Universidade Federal da Paraíba
Centro de Informática
Programa de Pós-Graduação em Informática

Transmissão de Áudio com Baixa Latência sobre Redes Wireless IEEE 802.11 com Foco em Acessibilidade para o Cinema Digital

Caio Marcelo Campoy Guedes

Proposta de Dissertação submetida à Coordenação do Curso de Pós-Graduação em Informática da Universidade Federal da Paraíba como parte dos requisitos necessários para obtenção do grau de Mestre em Informática.

Área de Concentração: Ciência da Computação
Linha de Pesquisa: Computação Distribuída

Guido Lemos de Souza Filho
(Orientador)

João Pessoa, Paraíba, Brasil

Catalogação na publicação
Seção de Catalogação e Classificação

G924t Guedes, Caio Marcelo Campoy.
Transmissão de áudio com baixa latência sobre redes wireless IEEE 802.11 com foco em acessibilidade para o cinema digital / Caio Marcelo Campoy Guedes. - João Pessoa, 2021.
147 f. : il.

Orientação: Guido Lemos de Souza Filho.
Dissertação (Mestrado) - UFPB/CI.

1. Ciência e tecnologia informática. 2. Transmissão de áudio. 3. Algoritmo adaptativo. 4. Smartphones. 5. Wireless. I. Souza Filho, Guido Lemos de. II. Título.

UFPB/BC CDU 004(043)

Elaborado por Gracilene Barbosa Figueiredo - CRB-15/794

"Imagination will often carry us to worlds that never were.

But without it we go nowhere."

(Carl Sagan)

Agradecimentos

Em primeiro lugar, gostaria de agradecer aos meus pais, Marcelo e Cleide, por todo o amor e carinho, pelos exemplos, oportunidades e conselhos que me foram dados ao longo dessa jornada.

Ao meu irmão Vitor, pelos bons momentos ao longo da minha vida, sempre trilhando comigo as dificuldades e as alegrias. Parabéns, meu grande ortopedista.

Ao professor Dr. Guido Lemos, meu orientador, por ter me acolhido durante a graduação e novamente durante a pós-graduação. Um profissional admirável e uma pessoa incrível. Agradeço pela confiança depositada, pelas cobranças e por extrair de mim o meu máximo potencial.

Aos professores Dr. Tiago Maritan e Dr. Denio Mariz, por todo o auxílio durante o desenvolvimento deste e de tantos outros projetos durante a minha formação.

À Danilo, Erickson, Bruno, Sindolfo, André, Jorismar e Manoel, pela ajuda, discussões e incontáveis horas de esforço e dedicação, que auxiliaram o desenvolvimento desse projeto.

Aos meus amigos Thiago, Renê, Josué, Stefano, Amon, Victor, Nycholas, Yure, Edson, João Vinicius, Deborah e a tantos outros que, embora não tenham sido citados, sempre residirão em minhas memórias. Saibam que nossos encontros e toda e qualquer palavra de incentivo foram de suma importância para o andamento e conclusão deste trabalho.

Por fim, gostaria de agradecer à empresa Assista Tecnologia pelo apoio durante o desenvolvimento deste trabalho.

Resumo

A transmissão do reforço de áudio de um servidor para celulares usados por pessoas com deficiência auditiva durante a exibição de filmes em salas de cinema é um exemplo das aplicações que utilizam redes Wi-Fi e requerem a transmissão de áudio com baixa latência, sincronismo e resiliência. Nessa aplicação é desejável que o retardo entre a captura e exibição do áudio fique abaixo de 40 ms, requisito que implica na otimização dos processos de codificação, transmissão e decodificação. Este estudo tem como objetivo desenvolver uma solução cliente-servidor capaz de transmitir áudio em tempo real sobre redes *wireless* IEEE 802.11 entre um servidor e clientes executando em dispositivos móveis. O trabalho busca limitar a latência máxima entre fonte e reprodutor a um *threshold* de 40 milissegundos definido como limiar para a percepção de *lip sync*. Manter baixa latência evitando variações no retardo e perda de pacotes em ambientes voláteis como redes *wireless* é um problema complexo. Portanto, almejando o máximo de resiliência, a solução faz uso de *Forward Error Correction* e propõe o uso de redundância temporal para mitigar os efeitos indesejáveis provocados pelas perdas e descartes de pacotes atrasados. Adicionalmente é proposto e utilizado um novo algoritmo adaptativo que busca ajustar o *buffer* de *playback* com o propósito de atenuar o *jitter* da rede.

Palavras-chave: Transmissão de áudio, Áudio, FEC, Algoritmo Adaptativo, Redundância, Smartphones, Wireless, Latência, Jitter.

Abstract

The transmission of audio reinforcement from a server to mobile phones used by the hearing impaired during the presentation of a film in cinemas is an example among applications that use Wi-Fi networks and require the transmission of audio with low latency, synchronism and resilience. In this application, it is meant that the delay between capturing and reproducing the audio stream lays below 40 ms, a requirement that focus on the optimization of encoding, transmission and decoding of the solution. This study aims to develop a client-server solution capable of transmitting audio in real time over IEEE 802.11 wireless networks between a server and clients running on mobile devices. The work seeks to limit the maximum latency between source and player to a threshold of 40 milliseconds defined as a threshold for the perception of lip sync. Maintaining low latency by avoiding variations in delay and packet loss in volatile environments such as wireless networks is a complex problem. Therefore, aiming for maximum resilience, the solution makes use of Forward Error Correction and proposes the use of temporal redundancy to mitigate the undesirable effects caused by the loss and discarding of delayed packages. Additionally, a new adaptive algorithm is proposed and used to adjust the buffer of playback in order to mitigate the network jitter.

Keywords: Audio Streaming, Audio, FEC, Adaptive Algorithm, Redundancy, Smartphones, Wireless, Latency, Jitter.

Conteúdo

Lista de Símbolos

ABNT : *Associação Brasileira de Normas Técnicas*

ACM : *Association for Computing Machinery*

ALSA : *Advanced Linux Sound Architecture*

API : *Application Programming Interface*

CELT : *Constrained Energy Lapped Transform*

COTS : *Commercial off-the-shelf*

DAC : *Digital Analog Converter*

DSSS : *Direct Sequence Spread Spectrum*

FAST : *Fast Audio Streamer*

FEC : *Forward Error Correction*

FHSS : *Frequency-hopping spread spectrum*

FM : *Frequency Modulation*

GNU : *GNU is Not Unix*

IEEE : *Institute of Electrical and Electronics Engineers*

IP : *Internet Protocol*

MAC : *Media Access Control Address*

OFDM : *Orthogonal Frequency Division Multiplexing*

PCM : *Pulse Code Modulation*

RAM : *Random Access Memory*

TRS : *Tip-Ring-Sleeve*

UDP : *User Datagram Protocol*

VOIP : *Voice Over IP*

VHF : *Very High Frequency*

WLAN : *Wireless Local Area Network*

WPAN : *Wireless Personal Area Network*

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Com a constante luta para inclusão de pessoas com deficiência visual e auditiva em apresentações artísticas, têm-se observado o surgimento de diversas tecnologias desenvolvidas com o propósito de auxiliar estes indivíduos a terem melhor compreensão sobre conteúdos apresentados, seja uma peça teatral ou um filme exibido em salas de cinema. As tecnologias presentes no mercado variam desde o uso de painéis textuais onde é realizada estenografia, ao uso de transmissores de áudio e a presença de intérpretes de línguas de sinais.

O reforço de áudio individual é um recurso bastante delicado, uma vez que variações no envio do conteúdo, seja por perda de pacotes ou latência, afetam diretamente a experiência do usuário, demandando soluções para contornar essas falhas. A proposta de uma solução tecnológica para transmissão de reforço de áudio é o tema do presente trabalho.

A Figura 1.1, apresenta uma solução baseada na transmissão de conteúdo sobre sinais FM, produzida pela Dolby *Laboratories*[1], denominada Fidelio[2]. A vantagem desse tipo de equipamento é a grande área de cobertura e a facilidade no transporte, uma vez que apenas é necessário mover a base de transmissão para se ter o equipamento completamente funcional em um novo ambiente.

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

[2]https://professional.dolby.com/product/dolby-accessibility-solutions-for-cinema/fidelio/

Figura 1.1: Componentes do Dolby Fidelio

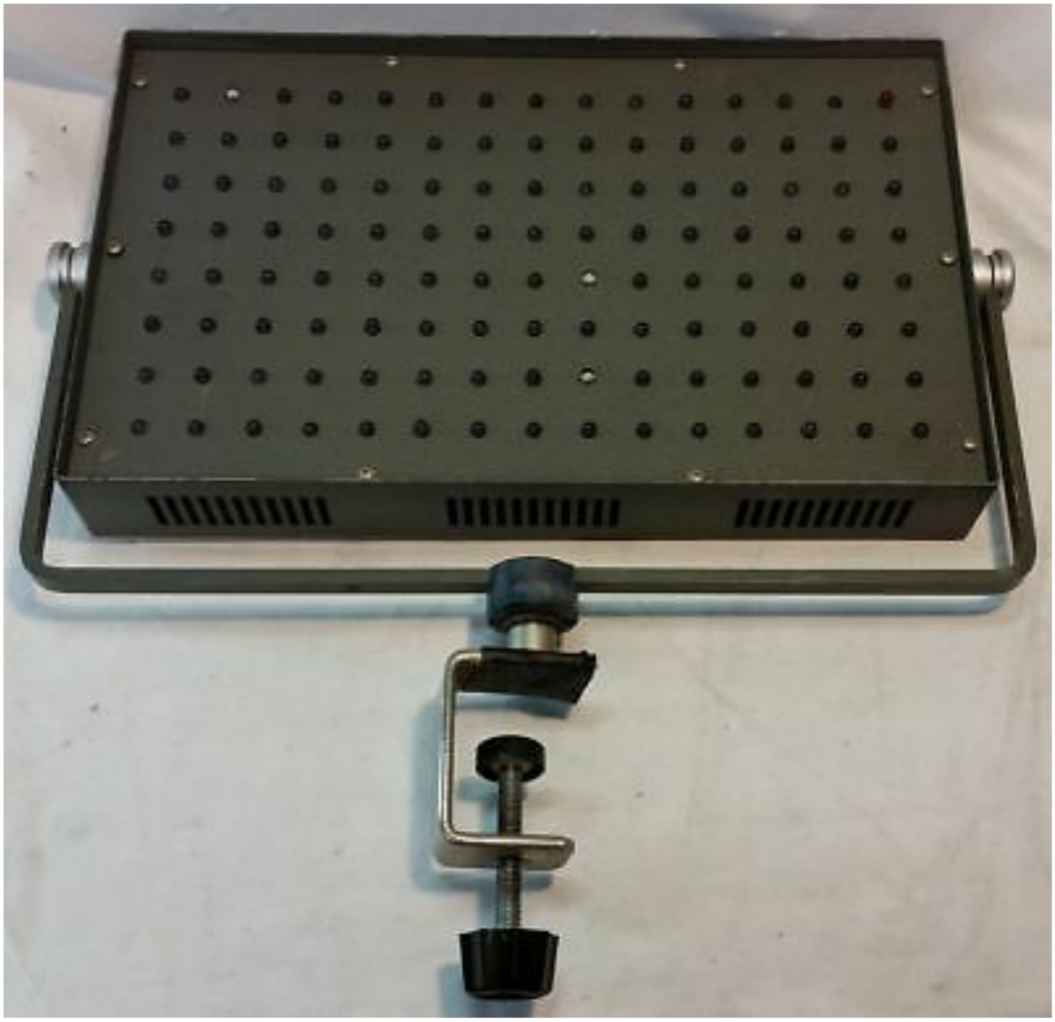
Fonte: https://cinemanext.com/dolby-fidelio, acessado em: 14/01/2021

Entretanto, a alocação de frequências FM deve ser efetuada junto a um órgão governamental especializado, fazendo-se necessário realizar um trâmite extenso em cada local onde se deseja implantar a solução. Ressalta-se também que a regulamentação depende dos órgãos nacionais especializados, que pode variar a depender do país em questão. Segundo Lan [Lan e Li 2010], o principal problema no acesso à tecnologia de radiofrequência está intrinsecamente relacionado à defasada regulamentação e à baixa efetividade nos padrões de alocação, o que limita o acesso aos recursos e vantagens oferecidas pela tecnologia. Outro fator relevante é o investimento necessário para a aquisição de equipamentos dedicados e aquisição da concessão pública para uso da frequência desejada que difere entre países e regiões.

Outra solução presente no mercado é a transmissão de reforço de áudio a partir de um transmissor infravermelho, representado na Figura 1.2 pelo painel infravermelho da Sennheiser SZI 1019[3]. Esse tipo de tecnologia possui alcance muito menor do que a solução que utiliza FM, entretanto, ela possui o benefício de não necessitar de homologação junto aos órgãos governamentais para atuação, o que a torna mais atrativa. Porém, ela ainda utiliza um *hardware* dedicado, o que encarece a solução.

[3]https://en-us.sennheiser.com/szi-1029

Figura 1.2: Painel infravermelho Sennheiser SZI 1029



Fonte: https://www.ebay.com/itm/Sennheiser-SZI-1019-Infrared-audio-transmitter-for-hea-Used-NO-AC-POWER-ADAPTER/273771995051, acessado em: 14/01/2021

Dentre as alternativas baseadas em transmissão não guiada, a comunicação através de redes *wireless* tem se difundido rapidamente por conta da sua praticidade e eficiência beneficiando diferentes aplicações como, por exemplo, a transmissão de conteúdos audiovisuais, jogos e centros de entretenimento multimídia [Kovacevic, Samardzija e Temerinac 2009]. A transmissão sem fio também se destaca pela flexibilidade no conjunto de soluções tecnológicas como as propostas nos padrões de comunicação elaborados pelo IEEE (*Institute of Electrical and Electronics Engineers*), que permitem o uso de diferentes frequências e

larguras de banda [Mangold et al. 2003]. Com a facilidade de implantação e baixo custo na instalação de bases *wireless*, serviços que utilizam transmissão de dados para múltiplos pontos podem ainda se beneficiar de protocolos que otimizam a transmissão de fluxos de áudio para grupos (*multicast*). Todavia, ao utilizar redes *wireless*, observa-se grande instabilidade quando se trata de transmissão de baixa latência que é, em geral, proveniente de atraso no envio do conteúdo e a colisão entre os pacotes [Hardman et al. 2006].

A escolha de uma tecnologia de rede sem fio para transmissão de reforço de áudio em soluções de acessibilidade para pessoas com deficiência auditiva e visual deve basear-se em fatores como custo de implantação, regulamentação legal, tempo para implantação e performance. No presente trabalho esses critérios foram utilizados para comparar as duas principais tecnologias candidatas para prover a transmissão sem fio de áudio com baixa latência: FM e Wi-Fi (IEEE 802.11 *Wireless Local Area Network*).

O padrão IEEE 802.11 faz uso de diversos métodos de transmissão na camada física. Os padrões utilizados nesta camada são: FHSS, DSSS, IR, OFDM e MIMO [Crow et al. 1997] [Banerji e Chowdhury 2013]. O uso dessas tecnologias na camada física, atrelados aos padrões estabelecidos para a camada de enlace, fazem com que as redes Wi-Fi sejam extremamente resilientes à interferência do sinal e, consequentemente, à perda de pacotes.

Embora a transmissão por modulação em frequência seja mais eficaz em transmitir o conteúdo, por possuir menor latência, o *tradeoff* entre alto custo e *delay* é extremamente crucial neste tipo de aplicação, dessa forma, durante o desenvolvimento da solução proposta neste trabalho serão utilizadas bases *wireless* IEEE 802.11.

Além do uso das bases *wireless* é importante definir o dispositivo que será utilizado para reproduzir os conteúdos transmitidos. Como um dos focos da solução é prover mobilidade ao usuário, já que a mesma será utilizada primordialmente em cinemas, decidiu-se utilizar *smartphones* como *hardware* de *playback*. *Smartphones* modernos possuem alto poder computacional, permitindo flexibilidade durante o desenvolvimento do *software* de reprodução, assim como fácil substituição caso haja a necessidade de troca do dispositivo ou realizar um *upgrade* em algum dos aparelhos.

O grande desafio da escolha da tecnologia que deve ser selecionada na resolução do problema proposto é a latência, caracterizada como o intervalo entre o que está sendo emitido pelo locutor e o que está sendo escutado pelo cliente, pois não se deseja que haja a

ocorrência de eco ou perda de sincronização labial, o que ocasiona uma experiência desagradável ao usuário. Eventos audiovisuais que necessitam de reforço de áudio são, em geral, limitados por um fator de tempo bem definido. Na literatura, um limiar onde não se percebe a divergência na apresentação entre as trilhas de áudio e vídeo, em televisores, deve ser um delta igual ou inferior a 40 ms entre o momento em que a imagem é exibida e o áudio é reproduzido [Ravindran e Bansal 1993][R37-2007 2007]. Logo, se a transmissão do conteúdo não conseguir suprir a latência em valor máximo de 40 ms, ocorrerá um grande desconforto para o usuário, já que para deficientes auditivos o filme poderá ser exibido na frente da trilha de áudio, e para deficientes visuais a audiodescrição poderá sobrepor diálogos do filme.

Dessa forma, o escopo do trabalho aqui proposto é a transmissão de áudio com valores de latência da ordem de 40 ms, que é baixa o suficiente para atender requisitos de sincronização labial e transmissão de reforço de áudio em serviços de acessibilidade em salas de cinema.

A fim de manter a latência próxima ao valor supracitado, deve-se ajustar a solução proposta para que a mesma trabalhe da forma mais eficiente possível. Sendo assim, durante o desenvolvimento, chamadas privilegiadas junto ao *kernel* do sistema operacional do servidor tiveram que ser realizadas para ajustar o escalonamento do processo, assim como a solicitação de registro de área de memória especial com o intuito de evitar a paginação. Também é importante que se defina a frequência que será utilizada pelos roteadores, como este trabalho busca o uso primordial, em salas de cinema, optou-se pelo uso de banda 5 GHz. Essa faixa de frequência foi selecionada pois, embora possua baixo alcance, a mesma oferece maior *throughput* com menor interferência. E por fim, deve-se definir um *player* adequado para a solução que deve ser capaz de reproduzir áudio PCM com buffer reduzido

O ambiente de execução da solução proposta é baseado na captura, codificação e transmissão do áudio de um servidor para clientes executando em dispositivos móveis conectados através de uma rede sem fio. O servidor de cinema, que é a fonte de áudio, é responsável por transmitir de forma síncrona o vídeo e os canais de áudio pertencentes a um filme, esses dados são então capturados pelo servidor da solução que transmite o conteúdo acrescido de metadados através da rede *wireless* utilizando o protocolo IP *multicast*. Os clientes então recebem os dados e aplicam os algoritmos necessários para reproduzir o conteúdo da sala com atraso menor ou igual a 40 ms. Assim, trazendo uma melhor qualidade de experiência para o usuário, que não percebe eco ou *delay* entre o som da sala e o som do dispositivo

móvel.

Este trabalho propõe a criação de uma solução cliente-servidor, denominada FAST (*Fast Audio Streamer*), que gerencia a transmissão e reprodução de fluxos de áudio em tempo real com baixo retardo através do uso de redes Wi-Fi para múltiplos destinatários com foco em melhorar a experiência de usuários portadores de deficiência visual e auditiva em salas de cinema.

Em resumo, esse trabalho se propõe a responder a seguinte questão de pesquisa: Como transmitir áudio para grupos de usuários utilizando dispositivos móveis como receptores com latência entre captura e exibição igual ou inferior a 40 ms?

1.1 Objetivo

O objetivo geral deste trabalho é desenvolver uma solução de transmissão e recepção de áudio em tempo real de baixo retardo, a qual subdivide-se em dois componentes: o servidor, responsável pelo envio do conteúdo, e a aplicação cliente para dispositivos móveis, que irá reproduzir a mídia transmitida. A comunicação entre módulos será realizada através do protocolo IP e os dados serão transmitidos através de uma rede sem fio Wi-Fi.

A fim de obter sucesso no objetivo geral, faz-se necessário que os seguintes objetivos específicos sejam alcançados:

- Definir a arquitetura da solução proposta;
- Definir o protocolo de comunicação, que será utilizado para transmitir o conteúdo;
- Definir o player de áudio, que deve ser utilizado pela aplicação cliente para reproduzir o áudio;
- Desenvolver a aplicação servidor, que deve ser capaz de capturar, comprimir e transmitir segmentos de áudio;
- Desenvolver a aplicação cliente, que deve receber uma *stream* de dados de áudio e reproduzi-los;

- Definir um processo de experimentação quantitativo e qualitativo, utilizando múltiplos dispositivos em ambiente de laboratório, cujo propósito é a aferição da latência e perda de pacotes para validar a solução proposta.

1.2 Organização do Trabalho

Esse trabalho está estruturado em cinco capítulos, organizado da seguinte maneira:

O primeiro capítulo apresenta a motivação do trabalho, introduzindo conceitos relevantes como a diferença entre a operação com sinal FM e redes WLAN. Além disso, também são apresentados os objetivos do trabalho.

O segundo capítulo apresenta uma revisão sistemática da literatura sobre a transmissão de áudio em baixa latência sobre redes *wireless*.

O terceiro capítulo detalha o funcionamento do FAST, descrevendo os seus componentes e as estratégias adotadas para o seu desenvolvimento.

O quarto capítulo descreve o processo de experimentação utilizado para avaliar a solução proposta, incluindo o planejamento dos experimentos, definindo: hipóteses, variáveis e instrumentação, além da discussão dos resultados.

Por fim, no quinto capítulo são apresentadas as considerações finais, contribuições e propostas para trabalhos futuros.

Capítulo 2

Revisão Sistemática da Literatura

A revisão sistemática da literatura é um estudo secundário que busca avaliar e identificar os resultados pertinentes a um tópico de pesquisa. O método busca evitar a duplicidade de esforços e a repetição de erros cometidos previamente por outros autores [Mafra e Travassos 2006]. Para garantir que o processo seja conduzido de forma apropriada, ele deve ser realizado com o auxílio de um protocolo bem estabelecido, caso contrário, o viés dos pesquisadores pode levar a resultados pouco confiáveis, pouco abrangentes e com viés de publicação.

Neste capítulo, uma revisão sistemática da literatura será apresentada, cujo objetivo é mapear os principais estudos relacionados à transmissão de áudio em baixa latência sobre redes *wireless*, os meios de transporte utilizados, assim como as diferentes estratégias a fim de mitigar os efeitos de latência, *jitter* e perda de pacotes. Portanto, busca-se investigar as estratégias utilizadas pelos autores para transmitir e corrigir falhas em conteúdos de áudio. O processo de análise foi dividido entre o autor deste trabalho e o pesquisador Danilo Assis Nobre dos Santos Silva. A participação de mais de um pesquisador é importante, pois etapas subjetivas do processo tais como a fase de classificação dos artigos foram julgadas de forma imparcial e, desse modo, os resultados apresentados possuem maior confiabilidade.

O protocolo de revisão sistemática desenvolvido para essa revisão é apresentada na Seção 2.1. Em seguida, serão identificados os processos de busca, seleção e classificação dos artigos. Por fim, será efetuada a apresentação dos resultados e discussões, assim como as considerações finais.

2.1 Protocolo de Revisão Sistemática

Para o tema selecionado foram definidas as seguintes questões de pesquisa:

- QP1: Como estão distribuídas as publicações, elas utilizam sinal FM ou redes Wi-Fi IEEE 802.11?
- QP2: Quais as metodologias utilizadas para mitigar falhas na rede envolvendo atraso e colisão?
- QP3: Quais dispositivos são utilizados para a reprodução do conteúdo?

Em seguida, foi efetuada uma busca em portais de pesquisa com o intuito de encontrar trabalhos similares à pesquisa aqui realizada. Nesse estudo foram utilizados os seguintes portais:

- ACM Digital library [1]
- IEEEXplore [2]

A fim de realizar o processo de busca automática é necessário definir uma "*string* de busca", o que requer que as palavras chaves da pesquisa estejam bem definidas. Esses termos encontram-se na Tabela 2.1, junto a *string* de busca:

- (audio OR music OR speech) AND (stream* OR transmitter OR transmission) AND (wireless OR wi-fi OR Wi-Fi) AND (low-delay OR "low delay" OR delay OR latency OR "low latency") AND (smartphone OR mobile OR tablet)

Após a realização do processo de busca, os documentos foram filtrados com base nos critérios de inclusão (I) e exclusão (E):

- E1: Excluir trabalhos repetidos, preservando apenas o mais recente;
- E2: Excluir todos os documentos que não são artigos científicos;

[1] `https://dl.acm.org/`
[2] `https://ieeexplore.ieee.org/`

Tabela 2.1: Palavras-chave e sinônimos da *string* de busca

Palavras-chave	**Sinônimos**
audio; streamer; low latency; mobile; wi-fi	music; speech; streamer; transmitter; transmission; wireless; Wi-Fi; low delay smartphone; tablet

- E3: Excluir trabalhos que não possam ser classificados de acordo com a taxonomia definida na revisão sistemática;
- I1: Incluir todos os artigos completos relacionados ao contexto da revisão (através da leitura do título e do resumo e, em caso de informações insuficientes, da leitura completa do artigo);

Após o processo de inclusão e exclusão com base nos critérios estabelecidos pelo protocolo definido na Seção 2.1, realizou-se a classificação a partir das seguintes regras:

1. **Estratégia de transmissão**: Identifica o meio de transmissão utilizado pelo autor do trabalho para encaminhar o áudio. Nesse caso, as redes mais utilizadas são:
   (a) WPAN: Redes de acesso pessoal;
   (b) WLAN: Redes de acesso local;
   (c) Ad hoc: Redes em que os nós participam na transmissão do conteúdo;
   (d) Bluetooth: Protocolo de comunicação que visa baixo consumo de recursos;
   (e) Sem classificação: O autor não identifica em seu trabalho;
2. **Estratégia de Correção**: Identifica a estratégia utilizada para mitigar os erros advindos da perda de pacotes. Os tipos mais comumente utilizados são:
   (a) FEC: Técnica utilizada para corrigir a perda de informação em meios de distribuição que apresentam grande instabilidade;
   (b) Redundância temporal: Técnica utilizada para recriar trechos de áudio perdidos no processo de envio;

   (c) Algoritmo adaptativo: Técnica utilizada para mitigar os efeitos de *jitter* em redes com grande instabilidade;

   (d) Sem classificação: O autor não define a técnica utilizada;

3. **Dispositivo de Reprodução**: Identifica o dispositivo utilizado pelo autor para reproduzir o conteúdo transmitido. Dentre os mais utilizados:

   (a) Computador: *Desktop*;

   (b) Dispositivo móvel: *Palmtops*, *laptops*, *smartphones* e *tablets*

   (c) Sem classificação: O autor não cita o dispositivo utilizado para reprodução;

4. **Reprodutibilidade do Experimento**: Identifica se o autor documentou o seu experimento tornando-o reprodutível. Nesse caso, o experimento pode ser classificado como:

   (a) Reprodutível: O autor documenta os experimentos, possibilitando que o mesmo seja replicado;

   (b) Não reprodutível: O autor não documenta seus experimentos;

2.2 Processo de Busca, Seleção e Classificação das publicações

Através da aplicação da *string* de busca nos portais de artigos definidos no protocolo especificado neste trabalho, realizada no dia 10 de Março de 2020, foram encontrados 374 artigos, distribuídos de acordo com a Tabela 2.2.

Em seguida, um processo de seleção foi efetuado junto aos trabalhos selecionados em duas iterações. Na primeira iteração, os artigos foram selecionados com base nos critérios de inclusão e exclusão, considerando-se título, resumo e leitura completa do artigo quando necessário. Na segunda iteração, os artigos considerados indeterminados foram novamente analisados e excluídos caso não haja senso comum acerca da classificação.

Uma grande quantidade de artigos foi removida na primeira etapa por fazerem uso de estratégias de mitigação de perda de dados nos protocolos das bases *wireless*. Como o objetivo desse trabalho é a criação de uma solução em *software*, esses trabalhos foram descartados.

Tabela 2.2: Resultado da aplicação da *string* de busca nos portais de pesquisa

	IEEE	**ACM**	**Total**
Processo de Busca	207	167	374
Seleção (1a iteração)	6	11	17
Seleção (2a iteração)	4	8	12

Além disso, muitos trabalhos foram encontrados em ambas as plataformas de busca, o que resultou em grande duplicidade de publicações. Também foram encontrados diversos documentos técnicos e propostas de trabalho que foram desconsiderados, conforme o critério de exclusão E2.

Após o processo de seleção, os artigos foram lidos integralmente pelos pesquisadores e classificados com base nos critérios definidos na Seção 2.1. O resultado do processo de classificação é apresentado na Tabela 2.3.

Tabela 2.3: Classificação dos trabalhos encontrados

Ref.	**Estratégia de Transmissão**	**Estratégia de Correção**	**Dispositivo de Reprodução**	**Reprodutibilidade do Experimento**
Mckinley 2000	WLAN	FEC	Dispositivos móveis	Reprodutível
Doherty 2013	WLAN	FEC	Sem classificação	Reprodutível
Doherty 2015	WLAN	FEC	Sem classificação	Reprodutível
Lin 2004	Ad hoc	Redundância temporal	Sem classificação	Reprodutível
Raju 2006	Ad hoc	FEC	Sem classificação	Reprodutível
Jelassi 2006	Ad hoc	Algoritmo Adaptativo	Sem classificação	Reprodutível
Zhang 2008	Ad hoc	Algoritmo Adaptativo	Computador	Reprodutível
Deshpande 2006	WPAN	Algoritmo Adaptativo	Dispositivos móveis	Reprodutível
Chen 2004	Bluetooth	Algoritmo Adaptativo	Computador	Reprodutível
Leslie 2001	Sem classificação	Redundância temporal	Sem classificação	Não reprodutível
Hardman 1998	Sem classificação	Redundância temporal	Computador	Não reprodutível
Wang 2003	Sem classificação	FEC	Computador	Reprodutível

Conforme observado na Figura 2.1, os trabalhos estão distribuídos no período de 1998 a 2015, com o maior número de publicações em 2006, concentrando 25% das pesquisas efetuadas sobre o tema deste trabalho. Ainda é possível observar que entre os anos de 2003 e

2008 há o maior número de publicações, totalizando 7 trabalhos, ou seja, 58.33% dos artigos avaliados.

Figura 2.1: Trabalhos distribuídos por ano de publicação

Com relação às estratégia de transmissão, conforme a Figura 2.2, 75% dos trabalhos utilizam algum mecanismo de transmissão *wireless*, e, dentre esse percentual, 33.3% utilizam de redes Ad hoc.

Figura 2.2: Distribuição das publicações por estratégia de transmissão

No que se refere às estratégias de correção, 41.70% dos artigos utilizam *Forward Error*

Correction como uma de suas abordagens, enquanto 33.30% dos trabalhos lidam com algoritmos adaptativos, como pode ser observado na Figura 2.3. O número alto de trabalhos que utilizam FEC é espelhado na efetividade da estratégia na mitigação de perdas.

Figura 2.3: Distribuição das publicações por estratégia de correção

As publicações também utilizam diferentes dispositivos para realizar a reprodução do conteúdo transmitido e essa distribuição é ilustrada na Figura 2.4. Os trabalhos foram divididos entre aplicações que usam dispositivos *desktop* conectados diretamente a um *switch*/roteador, que totalizam 33.30% dos *devices*; dispositivos móveis como *smartphones*, *tablets*, *palmtops* e *laptops*, que comunicam-se utilizando antenas *wireless* que compõe 16.70% dos meios; e os dispositivos sem classificação, que compõem o índice majoritário de 50.00%. A categoria "Sem classificação" é utilizada quando o autor não diz explicitamente o tipo de máquina utilizada em seus testes, que, na maioria dos casos, assume-se que é um computador de mesa. Entretanto, os pesquisadores que realizaram essa revisão sistemática decidiram não classificar esses trabalhos para o citado caso como "Sem classificação", evitando assim suposições.

Figura 2.4: Distribuição das publicações por dispositivo de reprodução

A partir da documentação dos testes, foi possível agrupar os artigos com base na sua reprodutibilidade. Um artigo é qualificado como reprodutível quando ele fornece dados suficientes para que o seu experimento possa ser replicado em outro ambiente. Dentre os trabalhos analisados, 83.30% descrevem com fidedignidade o ambiente de testes, citando os dispositivos e tecnologias utilizadas, conforme observado na Figura 2.5.

Figura 2.5: Reprodutibilidade dos trabalhos

2.3 Avaliação Qualitativa das Publicações

Nessa seção será realizada uma análise qualitativa dos artigos classificados na Tabela 2.2. Dentre os artigos analisados, dois deles (16.70%) não descrevem os seus experimentos, o que inviabiliza a obtenção de conclusões sobre os seus resultados. Dessa forma, somente os dez artigos (83.30%) que possuem uma documentação clara sobre os seus experimentos serão avaliados qualitativamente. Com o objetivo de organizar a apresentação dos trabalhos, a análise será dividida em subseções com base nas estratégias de correção definidas na Seção 2.2.

2.3.1 Estratégia de correção: FEC

Experimental Evaluation of Forward Error Correction on Multicast Audio Streams in Wireless LANs

Os autores [McKinley e Gaurav 2000] propuseram um estudo que busca utilizar um serviço de *proxy* para melhorar o envio de fluxos de áudio sobre protocolo *multicast*.

A solução chamada Pavilion utiliza um algoritmo de FEC de domínio público desenvolvido pela Rizzo. O ambiente de testes engloba diversos dispositivos: computadores de mesa, *notebooks* e *palmtops*, conectados em rede *wireless* IEEE 802.11 sobre protocolo IP *multicast*. Os testes efetuados por McKinley foram realizados em laboratório com três variáveis independentes: distância entre fonte e dispositivo, *bitrate* da *stream* e níveis de correção. Dessa forma, o autor busca analisar as seguintes variáveis dependentes: perda de pacotes e latência, em testes laboratoriais.

É interessante ressaltar o uso de dispositivos físicos ao invés de simulações, pois mais variáveis externas são inseridas durante os testes, como a variação da potência da antena de cada dispositivo, a capacidade de processamento de cada máquina, o sistema operacional, dentre outros. Estes fatores são extremamente desejáveis pois possibilita que os testes laboratoriais aproximem-se de um cenário real.

A Self-Similarity Approach to Repairing Large Dropouts of Streamed Music

A pesquisa de [Doherty, Curran e Mckevitt 2013] propõe um novo método para correção de perda de pacotes em *streams* de música, o SoFI (*Song Form Intelligence*), definido como

o algoritmo desenvolvido na pesquisa capaz de entender os padrões do espectro da trilha e reconstruir partes da música baseando-se nesses padrões.

O sistema utiliza um pré-processamento para gerar "arquivos de similaridade", que são salvos em um banco de dados e transmitidos em paralelo com a trilha principal. Para gerar o arquivo de similaridade, diversos algoritmos são utilizados para analisar o espectro da música, que são o *Audio Spectrum Envelope* (ASE), o *Audio Spectrum Flatness* (ASF), o *Audio Spectrum Basis/Projection*, o *Audio Spectrum Centroid* (ASC) e o *k-means clustering*.

Para testar o seu algoritmo de FEC, Doherty utilizou 8 amostras de áudio com diferentes reparos, dentre eles, *linear interpolation*, *down sampling*, *jitter*, reparos com FEC e o áudio de controle. Dentre as amostras, 3 apresentavam falhas, 1 era de controle e 4 foram corrigidas utilizando SoFI. O experimento executado através de uma pesquisa, em que as amostras de áudio são apresentadas e cada participante deve avaliá-las com um valor entre 1 e 8, em que valores maiores identificam um áudio de melhor qualidade. Dentre os testes, SoFI obteve resultados 200% mais satisfatórios, em média, do que as demais abordagens. É importante mencionar que os testes foram realizados em simulações e a transmissão *broadcast* citada pelo autor não é utilizada em nenhum momento, o que faz com que os resultados não se assemelhem a um cenário real de *streaming*.

Pattern Matching Techniques for Replacing Missing Sections of Audio Streamed across Wireless Networks

Em seu mais recente trabalho, [Doherty, Curran e McKevitt 2015] revisita SoFI, realizando análises objetivas e subjetivas sobre o seu funcionamento através de simulações quando comparado com diferentes abordagens de reparo e de sobrescrita de pacotes em caso de falhas ou perda de dados. Novamente é possível perceber que sua abordagem não envolve aplicações de tempo real, uma vez que a sua solução busca reverter *dropouts* de tamanho considerável, entre 15 e 20 segundos, fazendo-se necessário um *delay* superior aos 20 segundos para que a transmissão ocorra com o mínimo de interrupções.

Embora o trabalho traga bastante informação sobre as diferentes formas de analisar frequência e uma visão superficial sobre problemas de *jitter*, esse trabalho não traz novas contribuições por ser um complemento do trabalho discutido anteriormente.

On Supporting Real-time Speech over Ad hoc Wireless Networks

Em seu trabalho, [Raju et al. 2006] busca uma estratégia que seja factível suportar comunicação por voz em redes ad hoc. Para alcançar o seu objetivo, o autor utiliza a estratégia de FEC, assim como transmissão através de caminhos distintos, cujo propósito é a produção de uma heurística capaz de transmitir os dados com o mínimo de perdas e com o *delay* abaixo de 200 ms para que a comunicação entre os usuários não seja comprometida.

Assim como a maioria dos trabalhos que fazem uso de redes ad hoc, o trabalho de Raju utiliza um simulador, cujo nome é NS-2 *Network Simulator*. Dessa forma, diversos parâmetros podem ser simulados, dentre eles, a área em metros quadrados em que a rede será inserida, o total de clientes na rede, o protocolo, o tipo de tráfego, dentre outros. O autor então demonstra a efetividade da sua implementação quando comparado a outras duas soluções: *Layered scheme* e *MD Scheme*. Os valores de *delay* são mantidos em 200 ms enquanto *frame loss* é aferido, *buffer size* e *PESQ score*.

Enquanto o seu método intitulado *Packetization scheme* é de fato mais eficiente do que os dois esquemas utilizadas para comparação, o seu *delay* ainda é consideravelmente alto quando comparado a estratégias que pretendem ficar abaixo de *delay* de *lip-sync* que é de 40 ms [Ravindran e Bansal 1993].

Content-based UEP: A new Scheme for Packet Loss Recovery in Music Streaming

O trabalho de [Wang et al. 2003] propõe um novo algoritmo de proteção contra erros em camadas voláteis, como redes *wireless*, intitulado *content-based unequal error protection* - C-UEP. O algoritmo proposto busca reduzir a redundância de dados imposta por algoritmos de FEC e o *overhead* na retransmissão da rede.

Os testes realizados pelos pesquisadores foram mesclados com testes informais efetuados pelos autores, comparando o C-UEP com outras ferramentas de recuperação e testes realizados com usuários do sistema. Porém, a quantidade de participantes nestes testes não é mencionada no artigo.

2.3.2 Estratégia de correção: Redundância temporal

Supporting Real-time Speech on Wireless Ad Hoc Networks: Inter-packet Redundancy, Path Diversity, and Multiple Description Coding

A pesquisa de [Lin et al. 2004] trabalha com o envio de diálogo sobre redes ad hoc e o foco da sua pesquisa é eliminar a retransmissão em camada MAC com o propósito de reduzir tanto o *delay* ponto-a-ponto quando a quantidade de dados trafegados sobre a rede.

Para alcançar o seu objetivo, o autor trabalha com a redundância de pacotes. Como a quantidade de bytes necessários para transmitir diálogo é muito baixa, agrupar diversos *frames* é interessante para reduzir a colisão, especialmente em redes ad hoc em que a quantidade de dados transmitidos aumenta conforme mais clientes são inseridos na rede.

Em seus testes, Lin realiza uma comparação direta entre sua abordagem de *inter-packet redundancy* e a retransmissão em camada MAC do padrão IEEE 802.11. A abordagem do autor transmite cada novo pacote acrescido do pacote transmitido anteriormente. Além disso, o algoritmo proposto também explora o uso de *Multiple Description Coding* (MD) para auxiliar na correção de pacotes corrompidos por *bursty loss*. Assim como o trabalho de Raju et al. (2006), Lin et al. (2004) utiliza simulações para criar sua topologia de rede. Por fim, o autor percebe bons resultados em suas simulações, indicando que o uso de *inter-packet redundancy* auxiliado por MD pode ser mais eficiente do que a retransmissão em camada MAC.

2.3.3 Estratégia de correção: Algoritmo adaptativo

Adaptive Playback Algorithm for Interactive Audio Streaming Over Wireless Ad-Hoc Networks

Em sua pesquisa [Jelassi e Youssef 2006] propõe um novo algoritmo de *playback* intitulado *Periodic Adaptive Algorithm* (PAA), que busca adaptar a latência de reprodução periodicamente a partir de observações na variação em redes ad-hoc.

O algoritmo proposto busca manter a latência em uma taxa constante de 400 ms, e, para isso, a solução avalia a diferença entre o *delay* de *playback* e o *delay* da rede. Dessa forma, a partir dessa divergência opera-se o *buffer* acumulando ou reduzindo o número de amostras

de áudio.

Para validar a solução, os pesquisadores utilizaram um ambiente simulado através do sistema *network simulator* ns2. A simulação conta com 50 nós distribuídos pela rede, em que dois são selecionados aleatoriamente para serem emissor e receptor. Para melhorar o seu algoritmo, o autor utiliza o indicador MOS - *Mean Opinion Scores* - com pesos em latência média dos pacotes transmitidos e taxa de perda de pacotes. Por fim, o autor compara o seu algoritmo com outros dois intitulados: *I-Policy* e *E-Policy*, e através de observações na latência de reprodução a conclusão foi que a sua implementação atinge um maior índice de satisfação com base em MOS *metric*, em que a PAA obtém 78% de satisfação enquanto *E-Policy* ontem 59% e *I-Policy* 0%. Além disso, a solução lida com problemas de latência com simplicidade e com baixa complexidade computacional, que é um requisito desejado quando são utilizados dispositivos móveis.

Adaptive Low-Bitrate Streaming Over IEEE 802.15.4 Low Rate Wireless Personal Area Networks (LP-WPAN) Based On Link Quality Indications

Em seu trabalho, [Deshpande 2006] busca investigar a viabilidade do uso do padrão IEEE 802.15.3 LR-WPAN para transmissão de áudio e vídeo com baixo bit rate.

O autor inicialmente cria uma topologia de rede no software de simulação NS2, em que dois nós comunicam-se transmitindo áudio e outros dois nós transmitem quaisquer informações com o propósito de perturbar o sinal que está sendo transmitido. Em seguida, o uso do protocolo de aceitação em camada MAC é avaliado, constatando que a não ativação do protocolo faz com que a porcentagem de pacotes perdidos assim como a latência dos pacotes transmitidos seja reduzida consideravelmente.

Além das avaliações relativas a *stream* do conteúdo, o autor também propõe um algoritmo adaptativo que faz uso de *Link Quality Indicatio*n (LQI). A heurística é feita de forma distribuída, ou seja, inicialmente o cliente transmite o valor de LQI ao servidor, que refere-se ao pacote que acabou de ser recebido. Em seguida, o servidor analisa o valor e o compara com um *threshold* e avalia o melhor bit rate a ser transmitido pela *stream*. O autor novamente utiliza o NS2 para simular a transmissão e, como resultado, percebe-se uma melhora significativa na porcentagem de perda de pacotes que cai de 6.28% para 0.07%.

Audio Streaming over Bluetooth: An Adaptive ARQ Timeout Approach

A pesquisa de [Chen et al. 2004] propõe um algoritmo adaptativo que ajusta o mecanismo de retransmissão *Automatic Repeat Request* (ARQ) em redes *Bluetooth*. Segundo o autor, a maioria dos meios de transmissão *wireless* possuem um tipo de implementação de ARQ, embora essa seja uma *feature* desejável nesse tipo de ambiente, ela pode acabar piorando o *delay* na transmissão uma vez que várias retransmissões são enviadas na rede.

A proposta de Chen é usar o *round trip time* (RTT) de cada pacote de áudio para ajustar a janela de retransmissão do protocolo *Bluetooth* que, a priori, é infinito. Desse modo, ao limitar a retransmissão para um intervalo de tempo bem definido a antena poderá novamente ser utilizada para a transmissão de novos pacotes.

Os testes foram executados em dois *laptops*, conectados através de placas *Bluetooth* com protocolo IEEE 802.11b, os quais utilizam a API Bluez, que é uma stack open source para transmissão e captura de dados *bluetooth*. Ainda no *setup*, os pesquisadores espalharam dispositivos que operam na mesma frequência de 2.4GHz para gerar interferência no sinal, trazendo maior fidedignidade aos testes. Os pesquisadores então compararam a sua solução com intervalos fixos de *delay* e, dessa forma, foi possível observar uma melhora tanto no *delay* médio quanto na taxa de perda de pacotes.

2.4 Considerações Finais

Nessa seção foram avaliados os trabalhos que fazem parte do processo revisão sistemática desenvolvido sobre a transmissão de áudio em baixa latência sobre redes *wireless*. Para auxiliar a pesquisa, foi desenvolvido um protocolo de revisão sistemática, cujo intuito é a pesquisa e classificação dos trabalhos selecionados. Por fim, uma avaliação foi realizada a fim de descrevê-los com base nas suas contribuições.

Durante a avaliação qualitativa, foi possível perceber que uma parte majoritária dos trabalhos lida com a transmissão de áudio sobre redes ad hoc, e essa escolha é justificada pela tecnologia, que é bastante promissora pela sua facilidade de uso e custo reduzido. Os trabalhos também indicam o uso de FEC e algoritmos adaptativos como as estratégias mais eficientes em reduzir perda de pacotes e latência, totalizando 75% dos trabalhos selecionados. Dentre os trabalhos selecionados, apenas 16.70% utilizam dispositivos móveis para

efetuar os testes, e esse número reduzido pode ser justificado pelo ano das publicações, uma vez que em 2006, ano em que foram publicados a maioria dos artigos selecionados, não existiam celulares com alta capacidade de processamento. Embora os trabalhos tenham o mesmo foco, que é a redução da latência e perda de pacotes, nenhum deles trabalha com o critério de latência próxima a *lip sync*, definida como a diferença temporal entre o que é enviado pela fonte de áudio e o que é recebido pelo dispositivo de reprodução, que deve ser igual ou inferior a 40 milissegundos [Ravindran e Bansal 1993]. Em sua maioria, os trabalhos lidam com latência para VOIP, que oscila próximo aos 200 milissegundos.

A baixa quantidade de publicações que lidam com redes WLAN e transmissão para dispositivos móveis, assim como o valor de latência bastante acima do valor de *lip sync*, motivaram o desenvolvimento da solução proposta que será discutido no próximo capítulo.

Capítulo 3

FAST - *Fast Audio Streamer*

Neste capítulo, será apresentada a arquitetura e o funcionamento do FAST. Como descrito no Capítulo 1, o FAST busca desenvolver um serviço de transmissão de áudio em tempo real, que será utilizado para aprimorar a experiência de usuários em aplicações em que a reprodução do conteúdo em tempo real e o retardo de transmissão são de extrema importância.

Conforme observado no Capítulo 2, uma porção bastante pequena das publicações (16.70%) lida com o uso de dispositivos móveis e, dentre esses trabalhos, nenhum deles cita o uso de *smartphones*, apenas *palmtops* e *laptops*. O uso desse tipo de dispositivo é bastante desafiador, uma vez que o hardware possui limitações quando comparado aos tradicionais *desktops*.

Com o propósito de evitar problemas de *lip sync* durante a apresentação do conteúdo, a proposta descrita também limita a latência máxima a 40 milissegundos. Conforme observado anteriormente, os trabalhos que possuem algum limiar são os que lidam com VOIP, definindo um *threshold* de 200 milissegundos, ou seja, este trabalho lida com um desafio 5 vezes maior. A proposta também busca evitar perdas de pacotes, e, para isso, foram utilizados o FEC, redundância temporal e um algoritmo adaptativo otimizado para mitigar as variações no meio *wireless*.

Após análise extensa sobre os *codecs* do mercado, optou-se pelo uso do Opus pela grande adesão da comunidade por apresentar licença *open source*, excelente qualidade de compressão e alta eficiência quando comparado aos demais *codecs* para *streaming* [Opus Codec, 2012]. Além da alta qualidade do sinal e da eficiência ao comprimir a informação, o *codec*

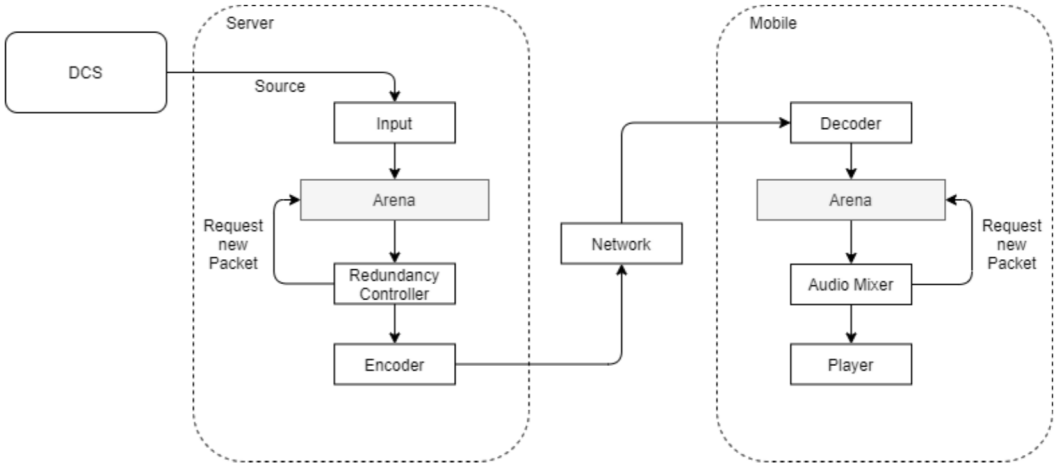
também permite transmitir uma quantidade maior de dados pela rede sem que a informação seja fragmentada em múltiplos *chunks*. O *codec* ainda auxilia a implementação, fornecendo em sua API o acesso à implementação de um algoritmo de FEC.

Com o intuito de preservar o máximo de qualidade na reprodução, é utilizada a redundância temporal. Dessa forma, o FEC é utilizado em última circunstância, uma vez que essa abordagem causa degradação da informação. É visto na literatura que a redundância temporal também é bastante efetiva mesmo possuindo um *tradeoff* entre eficiência e banda utilizada.

Por fim, o FAST utiliza um algoritmo adaptativo, que busca reduzir os efeitos causados pelo *jitter* da rede. A solução desenvolvida para a aplicação móvel monitora a entrada de *chunks* e ajusta a taxa de amostragem do áudio conforme o *buffer* da aplicação é esvaziado ou preenchido. Dessa forma, a chance de falha na sincronização labial, assim como o esvaziamento do *buffer* (*underflow*) podem ser mitigados.

A Figura 3.1 apresenta uma visão esquemática da solução proposta, descrevendo os módulos que fazem parte do funcionamento do servidor e do dispositivo de reprodução da solução.

Figura 3.1: Arquitetura da solução proposta



Inicialmente, no servidor, o módulo *Input* recebe um sinal de áudio digital através da placa de áudio *onboard* do computador, responsável pela captura de *samples* do áudio original a cada 4 milissegundos.

O sinal consumido é então enviado a uma região de memória contínua, que funciona

como um *buffer* entre a captura e o processamento do áudio, denominada *Arena*. Este segmento de memória RAM é alocado de forma sequencial para que se possa garantir o acesso rápido aos dados, uma vez que grande quantidade de informação estará na região de *cache* do processador. Além disso, manter o *buffer* em uma sequência contínua evita que a região sofra paginação, assim, evitando que o *kernel* mova informação para o disco, o que pode comprometer o acesso à informação em uma aplicação com restrições severas de processamento.

Os *samples* de áudio, que são fragmentos estéreo do áudio original, capturados a cada 4 milissegundos, são agrupados pelo módulo *Redundancy Controller* para formar uma *chunk* de 20 milissegundos, ou seja, 5 capturas do áudio original formam um segmento que pode ser comprimido e futuramente transmitido. Dessa forma, nessa solução, o segmento de áudio sempre irá conduzir um *frame* de áudio síncrono com o fluxo de dados de entrada e 4 *frames* atrasados que funcionam como redundância no envio.

O *chunk* de 20 ms criado pelo módulo de redundâncias é então comprimido pelo Compressor, que realiza chamadas junto a API do *codec* Opus que, além de comprimir o áudio, também insere metadados junto ao pacote de rede para que o sinal possa ser reconstruído pelo algoritmo de FEC caso algum pacote de rede seja perdido.

O pacote é então transmitido na rede por protocolo IP *multicast* e é recebido por um módulo de rede nativo no dispositivo móvel, que realiza o repasse do pacote de rede com o *chunk* de 20 ms de áudio comprimido para o *Extractor*. Esse módulo então realiza a descompressão da informação ou utiliza o algoritmo de FEC para reconstruir um pacote que não foi recebido. Dessa forma, voltando a possuir os 5 *samples* de áudio digital, que são novamente inseridos em uma *Arena*. Assim, o *Audio Mixer* pode iniciar o consumo dos dados que foram transmitidos do servidor, preparando-os para conversão e posteriormente *playback*.

O *mixer* é responsável por executar o algoritmo adaptativo com base na quantidade de dados disponível no *buffer*, podendo aumentar o tamanho dos *samples* para aguardar a chegada de mais dados, ou encurtá-los para manter os dados no limiar de 40 ms. O processo de aumentar ou contrair um *sample* de áudio consiste em realizar *upsampling* e *downsampling* da informação, ou seja, o módulo aumenta ou reduz, artificialmente, a taxa de amostragem de um fragmento de áudio. Dessa forma, o *buffer* possui maiores chances de não ser drenado

completamente enquanto a solução aguarda novos dados.

Por fim, os dados são enviados para o Player, que gerencia o envio do áudio PCM para o hardware de *playback* do dispositivo móvel, convertendo o sinal através de um DAC e enviando o áudio analógico de forma apropriada para os fones de ouvido do usuário.

Maiores detalhes sobre os componentes da solução são apresentados nas próximas seções.

3.1 Entrada de Áudio

O componente *Input* é responsável pela captura de dados a partir de uma fonte de áudio, e, para isso, a solução utiliza a API da biblioteca ALSA[1] do sistema operacional GNU/Linux[2]. Essa API, que faz parte do *kernel* do sistema operacional, é dedicada à manipulação de áudio em nível de *hardware*, contendo a implementação de diversos *drivers* para placas de áudio de diversos fornecedores.

A solução proposta utiliza uma placa de áudio genérica, que captura os dados a uma taxa constante com *sampling* rate de 48 kHz, *bit depth* de 16 *bits* e dois canais por sample.

Os dados são capturados constantemente a cada 4 ms e enviados ao módulo de criação de redundância para que um conjunto de *samples* de áudio possa gerar um *chunk*, que é posteriormente enviado para o dispositivo de reprodução.

A solução desenvolvida é genérica e pode utilizar diversas entradas de áudio com diferentes taxas de amostragens e precisão de amostras (*bit depth*). Entretanto, cabe ressaltar que esses valores devem ser compatíveis com o *encoder* que será utilizado para a compressão de dados.

Para este trabalho, a fonte de áudio utilizada foi um fluxo proveninte de um filme transmitida pelo DCS (Digital Cinema Server) IMS 2000[3] provido pela Dolby Labs.

[1] https://alsa-project.org/

[2] https://www.linux.org/

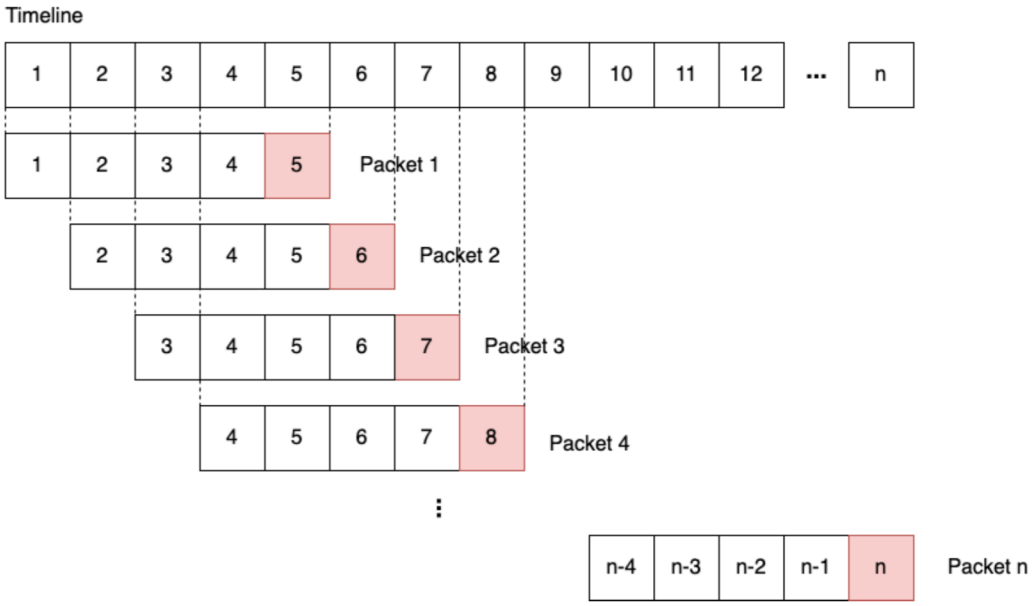
[3] https://centauro-cinema.com.br/2018/produto/ims2000/

3.2 Controlador de Redundância Temporal

Após a captura dos dados, é possível iniciar o agrupamento dos *samples* do áudio de entrada, cujo intuito é a formação de um *chunk* de 20 ms, que será então comprimido e transmitido pela rede. O módulo de criação de redundâncias agrupa 5 *samples* de 4 milissegundos e envia o *chunk* de 20 milissegundos resultante para o *encoder*. Esse valor é definido com base nos requisitos do *codec* de áudio. Segundo a sua documentação, o meio mais eficaz de comprimir dados é fornecer *chunks* com 20 milissegundos, assim, habilitando os algoritmos de compressão de voz e de faixas de áudio simultaneamente [Valin e Terriberry 2012].

Conforme observado na Figura 3.2, na parte superior da imagem é apresentada a linha do tempo do áudio dos *samples* de 1 até *n*, cada uma possuindo 4 ms. O segmento de áudio é então formado com 5 *samples* onde a mais à direita, representada com a cor vermelha, é o segmento de áudio capturado mais recentemente e, portanto, mais sincronizado com a fonte que, consequentemente, deverá ser reproduzido primeiro.

Figura 3.2: Pacotes em relação à *timeline*



3.3 *Encoder e Decoder*

Nessa seção serão apresentados detalhes sobre o funcionamento dos módulos *encoder* e *decoder*. Além disso, são apresentadas na Subseção 3.3.3 as justificativas para o uso do

codec Opus, adotado nessa solução.

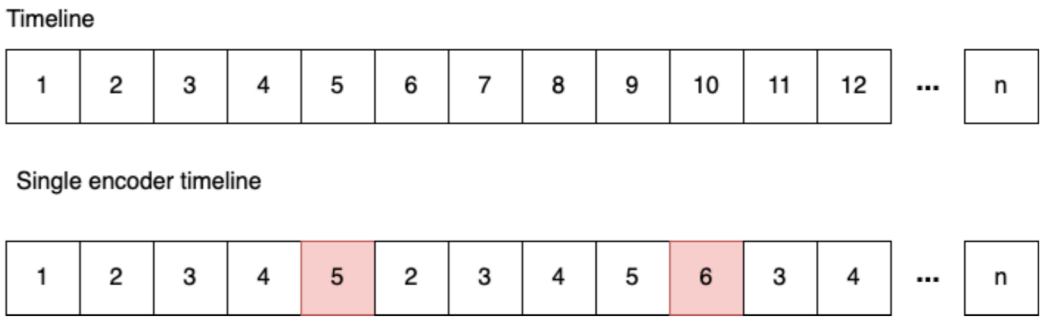
3.3.1 *Encoder*

No FAST, a compressão do conteúdo é realizada assim que o *chunk* de 20 ms de áudio é formado. Como o Opus é tanto utilizado para compressão quanto para a aplicação do FEC, o uso da estratégia de redundância não pode ser utilizado com a mesma instância do *encoder* ou *decoder*, pois ruídos são inseridos na reconstrução do áudio.

A Figura 3.3 apresenta a *timeline* dos *frames* de áudio de 1 até *n*, e em seguida é exibida a *timeline* para um único *encoder*. Como pode ser observado, informações previamente processadas são novamente inseridas no *encoder*. Na imagem, são exemplificados dois pacotes de rede com cinco *samples*, o primeiro com dados de 1 até 5 e o segundo com dados de 2 até 6. Como observado, os samples 2, 3, 4 e 5 são reinseridos no pacote seguinte, e, dessa forma, ao gerar os metadados para a reconstrução de áudio o algoritmo de FEC acaba não entendendo a forma apropriada de interpolar os dados. Essas repetições causam ruídos quando o áudio é reconstruído por FEC no lado do *decoder*. Ao escutar a trilha de saída, dois resultados podem ser percebidos:

1. O áudio apresenta ruídos na união entre o segmento decodificado com e sem FEC;
2. O áudio apresenta ruídos na união entre os segmentos de áudio sem tratamento de FEC e com tratamento, exclusivamente em frequências graves;

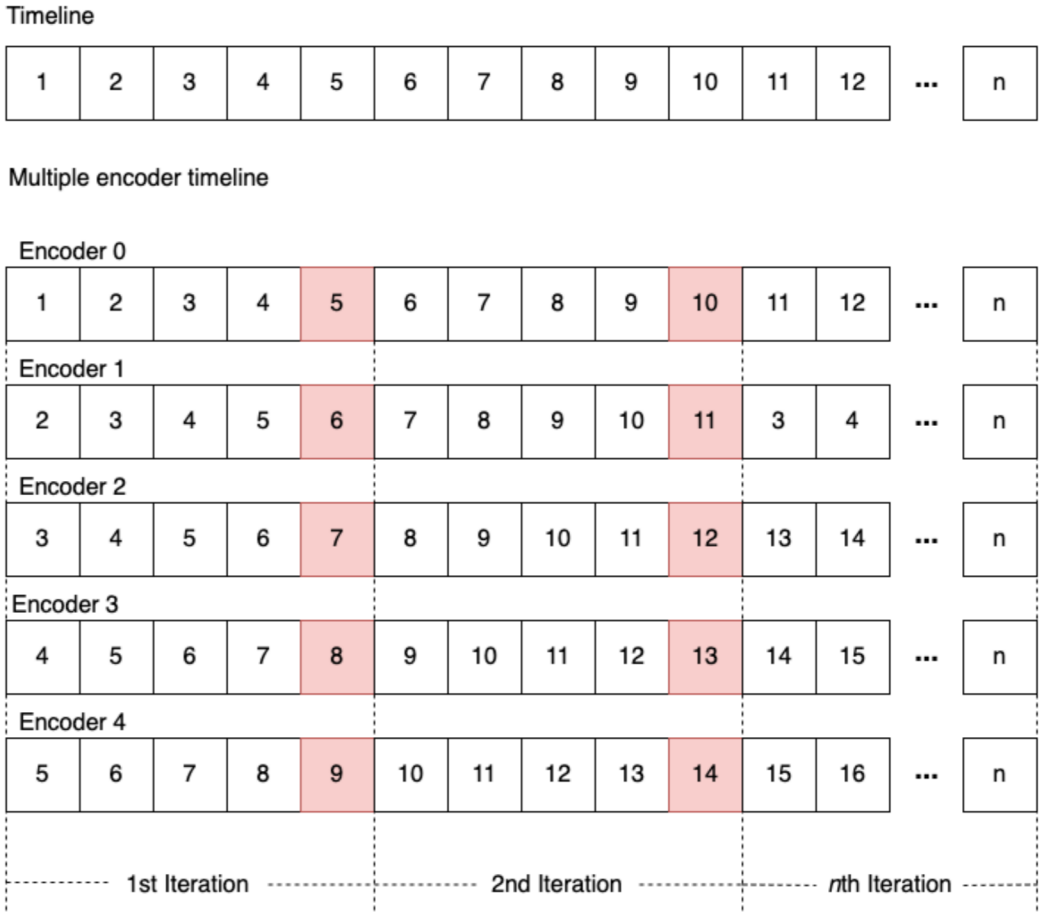
Figura 3.3: *Timeline* do áudio para um compressor



Buscando evitar os ruídos, foi adotada uma abordagem que utiliza múltiplos *encoders* para compressão de dados a partir da *timeline* original, exemplificada pela Figura 3.4. Os

samples em vermelho são os segmentos de áudio que devem ser, idealmente, reproduzidos pelo cliente, enquanto os samples em branco são redundâncias de samples já transmitidos. É importante ressaltar que a quantidade de *encoders* e *decoders* é justificada pela razão entre o *chunk* de áudio usado no *encoder* e o tamanho de um *sample*, que é a menor unidade de áudio. Na solução proposta tem-se 20 ms de áudio a ser comprimido dividido por 4 ms, ou seja, o tamanho de um *sample*, totalizando os 5 *encoders* exibidos na imagem. É importante ressaltar que o número de *encoders* muda dependendo da quantidade de dados replicados, sendo assim, se os *samples* fossem capturados a cada 2 ms, e o pacote de rede se mantivesse em 20 ms, calcular-se-á a necessidade de um total de 10 *encoders*. Ou seja, deve existir um *encoder* para cada sample a fim de se ter uma *timeline* sem repetições do sinal. Ainda deve-se salientar que os *encoders* são executados sequencialmente, e isso não aumenta significativamente o número de instruções em CPU, porém aumenta a quantidade de memória alocada para manter o contexto dos *encoders*.

Figura 3.4: Uso de múltiplos *encoders* para eliminar artefatos no áudio



Ainda na Figura 3.4, percebe-se que ao comparar a *timeline* de um dos *encoders* com a *timeline* da implementação com apenas um *encoder* da Figura 3.3, os *samples* não se repetem, uma vez que os segmentos são distribuídos de forma intercalada entre os *encoders*.

A implementação com um *encoder* comprime consecutivamente informação referente à amostra de número 5, seguido das amostras 6, 7, 8, ..., n. Como discutido anteriormente, a inserção do sinal repetidas vezes é o causador dos ruídos percebidos no áudio reproduzido. Já na abordagem com múltiplos *encoders*, divide-se os *samples* entre os *encoders* com o intuito de representar uma *timeline* contínua.

Com o objetivo de evitar os ruídos, na primeira iteração, o *encoder* zero comprime o *frame* de áudio referente ao *sample* de número 5, o *encoder* um realiza a compressão do *sample* 6, e assim sucessivamente. Na iteração seguinte, os valores observados por cada *encoder* são uma continuação do *sample* anterior, logo, ao invés do *encoder* zero receber

samples 5, 6, 7, . . . , *n*, ele recebe *samples*, 5, 10, 15, . . . , *n*+5. Dessa forma, cada *encoder* é responsável por uma sequência de áudio sem repetições e, consequentemente, sem falhas.

Para identificar o *encoder* responsável pelo *sample* analisado, uma simples operação de resto de divisão entre identificador e número de *encoders* é realizada. Como os *chunks* comprimidos são gerados por múltiplos *encoders*, o *chunk n*+1 pode ser enviado 4 ms após o envio do *chunk n*, fazendo com que não seja necessário aguardar 20 ms como seria o caso se o empacotamento fosse feito sequencialmente. Esse é outro ponto de otimização utilizado no FAST para evitar os 16 ms impostos pelo atraso de empacotamento do *codec* Opus. Embora ocorra um aumento no número de pacotes na rede, espera-se uma redução na latência do sistema mesmo com o aumento da largura de banda.

Dessa forma, cada *sample* é designado para o *encoder* correto, preservando a *timeline* e, consequentemente, evitando os ruídos no áudio, fazendo com que tanto FEC quanto redundância possam coexistir.

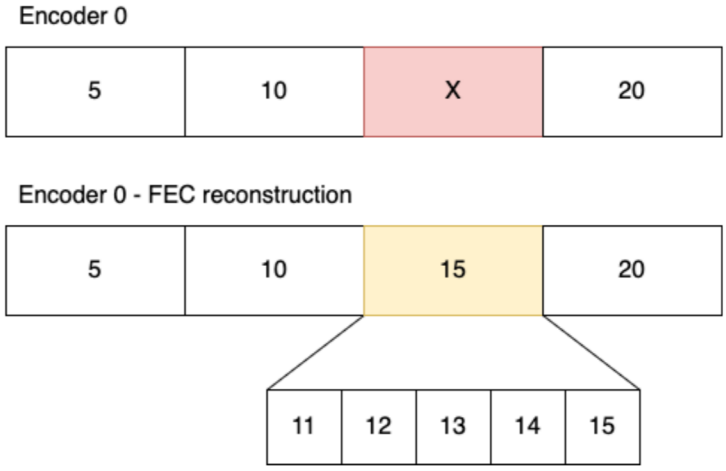
O algoritmo de FEC disponibilizado pela libopus é opcional e vem por padrão desativado. Na solução explorada nesse trabalho, o *in-band* FEC do *codec* é habilitado para que o algoritmo possa inserir dados do *chunk* anterior no *chunk* que será enviado em sequência. Os dados inseridos pelo *codec* são uma versão de baixo *bitrate* do *chunk* anterior, que será utilizada para reconstruí-lo caso o mesmo não seja recebido apropriadamente no dispositivo móvel.

3.3.2 *Decoder*

O *decoder* realiza os passos inversos do *encoder*, sendo responsável por receber a informação comprimida e decodificá-la, e, quando necessário, efetuar a aplicação do algoritmo de FEC. Esse algoritmo é utilizado quando um pacote é perdido na rede e os dois pacotes vizinhos são recebidos, tornando possível reconstruir os dados do pacote não recebido através dos metadados da vizinhança.

A Figura 3.5 exemplifica uma situação em que o terceiro *chunk* do *encoder* zero foi perdido na rede. O *chunk* de número 15 pode ser reconstruído utilizando FEC, uma vez que o *chunk* com a amostra 20 foi recebido com sucesso no *socket* da aplicação. Dessa forma, além da redundância temporal, ainda existe a possibilidade de corrigir a perda de *chunks* através da reconstrução dos dados por *Forward Error Correction*.

Figura 3.5: Reconstrução de amostra perdida através do uso de FEC



E, assim como o *encoder*, o *decoder* deve criar mais de uma instância do Opus para que os ruídos percebidos em uma *timeline* intercalada sejam evitados. Conforme apresentado na Subseção 3.3.1, essa abordagem descarta os ruídos causados pela falha de continuidade no sinal com um baixo *tradeoff* de ocupação em memória.

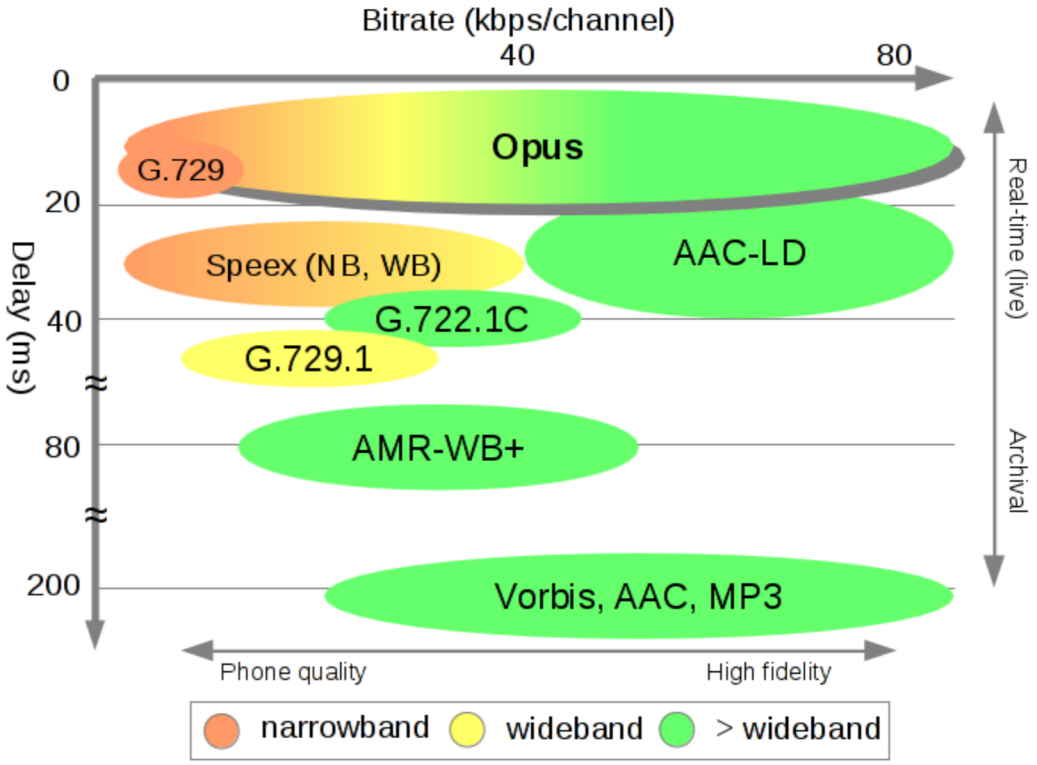
3.3.3 Opus

Como mencionado nas seções anteriores, os módulos *encoder* e *decoder* fazem uso do *codec* Opus. Essa ferramenta desenvolvida pela *Xiph.org Foundation*[4] utiliza os *codecs* Skype SILK, que é baseado no método de predição linear, e o *codec* CELT, que também pertence à Xiph.org e é baseado na *Modified Discrete Cosine Transform* (MDCT) [Valin et al. 2013]. Esse *codec* possui foco no uso *real-time* para aplicações na internet, com um *delay* de compressão e extração que pode chegar a um valor mínimo de 5 ms.

Conforme a Figura 3.6, quando comparado aos demais *codecs* do mercado o Opus apresenta melhor performance quando avaliada a latência por *bitrate*, mantendo um valor abaixo de 20 ms para *bitrates* inferiores a 80 kbps por canal. Neste trabalho, por exemplo, utiliza-se um *bitrate* de 76.80 kbps por canal. Logo, o *delay* inserido pela compressão/extração se mantém abaixo do valor citado na literatura para sincronização labial.

[4]https://xiph.org/

Figura 3.6: Comparação entre Opus e os demais *codecs* do mercado



Fonte: https://opus-codec.org/static/comparison/opus_comparison.png, acessado em: 16/07/2020

O codificador Opus também possui algumas restrições em relação à sua configuração:

1. A taxa de amostragem do *encoder* varia entre 8 e 48 kHz, não suportando taxas maiores. Entretanto, esse valor é justificado, uma vez que, por utilizar compressão com perdas, o *encoder* considera o espectro de frequência audível pelos humanos, que varia entre 20Hz e 20kHz durante a compressão. Nesse caso, usando resultados propostos no teorema de Nyquist, conclui-se que a frequência 48 KHz atende ao requisito de codificação de áudio para humanos.

2. O *encoder* também não fornece suporte de forma nativa à taxa de amostragem 44.1 kHz, sendo necessário o uso *resampling* para aproximar o valor a uma taxa aceitável.

3. O Opus também não permite o uso de quantidades de bits por amostra (*bit depth*) diferente de 16 *bits* para valores inteiros. Logo se a captura de entrada for feita com

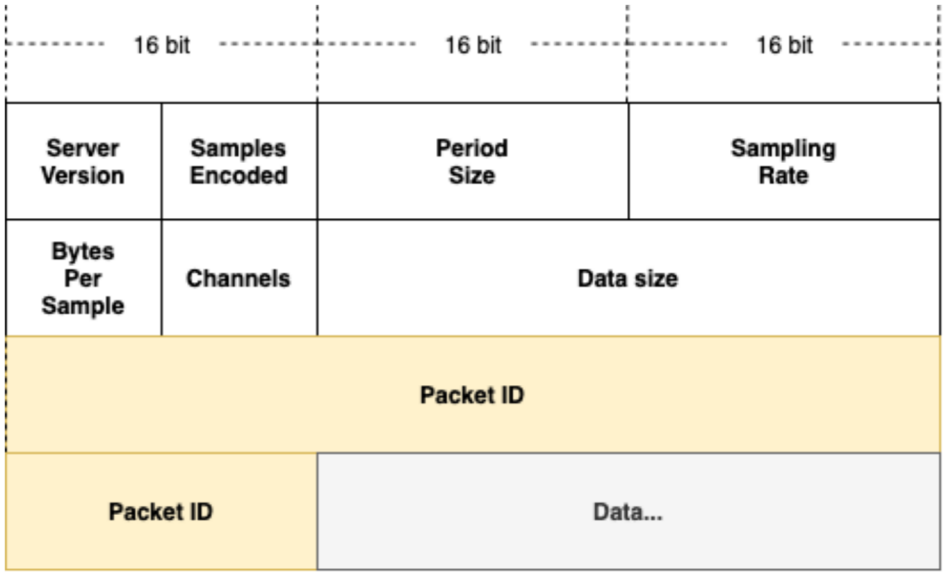
24 *bits*, por exemplo, o valor terá de ser convertido para ponto flutuante para poder ser processado.

Embora o Opus possua certas limitações, os benefícios trazidos pelo *codec* se sobressaem às dificuldades impostas por ele. É por esta razão que a solução proposta neste trabalho utiliza essa ferramenta que, atualmente, é caracterizada como um dos *codec open source* mais robustos e eficientes do mercado.

3.4 Sintaxe dos Pacotes FAST

O FAST utiliza o protocolo de camada de transporte UDP com a estratégia de comunicação usando o suporte a transmissão IP *multicast* para otimizar a transmissão para grupos de usuários. Como o protocolo não garante a ordem de chegada dos pacotes, um cabeçalho deve ser definido, o formato do pacote é mostrado na Figura 3.7.

Figura 3.7: Cabeçalho utilizado na transmissão dos pacotes



Esse cabeçalho deve conter identificação do pacote, configurações do áudio e tamanho da amostra. Seus campos são:

- *Server version*: 8 *bits* indicando a versão do servidor, informação bastante importante, pois diferentes servidores poderão estar em uso em um mesmo ambiente, trazendo

mudanças tanto do *encoder* quanto da forma como a redundância é implementada. Dessa forma, dados de implementações diferentes podem ser rejeitados pelo cliente;

- *Samples encoded*: 8 *bits* para indicar a quantidade de *samples* que foram comprimidas. O valor *default* utilizado no FAST é de cinco *samples*;
- *Period size*: 16 *bits* referentes aos *frames* capturados. Os *samples* são recebidos a cada 4 ms, o que equivale a um *frame* de tamanho 192;
- *Sampling rate*: Os 16 *bits* seguintes são referentes à taxa de amostragem do áudio transmitido;
- *Bytes per sample*: 8 *bits* indicam a quantidade de *bytes* em um *sample* do áudio de entrada. Essa informação é utilizada exclusivamente pelo *decoder*;
- *Channels*: 8 *bits* do cabeçalho indicam a quantidade de canais em um *sample*;
- *Data size*: 32 *bits* indicando o tamanho do áudio comprimido;
- *Packet* ID: 64 *bits* referentes ao identificador do pacote atual, destacado na Figura 3.7 em amarelo.

Dessa forma, a aplicação cliente pode ajustar-se com base no primeiro pacote recebido da rede. A versão do servidor permite que inconsistências entre diferentes abordagens sejam evitadas e a informação enviada referente a entrada de áudio faz com que não seja necessário manter os valores *hard coded* na aplicação. Logo, além da numeração dos pacotes, o cabeçalho também traz flexibilidade à solução proposta.

3.5 *Arena*

Conforme mencionado no início da Seção 3, neste trabalho é utilizado um gerenciamento de memória denominado *region-based memory management*. A *arena*, como também é denominada, é um bloco contíguo de memória que é manipulado diretamente pelo detentor da região. O propósito desse tipo de alocação é facilitar a manipulação de uma grande quantidade de dados que deve ser acessada e realocada com frequência.

Na FAST a *arena* é utilizada tanto no servidor quanto na aplicação do dispositivo móvel. Entretanto, no servidor ela possui maior performance por residir na memória sem que haja a possibilidade de paginação, uma vez que existe a possibilidade de efetuar chamadas de métodos privilegiados juntos ao sistema operacional. Ainda no servidor, o trabalho da *arena* é persistir os *frames* de áudio capturados da placa de som para que sejam consumidos pela *thread* de compressão e *streaming*.

No dispositivo móvel, a *arena* é utilizada para ”bufferização”, funcionando para compensar variações no retardo de transmissão (*jitter*). Para auxiliar na mitigação do atraso é proposto um algoritmo adaptativo que auxilia o contexto de memória, buscando evitar que todos os dados sejam drenados. Mais detalhes sobre a implementação serão discutidos na Subseção 3.6.

3.6 *Mixer* de Áudio

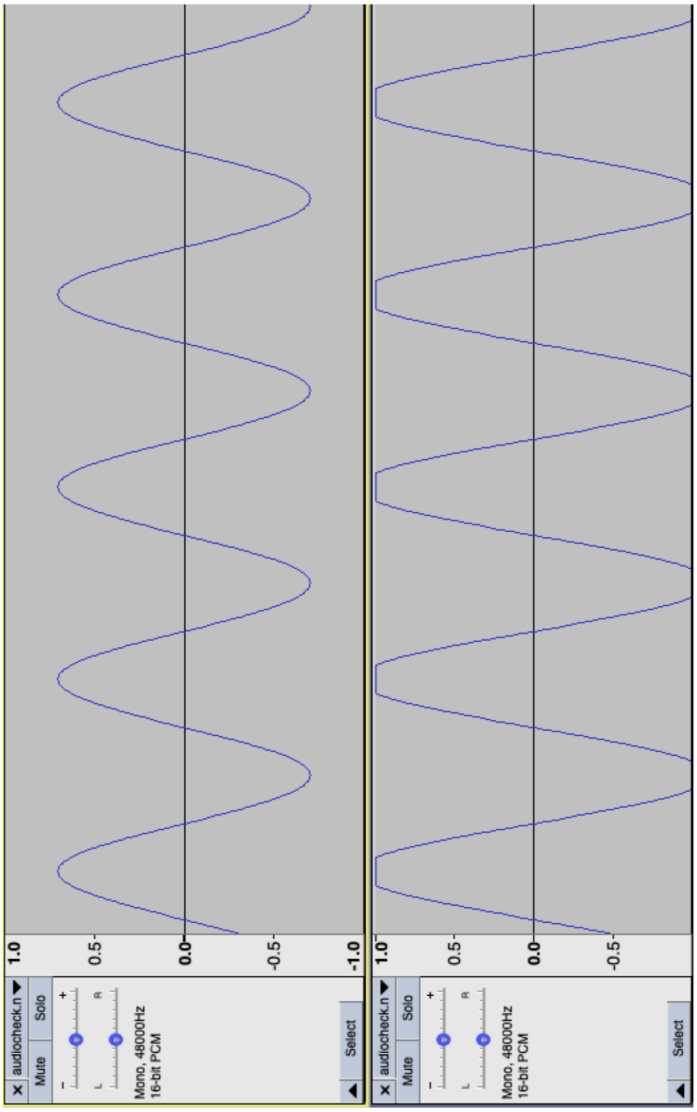
O *mixer* de áudio é iniciado assim que a arena recebe uma quantidade de dados pré-estabelecida. Esse limiar é definido com base na performance do dispositivo móvel que está sendo utilizado e no retardo entre fonte e destino que se deseja alcançar. Mesmo que a maioria dos celulares modernos sejam capazes de reproduzir o áudio em baixa latência, alguns dispositivos não possuem otimização suficiente para manter a apresentação em tempo real. Dessa forma, para atingir valores abaixo do atraso que provocam a sensação de erros de sincronização labial, deve-se definir um *tradeoff* entre performance do dispositivo e valores de *buffer*.

Este componente da solução é responsável por preparar a trilha de áudio para ser executada para o ouvinte. Dentre as alterações que o *mixer* pode realizar sobre o áudio original estão: amplificação do áudio, replicação de um dos canais e reamostragem. Ressalta-se que cada uma dessas opções pode ser desativada pelo usuário da aplicação.

A amplificação do áudio é o processo em que aumenta-se a informação referente ao áudio de entrada de forma uniforme. Esse processo é feito de forma linear em que um fator numérico é multiplicado a cada amostra de 16 *bits* do áudio original. É importante definir um valor máximo para o processo de ganho, uma vez que existe a possibilidade de *clipping*, visto que o valor resultante da multiplicação pode exceder a precisão dos samples (*bit depth*)

definido para o áudio em questão. A Figura 3.8 exibe duas amostras de áudio senoidal de 500 Hz com precisão de 16 bits e taxa de amostragem de 48 kHz. A amostra superior corresponde ao áudio original, já a amostra inferior é o áudio acrescido de ganho. Como pode ser observado, o áudio que foi amplificado possui *clipping* em suas extremidades, ou seja, o áudio teve sua informação degradada.

Figura 3.8: Presença de *clipping* durante a aplicação de ganho em faixas de áudio

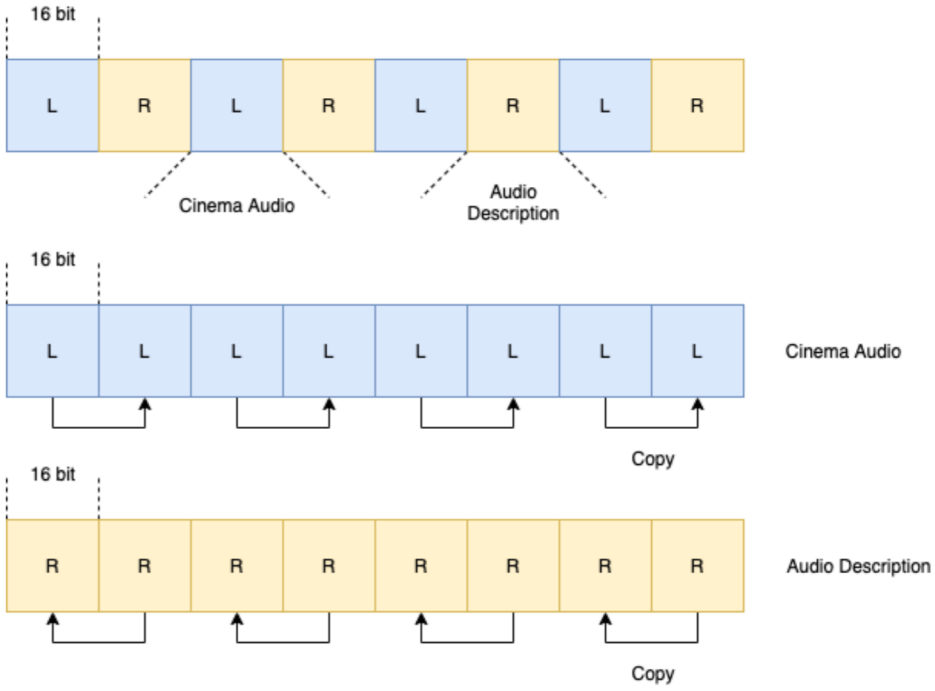


Neste trabalho, o limite superior de ganho, assim como o fator multiplicador deve ser definido de antemão pelo desenvolvedor. Existem formas de evitar *clipping*, porém, nesta pesquisa foram utilizados valores pré-definidos para adiantar o desenvolvimento e os testes

da aplicação.

A pesquisa foi desenvolvida em parceria e com financiamento da Assista Tecnologia[5], uma empresa que desenvolve soluções de acessibilidade. A empresa pretende utilizar a solução desenvolvida para gerar um serviço de áudio que será utilizado em cinemas digitais. Dessa forma, um dos requisitos levantados pela Assista Tecnologia é replicar um canal de áudio, fazendo com que o som de entrada, que é estéreo, seja reproduzido como uma trilha mono. O canal esquerdo, que na Figura 3.9 é representado em tom azul com a letra L, transmite o áudio amplificado do filme da sala de cinema, e poderá ser utilizado por pessoas com deficiência auditiva severa, que necessitam desse recurso para poderem assimilar a informação. Já o canal do lado esquerdo, que na imagem está representado em tom amarelo e letra R, contém a audiodescrição do filme, que poderá ser utilizada por pessoas com deficiência visual. A Figura 3.9 exibe o comportamento esperado em um ambiente de cinema, em que o sinal que é recebido pela rede de forma intercalada é replicado *in-place* dependendo do tipo de conteúdo solicitado pelo usuário. Na sequência, pode-se ver o áudio de entrada seguido da exemplificação da replicação do áudio para cada recurso selecionado.

Figura 3.9: Canais de áudio observados no cinema digital



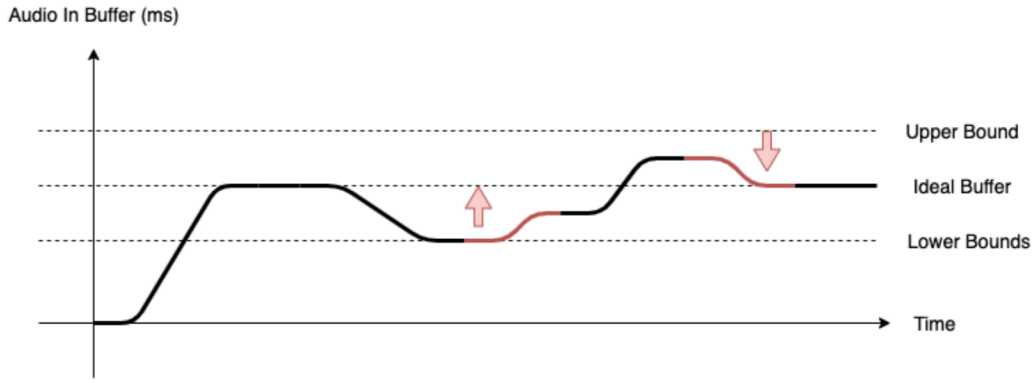
[5]https://assistatecnologia.com.br/

O algoritmo de replicação de áudio tem complexidade linear, ou seja, dada uma entrada de áudio, é necessária uma iteração sobre metade da informação para replicar os dados, e, dessa forma, o algoritmo não é impactante para a latência final da proposta. Além disso, o processo de replicação do conteúdo e a aplicação do ganho podem ser feitos concomitantemente, e, dessa forma, apenas uma iteração é necessária para aplicar ambos os algoritmos.

Por fim, o *mixer* tem o papel de amenizar os efeitos de *jitter* causados pela rede através do uso de um algoritmo adaptativo, que é responsável por expandir e contrair as *samples* de áudio, assim como o funcionamento de um *buffer* elástico (uma espécie de sanfona), reproduzindo o áudio de forma mais lenta quando o *buffer* está se esvaziando e de forma mais rápida quando ele retorna a um nível aceitável. O áudio é manipulado através da adição ou remoção de dados dos *samples* de áudio, e, para isso, a ferramenta proposta utiliza a libsamplerate, um projeto *open source* para *upsampling* e *downsampling* de dados PCM. A API fornece 5 mecanismos de *resampling*, variando entre algoritmos lineares e algoritmos por interpolação, e, dessa forma, possibilita a flexibilidade no ajuste da qualidade do algoritmo de *resampling* sem que se faça necessário a troca de bibliotecas.

A Figura 3.10 exemplifica o funcionamento do algoritmo adaptativo através de um plano cartesiano em que a coordenada 'Y' representa os dados no *buffer* em milissegundos e a coordenada 'X' representa a flutuação de dados no *buffer* ao longo do tempo. Os segmentos de linha avermelhados indicam um ponto em que ocorre *resampling* na amostra. Conforme observado na Figura 3.10, existem três limiares, são eles: *ideal buffer*, *upper bound* e *lower bound*. O *ideal buffer*, como sugere o nome, é o ponto em que o *buffer* deveria se manter durante toda a reprodução do conteúdo. *Upper bound* e *lower bound* são, respectivamente, os limiares superior e inferior para que o algoritmo seja ativado, a partir do momento em que os valores do *buffer* atingem o *lower bound* começa o processo de upscaling nas amostras e o inverso ocorre quando o *upper bound* é alcançado.

Figura 3.10: Funcionamento do algoritmo adaptativo sobre plano cartesiano



O valor de *buffer* ideal deve ser configurado com base na avaliação de performance de cada dispositivo celular, e para os valores de *upper bound* e *lower bound* são considerados nesse trabalho três *chunks* acima e abaixo do valor ideal.

A primeira ocorrência de *resampling* na Figura 3.10 ocorre quando o limiar inferior é atingido. Nesse caso, como indicado pela seta vermelha, o processo de *upsampling* ocorre e, dessa forma, o número de pacotes consumidos é levemente reduzido, fazendo com que seja possível recuperar uma pequena quantidade de pacotes da rede. Assim que o *buffer* ideal é atingido, o processo de *upscaling* é interrompido e inicia-se a compensação de atraso, conforme indicado na segunda seta. O processo de compensação é necessário para que o áudio escutado no dispositivo mantenha-se em sincronia com o áudio da tela. Caso contrário, o algoritmo somente inserirá atraso na reprodução do áudio.

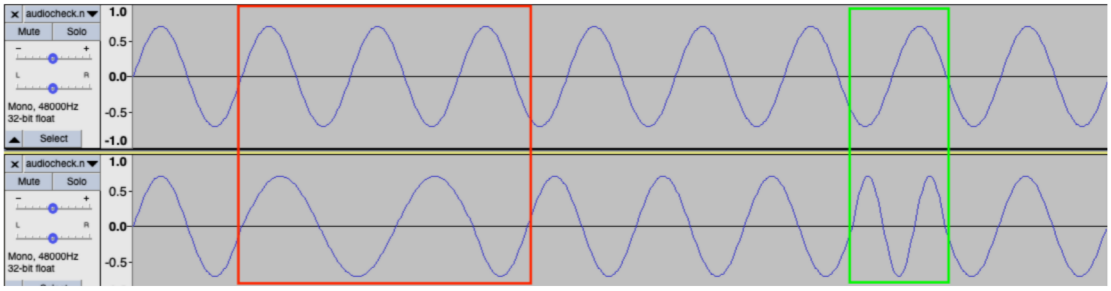
O *upper bound* existe para acelerar a reprodução caso uma rajada de dados seja recebida na rede. Nesse caso, é realizada uma subamostragem para acelerar a reprodução do conteúdo com o propósito de manter a reprodução do áudio em uma janela ideal. Após atingir o valor *ideal buffer*, o mesmo algoritmo de compensação é acionado para novamente reajustar a latência.

Como o FAST lida com oscilações na transmissão de dados muito rápidas, a correção efetuada pelo algoritmo precisa ocorrer rapidamente trazendo benefícios para aplicação pois ao invés de um intervalo de silêncio percebido como um ruído, o usuário escuta o áudio com uma pequena mudança de tonalidade.

A Figura 3.11 traz outra perspectiva sobre o comportamento que o algoritmo adaptativo impõe ao fluxo de áudio. A forma de onda superior simula o áudio transmitido pela sala de

cinema, ou seja, esse é o áudio que deveria ser reproduzido integralmente pelo dispositivo móvel. Entretanto, nesta simulação o algoritmo adaptativo verificou que o *buffer* atingiu o *lower bound* e estendeu a amostra do áudio inferior, que está destacada pelo quadrado vermelho. Logo em seguida, no quadrado verde, o *buffer* retornou ao seu estado normal e foi realizado o *downsampling* para compensação do atraso, dessa forma, percebe-se que o áudio do dispositivo foi acelerado.

Figura 3.11: Funcionamento do algoritmo adaptativo através da observação de faixas de áudio em *waveform*



3.7 *Player*

A aplicação móvel proposta neste trabalho será desenvolvida para o uso em *smartphones* e *tablets* e, dessa forma, é importante definir um sistema operacional em que ela será desenvolvida.

Dentre os inúmeros sistemas operacionais existentes no mercado, os sistemas desenvolvidos pela *Apple Inc.*[6] e pela *Google LLC*[7], respectivamente, *iOS*[8] e *Android*[9], são os mais populares do mercado. Segundo pesquisas do *International Data Corporation*[10] (IDC), o número de dispositivos *Android* no mercado totalizou 86.10% dos *smartphones* vendidos no mundo no ano de 2019, enquanto dispositivos *iOS* mantém o restante do mercado, um total de 13.90% dos celulares vendidos. Outro fator importante que deve ser considerado

[6]https://www.apple.com/
[7]https://abc.xyz/
[8]https://www.apple.com/br/ios
[9]https://www.android.com/
[10]https://www.idc.com/promo/smartphone-market-share/os

durante a pesquisa da definição do sistema operacional é o valor requerido para a aquisição de um dispositivo com o *software* instalado. Em uma pesquisa rápida no mercado é possível constatar que dispositivos *Android* são muito mais baratos do que dispositivos *iOS*. O valor é justificado pelo fato da existência de muitas empresas desenvolvendo aparelhos para o sistema operacional e, ao contrário do *iOS*, que é proprietário, o *Android* possui código aberto e pode ser comercializado por diversos fabricantes.

Além do fator acessibilidade financeira, também é possível ter acesso ao código fonte do sistema operacional *Android*. Dessa forma, otimizações de baixo nível podem ser realizadas no *kernel* do aparelho caso seja necessário.

Desse modo, para a solução proposta optou-se pelo uso do sistema operacional *Android*, tanto pela facilidade no acesso a dispositivos quanto pelo baixo custo das unidades.

O próximo passo foi a definição de um reprodutor de áudio apropriado para aplicações de baixa latência. Ao observar a documentação do *Android*, o usuário depara-se com uma interface genérica sobre os *drivers* do *kernel* denominada *MediaPlayer*[11]. Embora seja possível reproduzir dados a partir de um arquivo, o *framework* não permite que dados customizados sejam reproduzidos e, dessa forma, o uso dessa solução foi descartado.

Para obter resultados mais contundentes durante a pesquisa do *player* foram definidas, conforme a enumeração a seguir, as funcionalidades desejadas em um *player* apropriado para *streaming* de baixa latência.

1. *Playback* a partir de um fluxo de *bytes*;
2. Controle do *buffer* do player;
3. Flexibilidade nos valores de *sample rate* e *bit depth*.

Após pesquisa extensiva na documentação fornecida pelo *Google* e na ampla comunidade *Android*, definiu-se como os melhores *players* para aplicações de baixa latência no mercado as APIs Opensl ES[12] e AAudio[13]. Segundo as suas documentações, ambos têm foco no *playback* em baixa latência, permitindo uma configuração avançada sobre as características do áudio que será reproduzido e do controle do *buffer* de *playback*.

[11] https://developer.android.com/reference/android/media/MediaPlayer

[12] https://www.khronos.org/opensles/

[13] https://developer.android.com/ndk/guides/audio/aaudio/aaudio

Definir a melhor API dentre as duas é um desafio, uma vez que a AAudio tornou-se o padrão para reprodução de áudio em real time no *Android*, embora apenas a partir da versão 8. O OpenSL ES, por outro lado, está a mais tempo no mercado, embora os seus resultados não sejam tão bons quando comparado à primeira API analisada, apresentando uma pequena diferença na latência de reprodução. Entretanto, o OpenSL ES é interessante pois dá suporte a versões do *Android* até a release 2, ou seja, a janela de dispositivos suportados é muito maior.

Desse modo, o ideal é manter ambas as APIs para suprir a maior diversidade de dispositivos. Pensando nisso, a *Google* desenvolveu uma ferramenta denominada Oboe[14]. O *framework*, que é uma interface sobre os dois *players* discutidos anteriormente, é capaz de selecionar o *player* ideal com base na versão do *software* do sistema operacional.

Logo, para a solução proposta, o *player* utilizado é escolhido diretamente pelo *framework* Oboe.

3.8 Considerações Finais

Neste capítulo a arquitetura do FAST, o seu funcionamento e as suas principais características foram apresentadas. Os componentes da solução foram discutidos em detalhe, seguidos das considerações a respeito das tecnologias utilizadas.

No Capítulo 4 serão apresentados os experimentos que foram conduzidos para avaliar a solução com relação à qualidade da *stream* de áudio, mitigação de *jitter* e aliviamento da perda de pacotes de rede.

[14]https://github.com/google/oboe

Capítulo 4

Avaliação Experimental

Neste capítulo, será apresentado um conjunto de experimentos conduzidos com o objetivo de avaliar a solução proposta.

Durante a execução dos experimentos, pretende-se avaliar se a solução deste trabalho é capaz de transmitir o conteúdo de áudio com base nas restrições impostas pelo fator de sincronização labial. Além disso, pretende-se avaliar se a solução é capaz de manter um *buffer* mínimo com o propósito de mitigar falhas na rede e, consequentemente, ruídos causados por *underflow*.

A seguir, serão apresentados a definição e o planejamento do experimento seguido de sua execução e análise.

4.1 Definição e Planejamento do Experimento

Nesta seção serão apresentados a definição e o planejamento dos experimentos conduzidos para avaliar a solução. Esta etapa envolve a definição do propósito do experimento, a formulação de suas hipóteses, definição das variáveis e sua instrumentação.

O objetivo do experimento realizado foi avaliar os pontos do sistema FAST que injetam perda e atraso na transmissão de uma *stream* de áudio e o impacto desses problemas na qualidade da experiência dos usuários do sistema. O experimento realizado foi definido da seguinte forma:

- **Objeto de estudo**: O objeto de estudo é a análise da qualidade do conteúdo transmitido.

- **Propósito**: O propósito foi avaliar se as falhas na transmissão são perceptíveis. Para isso, a eficácia (quantidade de pacotes que foram entregues com sucesso ao cliente) e eficiência (pacotes que foram apresentados no tempo apropriado no cliente) do sistema foram avaliadas.

- **Foco de qualidade**: Os principais efeitos estudados são a eficácia e eficiência do sistema.

- **Perspectiva**: A partir do ponto de vista dos pesquisadores.

- **Contexto**: Essa parte do experimento foi conduzida através de um conjunto de testes realizados no ambiente de cinema digital. Para isso, ferramentas proprietárias da empresa Assista Tecnologia especialmente desenvolvidas para capturar, analisar e manipular os dados, foram utilizadas por pesquisadores para avaliar o desempenho da transmissão do conteúdo.

Resumidamente, essa parte do experimento pode ser definida da seguinte forma:

- Analisar a transmissão do conteúdo
- com o propósito de avaliação
- com respeito a eficiência e eficácia
- a partir do ponto de vista dos pesquisadores
- no contexto de testes computacionais com objetivos realizados em um cenário experimental de transmissão de conteúdo multimídia em cinema digital.

4.1.1 Planejamento do Experimento

Nesta subseção, o planejamento de cada uma das partes do experimento será descrito levando-se em consideração as hipóteses, variáveis, projeto e instrumentação.

4.1.2 Seleção do Contexto

Os experimentos deste trabalho foram conduzidos no escritório da empresa Assista Tecnologia com o auxílio de pesquisadores e engenheiros da empresa. Nesses testes, a solução proposta foi avaliada a partir de três observações, são elas:

1. Ocupação de pacotes no *buffer* de *playback*;
2. Divergência em milissegundos entre o áudio da sala e o áudio do dispositivo móvel;
3. Percepção de ruídos e pausas através da exibição completa de ambas as trilhas de áudio.

4.1.3 Formulação das Hipóteses

Informalmente, as hipótese formuladas nesse experimento são:

1. A solução proposta (FAST) reproduz os pacotes em um tempo máximo de 40 milissegundos de *delay* com relação à fonte de áudio;
2. A solução proposta (FAST) é capaz de manter o *buffer* de *playback* com ao menos um *sample*;

Logo, as hipóteses para esse experimento são definidas formalmente da seguinte forma:

1. **Hipótese Nula** $H0$: A solução proposta (SP) não é capaz de reproduzir conteúdos com tempo médio de 40 milissegundos de divergência, ou seja, o tempo médio entre transmissão e reprodução é superior a 40 milissegundos.

$$\mu\, atraso(SP) > 40ms \tag{4.1}$$

Hipótese alternativa $H1$: A solução proposta (SP) é capaz de reproduzir o conteúdo com atraso igual ou inferior a 40 milissegundos de divergência.

$$\mu\, atraso(SP) <= 40ms \tag{4.2}$$

2. **Hipótese Nula** $H0$: A solução proposta (SP) não é capaz de manter o *buffer* de *playback* com ao menos um *sample*.

$$\mu\, descarte(SP) = 0 \tag{4.3}$$

Hipótese alternativa $H1$: A solução proposta (SP) é capaz de manter o *buffer* de *playback* com um ou mais *samples*.

$$\mu\, descarte(SP) >= 1 \tag{4.4}$$

4.1.4 Seleção das Variáveis

As **variáveis independentes** deste experimento são a distância dos dispositivos móveis até o roteador, a quantidade de dispositivos presentes na rede e a quantidade de redes que utilizam o mesmo canal de transmissão.

A **variável dependente** deste experimento é a quantidade de *samples* de áudio no *buffer* de *playback*. Dessa forma, utiliza-se apenas essa variável, pois sabendo-se a quantidade de dados no *buffer* em milissegundos, é possível inferir a latência entre o conteúdo apresentado na sala de cinema e o dispositivo móvel, assim como a ocupação de dados no *smartphone*.

4.1.5 Projeto do Experimento

Segundo os padrões de experimento apresentados por [Wohlin et al. 2012], a definição e hipóteses indicam que o experimento deve seguir o tipo fatorial 2^k(do inglês 2^kfactorial). Assim, assumindo um fator $k = 3$, onde k são as variáveis independentes do experimento, tem-se um total de 8 experimentos ($2^3 = 8$).

Dessa forma, os três fatores definidos para esse experimento são:

- F1: Distância entre roteador e dispositivo móvel;
- F2: Quantidade de redes na mesma faixa de canal;
- F3: Quantidade de dispositivos conectados na rede;

Seguindo a metodologia dois elevado a k, cada um desses fatores deve ter no máximo dois estados. Para o nosso experimento em questão, tem-se:

- F1
  - (-1) 3.5 metros (teste próximo ao roteador)
  - (1) 7.3 metros (distância padrão entre projetor e tela em uma sala de cinema, ou seja, o mais distante possível do roteador)
- F2
  - (-1) 1 roteador
  - (1) 2 roteadores
- F3
  - (-1) 1 dispositivo
  - (1) 3 dispositivos

Logo, a Tabela 4.1 exibe os experimentos com base na metodologia:

Tabela 4.1: Experimentos com base na metodologia 2^k

Experimentos	**Fatores**		
	F1	**F2**	**F3**
1	-1	-1	-1
2	-1	-1	1
3	-1	1	-1
4	-1	1	1
5	1	-1	-1
6	1	-1	1
7	1	1	-1
8	1	1	1

As variáveis independentes são cruciais durante o processo experimental, uma vez que a mutabilidade delas é o fator que irá determinar o resultado das variáveis dependentes. Logo, é necessário que seja discutido o motivo pelo qual cada variável foi selecionada e qual é o seu impacto no resultado do experimento.

Fator 1 - Distância entre roteador e dispositivo móvel

A distância entre roteador e dispositivo móvel é uma boa forma de avaliar se a solução é resiliente a falhas na rede. Quanto mais longe do roteador, mais suscetível é o sistema a interferências como colisão entre redes, obstáculos e a potência do sinal.

Dessa forma, foram definidas duas distâncias, a primeira onde o dispositivo móvel é situado a 3,5 metros do roteador, ou seja, uma localização em que o espectador está muito próximo do equipamento de transmissão; e a segunda onde o usuário está a 7,3 metros, ou seja, aproximadamente o dobro da distância inicial.

Os testes foram realizados em ambiente interno e buscou-se preservar a transmissão dos dados com o mínimo de obstáculos, evitando, por exemplo, paredes, vidros e móveis. Assim, é possível analisar o impacto na entrega de pacotes conforme a distância entre os equipamentos aumenta independentemente dos fatores externos ao fator.

Fator 2 - Quantidade de redes na mesma faixa de canais

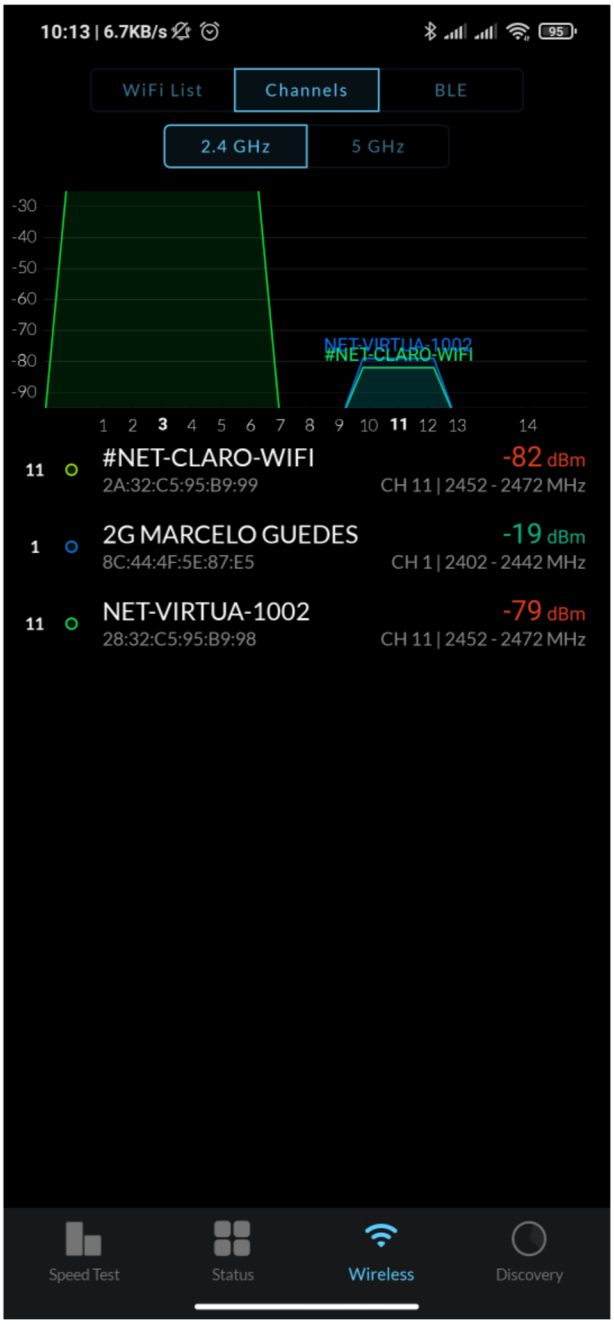
O segundo fator definido para o experimento busca analisar a flexibilidade da aplicação a instabilidades na rede. Ao utilizar somente um dispositivo - primeira variação do fator - busca-se evitar o conflito entre ondas em uma mesma frequência SHF (*Super high frequency*), ou seja, um cenário ideal onde apenas uma rede transmite na região. A segunda alteração do fator busca inserir mais redes em um mesmo canal, assim, agravando a degradação do sinal.

Para evitar a colisão de pacotes com outras redes que não fazem parte dos testes, as redes *wireless* na área são analisadas com um aplicativo que mapeia redes e canais, para este trabalho, foi utilizado o aplicativo da Ubiquiti Inc.[1], denominado WiFiman[2], cujo funcionamento é exemplificado pela Figura 4.1. Dessa forma, é possível isolar as redes que estão sendo utilizadas nos testes das demais redes no local, e assim, reduzindo o risco de interferência que possa comprometer o experimento.

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

[2] https://play.google.com/store/apps/details?id=com.ubnt.usurvey&hl=pt_BR&gl=US

Figura 4.1: Screenshot do aplicativo WifiMan da Ubiquiti Inc.



Fator 3 - Quantidade de dispositivos conectados na rede

Embora o último fator dependa quase que exclusivamente da capacidade computacional do roteador, uma vez que a transmissão dos dados e o controle do grupo *multicast* são tarefas de sua responsabilidade, é importante observar o impacto na transmissão causado pelo

aumento de celulares na rede.

Para os testes realizados, decidiu-se utilizar a variação de um e três dispositivos. O valor de três dispositivos foi selecionado pois, segundo instrução normativa estipulada pela Ancine [Ancine - Instrução Normativa N 128, 2016], este é o número mínimo de aparelhos de reforço utilizados em um complexo. Dessa forma, é possível aferir se a solução proposta consegue manter o fluxo de dados independentemente da quantidade de dispositivos inseridos na rede.

4.1.6 Instrumentação

A amostra de áudio a ser utilizada nesse trabalho segue os padrões de qualidade estipulados pelo servidor de cinema digital. Sendo assim, o áudio foi retirado de uma narração de áudio profissional de alta qualidade com 16 *bits* de quantização, 48000 Hertz (Hz) de amostragem e sinal estéreo.

A máquina de *streaming* possui processador Intel Pentium J4205, 4 GB de memória RAM e sistema operacional Ubuntu Linux 16.04 com *kernel* 4.10.0. Os dispositivos de reprodução utilizados foram *smartphones* Asus Zenfone Max Plus M2, cujo sistema operacional é o Android 7.0.1 (Nougat).

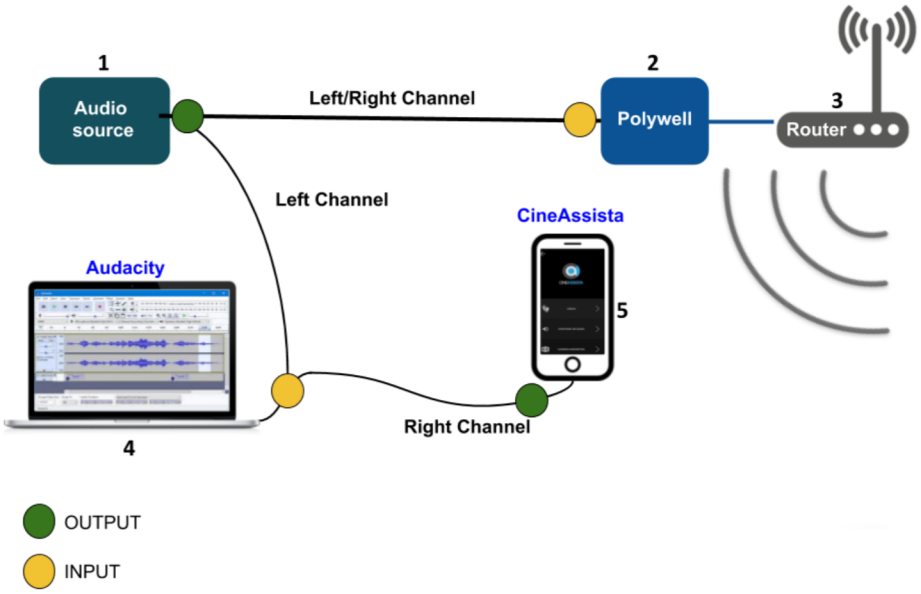
4.2 Execução e Análise

O experimento para avaliar o desempenho da solução proposta foi realizado através de testes em laboratório. Conforme discutido na Seção 4.1, este experimento possui oito testes distintos que mapeiam diferentes cenários, que podem ser encontrados em ambientes reais de cinema. Com o intuito de gerar resultados mais fiéis, cada um dos oito testes foi executado quatro vezes em baterias de 10 minutos. Dessa forma, para levantar os dados que serão analisados foram realizados 32 testes de 10 minutos onde foram coletados o áudio da fonte e do dispositivo móvel, assim como a ocupação dos pacotes no *buffer* de áudio.

O pipeline dos testes pode ser observado a partir da Figura 4.2. Os dispositivos 1, 2, 4 e 5 comunicam-se através de um cabo customizado, ilustrado na Figura 4.3. O conector é uma solução de baixo custo utilizada para capturar mais de uma fonte de áudio simultaneamente, delegando a fonte de áudio para o canal da esquerda e o áudio reproduzido pelo dispositivo móvel para o canal da direita do áudio capturado. Dessa forma, ao analisar a captura do

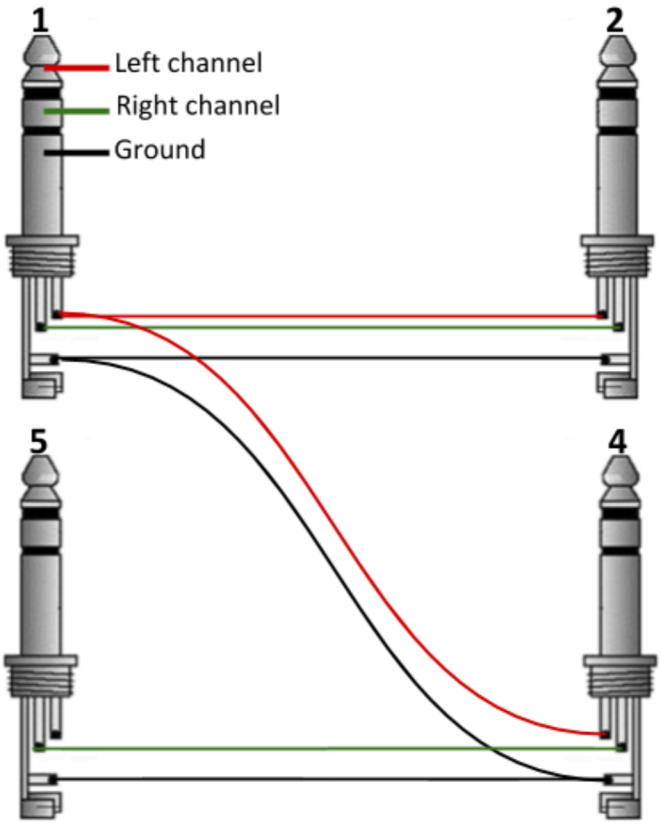
playback pode-se aferir com facilidade a latência entre *playback* e fonte. Ainda na Figura 4.2, pode-se observar que os dispositivos 1 e 5 possuem duas marcações com coloração verde, que indicam que eles são dispositivos de saída, ou seja, de onde o áudio será retirado para análise da latência. Já os dispositivos 2 e 4 são marcados com círculos amarelos, indicando que eles consomem dados. O dispositivo 2 é a máquina de *streaming* que repassa os dados pelo roteador Wi-Fi (3) para o dispositivo móvel e a máquina 4 é a responsável por capturar o conteúdo da fonte (1) e do *smartphone* (5) simultaneamente.

Figura 4.2: Conexão entre dispositivos através do cabo TRS de 4 pontas



A Figura 4.3 exemplifica a pinagem do cabo TRS (*Tip-Ring-Sleeve*) de 3.5mm em cada um dos dispositivos da Figura 4.2. Conforme observado, o conector 1 recebe informação da fonte e repassa os dados para o conector 2 para que sejam transmitidos. Ainda no conector 1, o sinal do canal esquerdo é replicado para o conector 4, este ainda recebe o canal direito do conector 5 e, dessa forma, um canal da fonte e um canal do smartphone são repassados para a máquina de captura. Assim, é possível gravar simultaneamente os dados vindos da fonte e do celular propiciando uma avaliação da latência sem a necessidade de utilização de dispositivos de alto custo como osciloscópios.

Figura 4.3: Cabeamento entre os conectores do cabo TRS de 4 pontas



O Apêndice A apresenta um trecho do *log* de captura referente à ocupação de pacotes no *buffer* de *playback*. Cada linha do *log* é escrita na medida em que os pacotes são avaliados e considerados apropriados para *playback*:

1. ID do teste: Indicador do tipo de teste que, para a captura referente à ocupação de pacotes no *buffer* de *playback*, é a letra B;

2. *Timestamp*: Tempo entre a inserção do primeiro pacote e o pacote atual em microssegundos;

3. Tamanho do *buffer*: Quantidade máxima de áudio no *buffer* em milissegundos;

4. Ocupação: Ocupação do *buffer* em milissegundos.

A seguir, serão apresentados cada uma dos oito experimentos. Em cada uma das seções serão avaliados média, desvio padrão e ocupação máxima e mínima do *buffer*, assim como histogramas e gráficos de *timeline* da captura.

4.2.1 Experimento 01

O primeiro experimento realizado lida com a seguinte variação dos fatores:

- F1: 3,5 metros
- F2: 1 roteador
- F3: 1 dispositivo móvel

A Tabela 4.2 apresenta os valores encontrados durante cada uma das quatro capturas efetuadas para o primeiro experimento.

Tabela 4.2: Capturas do primeiro experimento

Captura	Média de Ocupação (ms)	Desvio Padrão (ms)	Ocupação Mínima (ms)	Ocupação Máxima (ms)	Histo-grama	*Timeline*
A	34.55	2.15	4	40	Figura B.1	Figura C.1
B	28.32	7.55	4	40	Figura B.2	Figura C.2
C	24.67	2.24	4	40	Figura B.3	Figura C.3
D	36.53	2.03	4	40	Figura B.4	Figura C.4

Ao analisar os dados dos histogramas, do gráfico de *timeline* e realizar a comparação junto à Tabela 4.2, é possível obter as seguintes conclusões sobre o primeiro experimento.

Para a captura A, o resultado médio de ocupação situa-se em uma posição confortável, uma vez que a média está mais próxima do limite superior do que do limite inferior, o que auxilia a mitigar efeitos de *underflow*. O desvio padrão foi baixo, o que indica que a maioria dos dados foram mantidos próximos a este limiar. Entretanto, ao analisar o gráfico de *timeline*, é possível perceber que em determinado momento, próximo aos 200 segundos da exibição, ocorreu uma perda significativa da informação, que não foi grande o suficiente para realizar uma perda completa de informação.

Para as capturas B e C, o resultado médio da ocupação acaba sendo bastante comprometido uma vez que o *buffer* oscila próximo aos 26 ms, o que reduz as chances que os algoritmos propostos possuem de eliminar o *underflow*. Ainda na captura B, percebe-se que o valor de desvio padrão oscila bastante, o que é corroborado pela imagem da *timeline* que

indica grande oscilação no *buffer* de *playback*. Já a captura C, apresenta uma *timeline* mais constante, porém com uma média muito pior do que a primeira captura.

Já a captura D, apresenta um bom valor de média e desvio padrão e, consequentemente, uma *timeline* muito mais uniforme, com a ressalva de que nos primeiros 180 segundos do experimento ocorreu uma pequena, porém longa oscilação no *buffer*.

Mesmo não tendo apresentado valores homogêneos, o experimento apresenta dados que indicam que, mesmo com as diversas oscilações, a solução foi capaz de manter os dados nos limites estipulados.

4.2.2 Experimento 02

O segundo experimento realizado lida com a seguinte variação dos fatores:

- F1: 3,5 metros
- F2: 1 roteador
- F3: 3 dispositivos móveis

A Tabela 4.3 apresenta os valores encontrados durante cada uma das quatro capturas efetuadas para o segundo experimento.

Tabela 4.3: Capturas do segundo experimento

Captura	Média de Ocupação (ms)	Desvio Padrão (ms)	Ocupação Mínima (ms)	Ocupação Máxima (ms)	Histo-grama	*Timeline*
A	38.62	1.98	4	40	Figura B.5	Figura C.5
B	37.63	2.26	4	40	Figura B.6	Figura C.6
C	38.43	2.09	4	40	Figura B.7	Figura C.7
D	37.53	2.39	4	40	Figura B.8	Figura C.8

Ao analisar os dados dos histogramas, do gráfico de *timeline* e realizar a comparação junto à Tabela 4.3, é possível obter as seguintes conclusões sobre o segundo experimento.

Quando comparado ao primeiro experimento, apresentado na Seção 4.2.1, foi possível obter dados muito mais consistentes. Todas as capturas efetuados foram capazes de manter

o seu *buffer* com uma média geral de 38.05 ms de áudio em *buffer*, o desvio padrão geral também foi bastante baixo com oscilação de 2.38 ms. Dessa forma, o teste conseguiu manter os dados em um bom limiar em todas as capturas.

Ao analisar os gráficos de *timeline*, foi possível confirmar o que é descrito pelo histograma, ou seja, não houve perdas severas de pacotes nas capturas. Dessa forma, não se pode afirmar que o aumento no número de dispositivos na rede é um fator que causa sobrecarga no sistema.

4.2.3 Experimento 03

O terceiro experimento realizado lida com a seguinte variação dos fatores:

- F1: 3,5 metros
- F2: 2 roteadores
- F3: 1 dispositivo móvel

A Tabela 4.4 apresenta os valores encontrados durante cada uma das quatro capturas efetuadas para o terceiro experimento.

Tabela 4.4: Capturas do terceiro experimento

Captura	Média de Ocupação (ms)	Desvio Padrão (ms)	Ocupação Mínima (ms)	Ocupação Máxima (ms)	Histo-grama	*Timeline*
A	38.85	1.90	4	40	Figura B.9	Figura C.9
B	38.48	2.11	4	40	Figura B.10	Figura C.10
C	38.60	2.91	4	40	Figura B.11	Figura C.11
D	38.17	2.30	4	40	Figura B.12	Figura C.12

Ao analisar os dados dos histogramas, do gráfico de *timeline* e realizar a comparação junto à Tabela 4.4, é possível obter as seguintes conclusões sobre o terceiro experimento.

Todas as capturas efetuadas foram capazes de manter o seu *buffer* com uma média inferior a 39 ms, ou seja, para esse experimento a solução foi capaz de mitigar os efeitos de *jitter* com maior eficácia, como pode ser observado pelo valor obtido do desvio padrão.

Destaca-se que a captura C apresentou um desvio padrão um pouco maior que o das demais capturas. Ao visualizar o seu gráfico de *timeline*, é possível perceber uma forte oscilação na alocação de dados no *buffer* em aproximadamente 585 segundos da captura, onde duas barras brancas são facilmente visualizadas, indicando que houve uma pausa repentina da chegada de dados no dispositivo móvel.

Novamente têm-se um experimento com bons valores de média, desvio padrão e ocupação, logo, não é possível inferir que o número de redes causou uma influência sobre o envio dos dados.

4.2.4 Experimento 04

O quarto experimento realizado lida com a seguinte variação dos fatores:

- F1: 3,5 metros
- F2: 2 roteadores
- F3: 3 dispositivos móveis

A Tabela 4.5 apresenta os valores encontrados durante cada uma das quatro capturas efetuadas para o quarto experimento.

Tabela 4.5: Capturas do quarto experimento

Captura	Média de Ocupação (ms)	Desvio Padrão (ms)	Ocupação Mínima (ms)	Ocupação Máxima (ms)	Histo-grama	*Timeline*
A	37.14	2.48	4	40	Figura B.13	Figura C.13
B	38.15	2.32	4	40	Figura B.14	Figura C.14
C	37.91	2.31	4	40	Figura B.15	Figura C.15
D	37.61	2.39	4	40	Figura B.16	Figura C.16

Ao analisar os dados dos histogramas, do gráfico de *timeline* e realizar a comparação junto à Tabela 4.5, é possível constatar que assim como o segundo experimento, apresentado na Subseção 4.2.2, o quarto experimento apresentou dados bastante estáveis, mantendo uma boa proporção entre média e desvio padrão.

Dessa forma, para o experimento com as piores variações a curta distância, que são, múltiplos dispositivos na rede e múltiplas redes gerando interferência, não foi constatada grande variação dos dados, sendo assim, a aplicação tende a se comportar de forma adequada a curta distância independentemente da variação dos demais fatores.

4.2.5 Experimento 05

O quinto experimento realizado lida com a seguinte variação dos fatores:

- F1: 7,3 metros
- F2: 1 roteador
- F3: 1 dispositivo móvel

A Tabela 4.6 apresenta os valores encontrados durante cada uma das quatro capturas efetuadas para o quinto experimento.

Tabela 4.6: Capturas do quinto experimento

Captura	Média de Ocupação (ms)	Desvio Padrão (ms)	Ocupação Mínima (ms)	Ocupação Máxima (ms)	Histo-grama	*Timeline*
A	38.67	2.00	4	40	Figura B.17	Figura C.17
B	38.52	2.05	4	40	Figura B.18	Figura C.18
C	38.32	2.12	4	40	Figura B.19	Figura C.19
D	38.11	2.20	4	40	Figura B.20	Figura C.20

Ao analisar os dados dos histogramas, do gráfico de *timeline* e realizar a comparação junto à Tabela 4.6, é possível obter as seguintes conclusões sobre o quinto experimento.

Assim como a captura anterior, o experimento cinco apresentou dados sem grandes variações, mantendo seu *buffer* em todas as capturas próximo ao limiar superior de 40 ms e apresentando baixo desvio padrão.

Os gráficos de *timeline* também não exibem um problema significativo e, portanto, esse teste também é considerado um cenário em que os resultados obtidos da solução proposta foram de acordo com o esperado.

Dessa forma, o primeiro teste de longa distância evidencia que o distanciamento entre base e dispositivo móvel não causa um mau funcionamento para a solução proposta.

4.2.6 Experimento 06

O sexto experimento realizado lida com a seguinte variação dos fatores:

- F1: 7,3 metros
- F2: 1 roteador
- F3: 3 dispositivos móveis

A Tabela 4.7 apresenta os valores encontrados durante cada uma das quatro capturas efetuadas para o sexto experimento.

Tabela 4.7: Capturas do sexto experimento

Captura	Média de Ocupação (ms)	Desvio Padrão (ms)	Ocupação Mínima (ms)	Ocupação Máxima (ms)	Histo-grama	*Timeline*
A	38.54	2.37	4	40	Figura B.21	Figura C.21
B	38.12	2.34	0	40	Figura B.22	Figura C.22
C	37.96	2.39	4	40	Figura B.23	Figura C.23
D	34.39	2.62	0	40	Figura B.24	Figura C.24

Ao analisar os dados dos histogramas, do gráfico de *timeline* e realizar a comparação junto à Tabela 4.7, é possível obter as seguintes conclusões sobre o sexto experimento.

As capturas de A, B e C apresentaram bons valores de ocupação do *buffer* e baixo desvio padrão, informações que são ratificadas pelos gráficos de *timeline*. Foi possível perceber através dessa análise que o *buffer* é majoritariamente preenchido pela informação, ou seja, um cenário ideal em que os algoritmos podem atuar para manter o *buffer* com dados.

Já a captura D apresentou um declive na ocupação média, cujo valor foi de 34.39 ms. Essa redução no valor médio pode ser visualizada com facilidade no gráfico de *timeline*, pois ocorreu uma pequena variação a partir dos 180 segundos, e os dados caíram de um *buffer* com teto em 40 ms para um teto em 36 ms.

Destaca-se nas capturas B e D presença de drenagem completa, fato que ocasionou uma ocupação mínima de 0 ms de áudio.

Dessa forma, pode existir uma relação entre o aumento no número de dispositivos móveis e a drenagem do buffer de rede, uma vez que, comparado ao experimento cinco, o aumento de dispositivos móveis aparentemente causou problemas junto à transmissão dos dados.

4.2.7 Experimento 07

O sétimo experimento realizado lida com a seguinte variação dos fatores:

- F1: 7,3 metros
- F2: 2 roteadores
- F3: 1 dispositivo móvel

A Tabela 4.8 apresenta os valores encontrados durante cada uma das quatro capturas efetuadas para o sétimo experimento.

Tabela 4.8: Capturas do sétimo experimento

Captura	Média de Ocupação (ms)	Desvio Padrão (ms)	Ocupação Mínima (ms)	Ocupação Máxima (ms)	Histo-grama	*Timeline*
A	37.90	2.29	4	40	Figura B.25	Figura C.25
B	38.93	1.85	4	40	Figura B.26	Figura C.26
C	38.26	2.21	4	40	Figura B.27	Figura C.27
D	38.20	2.23	4	40	Figura B.28	Figura C.28

Ao analisar os dados dos histogramas, do gráfico de *timeline* e realizar a comparação junto à Tabela 4.8, é possível obter as seguintes conclusões sobre o sétimo experimento.

Assim como o quinto experimento, Seção 4.2.5, o sétimo experimento apresentou dados sem grandes variações, mantendo seu *buffer* em todas as capturas próximo ao limiar superior de 40 ms e apresentando baixo desvio padrão.

Os gráficos de *timeline* também não exibem um problema significativo e, portanto, esse teste também é considerado um cenário onde a solução proposta fez o seu trabalho como

esperado. Destaca-se a segunda captura do experimento, que apresentou baixo desvio padrão e ocupação quase que total durante a execução.

4.2.8 Experimento 08

O oitavo experimento realizado lida com a seguinte variação dos fatores:

- F1: 7,3 metros
- F2: 2 roteadores
- F3: 3 dispositivos móveis

A Tabela 4.9 apresenta os valores encontrados durante cada uma das quatro capturas efetuadas para o oitavo experimento.

Tabela 4.9: Capturas do oitavo experimento

Captura	Média de Ocupação (ms)	Desvio Padrão (ms)	Ocupação Mínima (ms)	Ocupação Máxima (ms)	Histo-grama	*Timeline*
A	38.32	2.61	4	40	Figura B.29	Figura C.29
B	37.32	2.36	0	40	Figura B.30	Figura C.30
C	37.71	2.50	4	40	Figura B.31	Figura C.31
D	37.90	3.23	0	40	Figura B.32	Figura C.32

Ao analisar os dados dos histogramas, do gráfico de *timeline* e realizar a comparação junto à Tabela 4.9, é possível obter as seguintes conclusões sobre o sétimo experimento.

As capturas A e C apresentaram uma média de ocupação extremamente relevante, pois os dados oscilaram muito próximo ao limite superior de 40 ms, com desvio padrão próximo aos 2.55 ms. A captura A apresenta uma pequena região que representa perda de pacotes próximo aos 280 segundo da exibição, porém, não foi observado nenhum *underflow*.

Já as capturas B e D ainda apresentam uma boa média, porém a captura D apresenta um desvio padrão muito pior do que o que foi observado nos testes anteriores, e também ressalta-se que foi possível observar alguns *underflows*, ou seja, os dados foram totalmente

consumidos do *buffer* de *playback* em ambas as capturas. Também foi possível escutar pequenos e breves ruídos durante a reprodução do áudio, o que não foi constatado nas demais capturas. Ressalta-se que esses ruídos são breves e na maioria das vezes não são sequer percebidos, ou seja, apenas pessoas com excelente acuidade auditiva seriam capazes de minimamente identificá-los.

O gráfico de *timeline* da captura C não apresenta grandes oscilações, já o da captura D apresenta forte variação nos primeiros 100 segundos da apresentação.

4.2.9 Considerações Finais

Nesse capítulo foi apresentado um processo de experimentação para a solução proposta neste trabalho. Esse processo englobou a definição, planejamento e execução de uma série de experimentos conduzidos para testar as hipóteses gerais deste trabalho apresentadas no Capítulo 1. Nesse processo, hipóteses específicas foram propostas para avaliar o comportamento da aplicação a partir de diferentes cenários de execução.

Os resultados do experimento demonstram que uma das duas hipóteses formuladas pode ser rejeitada, indicando que a solução proposta é capaz de reproduzir o conteúdo acessível em tempo menor ou igual ao limiar de 40 milissegundos. Assim, trazendo conforto para o espectador durante a exibição do conteúdo acessível, mantendo-se o sincronismo da mídia de áudio abaixo do valor de *lip sync*.

Ainda, a hipótese de que é possível evitar *underflow* no *buffer* não foi rejeitada, uma vez que nas Subseções 4.2.8 e 4.2.6 pode-se constatar que ocorreram situações em que o *buffer* foi totalmente esvaziado. Porém, é importante ressaltar que o FAST é capaz de mitigar muitas destas falhas ao utilizar as suas estratégias de FEC, redundância temporal e algoritmo adaptativo, pois, conforme pode-se observar nos demais experimentos, o *buffer* foi quase que totalmente drenado, porém a solução foi capaz de mantê-lo com ao menos um pacote armazenado, indicado pelos 4 ms de áudio que é apresentado nos histogramas.

Através da análise dos dados, pode-se assumir erroneamente que o aumento na quantidade de dispositivos móveis nos testes é o causador da drenagem dos dados no *buffer*. É sabido que a base *wireless* utilizada nesse trabalho possui largura de banda suficiente para transmitir os dados a um número muito maior do que dois *smartphones*. Além disso, os experimentos a 3,5 metros invalidam esse raciocínio. Assim, descarta-se a hipótese de que

o aumento do número de dispositivos moveis é o principal causador de *underflow*. Porém, deve-se salientar que o aumento na distância entre a base de transmissão e o dispositivo de reprodução pode ser a causa da drenagem do *buffer*. A redução de dados deve-se ao fato de que quanto maior for a distância, mais propício é o sinal a interferência de demais equipamentos que podem estar transmitidos dados no meio, como por exemplo, redes *wireless* de terceiros, dispositivos *bluetooth* e ondas de rádio. Dessa forma, em cenário ideal, como salas de cinema onde a quantidade de redes em uma mesma área tende a ser reduzido a solução deve, de forma geral, comportar-se adequadamente.

No próximo capítulo, serão apresentadas as conclusões sobre o presente trabalho, incluindo a discussão das principais contribuições e propostas de trabalho futuro.

Capítulo 5

Considerações Finais

Neste trabalho, foi proposta uma solução denominada FAST para transmissão de áudio com baixa latência em redes Wi-Fi para *smartphones*. A solução foi desenvolvida com o intuito de auxiliar usuários com deficiência visual e auditiva a terem acesso a espetáculos e, particularmente, a filmes que são exibidos em salas de cinema. Na proposta, o áudio do filme é capturado, processado, transmitido e reproduzido em uma latência máxima de 40 ms, que é o limiar entre perceber ou não a divergência entre as trilhas de áudio e vídeo de uma transmissão multimídia.

Dessa forma, esse é o principal desafio desse trabalho, cuja questão de pesquisa estudada foi: "Como transmitir áudio para grupos de usuários utilizando dispositivos móveis como receptores com latência entre captura e exibição igual ou inferior a 40 ms?"

Para responder a questão de pesquisa formulada, foi desenvolvida uma solução cliente-servidor, que é responsável por transmitir o áudio do espetáculo e reproduzi-lo de forma apropriada, a qual utiliza redundância temporal, um algoritmo de FEC e um novo algoritmo adaptativo, capaz de amortizar os efeitos causados pelas variações na rede.

Durante a pesquisa acerca das tecnologias candidatas que seriam utilizadas pela solução, o presente trabalho elaborou um documento de revisão sistemática, que serve como referência para os demais pesquisadores que desejam engajar na área da transmissão de áudio em baixa latência. Esse documento enumera os diversos trabalhos que lidam com diferentes tecnologias e métodos com o propósito de atingir baixíssima latência.

Adicionalmente, foi desenvolvido um processo de experimentação, cujo propósito foi a simulação de situações encontradas pelos espectadores em um ambiente de cinema digital.

Com a execução dos experimentos, foi possível realizar uma análise crítica sobre os dados e constatar que a solução, de fato, mantém o *playback* em limiar menor ou igual aos 40 ms propostos, além de mitigar significativamente os efeitos de *jitter* e colisão de pacotes através dos algoritmos propostos. Além disso, a solução proposta demonstra que é madura o suficiente para competir com outras tecnologias do mercado, como a transmissão de dados por infravermelho e a transmissão por modulação em frequência.

Por fim, este trabalho possui contribuição com uma produção técnico-científica através da patente de inovação registrada junto ao INPI (Instituto Nacional da Propriedade Industrial) sob o número de processo BR 10 2021 000210 7.

5.1 Proposta de Trabalhos Futuros

Apesar da solução proposta possuir escopo limitado, é possível definir novos desafios de pesquisa. Uma das possibilidades é, por exemplo, analisar a performance computacional dos *smartphones* disponíveis no mercado no tocante à captura e processamento de dados. Esse desafio busca avaliar a performance do adaptador de rede, do *hardware* e *software* de gerenciamento de áudio, através da análise do RTT (*Round Trip Time*) entre a solicitação de reprodução do áudio e a sua saída em um fone de ouvido. Dessa forma, seria possível através desses dados obtidos definir um *hardware* que permita máxima performance ao receber e reproduzir trilhas de áudio.

Uma segunda proposta de trabalho futuro é a análise da performance de bases *wireless*, avaliando a efetividade na entrega de pacotes, buscando compreender se tecnologias como MU-MIMO (*Multi User Multiple Input, Multiple Output*) e OFDMA (*Orthogonal Frequency Division Multiplexing*) trazem benefícios contundentes em transmissões multiusuário. Além disso, pode-se desenvolver um roteador próprio através do uso de um sistema operacional especializado como OpenWRT e comparar a sua performance com a do roteador utilizado na solução proposta.

Por fim, sugere-se que uma bateria de testes seja realizada com um grupo de indivíduos formado por deficientes visuais e por pessoas com diferentes graus de acuidade auditiva e ruídos de *underflow* sejam inseridos propositalmente. Dessa forma, será possível calcular o valor de MOS (*Mean Opinion Score*) do grupo. Assim, analisar-se-á se os *dropouts* que

acontecerão, comentados no Experimento 4.2.8, causam desconforto para os usuários.

Bibliografia

[Ancine - Instrução Normativa N 128, 2016]ANCINE - Instrução Normativa N 128, 2016. https://antigo.ancine.gov.br/pt-br/legislacao/instrucoes-normativas-consolidadas/instru-o-normativa-n-128-de-13-de-setembro-de-2016. Accessed: 2021-01-14.

[Banerji e Chowdhury 2013]BANERJI, S.; CHOWDHURY, R. On ieee 802.11: Wireless lan technology. *International Journal of Mobile Network Communications & Telematics*, v. 3, 07 2013.

[Chen et al. 2004]Chen, L. . et al. Audio streaming over bluetooth: an adaptive arq timeout approach. In: *24th International Conference on Distributed Computing Systems Workshops, 2004. Proceedings.* [S.l.: s.n.], 2004. p. 196–201.

[Crow et al. 1997]Crow, B. P. et al. Ieee 802.11 wireless local area networks. *IEEE Communications Magazine*, v. 35, n. 9, p. 116–126, 1997.

[Deshpande 2006]DESHPANDE, S. Adaptive low-bitrate streaming over ieee 802.15.4 low rate wireless personal area networks (lr-wpan) based on link quality indication. In: *Proceedings of the 2006 International Conference on Wireless Communications and Mobile Computing*. New York, NY, USA: Association for Computing Machinery, 2006. (IWCMC '06), p. 863–868. ISBN 1595933069. Disponível em: <https://doi.org/10.1145/1143549.1143722>.

[Doherty, Curran e Mckevitt 2013]DOHERTY, J.; CURRAN, K.; MCKEVITT, P. A self-similarity approach to repairing large dropouts of streamed music. *ACM Trans. Multimedia Comput. Commun. Appl.*, Association for Computing Machinery,

New York, NY, USA, v. 9, n. 3, jul. 2013. ISSN 1551-6857. Disponível em: <https://doi.org/10.1145/2487268.2487273>.

[Doherty, Curran e McKevitt 2015]DOHERTY, J.; CURRAN, K.; MCKEVITT, P. Pattern matching techniques for replacing missing sections of audio streamed across wireless networks. *ACM Trans. Intell. Syst. Technol.*, Association for Computing Machinery, New York, NY, USA, v. 6, n. 2, mar. 2015. ISSN 2157-6904. Disponível em: <https://doi.org/10.1145/2663358>.

[Hardman et al. 2006]HARDMAN, V. et al. Reliable audio for use over the internet. In: . [S.l.: s.n.], 2006.

[Jelassi e Youssef 2006]JELASSI, S.; YOUSSEF, H. Adaptive playback algorithm for interactive audio streaming over wireless ad-hoc networks. In: *Proceedings of the 2006 International Conference on Wireless Communications and Mobile Computing*. New York, NY, USA: Association for Computing Machinery, 2006. (IWCMC '06), p. 218–226. ISBN 1595933069. Disponível em: <https://doi.org/10.1145/1143549.1143594>.

[Kovacevic, Samardzija e Temerinac 2009]Kovacevic, J.; Samardzija, D.; Temerinac, M. Joint coding rate control for audio streaming in short range wireless networks. *IEEE Transactions on Consumer Electronics*, v. 55, n. 2, p. 486–491, 2009.

[Lan e Li 2010]Lan, K.; Li, M. Feasibility study of using fm radio for data transmission in a vehicular network. In: *2010 International Computer Symposium (ICS2010)*. [S.l.: s.n.], 2010. p. 55–60.

[Lin et al. 2004]LIN, C.-h. et al. Supporting real-time speech on wireless ad hoc networks: Inter-packet redundancy, path diversity, and multiple description coding. In: *Proceedings of the 2nd ACM International Workshop on Wireless Mobile Applications and Services on WLAN Hotspots*. New York, NY, USA: Association for Computing Machinery, 2004. (WMASH '04), p. 11–20. ISBN 1581138776. Disponível em: <https://doi.org/10.1145/1024733.1024736>.

[Mafra e Travassos 2006]MAFRA, S.; TRAVASSOS, G. Estudos primários e secundários apoiando a busca por evidência em engenharia de software. In: . [S.l.: s.n.], 2006.

[Mangold et al. 2003]Mangold, S. et al. Analysis of ieee 802.11e for qos support in wireless lans. *IEEE Wireless Communications*, v. 10, n. 6, p. 40–50, 2003.

[McKinley e Gaurav 2000]MCKINLEY, P. K.; GAURAV, S. Experimental evaluation of forward error correction on multicast audio streams in wireless lans. In: *Proceedings of the Eighth ACM International Conference on Multimedia*. New York, NY, USA: Association for Computing Machinery, 2000. (MULTIMEDIA '00), p. 416–418. ISBN 1581131984. Disponível em: <https://doi.org/10.1145/354384.376304>.

[Opus Codec, 2012]OPUS Codec, 2012. `https://auphonic.com/blog/2012/09/26/opus-revolutionary-open-audio-codec-podcasts-and-internet-audio/#:~:text=Opus%20is%20a%20lossy%20audio,as%20such%20is%20royalty%2Dfree.&text=Moreover%2C%20this%20quality%20is%20achieved,interactive%20music%20and%20speech%20transmission`. Accessed: 2021-01-14.

[R37-2007 2007]R37-2007, E. R. The relative timing of the sound and vision components of a television signal. In: . [S.l.: s.n.], 2007.

[Raju et al. 2006]Raju, G. V. et al. On supporting real-time speech over ad hoc wireless networks. In: *2006 14th IEEE International Conference on Networks*. [S.l.: s.n.], 2006. v. 2, p. 1–6.

[Ravindran e Bansal 1993]RAVINDRAN, K.; BANSAL, V. Delay compensation protocols for synchronization of multimedia data streams. *IEEE Trans. on Knowl. and Data Eng.*, IEEE Educational Activities Department, USA, v. 5, n. 4, p. 574–589, ago. 1993. ISSN 1041-4347. Disponível em: <https://doi.org/10.1109/69.234770>.

[Valin et al. 2013]VALIN, J. et al. The opus codec. AES Convention, fev. 2013.

[Valin e Terriberry 2012]VALIN, K. V. J.; TERRIBERRY, T. *A Border Gateway Protocol 4 (BGP-4)*. [S.l.], September 2012. 1-326 p. Disponível em: <https://tools.ietf.org/html/rfc6716>.

[Wang et al. 2003]WANG, Y. et al. Content-based uep: A new scheme for packet loss recovery in music streaming. In: *Proceedings of the Eleventh ACM International*

Conference on Multimedia. New York, NY, USA: Association for Computing Machinery, 2003. (MULTIMEDIA '03), p. 412–421. ISBN 1581137222. Disponível em: <https://doi.org/10.1145/957013.957099>.

[Wohlin et al. 2012]WOHLIN, C. et al. *Experimentation in Software Engineering*. [S.l.]: Springer Publishing Company, Incorporated, 2012. ISBN 3642290434.

Apêndice A

Log de captura

B 0 40 4

B 3597 40 4

B 7241 40 4

B 10854 40 4

B 14454 40 4

B 18083 40 4

B 21812 40 4

B 25424 40 4

B 29081 40 4

B 32717 40 4

B 36341 40 8

B 39974 40 16

B 43589 40 20

B 47199 40 24

B 50908 40 24

B 54607 40 28

B 58204 40 36

B 61801 40 40

B 65407 40 36

B 69042 40 36

Apêndice B

Histograma dos Experimentos

B.1 Experimento 01

Figura B.1: Histograma da primeira captura do primeiro experimento

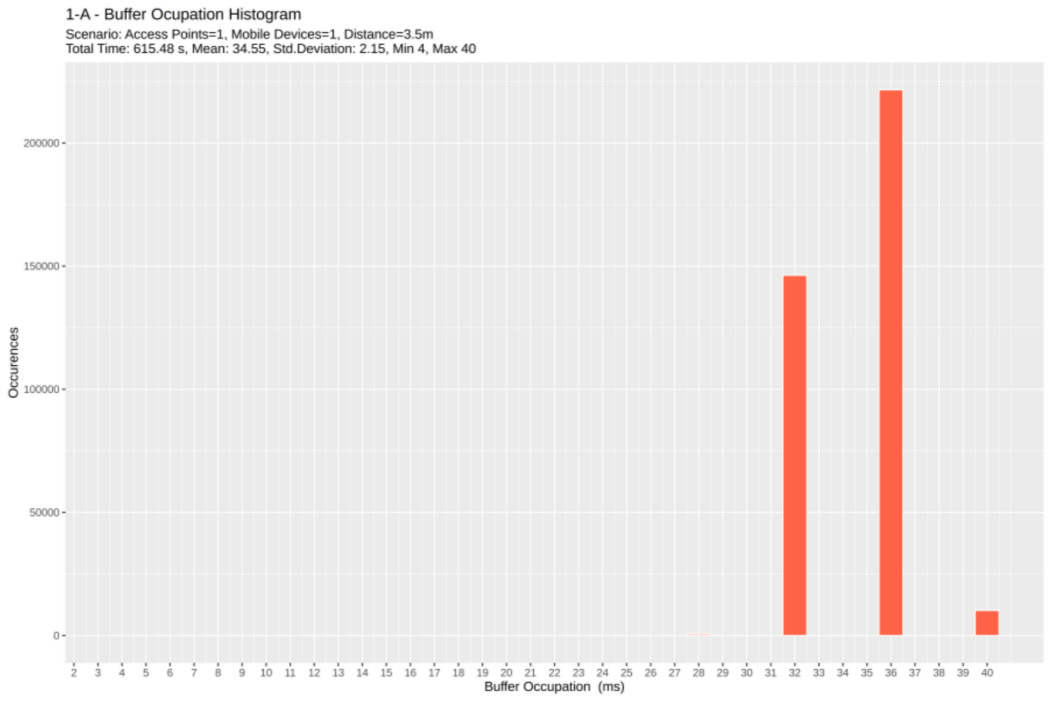


Figura B.2: Histograma da segunda captura do primeiro experimento

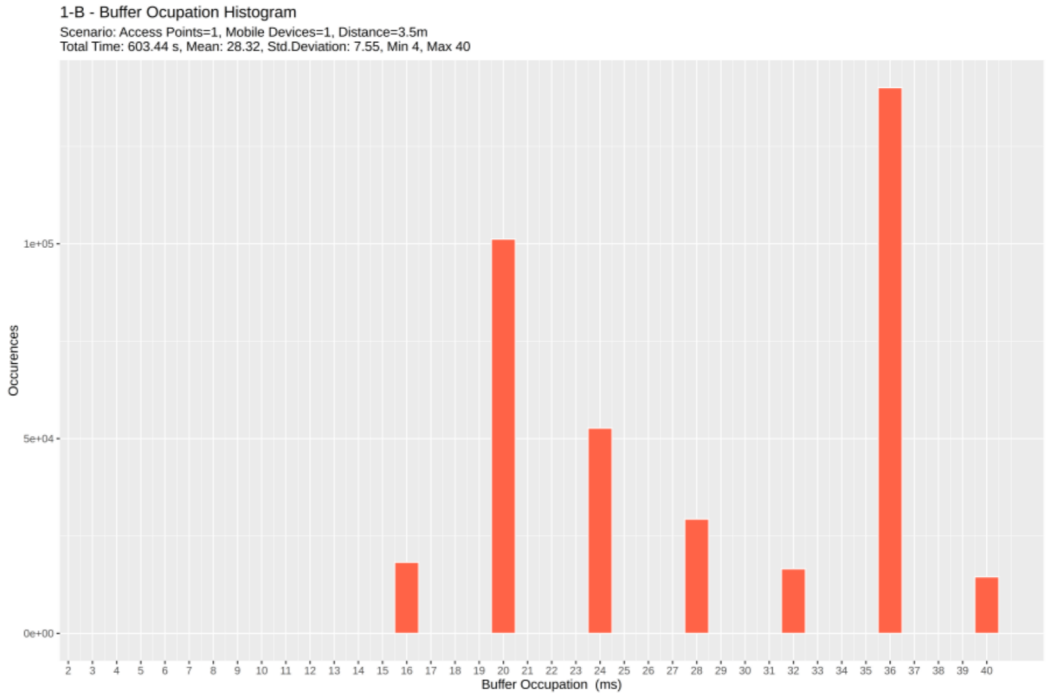


Figura B.3: Histograma da terceira captura do primeiro experimento

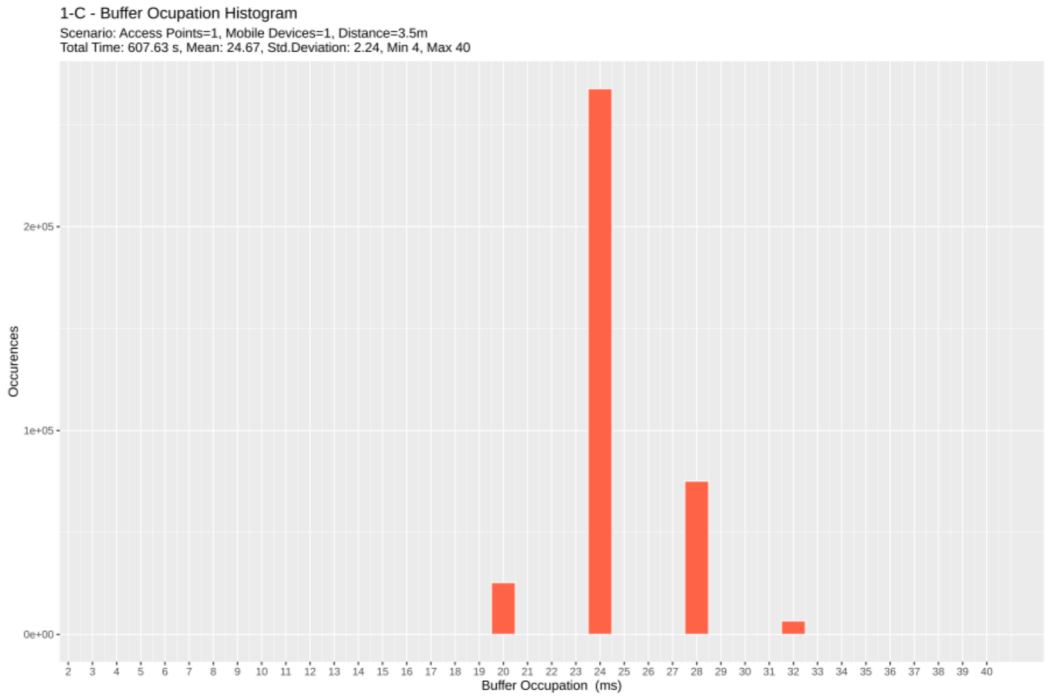
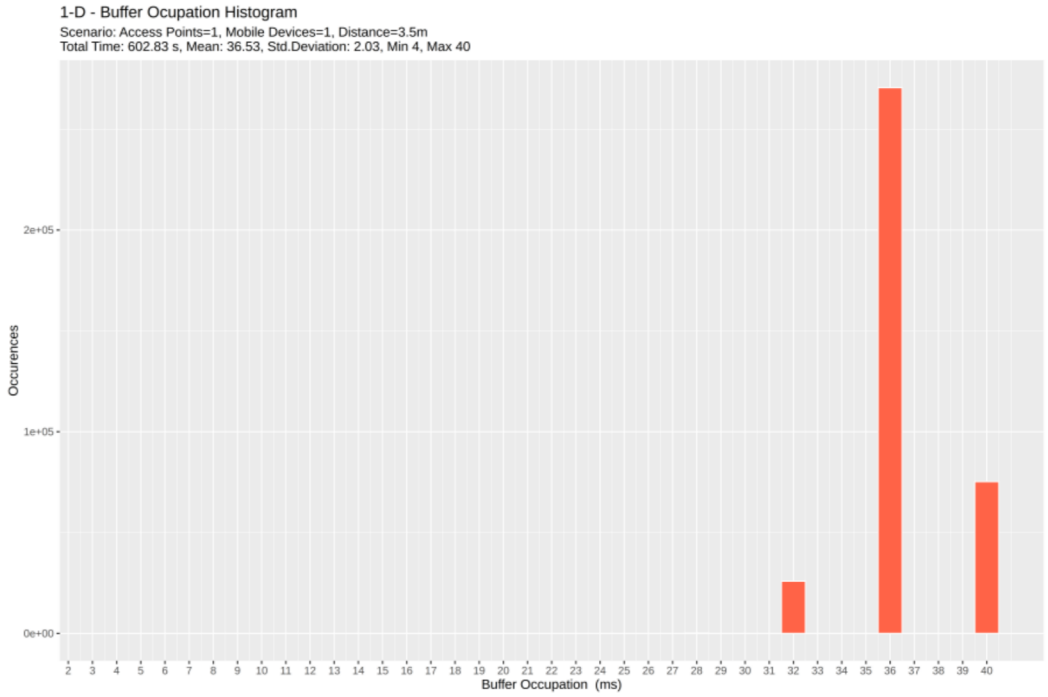


Figura B.4: Histograma da quarta captura do primeiro experimento



B.2 Experimento 02

Figura B.5: Histograma da primeira captura do segundo experimento

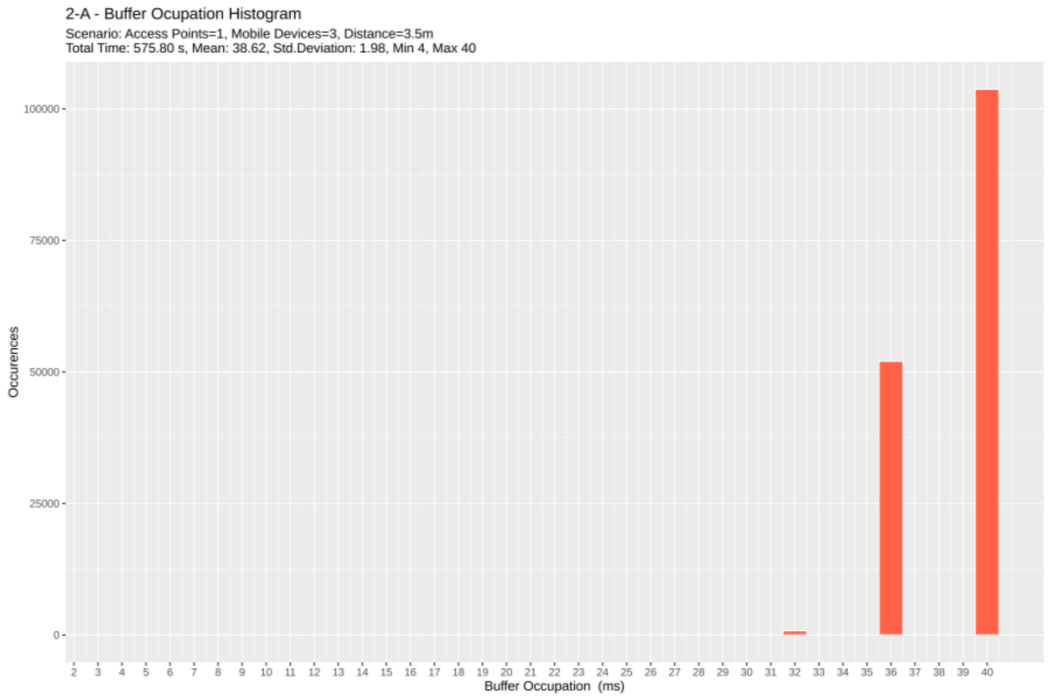


Figura B.6: Histograma da segunda captura do segundo experimento

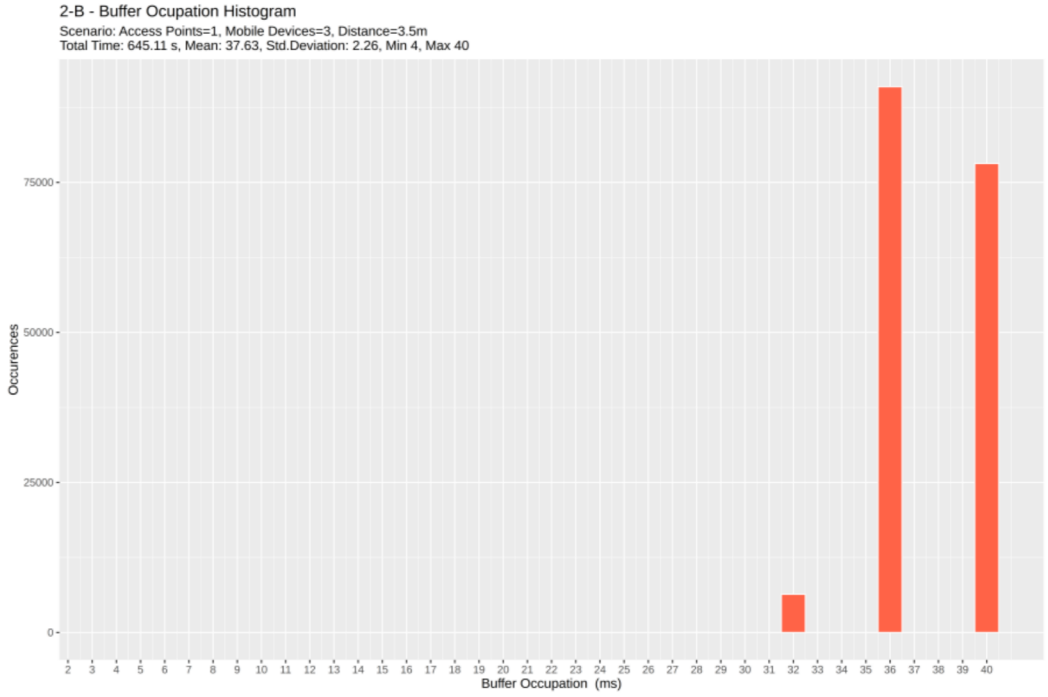


Figura B.7: Histograma da terceira captura do segundo experimento

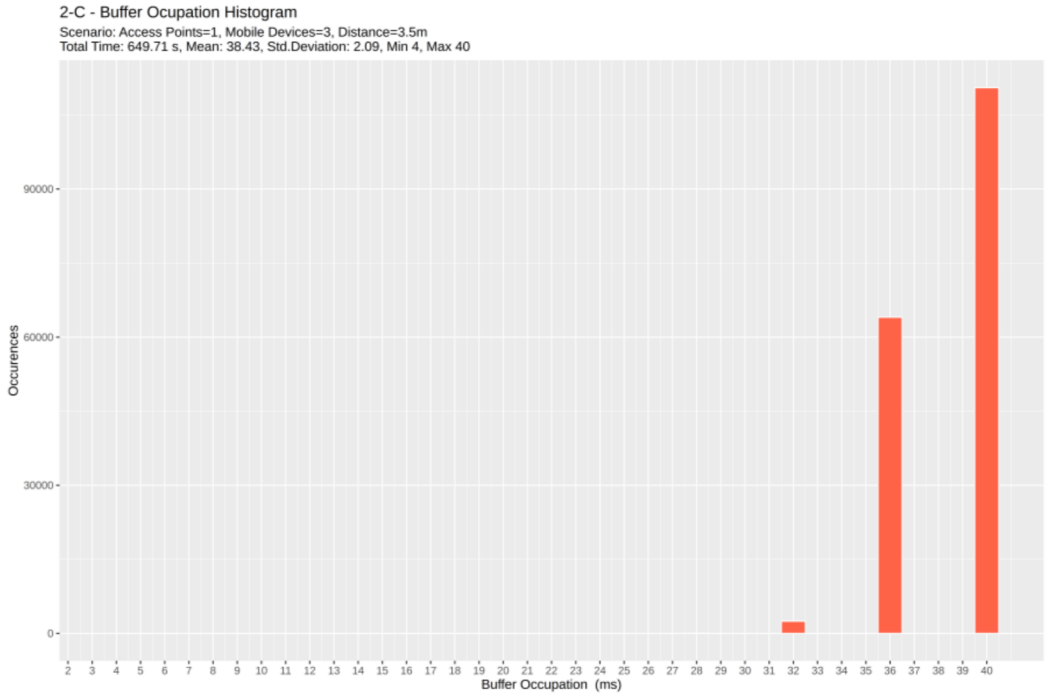
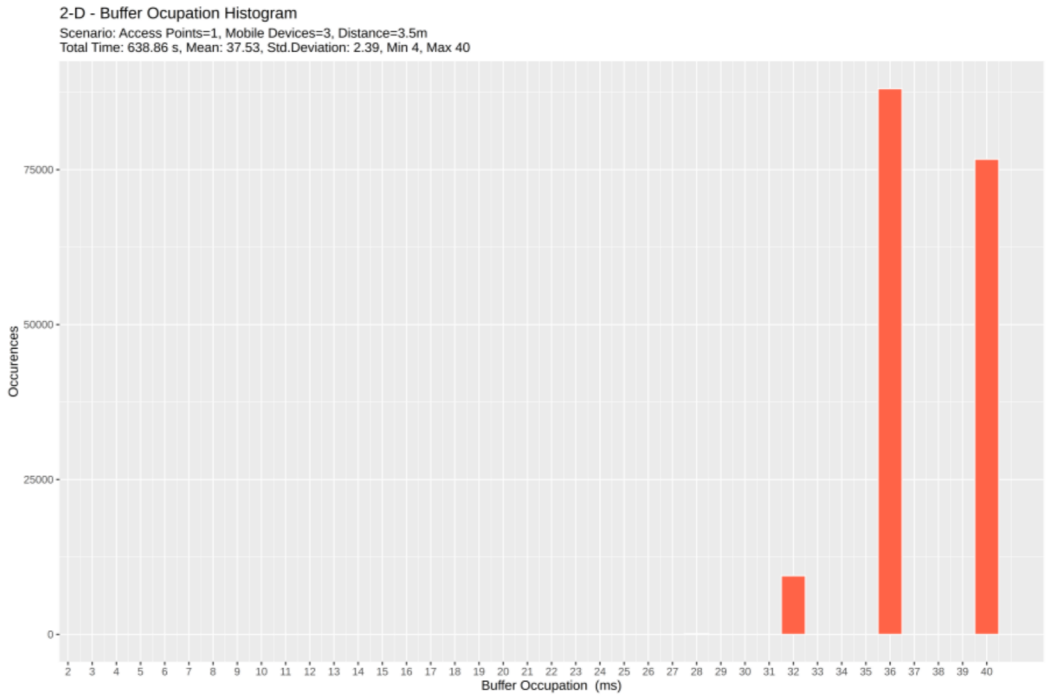


Figura B.8: Histograma da quarta captura do segundo experimento



B.3 Experimento 03

Figura B.9: Histograma da primeira captura do terceiro experimento

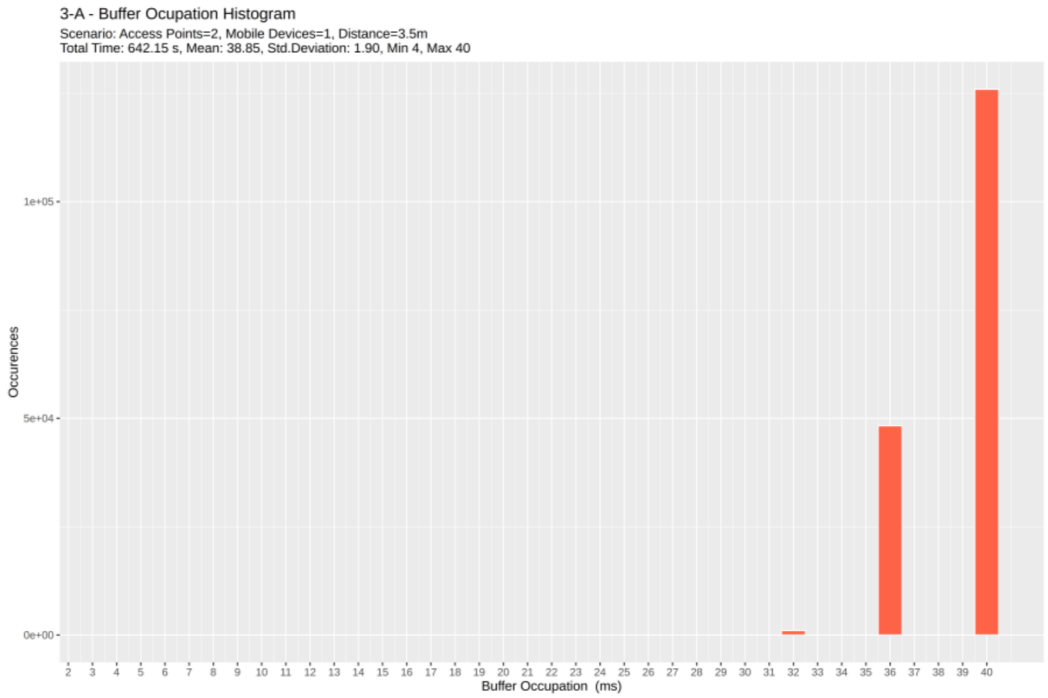


Figura B.10: Histograma da segunda captura do terceiro experimento

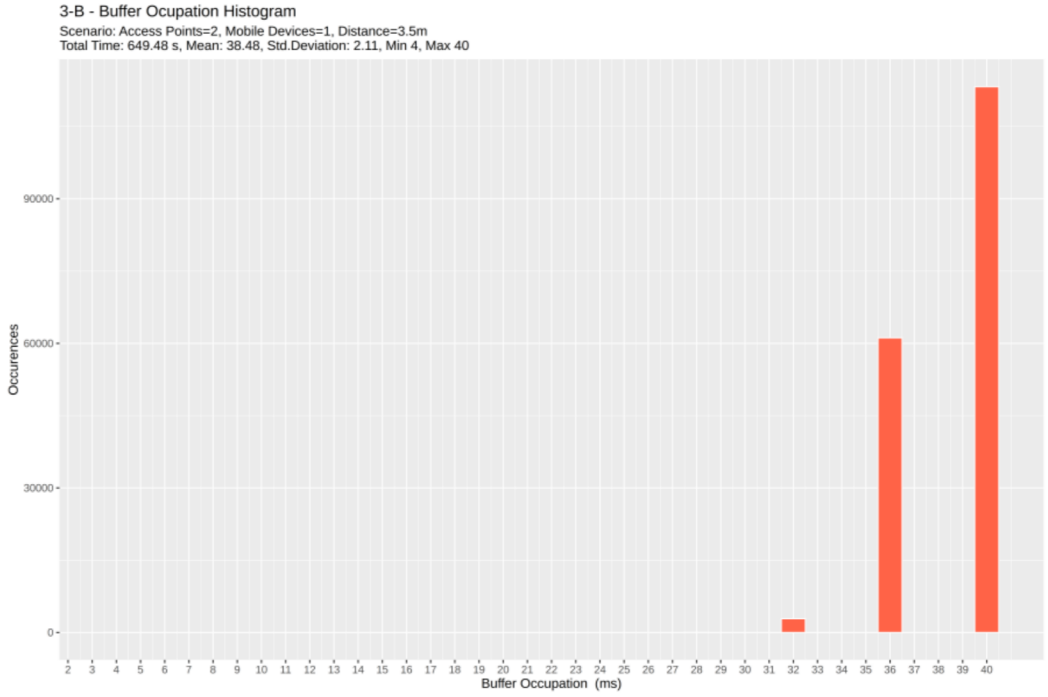


Figura B.11: Histograma da terceira captura do terceiro experimento

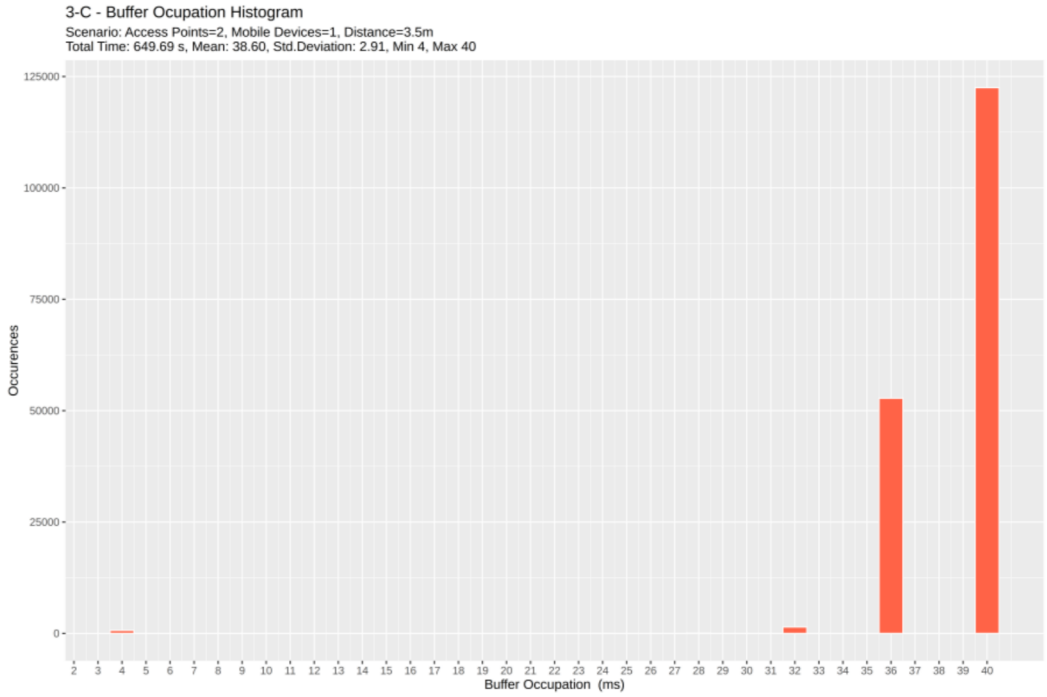
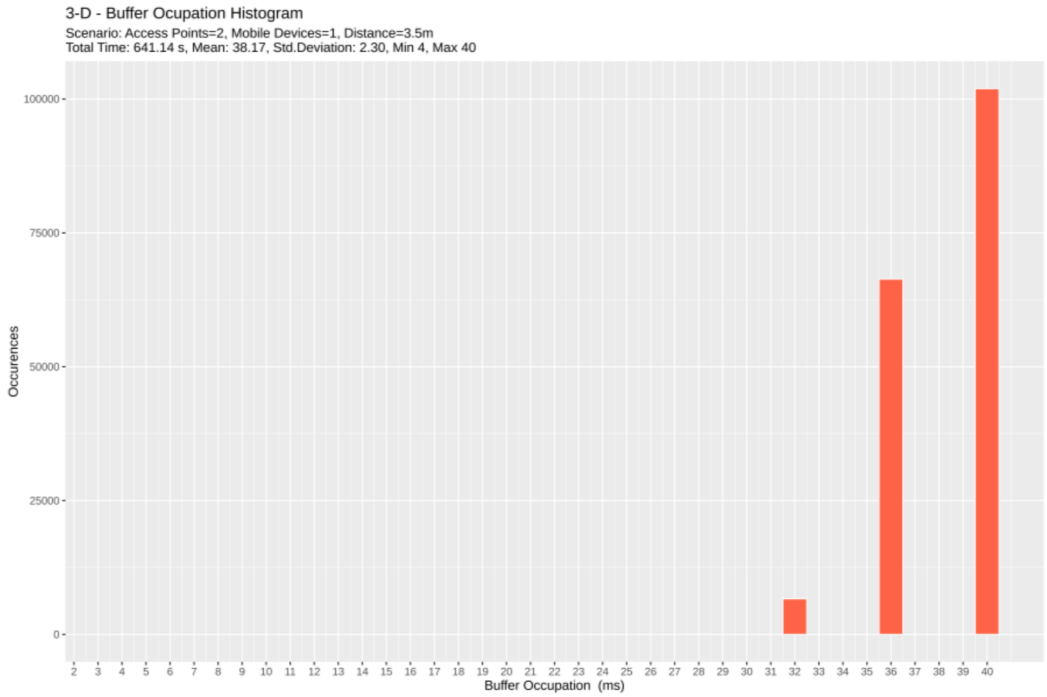


Figura B.12: Histograma da quarta captura do terceiro experimento



B.4 Experimento 04

Figura B.13: Histograma da primeira captura do quarto experimento

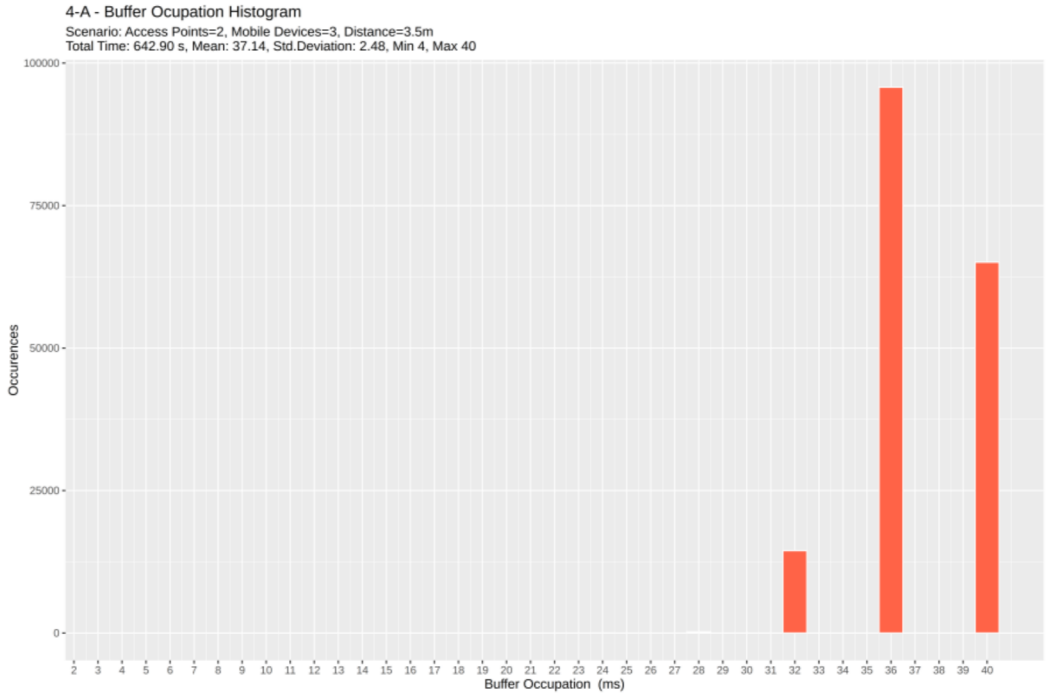


Figura B.14: Histograma da segunda captura do quarto experimento

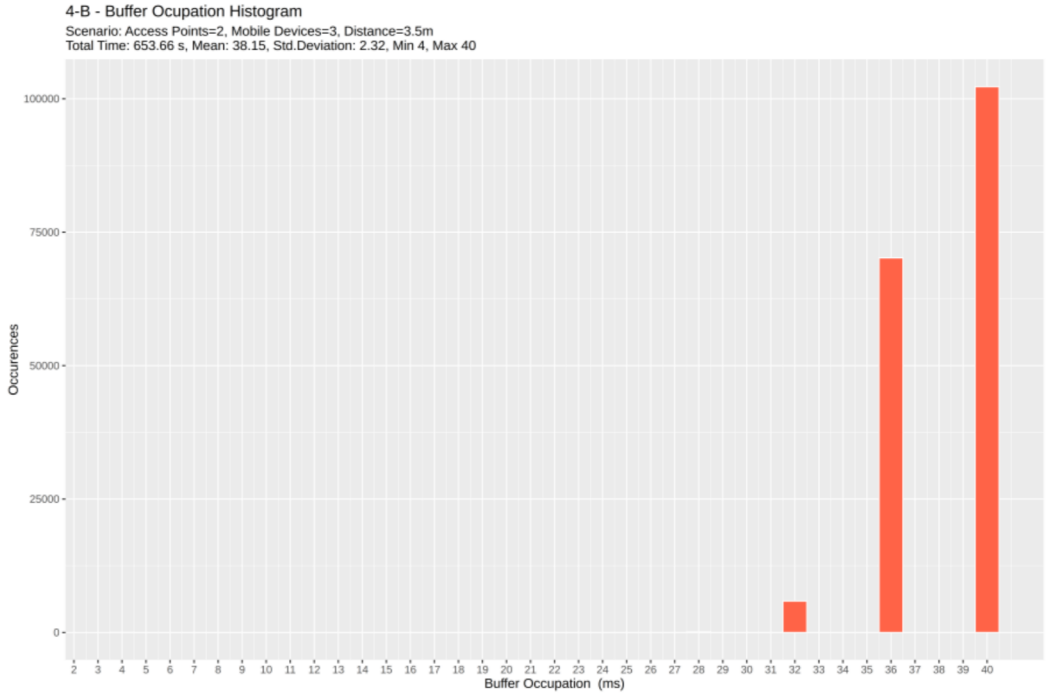


Figura B.15: Histograma da terceira captura do quarto experimento

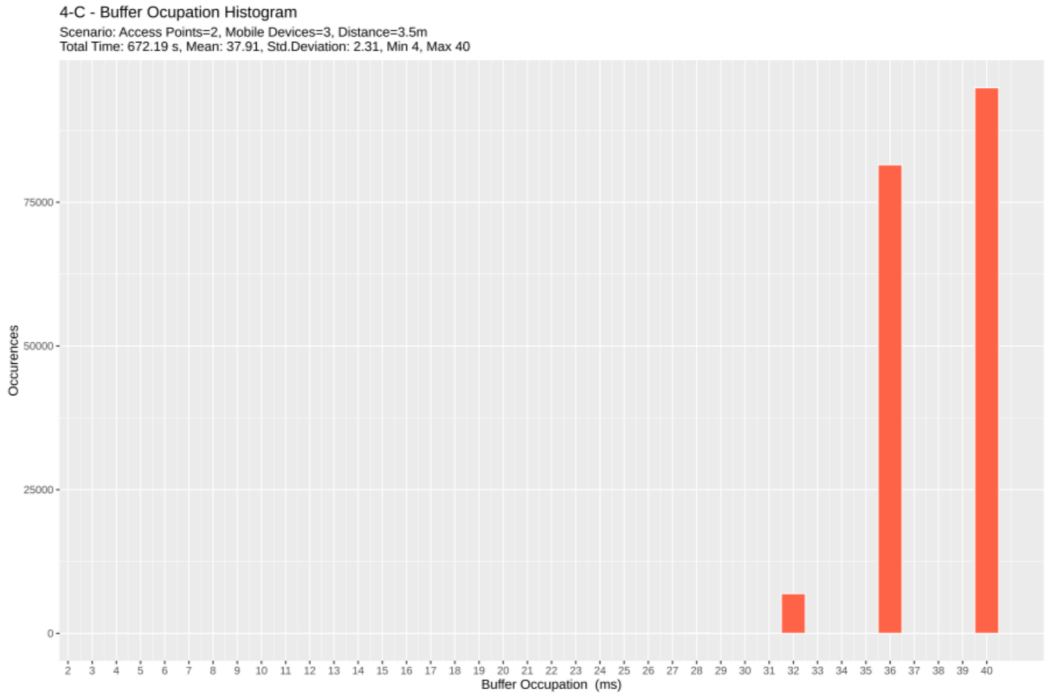
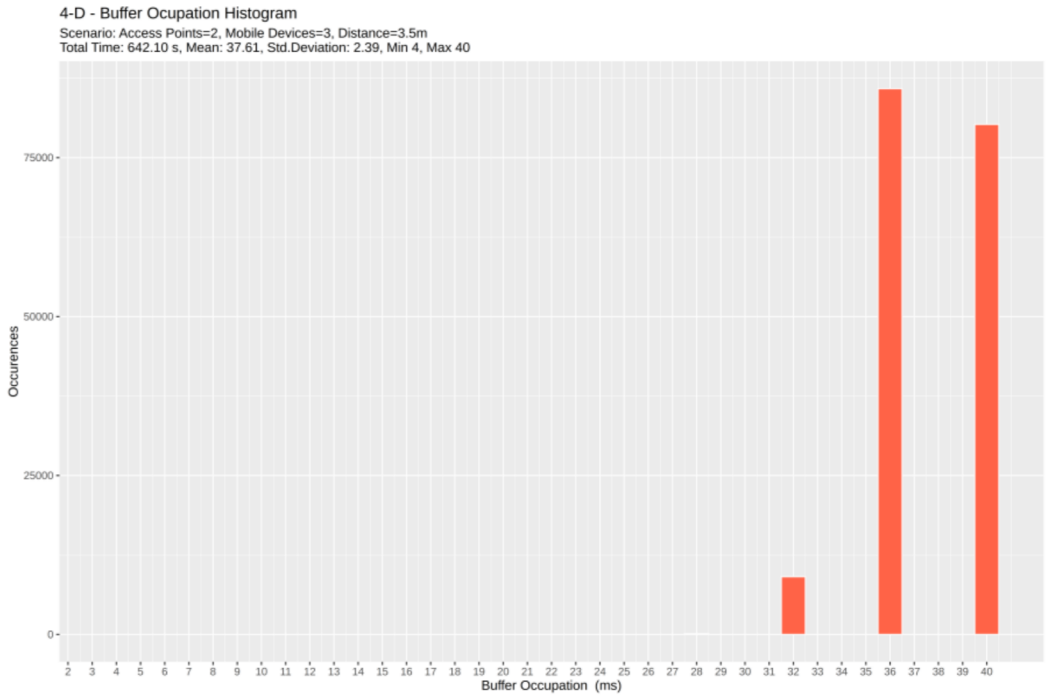


Figura B.16: Histograma da quarta captura do quarto experimento



B.5 Experimento 05

Figura B.17: Histograma da primeira captura do quinto experimento

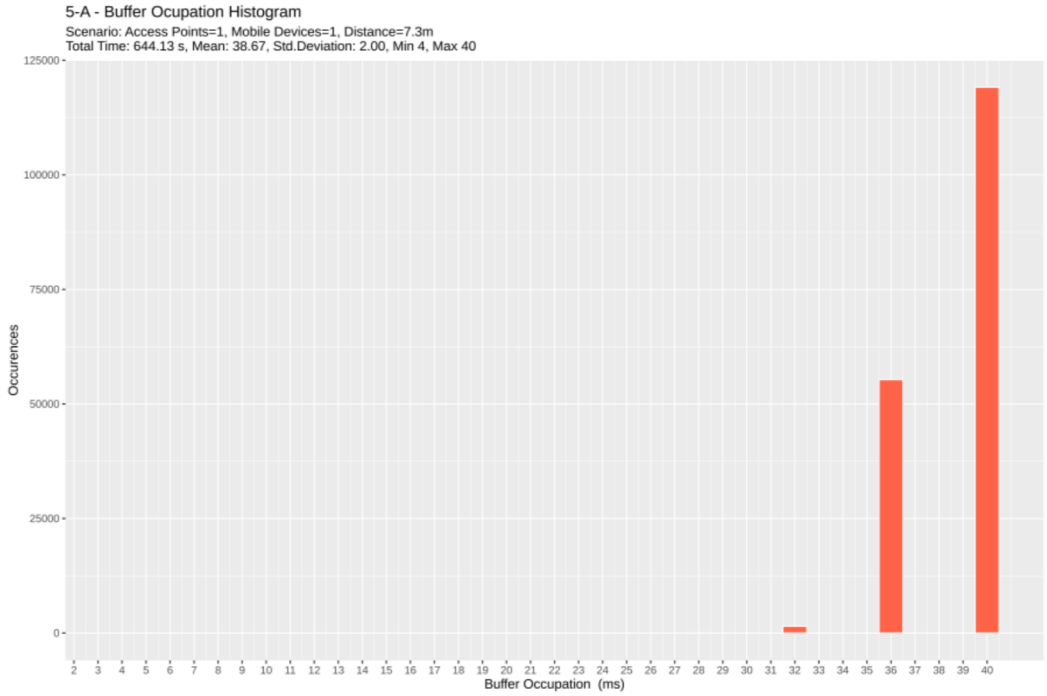


Figura B.18: Histograma da segunda captura do quinto experimento

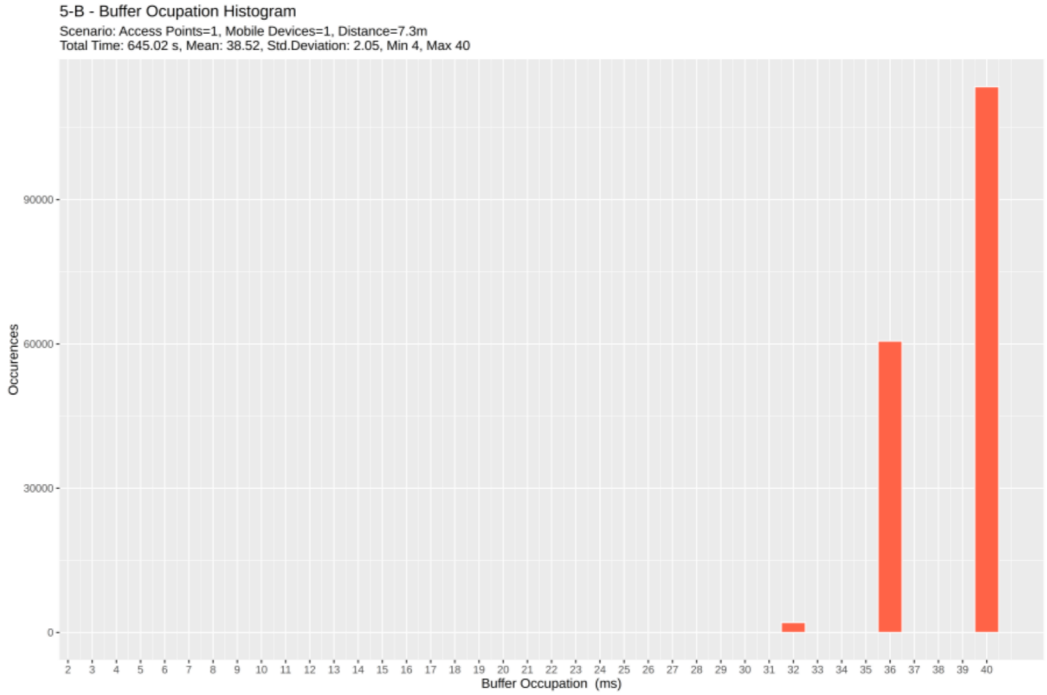


Figura B.19: Histograma da terceira captura do quinto experimento

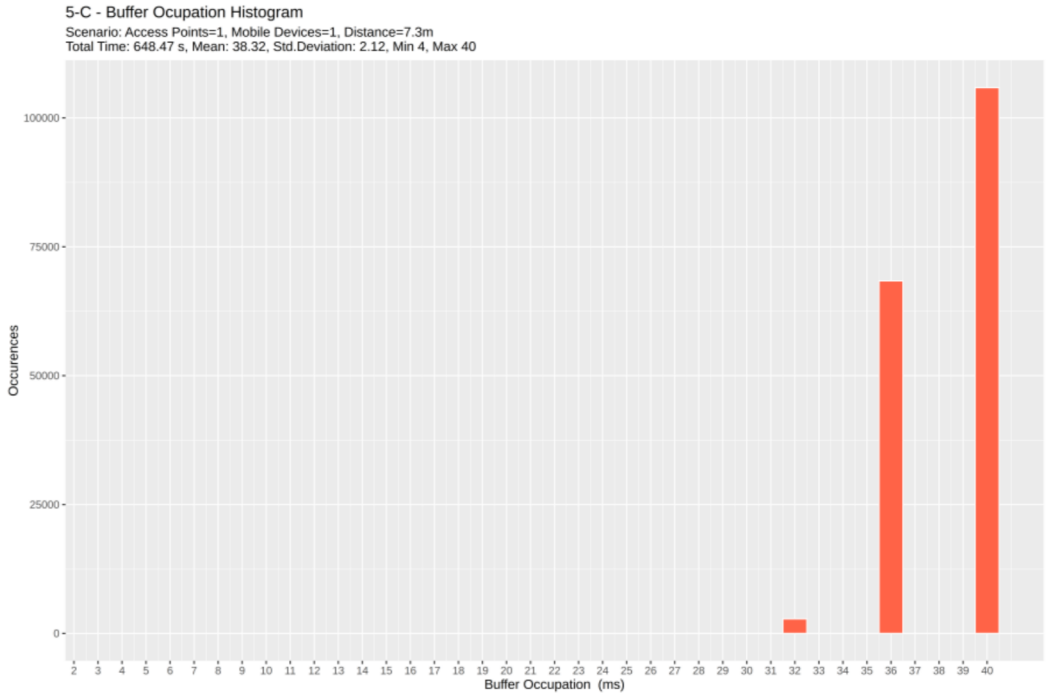
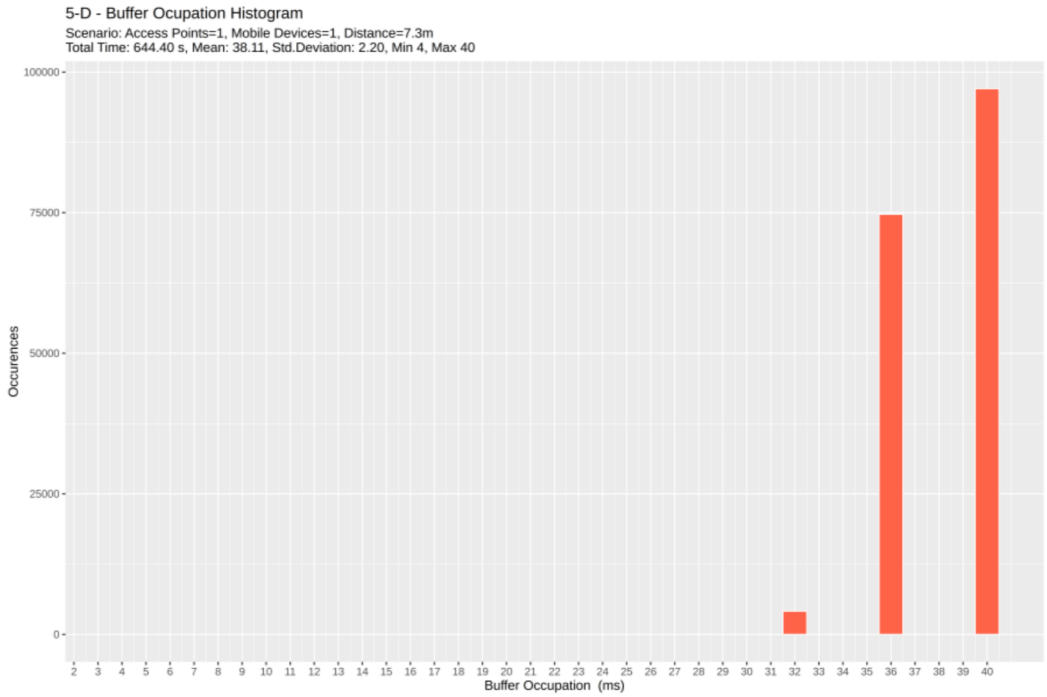


Figura B.20: Histograma da quarta captura do quinto experimento



B.6 Experimento 06

Figura B.21: Histograma da primeira captura do sexto experimento

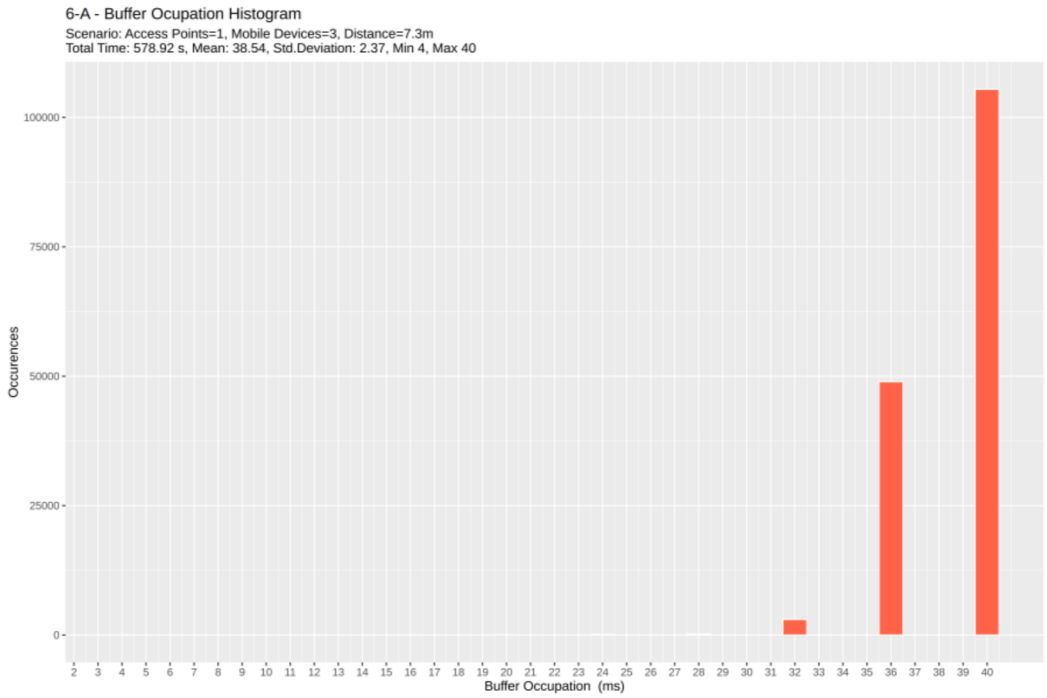


Figura B.22: Histograma da segunda captura do sexto experimento

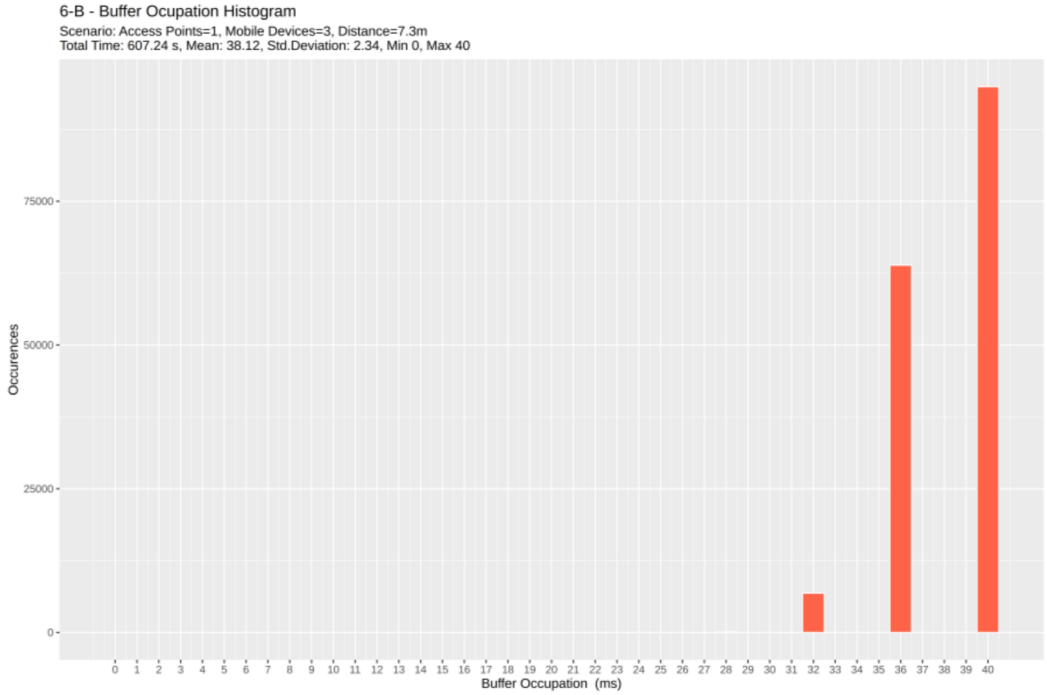


Figura B.23: Histograma da terceira captura do sexto experimento

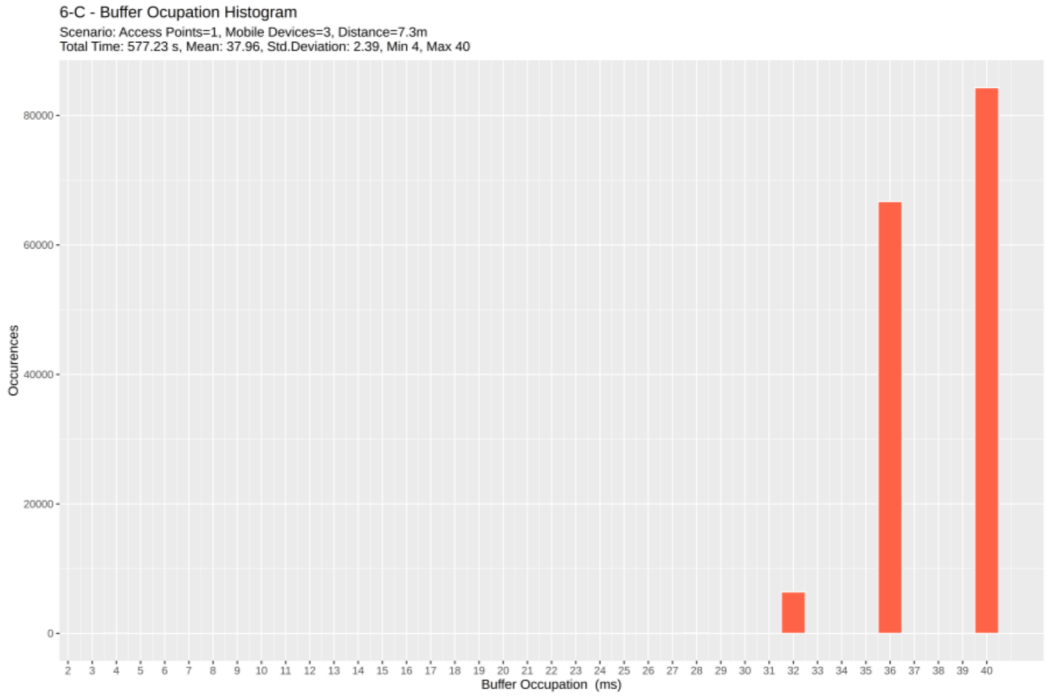
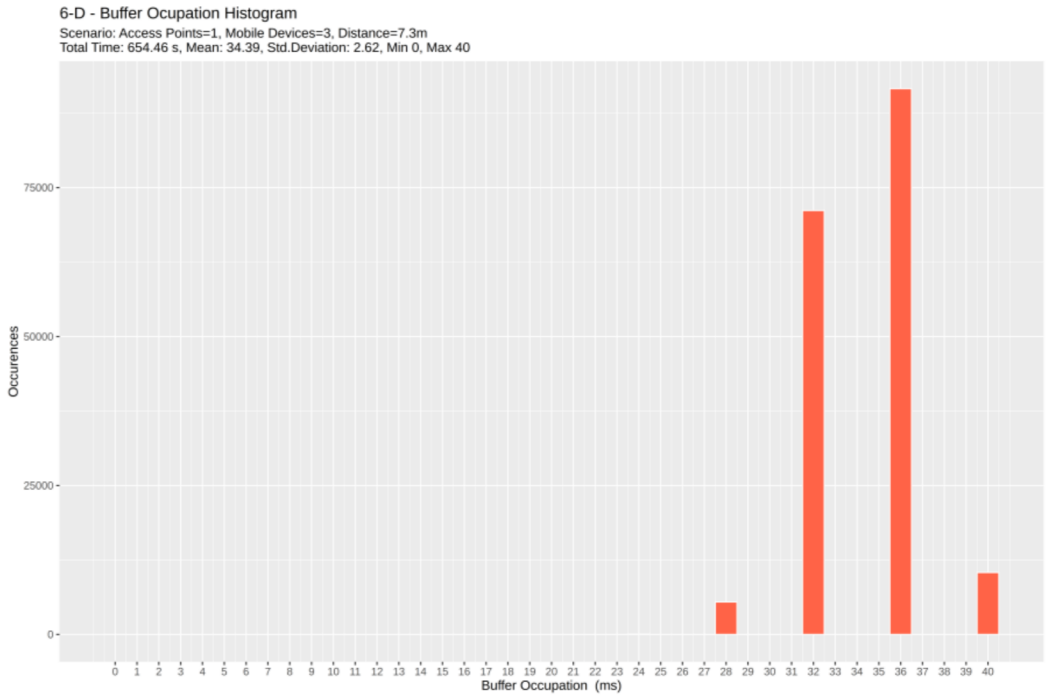


Figura B.24: Histograma da quarta captura do sexto experimento



B.7 Experimento 07

Figura B.25: Histograma da primeira captura do sétimo experimento

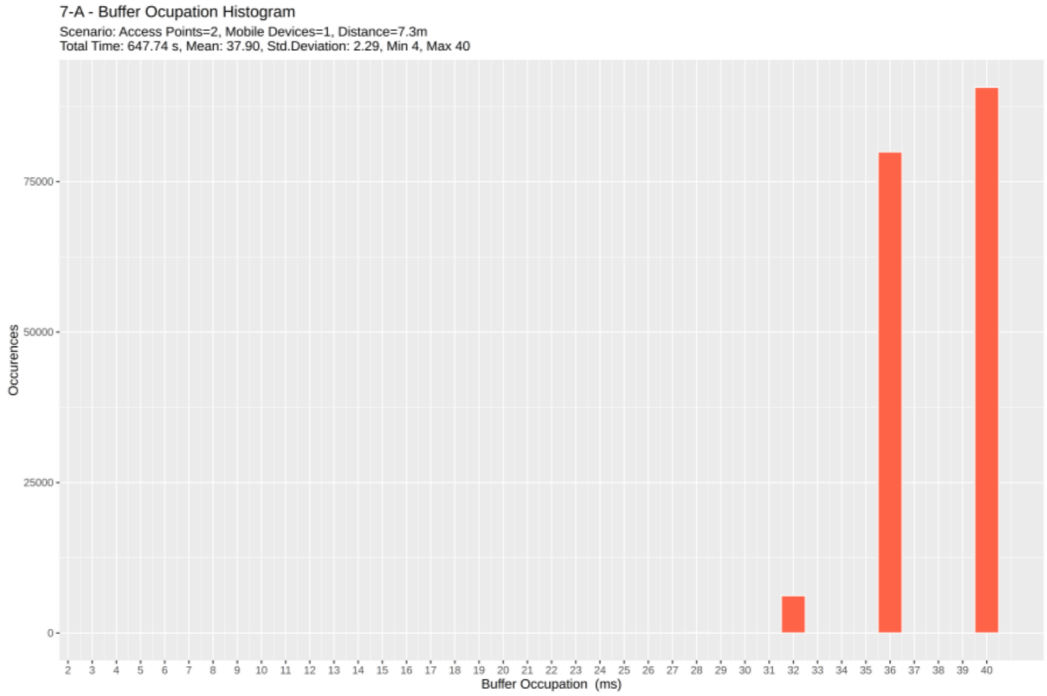


Figura B.26: Histograma da segunda captura do sétimo experimento

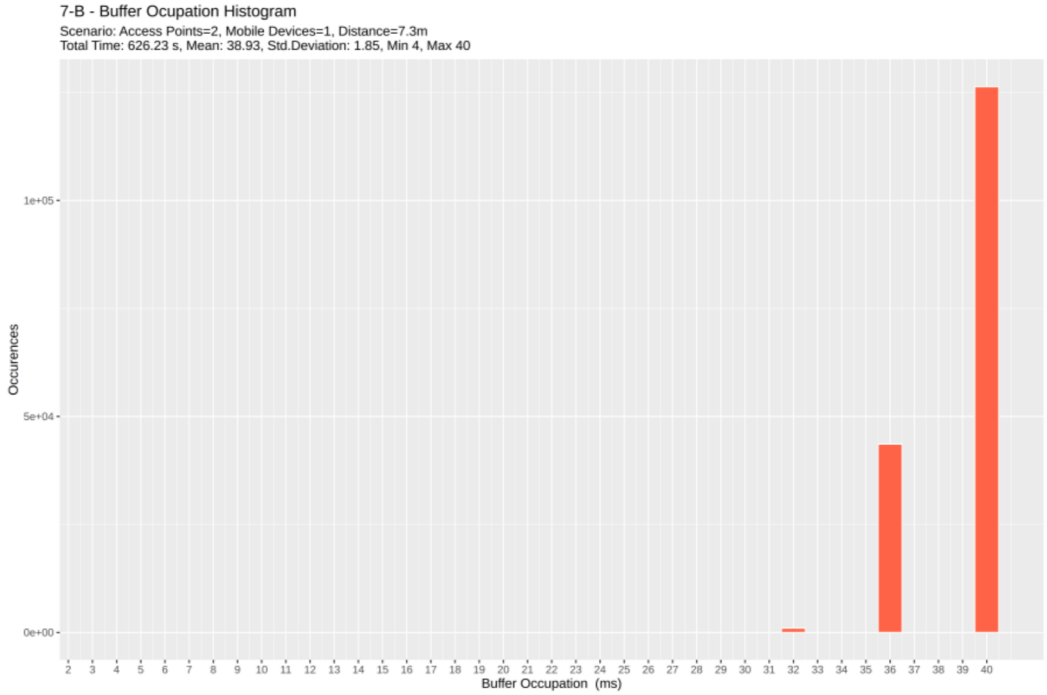


Figura B.27: Histograma da terceira captura do sétimo experimento

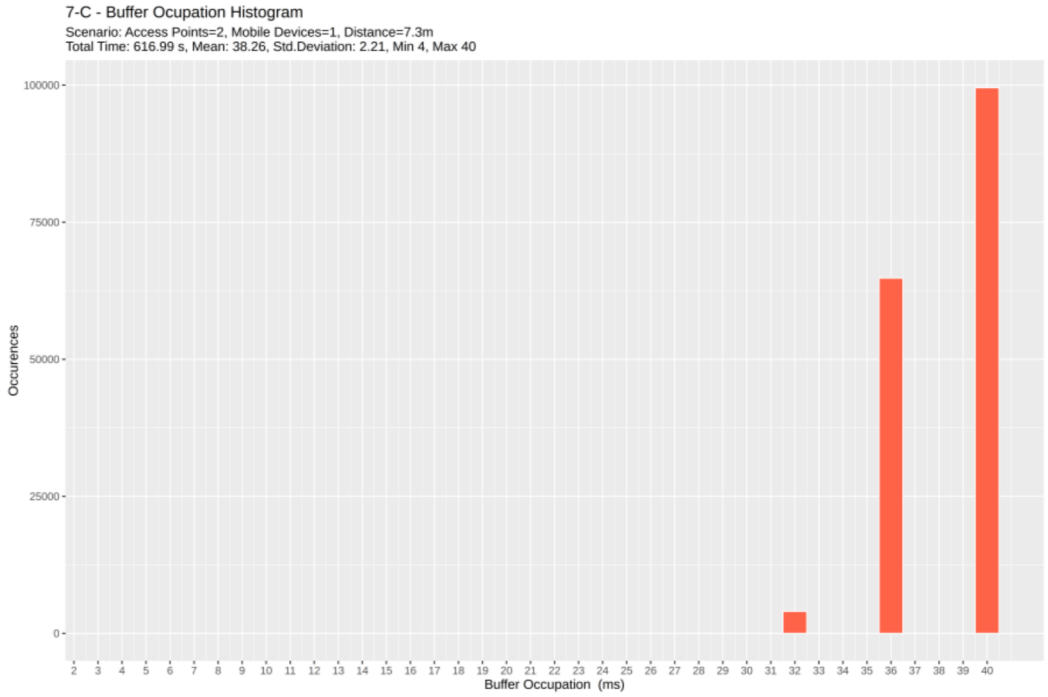
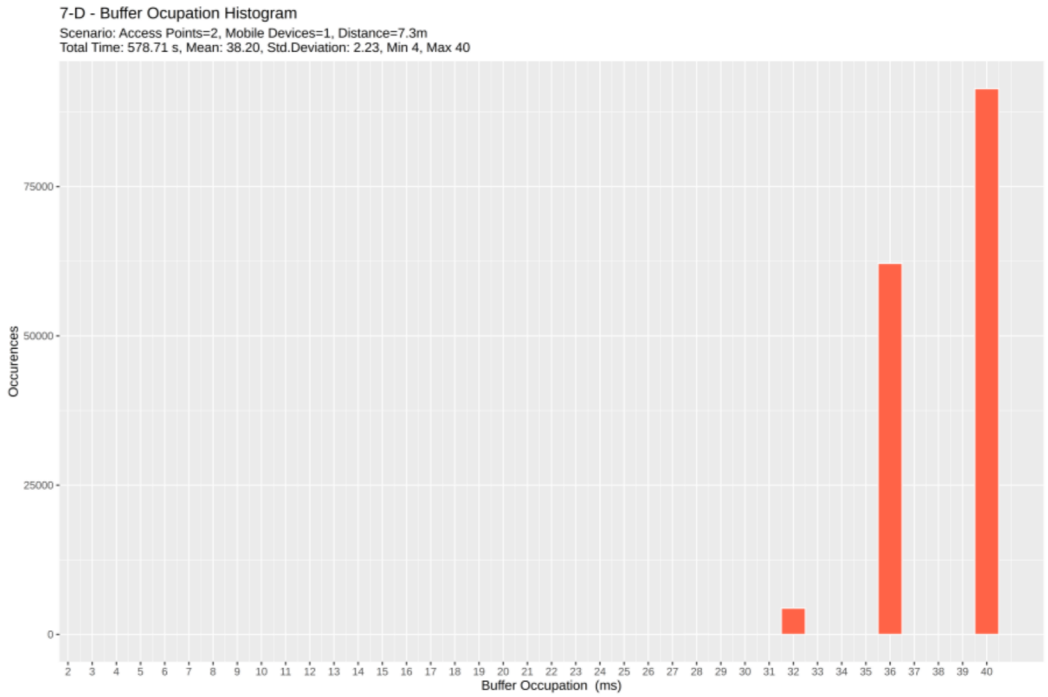


Figura B.28: Histograma da quarta captura do sétimo experimento



B.8 Experimento 08

Figura B.29: Histograma da primeira captura do oitavo experimento

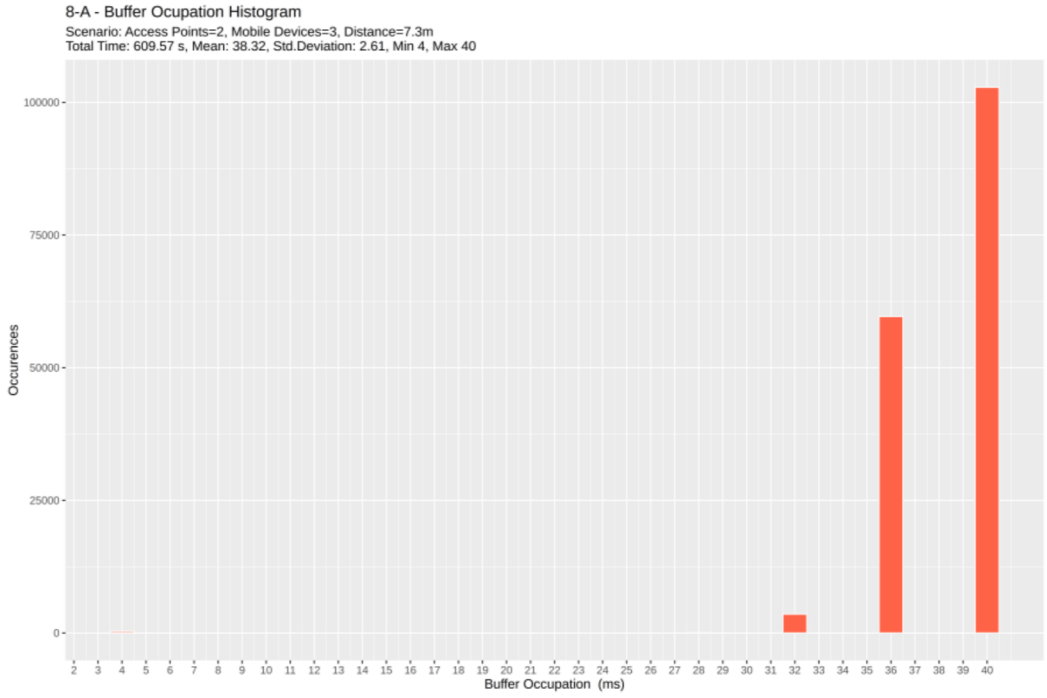


Figura B.30: Histograma da segunda captura do oitavo experimento

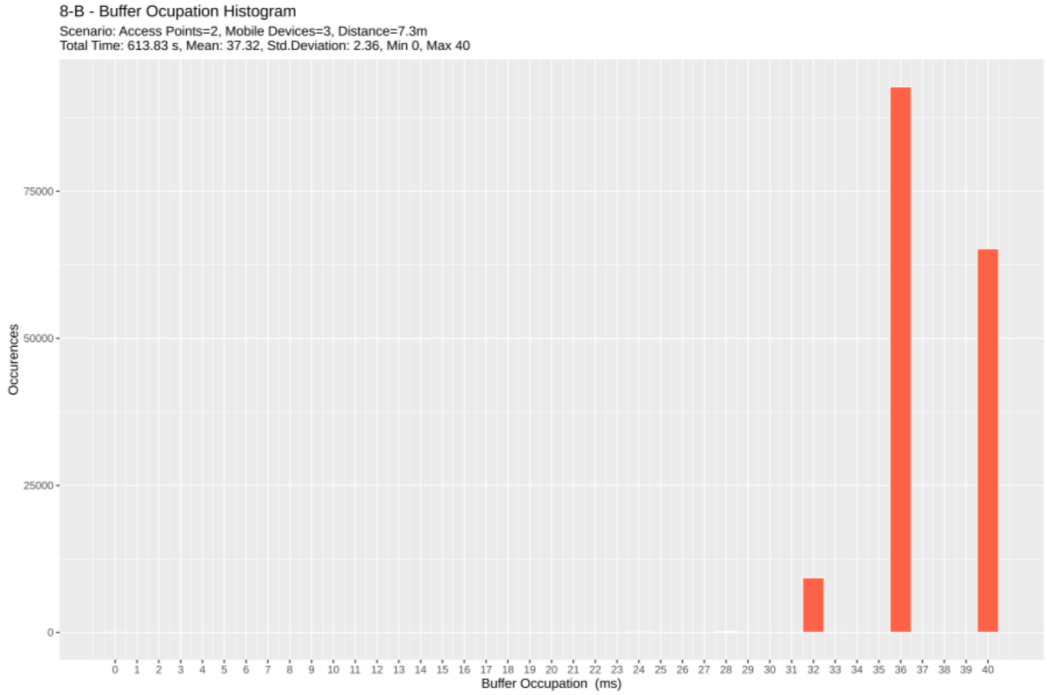


Figura B.31: Histograma da terceira captura do oitavo experimento

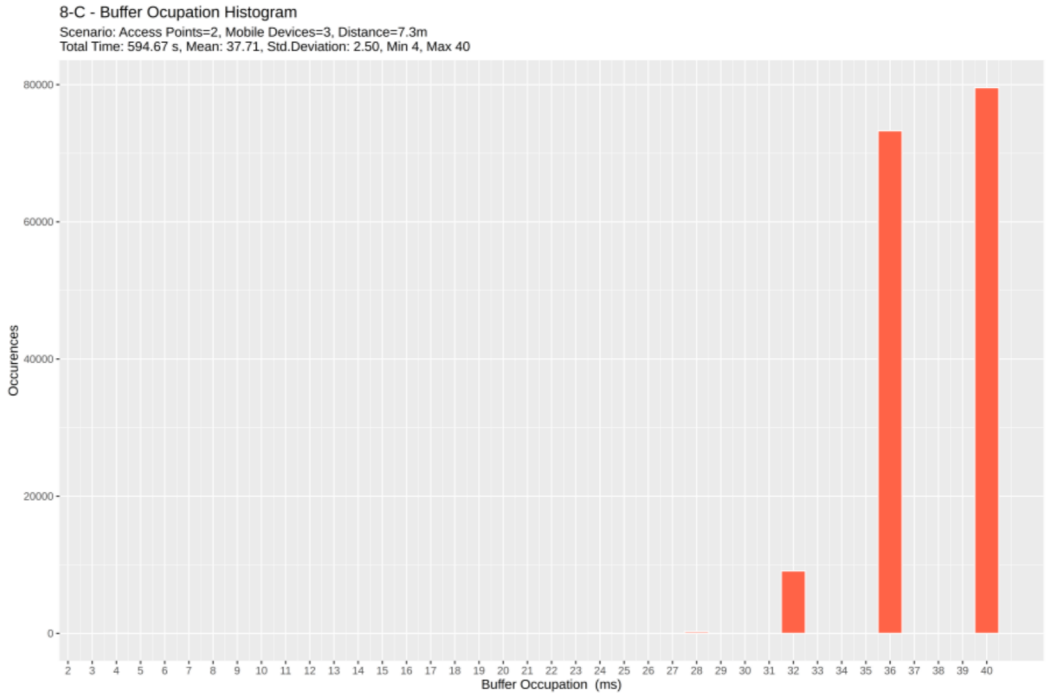
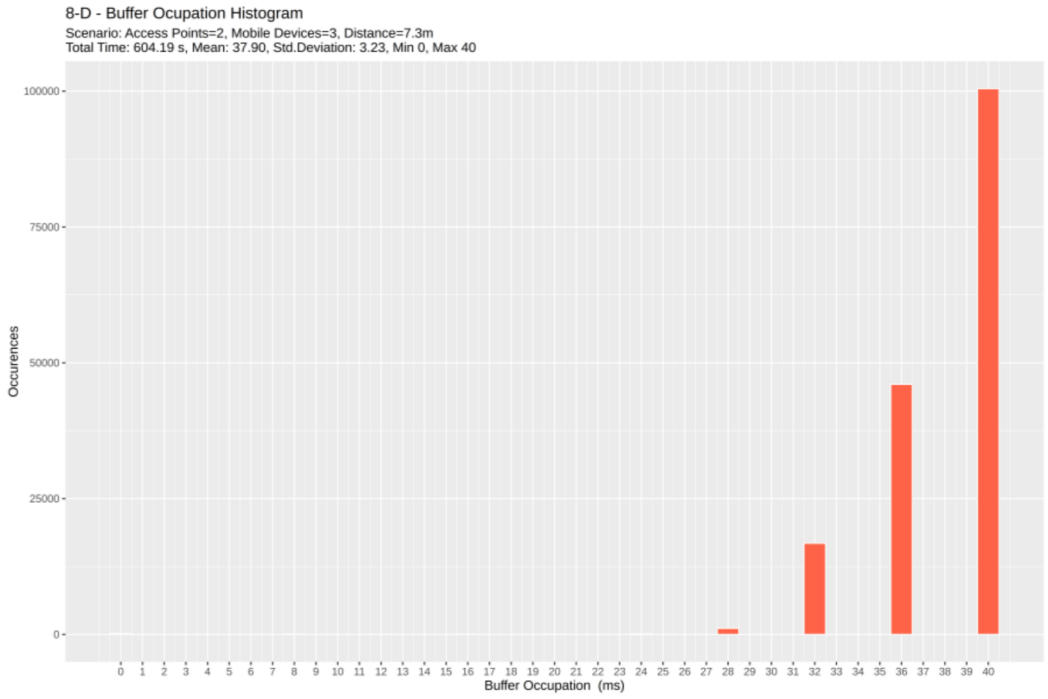


Figura B.32: Histograma da quarta captura do oitavo experimento



Apêndice C

Gráfico de *Timeline* dos Experimentos

C.1 Experimento 01

Figura C.1: *Timeline* da primeira captura do primeiro experimento

Figura C.2: *Timeline* da segunda captura do primeiro experimento

Figura C.3: *Timeline* da terceira captura do primeiro experimento

Figura C.4: *Timeline* da quarta captura do primeiro experimento

C.2 Experimento 02

Figura C.5: *Timeline* da primeira captura do segundo experimento

Figura C.6: *Timeline* da segunda captura do segundo experimento

Figura C.7: *Timeline* da terceira captura do segundo experimento

Figura C.8: *Timeline* da quarta captura do segundo experimento

C.3 Experimento 03

Figura C.9: *Timeline* da primeira captura do terceiro experimento

Figura C.10: *Timeline* da segunda captura do terceiro experimento

Figura C.11: *Timeline* da terceira captura do terceiro experimento

Figura C.12: *Timeline* da quarta captura do terceiro experimento

C.4 Experimento 04

Figura C.13: *Timeline* da primeira captura do quarto experimento

Figura C.14: *Timeline* da segunda captura do quarto experimento

Figura C.15: *Timeline* da terceira captura do quarto experimento

Figura C.16: *Timeline* da quarta captura do quarto experimento

C.5 Experimento 05

Figura C.17: *Timeline* da primeira captura do quinto experimento

Figura C.18: *Timeline* da segunda captura do quinto experimento

Figura C.19: *Timeline* da terceira captura do quinto experimento

Figura C.20: *Timeline* da quarta captura do quinto experimento

C.6 Experimento 06

Figura C.21: *Timeline* da primeira captura do sexto experimento

Figura C.22: *Timeline* da segunda captura do sexto experimento

Figura C.23: *Timeline* da terceira captura do sexto experimento

Figura C.24: *Timeline* da quarta captura do sexto experimento

C.7 Experimento 07

Figura C.25: *Timeline* da primeira captura do sétimo experimento

Figura C.26: *Timeline* da segunda captura do sétimo experimento

Figura C.27: *Timeline* da terceira captura do sétimo experimento

Figura C.28: *Timeline* da quarta captura do sétimo experimento

C.8 Experimento 08

Figura C.29: *Timeline* da primeira captura do oitavo experimento

Figura C.30: *Timeline* da segunda captura do oitavo experimento

Figura C.31: *Timeline* da terceira captura do oitavo experimento

Figura C.32: *Timeline* da quarta captura do oitavo experimento

Apêndice D

Submissões

[CEI] Agradecimento pela Submissão Caixa de entrada ×

periodicos@avisos.ufpb.br qua., 13 de jan. 20:00 (há 5 dias)
para mim

Caio Marcelo Campoy Guedes Guedes,

Agradecemos a submissão do seu manuscrito "Transmissão de Áudio em Tempo Real sobre Redes Wireless IEEE 802.11 com Foco em Acessibilidade para o Cinema Digital" para Comunicações em Informática. Através da interface de administração do sistema, utilizado para a submissão, será possível acompanhar o progresso do documento dentro do processo editorial, bastanto logar no sistema localizado em:

URL do Manuscrito: https://periodicos.ufpb.br/index.php/cei/authorDashboard/submission/57139
Login: caiomcg

Em caso de dúvidas, envie suas questões para este email. Agradecemos mais uma vez considerar nossa revista como meio de transmitir ao público seu trabalho.

Liliane S. Machado

Comunicações em Informática http://periodicos.ufpb.br/index.php/cei

INSTITUTO NACIONAL DA PROPRIEDADE INDUSTRIAL

07/01/2021 870210001629 10:18

29409161928117138

Pedido nacional de Invenção, Modelo de Utilidade, Certificado de Adição de Invenção e entrada na fase nacional do PCT

Número do Processo: BR 10 2021 000210 7

Dados do Depositante (71)

Depositante 1 de 1

Nome ou Razão Social: ASSISTA SERVIÇOS DE TECNOLOGIA LTDA

Tipo de Pessoa: Pessoa Jurídica

CPF/CNPJ: 22653294000193

Nacionalidade: Brasileira

Qualificação Jurídica: Pessoa Jurídica

Endereço: BR 230 - KM 12 - nº 11034 - Edifício Casa Nova Center - sala 202 - bairro Renascer

Cidade: Cabedelo

Estado: PB

CEP: 58108-012

País: Brasil

Telefone:

Fax:

Email: adm@assistatecnologia.com.br

PETICIONAMENTO ELETRÔNICO

Esta solicitação foi enviada pelo sistema Peticionamento Eletrônico em 07/01/2021 às 10:18, Petição 870210001629

Dados do Pedido

Natureza Patente: 10 - Patente de Invenção (PI)

Título da Invenção ou Modelo de Utilidade (54): SISTEMA E MÉTODO DE TRANSMISSÃO DE REFORÇO DE ÁUDIO DE BAIXA LATÊNCIA PARA RECEPTORES SOBRE REDES WIRELESS IEEE 802.11

Resumo: Patente de invenção que cobre um método e uma arquitetura de transmissão de fluxos de áudio que garante a reprodução do conteúdo em um limiar igual ou inferior a 40 milissegundos. A presente invenção é composta por um conjunto de componentes que manipulam o áudio em tempo real e por um método próprio de configuração e uso que permite que se atinja o limiar supracitado, fazendo com que a solução possa ser utilizada em diversos meios, tais como acessibilidade em cinema digital, narração ao vivo de eventos esportivos e apresentações artísticas. Os dois macro componentes da invenção são o Audio Server e o Audio Playback. Audio Server é o responsável pela recepção e tratamento do fluxo de áudio, que deve ser transmitido aos espectadores com o mínimo de latência. O Audio Playback, por sua vez, é responsável por exibir o conteúdo de forma síncrona com relação à fonte. Para isso, ele apresenta uma sequência de passos bem definidos que manipulam o sinal transmitido, que são a replicação do sinal, a compressão, a aplicação de algoritmos de reconstrução do sinal e a manipulação da sua frequência. Assim, o conteúdo é reproduzido com uma variação de tempo igual ou inferior ao limiar estipulado. A arquitetura e o processo combinados descrevem precisamente quais são os componentes envolvidos e como estão organizados, de forma a coordenar e realizar a comunicação entre eles. Diferentes cenários de uso são possíveis, uma vez que a arquitetura foi desenhada para suportar diversas aplicabilidades.

Figura a publicar: 1

PETICIONAMENTO ELETRÔNICO Esta solicitação foi enviada pelo sistema Peticionamento Eletrônico em 07/01/2021 às 10:18, Petição 870210001629

RESUMO

"SISTEMA E MÉTODO DE TRANSMISSÃO DE REFORÇO DE ÁUDIO DE BAIXA LATÊNCIA PARA RECEPTORES SOBRE REDES WIRELESS IEEE 802.11". Patente de invenção que cobre um método e uma arquitetura de transmissão de fluxos de áudio que garante a reprodução do conteúdo em um limiar igual ou inferior a 40 milissegundos. A presente invenção é composta por um conjunto de componentes que manipulam o áudio em tempo real e por um método próprio de configuração e uso que permite que se atinja o limiar supracitado, fazendo com que a solução possa ser utilizada em diversos meios, tais como acessibilidade em cinema digital, narração ao vivo de eventos esportivos e apresentações artísticas. Os dois macro componentes da invenção são o *Audio Server* e o *Audio Playback*. *Audio Server* é o responsável pela recepção e tratamento do fluxo de áudio, que deve ser transmitido aos espectadores com o mínimo de latência. O Audio Playback, por sua vez, é responsável por exibir o conteúdo de forma síncrona com relação à fonte. Para isso, ele apresenta uma sequência de passos bem definidos que manipulam o sinal transmitido, que são a replicação do sinal, a compressão, a aplicação de algoritmos de reconstrução do sinal e a manipulação da sua frequência. Assim, o conteúdo é reproduzido com uma variação de tempo igual ou inferior ao limiar estipulado. A arquitetura e o processo combinados descrevem precisamente quais são os componentes envolvidos e como estão organizados, de forma a coordenar e realizar a comunicação entre eles. Diferentes cenários de uso são possíveis, uma vez que a arquitetura foi desenhada para suportar diversas aplicabilidades.