alta vazão registrada (6,127 Mbps). Já o RTMP, por ser baseado em TCP, garantiu a entrega dos pacotes, mas ao custo de elevados atrasos de retransmissão, refletidos em *jitter* acentuado (109,48 ms) e elevado número de eventos de *rebuffering*, comprometendo a fluidez da reprodução. O SRT, por sua vez, operou no modo *live streaming* padrão, configurado para manter o atraso alvo e aplicar mecanismos de retransmissão seletiva (ARQ) e controle de congestionamento sempre que necessário, priorizando a consistência visual em detrimento da taxa bruta de transmissão. Esse comportamento explica a menor vazão média registrada (1,320 Mbps) em comparação ao RTP, mas também a maior estabilidade perceptual (VMAF de 93,98 e SSIM de 0,9828) e a baixa taxa de perdas (1,08%), mesmo sob mobilidade e tráfego intenso. Dessa forma, o SRT obteve maior estabilidade perceptual e qualidade visual consistente em condições adversas.

4.2.3.2 Cenário doisMoveis - Compressão de vídeo H.264 e bitrate 2Mbps

No cenário representado pela Figura 28, que utiliza compressão de vídeo em H.264 com *bitrate* fixo de 2Mbps, observou-se uma redução geral na qualidade de transmissão em relação ao cenário com *bitrate* variável, embora com menor disparidade entre os protocolos. Conforme indicado na TabelaTabela 31, o protocolo RTP apresentou o maior valor de VMAF (84,82), acompanhado de SSIM de 0,9346 e PSNR de 38,93dB. O protocolo SRT, apesar de atingir o maior valor de PSNR (44,48dB), obteve VMAF de 80,72, inferior ao do RTP, o que sugere diferenças na percepção visual da qualidade. O RTMP apresentou desempenho intermediário em termos de VMAF (82,63), porém com maior taxa de perda (12,67%) e *jitter* médio (13,01ms), conforme os dados da TabelaTabela 32.

A análise da curva de VMAF revela que o RTP manteve valores elevados com certa estabilidade, apesar de quedas pontuais; o SRT apresentou comportamento mais uniforme, porém com qualidade perceptual ligeiramente inferior; e o RTMP demonstrou maior instabilidade, com variações mais acentuadas ao longo dos quadros. Em relação aos eventos de *rebuffering*, os resultados evidenciam diferenças relevantes: o RTP, embora tenha alcançado maior qualidade média, foi o que apresentou o maior número de interrupções (158), seguido por RTMP (99) e SRT (69). Esses dados indicam que o SRT proporciona o melhor equilíbrio entre qualidade visual e continuidade da reprodução.

Figura 28 – Comparação de VMAF no cenário doisMoveis - Compressão H.264 e *bitrate* 2 Mbps



Fonte: Elaborada pelo autor.

Tabela 31 – Métricas de qualidade de vídeo para o cenário doisMoveis – H.264 a 2 Mbps

Protocolo	**PSNR (dB)**	**SSIM**	**VMAF**
SRT	44,48	0,935	80,72
RTP	38,93	0,9346	84,82
RTMP	40,24	0,929	82,63

Fonte: Elaborada pelo autor.

Tabela 32 – Métricas de desempenho de rede cenário *doisMoveis* – H.264 a 2 Mbps

Protocolo	**Vazão Média (Mbps)**	**Perda**	**Jitter (ms)**	**Rebuffering**
SRT	3,624	4,23%	10,15	69
RTP	3,702	3,90%	14,86	158
RTMP	5,095	12,67%	13,01	99

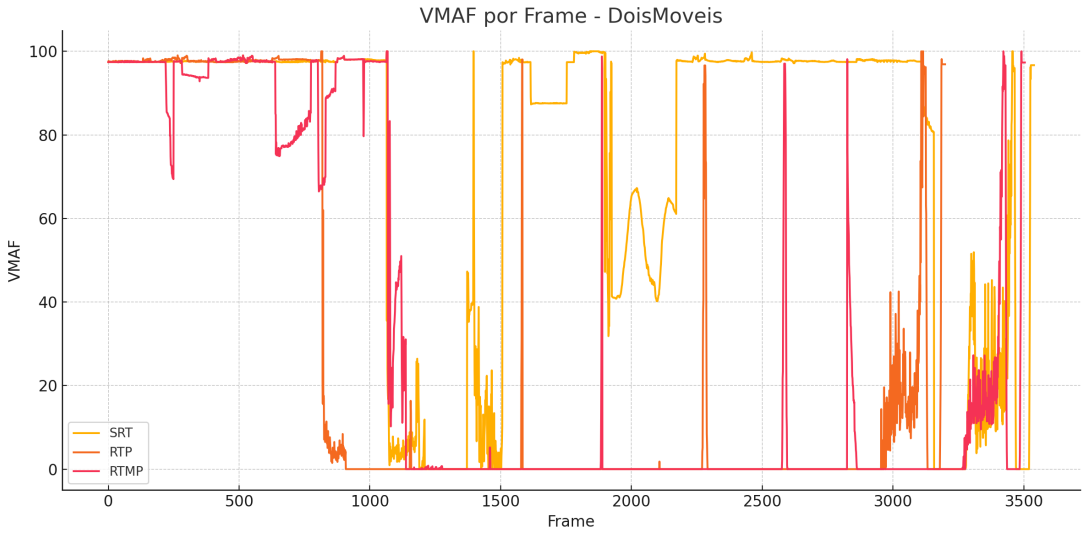
Fonte: Elaborada pelo autor.

4.2.3.3 Cenário doisMoveis - Compressão de vídeo H.264 e bitrate 4Mbps

A análise da Figura 29, em conjunto com os dados daTabela 33 e Tabela 34, evidencia os impactos da mobilidade combinada a uma taxa de *bitrate* fixa de 4Mbps sobre o desempenho dos protocolos avaliados. O protocolo SRT demonstrou o melhor equilíbrio entre qualidade visual e estabilidade de rede, com VMAF médio de 74,08, PSNR de 43,17dB e SSIM de 0,9103, mesmo sob perdas moderadas (8,77%) e 60 eventos de *rebuffering*. O protocolo RTP, por outro lado, apresentou o pior desempenho, com perdas elevadas (68,87%) e o menor valor de VMAF (27,55), indicando forte degradação perceptual.

O protocolo RTMP situou-se em uma posição intermediária, alcançando VMAF de 31,57 e perdas de 42,78%, porém com apenas 36 eventos de *rebuffering*. Esse comportamento sugere maior fluidez na reprodução, apesar da qualidade visual mais limitada. O gráfico da Figura 29 reforça essas observações, evidenciando uma curva de qualidade mais estável para o SRT, enquanto os demais protocolos, especialmente o RTP, apresentam quedas abruptas e instabilidade ao longo da transmissão.

Figura 29 – Comparação de VMAF no cenário *doisMoveis*. Compressão H.264 e bitrate 4Mbps



Fonte: Elaborada pelo autor.

Tabela 33 – Métricas de qualidade de vídeo para o cenário *doisMoveis* – H.264 a 4 Mbps

Protocolo	**PSNR (dB)**	**SSIM**	**VMAF**
SRT	43,17	0,9103	74,08
RTP	23,64	0,72	27,55
RTMP	26,49	0,8042	31,57

Fonte: Elaborada pelo autor.

Tabela 34 – Métricas de desempenho de rede cenário *doisMoveis* – H.264 a 4 Mbps

Protocolo	**Vazão Média (Mbps)**	**Perda**	**Jitter (ms)**	**Rebuffering**
SRT	5,652	8,77%	9,63	60
RTP	4,537	68,87%	14,6	191
RTMP	6,774	42,78%	11,39	36

Fonte: Elaborada pelo autor.

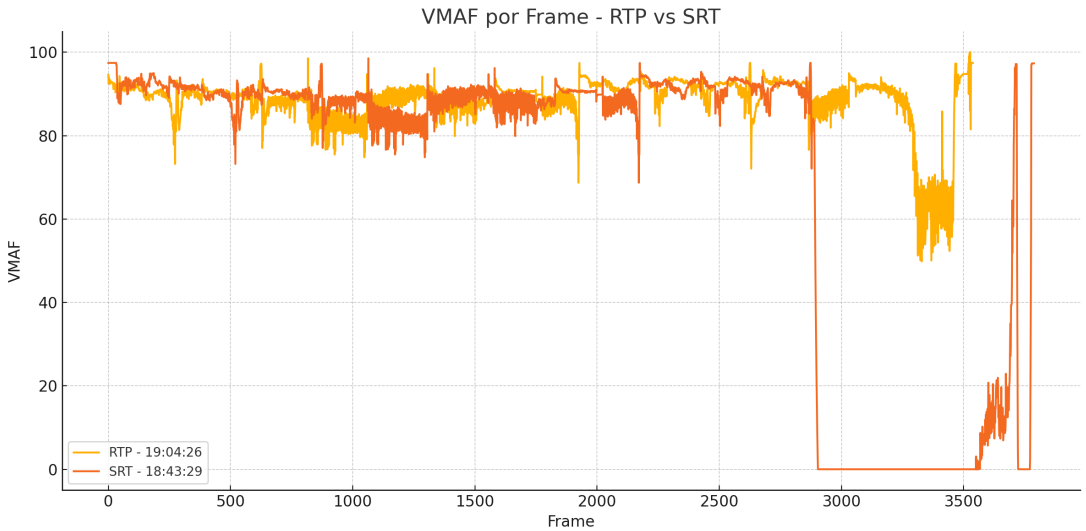
4.2.3.4 *Cenário doisMoveis - Compressão de vídeo H.265 e bitrate variável*

No cenário apresentado na Figura 30, com compressão de vídeo H.265 e *bitrate* variável, observam-se diferenças marcantes entre os protocolos RTP e SRT. Conforme os dados da Tabela 35, o protocolo RTP apresentou desempenho superior em termos de qualidade visual, com VMAF de 88,13, SSIM de 0,9619 e PSNR de 36,78dB. Por outro lado, o protocolo SRT obteve VMAF consideravelmente inferior (69,61), apesar de valores semelhantes de PSNR (36,19dB) e SSIM (0,9424), o que sugere que a compressão foi eficaz, mas a qualidade percebida foi impactada negativamente por fatores relacionados à rede.

Esses fatores são destacados na Tabela 36, onde o SRT apresentou a maior taxa de perda (16,44%) e *jitter* moderado (8,03ms), aspectos que justificam a degradação na qualidade visual percebida. Em contraste, o protocolo RTP apresentou menor perda de pacotes (6,65%), porém com *jitter* mais elevado (17,48ms), o que contribuiu para o aumento no número de eventos de *rebuffering* (28 eventos, frente a apenas 3 observados no SRT).

A Figura 30 reforça essas observações: o protocolo RTP manteve valores de VMAF elevados, com pequenas flutuações ao longo da transmissão, enquanto o SRT apresentou oscilações mais acentuadas e valores consistentemente mais baixos, indicando maior sensibilidade às variações da rede.

Figura 30 – Comparação de VMAF no cenário *doisMoveis*. Compressão H.265 e bitrate variável



Fonte: Elaborada pelo autor.

Tabela 35 – Métricas de qualidade de vídeo para o cenário *doisMoveis*. Compressão H.265 e bitrate variável

Protocolo	**PSNR (dB)**	**SSIM**	**VMAF**
SRT	36,19	0,9424	69,61
RTP	36,78	0,9619	88,13

Fonte: Elaborada pelo autor.

Tabela 36 – Métricas de desempenho de rede cenário *doisMoveis*. Compressão H.265 e bitrate variável

Protocolo	**Vazão Média (Mbps)**	**Perda**	**Jitter (ms)**	**Rebuffering**
SRT	1,160	16,44%	8,03	3
RTP	1,101	6,65%	17,48	28

Fonte: Elaborada pelo autor.

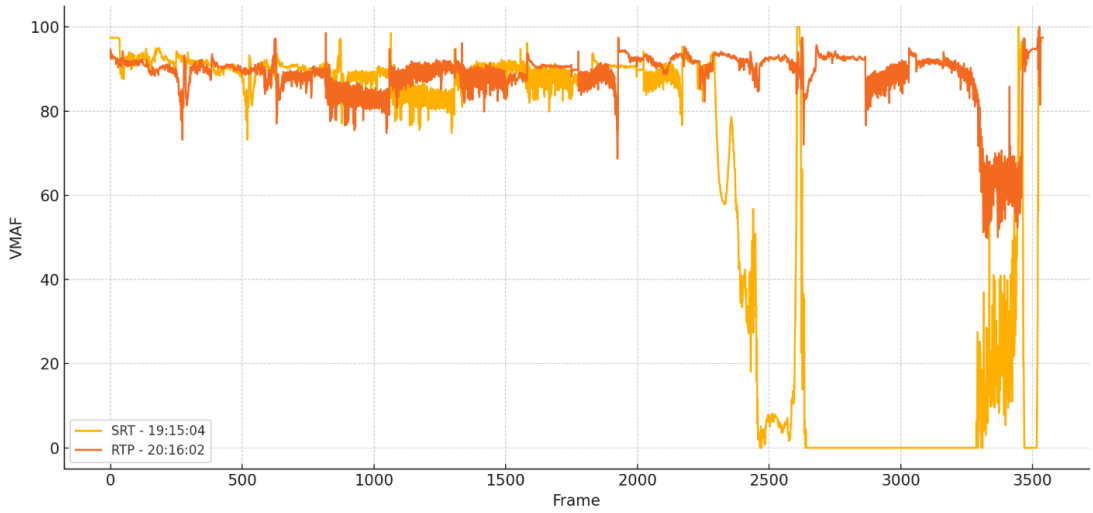
4.2.3.5 Cenário doisMoveis - Compressão de vídeo H.265 e bitrate 2Mbps

No cenário representado na Figura 31, com compressão H.265 e *bitrate* de 2 Mbps, o protocolo RTP apresentou os melhores indicadores de qualidade perceptual, alcançando VMAF de 88,13, SSIM de 0,9616 e PSNR de 36,78dB. Em contrapartida, o protocolo SRT obteve resultados significativamente inferiores, com VMAF de 62,82, SSIM de 0,8876 e PSNR de 31,06dB.

Essas diferenças podem ser atribuídas aos parâmetros de desempenho de rede apresentados na Tabela 38. O protocolo SRT apresentou uma taxa de perdas mais elevada (11,05%) e *jitter* moderado (8,22ms), o que impactou negativamente a qualidade visual, embora tenha registrado apenas 3 eventos de *rebuffering*. Já o protocolo RTP, apesar de apresentar melhor desempenho em termos de qualidade de vídeo, sofreu com *jitter* mais elevado (17,68ms) e um número maior de eventos de *rebuffering* (28), evidenciando menor estabilidade na continuidade da reprodução.

O gráfico de VMAF corrobora essas observações: o protocolo RTP manteve valores elevados e estáveis ao longo da transmissão, enquanto o SRT apresentou flutuações mais acentuadas e valores consistentemente mais baixos. Conclui-se, portanto, que o RTP foi superior na entrega de qualidade visual, ao passo que o SRT demonstrou maior resiliência às interrupções de reprodução.

Figura 31 – Comparação de VMAF no cenário *doisMoveis*. Compressão H.265 e bitrate 2Mbps



Fonte: Elaborada pelo autor.

Tabela 37 – Métricas de qualidade de vídeo para o cenário *doisMoveis*. Compressão H.265 e bitrate 2Mbps

Protocolo	**PSNR (dB)**	**SSIM**	**VMAF**
SRT	31,06	0,8876	62,82
RTP	36,78	0,9616	88,13

Fonte: Elaborada pelo autor.

Tabela 38 – Métricas de desempenho de rede no cenário *doisMoveis*. Compressão H.265 e bitrate 2Mbps

Protocolo	**Vazão Média (Mbps)**	**Perda**	**Jitter (ms)**	**Rebuffering**
SRT	1,247	11,05%	8,22	3
RTP	1,101	6,65%	17,68	28

Fonte: Elaborada pelo autor.

4.2.3.6 Cenário doisMoveis - Compressão de vídeo H.265 e bitrate 4Mbps

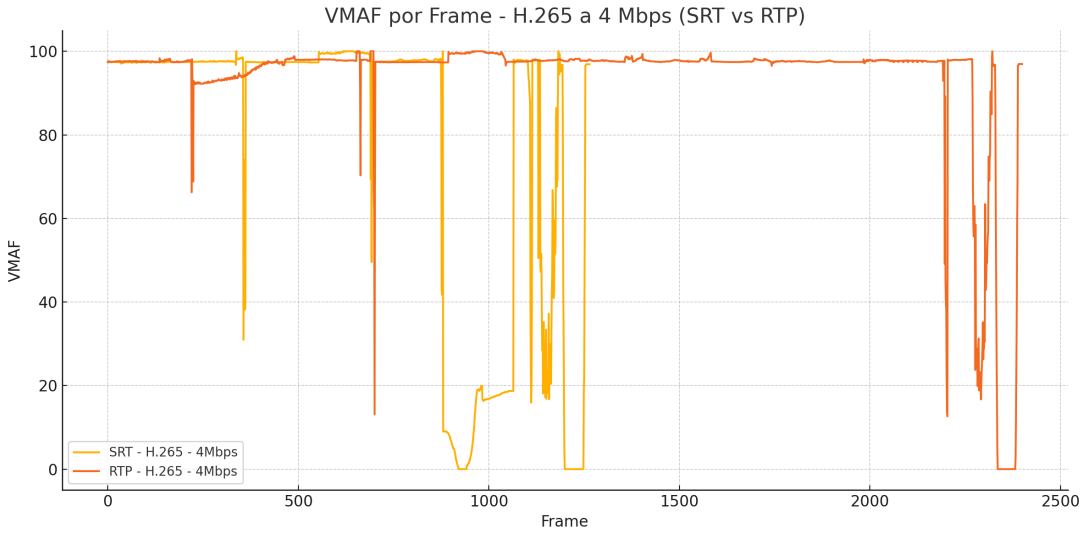
Neste último cenário, ilustrado pela Figura 32, foi considerada a compressão de vídeo em H.265 com *bitrate* fixo de 4 Mbps. O protocolo RTP manteve o desempenho observado nos cenários anteriores. Conforme apresentado na Tabela 39, o RTP atingiu um VMAF de 93,86, com SSIM de 0,8556 e PSNR de 33,19dB, enquanto o SRT obteve valores visivelmente inferiores: VMAF de 78,07, SSIM de 0,7854 e PSNR de 28,68dB.

Essa diferença de desempenho está relacionada às métricas de rede descritas na Tabela 40. Embora ambos os protocolos tenham mantido vazão média semelhante (em torno de 3,75 Mbps), o SRT apresentou uma taxa de perda mais elevada (16,86%) e

jitter significativamente maior (23,2ms), resultando em 71 eventos de *rebuffering*. Por outro lado, o RTP registrou uma menor taxa de perda (13,34%), *jitter* reduzido (2,9ms) e apenas 5 eventos de *rebuffering*, favorecendo uma reprodução mais fluida e com qualidade visual superior.

Essas características ficam evidentes no gráfico de VMAF, no qual o protocolo RTP mantém valores elevados e estáveis ao longo do tempo, enquanto o SRT apresenta flutuações acentuadas, refletindo uma degradação na qualidade percebida da transmissão.

Figura 32 – Comparação de VMAF no cenário *umMovel* e tráfego de fundo. Compressão H.265 e bitrate 4Mbps



Fonte: Elaborada pelo autor.

Tabela 39 – Métricas de qualidade de vídeo para o cenário *umMovel* e tráfego de fundo – H.265 a 4 Mbps

Protocolo	**PSNR (dB)**	**SSIM**	**VMAF**
SRT	28,68	0,7854	78,07
RTP - 4M	33,19	0,8556	93,86

Fonte: Elaborada pelo autor.

Tabela 40 – Métricas de desempenho de rede cenário *umMovel* e tráfego de fundo – H.265 a 4 Mbps

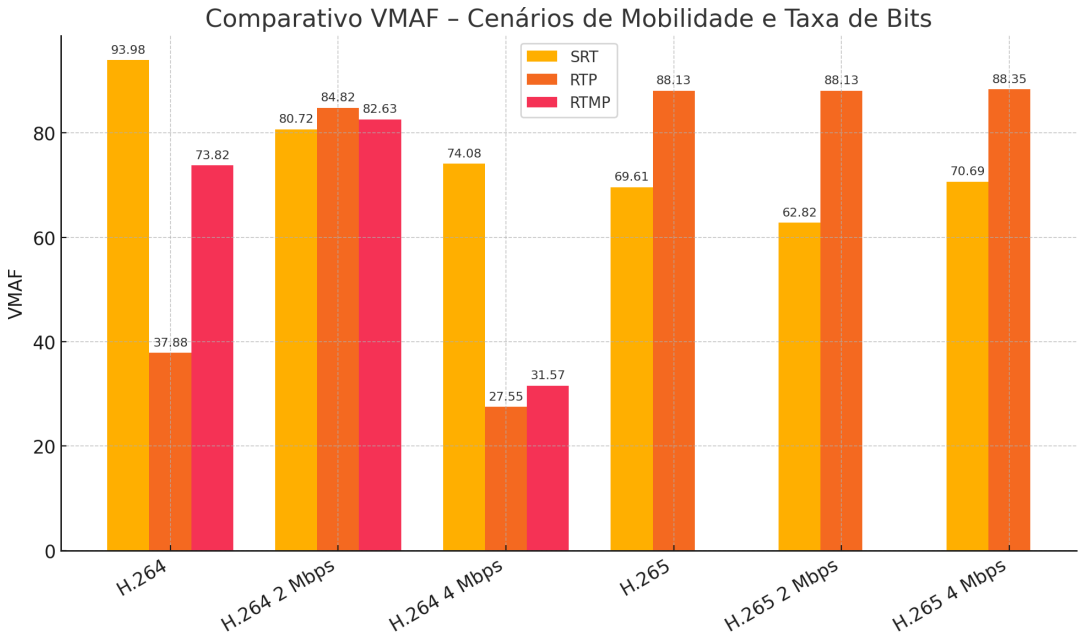
Protocolo	**Vazão Média (Mbps)**	**Perda**	**Jitter (ms)**	**Rebuffering**
SRT	4,235	16,86%	23,2	71
RTP	7,308	13,34%	2,9	5

Fonte: Elaborada pelo autor.

4.2.4 Comparação de VMAF no Cenário umMovel e tráfego de fundo

A Figura 33 apresenta os resultados dos valores de VMAF obtidos nos experimentos realizados no cenário *DoisMoveis*, o qual inclui tráfego de fundo na rede. O protocolo SRT manteve desempenho elevado, atingindo valor máximo de 93,98 com compressão H.264, mas apresentou queda significativa com H.265 a 2 Mbps (62,82), evidenciando maior sensibilidade à taxa de compressão. Por outro lado, o protocolo RTP demonstrou forte oscilação de desempenho: obteve baixa qualidade com H.264 a 4 Mbps (27,55), mas alcançou excelente resultado com H.265 a *bitrate* variável (88,13), mesmo com taxa reduzida. Já o protocolo RTMP apresentou desempenho inferior em todas as configurações, destacando-se como o menos indicado para cenários com mobilidade e congestionamento.

Figura 33 – Comparação de VMAF no cenário *umMovel* e tráfego de fundo



Fonte: Elaborada pelo autor.

5 CONCLUSÕES

A partir dos experimentos apresentados no Capítulo 4, foi possível avaliar o desempenho dos protocolos RTMP e SRT em diferentes condições de transmissão de vídeo, abrangendo cenários com rede cabeada, mobilidade e tráfego de fundo concorrente. Esta conclusão resume os principais resultados obtidos e os relaciona com achados da literatura técnica, em especial os estudos conduzidos pela *Haivision*, que comparam o desempenho dos protocolos em redes de longa distância e detalham os mecanismos internos de resiliência e correção de erros do SRT (HAIVISION, 2019).

5.1 ANÁLISE DOS RESULTADOS

O protocolo SRT demonstrou ser o mais robusto ao longo de todos os cenários, destacando-se pela estabilidade na qualidade do vídeo — evidenciada por curvas de VMAF mais suaves —, além de apresentar menor taxa de perdas e baixos níveis de *rebuffering*, especialmente em condições adversas com mobilidade ou tráfego de fundo. Essa resiliência pode ser atribuída aos seus mecanismos internos de correção de erros, retransmissão seletiva e controle de congestionamento, que o tornam particularmente adequado para redes instáveis, com variações de latência, perda ou *jitter*.

O protocolo RTP, embora não tenha sido o foco principal deste trabalho, foi utilizado como base comparativa nos testes e apresentou os maiores valores de VMAF em diversos cenários, sobretudo sob condições ideais e sem congestionamento. No entanto, sua ausência de mecanismos de controle e recuperação de pacotes o torna extremamente sensível a perdas e flutuações na rede, resultando em oscilações acentuadas na qualidade visual e aumento no número de eventos de *rebuffering*. Isso foi especialmente evidente nos testes com mobilidade e tráfego de fundo, como no cenário descrito na subseção 4.2.3.6, onde o RTP sofreu degradações significativas.

Já o protocolo RTMP apresentou um desempenho intermediário. Em condições estáveis, conseguiu manter uma boa qualidade perceptual, com VMAF na faixa de 70 a 80. No entanto, seu desempenho deteriorou-se rapidamente em cenários com mobilidade ou tráfego intenso, refletido por altos níveis de *jitter*, perdas expressivas e elevado número de eventos de *rebuffering*. Outro fator limitante é sua compatibilidade: o RTMP não oferece suporte nativo ao codec H.265, restringindo sua aplicação a transmissões com compressão H.264, o que pode impactar negativamente sua adoção em cenários que demandam maior eficiência de compressão.

5.1.1 Impacto da compressão de vídeo e taxa de transferência

A escolha do *codec* e da taxa de *bitrate* teve impacto direto nos resultados observados. A compressão H.265, embora mais eficiente em termos de qualidade por bit transmitido, demonstrou maior sensibilidade a perdas e *jitter*, especialmente em cenários com mobilidade. Essa característica é evidenciada, por exemplo, no comportamento do protocolo SRT na subseção 4.2.3.5, em que mesmo taxas de perda relativamente baixas comprometeram significativamente os valores de VMAF, indicando que a eficiência da compressão não necessariamente garante uma qualidade perceptual elevada em condições adversas de rede.

Por outro lado, o *codec* H.264 demonstrou maior tolerância às variações da rede, especialmente em taxas de *bitrate* mais elevadas (4 Mbps), favorecendo protocolos como o SRT em termos de estabilidade na reprodução. A adoção de *bitrate* variável nos testes com H.264 proporcionou melhores resultados para o SRT, permitindo ajustes dinâmicos frente às oscilações de rede. Ainda assim, mesmo sob *bitrate* constante, o SRT demonstrou capacidade de adaptação às condições adversas, ao passo que o RTP apresentou oscilações mais acentuadas na qualidade do vídeo, evidenciando sua menor resiliência frente a perdas e *jitter*.

5.2 EFEITOS DA MOBILIDADE E DO TRÁFEGO DE FUNDO

A mobilidade configurou-se como uma das variáveis mais críticas para os protocolos avaliados, ao representar cenários realistas em que usuários se deslocam com dispositivos móveis em redes congestionadas. Nessas condições adversas, o protocolo SRT demonstrou desempenho superior, preservando a qualidade da transmissão mesmo diante de perdas e variações de latência. Tal resiliência foi evidenciada nos gráficos de VMAF e nas métricas de *rebuffering*, que, em diversos casos, permaneceram abaixo de cinco eventos, indicando maior continuidade e fluidez na reprodução do vídeo.

O protocolo RTP, que havia se destacado em cenários com topologia fixa, teve seu desempenho significativamente comprometido diante da mobilidade, registrando perdas superiores a 60% e valores de VMAF inferiores a 30 em determinados testes. Por sua vez, o RTMP manteve um padrão intermediário de desempenho, mas apresentou tendência à degradação da qualidade visual e aumento no número de eventos de *rebuffering* à medida que as condições do canal se deterioravam. A inserção de tráfego de fundo intensificou esses efeitos, impactando especialmente o RTP, que não possui mecanismos de controle de congestionamento. Em contrapartida, o SRT demonstrou maior tolerância a esse tipo de interferência, mantendo níveis satisfatórios de qualidade mesmo sob competição por largura de banda.

5.3 ANÁLISE COMPARATIVA

A Tabela 41 apresenta um resumo qualitativo do comportamento dos três protocolos avaliados, considerando os principais critérios de desempenho observados nos experimentos. São eles: qualidade de vídeo (VMAF), estabilidade da transmissão (avaliada por meio de *jitter* e perdas), fluidez da reprodução (número de eventos de *rebuffering*) e capacidade de adaptação a cenários com mobilidade e instabilidade de rede. Essa síntese permite visualizar, de forma comparativa, os pontos fortes e as limitações de cada protocolo frente às diferentes condições de rede simuladas neste trabalho.

Tabela 41 – Resumo comparativo qualitativo entre os protocolos

Protocolo	**Qualidade de Vídeo**	**Estabilidade**	**Rebuffering**	**Mobilidade**
SRT	Alta	Alta	Baixo a moderado	Alta
RTP	Muito alta	Baixa	Elevado	Baixa
RTMP	Moderada	Moderada	Moderado a elevado	Baixa

Fonte: Elaborada pelo autor.

A elaboração da Tabela 41 ocorreu a partir de uma análise qualitativa dos resultados mais recorrentes ao longo dos experimentos, considerando o comportamento típico de cada protocolo diante das diferentes condições de rede. Não se trata de médias aritméticas, mas de uma síntese dos padrões observados: o SRT manteve qualidade de vídeo elevada e estável mesmo em cenários adversos, com menor impacto das perdas e melhor adaptação à mobilidade; o RTP apresentou os maiores picos de qualidade perceptual em condições ideais, mas se mostrou extremamente sensível a variações de rede, resultando em instabilidade e altos níveis de perda em cenários degradados; já o RTMP permaneceu em posição intermediária, com desempenho moderado em termos de qualidade e estabilidade, mas mais suscetível a variações de *jitter* e ao aumento de eventos de *rebuffering*. Dessa forma, a classificação apresentada resume os pontos fortes e limitações de cada protocolo de acordo com os aspectos mais frequentes observados nos testes.

Os resultados obtidos neste trabalho estão em consonância com estudos independentes da Haivision, os quais destacam três aspectos fundamentais: (i) o protocolo SRT apresenta latência significativamente menor e capacidade de manter taxas de *bitrate* mais elevadas em redes públicas de longa distância; (ii) o RTMP requer buffers maiores, o que acarreta aumento considerável na latência em situações com crescimento do RTT; e (iii) fluxos em tempo real sobre redes imprevisíveis demandam mecanismos específicos de compensação para assegurar estabilidade e qualidade da transmissão (HAIVISION SYSTEMS INC., 2018).

5.4 CONCLUSÃO GERAL

Com base nas evidências experimentais, conclui-se que o protocolo SRT oferece o melhor equilíbrio entre qualidade de vídeo, estabilidade de rede e continuidade da reprodução em cenários realistas, que envolvem mobilidade e congestionamento. Embora não tenha obtido os maiores valores de VMAF em todas as situações, sua capacidade de lidar com perdas e *jitter* o torna a opção mais confiável para transmissões ao vivo em redes instáveis. O protocolo RTP, apesar de não ser o foco principal deste estudo, mostrou bom desempenho apenas em redes com alta qualidade e baixa variabilidade, sendo menos indicado para ambientes adversos. Já o RTMP demonstrou desempenho intermediário e pode ser considerado em cenários menos sensíveis a oscilações de qualidade.

De forma mais ampla, os resultados reforçam a importância de compreender a relação entre métricas de QoS e QoE. Este estudo analisou como cada protocolo converte as condições de rede (perda, atraso, vazão e *jitter*) em diferentes níveis de experiência percebida pelo usuário. Observou-se que, enquanto o SRT consegue transformar um QoS degradado em uma QoE aceitável graças a mecanismos de correção e adaptação, o RTP é altamente sensível a perdas e instabilidades, e o RTMP depende de *buffers* maiores, o que compromete a fluidez da reprodução. Como perspectiva para trabalhos futuros, sugere-se aprofundar essa análise por meio da modelagem sistemática da relação QoS e QoE, permitindo caracterizar cada protocolo não apenas com base em métricas objetivas de rede, mas também na forma como essas métricas se traduzem, de maneira quantificável, na percepção final do usuário.

5.5 AVALIAÇÃO DO ATENDIMENTO AOS OBJETIVOS

Os objetivos estabelecidos para este trabalho foram plenamente atendidos com base nos resultados obtidos e analisados neste capítulo. O primeiro objetivo — desenvolver um cenário híbrido que combinasse aplicações reais com rede simulada — foi alcançado por meio da integração do OMNeT++ em modo de emulação, utilizando tráfego real gerado pelo *FFmpeg* por meio de interfaces TAP. Além disso, foram incorporados modelos de mobilidade com *handover* na rede sem fio simulada, possibilitando análises sob condições realistas de deslocamento.

O segundo objetivo — identificar os principais fatores que impactam o desempenho dos protocolos em contextos de mobilidade — foi contemplado por meio da análise de métricas como perda de pacotes, *jitter*, eventos de *rebuffering* e VMAF. Esses fatores mostraram-se fundamentais na degradação da qualidade percebida pelo usuário final, especialmente em cenários com dispositivos móveis e presença de tráfego concorrente.

O terceiro objetivo — comparar a latência e a qualidade de vídeo entre os protocolos em diferentes condições — foi obtido através de métricas objetivas, incluindo VMAF,

PSNR, SSIM e número de eventos de *rebuffering*, permitindo evidenciar as vantagens e limitações de cada protocolo em diversos cenários.

Por fim, o último objetivo — avaliar a viabilidade de uso dos protocolos em aplicações como *streaming* ao vivo e vigilância — foi cumprido por meio da comparação entre estabilidade, fluidez e capacidade de recuperação frente a falhas. O protocolo SRT foi apontado como o mais indicado para aplicações críticas com mobilidade e variações de rede, enquanto o RTP e o RTMP apresentaram limitações em ambientes com instabilidade.

6 TRABALHOS FUTUROS

Com base nas observações e limitações dos experimentos realizados neste trabalho, foram identificadas diversas oportunidades de aprofundamento e melhorias para pesquisas futuras. Estas propostas visam aumentar a confiabilidade dos testes para os protocolos avaliados.

6.1 PADRONIZAÇÃO NOS TESTES DE MOBILIDADE

Uma das principais recomendações para trabalhos futuros consiste na padronização do movimento do nó móvel em todos os experimentos com mobilidade. Durante os testes realizados, pequenas variações nas trajetórias ou nos tempos de deslocamento podem ter influenciado os resultados, especialmente em cenários com eventos de *handover*. Ao garantir que todos os protocolos sejam avaliados a partir do mesmo ponto de início e com a mesma trajetória, torna-se possível realizar comparações mais justas e precisas entre os desempenhos observados.

Para isso, recomenda-se a implementação de um *script* de controle de mobilidade centralizado ou o uso de arquivos de rastreamento (*mobility trace files*) pré-definidos no *framework* INET do OMNeT++, assegurando que todos os experimentos utilizem exatamente o mesmo padrão de movimentação.

Adicionalmente, para reduzir o impacto da variabilidade inerente às medições, recomenda-se aumentar o número de repetições de cada experimento. Um maior conjunto de execuções possibilita a aplicação de métodos estatísticos mais robustos, aumentando o nível de confiança nos resultados obtidos e permitindo identificar com maior precisão eventuais diferenças de desempenho entre os protocolos.

6.2 ANÁLISE CRUZADA COM LOGS DO OMNET++

Outra extensão deste trabalho envolve a relação entre os resultados de desempenho e os logs gerados pelo OMNeT++, especialmente no que se refere aos eventos de *handover*. A análise cruzada entre os picos de perda, variação de *jitter* e eventos de *rebuffering* com os momentos em que ocorrem as transições entre roteadores pode esclarecer se os impactos observados estão diretamente ligados ao mecanismo de mobilidade.

Essa análise pode ser realizada por meio da extração de eventos de *handover* a partir dos logs de mobilidade dos módulos móveis do INET, combinada com o *timestamp* das métricas extraídas de ferramentas como *FFmpeg*, *tcpdump* e VMAF. O objetivo é compreender em detalhes como os protocolos lidam com as alterações introduzidas pelo

movimento em rede sem fio.

6.3 DIAGNÓSTICO DO TRAVAMENTO DO RTMP APÓS EXECUÇÕES REPETIDAS

Durante os testes, foi identificado um comportamento anômalo do protocolo RTMP, que, em determinados momentos, apresentava travamentos ou falhas de inicialização após execuções consecutivas de testes. Essa falha, que não se repetiu com os demais protocolos, pode ter origem em diversos fatores, como cache, falhas de *buffer* interno ou sobrecarga do servidor RTMP (nginx com módulo RTMP).

Como proposta de trabalho futuro, sugere-se a realização de uma investigação das causas do travamento do RTMP, incluindo:

- Monitoramento detalhado dos processos do FFmpeg e do nginx durante os testes;
- Limpeza automatizada de cache, arquivos temporários e buffers entre execuções;
- Análise dos logs internos do nginx e do sistema operacional;
- Avaliação do comportamento em diferentes versões do servidor RTMP;
- Testes com introdução de *delays* entre execuções para verificar o impacto da sobreposição de recursos.

Essas ações podem identificar gargalos não observados anteriormente e permitir ajustes no ambiente experimental, garantindo maior confiabilidade dos resultados com o protocolo RTMP.

6.4 EXPANSÃO PARA NOVOS PROTOCOLOS E REDES HETEROGÊNEAS

Além das sugestões relacionadas aos experimentos já realizados, trabalhos futuros podem explorar a inclusão de novos protocolos orientados a vídeo, como o *srt-live-transmit*, RIST ou *WebRTC* (BERGKVIST et al., 2012). Outra possibilidade é a expansão dos testes para redes heterogêneas, que combinem enlaces cabeados, Wi-Fi e 4G/5G, de modo a simular ambientes reais mais complexos, especialmente voltados para aplicações móveis e de IoT.

Nesse contexto, destaca-se em particular a análise em redes móveis de quinta geração (5G). Esse tipo de rede apresenta desafios adicionais, como maior variabilidade de latência, mecanismos de mobilidade mais complexos (*handover* entre células) e políticas de priorização de tráfego diferenciadas. A comparação entre os resultados obtidos em WLAN e aqueles possíveis em redes 5G permitiria uma avaliação mais abrangente do desempenho de protocolos de *streaming* em cenários heterogêneos de mobilidade.

Adicionalmente, recomenda-se considerar protocolos de transporte alternativos, como o **SCTP**, que oferece recursos de multistreaming e multihoming; o **QUIC**, que combina confiabilidade com baixa latência sobre UDP; e o próprio **WebRTC**, amplamente utilizado em aplicações interativas. A inclusão desses protocolos permite ampliar a análise comparativa e aproximar ainda mais os experimentos das soluções emergentes para comunicação multimídia em tempo real.

6.5 AVALIAÇÃO DA QUALIDADE DE ÁUDIO

Outra limitação observada neste trabalho foi a ausência de métricas específicas para a análise da qualidade do áudio transmitido. Embora os testes tenham incluído fluxos com áudio codificado (por exemplo, com o codec AAC), a avaliação concentrou-se exclusivamente na qualidade visual da transmissão, por meio de métricas como PSNR, SSIM e VMAF.

6.6 CONCLUSÃO

As propostas aqui apresentadas visam não apenas refinar a metodologia experimental adotada neste trabalho, mas também expandir o escopo de análise, aproximando ainda mais os experimentos das condições reais de operação. A continuidade dessas investigações contribuirá significativamente para a compreensão das características dos protocolos de *streaming* em tempo real e para a escolha adequada de tecnologias conforme os requisitos específicos de cada aplicação.

REFERÊNCIAS

ABOUT FFmpeg. FFmpeg official website. Acessado em 24 de fevereiro de 2025. Disponível em: <https://www.ffmpeg.org/about.html>.

ADDU, Raj Kiran; POTUVARDANAM, Vinod Kumar. **Effect of Codec Performance on Video QoE**. Ago. 2014. Diss. (Mestrado) – Blekinge Institute of Technology, Karlskrona, Sweden.

ALI, Muhammad et al. RES: Real-time Video Stream Analytics using Edge Enhanced Clouds. **IEEE Transactions on Cloud Computing**, PP, p. 1–1, jan. 2020. DOI: 10.1109/TCC.2020.2991748.

ALMEIDA, João. **Simulação de Redes com OMNeT++**. https://exemplo.com. Sem data.

ALOMAN, A. et al. Performance Evaluation of Video Streaming Using MPEG DASH, RTSP, and RTMP in Mobile Networks. In: 2015 8th IFIP Wireless and Mobile Networking Conference (WMNC). 2015. P. 144–151. DOI: 10.1109/WMNC.2015.12.

BERGKVIST, Adam et al. **Web Real-Time Communication (WebRTC): Media Capture and Streams**. 2012. W3C Candidate Recommendation. Acesso em: 04 ago. 2025. Disponível em: <https://www.w3.org/TR/webrtc/>.

COMER, Douglas E. **Interligação de Redes com TCP/IP: Princípios, Protocolos e Arquitetura**. Tradução: Tássia Fernanda Alvarenga. 6. ed. Rio de Janeiro, RJ, Brasil: Elsevier Editora Ltda., 2015. ISBN 978-85-352-7863-7. Disponível em: <https://www.elsevier.com/books/interligacao-de-redes-com-tcp-ip/comer/978-85-352-7863-7>.

FOROUZAN, Behrouz A. **Comunicação de dados e redes de computadores**. Tradução: Ariovaldo Griesi. 4. ed. Porto Alegre: AMGH Editora, 2010. Tradução de: Data Communications and Networking, 4th Edition. ISBN 978-85-63308-47-4.

FORTINET. **QoS (Qualidade de Serviço)**. 2023. https://www.fortinet.com/br/resources/cyberglossary/qos-quality-of-service. Acessado em: [19 de Fevereiro de 2025].

GONÇALVES, João Lucas; CAVALCANTI, Daniel; FIGUEIREDO, Felipe. Avaliação de métricas objetivas de qualidade de vídeo com FFmpeg. **Revista de Informática Teórica e Aplicada**, v. 27, n. 1, p. 35–47, 2020.

HAIVISION. **SRT Open Source Streaming for Low Latency Video**. 2023. Disponível em: https://www.haivision.com. Disponível em: <https://www.haivision.com>.

_____. **SRT: Secure Reliable Transport**. 2019. SRT Alliance. Disponível em: https://www.srtalliance.org/.

HAIVISION SYSTEMS INC. **Packet Loss Recovery in SRT**. 2024. SRT Documentation, Version 1.5.3. Disponível em: <https://doc.haivision.com/SRT/1.5.3/Haivision/packet-loss-rate>. Acesso em: 11 ago. 2025.

_____. **Secure Reliable Transport (SRT) Protocol Technical Overview**. 2018. Last Updated: 17 October 2018. Disponível em: <https://github.com/Haivision/srt>.

HORÉ, Alain; ZIOU, Derek. Image Quality Metrics: PSNR vs. SSIM. In: IEEE. 2010 20th International Conference on Pattern Recognition. 2010. P. 2366–2369. DOI: 10.1109/ICPR.2010.579.

HUYNH-THU, Q.; GHANBARI, M. Scope of validity of PSNR in image/video quality assessment. **Electronics Letters**, IET, v. 44, n. 13, p. 800–801, 2008. DOI: 10.1049/el:20080522.

INET FRAMEWORK. **IEEE 802.11 Handover**. 2023. INET Framework Documentation. Disponível em: <https://inet.omnetpp.org/docs/showcases/wireless/handover/doc/index.html>.

_____. **Mobility Models**. 2023. INET Framework Documentation. Disponível em: <https://inet.omnetpp.org/docs/showcases/mobility/basi%5C-c/doc/index.html>.

INET FRAMEWORK TEAM. **Using Real Applications in a Simulated Network**. 2024. A showcase from the INET Framework for OMNeT++ demonstrating video streaming emulation in a simulated network environment. Disponível em: <https://inet.omnetpp.org/docs/showcases/emulation/videostreaming/doc/index.html>.

ISAKA, Kazuki. SRT (Secure Reliable Transport). **Journal of Image Information and Media Society**, v. 74, n. 1, p. 102–104, 2020.

ITU-T. **Recommendation P.910: Subjective video quality assessment methods for multimedia applications**. Geneva, 2008.

JULURI, Parikshit; TAMARAPALLI, Venkatesh; MEDHI, Deep. Measurement of Quality of Experience of Video-on-Demand Services: A Survey. **IEEE Communications Surveys & Tutorials**, v. 18, p. 1–1, fev. 2015. DOI: 10.1109/COMST.2015.2401424.

KANAI, Kenji et al. Methods for Adaptive Video Streaming and Picture Quality Assessment to Improve QoS/QoE Performances. **IEICE Transactions on Communications**, The Institute of Electronics, Information e Communication Engineers, E102-B, n. 7, p. 1240–1247, 2019. DOI: 10.1587/transcom.2018ANI0003.

KAUFFMANN, B. **Protocolos de Transmissão de Vídeo sobre IP com Compressão**. Ad-Digital Tecnologia, 2023. Disponível em: https://setexperience2020.set.org.br/wp-content/uploads/2020/11/Protocolos-de-Transmissao-de-Video-sobre-IP.pdf.

KUROSE, James F.; ROSS, Keith W. **Redes de computadores e a internet: uma abordagem top-down**. 8. ed.: Bookman, 2021. P. 584. ISBN 9788582605592.

LAGHARI, Asif Ali et al. The state of art and review on video streaming. **J. High Speed Netw.**, IOS Press, NLD, v. 29, n. 3, p. 211–236, jan. 2023. ISSN 0926-6801. DOI: 10.3233/JHS-222087. Disponível em: <https://doi.org/10.3233/JHS-222087>.

LANGLEY, Adam et al. The QUIC Transport Protocol: Design and Internet-Scale Deployment. In: PROCEEDINGS of the Conference of the ACM Special Interest Group on Data Communication (SIGCOMM). ACM, 2017. P. 183–196. DOI: 10.1145/3098822.3098842.

MAYER, Christoph P; GAMER, Thomas. **Integrating real world applications into OMNeT++**. 2008. Disponível em: <http://doc.tm.uka.de/TM-2008-2.pdf>.

MORDOR INTELLIGENCE. **Brazil Surveillance IP Camera Market**. 2023. https://www.mordorintelligence.com/industry-reports/brazil-surveillance-ip-camera-market/market-trends. Acessado em 18 de fevereiro de 2025.

NETFLIX TECHNOLOGY BLOG. **Toward a Practical Perceptual Video Quality Metric**. 2016. https://netflixtechblog.com/toward-a-practical-perceptual-video-quality-metric-653f208b9652. Acesso em: jul. 2025.

NORDVPN. **VLAN Tagging**. 2023. https://nordvpn.com/pt-br/cybersecurity/glossary/vlan-tagging/. Accessed: date-of-access.

NUÑEZ, Lyndon; TOASA, Renato M. Performance evaluation of RTMP, RTSP and HLS protocols for IPTV in mobile networks. In: 2020 15th Iberian Conference on Information Systems and Technologies (CISTI). 2020. P. 1–5. DOI: 10.23919/CISTI49556.2020.9140848.

NURROHMAN, Anif; ABDUROHMAN, Maman. High Performance Streaming Based on H264 and Real Time Messaging Protocol (RTMP). In: 2018 6th International Conference on Information and Communication Technology (ICoICT). 2018. P. 174–177. DOI: 10.1109/ICoICT.2018.8528770.

PARMAR, H.; THORNBURGH, M. **Adobe's Real Time Messaging Protocol**. Dez. 2012. Accessed: 2025-02-19. Disponível em: <https://www.adobe.com/content/dam/acom/en/devnet/rtmp/pdf/rtmp_specification_1.0.pdf>.

PESSEMIER, Timothy De et al. Study of Temporal Effects on Subjective Video Quality of Experience. **Journal of Electronic Imaging**, SPIE, v. 24, n. 2, p. 023003, 2015. DOI: 10.1117/1.JEI.24.2.023003.

SCHULZRINNE, Henning et al. **RTP: A Transport Protocol for Real-Time Applications**. RFC Editor, jul. 2003. 104 p. RFC 3550. (Request for Comments, 3550). DOI: 10.17487/RFC3550. Disponível em: <https://www.rfc-editor.org/info/rfc3550>.

SCUSSEL, Vicente; MOREIRA, André; GASPARY, Luciano. Improvements in OMNeT++/INET Real-Time Scheduler for Emulation Mode. **Proceedings of Simutools**, 2015. Disponível em: <https://www.researchgate.net/publication/281670287_Improvements_in_OMNeTINET_Real-Time_Scheduler_for_Emulation_Mode>.

SHARABAYKO, M.; KIM, J. **SRT Protocol**. Jul. 2020. Online. Disponível em: https://datatracker.ietf.org/doc/html/draft-sharabayko-srt-01. Disponível em: <https://datatracker.ietf.org/doc/html/draft-sharabayko-srt-01>.

SHARABAYKO, Maxim. **Secure Reliable Transport (SRT)**. 2021. https://datatracker.ietf.org/doc/html/draft-sharabayko-srt-01. IETF Internet-Draft. Acesso em: 11 ago. 2025.

SISTEMA IRIS. **Armazenamento de imagens em nuvem para câmeras de segurança**. Acessado em: 18 de fevereiro de 2025. 2023. Disponível em: <https://www.sistemairis.com.br/armazenamento-de-imagens-em-nuvem/>.

SONONO, Tofik. **Interoperable Retransmission Protocols with Low Latency and Constrained Delay: A Performance Evaluation of RIST and SRT**. Jul. 2019. Master's Thesis – KTH Royal Institute of Technology, Stockholm, Sweden. Disponível em: <https://kth.diva-portal.org/smash/get/diva2:1331413/FULLTEXT01.pdf>.

SPASOJEVIC, Anastasia. **O que é MD5 (algoritmo de resumo de mensagem)?** Ago. 2024. Disponível em: https://phoenixnap.pt/glossÃąrio/algoritmo-de-resumo-de-mensagem-md5. Disponível em: <https://phoenixnap.pt/gloss%C3%A1rio/algoritmo-de-resumo-de-mensagem-md5>.

STEWART, Randall R. **Stream Control Transmission Protocol**. 2007. RFC 4960. Acesso em: 04 ago. 2025. Disponível em: <https://www.rfc-editor.org/rfc/rfc4960>.

WANG, Zhou et al. Image quality assessment: From error visibility to structural similarity. **IEEE Transactions on Image Processing**, v. 13, n. 4, p. 600–612, 2004. DOI: 10.1109/TIP.2003.819861.

WIKIPEDIA CONTRIBUTORS. **Differentiated services**. 2023. https://en.wikipedia.org/wiki/Differentiated_services. Accessed: date-of-access.

XU, Jinheng et al. The impact of bitrate and GOP pattern on the video quality of H.265/HEVC compression standard. In: 2018 IEEE International Conference on Signal Processing, Communications and Computing (ICSPCC). 2018. P. 1–5. DOI: 10.1109/ICSPCC.2018.8567817.

APÊNDICE A
EXECUÇÃO DOS TESTES DE SIMULAÇÃO NO OMNET++

Este anexo descreve, passo a passo, como instalar o ambiente, compilar, executar e monitorar os testes de simulação realizados com OMNeT++ e o *framework* INET. O procedimento segue as recomendações da documentação oficial do INET (INET FRAMEWORK TEAM, 2024).

Pré-requisitos do sistema

- Linux (Ubuntu 22.04 ou superior);
- OMNeT++ 6.0 ou 6.1 instalado;
- INET 4.5 instalado e compilado;
- Permissões `sudo` para uso de TAP;
- Kernel com suporte a TUN/TAP;
- Pacotes: `libpcap-dev`, `tcpdump`, `ffmpeg`, `python3-matplotlib`;
- Sincronização de horário recomendada via `ntp` ou similar.

Instalação do Projeto Base

Como ponto de partida, realize a instalação do software omnet.

Disponível em: [https://omnetpp.org/download/]

Instalação das dependências

Antes de compilar o OMNeT++, é necessário instalar os pacotes exigidos pelo ambiente de desenvolvimento. Execute o seguinte comando no terminal:

```
$ sudo apt update

$ sudo apt install -y build-essential gcc g++ bison flex perl&#x20;
python3 qtbase5-dev qtchooser qt5-qmake qtbase5-dev-tools&#x20;
libqt5opengl5-dev libxml2-dev zlib1g-dev doxygen graphviz&#x20;
libtool libopenmpi-dev openmpi-bin libpcap-dev default-jre&#x20;

$ cmake
```

INSTALAÇÃO DO NGINX COM MÓDULO RTMP

A versão disponível nos repositórios oficiais do Ubuntu não inclui o módulo RTMP. Para realizar transmissões com o protocolo RTMP, é necessário compilar o Nginx com suporte a esse módulo. A seguir, apresenta-se o passo a passo.

Instalação das dependências

```
$ sudo apt update
$ sudo apt install -y build-essential libpcre3 libpcre3-dev libssl-dev zlib1g-dev
    git
```

Download do Nginx e do módulo RTMP

```
$ cd ~
$ wget http://nginx.org/download/nginx-1.24.0.tar.gz
$ tar -zxvf nginx-1.24.0.tar.gz

$ git clone https://github.com/arut/nginx-rtmp-module.git
```

Compilação do Nginx com suporte a RTMP

```
$ cd nginx-1.24.0
./configure --with-http_ssl_module --add-module=../nginx-rtmp-module
make
$ sudo make install
```

Verificação do binário

```
/usr/local/nginx/sbin/nginx -V
```

O comando acima deve exibir algo semelhante a: `-add-module=../nginx-rtmp-module`

Configuração do RTMP no Nginx

Edite o arquivo de configuração:

```
$ sudo nano /usr/local/nginx/conf/nginx.conf
```

Adicione o bloco abaixo ao final do arquivo:

```
rtmp {
    server {
        listen 1935;
        chunk_size 4096;

        application live {
            live on;
            record off;
        }
    }
}
```

Salve e feche o editor.

Inicialização do Nginx com suporte RTMP

Para iniciar o serviço:

```
$ sudo /usr/local/nginx/sbin/nginx
```

Para parar o serviço:

```
$ sudo /usr/local/nginx/sbin/nginx -s stop
```

Recarregar as configurações

```
$ sudo /usr/local/nginx/sbin/nginx -s reload
```

Download e extração do OMNeT++

A versão mais recente do OMNeT++ pode ser obtida diretamente no site oficial (https://omnetpp.org). Para sistemas Linux, recomenda-se o pacote específico:

```
$ wget [https://github.com/omnetpp/omnetpp/releases/download/omnetpp-6.1/omnetpp-6.1-linux-x86_64.tgz](https://github.com/omnetpp/omnetpp/releases/download/omnetpp-6.1/omnetpp-6.1-linux-x86_64.tgz)
```

Após o download, mova o arquivo para o diretório desejado (por exemplo, o seu diretório `home`) e extraia o conteúdo:

```
$ tar xvfz omnetpp-6.1-linux-x86_64.tgz
$ cd omnetpp-6.1
```

Configuração das variáveis de ambiente

```

# Acesse o diretório do OMNeT++

$ cd ~/Downloads/omnetpp-6.1

# Ative o ambiente

$ . setenv

# No diretório OMNeT++, execute:

$ ./configure

# Quando ./configure terminar, digite no terminal:

$ make

# execute o software

$ omnetpp
```

Com o OMNeT++ instalado e compilado, é necessário instalar o INET Framework, que fornece os modelos de rede utilizados nos experimentos.

Dentro da interface gráfica do OMNeT++, vá até o menu **Help Install Simulation Models**, selecione a opção **INET Framework 4.5** e clique em **Install Project**. Aguarde até que a instalação e a compilação do projeto sejam concluídas.

Preparação para o modo de emulação

Para que as simulações possam ser executadas com tráfego real via interfaces TAP, é necessário habilitar o modo de emulação e garantir as permissões adequadas no sistema operacional.

- Primeiramente, é necessário habilitar a funcionalidade **Network Emulation** no projeto OMNeT++. Para isso, abra a IDE do OMNeT++, vá até o menu **Project Project Features...** e marque a opção **Network Emulation**.
- Em seguida, o executável `opp_run` deve ter permissões especiais para manipular *raw sockets*. Isso pode ser feito com o comando `setcap`, conforme abaixo:

```
$ sudo setcap cap_net_raw,cap_net_admin=eip ~/omnetpp-6.1/bin/opp_run
$ sudo setcap cap_net_raw,cap_net_admin=eip ~/omnetpp-6.1/bin/opp_run_dbg
$ sudo setcap cap_net_raw,cap_net_admin=eip ~/omnetpp-6.1/bin/opp_run_release
```

Essas permissões permitem que o OMNeT++ envie e receba pacotes reais através das interfaces de rede TAP sem a necessidade de executar o simulador como `root`, aumentando a segurança da operação.

Encerre a IDE do OMNeT e abra novamente o terminal,

```

# Acesse o diretório do OMNeT++

$ cd Downloads/omnetpp-6.1

# Ative o ambiente

$ . setenv

# Acesse o diretório do inet4.5

$ cd sample/inet4.5

# Ative novamente o ambiente OMNeT++

```

```
$ . setenv

# Compile o INET

$ make makefiles
$ make -j$(nproc)

# Acesse o diretório do videostreaming

$ showcases/emulation/videostreaming

# execute o script

$ ./setup.sh
$ ./run.sh

# Após a execução correta do showcase de exemplo, execute o comando:

$ ./teardown
```

APÊNDICE B
SCRIPTS DE TRANSMISSÃO E ANÁLISE

Repositório do projeto modificado

A versão modificada utilizada nesta pesquisa está disponível em:

- https://github.com/anahoog/videostreaming

Após clonar o repositório do projeto, substitua a pasta do videostreaming dentro do diretório showcases/emulation/.

Coleta e análise de métricas

Durante e após a simulação, podem ser coletadas métricas a partir dos arquivos gerados:

- `*.pcap`: pacotes capturados com tcpdump;
- `*.csv`: métricas de vídeo exportadas de FFmpeg;
- Scripts auxiliares do repositório podem calcular: PSNR, SSIM, VMAF, vazão média, jitter, perdas e rebuffering.

Instalação do VMAF (Avaliação de Qualidade de Vídeo)

```
1
2  # Clonar o repositório oficial
3
4  $ git clone [https://github.com/Netflix/vmaf.git](https://github.com/Netflix/vmaf.
       git)
5  $ cd vmaf
6
7  # Instalar dependências de compilação (Ubuntu)
8
9  $ sudo apt install meson ninja-build nasm pkg-config libpthread-stubs0-dev
10
11 # Gerar e compilar libvmaf
12
13 $ meson setup build --buildtype release
14 $ meson compile -C build
15 $ sudo meson install -C build
```

Integração com FFmpeg

```
$ ffmpeg -i vídeo_referência.mp4 -i vídeo_teste.mp4&#x20;
-lavfi libvmaf="model_path=/usr/local/share/model/vmaf_v0.6.1.pkl"&#x20;
-f null -
```

Automação nos scripts de análise

Os scripts `Python` do repositório `videostreaming` já chamam o FFmpeg com a opção `libvmaf`. Basta atualizar a variável `VMAF_MODEL` para o caminho adequado do modelo desejado.

```

# Cenário sem erro e delay

$ ./VideoStreamingShowcase -u Cmdenv -f omnetpp.ini -c General-01

# Cenário com mobilidade

$ ./VideoStreamingShowcase -u Cmdenv -f omnetpp.ini -c UmMovel

# Cenário com mobilidade e tráfego de fundo

$ ./VideoStreamingShowcase -u Cmdenv -f omnetpp.ini -c DoisMoveis
```

Execução da transmissão com FFmpeg

Esses scripts utilizam o `FFmpeg` para enviar fluxos de vídeo codificados com H.264 ou H.265, com diferentes taxas de bitrate e protocolos. Abaixo está um exemplo de execução:

```

# Vídeo com compressão H.264, bitrate adaptativo e protocolo SRT.
# É possível escolher entre os scripts abaixo, definindo o protocolo no final: srt,
    rtp ou rtmp.
# ./ffmpeg_264.sh
# ./ffmpeg_264_2m.sh
# ./ffmpeg_264_4m.sh
# ./ffmpeg_265.sh
# ./ffmpeg_265_2m.sh
# ./ffmpeg_265_4m.sh
```

```
10
11 $ ./ffmpeg_264.sh srt
```

Esses scripts realizam tanto a transmissão quanto a captura de pacotes (via `tcpdump`) e a avaliação de qualidade (usando `ffmpeg` com métricas como PSNR, SSIM e VMAF), conforme descrito nas seções posteriores. Após a finalização, execute a coleta do VMAF:

```
1 $ ffmpeg -i recebido.ts -i transmitido.mp4  -lavfi libvmaf="log_path=vmaf.json:
      log_fmt=json"  -f null -
```

Resultados das Métricas Obtidas

```
1  $ python3 obter_metricas-v2.py capturas/rtp_20250720_23-14-23/
2  Arquivo VMAF encontrado: capturas/rtp_20250720_23-14-23/vmaf.json
3  Total de frames carregados: 3068
4  Arquivo salvo com 3068 linhas em: capturas/rtp_20250720_23-14-23/vmaf_por_frame.txt
5  PSNR médio (dB): 17.67
6  SSIM médio: 0.6982
7  VMAF médio: 24.31
8  Vazão média (Mbps): 0.583
9
10 == Estatísticas de Transmissão/Recepção ==
11 Pacotes transmitidos: 3792
12 Pacotes recebidos: 3014
13 Pacotes perdidos: 778
14 Perda percentual: 20.52%
15 Jitter médio (ms): 32.71
16 Rebuffering (eventos): 254
```

Nota sobre o cálculo de vazão no `obter_metricas-v2.py`.

O valor exibido como "Vazão média (Mbps)" nesse script *não representa a vazão efetiva de rede*. Ele é obtido calculando uma média dos valores `bitrate=` reportados pelo `ffmpeg` transmissor. Esses valores já são médias acumuladas da *taxa de codificação de mídia* (bytes codificados / tempo), e não incluem o overhead dos protocolos de transporte nem refletem perdas, retransmissões ou limitação de throughput na rede. Além disso, fazer média aritmética de várias leituras acumuladas introduz viés (média de médias). Assim, trate esse campo apenas como uma estimativa aproximada da *taxa codificada* e **não** como medida comparável de desempenho de rede entre protocolos.

Cálculo recomendado de vazão de rede (`calcular_vazão.py`).

Para medir a *vazão média de rede* de forma consistente entre cenários, utilize um script separado (`calcular_vazão.py`) que opera sobre capturas de pacotes processadas pelo `tshark`.

APÊNDICE C
TABELA DE RESULTADOS DOS TESTES

Tabela 42 – Resultados para o Cenário 1 – H.264 a 2 Mbps

Protocolo	PSNR (dB)	SSIM	VMAF	Vazão (Mbps)	Perda	Jitter (ms)	Rebuffering
SRT	46,84	0,9593	86,32	3,901	0,41%	15,63	200
RTP	40,59	0,9610	93,90	3,906	0,05%	17,80	225
RTMP	47,30	0,9705	91,33	5,450	0,00%	22,93	217

Fonte: Elaborada pelo autor.

Tabela 43 – Resultados para o Cenário 1 – H.264 a 4 Mbps

Protocolo	PSNR (dB)	SSIM	VMAF	Vazão (Mbps)	Perda	Jitter (ms)	Rebuffering
SRT	50,10	0,9508	85,94	6,379	1,55%	12,51	174
RTP	41,58	0,9534	93,83	6,416	3,78%	13,43	202
RTMP	49,94	0,9580	91,22	8,590	3,95%	19,26	173

Fonte: Elaborada pelo autor.

Tabela 44 – Métricas de qualidade de vídeo e desempenho de rede para o Cenário 1 – H.264 a 4 Mbps - com delay e perdas

Protocolo	PSNR (dB)	SSIM	VMAF	Vazão (Mbps)	Perda (%)	Jitter (ms)	Rebuffering
SRT	41,08	0,9214	73,36	6,995	1,73	8,77	68
RTP	16,64	0,6521	23,96	6,589	4,89	11,72	143
RTMP	25,72	0,7726	32,24	5,294	62,98	10,16	23

Fonte: Elaborada pelo autor.

Tabela 45 – Resultados para o Cenário 1 – H.265 a 2 Mbps

Protocolo	PSNR (dB)	SSIM	VMAF	Vazão (Mbps)	Perda	Jitter (ms)	Rebuffering
SRT	40,62	0,9799	88,47	1,197	1,13%	8,19	2
RTP	36,78	0,9619	88,13	1,101	6,65%	19,71	28

Fonte: Elaborada pelo autor.

Tabela 46 – Resultados para o Cenário 1 – H.265 a 4 Mbps

Protocolo	PSNR (dB)	SSIM	VMAF	Vazão (Mbps)	Perda	Jitter (ms)	Rebuffering
SRT	63,20	0,9997	98,08	8,647	1,08%	4,14	3
RTP	42,16	0,9749	98,11	8,629	6,59%	4,94	10

Fonte: Elaborada pelo autor.

Tabela 47 – Métricas de desempenho para o cenário *UmMovel* com compressão H.264

Protocolo	PSNR (dB)	SSIM	VMAF	Vazão (Mbps)	Perda	Jitter (ms)	Rebuffering
SRT	42,07	0,9832	91,8	1,294	0,55%	21,39	231
SRT	42,73	0,9833	94,01	1,313	0,00%	21,32	219
SRT	22,59	0,7659	18,42	1,060	26,71%	24,08	238
RTP	26,30	0,8564	45,20	1,064	42,35%	34,63	249
RTP	17,67	0,6982	24,31	1,198	20,52%	32,71	254
RTP	28,99	0,8843	62,68	1,107	34,97%	33,92	249
RTMP	32,10	0,8887	58,52	1,016	42,11%	261,31	234
RTMP	37,80	0,9443	78,52	1,367	20,23%	77,09	223
RTMP	27,28	0,8356	44,71	1,041	55,14%	102,60	245

Fonte: Elaborada pelo autor.

Tabela 48 – Métricas de desempenho para o cenário *UmMovel* com compressão H.264 a 2 Mbps

Protocolo	PSNR (dB)	SSIM	VMAF	Vazão (Mbps)	Perda	Jitter (ms)	Rebuffering
SRT	44,71	0,9344	80,89	3,672	3,05%	11,15	89
SRT	20,10	0,7762	4,89	0,213	55,00%	86,96	19
SRT	44,56	0,93	80,91	3,686	2,99%	11,3	94
RTP	17,66	0,6908	2,89	3,319	12,93%	18,61	184
RTP	39,52	0,9418	88,42	3,736	2,80%	15,76	179
RTP	17,04	0,6688	31,75	3,745	2,86%	15,77	185
RTMP	26,24	0,7785	28,77	3,636	38,88%	27,72	146
RTMP	28,83	0,7952	33,83	4,897	10,68%	21,68	108
RTMP	28,98	0,7967	34,32	4,944	16,45%	12,95	111

Fonte: Elaborada pelo autor.

Tabela 49 – Métricas do cenário *UmMovel* – H.264 a 4 Mbps

Protocolo	PSNR (dB)	SSIM	VMAF	Vazão (Mbps)	Perda (%)	Jitter (ms)	Buffering
SRT	18,01	0,6343	9,84	4,463	68,53	18,03	89
SRT	26,63	0,7647	44,90	4,629	47,32	12,53	128
SRT	43,28	0,9070	73,91	5,619	8,80	9,20	50
RTP	24,88	0,7718	30,12	4,712	65,21	14,21	177
RTP	35,73	0,8873	75,72	5,928	3,43	11,54	108
RTP	32,92	0,8672	65,75	5,807	11,38	11,19	99
RTMP	28,19	0,8225	29,89	6,624	51,05	12,79	76
RTMP	28,71	0,7953	33,49	6,812	35,66	17,50	47
RTMP	22,22	0,7327	18,65	4,579	54,78	35,61	153

Fonte: Elaborada pelo autor.

Tabela 50 – Métricas do cenário *UmMovel* – H.265

Protocolo	PSNR (dB)	SSIM	VMAF	Vazão (Mbps)	Perda (%)	Jitter (ms)	Buffering
SRT	40,77	0,9800	88,45	1,206	0,29	7,02	2
SRT	39,41	0,9634	88,33	1,210	2,14	7,13	2
SRT	19,95	0,7785	2,95	0,663	48,50	19,32	3
RTP	17,59	0,6910	88,05	1,092	7,83	17,81	30
RTP	20,73	0,7330	25,75	0,837	61,74	21,49	107
RTP	16,54	0,6726	14,80	1,060	12,08	19,76	34

Fonte: Elaborada pelo autor.

Tabela 51 – Métricas do cenário *UmMovel* – H.265 a 2 Mbps

Protocolo	PSNR (dB)	SSIM	VMAF	Vazão (Mbps)	Perda (%)	Jitter (ms)	Buffering
SRT	40,71	0,9800	88,46	1,206	0,42	7,07	2
SRT	40,76	0,9800	88,45	1,206	0,37	7,02	2
SRT	22,18	0,8043	13,30	0,964	41,87	15,14	3
RTP	34,73	0,9476	87,27	1,088	11,58	17,86	37
RTP	18,54	0,7458	2,41	0,756	57,54	22,48	109
RTP	19,68	0,7270	24,33	0,755	71,84	22,77	127

Fonte: Elaborada pelo autor.

Tabela 52 – Métricas do cenário *UmMovel* – H.265 a 4 Mbps

Protocolo	PSNR (dB)	SSIM	VMAF	Vazão (Mbps)	Perda (%)	Jitter (ms)	Buffering
SRT	19,43	0,7007	46,06	3,812	47,64	27,39	85
SRT	13,29	0,3980	5,02	4,212	95,18	10,60	4
SRT	20,84	0,6843	70,69	5,716	51,94	5,63	2
RTP	31,99	0,8458	93,83	7,144	15,19	2,84	4
RTP	32,60	0,8724	87,57	7,125	20,36	3,25	8
RTP	32,98	0,8402	88,35	6,968	23,76	5,17	6

Fonte: Elaborada pelo autor.

Tabela 53 – Métricas para o cenário DoisMoveis – H.264

Protocolo	PSNR (dB)	SSIM	VMAF	Vazão (Mbps)	Perda (%)	Jitter (ms)	Buffering
SRT	41,79	0,9831	90,33	1,312	0,55%	21,24	219
SRT	21,14	0,7731	37,51	1,057	22,39%	33,05	131
SRT	42,66	0,9828	93,98	1,320	1,08%	20,61	198
RTP	17,67	0,6983	3,10	1,173	20,52%	33,05	251
RTP	18,74	0,7383	4,42	4,679	90,93%	8,01	9
RTP	23,96	0,7541	37,88	6,127	52,00%	5,30	5
RTMP	24,60	0,7570	24,50	7,721	44,55%	6,86	5
RTMP	36,39	0,9348	73,82	1,278	25,24%	109,48	225
RTMP	22,22	0,6480	29,95	0,827	69,89%	30,81	250

Tabela 54 – Métricas para o cenário DoisMoveis – H.264 – 2 Mbps

Protocolo	PSNR (dB)	SSIM	VMAF	Vazão (Mbps)	Perda (%)	Jitter (ms)	Buffering
SRT	43,84	0,9409	80,69	3,651	4,23%	11,75	97
SRT	44,48	0,9350	80,72	3,624	4,23%	10,15	69
SRT	21,50	0,7028	20,76	2,918	40,18%	16,08	151
RTP	20,10	0,5596	16,61	2,581	85,70%	20,78	225
RTP	38,93	0,9346	84,82	3,702	3,90%	14,86	158
RTP	17,60	0,6809	2,61	3,509	5,24%	16,99	180
RTMP	26,40	0,8063	39,01	3,466	60,74%	18,23	173
RTMP	24,03	0,7807	27,64	4,811	24,12%	12,65	85
RTMP	40,24	0,9290	82,63	5,095	12,67%	13,01	99

Tabela 55 – Métricas do cenário DoisMoveis – H.264 a 4 Mbps

Protocolo	PSNR (dB)	SSIM	VMAF	Vazão (Mbps)	Perda (%)	Jitter (ms)	Buffering
SRT	27,74	0,7298	43,73	4,664	53,01%	12,12	120
SRT	32,37	0,8706	50,55	5,106	16,28%	10,91	48
SRT	43,17	0,9103	74,08	5,652	8,77%	9,63	60
RTP	16,56	0,6659	24,76	5,867	4,56%	11,25	109
RTP	16,56	0,6659	24,76	5,867	4,56%	11,25	109
RTP	23,64	0,72	27,55	4,537	68,87%	14,60	191
RTMP	26,49	0,8042	31,57	6,774	42,78%	11,39	36
RTMP	20,67	0,6242	10,90	3,445	90,53%	28,70	238
RTMP	25,10	0,764	27,41	4,533	72,89%	18,32	184

Fonte: Elaborada pelo autor.

Tabela 56 – Métricas para o cenário DoisMoveis – H.265

Protocolo	PSNR (dB)	SSIM	VMAF	Vazão (Mbps)	Perda	Jitter (ms)	Rebuffering
SRT	22,74	0,7298	43,73	1,204	53,01%	12,12	120
SRT	36,19	0,9424	69,61	1,160	16,44%	8,03	3
SRT	18,82	0,6653	11,60	0,689	46,27%	19,04	3
RTP	30,49	0,9125	73,10	1,021	26,16%	18,46	47
RTP	31,34	0,9059	72,81	1,035	22,60%	18,05	43
RTP	36,78	0,9619	88,13	1,101	6,65%	17,48	28

Tabela 57 – Métricas para o cenário DoisMoveis – H.265 – 2MB

Protocolo	PSNR (dB)	SSIM	VMAF	Vazão (Mbps)	Perda	Jitter (ms)	Rebuffering
SRT	19,95	0,6618	18,19	0,979	35,54%	14,39	3
SRT	31,06	0,8876	62,82	1,247	11,05%	8,22	3
SRT	19,95	0,7785	2,95	0,331	69,64%	35,88	12
RTP	17,36	0,6892	5,88	1,054	16,77%	12,28	38
RTP	29,12	0,8971	68,41	0,986	34,57%	18,88	64
RTP	36,78	0,9616	88,13	1,101	6,65%	17,68	28

Tabela 58 – Métricas para o cenário DoisMoveis – H.265 – 4 Mbps

Protocolo	PSNR (dB)	SSIM	VMAF	Vazão (Mbps)	Perda (%)	Jitter (ms)	Buffering
SRT	26,88	0,7848	71,39	6,613	13,49%	3,54	2
SRT	17,32	0,6951	55,08	5,973	23,41%	3,75	2
SRT	28,68	0,7854	78,07	4,235	16,86%	23,20	71
RTP	31,36	0,8361	87,72	6,981	16,24%	2,76	4
RTP	15,39	0,6396	4,89	6,182	44,54%	5,35	8
RTP	33,19	0,8556	93,86	7,308	13,34%	2,90	5