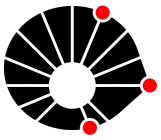

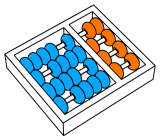

Transformação de TV Boxes piratas em computadores de baixo custo suportados por nuvem computacional

A. Ribeiro *D. Yoshioka* *G. Ramirez* *J. Silva*
M. Santos *M. Sousa* *M. Cândido* *M. Rodrigues*
V. Pietrobom *J. F. Borin*

Relatório Técnico - IC-PFG-23-46
Projeto Final de Graduação
2023 - Dezembro

UNIVERSIDADE ESTADUAL DE CAMPINAS
INSTITUTO DE COMPUTAÇÃO

Transformação de TV Boxes piratas em computadores de baixo custo suportados por nuvem computacional

Alan Freitas Ribeiro Douglas Daisuke Kaneiwa Yoshioka Guilherme Ramirez
Juliana Bronqueti da Silva Marcio Dos Santos Mariana Alves de Sousa
Matheus Augusto Da Silva Cândido Matheus Otávio Rodrigues
Vitor Rodrigues Pietrobom Juliana Freitag Borin

Resumo

A recente pandemia de Covid-19 escancarou problemas quanto ao acesso da população brasileira à Internet e demonstrou que o processo de inclusão digital no país não está completo. Durante esse período desafiador, tornou-se evidente que a falta de acesso à internet não é apenas uma questão de conveniência, mas sim uma barreira significativa para o acesso a informações essenciais, educação remota, serviços de saúde online e oportunidades de trabalho. A crise revelou desigualdades profundas, onde muitos brasileiros enfrentam dificuldades para participar de atividades cotidianas devido à falta de conectividade. Estudantes sem acesso à Internet enfrentaram obstáculos no aprendizado remoto e professores tiveram problemas para ministrar suas aulas, prejudicando a qualidade do ensino de muitos cidadãos.

Existem disparidades significativas de acesso à internet entre diferentes regiões do Brasil. Enquanto algumas áreas urbanas têm altos índices de conexão, áreas rurais e periféricas enfrentam maior dificuldade de acesso. No Brasil, tal realidade econômica impõe desafios no acesso a equipamentos tecnológicos para uma vasta gama da população, e é nesse contexto que a criação de computadores acessíveis se torna fundamental.

Levando em consideração os fatores supracitados, a construção de computadores pessoais de baixo custo pode ser uma solução relevante e é justamente o objetivo deste projeto. Em recente operação a Receita Federal brasileira confiscou um grande número de TV Boxes que seriam utilizadas para transmissão ilegal de conteúdo. Este trabalho explorou o uso desses equipamentos como base para a construção de computadores de baixo custo. A partir da instalação de um novo sistema operacional e de ajustes necessários, é possível aproveitar o hardware dessas TV Boxes de maneira positiva. Ao conectar esses dispositivos a uma infraestrutura de desktop remoto, abre-se a ainda a possibilidade de melhorar a capacidade de processamento do sistema e construir um dispositivo de baixo custo e que pode ser utilizado para diversos propósitos, dentre eles o pedagógico.

Índice

1 Introdução

Diversas operações da Receita Federal ao longo do país, alinhadas com o Plano de Ação de Combate à Pirataria, apreenderam inúmeras TV Boxes, pequenos aparelhos eletrônicos importados de forma irregular para realizar a pirataria de canais pagos, filmes e outros conteúdos restritos.

Este tipo de ação costuma resultar em destruição de todo o material apreendido, o que, além de custoso, tem como consequência a geração de resíduos que precisam ser descartados. A fim de evitar poluição ambiental e dar um destino mais apropriado para os equipamentos apreendidos, a Receita Federal estabeleceu parcerias com instituições de ensino superior para que pudessem explorar as oportunidades de uso dos equipamentos, especialmente, para fins de educação e inclusão digital. A UNICAMP é uma das instituições que estabeleceu parceria tendo recebido 200 TV Boxes[1]. Há previsão de receber outras 700 ainda este ano. Os equipamentos recebidos tem sido utilizados em pesquisas em nível de graduação e pós-graduação e também em disciplinas de projeto do cursos de Computação.

No caso de estudo deste Projeto Final de Graduação (PFG), optou-se por transformar as TV Boxes em um minicomputador usado para apoiar a realização de atividades da disciplina de Algoritmos e Programação de Computadores (MC102)[2], que é uma disciplina introdutória de programação presente no currículo de diversos cursos de graduação da UNICAMP. A disciplina é ministrada por professores do Instituto de Computação para aproximadamente 800 alunos a cada semestre.

A escolha deste estudo de caso se deu pois, como a disciplina envolve uma grande quantidade de atividades práticas de programação, há grande demanda por computadores, não só nos horários de aula, mas também para estudo extra-classe. Adicionalmente, há vários institutos/faculdades da Unicamp que têm número restrito de computadores disponíveis para os alunos fora dos horários de aula e muitos alunos não tem computador pessoal.

Uma equipe de 9 alunos do curso de Engenaria de Computação, sob orientação da Profa. Juliana Freitag Borin, se dividiu em três linhas de atuação para que o desenvolvimento deste projeto fosse possível. As linhas de atuação e seus objetivos são:

1. sistemas operacionais: levantamento de sistemas operacionais disponíveis para o hardware da TV Box bem como testes de desempenho do equipamento em comparação com outros equipamentos comumente utilizados pelos alunos;

2. segurança: testes de segurança com a TV Box para verificar a existência de comunicação com algum servidor centralizado para controle remoto ou coleta de dados sem a permissão do usuário e investigação de estratégia para inviabilizar que uma TV Box transformada para outro fim tenha o software original (e ilegal) restaurado;

3. computador de baixo custo: avaliação de uma estratégia para transformação da TV Box em computador de baixo custo baseada em uma arquitetura em nuvem para execução de um desktop remoto.

2 Sistemas operacionais para a TV Box

Para a realização deste trabalho, foi fornecido à nossa equipe o aparelho de TV Box *Tanix TX2 - RK3229*, um sistema que foi projetado para ter bom custo-benefício sem abrir mão de qualidade de *streaming*, possuindo suporte para saída de vídeo em 4K. O dispositivo possui as seguintes especificações:

- *CPU*: *RK3229*, 1.5GHz *Quad-core ARM Cortex-A7*
- Memória *RAM*: *DDR3 2GB*
- Memória *ROM*: *eMMC Flash 8GB*

Originalmente, a TV Box vinha com *Android 7* instalado e então buscamos alguns possíveis sistemas operacionais para substituir o original, levando em consideração desempenho e usabilidade, visto que, os estudantes

[1] Jornal da Unicamp: Unicamp vai transformar em computadores aparelhos de TV Box apreendidos pela Receita
[2] https://www.ic.unicamp.br/ mc102/

estão acostumados a utilizar nos laboratórios de ensino computadores com sistemas operacionais *Linux* ou *Windows*.

Após realizar algumas pesquisas, encontramos alguns sistemas operacionais (SO) que poderiam ser bons candidatos para funcionar com as TV Boxes, dentre eles estavam o *Armbian*, *Puppy Linux*, *Bodhi Linux*, *AntiX Linux* e *Porteus Linux*. Além disso, em paralelo, experimentamos instalar outras versões de *Android* no equipamento. O problema encontrado com os sistemas operacionais foi a incompatibilidade com o processador, *Rockchip RK3229 ARM 32-bits*. A hipótese inicialmente adotada era de que as distribuições *ARM* desses sistemas operacionais poderiam funcionar diretamente no *hardware* da TV Box, o que não se provou verdade ao longo do trabalho. Ao tentar instalar uma distribuição *ARM* do *Puppy LinuX* e do *Bodhi Linux* utilizando um programa disponibilizado pela equipe do *Armbian*, chamado *Multitool* [3], que facilita a gravação de imagens, *backups* e outras funções através do terminal ou *SSH*, a TV Box entra em um estado de instalação infinita e não completa. O único SO que completou a instalação e funcionou de fato foi o *Armbian*, o qual tinha uma versão própria para o *RK322X*. A concordância é que em termos de sistema operacional, uma distribuição customizada *ARM* deve ser feita para que este funcione corretamente com o processador que temos nessa TV Box.

Ao pesquisar sobre sistemas *Linux* mais específicos, alguns com aplicações em roteamento e mídia foram encontrados, como o *Libreelec* [4], sistema operacional focado em mídia e o *OpenWrt* [5] outro sistema operacional, com foco em roteamento de rede. O único encontrado que satisfazia nossas necessidades era o *Armbian*, que, além de ter distribuições adaptadas tanto com quanto sem interface visual, tem uma garantia de manutenção e suporte no futuro [6].

Porém, ao testarmos o *Armbian*, o aparelho não apresentou bom desempenho. Fizemos um teste na distribuição com interface gráfica e, no geral, notamos muitos engasgos e travamentos, até mesmo em tarefas mais simples, como execução no terminal e navegação *web*. Portanto, consideramos assim, o *Android*, como sistema operacional mais apropriado para utilização neste aparelho.

2.1 Tentativas com distribuições *Android*

Nosso objetivo principal era testarmos diferentes *ROMs* do *Android* para encontrarmos distribuições mais enxutas que pudessem melhorar o desempenho do aparelho. Para facilitar este processo, procuramos inicialmente instalar o *Team Win Recovery Project* (*TWRP*) na TV Box. Chegamos a extrair a imagem de *recovery* original do dispositivo, porém, não obtivemos sucesso no restante do procedimento, pois o *bootloader* do aparelho estava bloqueado. Tentamos desbloquear via linha de comando, mas, apesar do comando de *OEM Unlock* retornar resultado de sucesso, o *bootloader* continuou bloqueado.

Além do *TWRP*, tentamos instalar *ROMs* de aparelhos semelhantes, que utilizam o mesmo *chipset RK3229*, contudo, os métodos convencionais de *flash* utilizando *fastboot* e o *bootloader* não funcionaram.

2.2 Procedimentos

Utilizando este aparelho de TV Box, percebemos que alguns procedimentos padronizados de aparelhos *Android* não estavam funcionando como o esperado. Para conseguir conectar o dispositivo via *ADB* (*Android Debug Bridge*), precisamos realizar os seguintes passos:

1. Conectar uma das entradas do cabo *USB* macho-macho na primeira porta *USB* da TV Box (ao lado da entrada de cartão *SD*), a conexão só irá funcionar nela.

2. Conectar a outra entrada do cabo *USB* a uma porta *USB* do computador

3. Na TV Box, acessar Configurações (*Settings*) >Opções de desenvolvedor (*Developer Options*) >Habilitar a opção Configurações *USB* (*USB Settings*)

4. Caso a opção *Developer Options* não esteja habilitada, ir em: Sobre (*About*) e clicar repetidamente na informação de *Build*, pronto, aparecerá uma mensagem dizendo que você é um desenvolvedor e a TV Box está conectada via *ADB*.

[3] Ferramenta *Multitool*
[4] Libreelec
[5] OpenWrt 22.03-rc6 “Build” for TV-boxes (rk322x)
[6] Armbian for RK322X TV Boxes

Para obtermos a imagem *recovery* do aparelho seguimos o seguinte procedimento:

1. Após conectar ao *ADB*, utilize o comando *adb root*.

2. Digite *adb shell* e espere o terminal conectar a TV Box.

3. Com o terminal da TV Box conectado, vá até o diretório *platform* utilizando o comando *cd /dev/block/-platform*

4. Execute o comando *ls* para ver o nome da subpasta da sua plataforma (no caso do equipamento em que realizamos este passo é "30020000.rksdmmc") e a acesse por meio do comando *cd nome da pasta*

5. Agora navegue até a pasta *by-name*: *cd by-name*

6. Entrando na pasta, execute o comando *ls -l*, esse comando irá retornar uma lista contendo todos os blocos de memória e seu respectivo *link*, anote o diretório do bloco que corresponde ao do *Recovery* (no caso do dispositivo de teste é o diretório */dev/block/mmcblk0p8*)

7. Para salvar o arquivo *.img* execute o seguinte comando:

```
# Para salvar na pasta raiz da memoria da TV Box:
$ dd if={diretorio do seu recovery} of=/sdcard/recovery.img
# Exemplo, no caso do dispositivo de teste o diretorio
# era "/dev/block/mmcblk0p8", logo:
$ dd if=/dev/block/mmcblk0p8 of=/sdcard/recovery.img
```

2.3 Teste de desempenho

Por fim, com o intuito de testar o desempenho do aparelho, implementamos [7] uma aplicação em Flutter para rodar nos dispositivos e traçar alguns comparativos entre a TV Box e outros computadores comumente utilizados pelos alunos na Unicamp. Foram planejados três testes para avaliarmos: i) a capacidade de processamento, ii) a eficiência de memoria, desempenho gráfico, latência e iii) o desempenho do sistema de armazenamento do dispositivo. O aplicativo possui os seguintes testes:

1. Teste de ordenação: Para testarmos a capacidade de processamento do aparelho e sua eficiência de memoria, foram criados vetores de forma aleatoria contendo de 10^1 até 10^8 elementos. Medimos o tempo decorrido entre cada execução do algoritmo de ordenação padrão do *Dart* (*Quicksort dual pivot*).

2. Teste dos quadrados: Para testarmos a capacidade de processamento e o desempenho gráfico do dispositivo, foram criados quadrados que se moviam aleatoriamente pela tela. A cada 30 segundos de execução o número de quadrados aumentava. Foi medido o *FPS* (*Frames per Second*) de cada intervalo de tempo com uma quantidade x de quadrados na tela.

3. Teste de escrita e leitura: Por fim, com o intuito de testarmos a latência e o desempenho do sistema de armazenamento do dispositivo, neste teste, o programa criava arquivos e escrevia uma quantidade *n* de linhas com tamanho fixo de 20 caracteres em cada uma. Depois de finalizar a escrita dos arquivos, a aplicação lia e carregava os dados em uma variável. Foi medido o tempo de escrita e leitura para cada arquivo de *n* linhas.

Para executarmos os testes, escolhemos quatro dispositivos:

[7] Repositorio do código Flutter para teste de desempenho: <https://github.com/Necctares/PerformanceTest>

***Fedora Desktop* (Computador do laboratório de ensino do Instituto de Computação)**

- Sistema Operacional: *Fedora Linux*, versão 37
- Arquitetura: *x86_64*
- Memória *RAM*: 16 GB
- Processador: *Intel i7-12700*
- Nº de *Threads*: 20

Ubuntu Desktop

- Sistema Operacional: *Ubuntu*, versão 22.04
- Arquitetura: *x86_64*
- Memória *RAM*: 16 GB
- Processador: *Intel i5-7600*
- Nº de *Threads*: 4

***Tanix TX2 RK3229* (TV Box)**

- Sistema Operacional: *Android*, versão 7
- Arquitetura: *ARM 32 bits*
- Memória *RAM*: 2 GB
- Processador: *ARM-Cortex A7*
- Nº de *Threads*: 4

***Motorola Moto G Stylus* (Smartphone)**

- Sistema Operacional: *Android*, versão 13
- Arquitetura: *ARM 64 bits*
- Memória *RAM*: 6 GB
- Processador: 4x *Cortex-A78* + 4x *Cortex-A55*
- Nº de *Threads*: 8

Durante a execução dos testes em cada dispositivo, mantivemos a resolução 1920 x 1080p 60 Hz, com exceção do *Smartphone*, em que mantivemos a resolução nativa (1080 x 2400p 60 Hz). As Tabelas 1, 2, 3 e 4 apresentam os resultados alcançados:

Aparelhos / Elementos	*Fedora Desktop*	*Ubuntu Desktop*	*TV Box*	*Moto G Stylus*
10	3 µs	1 µs	30 µs	3 µs
100	23 µs	18 µs	244 µs	27 µs
1000	274 µs	226 µs	2468 µs	316 µs
10000	3600 µs	2827 µs	29518 µs	3808 µs
100000	33070 µs	24991 µs	223127 µs	33222 µs
1000000	0,34 s	0,24 s	2,23 s	0,37 s
10000000	3,24 s	2,42 s	21,80 s	3,28 s
100000000	32,33 s	23,72 s	250,95 s	32,84 s

Tabela 1 – Teste de Ordenação

Elementos \ Aparelhos	*Fedora Desktop*	*Ubuntu Desktop*	*TV Box*	*Moto G Stylus*
10	1000,00 fps	1000,00 fps	667,37 fps	920,00 fps
100	1000,00 fps	999,90 fps	446,77 fps	774,93 fps
500	989,37 fps	933,33 fps	149,07 fps	574,97 fps
1000	859,43 fps	907,13 fps	70,00 fps	523,10 fps
5000	193,03 fps	229,13 fps	9,93 fps	150,60 fps
10000	101,47 fps	93,50 fps	5,67 fps	58,13 fps
20000	40,67 fps	38,13 fps	2,77 fps	24,10 fps

Tabela 2 – Teste dos Quadrados

Linhas \ Aparelhos	*Fedora Desktop*	*Ubuntu Desktop*	*TV Box*	*Moto G Stylus*
100	1914 µs	2335 µs	7263 µs	1618 µs
1000	2846 µs	1981 µs	6482 µs	703 µs
5000	3661 µs	1478 µs	12412 µs	2488 µs
10000	6629 µs	2468 µs	22059 µs	3703 µs
150000	11059 µs	6085 µs	34885 µs	3692 µs

Tabela 3 – Teste de Escrita

Linhas \ Aparelhos	*Fedora Desktop*	*Ubuntu Desktop*	*TV Box*	*Moto G Stylus*
100	469 µs	442 µs	4257 µs	527 µs
1000	327 µs	383 µs	3855 µs	658 µs
5000	442 µs	604 µs	5146 µs	786 µs
10000	642 µs	811 µs	6860 µs	1313 µs
150000	4430 µs	903 µs	19114 µs	1581 µs

Tabela 4 – Teste de Leitura

Após a execução dos testes, conseguimos ter noção da diferença de *hardware* entre a TV Box utilizada e de alguns dispositivos que utilizamos em nosso dia a dia. A diferença é relativamente alta, principalmente se compararmos com o *Fedora Desktop*, o aparelho atualmente utilizado em nossos laboratórios de computação, e que serviu de base comparativa durante as pesquisas de experiência de usuário em nosso projeto.

Por fim, testamos também mudar a resolução da TV Box, taxa de atualização e os métodos de desenho na tela, *p* (varredura progressiva) e *i* (varredura entrelaçada). Executamos novamente o Teste dos Quadrados, obtendo os seguintes resultados:

Elementos \ Resolução	1920 x 1080 60 Hz i	1920 x 1080 60 Hz p	1920 x 1080 30 Hz p	1280 x 720 60 Hz p	720 x 576 60 Hz p
10	805,37	667,37	814,67	638,17	716,73
100	677,13	446,77	712,23	427,43	521,4
500	115,4	149,07	121,13	115,43	124,7
1000	75,37	70	76	65,6	58,47
5000	10,07	9,93	10,37	9,83	9,93
10000	5,47	5,67	5,4	5,07	5,17
20000	2,6	2,77	2,83	2,7	2,77

Tabela 5 – FPS médio no Teste dos Quadrados em Diferentes Resoluções

Pelos resultados percebemos que, ao contrário do que esperávamos, os fatores de taxa de atualização da tela e o método de desenho da tela contribuíram para um ganho maior de *FPS* do que ajustar para uma resolução mais baixa, mas a partir de 500 elementos na tela, percebemos um certo gargalo no sistema, onde o desempenho foi ficando muito parecido entre todas as combinações escolhidas.

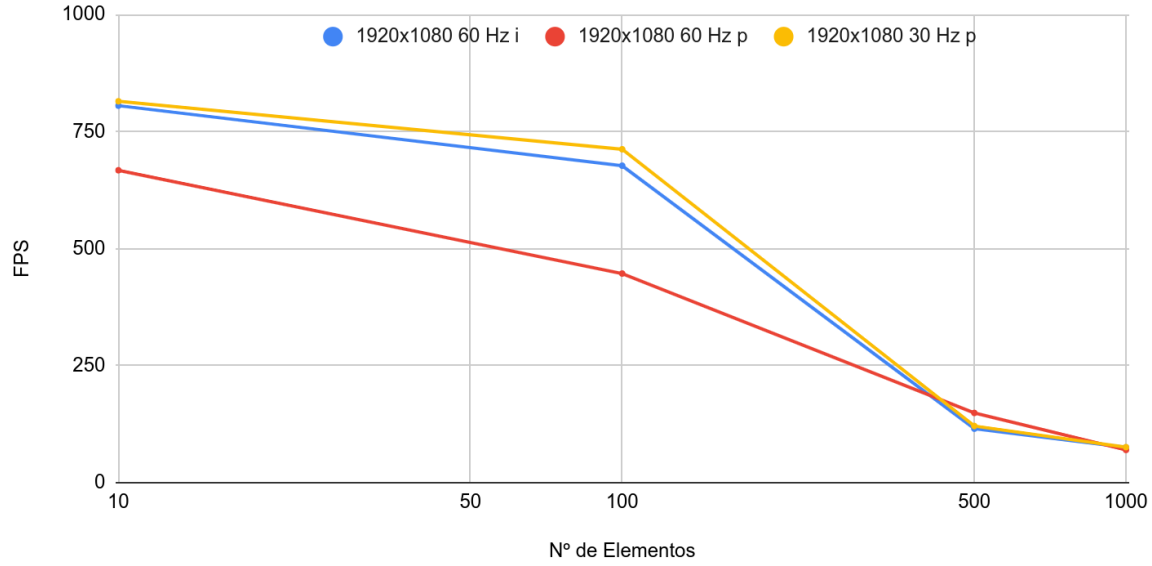

Figura 1 – FPS médio no Teste dos Quadrados em diferentes frequências de atualização e tipos de varredura

3 Infraestrutura para transformação de TV Boxes em computadores de baixo custo

Como mencionado anteriormente, para transformar a TV Box em um computador de baixo custo, neste trabalho, optamos por explorar uma estratégia em que a TV Box provê a interface necessária para o acesso à máquinas com maior poder computacional hospedadas em uma nuvem pública. O objetivo é avaliar a viabilidade de tal estratégia bem como seu desempenho, por meio de avaliação quantitativa e qualitativa.

Este estudo foi possível graças a uma parceria com as empresas *Amazon* e *DataRain*, que ofereceram, sem custo, a infraestrutura completa na nuvem AWS. A arquitetura havia sido previamente proposta por Cristiano Scandura, Arquiteto de Soluções do time Educação no Setor Público da AWS[8].

3.1 Arquitetura proposta

A arquitetura, construída e disponibilizada pela *DataRain* faz uso de diversos serviços da *Amazon Web Services* (*AWS*) e tem como finalidade o provisionamento de máquinas virtuais (*VMs*) efêmeras para serem utilizadas apenas quando necessárias e descartadas após uso.

Os professores por meio de ferramentas de *Learning Management System* (LMS) como *Moodle* ou um painel administrativo próprio podem realizar o agendamento da criação de um número variável de máquinas virtuais em uma determinada data e hora de forma que os permita criar laboratórios para aulas.

Tais agendamentos são realizados via eventos disparados no serviço *AWS Lambda*, serviço de execução de código *serverless* da *AWS*, que por sua vez dispara o provisionamento das *VMs* no horário e data desejados. Estas máquinas fazem uso do produto *EC2* (*Elastic Computing*) da *AWS*, e são da categoria *T3.xlarge*, possuindo *4vCPUs* e 16GB de *RAM* cada, em modelo de pagamento *spot*, isto é, um modelo de contratação mais econômico (0,1664 U$D por hora de uso), temporário e sem garantia de execução por períodos estendidos de tempo.

O provisionamento ocorreu utilizando uma imagem de instalação que faz uso do sistema operacional *Amazon Linux* unido a um *script* de inicialização que faz a instalação de utilitários necessários como *Python 3*, *Visual Studio Code* e *PyCharm* que são compatíveis com as necessidades do cenário de simular uma aula de introdução a programação.

[8] Blog da AWS: Educação remota através do Fire TV Stick e Laboratórios Virtuais com instâncias Amazon EC2 Spot

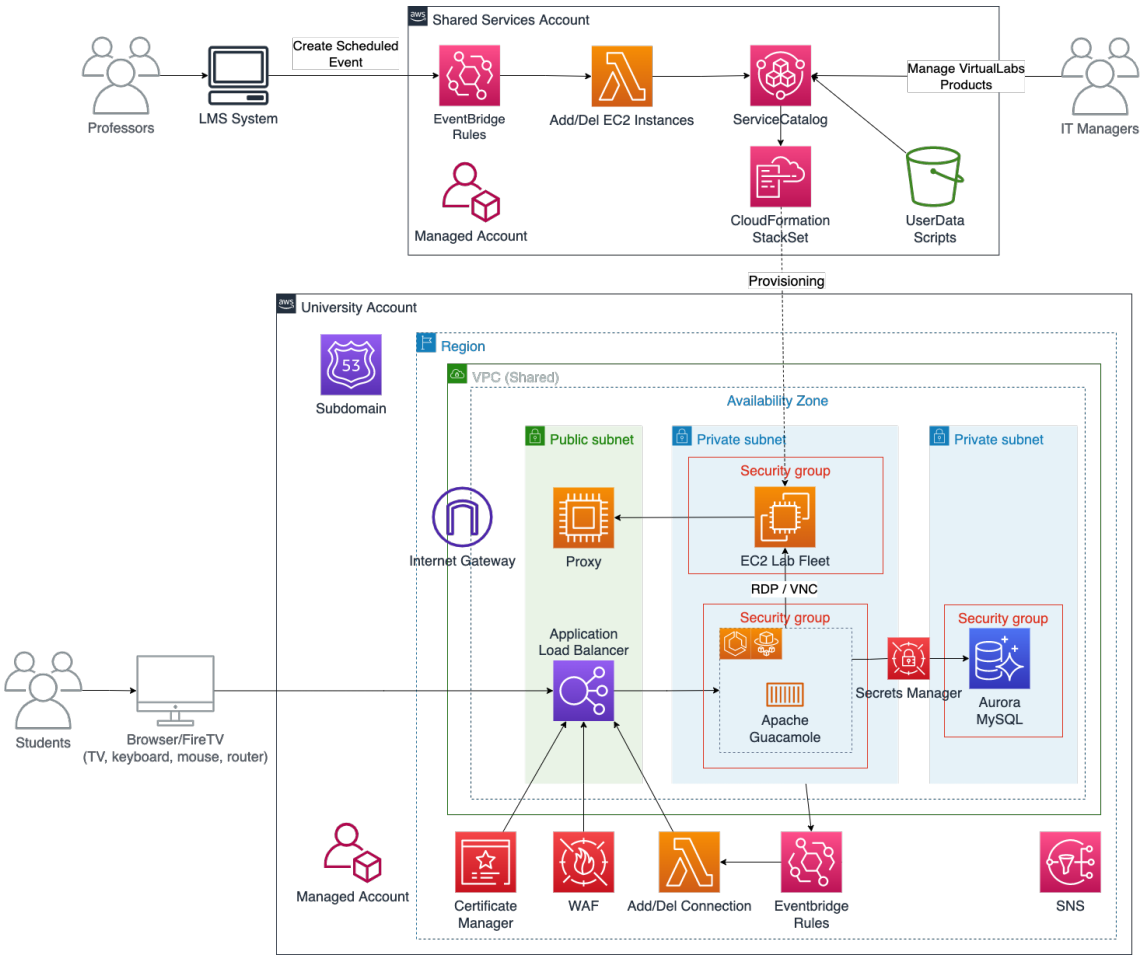

Figura 2 – Arquitetura da solução[7]

As *VMs* são criadas todas em um mesmo *Security Group* em uma sub-rede privada de forma a garantir que as máquinas possam se comunicar entre si mas estejam seguras e tenham limitações com tráfego externo da *Internet*. Nesta mesma sub-rede privada existe um grupo de *containers* executando o serviço *Apache Guacamole* [9] (serviço de *desktop* remoto) criado utilizando o *Amazon ECS* (*Elastic Container Service*). Este serviço se conecta com as *VMs* criadas na mesma *subnet* (sub-rede) para permitir o controle remoto das máquinas virtuais e também se conecta a um banco de dados *Aurora MySQL* para registro de informações referentes ao sistema interno do *Apache Guacamole*.

O *Apache Guacamole* foi o sistema de desktop remoto escolhido para este trabalho, e as motivações para sua escolha são o fato do mesmo ser gratuito, *open source* e por funcionar sem a necessidade da instalação de um cliente, executando de forma nativa em qualquer navegador com suporte ao *HTML5*, sendo o *Chromium* um deles.

O serviço de *desktop* remoto escolhido, por sua vez, é conectado a um *Application Load Balancer* que faz o balanceamento de carga das requisições de conexão externas entre o grupo de *containers* em execução de forma a impedir o sobrecarregamento do serviço devido a grande quantidade de *VMs* sendo controladas remotamente.

Por fim, este *Load Balancer* é exposto publicamente na *Internet* por meio de um *Internet Gateway*, de forma a permitir toda a infraestrutura receber requisições de conexões externas das TV Boxes ou quaisquer outros dispositivos por meio de uma *URL*.

Vale ressaltar que existem outros serviços presentes na arquitetura proposta que não foram citados devido a sua menor importância e menor impacto causado para o caso de uso utilizado e resultado final almejado.

[9] Apache Guacamole

3.2 Utilização e adaptações necessárias

Pelo fato do projeto proposto se tratar de um estudo de caso do uso de TV Boxes como computadores de baixo custo, optamos por desconsiderar a integração da infraestrutura com o *LMS* utilizado na universidade, o *Moodle*.

Por conta desta decisão, os usuários se autenticam diretamente na página de *login* do *Apache Guacamole* em vez de serem redirecionados e autorizados por um autenticador externo e houve a necessidade da criação de usuários de forma individual e manual uma vez que o processo de autenticação automatizado estava fora do escopo deste trabalho.

Outra adaptação necessária foi o uso de instâncias comuns para a realização dos testes. As instâncias utilizadas não são efêmeras, como as instâncias *spot*, e possuem valor mais alto. Os testes aconteceram no período de *Black Friday*, quando diversas empresas contratam instâncias *spot* em massa para garantir a disponibilidade de lojas virtuais, sistemas de processamento de pagamentos, entre outros. Associado a esse fato, o serviço *spot* tem uma limitação global de quantidade de máquinas, não havendo recursos disponíveis para provisionar máquinas deste tipo para este estudo no período dos testes.

É importante salientar que os autores deste projeto não tiveram envolvimento direto com a criação e manutenção desta infraestrutura, a empresa *DataRain* se encarregou de implementar a infraestrutura supracitada desenvolvida pela *AWS* e realizar os ajustes necessários e cabíveis para a janela de tempo que possuíamos.

Outro ponto a ser observado é o fato da infraestrutura proposta poder ser facilmente adaptada para outros provedores de nuvem pública como *Microsoft Azure*, *Google Cloud* e *Oracle Cloud*, caso necessário.

3.3 Metodologia de instalação nas TV Boxes

As TV Boxes passaram por um processo de preparação para que pudessem ser utilizadas como pontos de acesso aos *desktops* remotos.

Inicialmente foi instalado o sistema operacional *Armbian* em todos os dispositivos utilizados e em seguida alguns pacotes foram baixados para tornar a experiência possível, entre eles o *X.Org* que se trata de um serviço de exibição de janelas, permitindo o usuário utilizar mouse e teclado para navegação além do navegador *Chromium*.

Baseado no tutorial [10] criado pela equipe do *Smart Campus* UNICAMP realizamos algumas adaptações para instalar o sistema operacional *Armbian* nos dispositivos com o auxílio de um computador com *Windows*, um cartão de memória e um teclado. Primeiramente foram baixados e extraídos os arquivos *multitool.img* e a imagem do *Armbian* [11], além de baixar o software *Rufus*. Com os arquivos baixados, abrimos o Rufus [12], selecionamos o *Device* como sendo o cartão de memória, em seguida na opção *Boot selection* foi selecionado o arquivo *multitool.img* e iniciamos a formatação do cartão clicando em *start*.

Com a formatação concluída com a imagem do *multitool*, o cartão de memória foi removido do computador e inserido na TV Box desligada. Em seguida a TV Box foi ligada, ocorrendo assim a expansão dos arquivos do cartão de memória. Já com o teclado plugado na TV Box selecionamos a opção *Exit* no *popup* inicial. A fim de garantir uma imagem de *backup* do sistema operacional inicial da TV, selecionamos a opção *Backup flash*, ao final desse processo selecionamos a opção *Shutdown* e removemos o cartão da Box.

Novamente inserimos o cartão *SD* no computador, no entanto apenas o volume *BOOTSTRAP* é reconhecido pelo *Windows*. Assim, de forma manual montamos o volume *MULTITOOL*, abrindo o Gerenciador de Disco pressionando *Win+R* em seguida buscamos por *diskmgmt.msc* e selecionamos com *Enter*. Com a janela do gerenciamento de disco aberta identificamos o volume *MULTITOOL* sem drive associado, assim, selecionamos com o botão direito do mouse a palavra *MULTITOOL* escolhendo as opções *Change Drive Letter and Paths >Add* e definimos uma letra para o volume. Com o *drive* montado, utilizamos o gerenciador de arquivos para acessá-lo e salvarmos na pasta *images* a imagem do *Armbian* baixada. Com a imagem do sistema operacional no cartão de memória, inserimos ele novamente na TV Box desligada e ligamos a mesma novamente. A fim de instalarmos a imagem do sistema operacional selecionamos a opção *Burn Image to flash* e escolhemos a imagem salva anteriormente, após a finalização escolhemos novamente a opção *Shutdown* e removemos o cartão.

[10] Smart Campus Unicamp: Tutorial de instalação de Linux para TV Box - modelo Tx2

[11] Imagem do Armbian disponível em <https://imola.armbian.com/dl/rk322x-box/archive>

[12] Rufus

Ligamos a TV Box novamente, agora com o novo sistema funcionando, configuramos uma senha para o usuário *root*, configuramos um usuário padrão para todas as TV Boxes e configuramos a rede utilizando o Wi-fi, no entanto utilizamos o Wi-fi somente para terminarmos de configurar os dispositivos, para o teste de fato foi utilizado uma rede cabeada por oferecer menor latência. Caso o computador que servirá de apoio na instalação do SO seja um computador com *Linux*, pode ser utilizado o tutorial da equipe do *Smart Campus*.

Com o novo SO instalado, usamos como base novamente o tutorial do *Smart Campus* da UNICAMP a fim de instalar o modo *Kiosk*, que consiste em instalar uma interface gráfica no *Armbian* e também instalar o *Chromium* como navegador, além de definir uma *URL* padrão para sempre que a TV Box for ligada já inicialize com o *Chromium* aberto e em modo tela cheia com a *URL* definida. Para isso, usamos a parte do tutorial do *Smart Campus*, criamos um executável e realizamos a instalação do modo *Kiosk* com os comandos a seguir:

```
$ wget rb.gy/zjf6y
# Trocamos a URL para a fornecida pela equipe da DataRain
$ mv zjf6y kiosk-installer.sh
$ sh kiosk-installer.sh
$ sudo reboot
```

Assim, quando a TV Box é ligada, a mesma já inicializa diretamente na tela de *login* da aplicação.

Figura 3 – Tela inicial e de *login* da aplicação executando na *AWS*

Também foi criado um *script* que coleta métricas de performance na TV Box como uso de CPU, uso de memória, uso de disco, processos em execução e uso da rede (pacotes enviados/recebidos e erros de *RX*/*TX*). Esse *script* foi instalado através dos seguintes comandos:

```
$ wget rb.gy/hwhegw
$ mv hwhegw install.sh
$ sh install.sh
```

A fim de reduzir a lentidão das TV Boxes, substituímos a primeira linha da crontab por * * * * * *nice -19 sh metricas.sh* permitindo que o processo de monitoramento executasse em menor prioridade, liberando assim recursos para outras aplicações do sistema. Além disso, atualizamos o arquivo de *autostart* utilizando *sudo nano /home/kiosk/.config/openbox/autostart* deixando o comando do *Chromium* fora do *while* existente para que não ficasse abrindo repetidas vezes. Ademais, mudamos a resolução da TV Box, colocando o comando *xrandr –output HDMI-1 –mode 1280x720* no arquivo */home/kiosk/.config/openbox/autostart* antes do comando do *Chromium* para melhoria do desempenho.

Por fim, configuramos o teclado para PT-BR através do seguinte *script*:

```
$ sudo dpkg-reconfigure keyboard-configuration
```

```
# Confirmar padrao de 105 teclas
# Quando aparecer um menu com diversas opcoes de English, clicar em Other
# Encontrar a opcao Portuguese
# Prosseguir com as opcoes padrao ate o fim do utilitario
$ sudo reboot
```

Após a configuração de todas as TV Boxes, numeramos cada uma para podermos associar suas métricas e a avaliação de usabilidade, chegando assim em dezenove TV Boxes numeradas e espalhadas de forma aleatória no dia dos testes de usabilidade.

Figura 4 – TV Boxes ligadas e já *logadas* no sistema da *AWS* durante os testes de usabilidade

3.4 Metodologia de avaliação

3.4.1 Testes

Os testes foram conduzidos com 16 voluntários divididos em dois grupos. Um grupo realizou uma atividade básica de programação em computadores do laboratório de ensino do Instituto de Computação, enquanto o outro grupo fez a mesma atividade usando a TV Box. Após 30 minutos, trocamos os grupos para uma segunda atividade, onde os voluntários submeteram o código feito na primeira etapa.

As atividades foram criadas para serem concluídas em 30 minutos por estudantes iniciantes em programação, envolvendo conceitos como manipulação de cadeias de caracteres e busca em vetores.

Os voluntários acessaram uma disciplina no *Google Classroom*, onde encontraram duas atividades de programação. Eles leram os enunciados, escolheram uma *IDE* para escrever o código e o submeteram na plataforma. Esse processo foi projetado para simular uma aula prática da disciplina de MC102 - Algoritmos e Programação de Computadores ministrada pelo Instituto de Computação (IC) da UNICAMP.

3.4.2 Critérios de comparação

Para conseguirmos comparar a TV Box com computadores normais, utilizamos duas abordagens: coleta de métricas de uso e desempenho da TV Box e uma análise de usabilidade pelos usuários.

Para analisar a experiência que o voluntário teve durante as atividades criamos um formulário que deveria ser preenchido pelo voluntário em ambas as atividades, ou seja, tanto quando utilizou a TV Box, quanto quando utilizou o computador do IC, para que pudéssemos comparar as respostas. Como, no geral, testes de usabilidade são mais qualitativos é complicado conseguir indicar "o quão grande" é o problema de usabilidade, em uma escala numérica, decidimos utilizar uma escala numérica de usabilidade: a metodologia *System Usability Scale* (*SUS*) [13]. Esta escala avalia, basicamente, 3 critérios:

- Efetividade (os usuários conseguem completar seus objetivos?)
- Eficiência (quanto esforço e recursos são necessários para isso?)
- Satisfação (a experiência foi satisfatória?)

Em teoria, a Efetividade e a Eficiência são medidas de acordo com o que o avaliador percebe das ações do usuário. No entanto, como no nosso caso não havia um avaliador para cada voluntário, inserimos duas perguntas no formulário perguntando se este conseguiu completar a tarefa e se teve muita dificuldade na sua execução. Já a satisfação é medida a partir do formulário que o usuário preenche depois de ter usado o protótipo. Adicionamos uma pergunta (pergunta 2) às da metodologia original para captar um aspecto que precisávamos. Os indicadores funcionam da seguinte forma:

- Efetividade:
  - 0 = não conseguiu realizar a tarefa
  - 100 = conseguiu realizar a tarefa
- Eficiência:
  - 0 = realizou a tarefa com muita dificuldade
  - 50 = realizou a tarefa com pouca dificuldade
  - 100 = realizou a tarefa com facilidade
- Satisfação: foram realizadas 11 perguntas com notas de 1 a 5, de "discordo completamente" a "concordo completamente":
  - Eu acho que gostaria de usar esse sistema com frequência.
  - Eu achei o sistema lento.
  - Eu acho o sistema desnecessariamente complexo.
  - Eu achei o sistema fácil de usar.
  - Eu acho que precisaria de ajuda de uma pessoa com conhecimentos técnicos para usar o sistema.
  - Eu acho que as várias funções do sistema estão muito bem integradas.
  - Eu acho que o sistema apresenta muita inconsistência.
  - Eu imagino que as pessoas aprenderão como usar esse sistema rapidamente.
  - Eu achei o sistema atrapalhado de usar.
  - Eu me senti confiante ao usar o sistema.
  - Eu precisei aprender várias coisas novas antes de conseguir usar o sistema.

A partir dessas informações, podemos então calcular o chamado *SUS score*, o qual indica uma medida global da satisfação do sistema e subescalas de usabilidade e capacidade de aprendizagem [14] Para calcular o *SUS Score*, como adicionamos uma pergunta (2), fazemos os seguintes passos:

[13] O que é o SUS?

[14] *Measuring Usability with the System Usability Scale (SUS).*

- Para a resposta 1 e as respostas pares do formulário de satisfação (1, 2, 4, 6, 8, 10), subtraia 1 da pontuação que o usuário respondeu.

- Para as respostas ímpares (exceto a 1) do formulário de satisfação (3, 5, 7, 9, 11), subtraia a resposta de 5.

- Some todos os valores das dez perguntas, e multiplique por 2,273

Ao fim essa será a pontuação final, que pode ir de 0 a 100, sendo que a usabilidade do sistema, de acordo com a escala do *SUS Score*, é interpretada como:

- 0 a 51 é inaceitável, horrível;

- de 51 a 68, ruim;

- 68 é ok;

- 68 a 80,3 é bom;

- e maior que 80,3 é excelente.

Também coletamos a informação de qual atividade foi realizada no dispositivo, a fim de entender se a atividade realizada afeta os dados.

Além dessas 11 perguntas, incluímos uma sessão especial do formulário para colher *feedbacks* mais qualitativos sobre o uso da TV Box em específico. Ou seja, para a atividade em que o voluntário utilizou a TV Box, além das perguntas de múltipla escolha do *SUS*, ele respondeu mais 5 perguntas sobre seu uso, a fim de detalhar sua experiência. As perguntas foram:

- Qual o número da TV Box em que você realizou a atividade?

  - Coletamos a informação para comparar com as análises a partir da coleta de métricas quantitativas. Além disso, em momento nenhum coletamos a identidade do voluntário.

- Fez falta ter outros recursos na TV Box? Se sim, quais?

  - Saber se a avaliação foi influenciada pela falta de algum *software*.

- Qual dificuldade você teve com a interface do sistema (sistema operacional, navegador, editor e testador de código ou plataforma de submissão, por exemplo)?

- Você teve algum problema de conexão com a internet?

- Teve muito *delay* ou outros problemas de lentidão sem ser a internet? (Exemplo: você clicou e demorou para selecionar)

  - Entender se houve algum problema de lentidão de interface

Já, para analisar o desempenho da TV Box, fizemos um *script* que coleta informações como uso da CPU, uso da memória, uso de disco, uso de rede e *top* dez processos em execução. Essas informações são importantes para podermos identificar possíveis pontos que mostram um bom funcionamento ou um mau funcionamento dos dispositivos testados.

3.5 Análises comparativas: Computador *vs* TV Box para atividades de programação

3.5.1 Análise de usabilidade

Analisando inicialmente o *SUS score* no quesito Efetividade, a fim de identificar se o usuário conseguiu concluir seus objetivos, observamos que utilizando o computadores do Instituto de Computação (IC) todos os alunos conseguiram concluir ambas as atividades propostas, enquanto ¼ desses alunos não conseguiram concluir a atividade usando a TV Box, sendo importante salientar que todos os casos foram na atividade 2. Considerando que as atividades eram relativamente simples, planejadas para serem concluídas pelos alunos em menos de uma hora, e levando em conta que os mesmos alunos que não conseguiram terminar usando a TV Box conseguiram concluir as atividades nos computadores do instituto, torna-se crítico o fato de que 25% dos alunos não puderam finalizar a tarefa, o que nos leva a concluir que a estratégia de usar uma TV Box, com as configurações de *hardware* e *software* utilizadas neste trabalho, conectada à uma *VM* na nuvem, é ineficaz para realização de atividades envolvendo programação. No entanto, ainda há espaço para se investigar se outras configurações possibilitariam melhor desempenho.

Ao avaliar o segundo critério, Eficiência, que aborda a facilidade percebida pelos usuários ao realizar suas tarefas, foi observado que 87,5% dos usuários da TV Box enfrentaram dificuldades na utilização da interface, enquanto 100% relataram problemas de lentidão no sistema, dos quais 93,8% ocorreram repetidamente Figura 5. Os principais problemas destacados incluem dificuldades com o uso do mouse, levando a maioria dos usuários a preferir a navegação pelo dispositivo utilizando o teclado, além da lentidão excessiva do sistema. Em contrapartida, não foram registradas reclamações sobre o uso dos Computadores do IC, evidenciando uma eficiência excelente.

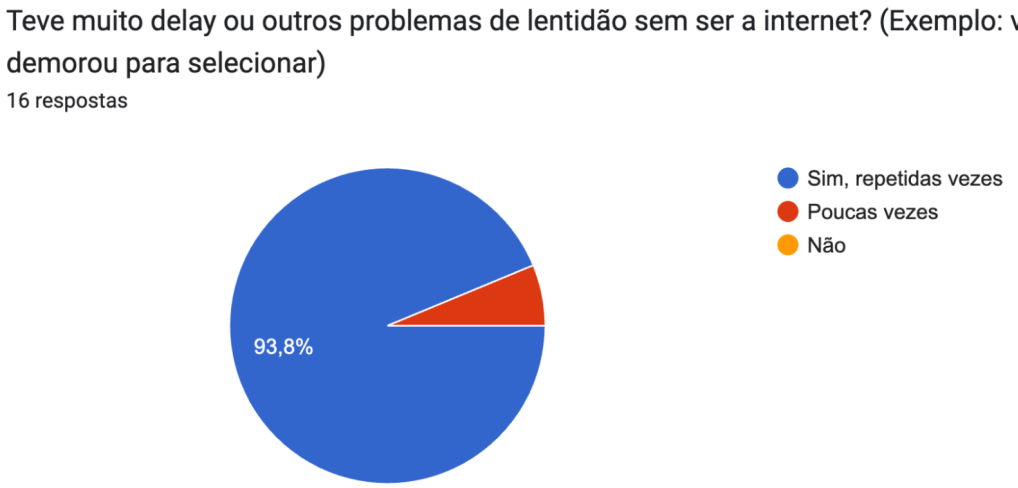

Figura 5 – Distribuição de usuários que enfrentaram lentidão usando a TV Box

Por último, ao analisar a Satisfação na escala de *SUS score*, calculamos a avaliação para cada uso das máquinas, determinando a média das notas atribuídas a cada aparelho, o que resultou nos seguintes resultados:

- Média da nota de satisfação com o Computador do IC: 93,2 (nota excelente)
- Média da nota de satisfação com a TV Box: 60,2 (nota ruim)

Os pontos que se destacaram negativamente foi que nenhuma pessoa gostaria de usar a TV Box com frequência e todas consideraram o sistema lento concordando quase ou completamente com essa afirmação (Figura 6). Ao mesmo tempo as perguntas relacionadas à complexidade (Figura 8) do sistema e conhecimentos técnicos prévios para utilização (Figuras 9 e 10), em sua maioria, apontam que isso não era um problema no uso da TV Box, pois as notas acompanharam a distribuição similar em relação às respostas referente aos Computadores do IC. Isso sugere uma considerável familiaridade dos usuários com a interface, indicando que sua

usabilidade era percebida como simples. Contudo, a persistente lentidão (Figura 7) comprometeu uma boa experiência para os alunos na elaboração das atividades quando usado o sistema na TV Box. Importante ressaltar que o problema de lentidão não foi resultado do problema de conexão com a *Internet* tendo em vista que 93,8% não teve nenhum problema com a *Internet* durante o uso (Figura 11).

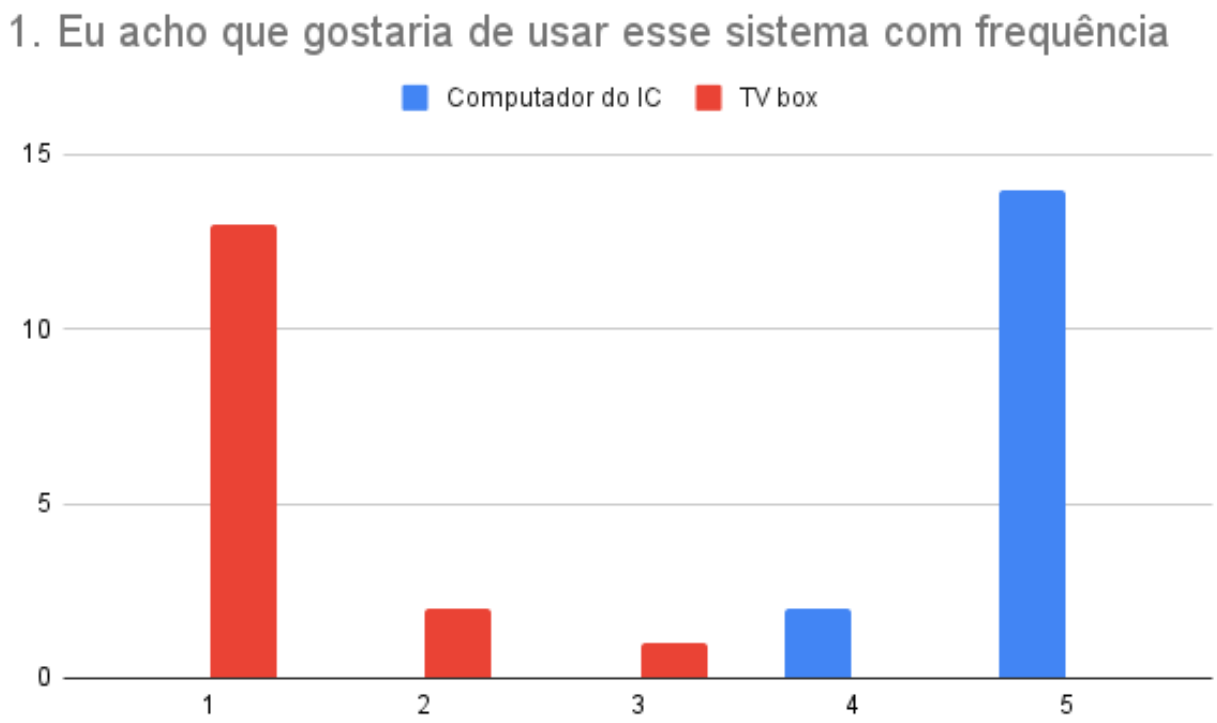

Figura 6 – Distribuição das respostas obtidas na primeira pergunta

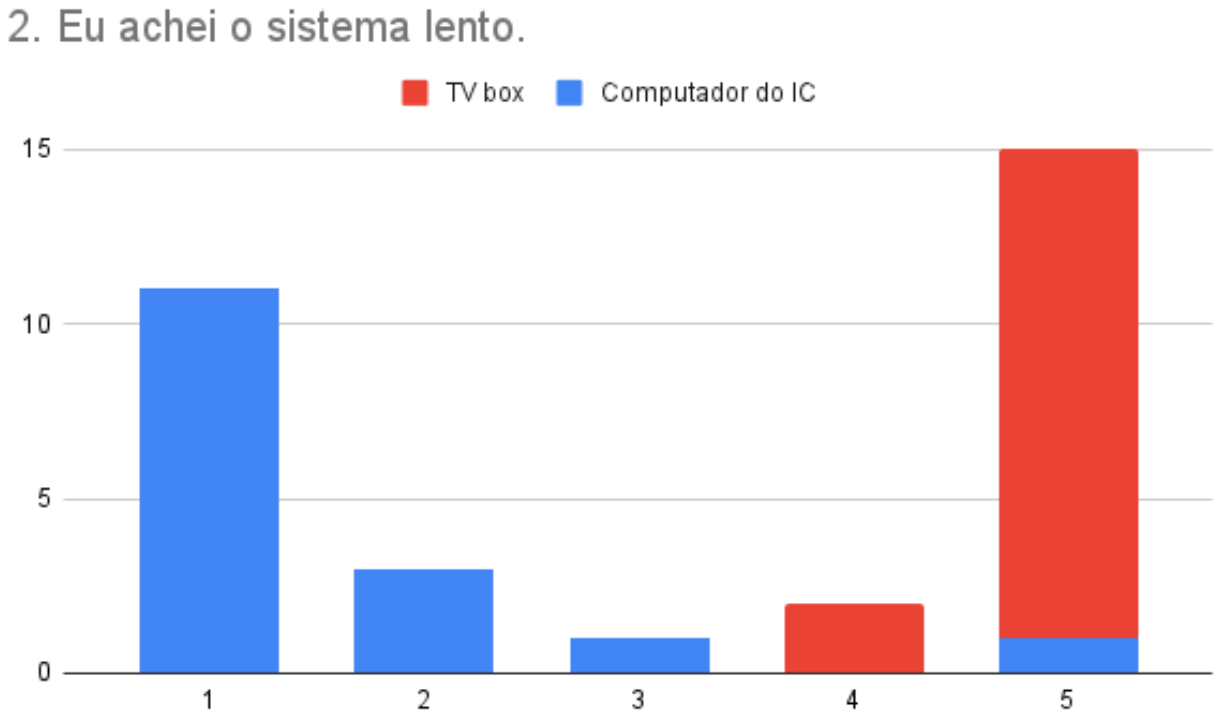

Figura 7 – Distribuição das respostas obtidas na segunda pergunta

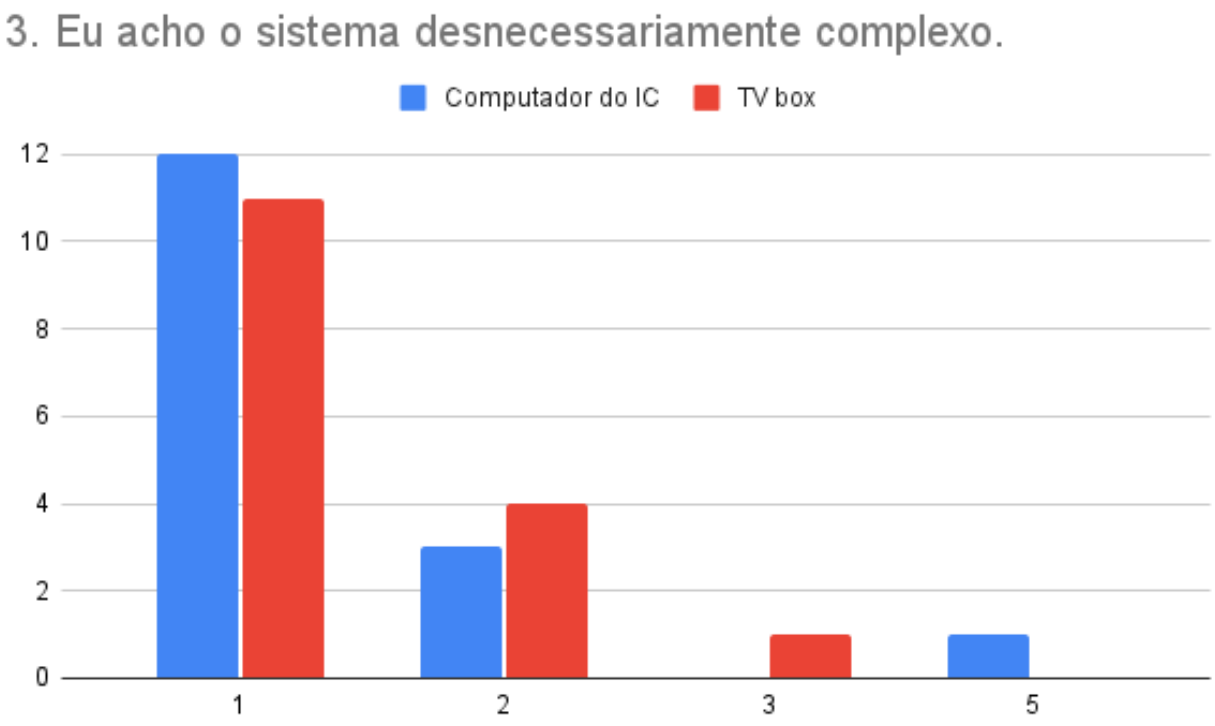

Figura 8 – Distribuição das respostas obtidas na terceira pergunta

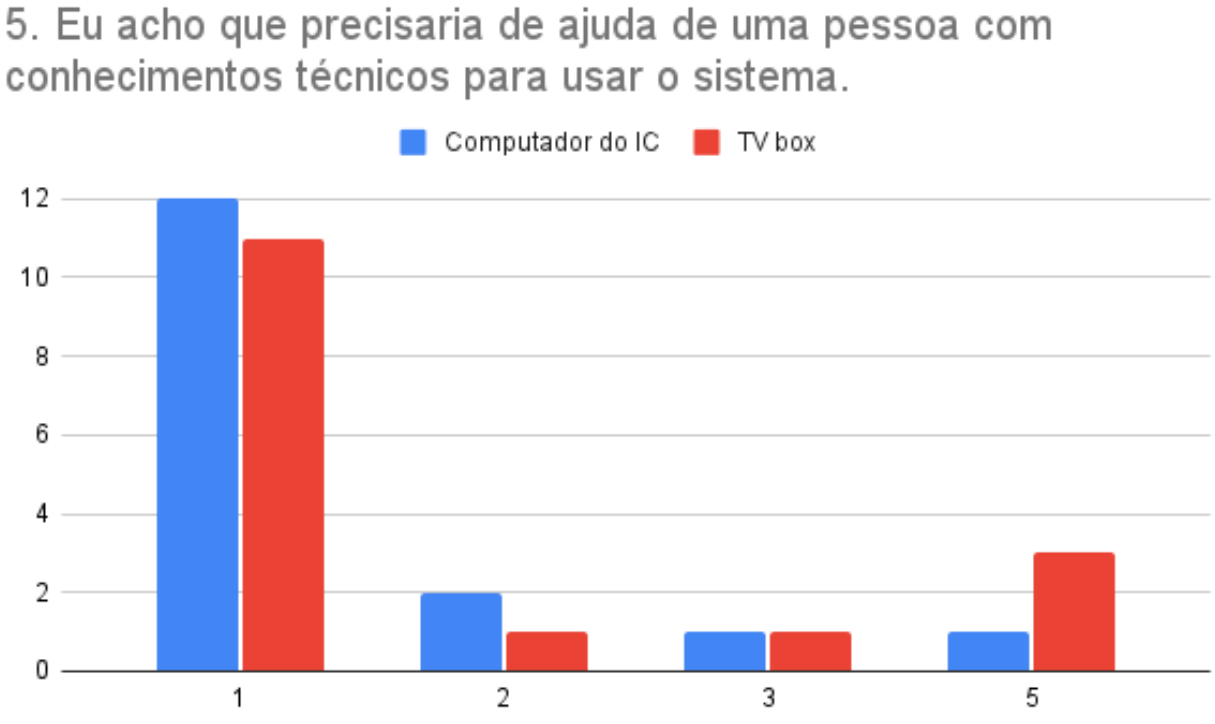

Figura 9 – Distribuição das respostas obtidas na quinta pergunta

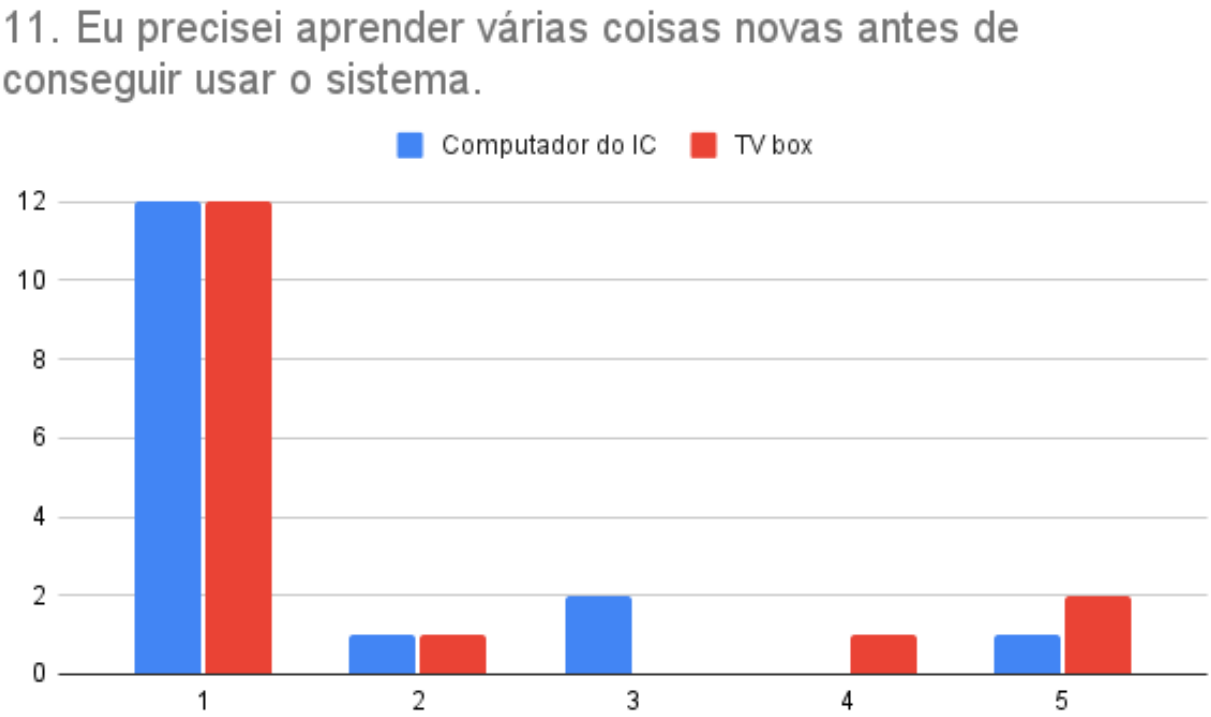

Figura 10 – Distribuição das respostas obtidas na décima primeira pergunta

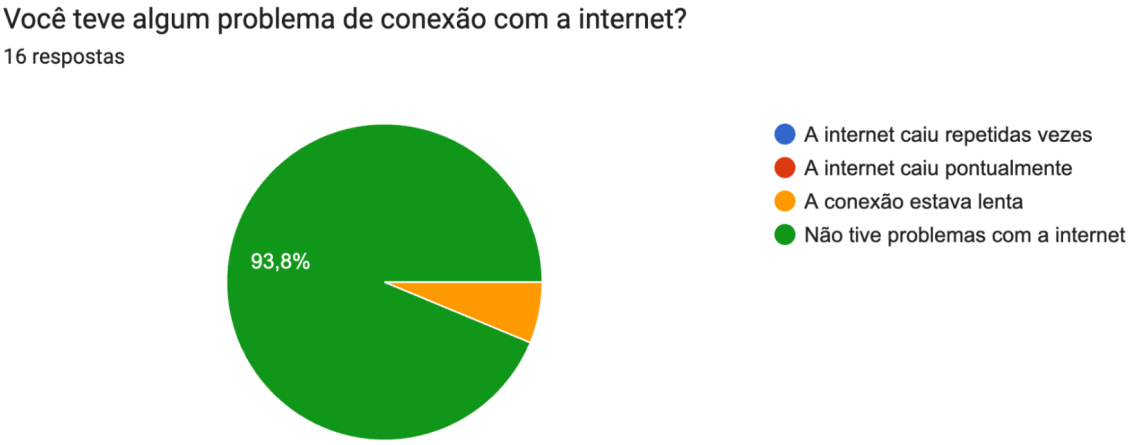

Figura 11 – Quantidade de usuários que tiveram problemas de conexão

3.5.2 Análise de desempenho computacional

As métricas coletadas a cada minuto pelo *script* de monitoramento anteriormente citado foram consolidadas, analisadas e comparadas. A consolidação foi realizada por um *script Python* [15] feito para este fim, que recorta os dados pertinentes do arquivo de *log* criado pelo *script* de monitoramento e gera um arquivo CSV como saída. Utilizando os diversos arquivos CSV, foi gerada uma planilha com todos os dados consolidados para todas as TV Boxes utilizadas.

Ao observar os gráficos da Figura 12, é possível notar que em todas as TV Boxes, incluindo as mais lentas e problemáticas no teste com usuários, como as de número 1 e número 19, o uso de CPU e memória *RAM* esteve o tempo todo abaixo de 10% para todos os dispositivos utilizados nos testes. O uso de disco também esteve baixo durante o teste, menor do que 15% para todos os *mountpoints*.

[15] Script Python disponível em <https://gist.github.com/matheusot/305310e4a38333c9b3d94c5a2744aaad>

O uso baixo de CPU e *RAM* para um dispositivo que dispõe de poucos recursos gerou surpresa, pois se esperava que o navegador *Chromium* em conjunto com o sistema operacional e sistema de janelas *X.Org* consumissem os recursos disponíveis em maior grau. Uma hipótese é que o sistema operacional *Armbian* possivelmente não seja capaz de utilizar os recursos da TV Box em sua totalidade.

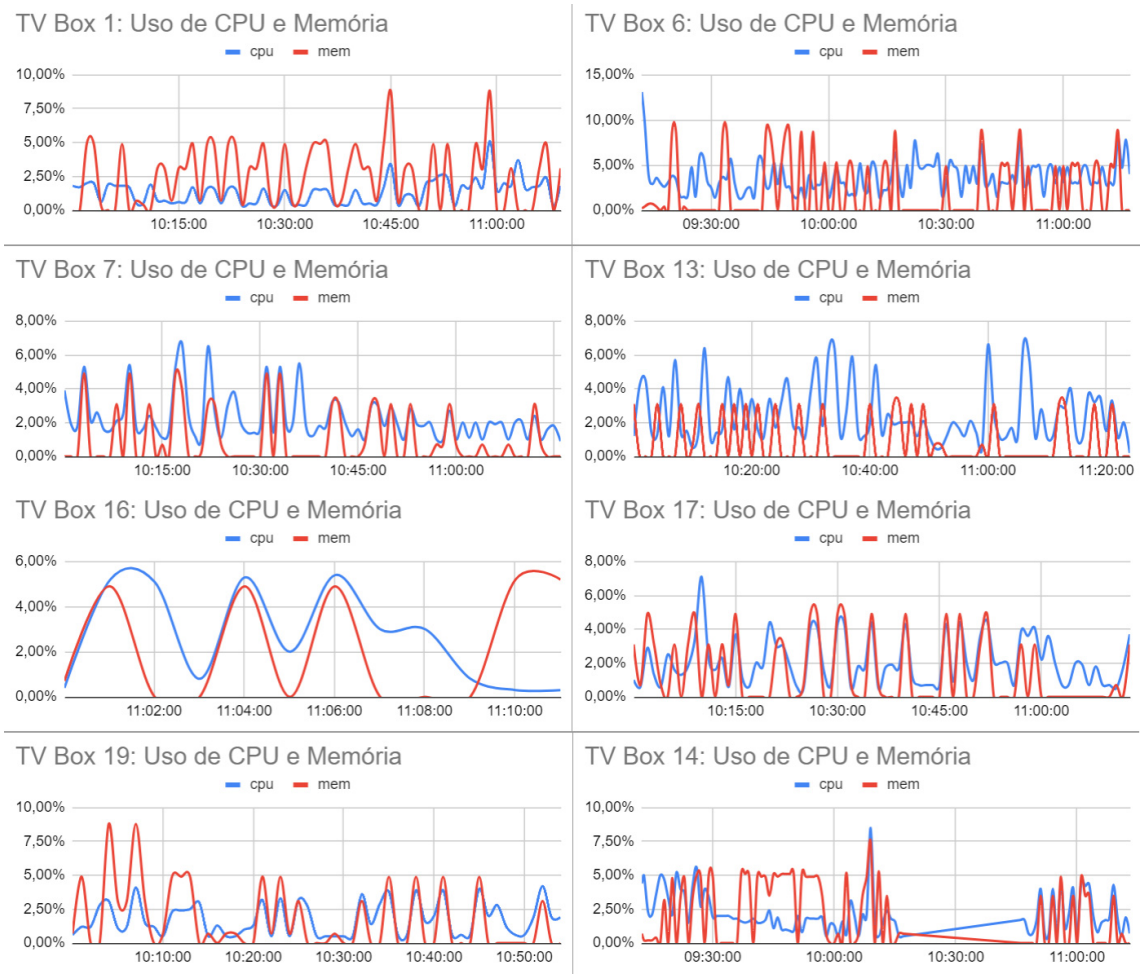

Figura 12 – Gráficos de uso de CPU e memória para as TV Boxes cujo as métricas foram coletadas.

Referente ao uso de rede, a quantidade de dados enviados (*TX*) e recebidos (*RX*) também foi analisada de forma acumulada durante todo o período do teste. As TV Boxes durante o teste utilizaram rede cabeada *Ethernet* (interface *end0*), rede esta que foi prontamente disponibilizada e montada pela equipe de funcionários do Instituto de Computação da UNICAMP de forma exclusiva para o teste.

Os dados (Figura 13) mostram que o uso de rede foi primariamente para o recebimento de dados, e que a banda total utilizada não excedeu o valor de 250 milhões de *bytes* (250 *megabytes*) durante todo o teste.

Outra análise que pode ser obtida é que no caso mais extremo, houve um crescimento de banda de 8,3 milhões de *bytes* para 25,3 milhões de *bytes* em um período de 13 minutos na TV Box de número 1. Isso permite inferir que a velocidade mínima de download necessária para que o vídeo seja exibido com êxito é cerca de 173,6 *kbps* (*kilobits* por segundo).

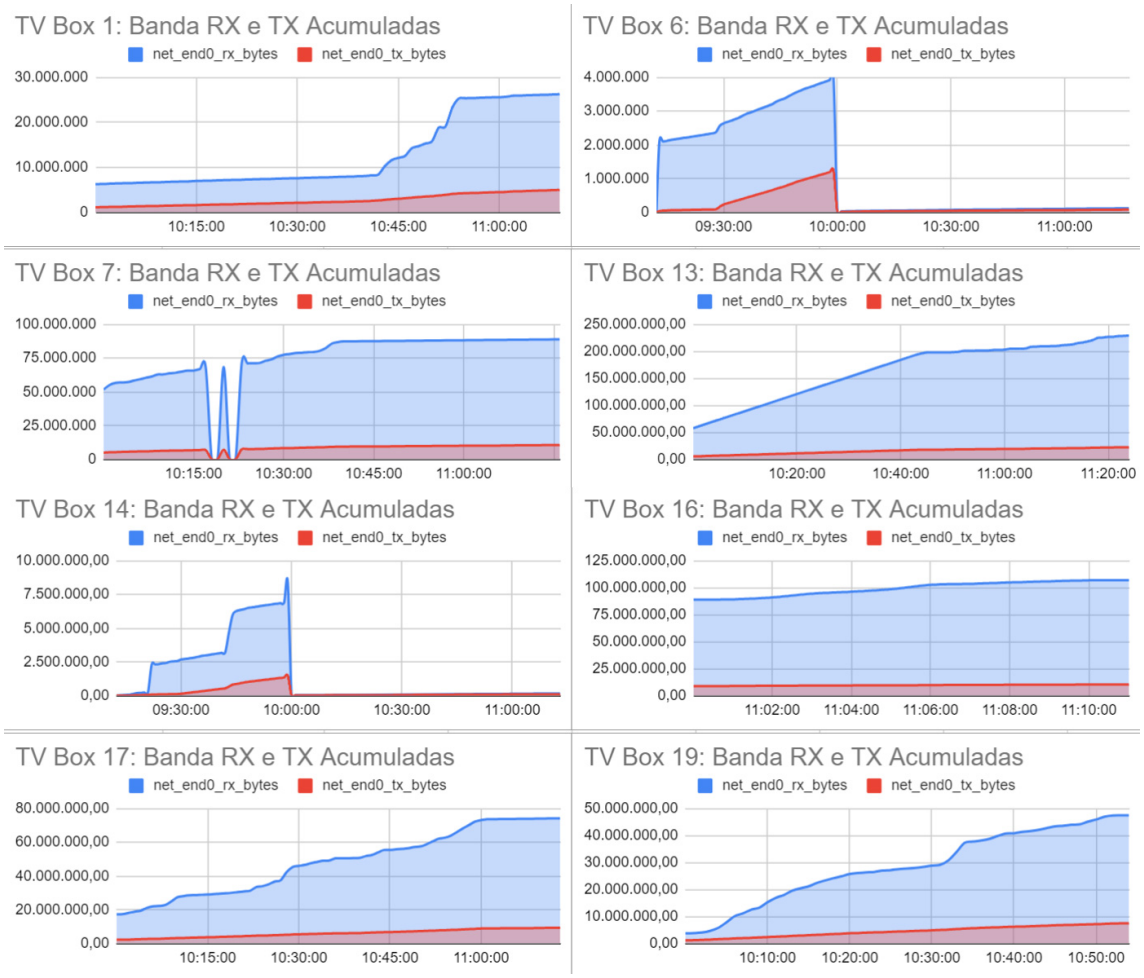

Figura 13 – Gráficos da banda RX (net_end0_rx_bytes) e TX (net_end0_tx_bytes) acumuladas na interface end0 em bytes.

Infelizmente, durante o teste foi possível notar uma grande diferença entre o desempenho de diferentes TV Boxes. Algumas tiveram um desempenho mais satisfatório para o procedimento, enquanto uma minoria demonstrou um péssimo desempenho (número 1 e número 19). Com os dados coletados e análises realizadas, neste momento, não podemos obter conclusões sobre as razões desta lentidão em apenas alguns dispositivos, uma vez que o uso de recursos foi compatível entre todo o grupo.

4 Segurança

Para a verificação de segurança referente ao tráfego de pacotes na rede e a análise de *logs* do sistema *Android* nativo na TV Box, foi realizada uma investigação técnica. Utilizou-se duas ferramentas principais para realizar essa investigação: *Wireshark* para o monitoramento e captura do tráfego de rede e o *ADB* (*Android Debug Bridge*), para analisar *logs* do sistema no dispositivo com sistema operacional *Android* nativo.

Após as investigações técnicas foi feito um processo de pesquisa para tentar bloquear o botão *reset* da TV Box quando o sistema instalado é um sistema operacional *Android*. O objetivo deste bloqueio é evitar que o uma TV Box descaracterizada seja revertida para sua função original com *software* ilegal.

Na primeira etapa da investigação foi utilizado o *Wireshark*, uma ferramenta de análise de tráfego de rede. Essa investigação foi feita para descobrir se existiam alguns pacotes com informações confidenciais que estariam sendo enviados para endereços *IPs* (*Internet Protocol*) desconhecidos ou com um grau de segurança menor, como por exemplo, informações enviadas por protocolos menos seguros como *HTTP* (*Hypertext Transfer Protocol*) dentre outros.

A metodologia utilizada para a investigação com *Wireshark* foi: criar uma rede separada na rede interna - alguns roteadores permitem a criação de uma rede separada para Convidado, no caso o roteador utilizado para a criação da rede foi um *TP-Link* modelo *Archer C20* - onde somente a TV Box e o computador com

Wireshark instalado usariam essa rede, isso garantiria que se houvesse algo suspeito seria mais fácil de rastrear. Com a rede criada e com os dispositivos conectados nela, basta capturar os pacotes dessa rede com o *Wireshark* enquanto roda as aplicações da TV Box. Nesta investigação foram coletados aproximadamente 1000 pacotes, tanto com a TV Box ociosa quanto com as aplicações rodando nela. Após a captura dos pacotes gerou-se com um arquivo *.pcapng* (extensão do *Wireshark*) e nesse arquivo foram aplicados filtros [16]:

- Filtro para ver quais pacotes estavam saindo da TV Box:
  - *ip.src* == <*IP* atribuído à TV Box na rede de teste>

  Com esse filtro foi possível descobrir que os pacotes que saem da TV Box usam protocolo *SSDP* (*Simple Service Discovery Protocol*), que é um protocolo de descoberta de serviços usado em redes *UPnP* (*Universal Plug and Play*) e usa *UDP* (*User Datagram Protocol*) como protocolo de transporte. Além disso, também foi possível descobrir o *IP 239.255.255.250*, que indica o endereço e *multicast* usado pelo *SSDP*. Os resultados são os mesmo tanto para os pacotes capturados com a TV Box em estado ocioso quanto para os pacotes capturados com a TV Box rodando aplicações.
- Filtro para verificar se existia algum pacote sendo transferido com algum protocolo menos seguro como por exemplo, *HTTP*.
  - *ip.src* == <*IP* atribuído à TV Box na rede de teste> *and http*

  Não havia nenhum pacote sendo enviado com protocolo *HTTP*.
- Outros filtros para analisar pacotes com protocolos menos seguros também foram investigados, como por exemplo: protocolos *DNS* (*Domain Name System*), *Telnet* (*Terminal Network*), *FTP* (*File Transfer Protocol*), *SMTP* (*Simple Mail Transfer Protocol*), *POP3* (*Post Office Protocol version 3*) e *IMAP* (*Internet Message Access Protocol*). Estes protocolos podem ser enviados sem proteção por criptografia, porém nenhum deles foi identificado na investigação.

Na segunda etapa da investigação, foi utilizado o *ADB* [17], uma interface por linha de comando que facilita a comunicação com dispositivos *Android*, para extrair e analisar os *logs* do sistema. Essa abordagem foi escolhida para analisar possíveis atividades que poderiam estar sendo executadas em segundo plano, eventos do sistema e possíveis ameaças ou comportamentos suspeitos no dispositivo com SO *Android* nativo.

A metodologia utilizada para a investigação com o *ADB* foi: verificar o endereço *IP* da TV Box e conectar ao dispositivo pelo *ADB*. Com o dispositivo conectado, foi possível salvar o arquivo de *log* do sistema com auxílio do *ADB* e da ferramenta *logcat* [18], que é uma ferramenta de linha de comando que mostra mensagens do sistema e é útil para acompanhar e entender o que está acontecendo no aplicativo. Para salvar os logs em um arquivo de texto basta usar os seguintes comandos:

- Para conectar o dispositivo ao *ADB*:
  - *adb connect <IP atribuído à TV Box na rede de teste>:5555*
- Para salvar o *logcat* em um arquivo de texto:
  - *adb logcat > logcat_tv_box_ociosa.txt* (salva o arquivo de log com a TV Box em estado ocioso)
  - *adb logcat > logcat_tv_box_rodando_aplicacao.txt* (salva o arquivo de log com a TV Box rodando alguma aplicação)

Com os *logs* do sistema - tanto em repouso, quanto com aplicações sendo executadas nele - salvos em um arquivo de texto, basta usar filtros do terminal do *Linux* para verificar possíveis erros e anomalias que estavam acontecendo no sistema. Os filtros utilizados foram:

[16] *Wireshark CaptureFilters,* disponível em <https://wiki.wireshark.org/CaptureFilters>

[17] *Android Debug Bridge (adb),* disponível em <https://developer.android.com/tools/adb?hl=pt-BR>

[18] Ferramenta de linha de comando Logcat, disponível em <https://developer.android.com/studio/command-line/logcat?hl=pt-BR>

- Filtro para verificar possíveis erros no sistema com a TV Box ociosa:

  – *cat logcat_tv_box_ociosa.txt | grep " E cut -d "E-f2*

  Filtro que mostra todos os erros que aconteceram no sistema, isso é feito fazendo uma busca (*grep*) pelo caractere "E" (Erro) que indica uma *flag* de erro e é a terceira *flag* na prioridade das *flags*. Analisando os erros, pode-se verificar que a grande maioria dos erros ocorrem ao tentar encontrar algum arquivo do sistema e esse arquivo não ser encontrado. Outros acontecem por algum serviço não conseguir obter informações sobre controladores específicos, como por exemplo o erro *BatteryStatsService: no controller energy info supplied*, que indica que o serviço *BatteryStats* não recebeu informações sobre o consumo de energia de algum controlador. Outros filtros com *flags* "W" (Aviso) e "F" (Fatal), que também são *flags* com alta prioridade do *logcat* também não entregam problemas que indiquem que tem algum software malicioso instalado no sistema nem nenhuma anomalia que indicasse algo errado no sistema.

- O filtro para verificar possíveis erros no sistema com a TV Box rodando aplicações foram iguais ao da TV Box em estado ocioso. Como resultado, não foi encontrado nenhum software malicioso instalado no sistema com a TV Box rodando aplicações nem anomalias.

Uma terceira etapa da investigação consistiu em tentar bloquear o botão *reset* presente na placa para que a TV Box não fosse formatada para configuração de fábrica. Essa etapa visa garantir que a TV Box não volte a executar aplicações que pirateiem serviços de *streaming*. Com *Linux* instalado na TV Box o botão *reset* não funciona, então o problema acontece quando se tem o sistema *Android* instalado na TV Box, nesse sistema, ao apertar o botão *reset*, abre-se a tela de recuperação do sistema, onde nos permite restaurar a TV Box para configuração de fábrica.

Existem duas maneiras de remover um botão de uma placa, uma das formas é com uma estação (soprador) e outra com um ferro de solda. Para ambas existem tutoriais no *Youtube* explicando como remover o botão, os procedimentos são listados abaixo:

Para remover com uma estação deve-se seguir o seguinte procedimento [19]:

- Proteger a parte de baixo da placa com fita reflexiva, posicionando-a embaixo do botão a ser retirado e um pouco a frente (saindo) da placa.
- Adicione pasta de solda no botão.
- Colocar a estação (soprador) na temperatura suportada pela placa, exemplo, 410ºC para placas da *Samsung*, e usar bico médio na estação.
- Colocar a estação embaixo do botão fazendo movimentos circulares.
- Verificar aos poucos, com uma pinça, se o botão está saindo.
- Limpar o lugar onde foi removido o botão.

Para remover com um ferro de solda, basta seguir o procedimento [20]:

- Procurar os *Pads* de solda do botão.
- Adicione bastante solda nos *Pads* encontrados.
- Adicionar fluxo de solda.
- Adicione bastante solda nova nos *Pads*, para ao adicionar solda usar o ferro de solda fazendo movimentos frente e trás sem forçar o botão.
- Posicionar o ferro de solda na parte contrária para onde desejamos remover o botão - por exemplo se quisermos remover o botão da esquerda para a direita então o ferro de solda é colocado na esquerda do botão - adicionar mais estanho, mexer o ferro de solda em movimento frente trás sem forçar o botão.

[19] Removendo botão com estação (soprador), disponível em <https://www.youtube.com/watch?v=P1hH00sl-Ko>

[20] Removendo botão com ferro de solda, disponível em <https://www.youtube.com/watch?v=bB0ldodsNC8>

- Forçar um pouco o botão e ele será removido.
- Limpar o fluxo que sobrou com malha de soldador e mais fluxo.

5 Desafios futuros e conclusão

A tentativa de utilizar aparelhos de TV Box como substitutos de computadores de baixo custo resultou em experiências insatisfatórias e limitações significativas. Embora esses dispositivos tenham sido originalmente projetados para *streaming* de conteúdo multimídia na televisão, sua adaptação para funções mais complexas, como navegação na *Internet*, uso de *IDEs* e terminal, revelou-se inadequada dentro do contexto da nossa metodologia específica, sendo necessário portanto a busca de novas abordagens a fim de tornar o dispositivo adequado ao uso pedagógico proposto.

Os principais pontos de dificuldade analisados pelos voluntários foram a usabilidade do mouse e a lentidão da plataforma, lentidão esta que dificulta a execução de tarefas simples, como abrir um navegador ou um editor de texto. Outro ponto que chama a atenção foi a variabilidade de desempenho entre dispositivos.

As investigações referentes à segurança do dispositivo foram satisfatórias no sentido de não encontrar nada sendo enviado para *hosts* suspeitos ou mesmo aplicações rodando em segundo plano, que poderiam comprometer a segurança dos dados do usuário da TV Box.

A principal dificuldade em segurança foi encontrar uma solução para o botão *reset* quando se tem o sistema operacional *Android* instalado na TV Box, pois, com o sistema operacional *Linux* o botão *reset* fica inativo. Ao tentar instalar um novo sistema operacional *Android*, diferente do que veio instalado na TV Box, alguns erros ocorreram, erros esses referentes à incompatibilidade de versões de *ROMs* (*Ready-Only Memory*), encontradas na *Internet*, com o chip da placa, isso limitou o estudo para saber se em outros sistemas *Android* o botão *reset* estaria ativo ou inativo.

As pesquisas referentes ao botão *reset* da placa foram pouco conclusivas e a solução encontrada foi removê-lo, portanto, a solução não foi a melhor possível, pois se o procedimento for feito errado, isso pode inviabilizar o uso da placa e consequentemente da TV Box.

Desta forma, o projeto ainda demanda melhorias e otimizações de forma a se tornar uma solução viável aos usuários. A arquitetura na nuvem por trás desses dispositivos também demanda uma reavaliação. A lentidão da plataforma, dificultando tarefas simples como abrir um navegador ou editor de texto, sugere que as características da arquitetura em nuvem utilizada podem não estar sendo plenamente aproveitadas ou talvez demandem ajustes e modificações para melhor atender às demandas.

Olhando também para o lado de sistemas operacionais, é possível concluir que, hoje, existem pouquíssimas opções viáveis para instalação nesse tipo de processador. As que existem são destinadas em sua maioria para usos fora do escopo deste projeto. A opção mais promissora que estava próximo da realidade de experiência de uso do laboratório de computação foi o *Armbian*, que tem prospecção de manutenção e suporte. Porém, a distribuição deixou a desejar no quesito desempenho, onde ao utilizarmos uma *distro* (distribuição *Linux*) com interface gráfica, o sistema apresentou lentidão, travamentos e engasgos em tarefas básicas. Um bom e mais profundo próximo passo seria buscar um meio de fazer de fato a portabilidade de uma distribuição *Linux* customizada para essa caixa em específico, respeitando suas limitações e vantagens, algo que talvez poderia ser feito de forma *open source* para facilitar e acelerar o desenvolvimento. Outra coisa importante de se observar é que os sistemas operacionais encontrados e as discussões acerca destes são consideravelmente recentes, e sendo assim, é possível presumir que novidades acerca desse assunto continuem aparecendo ao longo do tempo, auxiliando num possível trabalho futuro.

Portanto, é possível concluir que, para o objetivo de produzir um computador de baixo custo de uso em propósitos pedagógicos, o emprego das TV Boxes apresenta oportunidades, porém, ainda demanda esforços para viabilizar sua utilização para as tarefas propostas. É imprescindível que tanto sua usabilidade quanto seu desempenho sejam avaliados para que seu emprego, de fato, contribua com a inclusão digital e o avanço da educação no país.

6 Agradecimentos

Gostaríamos de agradecer a todos que nos auxiliaram neste projeto:

- Cristiano, Arquiteto de Soluções da *AWS*, que viabilizou o estudo de caso utilizando a infraestrutura em nuvem da Amazon e trouxe a equipe de especialistas da *DataRain* para nos apoiar no projeto.
- Leonardo, Melissa e Luciane, da empresa *DataRain*, que fizeram a implementação da infraestrutura em nuvem e, sempre, de maneira rápida e precisa nos atenderam nas solicitações dos ajustes necessários.
- André e Thiago da equipe de TI do Instituto de Computação da UNICAMP, que não mediram esforços para criar a infraestrutura de rede para o nosso teste, criando uma rede Wi-fi exclusiva e até mesmo criando uma rede cabeada particular para o teste presencial com os voluntários.
- À todos os voluntários que dispuseram do seu valioso tempo em apoio ao desenvolvimento científico e tecnológico, provendo opiniões valiosas e indispensáveis para este projeto.

Referências

Tutorial de instalação de Linux para TV Box - modelo TX2. Disponível em https://smartcampus.prefeitura.unicamp.br/pub/artigos_relatorios/Smart_Campus_Tutorial_TV_Box_TX2.pdf. Acesso em 18 de Agosto de 2023.

Multitool latest version. Disponível em: https://users.armbian.com/jock/rk322x/multitool. Acesso em: 27 de Agosto de 2023.

Armbian versions for Rk322x-box Disponível em: https://imola.armbian.com/dl/rk322x-box/archive/. Acesso em: 27 de Agosto de 2023.

WIRESHARK. Filtros de Captura. Wireshark Wiki. Disponível em: https://wiki.wireshark.org/CaptureFilters. Acesso em: 30 de Agosto de 2023.

Educação remota através do Fire TV Stick e Laboratórios Virtuais com instâncias Amazon EC2 Spot | O blog da AWS. Disponível em: https://aws.amazon.com/pt/blogs/aws-brasil/educacao-remota-atraves-do-fire-tv-stick-e-laboratorios-virtuais-com-instancias-amazon-ec2-spot. Acesso em: 5 de Setembro de 2023.

ANDROID DEVELOPERS. ADB (Android Debug Bridge). Android Developers. Disponível em: https://developer.android.com/tools/adb?hl=pt-BR. Acesso em: 11 de Setembro de 2023.

TEIXEIRA, F. O que é o SUS (System Usability Scale) e como usá-lo em seu site. Disponível em: https://brasil.uxdesign.cc/o-que-%C3%A9-o-sus-system-usability-scale-e-como-us%C3%A1-lo-em-seu-site-6d63224481c8. Acesso em: 23 de Setembro de 2023.

PHD, J. S. Measuring Usability with the System Usability Scale (SUS) – MeasuringU. Disponível em: https://measuringu.com/sus. Acesso em: 27 de Setembro de 2023.

ANDROID DEVELOPERS. Logcat. Android Developers. Disponível em: https://developer.android.com/studio/command-line/logcat?hl=pt-BR. Acesso em: 28 de Setembro de 2023.

JOCK. CSC Armbian for RK322X TV Boxes. Armbian Forum, 8 jan. 2020. Disponível em: https://forum.armbian.com/topic/12656-csc-armbian-for-rk322x-tv-boxes/. Acesso em: 5 de Outubro de 2023.

Como Trocar Botão de Celular Soldado na Placa (O Jeito Mais Fácil). Disponível em: https://www.youtube.com/watch?v=P1hH00sl-Ko. Acesso em: 13 de Outubro de 2023.

WYK72. OpenWrt 22.03-rc6 “Build” for TV-boxes (rk322x). OpenWrt Forum, 22 ago. 2022. Disponível em: https://forum.openwrt.org/t/openwrt-22-03-rc6-build-for-tv-boxes-rk322x/134738. Acesso em: 16 de Outubro de 2023.

ILMICH. [UNOFFICIAL][RK3228/RK3229][box]Libreelec Builds. Libreelec Forum, 3 fev. 2022. Disponível em: https://forum.libreelec.tv/thread/25236-unofficial-rk3228-rk3229-box-libreelec-builds/. Acesso em: 18 de Outubro de 2023.

Como Trocar Qualquer Botão de Celular Com Ferro de Solda (Muito Fácil). Disponível em: https://www.youtube.com/watch?v=bB0ldodsNC8. Acesso em: 24 de Outubro de 2023.