UNIVERSIDADE FEDERAL DE MINAS GERAIS
Departamento de Ciências da Computação
Programa de Pós-graduação em Ciências da Computação

Daniel Henriques Cézar Miranda Soares

MODELAGEM DE QOE PARA JOGOS EM NUVEM

Belo Horizonte
2022

Daniel Henriques Cézar Miranda Soares

Modelagem de QoE para jogos em nuvem

Versão Final

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação do Instituto de Ciências Exatas da Universidade Federal de Minas Gerais como requisito parcial para a obtenção do grau de Mestre em Ciência da Computação.

Orientador: Daniel Fernandes Macedo

Belo Horizonte

2022

Soares, Daniel Henriques Cézar Miranda

S676m Modelagem de QOE para jogos em nuvem [manuscrito] / Daniel Henriques Cézar Miranda Soares — 2022.
71 f. il.

Orientador: Daniel Fernandes Macedo
Dissertação (mestrado) - Universidade Federal de Minas Gerais, Instituto de Ciências Exatas, Departamento de Ciência da Computação
Referências: f. 65-67

1. Computação – Teses. 2. Jogos em nuvem –Teses. 3. Video games –Teses. 4.Computação em nuvem – Teses. I. Macedo, Daniel Fernandes. II.Universidade Federal de Minas Gerais, Instituto de Ciências Exatas, Departamento de Ciência da Computação. III. Título.

CDU 519.6*82 (043)

Ficha catalográfica elaborada pela bibliotecária Belkiz Inez Rezende Costa
CRB 6/1510 – Insituto de Ciências Exatas da UFMG

UNIVERSIDADE FEDERAL DE MINAS GERAIS
INSTITUTO DE CIÊNCIAS EXATAS
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

FOLHA DE APROVAÇÃO

MODELAGEM DE QOE PARA JOGOS EM NUVEM

DANIEL HENRIQUES CÉZAR MIRANDA SOARES

Dissertação defendida e aprovada pela banca examinadora constituída pelos Senhores:

PROF. DANIEL FERNANDES MACEDO - Orientador
Departamento de Ciência da Computação - UFMG

PROFA. JUSSARA MARQUES DE ALMEIDA GONÇALVES
Departamento de Ciência da Computação - UFMG

PROF. EDUARDO COELHO CERQUEIRA
Instituto de Tecnologia - UFPA

DR. DIEGO NEVES DA HORA
Software Engineer - Google

Belo Horizonte, 17 de Janeiro de 2022.

Agradecimentos

Agradeço primeiramente aos meus pais que sempre incentivaram o estudo e apoiaram durante a carreira acadêmica. Agradeço a minha parceira Barbara que apoiou e esteve presente durante esta pesquisa. Ao meu orientador Daniel que ajudou muito durante todo o trajeto. Aos amigos que estiveram presentes e aos novos amigos feitos na trajetória, como o Marcos e o Michael que ajudaram em vários aspectos durante a realização deste trabalho.

Resumo

O número de usuários das plataformas de jogos tradicionais (videogame e computador) em janeiro de 2020 era de 312 milhões ao somar os usuário da *Steam*, *PlayStation* e *Xbox Live*. Já a plataforma de jogos em nuvem *GeForce Now* contava com 10 milhões de usuários totais em abril de 2021. Além disso, relatórios de mercado prevem um ritmo de crescimento de jogos em nuvem entre 50-60% ao ano nos próximos dez anos. Esta dissertação propõe um modelo de QoE em jogos em nuvem que utiliza parâmetros de rede e de nuvem. Além disso, consideramos separadamente os efeitos da rede para os fluxos de vídeo e comandos do jogo. Como saída do modelo temos a previsão de QoE avaliado pelo usuário.

Tais modelos podem ser empregados para o gerenciamento de ativos de rede de um provedor ou computacionais de uma plataforma de nuvem, visando melhoria da experiência do usuário. Outras possíveis aplicações são: estimar o impacto de alterações na rede de outros trabalhos sobre jogos em nuvem, validar requisitos de uma nova plataforma de jogos em nuvem e balanceamento de cargas por um provedor de rede de acordo com a previsão de QoE.

Nosso trabalho é o primeiro a usar uma pesquisa de QoE para jogos em nuvem e criar um preditor de QoE baseado na rede. Visto que os modelos existentes são baseados somente em QoS, nosso modelo é relevante pois pode ser utilizado em cenários que não são possíveis com os modelos anteriores.

Utilizamos uma *testbed* com servidor e cliente na mesma rede, gerando a degradação de rede por partida de acordo com o documento P809 da ITU. Foram coletados dados de 2020 partidas com nove usuários. Estes dados foram utilizados para treinar um regressor para dois modelos distintos, um com acordo e outro sem acordo. O modelo sem acordo parte da premissa que o provedor de rede e o provedor que fornece o serviço de jogos em nuvem não possuem um compartilhamento de informações, logo o modelo só tem acesso aos dados de rede. Já o modelo com acordo parte da premissa que existe um acordo e tem mais dados, permitindo uma análise mais profunda.

Ambos os modelos conseguiram prever o QoE avaliado dentro de um intervalo de erro de mais ou menos um em uma escala MOS de sete pontos para o modelo sem acordo e de mais ou menos 0.9 para o modelo com acordo. A taxa de acerto do modelo sem acordo é de 40% e a do modelo com acordo é 50%. Este valor sobe para até 90% considerando avaliações com erro de até uma unidade como corretas. Além disso, analisamos aspectos tais como a criação de modelos hierárquicos, generalização e explicabilidade.

Palavras-chave: Jogos, Nuvem, QoE, Modelo, *Streaming*.

Abstract

The number of users of traditional gaming platforms (consoles and computers) in January 2020 was 312 million when adding the users of *Steam*, *PlayStation* and *Xbox Live*. The cloud gaming platform *GeForce Now* had 10 million total users as of April 2021. In addition, market reports predict a growth rate of cloud gaming between 50-60% per year for the next ten years. This dissertation proposes a model for QoE in cloud games that uses network and cloud parameters. In addition, we separately consider the network effects for video streams and game commands. As an output of the model, we have the predicted evalueted QoE by the user.

Such models can be used to help manage network provider's assets or a cloud provider's computing assets, aiming to improve the user experience. Other possible applications are: estimating the impact of network changes from other cloud gaming works, validating requirements of a new cloud gaming platform, and balancing load by a network provider according to QoE prediction.

Our work is the first to use a QoE survey for cloud gaming and create a network-based QoE predictor. Since the existing models are based only on QoS, our model is relevant as it can be used in scenarios that are not possible with the previous models.

We used a *testbed* with server and client on the same network, generating network degradation per match according to ITU document P809. Data were collected from 2020 matches with nine users. The data was used to train a regressor for two different models, one with agreement and the other without agreement. The without agreement model assumes that the network provider and the cloud gaming provider do not share information, so the model only has access to network data. The model with agreement, on the other hand, assumes that there is an agreement and has more data, allowing for a deeper analysis.

Both models were able to predict the evaluated QoE within an error range of plus or minus one on a seven-point MOS scale for the without agreement model and plus or minus 0.9 for the with agreement model. The hit rate of the model without agreement is 40% and that of the model with agreement is 50%. This value goes up to 90% considering evaluations with error of up to one unit as correct. In addition, we analyze aspects such as the creation of hierarchical models, generalization and explainability.

Keywords: Games, Cloud, QoE, Model, Streaming.

Lista de Figuras

Lista de Tabelas

Sumário

Capítulo 1

Introdução

A cada geração de vídeo games os jogos lançados se tornam mais realistas quanto aos gráficos e mais imersivos quanto à jogabilidade (Rodrigues et al. [2007]). Essas melhorias exigem melhores recursos de *hardware* que têm custos elevados. Outra forma de jogar, que permite baixo custo sem perder qualidade é utilizar de jogos em nuvem, que consiste em executar o jogo remotamente, realizando *streaming* do jogo para o dispositivo do usuário. Este pode ser um *tablet* ou um computador de baixo custo, com a comodidade de ser possível jogar em qualquer plataforma, não sendo limitado a um computador *Windows* ou um videogame. Vários destes serviços de jogos em nuvem estão disponíveis no mercado, sendo alguns pagos e outros gratuitos e de código aberto.

Jogo em nuvem é um novo paradigma que possibilita reduzir os custos tanto para os jogadores quanto para os desenvolvedores. De acordo com Allied Market Research [2021] se espera que este mercado cresça anualmente a uma taxa de 50-60%, alcançando 22 bilhões de dólares em 2030. Os provedores deste tipo de serviço e provedores de rede necessitam de modelos de previsão de satisfação de usuário para melhorar o seu serviço.

É importante diferenciar jogos em nuvem de *streaming* de jogos passivo, onde no segundo uma partida de um jogo está sendo transmitida para vários espectadores, da mesma forma que uma partida de esporte como futebol. Neste tipo de *streaming* passivo o usuário não está jogando, mas somente assistindo. Nos jogos em nuvem o usuário interage com o jogo da mesma forma se ele estivesse sendo executado localmente.

Poucos dos serviços de jogos em nuvem fornecem a infraestrutura para executar os jogos, sendo necessário somente o *hardware* do cliente e o pagamento de uma assinatura, como o Microsoft xCloud (Microsoft [2020]), Sony PlayStation Now (Sony [2020]), Geforce Now (Nvidia [2020]), e o pouco utilizado *Google Stadia* (J. Schreier [2021]). A infraestrutura consiste em um computador com capacidade de rodar os jogos assim

como seria feito da forma tradicional, então o serviço transmite deste computador para outro dispositivo do cliente. Desta forma, somente o serviço de *streaming* está sendo fornecido.

Alguns são gratuitos, de código fechado e precisam que o usuário tenha a infraestrutura de servidor, como o Steam Link (Steam [2020]) e Parsec (Parsec [2020]) e outros de código aberto e o usuário também precisa ter a infraestrutura como o Rainaway (Rainway [2019]) e o Moonlight (Moonlight [2013]).

O *streaming* pode ser feito de duas formas: o usuário possui a infraestrutura, ou o serviço fornece a infraestrutura na nuvem além do serviço. Para ter a infraestrutura é necessário um computador com placa de vídeo compatível com o serviço a ser utilizado que funcionará como servidor, um ou mais dispositivos clientes, que podem ser tablets, celulares ou outros computadores. Finalmente, estes devem ter conexão de boa velocidade e baixa latência entre cliente e servidor (*Steam Link* sugere conexão cabeada Gigabit ou sem fio na frequência de 5 Ghz (Steam [2020])).

Tendo em vista que o dispositivo cliente pode ser de baixo custo como *tablets*, celulares e *notebooks* simples, é valido apontar que grande parte dos usuários vão estar conectados via rede sem fio, que é um potencial limitador de banda (Bankov et al. [2019]). De acordo com Cai et al. [2015] ao utilizar jogos em nuvem é preciso de tempo de resposta baixo, ou baixa latência, já que diferente de um filme, nos jogos acontece interação com o ambiente virtual a todo momento.

O foco deste trabalho é modelar a qualidade de experiência do usuário, ou QoE (*Quality of Experience*), em jogos em nuvem. Ao modelar o problema são analisados parâmetros de rede que influenciam no QoE como atraso, perda e *jitter*. Ao final do trabalho será possível avaliar qual parâmetro é mais crítico para o QoE e tentar prever o impacto das alterações em tempo real na rede, tornando mais prático validar novas propostas de melhoria, comparar as já existentes e auxiliar operadoras e gerentes de rede na gestão de recursos para este tipo de serviço.

1.1 Motivação

Existem vários trabalhos dedicados a melhorar o QoE em jogos em nuvem, porém não há um modelo definido para avaliar o impacto de cada parâmetro de rede no QoE final. Nossa motivação é entender a relação entre cada variável de rede e as correlações e dependências entre estas e o QoE. É possível utilizar estas dependências para criar um modelo que facilite a avaliação do impacto da rede no QoE e possivelmente melhor otimizar um cenário de jogos em nuvem.

Um modelo com tais características pode ser utilizado por pesquisadores e desenvolvedores para tarefas tais como (*i*) auxiliar o gestor da rede ou *cloud* no gerenciamento da infraestrutura e como os parâmetros estão influenciando o QoE; (*ii*) prever como alterações em parâmetros da rede ou do codificador vão afetar o QoE; e (*iii*) melhor direcionar estudos e projetos sobre QoE em jogos em nuvem.

De acordo com D. Peppiatt [2021], o número de usuários ativos mensalmente na plataforma de jogos *Steam* foi de 120 milhões em janeiro de 2020, enquanto no *PlayStation* foi de 102 milhões e no *Xbox Live* foi de 90 milhões. Nestas plataformas os jogos são jogados de forma tradicional, ou seja, são instalados no computador ou vídeo game do usuário e rodam localmente. Já a plataforma *GeForce Now* de jogos em nuvem, lançada em fevereiro de 2020 (R. Smith [2020]), alcançou 10 milhões de usuários totais em abril de 2021 de acordo com J. Bitner [2021].

Por mais que os números de usuários da recém lançada *GeForce Now* pareçam pequenos perto das plataformas tradicionais, o ritmo de adesão é acelerado, com servidores sendo lançados em vários países, inclusive no Brasil em setembro de 2021 (W. Landim [2021]). O serviço *Google Stadia* era um dos mais esperados e um dos primeiros, lançado em 2019, porém foi um fracasso devido ao seu desempenho baixo, poucos jogos e obrigatoriedade de compra de um controle exclusivo para uso do serviço (J. Schreier [2021]).

Dado o crescimento de jogos em nuvem, nossa principal motivação consiste em auxiliar tomadores de decisão em operadoras, provedoras e em ambientes de nuvem, no ajuste dos seus serviços de forma a prover uma boa experiência aos usuários de jogos em nuvem. Sabemos que esse serviço consome alta largura de banda (Domenico et al. [2021]), podendo variar entre 30 e 35 Mbps para os serviços *Google Stadia* e *Nvidia GeForce Now* e entre 10 e 13 Mbps para o *Sony PSNow*. Sabemos também que o meio a ser utilizado (rede com fio ou sem fio) pode afetar a qualidade do serviço (QoS), e consequentemente o QoE. Ainda assim, as soluções existentes não apresentam modelo de desempenho adequado, gerando uma dificuldade em medir e comparar os resultado obtidos.

1.2 Objetivos e Metas

Neste trabalho pretendemos modelar o QoE em jogos em nuvem de acordo com as variáveis de rede e de *hardware* e estudar as relações e correlações entre elas e o QoE final e criar um modelo que avalie corretamente os impactos destas variáveis no QoE.

Diante disso, a principal questão de pesquisa neste trabalho é: *Como quantificar*

e modelar o QoE do usuário em aplicações de jogos em nuvem? Para elaborar esse problema geral, perguntas específicas de pesquisa serão abordadas, e respondê-las faz parte dos nossos objetivos.

1. **Como definir e validar os parâmetros que mais afetam o QoE?**
   É preciso categorizar os parâmetros do fluxo, além de quantificar o seu efeito final no QoE em jogos em nuvem. Este efeito deveria ser analisado isoladamente e por agrupamentos de parâmetros.

   Abordagem: Empregamos mecanismos de explicabilidade, como o SHAP (Lundberg & Lee [2017]), para quantificar a influência de cada atributo no resultado final dos modelos desenvolvidos. Além disso fizemos um estudo preliminar estatístico usando o método fatorial 2K para comprovar a interação entre os parâmetros, o que nos levou ao uso de um modelo com aprendizado de máquina para o trabalho final.

2. **Como separar a transmissão de vídeo e envio de comandos?**
   Separar o envio de comandos de mouse e teclado do cliente da transmissão de vídeo do servidor é importante para analisar a relação entre eles separadamente e em interação.

   Abordagem: Fizemos uma captura de pacotes para mapear as portas utilizadas pela API da *Nvidia*. Isso nos permitiu gerar atrasos e perdas de forma independente para comandos e vídeo.

3. **Como validar o modelo desenvolvido em testes com usuários?**
   É necessário definir e realizar testes com usuários, tendo em vista a inexistência de modelos objetivos de qualidade de jogos em nuvem, bem como a subjetividade intrínseca do QoE.

   Abordagem: Criamos uma *tesbed* e uma avaliação de QoE seguindo os padrões sugeridos pelo documento P809 da International Telecommunication Union [2018] e convidamos alunos da UFMG para jogar partidas de três minutos e avaliar cada partida.

Capítulo 2

Referencial Teórico

Neste capítulo são revisados os conceitos necessários para compreender o trabalho realizado, apresentando o conceito de qualidade de experiência do usuário, além de explanar a codificação e transmissão de vídeo.

2.1 Qualidade de experiência em jogos

A QoE (Qualidade de Experiência) é em termos gerais o grau de satisfação do usuário com um serviço (Callet et al. [2012]). As variáveis utilizadas para definir QoS (Qualidade de Serviço) não são suficientes para garantir que um usuário ficará satisfeito com a experiência, já que por mais que elas tenham efeitos na experiência (Juluri et al. [2015]), elas remetem à qualidade da prestação do serviço, e não à qualidade da experiência final do usuário. Assim como o trabalho de Juluri et al. [2015], o trabalho de Fiedler et al. [2010] se dedica a estudar a relação entre QoS e QoE, e que em estudo de *streaming* esta relação é exponencial. O autor dá o nome desta relação, hipótese IQX, *Exponential interdependency of quality of experience and quality of service* ou Interdependência exponencial de QoE e QoS.

O documento P809(International Telecommunication Union [2018]) da ITU detalha várias recomendações sobre medição de QoE em *streaming* e jogos em nuvem (*cloud gaming*), e este será utilizado como referência na dissertação. Nos capítulos de metodologia e resultados iremos referir às seções deste documento da ITU.

2.1.1 Aspectos de QoE para jogos

De acordo com a seção 6 do documento P809 da ITU, as partes que formam a classificação final de QoE do jogador são divididas em: estética e apelo, qualidade de interação,

tipos de jogador, qualidade de jogo, envolvimento, imersão, presença, fluxo de jogo, absorção, aceitabilidade. A relação que existe entre estes conceitos pode ser visto na Figura 2.1. Cada uma dessas características tem peso diferente de pessoa para pessoa. A seguir apresentamos cada uma delas, seguindo a ordem apresentada pelo documento da ITU-T.

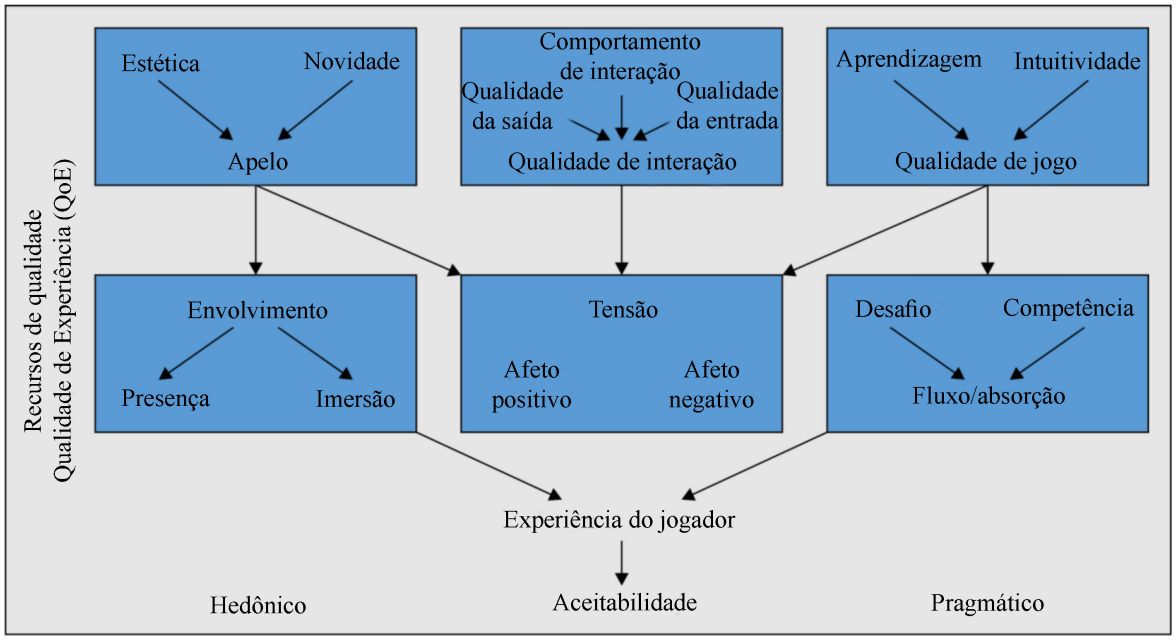


Figura 2.1: Aspectos de QoE em jogos
Autor: Adaptador e traduzido do documento P809 da International Telecommunication Union [2018]

Estética e apelo são a aparência do jogo, se é visualmente bonito e que gere apelo a ser jogado. O tipo de jogador é definido para cada gênero de jogo, um exemplo de tipos de jogador para o gênero de RPG são: ***achiever*** que procura por uma diversão difícil, ***explorer*** que procura por diversão fácil, ***socializer*** que procura por diversão social e ***killer*** que procura por diversão difícil e social. Qualidade de interação é uma classificação análoga à usabilidade quando nos referimos a um sistema, logo se refere aos comandos do jogo, quão simples é o menu e quão complexo é de chegar ao jogo em si[1] para um novo jogador.

Qualidade de jogo é definida pelo quão intuitivo é a sua jogabilidade e mecânica, em termos simples a curva de aprendizado do mesmo. Envolvimento é o quão dedicado o jogador fica ao jogo que está jogando: se está focado no jogo totalmente ou se está jogando enquanto realiza outras atividades.

[1]Tempo e dificuldade entre iniciar o jogo, navegar pelo menu, até estar em controle do personagem.

Imersão é um conceito complexo e ainda sem uma definição fixa no estado da arte. Existem três estados de imersão: engajamento, absorção e imersão total. Para entrar no nível de engajamento a barreira de preferência de jogo precisa ser quebrada para que seja possível dedicar atenção e aprender como jogar aquele jogo. Para o nível de absorção o jogador precisa unir as características e detalhes do jogo com a maestria dos controles para ficar emocionalmente conectado com o mesmo. Jogadores no estado de absorção já estão menos conscientes dos arredores e de si mesmos, mas não totalmente. Ao entrar em imersão total, os jogadores se sentem presentes no jogo e descrevem estar desligados da realidade a um ponto que o jogo é somente o que importa.

Presença é a sensação de estar realmente no universo do jogo e depende das características do jogo e ambiente do mesmo, como tarefas, reações e estímulos, mas também depende do estado de imersão do jogador. Fluxo de jogo é rapidamente resumido pelo documento P809 da ITU como uma experiência positiva causada por um balanço ótimo de desafios e habilidades em um ambiente orientado por objetivos.

Absorção em jogos pode ser descrita como a noção de engajamento cognitivo, que é descrita por três dimensões: foco de atenção, curiosidade e interesse. A absorção é idêntica ao fluxo, porém sem a dimensão de controle. Controle individual não é necessário para engajamento cognitivo, é possível ter engajamento cognitivo passivo assistindo TV, mas é impossível ter fluxo de jogo passivo.

Aceitabilidade descreve quão um jogador está disposto a jogar aquele jogo. Este fator é influenciado por vários outros, como preço do jogo, acessibilidade, gênero e a experiência que já teve ou não com o mesmo.

2.1.2 Paradigmas de teste

Os paradigmas de teste precisam representar uma situação real de jogo, na qual o jogador fica uma quantidade de tempo jogando e ao fim da partida avalia o QoE da mesma. Alguns fatores podem alterar o resultado final da avaliação de QoE, como a duração da partida. Uma partida muito longa ajuda a avaliar com mais critério, porém pode gerar fatiga no usuário. Uma forma sugerida de diminuir a fatiga é diminuir o tempo de partida e ser feito em múltiplas seções, o que aumenta o conjunto de dados, porém pode alterar o comportamento de avaliação dependendo do dia ou momento.

Partidas rápidas podem dificultar a avaliação de fluxo e imersão, mas ajudam a avaliar qualidade de vídeo e taxa de quadros por segundo. O contrário também é válido: partidas longas atrapalham a avaliação de qualidade e taxa de quadros, porém auxiliam a avaliação de fluxo e imersão, visto que o jogador tem mais tempo para focar no jogo em si e tira o foco do desempenho do computador em executar o jogo.

A Tabela 2.1 exemplifica os casos de uso para os paradigmas de teste. Por mais que não seja recomendado um teste passivo rápido com objetivo de medir fluxo, isso não significa que seja impossível. Quando o foco do estudo é avaliar a qualidade de interação, por exemplo a influência do atraso nos controles do jogador no QoE, uma interação rápida é recomendada.

Foco do estudo/paradigma de testes	**Passivo rápido**	**Passivo longo**	**Interação rápida**	**Interação longa**
Qualidade da saída	X			
Qualidade da interação			X	
Conceitos de engajamento				X

Tabela 2.1: Tabela 1 da seção 7 do documento P809 da ITU: *Visão geral dos paradigmas de teste da ITU-T P.809*
Fonte: International Telecommunication Union [2018]

2.1.3 Testes experimentais

Esta subseção trata de recomendações do documento P809 da ITU.T para os testes experimentais, sendo algumas partes relativas a testes de imersão, fluxo e avaliação da qualidade do jogo (não do QoE), que não são relevantes para este trabalho. Será dada atenção às recomendações que cobrem qualidade de interação, qualidade de controles e qualidade de vídeo.

De acordo com o documento da ITU, foi feita uma pesquisa com 104 jogadores pela Universidade de Zagreb(M. et al. [2013]). Nela, foi indicado que os participantes conseguem se divertir com um jogo (sem degradações) indiferente dos equipamentos de entrada e saída (monitores, teclado). Estes equipamentos normalmente são diferentes dos usados em casa pelos jogadores, logo é concluído que os equipamentos usados para testes subjetivos não precisam de requerimentos estritos. Para testes de imersão foi reportado que supervisores em uma sala de teste podem diminuir a concentração.

Em relação ao tamanho da tela a ser usada, ela não pode ser muito grande pois afeta o tempo de resposta do jogador. Neste sentido, no meio dos *eSports* (Esportes Eletrônicos) uma tela de 24 polegadas é o comum.

Os tipos de jogo podem ser separados em três categorias de acordo com processamento e rede: jogos locais, jogos online e jogos em nuvem. Nos jogos locais todas as tarefas são processadas no dispositivo local, podendo ser um videogame, celular, *tablet* ou computador pessoal. Nos jogos online o cálculo dos estados de jogadores e objetos e a comunicação entre eles é feita em um servidor remoto, mas o processamento das imagens e captação de entradas de controle é feito no dispositivo local. Nos jogos em nuvem somente a captura de entradas e a exibição da imagem é feito localmente,

enquanto todo o processamento é realizado remotamente.

Cada um destes tipos de jogo requer um cenário de testes para QoE específico, pois cada tipo tem características diferentes de processamento e de rede. Como não existe um padrão para cada tipo de dispositivo (videogame, celular, computador, entre vários outros), a ITU recomenda levar alguns dos parâmetros a seguir em consideração, visto que alguns deles podem ser o foco da pesquisa de QoE e outros somente interferem.

- Características de entrada do dispositivo (telas de toque, controles, mouse e teclado);
- Quadros por segundo exibidos pela tela;
- Parâmetros de codificação de vídeo;
- Vazão de rede;
- Atraso de rede;
- Perda de pacotes de rede;
- Atraso de processamento no servidor;
- Atraso nas entradas (taxa de atualização monitor e teclado);
- Tipo do jogo;
- Perspectiva do jogo (primeira pessoa, terceira pessoa, isométrico);
- Percepção espacial e temporal do visual do jogo;
- Efeito de variação dinâmica de alguma das características acima, como *jitter*.

Quanto aos participantes do teste, alguns pontos são recomendados. É possível filtrar os participantes de acordo com experiencia no gênero do jogo ou não filtrar. Ao separar os participantes serão obtidos dois conjuntos de dados diferentes que devem ser analisados separadamente. Ao não filtrar, se cria um conjunto mais heterogênico e generalizado. Quando um participante joga mais de dez horas por semana daquele gênero de jogo, ele é considerado experiente de acordo com a International Telecommunication Union [2018].

Os participantes também podem ser testados antes da partida para separar os que tenham boa visão e audição dos que não possuem. Visto que jogos são baseados em estímulos visuais e auditivos, a classificação seria diferente para cada tipo de jogador.

O último ponto a ser avaliado em um jogador é seu estado emocional. Dependendo do humor, o jogador vai avaliar de forma diferente a qualidade de experiência. Este estado pode ser obtido a partir de questionários sobre o humor. O documento recomenda adaptar uma versão simplificada do *National Aeronautics and Space Administration task load index* (NASA-TLX) International Telecommunication Union [2018].

Também deve ser levado em conta o material a ser jogado, que não necessariamente é o jogo em si, mas sim um cenário do mesmo. Cada gênero de jogo tem sensibilidade diferente a diferentes degradações como atraso de rede e influências externas. Dentro do jogo escolhido também deve ser escolhido um cenário que represente bem o que pretende ser analisado.

Ao participar da avaliação, é recomendado pela ITU que os participantes recebam um questionário pré teste e um pós teste, de forma a ter mais informações de cada participante. O questionário pré teste pode ser composto de informações básicas como idade, gênero e profissão e ser complementado com experiência com jogos. Esta experiência com jogos se resume em quantas horas por semana o participante joga, quais gêneros, se já jogou o jogo da avaliação e conhecimento sobre degradação.

Para classificar a experiência, a ITU recomenda o uso da *absolute category rating* (ACR), classificação absoluta de categoria em tradução livre. Por mais que escalas discretas sejam comuns, a ITU recomenda o uso da ACR com uma escala continua de sete pontos. Os sete pontos são do pior ao melhor: extremamente péssimo, péssimo, ruim, razoável, bom, excelente, ideal.

2.1.4 Interação passiva

No método de interação passiva o participante assiste uma cena gravada previamente de um jogo, podendo esta cena conter ou não interferências na qualidade de imagem dentre outras, e avalia de acordo com o que foi observado. A interação passiva tem duas grandes vantagens: é garantido que todos os participantes vão experimentar o mesmo conteúdo do jogo e que as habilidades como jogador não serão limitantes na avaliação. Como jogos são experiências interativas, alguns aspectos não podem ser avaliados desta forma, como atraso de rede e fluxo de jogo.

Antes do teste, os participantes devem ser informados das regras do jogo que será exibido e de qualquer conteúdo base para compreender o jogo. Por mais que o jogo seja assistido e não jogado de forma interativa, o participante ainda pode avaliar a qualidade e degradação da imagem, como se foi possível captar o objetivo do jogo.

O cenário apresentado ao participante deve cobrir de forma representativa o jogo, contendo movimentação e aspectos que são presenciados durante uma partida intera-

tiva. De acordo com a ITU, uma duração de 30 segundos para avaliar qualidade de imagem é o suficiente, já para avaliar imersão é recomendado entre 10 e 15 minutos.

Se o objetivo do estudo for ter relação com a experiência interativa, é recomendado usar cenas gravadas de um teste interativo real, para captar quaisquer interferências de rede ou dispositivo.

2.1.5 Interação ativa com cenário de jogos

Ao avaliar a interação, duas abordagens são possíveis, a interação longa e a curta. Quando o objetivo do estudo é avaliar a qualidade da interação e os efeitos da degradação no QoE, um intervalo entre 90 e 180 segundos é o suficiente. Os equipamentos usados precisam ser capazes de executar ou transmitir o jogo sem degradações causadas por sistema ou *hardware*. Caso o jogo avaliado seja de vários jogadores, é interessante avaliar o aspecto social da interação entre eles durante a avaliação.

2.1.6 Questionários

Ao classificar a experiencia, caso o foco do estudo seja uma analise detalhada da interação com o jogo, a ITU recomenda o *game experience questionnaire* (GEQ), questionário de experiência de jogo em tradução livre, desenvolvido pela FUGA (*Fun of Gaming*, Ravaja et al. [2006]).

O GEQ consiste de 33 perguntas de 5 pontos na escala ACR para avaliar as sete dimensões de jogo: imersão sensorial e imaginativa, tensão, competência, fluxo, efeito negativo, efeito positivo e desafio. Após a avaliação da experiencia, é possível avaliar a experiência em si (não o jogo), se foi prazerosa ou não, sendo útil para classificar o lado pessoal de estar jogando em uma avaliação.

Ao avaliar fluxo, a ITU recomenda dois tipos de questionários, o *flow-short-scale* criado por Rheinberg [2015] e o *flow state scale* criado por Jackson & Marsh [1996].

O *flow-short-scale* divide as características de fluxo em duas dimensões, fluência de performance (concentração e foco, controle e claridade) e absorção por atividade (envolvimento, sensação de distorção temporal, desafio ideal e desconexão com a realidade). A escala usa escala ACR de 7 pontos para as perguntas e mesmo não sendo desenvolvida para jogos, é utilizada em vários estudos de jogos de acordo com a ITU.

Ao avaliar o fluxo com a *flow state scale*, vários pontos são avaliados pelo participante sobre o fluxo durante um evento ou atividade, com notas de um (discordo totalmente) a cinco (concordo totalmente). Estes pontos são:

- Balanço entre desafio e habilidade: se a habilidade condiz com o desafio, ele será um sucesso;
- Junção de consciência e ação: resposta automática à tarefa;
- Objetivos claros: experiencia de ter um conjunto de objetivos pré determinados;
- *Feedback* não ambíguo: *feedback* sobre desempenho;
- Concentração na tarefa: focado na tarefa;
- Paradoxo de controle: desempenha tarefa com facilidade;
- Perda de auto consciência: imerso na tarefa;
- Alteração temporal: tempo acelera ou desacelera durante a atividade;
- Experiência que se guia: atividade intrinsecamente recompensadora;

Ao avaliar o engajamento, são usados questionários divididos entre engajamento de jogo e experiência de imersão, sendo que o segundo engloba fatores de fluxo, absorção e presença para avaliar o nível de imersão do jogador.

Os questionários de presença de grupo e presença individual avaliam a experiência em um ambiente virtual e devem levar em consideração presença espacial, envolvimento e realismo percebido para avaliar a qualidade da imersão de um grupo ou pessoa em um ambiente virtual. Estes são chamados de questionários de presença, pois o nível de imersão é suficiente para que o jogador se sinta presente no ambiente virtual.

Ao fim de uma experiência pode ser avaliada a necessidade de satisfação do jogador, que deve tentar avaliar os fatores de jogo que levam a uma experiência agradável e significativa.

Todos os tipos de questionários exemplificados já foram usados em vários estudos de acordo com a ITU. Entretanto, vários deles medem ou avaliam fatores que se sobrepõem e não são úteis para todos os casos de estudo. Existe pouca informação disponível comparando os resultados destes questionários.

2.1.7 Medições de desempenho do jogador

De acordo com a ITU existe uma relação entre o resultado final de um jogo competitivo e o QoE percebido pelo jogador. Logo, de uma forma limitada, o QoE pode ser medido utilizando métricas de desempenho do jogador.

A seleção destas métricas depende do tipo de jogo e do objetivo do experimento. Frequentemente pontos de jogo (pontos, número de mortes, objetivos concluídos e razão entre unidades construídas e destruídas) são utilizados. outros parâmetros como duração da sessão ou frequência podem ser utilizados.

Como estas medições não são intrusivas, métricas de desempenho podem ser coletadas sem interrupções e interferências ao jogador durante uma partida competitiva. Por mais que pareça uma forma simples de avaliar o QoE, as métricas de desempenho oferecem somente uma noção limitada sobre a qualidade subjetiva do jogador.

2.2 *Software* de *streaming* de jogos *Moonlight*

Nosso trabalho utiliza o *software* de código aberto *Moonlight*, tanto por ter o código disponível e comentado em C++, mas também por utilizar a eficiente plataforma da *Nvidia* que é utilizada no *GeForce Now*. Nessa seção vamos comentar o funcionamento geral do programa.

O *software* Moonlight é uma implementação de código aberto do protocolo de jogos em nuvem *NVIDIA Shield* (Moonlight [2013]). Foram escritas várias versões para várias plataformas como *Android*, IOS, *Chromebook*, *Sony PS Vita*, *Windows* e várias distribuições Linux. Estas versões atuam no cliente, enquanto o servidor utiliza o *software GeForce Experience* da *Nvidia* sobre um ambiente Windows.

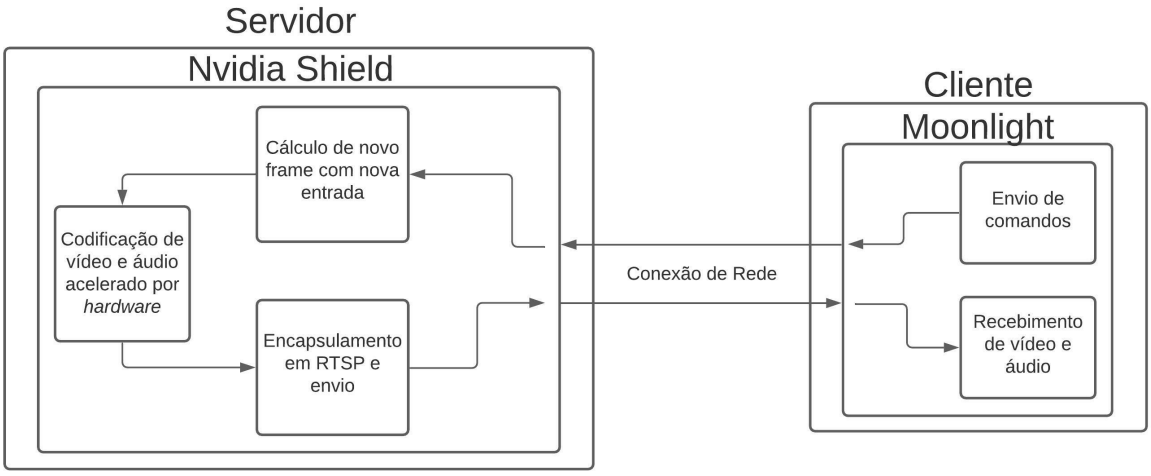


Figura 2.2: Arquitetura de funcionamento Moonlight

Para utilizar o Moonlight (ver Figura 2.2), o usuário necessita de um computador atuando como servidor, executando o *software GeForce Experience* da *Nvidia* com

hardware que suporte o protocolo *Nvidia Shield*. O Moonlight é utilizado somente no cliente, comunicando com o *software* da *Nvidia* através do protocolo *Nvidia Shield*, logo é uma implementação somente do lado do cliente.

Para executar o *GeForce Experience* com o *NVIDIA Shield*, o computador servidor deve utilizar sistema operacional *Windows* 7 ou superior e ter uma placa de vídeo *Nvidia GeForce* GTX 650 ou superior (Nvidia [2020]) para ter suporte ao protocolo *NVIDIA Shield*. Já no lado do cliente, é necessário utilizar somente o *Moonlight*. Para todos os testes e coleta de dados utilizamos uma *GeForce* GTX 1050Ti no *Windows* 10.

Diferente do uso do DASH (*Dynamic Adaptive Streaming over HTTP*, Krishnappa et al. [2013]) utilizado no *YouTube* para alterar dinamicamente a resolução do *streaming*, o *Moonlight* mantém uma resolução fixa durante toda a transmissão. Esta resolução é escolhida antes de iniciar a partida, que no nosso caso foi 1080p a 60 quadros por segundo.

2.3 Protocolos de transporte de vídeo

Vários protocolos de rede podem ser utilizados para transmissão de vídeo em tempo real, no trabalho é dado foco aos protocolos RTP (*Real-time Transport Protocol*) e RTSP (*Real Time Streaming Protocol*). Estes protocolos são utilizados pela API da *Nvidia* e pelo programa utilizado na pesquisa, o *Moonlight*.

2.3.1 RTP

O RTP foi apresentado em 1992 pela IETF (*Internet Engineering Task Force*, Schulzrinne [1992]) com objetivo de padronizar funcionalidades para os aplicativos de transmissão de dados em tempo real, tanto em redes *unicast* como em *multicast*, entretanto sem garantir QoS (qualidade de serviço) ou reservar recursos de endereçamento.

O RTP é encapsulado sobre a camada UDP/IP, utilizando os serviços de multiplexação e *cheksum* do UDP e estabelecendo uma comunicação fim a fim. Os dados a serem transmitidos que são produzidos pelo aplicativo remetente são encapsuladas em pacotes RTP, que por sua vez são encapsulados em um segmento UDP e enviados.

Normalmente o RTP é implementado como parte da aplicação e não como parte do *kernel* do sistema operacional. Em resumo o protocolo permite a especificação dos requisitos de tempo e conteúdo da transmissão multimídia, tanto no envio quanto na recepção da seguinte forma:

- Numeração sequenciada: o RTP atribui número de ordem aos pacotes, podendo ser usado para verificar perdas, sequenciamento e redirecionamento de pacotes;
- Selo de temporização: possibilita a correta temporização dos pacotes contendo áudio e/ou vídeo;
- Envio de pacotes sem retransmissão: característica necessária em transmissões multimídia, pequena perdas não atrapalham a qualidade e a não retransmissão torna o sistema mais robusto. O RTP apenas permite ao receptor notar as perdas e ou atrasos;
- Identificação de origem: necessário para indicar quem enviou o pacote. Em uma conferência *multicast*, um mesmo fluxo pode ter várias origens;
- Identificação de conteúdo: permite a alteração dinâmica dos *vocoders* (codificadores de voz) em redes sem garantia de QoS em função das perdas e do atraso a fim de melhorar a qualidade final acústica;
- Sincronismo: pacotes de um mesmo fluxo podem sofrer diferentes atrasos. A variação deste atraso (*jitter*) é prejudicial à reprodução da mídia. *Buffers* adicionais podem então ser utilizados para eliminar o *jitter*;

2.3.2 RTSP

O RTSP é um protocolo de domínio público apresentado em 1996 pela IETF (*Internet Engineering Task Force*, Rao & Lanphier [1996]) que permite uma interação cliente-servidor entre a fonte de mídia (servidor) e o usuário (transdutor). Essa interatividade vem da necessidade do usuário ter um maior controle sobre a reprodução da mídia no transdutor.

As funcionalidades do RTSP são similares à funcionalidade que um aparelho reprodutor de CD disponibiliza para se ouvir música gravada, porém em execução de arquivo. O protocolo permite que um transdutor controle a corrente de mídia através de comandos como: pausa e reinício, retrocesso e avanço rápidos, e reposicionamento da reprodução.

O RTSP não encapsula os comandos no mesmo pacote de transmissão dos dados de mídia, ele os envia paralelamente em portas diferente, de forma semelhante ao FTP (*File Transfer Protocol*). Dessa forma o RSTP é chamado de protocolo fora da banda, sendo a banda considerada como o canal de transmissão da mídia.

Cada sessão RTSP possui um identificador atribuído pelo servidor. O cliente faz uma requisição para iniciar a sessão (*SETUP*), ao passo que o servidor responde com

um identificador. O cliente fornecerá esse identificador para cada requisição que fizer ao servidor até que feche a sessão com uma requisição *TEARDOWN*. A transmissão de mídia acontece quando o transdutor no cliente envia uma requisição *PLAY*. É possível ainda usar requisições de *STOP e PAUSE*, para parar totalmente e pausar, respectivamente. Em termos de aplicabilidade, o RTSP é nitidamente eficiente ao atuar sob servidores sob demanda e apresenta várias aplicações, como: e-mail de voz, secretária eletrônica para telefonia IP, vídeo e voz sob demanda, jogos em nuvem entre outros.

2.4 Codificação de vídeo

Realizar jogos em nuvem em tempo real requer codificação de vídeo que seja rápida mas sem afetar a qualidade de imagem para alcançar esse objetivo. Um codificador (*codec*) é um algoritmo que utiliza de uma técnica para reduzir o tamanho do arquivo de vídeo para que seja utilizado menos largura de banda para enviar o mesmo pela rede. Um decodificador é a outra parte desse algoritmo que fica no lado do cliente, que decodifica e exibe o vídeo. Todo esse processo é semelhante a um compactador de arquivos porém leva em conta nuances de um arquivo de vídeo como similaridade entre um quadro e o próximo.

Existem vários algoritmos para realizar codificação de vídeo, iremos analisar alguns dos mais utilizados para que o funcionamento geral de um *codec* seja compreendido. As técnicas de codificação são definidas pela *Motion Pictures Experts Group*(MPEG) e pela *International Telecommunication Union*(ITU) (Maia [2015]), sendo MPEG o nome de um dos primeiros codificadores de vídeo.

O padrão MPEG utiliza de três tipos de quadros de imagens para gerar um vídeo, o tipo I (*intra*) que é uma imagem completa do tipo JPEG, o P (*predicted*) que armazena a diferença entre o I e o próximo P e entre eles vários do tipo B (*bidirectional* ou *inter*) que armazena vetores de movimento detalhados para cada grupo de quadros B que são capazes de referenciar os quadros anteriores e posteriores.

Em 2002 foi introduzido no mercado o MPEG-4 Parte 10, conhecido como H.264, que consegue entregar a mesma qualidade de imagem com metade do tamanho comparado com MPEG e MPEG-2. Essa melhoria foi alcançada com uso de vetores mais complexos nos quadros B e P, para utilizar o menor número de quadros I possíveis. O H.264 ainda é muito utilizado mas está sendo lentamente substituído pelo HEVC ou H.265 que consegue a mesma qualidade com metade do tamanho, utilizando de pedaços menores nas divisões dos quadros I e criando blocos de codificação para que cada parte da imagem gere vetores e predições mais precisas e mais eficientes nos quadros P e B

(Mohr [2018]).

A melhor utilização do espaço e consequentemente da rede para armazenar e transmitir vídeo com uso de algoritmos cada vez mais complexos exige *hardware* cada vez melhor para codificar e decodificar e exibir o vídeo nos dispositivos de forma rápida e isso deve ser considerado para realizar uma escolha balanceada entre codificador e *hardware* para cada aplicação.

Capítulo 3

Trabalhos Relacionados

Este capítulo revisa os mais relevantes trabalhos encontrados na literatura relacionados ao tema desta dissertação. A metodologia de pesquisa consiste primariamente em buscas em plataformas virtuais como Google Scholar[1], IEEE Xplore Digital Library[2] e o Portal de Periódicos CAPES/MEC[3]. O conjunto de palavras chave utilizadas no processo de busca e combinações foi: QoE, Gaming Streaming, Cloud Gaming, video-conference, voip e QoE Model.

A natureza de jogos em nuvem é parecida com a de uma vídeo conferência ou chamada *online* porque são em tempo real e bidirecionais, diferentes de um *streaming* de vídeo. Ao buscar por trabalhos relacionados, foram encontradas pesquisas parecidas realizadas em torno do aplicativo *Skype* que realiza chamadas de vídeo em tempo real.

3.1 *Streaming* com ciência de QoE

Os trabalhos desta subsecção estão ligados diretamente com análise de QoE e de qualidade do serviço gerado para os usuários finais dos serviços discutidos.

O trabalho de Chen et al. [2014] analisa e modela como as variações de taxa de transmissão na rede afetam o QoE percebido ao utilizar o *Skype* para chamadas de voz. O estudo utiliza um arquivo de áudio de 30s de uma conversa que representa o QoE ideal e transmite via *Skype* com 40 taxas de transmissão diferentes para analisar o impacto no QoE. Foram utilizados 14 participantes que avaliaram durante 20 minutos cada taxa de transmissão, sem informar ao participante as características da transmissão. O estudo

[1]https://scholar.google.com/
[2]https://ieeexplore.ieee.org
[3]https://www.periodicos.capes.gov.br

conclui que o QoE e a taxa de transmissão em áudio têm uma relação logarítmica, o QoE e a frequência de mudanças na taxa também é logarítmica. O trabalho de Reichl et al. [2010] também estuda a natureza logarítmica do QoE, com resultados similares.

No trabalho de Peñaherrera-Pulla et al. [2021] são estudados quais são os fatores chave para jogos em nuvem em vários meios, como *Ethernet*, Wi-Fi e LTE. A pesquisa também utiliza a plataforma de código aberto *Moonlight* utilizada neste trabalho. Foram utilizadas resoluções de 720p, 1080p, 1440p, e 4K com 30 e 60 quadros por segundo. Ao final do trabalho, de 14 fatores, 7 foram filtrados como realmente importantes utilizando correlação de impacto na saída. Os indicadores quantitativos de qualidade vistos utilizados na Figura 3.1 representam: tempo médio de recebimento em milissegundos, tempo médio de renderização em milissegundos, tempo médio de decodificação em milissegundos, porcentagem de quadros perdidos pela rede, porcentagem de quadros perdidos por *jitter*, taxa de quadros de decodificação em FPS (quadros por segundo) e taxa de quadros de renderização em FPS.

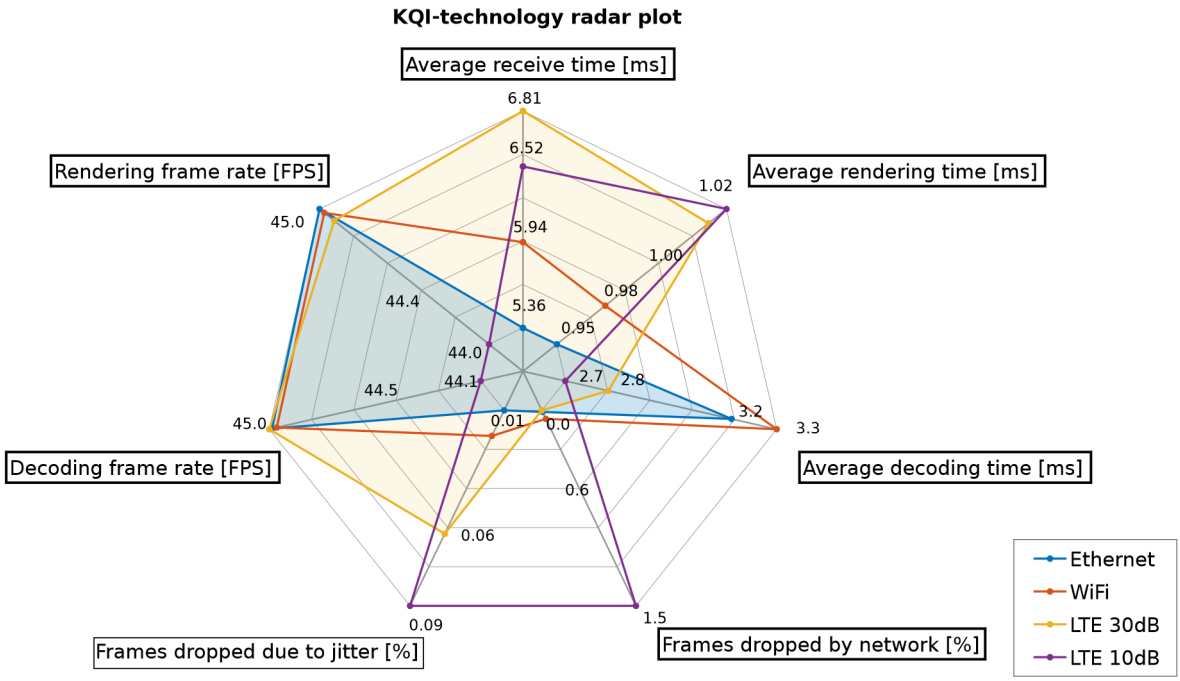


Figura 3.1: *Key Features* em cada tipo de rede
Autor: Peñaherrera-Pulla et al. [2021]

O trabalho de Sabet et al. [2020] tem como objetivo classificar um jogo em relação à sua sensibilidade a atraso. Participaram 9 jogadores que jogaram 12 cenários de diferentes tipos de jogos cada um com atrasos de 0, 150 e 300ms. Todos os participantes são jogadores, divididos em casuais (joga pelo menos duas vezes ao mês), intermediários

(joga pelo menos quatro vezes por semana) e experientes (joga mais de 8 anos jogando com frequência superior ao casual).

Com uso de um moderador, os jogadores discutiram após o experimento e chegaram em 9 características que geram a diferença de tolerância ao atraso para cada tipo de jogo. As características são:

- Acurácia temporal (TA), é o intervalo de tempo disponível para que o jogador realize a interação desejada;
- Consequências (CQ), é o tamanho do impacto gerado na experiencia do jogador por uma interação com atraso;
- Acurácia espacial (SA), é o nível de precisão necessária para completar uma interação com sucesso;
- Importância das ações (IoA), é o quanto cada comando de jogador (*input*) impacta no cenário do jogo;
- Previsibilidade (PR), é o quanto o jogador consegue estimar dos eventos futuros no jogo;
- Número de ações necessárias (NRA), é o número mínimo de ações por minuto para se jogar um cenário;
- Número de direções de entrada (NID), é a quantidade de entradas possíveis para um elemento (teclado + mouse ou manete), e consiste de translação (frente e trás, cima e baixo, esquerda e direita) assim como rotação (eixo vertical e altura). É quantificado em uma escala de 5 pontos sendo eles: 1, 2, 3, 4 e mais de 4;
- frequência de *feedback* (FF), é a frequência na qual o jogo dá um retorno visual, auditivo, ou háptico (tecnologia pela qual um sistema fornece ao usuário uma realimentação física, como, por exemplo, uma manete com tecnologia de realimentação de força);
- tipo de entrada (ToI), é uma escala de 5 pontos que classifica a entrada temporal do jogo entre discreta e contínua. Em jogos de tiro a entrada tende a ser mais contínua, já em jogos de estratégia mais discreta;

Cada característica foi avaliada em uma escala de 3 pontos de interferência para cada uma, sendo eles baixo, médio e alto. Com uso de mais 14 jogadores experientes, foram validadas estas características. Os autores notaram que 300ms deixa o jogo

muito desagradável, e reduziram a escala para dois valores de 0 e 200ms de atraso. Com o uso do método *K-means*, identificaram que os jogos se encaixam em baixa ou alta sensibilidade a atraso.

Ao final da pesquisa foi utilizado um conjunto de dados gerado de 570 pessoas que jogaram 30 jogos para gerar uma árvore de decisão que avaliar se o jogo é pouco ou muito sensível a atraso de acordo com as características encontradas anteriormente. A árvore atinge a melhor taxa de acurácia (86.6%) utilizando somente as características de ToI, NID, PR e TA para classificar se um jogo é ou não sensível a atraso.

No trabalho de Jarschel et al. [2011] foi avaliado a viabilidade de jogos em nuvem com o uso de participantes e de forma subjetiva, como o nosso trabalho, porém sem criar nenhum modelo ou previsor. O trabalho foi escrito em 2011 e jogos em nuvem já eram discutidos, mas ainda não eram uma realidade viável. Para realizar o experimento utilizaram um videogame *PlayStation 3* conectado a um dispositivo de transmissão para enviar o vídeo e receber os comandos do controle. E por fim esse dispositivo passava por um computador executando Linux que foi utilizado para gerar as interferências antes de chegar no computador cliente. Todas as transmissões feitas em 720p de resolução com 3 Mbps.

O trabalho cria vários cenários de rede listados na Tabela 3.1. Os participantes jogaram primeiro o cenário B sem nenhuma degradação por dez minutos e depois de forma aleatória jogavam os outros cenários por um minuto cada. Entre cada cenário eram pedidos para classificar o QoE entre um e cinco sendo um equivalente a ruim e cinco a excelente. Participaram 58 pessoas que jogaram cada cenário uma vez, gerando assim 580 dados.

Cenário	Atraso	Perda	Direção
B	0 ms	0.0%	Ambas
D1	80 ms	0.0%	Ambas
D2	200 ms	0.0%	Ambas
D3	300 ms	0.0%	Ambas
L1	0 ms	0.3%	Ambas
L2	0 ms	1.0%	Ambas
M1	40 ms	1.5%	Ambas
M2	180 ms	0.3%	Ambas
A1	120 ms	1.0%	Cliente para servidor
A2	120 ms	1.0%	Servidor para cliente

Tabela 3.1: Tradução livre da Tabela I do trabalho de Jarschel et al. [2011].

Os jogadores podiam escolher três tipos de jogos, sendo eles *Pro Evolution Soccer*

como um jogo mais lento, *Final Fantasy XIII* como um jogo de média velocidade e *Gran Turismo HD Concept* como um jogo rápido e de primeira pessoa (visão interna do carro). Antes de começar a jogar cada jogo o jogador era questionado se já tinha experiência com o tipo de jogo e se gostava ou não do jogo em questão.

Ao fim do trabalho de Jarschel et al. [2011] foi utilizado a implementação do algoritmo *REPTree* do WEKA (Eibe Frank & Witten [2012]) para construir uma árvore de decisão e utilizar a importância de cada parâmetro da base de dados, que pode ser vista na Tabela 3.2.

Parâmetro	Peso
Perda no *Downstream*	1.0
Atraso no *Downstream*	0.583
Perda no *Upstream*	0.370
Atraso no *Upstream*	0.212
Tipo de jogo	0.067
Habilidade do jogador	0.006
Opinião sobre o jogo	0.006
Idade	0.0

Tabela 3.2: Tradução livre da Tabela II do trabalho de Jarschel et al. [2011]. *Downstream* é o fluxo de dados do servidor para o cliente e *Upstream* é o contrário.

3.2 jogos em nuvem sem ciência de QoE e *Streaming* de vídeo

Os trabalhos desta subsecção estão relacionados com jogos em nuvem ou serviços em tempo real, porém não são compostos de análise de QoE ou diretamente sobre o usuário final deste serviço. Também incluímos trabalhos com ciência de QoE, porém que não tratam diretamente de jogos em nuvem. São trabalhos sobre *Streaming* de vídeo ou jogos que cobrem outros aspectos diferentes do tema principal, também incluindo QoE relacionado com outras características de *streaming* diferente de jogos que podem ser interessantes.

O trabalho de Domenico et al. [2021] analisa o *Google Stadia*, o *Nvidia GeForce Now* e o *Sony PSNow* que são as plataformas de jogos em nuvem disponíveis no momento da escrita do trabalho, sendo a *Microsoft xCloud* e a *Amazon Luna* a serem lançadas. O trabalho estuda os parâmetros de rede necessários para que cada plataforma funcione dentro do especificado pelas empresas, sendo interessante traçar um paralelo entre o *GeForce Now* e o *Moonlight* que utilizam os mesmos *codecs* a nível

de *hardware*, proprietários da *Nvidia* e utilizam os mesmos protocolos de rede para transmitir a partida.

De acordo com Domenico et al. [2021], é possível utilizar até 15Mbps de largura de banda para transmitir 1080p com 60 quadros por segundo com o *GeForce Now* sem ter perdas de quadros (mesmo que o recomendado seja 20Mbps). Isto ocorre pois o *codec* H.264 muda o nível de compressão, e consequentemente o tamanho dos pacotes RTSP de acordo com a rede.

Na tese de doutorado de Barman [2019] é analisado o QoE de quem assiste um *streaming* de vídeo de um jogo em tempo real. Enquanto o foco do nosso trabalho é jogos em nuvem diretamente para quem joga, o trabalho do autor considera um espectador que assiste passivamente a partida de outro jogador remotamente. Foram gravados clipes de 30 segundos em diferentes cenários de qualidade para serem avaliados pelos usuários, os jogos utilizados foram: *Counter Strike: Global Offensive* (CSGO), *Diablo III* (Diablo), *Defense of the Ancients 2* (Dota2), FIFA (FIFA), *Heathstone* (HS), *League of Legends* (LoL), *Need for Speed* (NFS) e *World of Warcraft* (WoW).

De acordo com Barman [2019] os vídeos foram capturados em resolução 1080p com 24 quadros por segundo e a partir do vídeo original foram gerados 576 outros vídeos utilizando os *codecs* x264, x265 e VP9. Os vídeos gerados tem *bitrates* entre 800 e 4000 kbps e resoluções de 1080p, 720p, 480p e 360p, que são as opções disponíveis nos serviços de *streaming* de jogos passivos.

Barman [2019] conclui que o QoE de um *streaming* de vídeo tradicional, como *Netflix* e *YouTube* não é afetado pelo conteúdo do vídeo, enquanto o de vídeo games é, ou seja ele é dependente do jogo que está sendo assistido. Este impacto está relacionado com o fato do *codec* x264 ter sido pensado para vídeos tradicionais.

Os vídeos de jogos são diferentes por conter menos informação espacial (o ambiente visual muda pouco), mas contém mais informação temporal (a visão do ambiente e de movimento é maior). Com os resultados das avaliações dos participantes foi gerado um modelo de QoE com métricas de avaliação de qualidade de vídeo, porém para *streaming* de vídeo de jogos. O modelo do autor, NR-GVSQE (*No-Reference Gaming Video Streaming Quality Estimator*), tem uma correlação de 97% ao avaliar vídeos de *streaming* de jogos enquanto o estado da arte, PSNR (*Peak Signal to Noise Ratio*), que obtém 89%.

3.3 Revisão dos trabalhos relacionados

	Participantes	Partidas	Método	Métricas	Coleta de Dados	Aplicação
Chen et al. [2014]	14	560	AnM	R	Internet	A
Peñaherrera-Pulla et al. [2021]	0	0	CP	R	Local	JN
Sabet et al. [2020]	570	17100	AM	R&H	Local	JN
Domenico et al. [2021]	0	0	CP	R	Internet	JN
Barman [2019]	25	14400	AnM	ReB	Local	SJP
Jarschel et al. [2011]	58	580	AnM	R	Local	JN
Nosso trabalho	9	2020	AM	R	Local	JN

Tabela 3.3: Comparação entre os trabalhos relacionados.
AnM - Analise Matemática, CP - Comparação entre entrada e saida, AM - Aprendizado de Máquina
R - Rede, R&H - Rede e *Hardware*, ReB - Resolução e *bitrate*
A - Áudio, JN - jogos em nuvem, SJP -*Streaming* de Jogos passivo

Antes de entrarmos no desenvolvimento do trabalho é interessante analisar os trabalhos relacionados, tendo em vista que algumas conclusões são de grande importância para nosso trabalho. Uma comparação entre os trabalhos relacionados e nosso trabalho pode ser visto na Tabela 3.3.

Com o trabalho de Domenico et al. [2021] é possível verificar o uso de banda e recursos de rede necessários para um experiência agradável com o *GeForce Now*. Este utiliza das mesmas API's utilizadas pelo *Moonlight* e por consequência o nosso trabalho.

Peñaherrera-Pulla et al. [2021] em seu trabalho demonstra que ainda é inviável o uso de redes sem fio para jogos em nuvem, sendo *Ethernet* o mais viável, mesmo que 100 Mbps e não *Gigabit*. LTE com 30 dB já é razoável, mas ainda tem tempos de resposta consideravelmente mais altos que da rede cabeada.

De acordo com Chen et al. [2014], a relação entre o QoE e a taxa de transmissão de áudio é logarítmica, que pode ser algo a ser analisado no nosso estudo. É possível analisar se existe uma relação matemática entre QoE em jogos em nuvem e o QoS da rede e se é logarítmica.

Os métodos de árvore de decisão e uso do *Kmeans* utilizados por Sabet et al. [2020] em seu trabalho são interessantes para analisar um grande conjunto de dados e podem ser de grande utilidade no nosso trabalho, para filtrar e entender os dados gerados.

O trabalho de Jarschel et al. [2011] é um dos poucos a fazer quase tudo o que fizemos, porém sem uma análise mais profunda dos dados gerados como o nosso trabalho. Deve ser levado em consideração que quando o trabalho foi publicado ainda não existia um documento para guiar a coleta de dados como o P809 da International Telecommunication Union [2018] usado neste trabalho.

Ainda temos o trabalho de Barman [2019] onde é analisado o QoE em um *Streaming* de jogos passivo, onde o espectador assiste o jogo de outra pessoa. O trabalho é interessante pela conclusão de que os codificadores de vídeo mais utilizados (x264 e x265) não são otimizados para o tipo de conteúdo de jogos. Estes codificadores são os utilizados no *Moonlight*, *GeForce Now* entre várias outras plataformas, mostrando que o uso de rede ainda pode ser aprimorado por estas plataformas.

O nosso é o único a analisar os impactos de interferências de rede geram no QoE em jogos em nuvem, bem como criar um modelo de previsão de desempenho a partir de dados de QoE gerados pela experiência real de jogos. Nosso trabalho une diversas características vistas nos outros trabalhos, como: utilizar de participantes ao invés de simulações, empregar métricas de qualidade de rede como atributos do modelo, construção de um preditor empregando aprendizado de máquina.

Capítulo 4

Metodologia de análise para jogos em nuvem

Neste capítulo vamos explicar a metodologia utilizada para a análise de jogos em nuvem. O objetivo é compreender as relações entre as variáveis de rede e a qualidade de experiência do usuário, e a partir deste estudo identificar quais variáveis são mais importantes para um melhor QoE e modelar a qualidade de experiência do jogo. Todo o processo, iniciado com a escolha dos cenários avaliados, seguido da montagem da infraestrutura da *testbed*, a análise dos dados coletados e o desenvolvimento do modelo com seus estimadores será discutido neste capítulo.

4.1 Cenários de avaliação

Após a coleta dos dados e antes do treinamento do modelo, é necessário ponderar sobre os cenários que serão avaliados. Com os dados coletados é possível pensar em vários casos de estudo e possibilidades, porém duas opções interessantes e com possíveis aplicações práticas são discutidas nesta seção e um modelo para cada cenário foi criado e explicado nas próximas seções e capítulos.

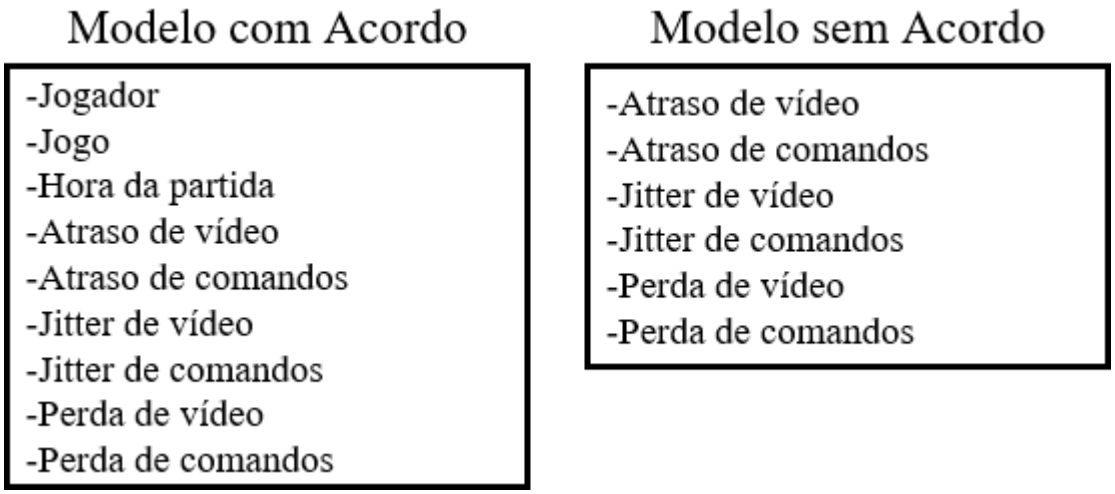


Figura 4.1: Informações que cada modelo tem acesso

Um provedor de conteúdo como *GeForce Now* ou *Xcloud* consegue ter informações de qual jogo cada usuário prefere, por quanto tempo e quais horários ele joga, e possivelmente o quão satisfeito ele está com o serviço utilizado. Já um provedor de rede, que somente fornece a conexão com a internet para o usuário e para o provedor de conteúdo, não consegue distinguir estas informações observando o fluxo de rede, ainda que possível identificar que se trata de jogos em nuvem. Esta diferença pode ser vista na Figura 4.1.Como consequência existem duas possibilidades distintas de aplicação de preditores de QoE:

- **Modelo com acordo:** o provedor de rede e de conteúdo possuem um acordo de compartilhamento de dados, e com isso possuem acesso a um conjunto de dados mais completo sobre as partidas. Neste modelo, todos os dados coletados (Secção 4.3) são utilizados para prever o QoE.
- **Modelo sem acordo:** o provedor de rede somente tem acesso aos dados de rede. Neste modelo somente os dados de rede são utilizados.

4.1.1 Modelo de avaliação de QoE com acordo

Na Equação 4.1 temos um modelo que tem como entrada os parâmetros de rede R, os parâmetros de acordo C (usuário, jogo e horário da partida) e os hiper parâmetros Φ. Este modelo gera um número real para o QoE como saída, que no momento de análise

é arredondado para um inteiro mais próximo. Este modelo tem alguns parâmetros de entrada a mais que o sem acordo e se espera maior precisão.

$$f(R, C, \Phi) \mapsto QoE \in R \tag{4.1}$$

Neste caso de uso, o provedor de rede tem cooperação com o provedor de serviço de jogos em nuvem e com isso é possível ter mais informações sobre o usuário e o jogo jogado, como exemplificado na Figura 4.1, modelo com acordo. Podemos supor que o serviço *GeForce Now* fez uma parceria com uma grande provedora de rede como a Oi para melhorar a qualidade do serviço prestado por ambas.

Com este modelo é possível prever como cada usuário se comporta em cada jogo, combinação usuário e jogo, classifica a qualidade de acordo com as características da rede. Com isso o provedor de rede pode identificar usuários mais ou menos exigentes e jogos mais ou menos suscetíveis a quedas de qualidade na rede e balancear a rede de acordo.

Um benefício deste modelo é a diferenciação por tipo de jogo. Isto cobriria casos como, por exemplo, de diferenças de dinâmicas de jogo. Um usuário em um jogo competitivo que precisa de respostas rápidas pode utilizar a maior parte do serviço com menos atrasos e perdas do que um usuário em um jogo com vários diálogos e menus de conversa (M. et al. [2013]). Outro caso é a diferenciação por experiência e expectativa de desempenho de cada usuários. Mesmo que ambos usuários joguem o mesmo jogo ou estilo de jogo, sabendo que um deles é mais experiente e exigente, o provedor de rede pode melhorar o serviço para o usuário exigente, sabendo que ambos ainda vão ter um bom QoE.

Com consequência do número crescente de usuários deste tipo de serviço (J. Bitner [2021]), uma desvantagem é que este modelo pode ter dificuldade com novos usuários ou um usuário que passou a jogar mais[1]. A adição de jogos de forma frequente também não é levada em conta pelo modelo. De forma geral este modelo não irá generalizar tão bem quanto o sem acordo, gerando a necessidade de treinamento constante do modelo para mitigar este efeito.

Em um caso de uso do modelo por um provedor de serviços, é possível treinar com um número muito maior de jogos e usuários, sendo possível ainda o agrupamento de jogos e jogadores em categorias. Um jogo de tiro em primeira pessoa recém lançado por

[1]De acordo com o documento P809 da International Telecommunication Union [2018], o que diferencia um jogador experiente de um casual é a quantidade de horas de jogo semanais, ou seja quanto mais uma pessoa joga, mais experiente se torna. Ao mesmo tempo, o usuário se torna ais exigente será com o serviço prestado, podendo ser tratada como outro usuário, já que a sua subjetividade foi alterada.

exemplo pode encaixar na mesma categoria de *Half-Life 2*, evitando assim a necessidade de treinar o modelo novamente. Da mesma forma, um usuário novo que joga de forma frequente é classificado na categoria de experiente, evitando novamente o treinamento do modelo.

Entendemos que este modelo pode ser empregado tanto por provedoras quanto por serviços de jogos em nuvem para a melhoria dos seus serviços.

4.1.2 Modelo de avaliação de QoE sem acordo

Na Equação 4.2 temos um modelo que tem como entrada os parâmetros de rede R e os hiper parâmetros Φ. Este modelo gera um número real para o QoE como saída, que no momento de análise é arredondado para um inteiro mais próximo. Este modelo tem menos parâmetros de entrada, se espera maior generalização em relação ao com acordo.

$$f(R, \Phi) \mapsto QoE \in R \tag{4.2}$$

Neste caso de uso, o provedor de rede não tem cooperação com o provedor de serviço de jogos em nuvem e fica limitado somente às informações da rede, como exemplificado na Figura 4.1, modelo sem acordo. Neste caso podemos ter como exemplo um provedor de rede como a Oi que quer melhorar a qualidade do serviço somente observando o uso de rede de cada cliente. De acordo com o trabalho de Gigis et al. [2021] é possível analisar os certificados TLS (*Transport Layer Security*) para descobrir com qual serviço ou empresa o usuário está se comunicando. E com isso é possível saber que se trata de comunicação com o servidor do *GeForce Now* por exemplo.

O modelo sem acordo com o provedor de conteúdo precisa de mais dados para conseguir prever o QoE com mais precisão do que o modelo com acordo, e com isso deve generalizar melhor. Para explicar melhor o conceito, precisamos expandir a desvantagem do modelo anterior, que é a vantagem do modelo sem acordo.

Ao olhar somente para os dados da rede e com o conhecimento de se tratar de jogos em nuvem, este modelo deveria prever o QoE de dois fluxos de *streaming* sem saber qual o jogo sendo jogado e qual o usuário. Por mais que preveja com menor acurácia (menos informações, logo menos precisão), ele generaliza melhor e consegue prever mesmo que novos tipos de jogadores ou novos estilos de jogos sejam inseridos ao passar do tempo.

A desvantagem deste modelo é que esperamos que a acurácia seja inferior ao modelo com acordo, mas ainda sendo possível identificar fluxos com QoE muito alto no

qual uma porção dos recursos podem ser redirecionados para fluxos com QoE baixo, ajudando a balancear a rede.

4.2 Ambiente de avaliação

A *testbed* escolhida utiliza de rede local, para garantir que todas as variáveis de rede sejam controladas, permitindo ter dados com menos ruído para o treino do modelo. Outro ponto importante é a possibilidade de simular vários cenários de rede, como redes muito rápidas ou muito lentas, valores de perdas altos ou baixos. Caso fosse utilizada a internet com o controle somente no servidor, seriam introduzidos parâmetros indesejáveis como por exemplo: excesso de atraso, maior número de perda, partidas com mais de 3 minutos, entre outros.

A infraestrutura utilizada nas duas *testbeds* consiste de dois computadores, um agindo como servidor e executando Windows 10 e um cliente rodando Ubuntu 20.04, além de um roteador *gigabit* realizando a comunicação entre o servidor e o cliente. Suas configurações podem ser vistas na Figura 4.2. Todas as partidas foram jogadas em resolução de 1920x1080p e 60 quadros por segundo.

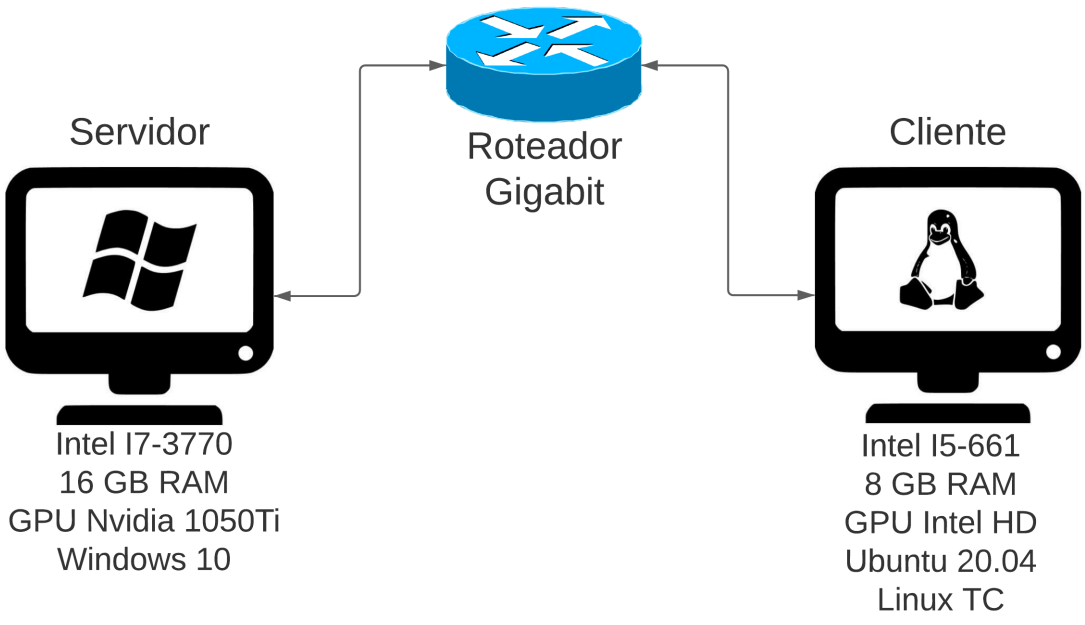


Figura 4.2: Diagrama de montagem da *testbed*

Para obter os dados de impacto das variáveis de rede no QoE, é gerado atraso, *jitter* e perda de pacotes entre o servidor e o cliente com uso das ferramentas TC e

QDisc. Não foi gerado limite na vazão pois a plataforma do *Moonlight* com a API da *Nvidia* trabalha com uso de banda fixo, visto que não suporta troca dinâmica de resolução (assim como o *GeForce Now*, Barman [2019]).

Foram separados o envio de comandos do recebimento de vídeo para diferenciar o impacto individual no QoE. Para isto foram mapeadas as portas da API da *Nvidia* e foram identificadas 2 portas fixas do protocolo RTSP, uma para vídeo/áudio (porta 47998) e outra para comandos (porta 47999). Os valores utilizados nos testes podem ser vistos na Tabela 4.1. Para definir os valores de atraso foram feitos testes preliminares que podem ser vistos no apêndice deste trabalho. Os autores Saif & Othman [2011] realizaram uma pesquisa de valores médios de *jitter* e perda aceitáveis e ruins em transmissões em tempo real na internet e utilizamos os valores encontrados, que podem ser vistos na Tabela 4.1.

	Mínimo	Máximo	Intervalo
Atraso	0 ms	150 ms	25 ms
Jitter	0 ms	25 ms	5 ms
Perda	0 %	3 %	0.5 %

Tabela 4.1: Valores de *jitter*, atraso e perda utilizados.

Seguindo as recomendações da ITU e o documento P809 (International Telecommunication Union [2018]), foram definidos dois jogos de gêneros diferentes, pois se espera que cada um tenha comportamentos diferentes para com as requisições de rede. Ambos os jogos possuem baixo custo computacional de renderização, garantindo sobra de recursos para a codificação e transmissão da partida. Os jogos escolhidos foram:

- *Half-Life 2*, lançado em 2004. De acordo com Steam [2020], é um jogo de tiro de primeira pessoa ambientado em um futuro próximo onde alienígenas invadem a terra.

- *Spelunky*, lançado em 2013. De acordo com Steam [2020], é um jogo de plataforma 2D que objetiva coletar a maior quantidade de ouro enquanto explora uma caverna abandonada.

É esperado que cada jogo gere um conjunto de dados diferentes (International Telecommunication Union [2018]). *Spelunky* é de plataforma e necessita de comandos de controle mais precisos, logo deve ser mais sensível a interferências nestes. Enquanto isso, *Half-Life 2* é de tiro em primeira pessoa, e por isso depende mais de visualização

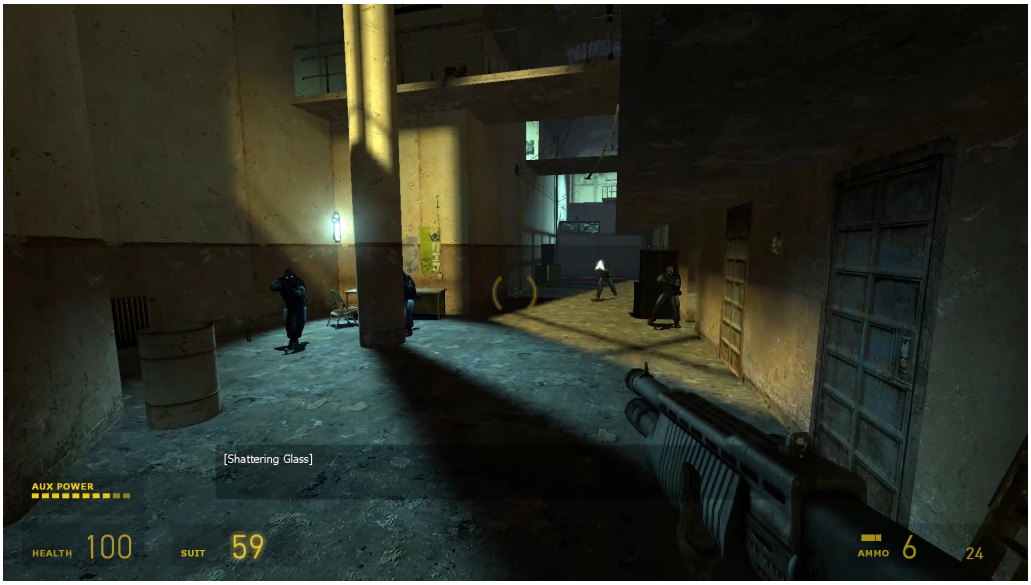
de cenário e reflexos rápidos, logo esperamos que seja mais sensível a interferências no vídeo.

Seguindo as recomendações da ITU no documento P809 (International Telecommunication Union [2018]), cada participante jogou cerca de três minutos em cada partida[2], sendo que cada partida tem um cenário de rede escolhido de forma aleatória dentro dos valores da Tabela 4.1, cobrindo desde o ideal até um extremamente péssimo, distribuição confirmada ao analisar os dados finais. Ao fim de cada partida o participante classifica a qualidade da experiência.

De acordo com a pesquisa de HUANG et al. [2014] e as recomendações tanto de Moonlight [2013] e da Steam [2020], quando a vazão está abaixo do mínimo necessário o mesmo fica não jogável, ou seja, QoE mínimo. No caso da pesquisa de HUANG et al. [2014], para 720p a 30 quadros o recomendado é entre quatro e seis Mbps, enquanto o recomendado pela Moonlight [2013] para 1080p a 60 quadros é 20 Mbps. Levando em conta as pesquisas e recomendações realizadas, utilizamos a vazão como um parâmetro fixo. Todos os cenários de teste e coleta de dados foram realizados em 1920x1080 a 60 quadros por segundo com 20 Mbps de vazão. Os valores utilizados de atraso, *jitter* e perda estão explicados na Tabela 4.1.

A partir dos estudos preliminares foi criado um programa que controla o fluxo de coleta de dados, com uma versão no cliente e uma no servidor, que comunicam usando *socket*. A versão do cliente tem uma interface gráfica na qual o jogador insere o seu nome e escolhe qual jogo deseja jogar, sendo duas opções: *Half-Life 2* (Figura 4.3a) e *Spelunky* (Figura 4.3b). A partir disso o programa utiliza de comandos enviados ao TC e ao QDisc (comandos do *Linux Traffic Control*) para gerar valores de atraso, *jitter* e perda tanto para a porta de recebimento de vídeo quanto para a porta de envio de comandos baseados nos valores vistos na Tabela 4.1. Nos testes preliminares observamos que os valores gerados são uniformes e constantes.

[2]A ITU recomenda para avaliações de desempenho um tempo entre 90 e 180 segundos de jogo.



(a) Captura jogo *Half-Life 2*

(b) Captura jogo *Spelunky*

Figura 4.3: Captura de tela dos jogos utilizados

Após gerar as interferências de rede, o programa do cliente (Figura 4.4a) comanda a plataforma *Moonlight* (Secção 2.2) para iniciar uma partida de 3 minutos de duração com o jogo escolhido (Figura 4.4b) e avisa o programa no servidor para inciar a contagem de tempo. Ao fim dos 3 minutos o programa do cliente termina a partida criada pelo *Moonlight* e pede para o jogador classificar o QoE da partida (Figura 4.4c). O QoE é classificado em uma escala MOS de 7 opções, sendo elas: Extremamente péssimo, Péssimo, Ruim, Razoável, Bom, Excelente e Ideal, que são armazenadas como valores inteiros entre -3 e 3 respectivamente.

(a) Tela inicial

(b) Escolha do jogo

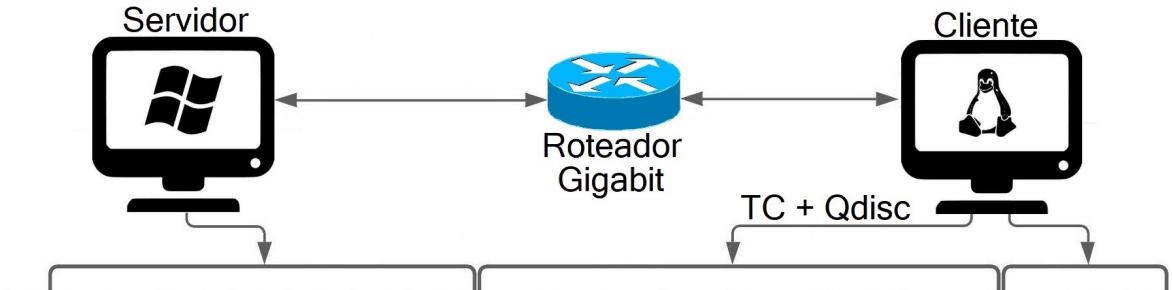
(c) Classificação da partida

Figura 4.4: Captura de tela dos jogos utilizados

Após avaliar a partida, o programa do cliente envia todos os dados para o programa do servidor, que concatena com os dados de uso de *hardware* e tempo de partida e salva os dados em uma tabela no servidor. Um exemplo desta tabela pode ser visto na Figura 4.5, onde as colunas de uso de *hardware* possuam valores variando entre 0 e 100%, as de atraso e *jitter* estão em milissegundos e de perda os valores variam entre 0 e 3%. Os dados coletados em cada partida, e que podem ser vistos no exemplo dado na Figura 4.5, são:

- *QoE* avaliado pelo participante. Ao terminar a partida uma janela aparece pedindo para que o usuário classifique a partida entre extremamente péssimo e ideal.

- Nome utilizado pelo participante[3].
- Tempo de jogo, coletado pela própria máquina.
- Horário e dia da partida, coletado pela própria máquina.
- Média de uso de CPU, coletado pela própria máquina.
- Média de uso de RAM, coletado pela própria máquina.
- Média de uso de GPU (Processador da placa de vídeo), coletado pela própria máquina.
- Média de uso de VRAM (RAM exclusiva da placa de vídeo), coletado pela própria máquina.
- Atraso de vídeo, gerado pelo TC no linux.
- Atraso de comando, gerado pelo TC no linux.
- Perda de vídeo, gerado pelo TC no linux.
- Perda de comando, gerado pelo TC no linux.
- *Jitter* de vídeo, gerado pelo TC no linux.
- *Jitter* de comando, gerado pelo TC no linux.



	Termino Partida	Tempo Jogo	Uso CPU	Uso RAM	Uso GPU	Uso VRAM	Atraso Video	Atraso Comandos	Jitter Video	Jitter Comandos	Perda Video	Perda Comandos	Jogo	NOME	QoE
2	25-06-2021-18-03	3	41	13	27	23	0	75	0	0	0	2	Half-Life 2	Daniel	2
3	13-10-2021-20-13	4	78	28	33	29	0	25	0	10	0	0	Half-Life 2	Daniel	3
4	14-10-2021-17-23	3	75	45	23	20	25	0	0	0	0.5	0	Half-Life 2	Daniel	2
5	26-07-2021-17-30	3	35	15	9	9	150	25	15	5	3	3	Spelunky	Bob	-2
6	26-07-2021-17-36	3	38	15	9	9	25	0	15	0	2.5	0	Spelunky	Bob	2
7	26-07-2021-17-39	3	38	15	9	9	0	50	0	25	0	2	Spelunky	Bob	2
8	13-08-2021-17-34	3	47	14	28	23	100	0	0	0	3	0	Half-Life 2	kattiana	2
9	13-08-2021-17-38	4	27	14	8	8	0	75	0	20	0	0.5	Spelunky	kattiana	3
10	13-08-2021-17-57	3	32	14	9	9	150	0	5	0	0	0	Spelunky	kattiana	2
11	15-10-2021-19-09	4	40	13	23	21	0	125	0	20	3	1.5	Half-Life 2	Marcos	-2
12	15-10-2021-19-12	3	45	13	27	23	0	100	0	0	3	1.5	Half-Life 2	Marcos	1
13	15-10-2021-19-15	3	42	14	24	22	150	150	20	25	1	1	Half-Life 2	Marcos	-1

Figura 4.5: Diagrama de coleta de dados

[3]O nome do participante foi anonimizado na versão final do conjunto de dados

Cada *testbed* salva os dados em um arquivo com extensão *csv* no qual cada partida corresponde a uma linha. Ao fim desta pesquisa os dados vão estar disponibilizados de forma livre para que possam ser utilizados por outros pesquisadores no link https://github.com/danielgtiherege/QoE-Model-for-CloudGaming. Os dados de uso de *hardware* foram utilizados somente para garantir que a máquina não esteja sobrecarregada. Caso em alguma coleta o uso de *hardware* passe de 80% no servidor, esta instância é descartada. Nos testes preliminares observamos que o computador do cliente nunca passa de 50% e 60% de uso no CPU e memória, respectivamente. As avaliações de QoE que estão armazenadas com valores entre menos três e três são mapeadas para entre zero e seis para facilitar a futura normalização dos regressores testados.

Além disso testamos se algum usuário tem avaliações inconsistentes, ou seja, a maioria dos seus dados tem QoE alto para valores de rede péssimos ou o contrário. Mesmo se tratando de uma avaliação subjetiva, cada jogador tem um padrão de avaliação que pode ser observado com a média para cada cenário de QoE. Caso um usuário seja classificado como inconsistente, seus dados são descartados (somente um jogador nos dados coletados). Por último, os dados de rede são normalizados entre 0 e 1, para melhorar a precisão dos algoritmos testados.

4.3 Hiper parâmetros e métricas de avaliação dos modelos

Ao definir o método de avaliação de um regressor é comum o uso do MSE (*Mean Square Error*) ou Erro médio Quadrático. É possível utilizar outras métricas como por exemplo a taxa de acerto do modelo na validação.

Já que um dos nossos focos é a generalização do modelo, e a melhor previsão do QoE, ao utilizar a taxa de acerto na validação nosso modelo deve generalizar melhor que um modelo com MSE, porém com menos precisão. Foram utilizadas ambas, ou seja, cada modelo foi treinado duas vezes, uma utilizando o acerto como métrica e outra utilizando o MSE. Desta forma temos como observar as duas possibilidades e escolher a melhor.

Antes de elaborar sobre os resultados de cada cenários de avaliação, alguns algoritmos foram testados, regressores e classificadores. Todos os algoritmos testados foram implementados utilizando as bibliotecas *sklearn* e *keras* em Python. Escolhemos dois algoritmos de árvore, sendo um classificador (*Decision Tree*) e um regressor (*Random Forest*). Dois neuronais, sendo um o tradicional MLP ((*Multi Layer Perceptron*)) e outro uma implementação de MLP com uso de aceleração de GPU (chamaremos

de *TensorFlow* MLP). E por último o *AdaBoost* que tenta melhorar um algoritmo já treinado de árvore, que utilizamos em conjunto com o *Decision Tree*. Os algoritmos escolhidos tem como objetivo variar os tipos de aprendizado de máquina, para buscar o que melhor se adaptasse ao nosso conjunto de dados. Alguns destes são utilizados por trabalhos como o de Sabet et al. [2020] para gerar uma árvores de decisão.

Foi realizada uma busca de hiper parâmetros nos algoritmos testados utilizando o método *Grid search*, com os valores pesquisados na Tabela 4.2. Este método consiste em uma busca e teste por todas as combinações até encontrar a melhor. Em todos os algoritmos foi utilizado o método de *cross validation*, ou validação cruzada em uma tradução livre, no qual o conjunto de dados é dividido em pedaços iguais e um deles é utilizado para teste e os outros para treino, em busca da melhor divisão. Inicialmente dividimos o conjunto de dados em 70% treino, 20% teste e 10% validação, e no *cross validation* utilizamos dez divisões. Nestas dez divisões, nove são para treino e uma para teste. para cada combinação de parâmetros o algoritmo foi repetido dez vezes para busca do melhor resultado.

	AdaBoost	*Random Forest*	*Multi Layer Perceptron*	*Decision Tree*	*TensorFlow MLP*
Número de estimadores	25 a 150	1 a 100	NA	NA	NA
Máximo de *features*	NA	1 a 6	NA	1 a 6	NA
Máximo de profundidade	NA	5 a 20	NA	1 a 20	NA
Critério	*Square, Linear* ou *Exponential*	NA	NA	*Poisson, MAE , MSE* ou *Friedman MSE*	MSE
Splitter	NA	NA	NA	*best* ou *random*	NA
taxa de aprendizado	0.001, 0.005, ou 0.01 ou 0.1	NA	0.001, 0.005, ou 0.01 ou 0.1	NA	NA
Método de ativação	NA	NA	*identity, logistic, tanh* ou *relu*	NA	*relu*, *linear*, *softmax*, *softplus*, *softsign*, *tanh*, *selu*, *gelu* e *elu*
Número camadas ocultas	NA	NA	2 a 4	NA	1 a 9
Número de neurônios	NA	NA	32 ou 64	NA	4 a 128 intervalo 4
Máximo número de iterações	NA	NA	250, 500, 750 ou 1000	NA	1000

Tabela 4.2: Hiper parâmetros dos algoritmos avaliados, NA - não aplicável.

Para o algoritmo de *Decision Tree* no modelo com acordo, os melhores hiper parâmetros foram: critério *Friedman MSE*, *splitter best*, seis de máximo de *features* e oito de máximo de profundidade. Para o algoritmo *AdaBoost* no modelo com acordo, os melhores hiper parâmetros foram: 75 de número de estimadores, taxa de aprendizado de 0.1 e critério linear. Para o algoritmo *RandomForest* no modelo com acordo, os melhores hiper parâmetros foram: 46 de número de estimadores, seis de máximo de *features* e nove de máximo de profundidade. Para o algoritmo MLP no modelo com acordo, os melhores hiper parâmetros foram: três camadas ocultas, 32 neurônios por

camada, 750 de número de iterações, método de ativação *logistic* e taxa de aprendizado de 0.001. Para o algoritmo *TensorFlow* MLP no modelo com acordo, os melhores hiper parâmetros foram: ativação *softmax*, três camadas internas e 16 neurônios por camada.

Para o algoritmo de *Decision Tree* no modelo sem acordo, os melhores hiper parâmetros foram: critério MSE, *splitter best*, seis de máximo de *features* e cinco de máximo de profundidade. Para o algoritmo *AdaBoost* no modelo sem acordo, os melhores hiper parâmetros foram: 125 de número de estimadores, taxa de aprendizado de 0.01 e critério *square*. Para o algoritmo *RandomForest* no modelo sem acordo, os melhores hiper parâmetros foram: 41 de número de estimadores, três de máximo de *features* e sete de máximo de profundidade. Para o algoritmo MLP no modelo sem acordo, os melhores hiper parâmetros foram: duas camadas ocultas, 32 neurônios por camada, 250 de número de iterações, método de ativação *logistic* e taxa de aprendizado de 0.01. Para o algoritmo *TensorFlow* MLP no modelo sem acordo, os melhores hiper parâmetros foram: ativação *relu*, seis camadas internas e 120 neurônios por camada.

Modelo	Com acordo		Sem acordo	
	MSE	Acerto Validação	MSE	Acerto Validação
TensorFlow MLP	0.75	41.83%	1.00	37.87%
Decision Tree	0.91	43.41%	1.08	37.36%
AdaBoost	0.73	43.53%	1.02	39.01%
Random Forest	0.73	45.05%	0.97	41.21%
Multi Layer Perceptron	0.92	44.06%	1.0	37.62%

Tabela 4.3: Resultados dos algoritmos avaliados

O algoritmo de *AdaBoost* utiliza o de *Decision Tree* com os seus melhores hiper parâmetros como base. Assim como ele o *Random Forest* obteve o mesmo MSE no modelo com acordo, porém foi melhor no modelo sem acordo. Dado o melhor resultado em relação aos outros algoritmos, o *Random Forest* foi escolhido para ser utilizado para o nosso modelo. Todos os algoritmos foram testados com todas as 2020 instâncias coletadas, sendo que 90% são utilizadas no *cross validation* com dez divisões e 10% para validação. Os resultados podem ser vistos na Tabela 4.3, todos os resultados são com os melhores hiper parâmetros selecionados para cada algoritmo.

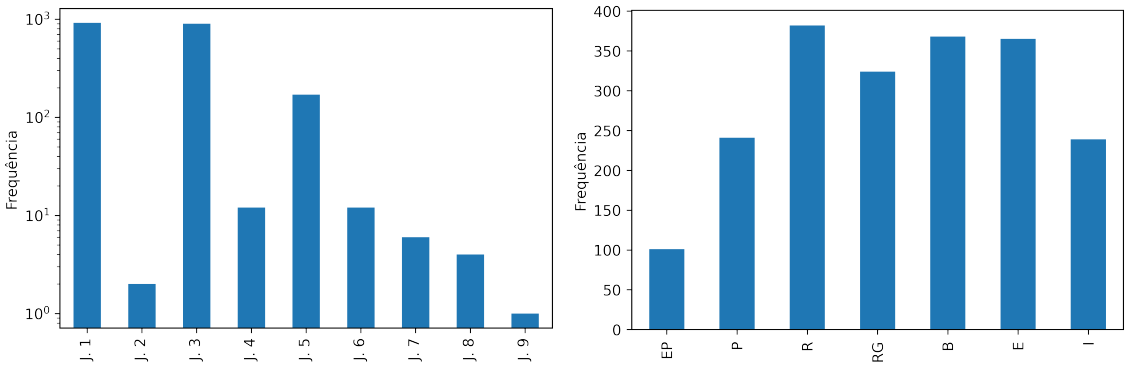
De acordo com Aroussi & Mellouk [2016], algoritmos de aprendizado de máquina baseados em árvores e florestas e de *boosting* tem melhor desempenho em bases de dado que mapeiam QoE para rede. Ainda de acordo com Miranda et al. [2020], algoritmos de *boosting* com árvore foram melhores para QoE sobre rede. Também obtivemos os mesmos resultados, nos quais *Random Forest* e *AdaBoost* foram os melhores.

Capítulo 5

Resultados

Foram coletadas 2020 partidas de nove jogadores, que totalizam mais de cem horas de jogo. Dos dados gerados 83.76% foram do jogo *Half-Life 2* e 16.24% do jogo *Spelunky*. O número de pessoas foi menor que o esperado, visto que a coleta ocorreu durante a pandemia do COVID-19 e era necessário participação presencial para evitar interferência da internet nos dados. Com a exceção de um participante da graduação, os restantes são alunos de pós-graduação.

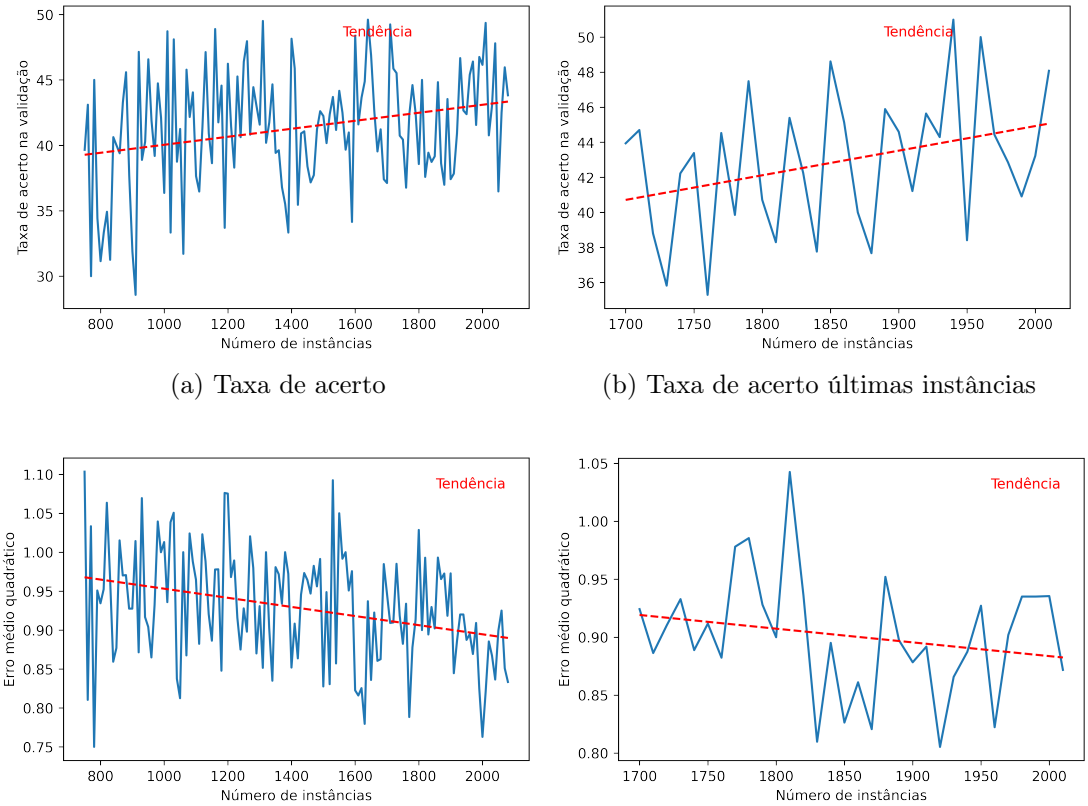
Um modelo baseado em simulação costuma ter um número de instâncias muito maior que o coletado. Entretanto, o volume medido em tempo de jogo é considerável. Considerando que cada instância equivale a aproximadamente 3 minutos de jogo, temos 102 horas e 29 minutos de dados coletados. Os valores de interferências na rede estão bem distribuídos, como esperado. As distribuições dos dados podem ser vistas na Figura 5.1. A distribuição de QoE está bem equilibrada enquanto a de jogadores está desbalanceada. O jogador 1 jogou 45.35% das partidas, o jogador 3 jogou 44.41% e o jogador 5 jogou 8.42%. Os outros jogadores jogaram juntos 1.82% das partidas.



(a) Jogadores (b) QoE

Figura 5.1: Histogramas de QoE e de Jogadores por Instâncias.
EP - Extremamente péssimo, P - Péssimo, R - Ruim, RG - Regular.
B - Bom, E - Excelente, I - Ideal.

5.1 Modelo com acordo



(a) Taxa de acerto

(b) Taxa de acerto últimas instâncias

(c) Erro médio quadrático

(d) Erro médio das últimas instâncias

Figura 5.2: Variação do modelo com acordo, por número de instâncias

Uma de nossas principais preocupações durante o treinamento foi o pequeno número de instâncias, que se devem principalmente à dificuldade em recrutar voluntários durante a pandemia do COVID-19. Por isso, analisamos a tendência do MSE e a taxa de acerto dos modelos à medida que o número de instâncias aumentava no treinamento. Essa tendência é um indicador de como a precisão do modelo deve se comportar para conjuntos de dados com mais instâncias de treinamento.

Ao analisar os acertos por instâncias do modelo com acordo, nas últimas 300 instâncias a taxa de acerto atinge um pico de 50.0% com MSE de 0.82 para 1960 instâncias que pode ser visto nas Figuras 5.2c e 5.2a. Esta taxa de acerto pode ser melhor visualizada no mapa de calor visto na Figura 5.3. A tendencia de melhora acontece ao analisar a curva total, e ao analisar somente os últimos dados (Figura 5.2d) vemos que o erro médio quadrático ainda está caindo, o que significa que se mais dados fossem coletados, uma melhora aparentemente seria encontrada. Mesmo com metade dos acertos sendo precisos, se levarmos em consideração o fato que a maioria

das predições está dentro de uma faixa de tolerância de mais ou menos uma casa, o modelo consegue prever muito bem a região de QoE que será avaliada.

Ao comparar o MSE entre o *dataset* de treino e teste, temos 0.82 contra 0.94 respectivamente, logo temos o MSE do teste pouco acima do treino, logo sem *overfit* ou *underfit*.

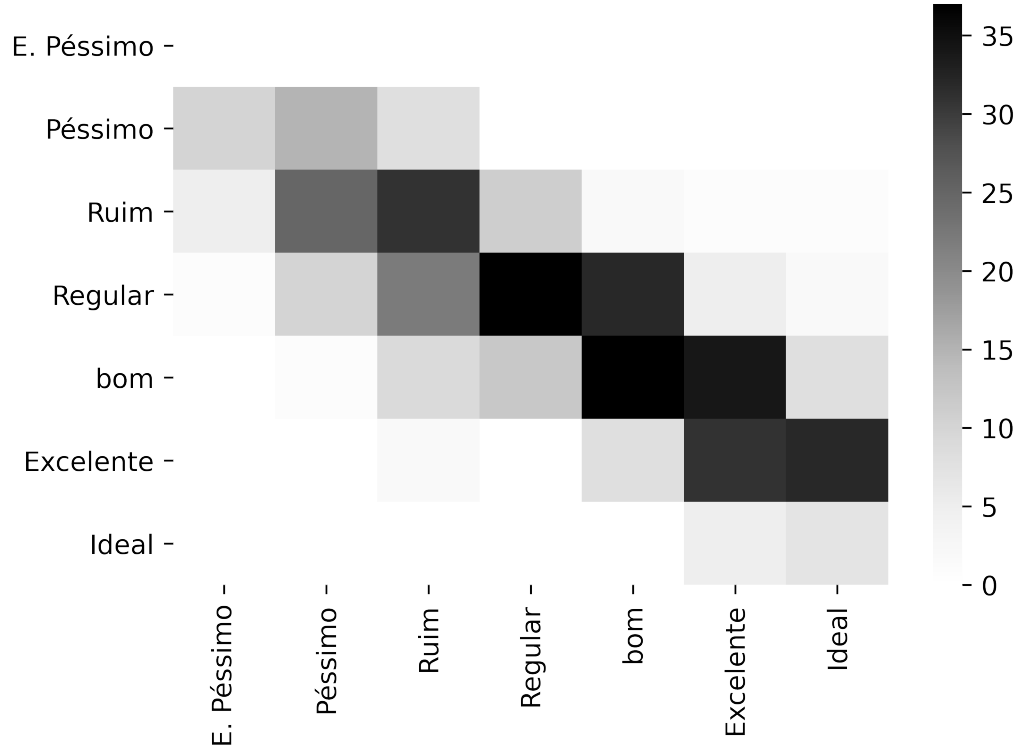


Figura 5.3: Mapa de calor das predições do modelo com acordo.
Eixo horizontal - Predições do modelo.
Eixo vertical - Valores reais.

5.1.1 Agrupamento dos jogadores

Utilizamos de dois métodos para testar os melhores agrupamentos. No primeiro método utilizamos do algoritmo de *KMeans* e *KNN* utilizando identificação do jogador e os dados de rede para encontrar os melhores grupos e depois treinamos este agrupamento no regressor *RandomForest*. Geramos o resultado de SSE (Soma dos erros quadráticos) por cada valor de k para nossa base de dados, gerando o gráfico da Figura 5.4. Utilizando os métodos de *Silhouette* e *Elbow Method* analisamos a métrica estatística de distância de cada grupo, concluindo que os números de grupos mais interessantes são de dois e três. Adicionamos também o valor quatro ao nosso teste para ficarmos

com os mesmos valores de grupos que o segundo método.

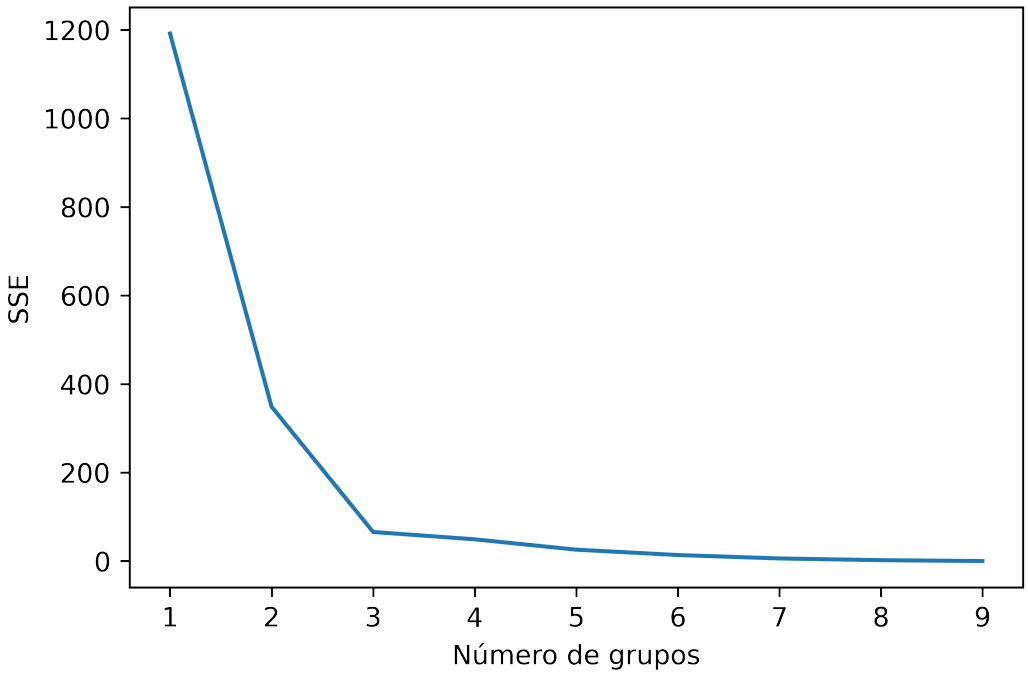


Figura 5.4: Teste para achar o melhor valor de K no algoritmo *KMeans*

No segundo método calculamos as possíveis combinações de jogadores em grupos, com dois, três ou quatro grupos e realizamos uma busca exaustiva utilizando o regressor *RandomForest* em cada possibilidade para selecionar a melhor. Esta opção é interessante por representar um cenário possível (Subseção 4.1.1) no mundo real, quando um novo jogador entra na plataforma ele pode ser encaixado em um grupo que case com o seu perfil. Lembramos que o modelo com acordo e sem agrupamento neste teste obteve MSE de 0.88 com acerto de 46.58%, já que o treinamento é estocástico. Ao treinar o modelo com agrupamento via *KMeans*, o melhor agrupamento obteve 0.85 de MSE e 46.81% de acerto, com a seguinte divisão:

- GRUPO 1: Jogador 1;
- GRUPO 2: Jogador 5;
- GRUPO 3: Jogadores 2, 3, 4, 6, 7, 8 e 9.

Ao treinar o modelo com agrupamento via busca exaustiva, o melhor agrupamento obteve 0.75 de MSE e 49.07% de acerto, com a seguinte divisão:

- GRUPO 1: Jogador 5 e 8;

- GRUPO 2: Jogador 1, 2 e 7;
- GRUPO 3: Jogador 3, 4, 6 e 9.

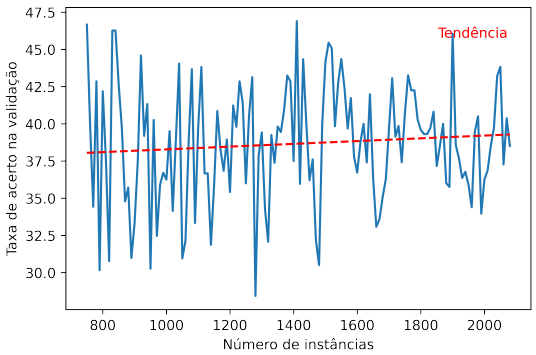
Com este agrupamento foi possível reduzir o MSE em aproximadamente 15%, os resultados dos agrupamentos podem ser vistos na Tabela 5.1. O documento P809 da International Telecommunication Union [2018] separa jogadores gerais em dois grupos: os novatos que jogam menos de três horas semanais e os experientes que jogam mais de três. Outro método de separação foi coberto na Seção 2.1 onde os jogadores são separados por motivação dentro do jogo, mostrando que é interessante agrupar os jogadores de acordo com alguma característica comum.

Agrupamento também é uma solução interessante para evitar retreino do modelo, visto que ao invés de fazer uma busca por melhores grupos sempre que novos jogadores forem inseridos, é possível classificar e inserir os novos jogadores em grupos já existentes. Neste caso cada novo jogador a aderir o sistema de jogos em nuvem pode ser inserido em um grupo e o QoE previsto daquele jogador será semelhante ao do do grupo.

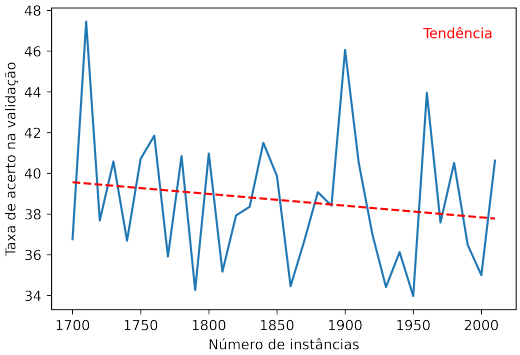
Método	MSE	Acerto
Sem agrupamento	0.88	46.58%
KMeans	0.85	46.81%
Busca exaustiva	0.75	49.07%

Tabela 5.1: Resultados dos métodos de agrupamento

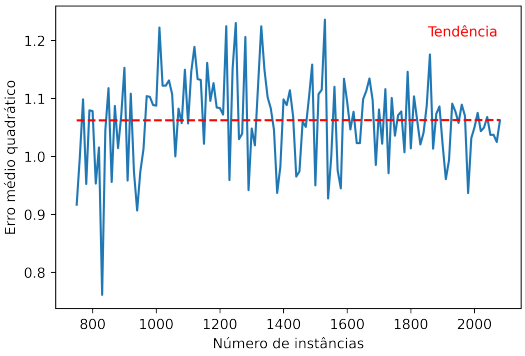
5.2 Modelo sem acordo



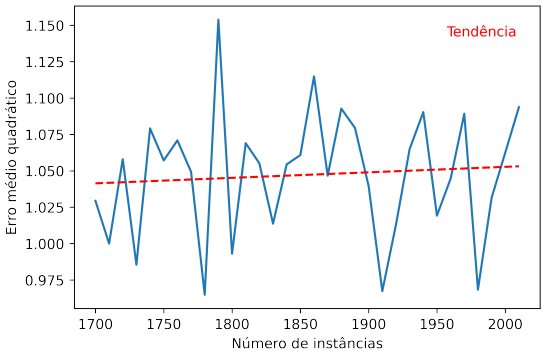
(a) Taxa de acerto



(b) Taxa de acerto das últimas instâncias



(c) Erro médio quadrático



(d) Erro médio quadrático das últimas instâncias

Figura 5.5: Variação do modelo sem acordo, por número de instâncias

De acordo com a Figura 5.6, que corresponde ao mapa de calor das predições do modelo sem acordo, podemos ver que a taxa de acerto está de acordo com a da Figura 5.5c onde nas últimas 300 instâncias, o MSE atinge é de 0.96 para 1980 instâncias com acerto de 40.50%. Considerando que o QoE varia entre 0 e 6 (Extremamente Péssimo até Ideal) temos que o modelo classifica corretamente várias instancias e também classifica várias instâncias que seriam classificadas como excelente como bom. Este é um erro aceitável, visto que está dentro do MSE atingido e se tratando de uma base de dados subjetiva.

Ao comparar o MSE entre o *dataset* de treino e teste, temos 0.96 contra 1.33 respectivamente, assim como o modelo com acordo, tempos aqui o MSE do teste um pouco acima do treino, logo sem *overfit* ou *underfit*.

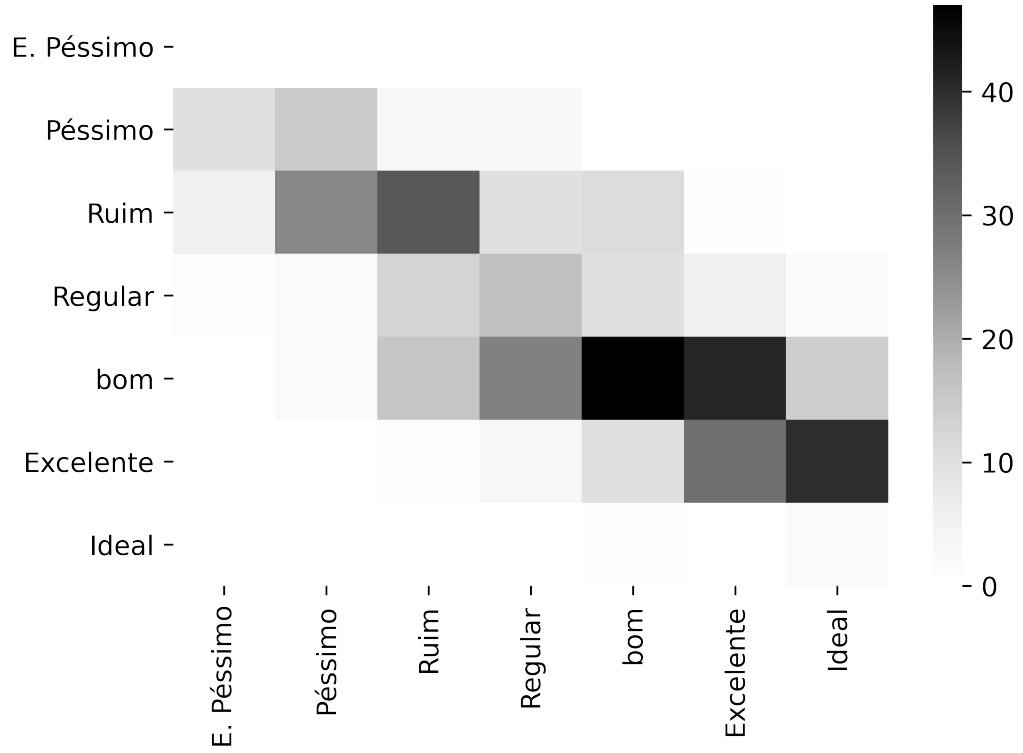
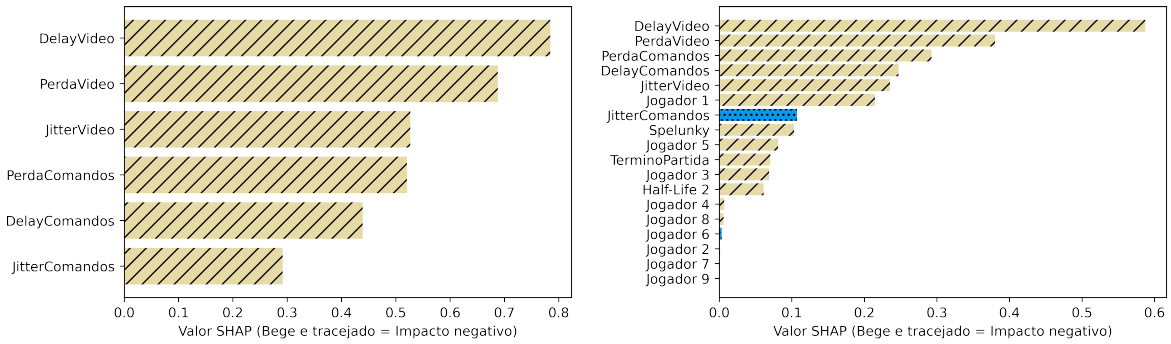


Figura 5.6: Mapa de calor das predições do modelo sem acordo.
Eixo horizontal - Predições do modelo.
Eixo vertical - Valores reais.

Em nenhum momento o modelo classifica como bom, excelente ou ideal cenários que são ruins, péssimos ou extremamente péssimos. Mesmo que a taxa de acerto seja de 40.50% como pode ser visto na Figura 5.5a e que o MSE esteja relativamente alto, por se tratar de um modelo subjetivo baseado em coleta real de dados, os resultados são positivos. Levando em consideração que a maioria das predições está dentro de uma faixa de tolerância de mais ou menos uma casa, o modelo consegue prever muito bem a região de QoE que será avaliada.

5.3 Explicação dos modelos

Como tratamento inicial foi utilizada a biblioteca SHAP (Lundberg & Lee [2017]) para validar os parâmetros de entradas do nosso modelo, ou seja, os parâmetros de rede que realmente interferem no QoE. SHAP é uma biblioteca para linguagem *Python* que usa teoria de jogos para explicar a saída de um modelo de aprendizado de máquina de acordo com a entrada.



(a) SHAP modelo sem acordo (b) SHAP modelo com acordo

Figura 5.7: SHAP de ambos os modelos

Como vemos na Figura 5.7, o atraso (*delay*) de vídeo é o parâmetro mais importante na degradação do QoE em ambos os modelos seguido de perda de vídeo. Porém ao analisarmos o modelo com acordo, vemos que a perda de comandos passa a ser mais significativo em comparação com o sem acordo e o *jitter* de comandos passa a ser positivo em relação ao sem acordo. Essa mudança pode significar uma interação do mesmo com os jogos ou jogadores.

De acordo com a Figura 5.7b é possível notar que o jogador seis tem impacto positivo, o que indica que comparado com outros jogadores, sente menos a degradação da rede, podendo ser um jogador menos experiente. Também existe uma diferença entre o jogo *Spelunky* em relação ao jogo *Half-Life 2*, sendo o segundo menos exigente. Por último podemos notar que o alguns jogadores por terem jogado poucas partidas tem baixa influência.

5.4 Capacidade de generalização dos modelos

Com objetivo de testar a capacidade de generalizar do modelo, alguns jogadores participaram do mesmo experimento porém com um jogo diferente que não fez parte da coleta principal. Este novo e pequeno conjunto de dados tem como objetivo participar somente da parte de validação do regressor para testar a taxa de acerto com um terceiro jogo.

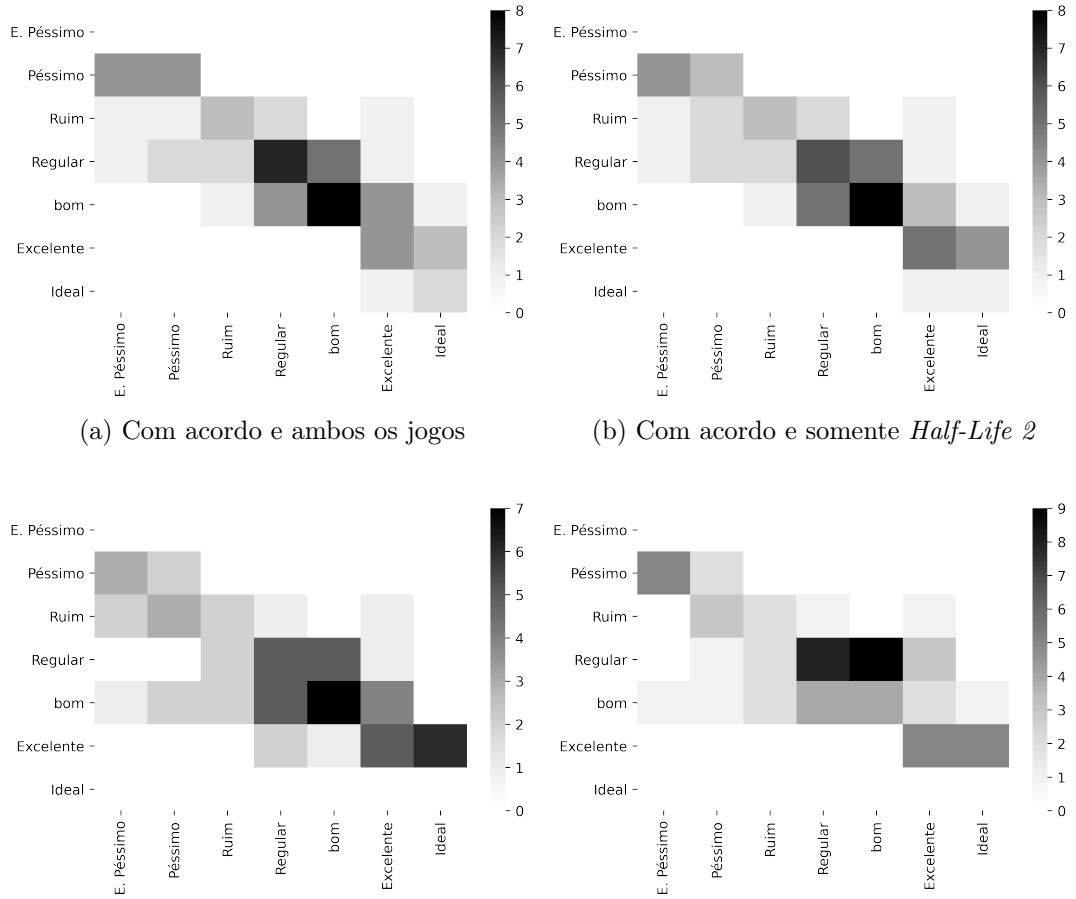
O jogo escolhido foi o jogo F.E.A.R, um jogo de tiro de primeira pessoa lançado em outubro de 2005 de acordo com a Steam [2020], aproximadamente um ano após *Half-Life 2*. O enredo tem como base vários acontecimentos sobrenaturais enquanto um grupo de soldados lida com uma organização terrorista. Foram realizadas 62 partidas do jogo F.E.A.R, com dois jogadores, os jogadores um e três.

O jogo escolhido é do mesmo gênero de *Half-Life 2* usado durante a coleta principal. Para este teste, usaremos os dois modelos (com acordo e sem acordo) e em duas situações: uma treinando o regressor com o conjunto de dados completo que contém os dois jogos principais (*Half-Life 2* e *Spelunky*) e uma somente com o *Half-Life 2*. E após teremos quatro cenários de validação para os dados coletados com o jogo F.E.A.R. Nenhum dos cenários utiliza agrupamento, somente o regressor do modelo base.

Modelo	Cenário	MSE	Acerto	MSE base	Acerto base
Com acordo	Ambos os jogos	1.1	45.16%	0.71	44.12%
	Somente *Half-Life 2*	1.13	41.94%		
Sem acordo	Ambos os jogos	1.63	33.87%	1.01	37.89%
	Somente *Half-Life 2*	1.5	33.87%		

Tabela 5.2: Resultados dos cenários de validação com jogo F.E.A.R

Ao realizar a validação do modelo com os dados do jogo F.E.A.R (Tabela 5.2), temos que no modelo com acordo o MSE piorou, subindo de 0.71 para 1.1 no melhor dos casos, porem o acerto foi ligeiramente superior. No modelo sem acordo subiu de 1.01 para 1.5 no melhor dos casos, com uma pequena perda na porcentagem de acertos. As taxas de acerto se mantiveram dentro dos valores de validação originais. Podemos ver na Figura 5.8 todos os mapas de calor dos acertos de todos os quatro cenários.



(a) Com acordo e ambos os jogos

(b) Com acordo e somente *Half-Life 2*

(c) Sem acordo e ambos os jogos

(d) Sem acordo e somente *Half-Life 2*

Figura 5.8: Mapas de calor validação com o jogo F.E.A.R

5.5 Discussão

Ao analisar ambos os modelos e a validação com um terceiro jogo, podemos dizer que alcançamos o objetivo inicial de gerar um previsor de QoE para as perturbações de rede. Nossos dois modelos conseguem prever com uma precisão dentro do intervalo do MSE (mais ou menos 0.72 para o com acordo e 1 para o sem acordo) o QoE avaliado não só para os jogos *Half-Life 2* e *Spelunky*, mas também para um terceiro jogo, *FEAR*, que mesmo que obtendo um menor MSE, consegue manter as taxas de acerto. Com isso mostramos que o modelo sem acordo generaliza, ou seja, consegue prever o QoE dentro da mesma precisão dos jogos originais. O modelo com acordo também generaliza considerando o mesmo gênero de jogo. Com isso é possível reduzir o custo de treinamento para grandes catálogos, ou evita retreino para novos jogos inseridos.

É interessante observar as diferenças do modelo com acordo e sem acordo, que ficam mais evidentes com um maior número de instâncias. O modelo sem acordo estabiliza em valores de MSE e de acerto, já o modelo com acordo rende melhores resultados e mostra que se beneficiaria de um maior conjunto de dados. Para o modelo com acordo, nosso teste com agrupamento mostra que o modelo pode ao mesmo tempo melhorar a precisão e facilitar a entrada de novos usuários no conjunto de dados. Entretanto, não conseguimos iniciar a coleta de dados antes do início da pandemia

As características mais impactantes para degradação de QoE em ambos os modelos foram o atraso de vídeo e a perda de vídeo (Figura 5.7). O terceiro e quarto mais importantes mudam de acordo com o modelo, sendo o *jitter* de vídeo e a perda de comandos no modelo sem acordo e a perda de comandos e o atraso de comandos no modelo com acordo. É interessante notar que o atraso de vídeo é o mais importante em ambos. Alguns trabalhos relacionados como Jarschel et al. [2011] e Barman [2019] mostram o mesmo, porém o nosso trabalho possui como diferencial a avaliação baseada em usuários reais e o modelo final.

Durante a realização do trabalho encontramos algumas limitações, sendo algumas delas:

- Número baixo de jogadores. A coleta de dados foi realizada durante a pandemia do COVID-19, o que limitou a disponibilidade das pessoas para se deslocar à UFMG. Por isso, nos disponibilizamos para levar o ambiente de testes à casa dos participantes, e mesmo assim tivemos uma quantidade baixa de pessoas interessadas.

- Mobilidade da *testbed*. Composta de dois computadores, dois monitores, teclado, mouse e um roteador, gerando dificuldade de transporte e necessitando a presença dos participantes no laboratório para jogar e gerar dados.

- Pelo número baixo de jogadores, alguns se dispuseram a jogar várias partidas, o que gerou um banco de dados desbalanceado em relação ao número de partidas por jogador. Mesmo assim os modelos ainda obtiveram bons resultados.

Se observarmos, todas as limitações têm relação com a pesquisa ter ocorrido durante a pandemia do COVID-19. Começamos a montagem das *testbeds* e preparo da metodologia ainda sem pandemia, esperando que o grande fluxo de alunos na universidade ajudaria na coleta de dados.

Capítulo 6

Conclusão

Nosso trabalho modelou o QoE em jogos em nuvem levando em conta as possíveis variações de qualidade na rede utilizada. Criamos dois modelos de QoE, baseados em dois cenários de cooperação entre a operadora de rede e a prestadora de serviços de jogos em nuvem. O modelo sem acordo pode ser utilizado por uma operadora de rede para tentar gerir e melhorar o QoE dos usuários que estão em partidas de jogos em nuvem. Já o modelo com acordo pode ser utilizado em uma parceria entre uma operadora de rede e uma prestadora de serviços de jogos em nuvem.

Para realizar a coleta de dados foi utilizado o programa *moonlight* em conjunto com a plataforma da *Nvidia Shield* para criar as *testbeds* utilizadas e o TC/QDisc em Linux para gerar as interferências na rede. Geramos atraso, *jitter* e perda nos fluxos de vídeo e comandos, sendo envio de comandos um fluxo cliente para servidor e recebimento de vídeo o oposto.

Analisamos os dados gerados com um regressor da biblioteca *Sklearn*, o *Random-Forest*. Ambos os modelos tem uma boa precisão, estimando o QoE que o jogador irá avaliar, com mais ou menos uma casa de erro, em uma escala de zero a seis (Extremamente Péssimo, Péssimo, Ruim, Regular, Bom, Excelente e Ideal). Entendemos que ambos os modelos possuem aplicações práticas com este nível de precisão. No modelo com acordo testamos o uso de agrupamento de usuários, e vimos que, dado um maior catálogo de jogos e jogadores, é possível agrupar e treinar o modelo para prever o QoE de novos jogadores de acordo com grupos que eles se encaixem. Prevemos que a mesma ideia seja aplicável a novos jogos de acordo com o gênero que se encaixarem.

Utilizamos um terceiro jogo somente na parte da validação do modelo para testar a generalização do mesmo e mostramos que ambos os modelos conseguem manter a mesma taxa de acerto com um terceiro jogo, mesmo que aumentando um pouco o MSE. Neste ponto, notamos a importância de utilizar também a taxa de acerto, e não

somente o MSE.

6.1 Trabalhos futuros

Como trabalhos futuros sugerimos a possibilidade de realizar a mesma pesquisa com o *GeForce Now* e realizar a coleta dos dados de rede no momento da partida, podendo inserir mais interferências além da internet. O *GeForce Now* é interessante por ser a plataforma mais utilizada no momento de escrita do trabalho.

Seria interessante realizar a pesquisa com monitores de resoluções mais altas como 4K e monitores de alta frequência de atualização como 144 Hz para verificar o aumento do uso de vazão e a sensibilidade a degradação de rede. Com isso seria possível avaliar a viabilidade destas plataformas em tipos de tela mais avançadas.

6.2 Resultados obtidos da pesquisa

Durante o processo desta pesquisa obtivemos os seguintes produtos:

- A base de dados gerada está pública, assim como o *software* desenvolvido para a coleta. Ambos estão disponibilizados em https://github.com/danielgtiherege/QoE-Model-for-CloudGaming.
- Co-orientação do aluno de graduação em Sistemas de Informação Michael Páris A. Xavier.
- Artigo em preparação em conjunto com Marcos Magno de Carvalho e Michael Páris A. Xavier sobre os modelos produzidos neste trabalho – em submissão para a conferência IEEE NetSoft 2022.

Referências Bibliográficas

Allied Market Research (2021). Cloud gaming market statistics: 2030. `https://www.alliedmarketresearch.com/cloud-gaming-market-A07461`. [Online; accessed 30-November-2021].

Aroussi, S. & Mellouk, A. (2016). Statistical evaluation for quality of experience prediction based on quality of service parameters. *23rd International Conference on Telecommunications*.

Bankov, D.; Khorov, E.; Lyakhov, A. & Sandal, M. (2019). Enabling real-time applications in wi-fi networks. *International Journal of Distributed Sensor Networks*, 15.

Barman, N. (2019). *An Objective and Subjective Quality Assessment for Passive Gaming Video Streaming*. Tese de doutorado, Kingston University London, Faculty of Science, Engineering and Computing, Kingston University London, United Kingdom.

Cai, W.; Hong, Z.; Wang, X.; Chan, H. C. B. & Leung, V. C. M. (2015). Quality-of-experience optimization for a cloud gaming system with ad hoc cloudlet assistance. *IEEE Transactions on Circuits and Systems for Video Technology*, 25(12):2092–2104.

Callet, P. L.; Möller, S. & Perkis, A. (2012). Qualinet white paper on definitions of quality of experience. *European Network on Quality of Experience in Multimedia Systems and Services (COST Action IC 1003)*, 4(5):2.

Chen, S.; Chu, C.-Y.; Yeh, S.-L.; Chu, H.-H. & Huang, P. (2014). Modeling the qoe of rate changes in skype/silk voip calls. *IEEE/ACM Transactions on Networking*.

D. Peppiatt (2021). Steam has more monthly active users than both xbox and playstation. `https://www.vg247.com/steam-users-overtake-xbox-playstation`. [Online; accessed 12-October-2021].

Domenico, A. D.; Perna, G.; Trevisan, M.; Vassio, L. & Giordano, D. (2021). A Network Analysis on Cloud Gaming: Stadia, GeForce Now and PSNow.

Eibe Frank, M. A. H. & Witten, I. H. (2012). The weka workbench. `https://www.cs.waikato.ac.nz/ml/weka/Witten_et_al_2016_appendix.pdf`. [Online; accessed 26-November-2021].

Fiedler, M.; Hossfeld, T. & Tran-Gia, P. (2010). A generic quantitative relationship between quality of experience and quality of service. *IEEE Network*, 24(2):36–41.

Gigis, P.; Calder, M.; Manassakis, L.; Nomikos, G.; Kotronis, V.; Dimitropoulos, X.; Katz-Bassett, E. & Smaragdakis, G. (2021). Seven years in the life of hypergiants' off-nets. Em *Proceedings of the 2021 ACM SIGCOMM 2021 Conference*, SIGCOMM '21, p. 516–533, New York, NY, USA. Association for Computing Machinery.

HUANG, C.-Y.; CHEN, K.-T.; CHEN, D.-Y.; HSU, H.-J. & HSU, C.-H. (2014). Gaminganywhere: The first open source cloud gaming system. *ACM Transactions on Multimedia Computing*.

International Telecommunication Union (2018). Itu-t p809: Subjective evaluation methods for gaming quality.

J. Bitner (2021). Nvidia geforce now has surpassed 10 million subscribers. `https://www.thegamer.com/geforce-now-10-million-subscribers/`. [Online; accessed 12-October-2021].

J. Schreier (2021). Google's stadia problem? a video game unit that's not googley enough. `https://www.bloomberg.com/news/articles/2021-02-26/google-video-game-unit-stadia-struggled-to-be-googley-enough`. [Online; accessed 18-October-2021].

Jackson, S. A. & Marsh, W. W. (1996). Development and validation of a scale to measure optimal experience: The flow state scale.

Jarschel, M.; Schlosser, D.; Scheuring, S. & Hoßfeld, T. (2011). An evaluation of qoe in cloud gaming based on subjective tests. pp. 330–335.

Juluri, P.; Tamarapalli, V. & Medhi, D. (2015). Measurement of quality of experience of video-on-demand services: A survey. *IEEE Communications Surveys & Tutorials*, 18(1):401--418.

Krishnappa, D. K.; Bhat, D. & Zink, M. (2013). Dashing youtube: An analysis of using dash in youtube video service. Em *38th Annual IEEE Conference on Local Computer Networks*, pp. 407–415.

Lundberg, S. M. & Lee, S.-I. (2017). A unified approach to interpreting model predictions. Em Guyon, I.; Luxburg, U. V.; Bengio, S.; Wallach, H.; Fergus, R.; Vishwanathan, S. & Garnett, R., editores, *Advances in Neural Information Processing Systems 30*, pp. 4765--4774. Curran Associates, Inc.

M., S.; L., S.-K. & M., M. (2013). The impact of user, system, and context factors on gaming qoe: a case study involving mmorpgs.

Maia, O. B. (2015). Um sistema de gerenciamento da qualidade de experiência orientada à transmissão de vídeos para dispositivos móveis em redes sem fio. *Programa de pós-graduação em engenharia elétrica*.

Microsoft (2020). Microsoft xcloud. `https://canaltech.com.br/games/microsoft-abrira-beta-do-streaming-de-games-project-xcloud-em-outubro-150661/`. [Online; accessed 14-April-2020].

Miranda, G.; Macedo, D. F. & Marquez-Barja, J. M. (2020). A qoe inference method for dash video using icmp probing. Em *2020 16th International Conference on Network and Service Management (CNSM)*, pp. 1–5.

Mohr, N. (2018). Everything you need to know about video encoding. `https://www.pcgamer.com/everything-you-need-to-know-about-video-encoding/`.

Moonlight (2013). Moonlight. `https://moonlight-stream.org/`. [Online; accessed 14-April-2020].

Nvidia (2020). Geforce now. `https://www.nvidia.com/en-us/geforce-now/`. [Online; accessed 14-April-2020].

Parsec (2020). Parsec. `https://parsecgaming.com/`. [Online; accessed 14-April-2020].

Peñaherrera-Pulla, O. S.; Baena, C.; Fortes, S.; Baena, E. & Barco, R. (2021). Measuring key quality indicators in cloud gaming: Framework and assessment over wireless networks. *Sensors 2021*.

R. Smith (2020). Geforce now leaves beta, game streaming service launches with new rtx servers. `https://www.anandtech.com/show/15451/geforce-now-leaves-beta-launches-with-rtx-on`. [Online; accessed 12-October-2021].

Rainway (2019). Rainway. https://rainway.com/. [Online; accessed 14-April-2020].

Rao, A. & Lanphier, R. (1996). Real time streaming protocol(rtsp). [Online; accessed 30-June-2021].

Ravaja, N.; Turpeinen, M.; Lindley, C.; Vorderer, P.; Mathiak, K. & IJsselsteijn, W. (2006). Fuga: Fun of gaming. [Online; accessed 21-June-2021].

Reichl, P.; Egger, S.; Schatz, R. & D'Alconzo, A. (2010). The logarithmic nature of qoe and the role of the weber-fechner law in qoe assessment. Em *2010 IEEE International Conference on Communications*, pp. 1–5.

Rheinberg, F. (2015). Die flow-kurzskala (fks) übersetzt in verschiedene sprachen the flow-short-scale (fss) translated into various languages.

Rodrigues, L. C.; Lopes, R. A. S. P. & Mustaro, P. N. (2007). Impactos sócio-culturais da evolução dos jogos eletrônicos e ferramentas comunicacionais: um estudo sobre o desenvolvimento de comunidades virtuais de jogadores. *SBGames*.

Sabet, S. S.; Schmidt, S.; Zadtootaghaj, S.; Griwodz, C. & Moller, S. (2020). Delay sensitivity classification of cloud gaming content. *International Workshop on Immersive Mixed and Virtual Environment Systems (MMVE'20)*.

Saif, A. & Othman, M. (2011). Network load and packet loss optimization during handoff using multi-scan approach. *International Arab Journal of Information Technology*, 8(1). ISSN 16833198.

Schulzrinne, H. (1992). Rtp: The real-time transport protocol. [Online; accessed 30-June-2021].

Sony (2020). Playstation now. https://www.playstation.com/en-us/explore/playstation-now/. [Online; accessed 14-April-2020].

Steam (2020). Steam hardware survey. https://store.steampowered.com/hwsurvey/Steam-Hardware-Software-Survey-Welcome-to-Steam?l=portuguese. [Online; accessed 27-March-2020].

W. Landim (2021). Geforce now começa a funcionar no brasil em teste limitado. https://adrenaline.com.br/noticias/v/70811/geforce-now-comeca-a-funcionar-no-brasil-em-teste-limitado. [Online; accessed 12-October-2021].

Apêndice

Testes preliminares

Para decidir os intervalos dos valores de atraso foi realizado um teste preliminar com dois participantes. Foram realizadas 3 repetições para cada experimento, sendo a classificação de QoE entre -3 e 3 e os valores para cada repetição podem ser vistos na tabela 6.1.

	Comandos	Vídeo	Ambos
0ms	3	3	3
50ms	3	2	1
100ms	1	1	-1
150ms	1	0	-2
200ms	0	-1	-3
250ms	0	-1	-3
300ms	-1	-2	-3
350ms	-1	-2	-3
400ms	-2	-3	-3
500ms	-3	-3	-3
600ms	-3	-3	-3
700ms	-3	-3	-3
800ms	-3	-3	-3
900ms	-3	-3	-3
1000ms	-3	-3	-3

Tabela 6.1: Atraso e QoE

Ao plotar todos os gráficos com o último ponto sendo 500ms temos a Figura 6.1 e podemos notar que a interação entre vídeo e comandos parece forte.

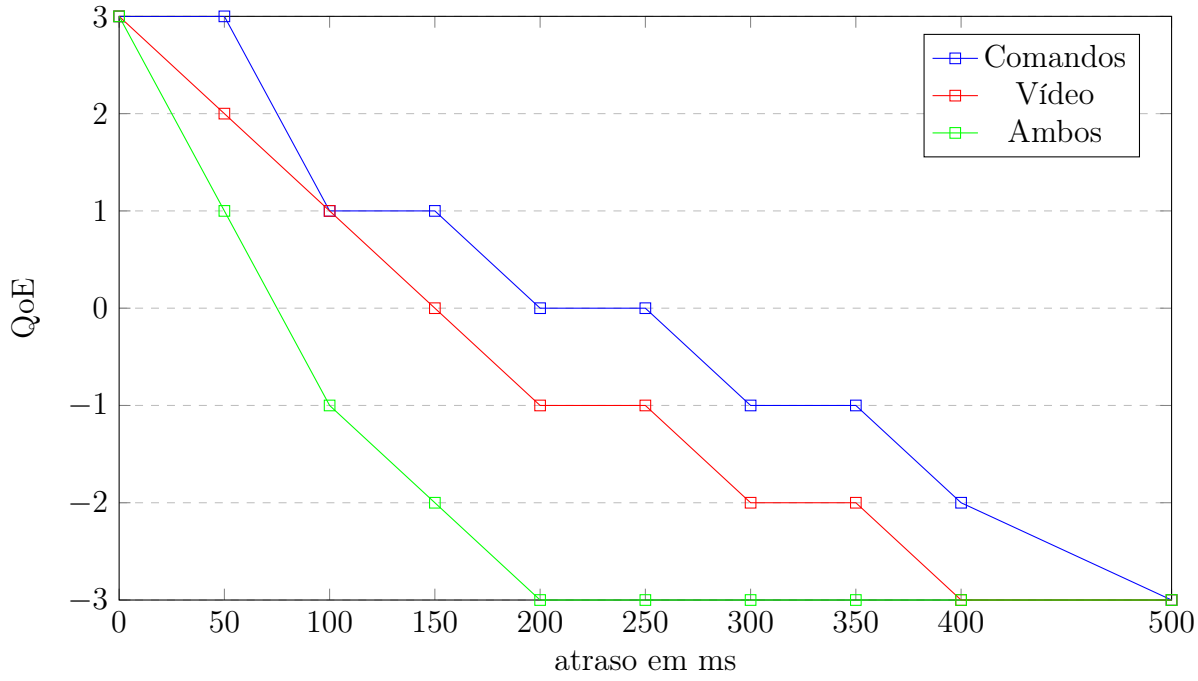


Figura 6.1: Impacto do atraso no QoE

Regressão Múltipla e Projeto fatorial 2K

A partir dos dados coletados na seção anterior será feita uma analise com projeto 2K e regressão múltipla para avaliar e definir quais devem ser os próximos passos.

Projeto fatorial 2K

Iremos chamar de X_a o atraso dos comandos e de X_b o atraso do vídeo. Tomaremos $X_a = -1$ quando o atraso for de 500ms e $X_a = 1$ quando for 0ms, o mesmo para X_b. Com isso temos $Y = Q_0 + Q_a X_a + Q_b X_b + Q_{ab} X_a X_b$. Resolvendo o sistema chegamos na função $Y = -1.5 + 1.5X_a + 1.5X_b + 1.5X_a X_b$, podemos calcular a variância amostral e o SST e a partir disto temos $SSA = SSB = SSAB = 9$ e cada um explica 33,33%.

Regressão somente do atraso nos comandos

Usando os dados da Tabela 6.1 na parte de atraso de comandos, podemos chegar em $QoE = 2.868 - 0.01203x$ onde x é o atraso em ms no envio de comandos. Agora é possível comparar o modelo com os dados e avaliar erro e R^2 como pode ser visto na Tabela 6.2. É possível chegar ao SSE somando os E^2, temos $SSE = 1.5196$, e calculamos a porcentagem explicada pela regressão de 95.64%. Ao observar somente o

impacto dos comandos temos uma boa regressão, mostra que o impacto de adicionar atraso ao envio de comandos impacta fortemente a experiência do usuário.

atraso	QoE	QoE Calculado	Erro	Erro^2
0ms	3	2.868	-0.132	0.0174
50ms	3	2.2665	-0.7335	0.5380
100ms	1	1.665	0.665	0.4422
150ms	1	1.0635	0.0635	0.0040
200ms	0	0.462	-0.538	0.2894
250ms	0	-0.1395	-0.1395	0.0195
300ms	-1	-0.741	0.259	0.0671
350ms	-1	-1.3425	-0.3425	0.1173
400ms	-2	-1.944	0.056	0.0031
450ms	-3	-3.147	-0.147	0.0216

Tabela 6.2: Erro e E^2 da regressão $QoE = 2.868 - 0.01203x$

Regressão somente do atraso no vídeo

Usando os dados da Tabela 6.1 na parte de atraso de vídeo, podemos chegar em $QoE = 2.258 - 0.01242y$ onde y é o atraso em ms no envio de vídeo. Agora é possível comparar o modelo com os dados e avaliar erro e R^2 como pode ser visto na Tabela 6.2. É possível chegar ao SSE somando os E^2, temos $SSE = 2.7425$, e a porcentagem explicada pela regressão de 92.03%.

atraso	QoE	QoE Calculado	Erro	Erro^2
0ms	3	2.258	-0.742	0.5506
50ms	2	1.637	-0.363	0.1318
100ms	1	1.016	0.016	0.0003
150ms	0	0.395	0.395	0.1560
200ms	-1	-0.226	0.774	0.5991
250ms	-1	-0.847	0.153	0.0234
300ms	-2	-1.468	0.532	0.2830
350ms	-2	-2.089	-0.089	0.0079
400ms	-3	-2.71	0.29	0.0841
450ms	-3	-3.952	-0.952	0.9063

Tabela 6.3: Erro e E^2 da regressão $QoE = 2.258 - 0.01242y$

Regressão de atraso em ambos

Usando os dados da Tabela 6.1 agora de forma completa, ou seja considerando ambos os atrasos. Foi encontrado um problema que não era possível calcular a matriz inversa ao utilizar todos os dados, pois a coluna de 1 e 2 eram iguais. por isso foram restringidos os dados para ter valores mais representativos, visto que com 200ms de atraso em ambos já era alcançado -3 de QoE. Os valores utilizados podem ser vistos na Tabela 6.4.

	0ms	50ms	100ms	150ms	200ms
0ms	3	3			
50ms	2	1			
100ms			-1		
150ms				-2	
200ms					-3

Tabela 6.4: Colunas: atraso comandos, linhas: atraso vídeo

Com isso é possível chegar em $QoE = 2.951 - 0.02605x - 0.006054y$ onde x é o atraso em ms no envio de comandos e y é o atraso em ms no envio de vídeo. Agora é possível comparar o modelo com os dados e avaliar erro e R^2 como pode ser visto na Tabela 6.5. É possível chegar ao SSE somando os E^2, temos $SSE = 1.1568$, e a porcentagem explicada pela regressão de 96.76%. Os resultados desta analise preliminar mostram que um intervalo entre 0ms e 150ms de atraso é o suficiente.

Atraso Vídeo	Atraso Comandos	QoE	QoE Calculado	Erro	Erro^2
0ms	0ms	3	2.951	-0.049	0.0024
0ms	50ms	3	2.6483	-0.3517	0.1237
50ms	0ms	2	1.6485	-0.3515	0.1236
50ms	50ms	1	1.3458	0.3458	0.1196
100ms	100ms	-1	-0.2594	0.7406	0.5485
150ms	150ms	-2	-1.8646	0.1354	0.0183
200ms	200ms	-3	-3.4698	-0.4698	0.2207

Tabela 6.5: Erro e E^2 da regressão $QoE = 2.951 - 0.02605x - 0.006054y$