Implementação e análise de desempenho de um sistema *single board* para monitoramento aéreo

Marianna de Pinho Severo[1], Arthur de Castro Callado[1]

[1]Universidade Federal do Ceará – Campus de Quixadá
Rua José de Freitas Queiroz, 5003 – 63.902-580 – Quixadá – CE – Brasil

mariannapinho@alu.ufc.br, arthur@ufc.br

Abstract. *Air traffic has been ever growing and ADS-B (Automatic Dependent Surveillance - Broadcast) technology has proven to be an important tool to help in the operation of this sector. However, proprietary and expensive devices are employed, limiting the adoption of this technology by many users with lower financial resources. Thus, in order to provide a lower cost solution, this work developed and performed the performance analysis of an open source air monitoring system based on ADS-B technology, applied to single board hardware. The results showed that the single board system performed well compared to the general purpose platform used for comparison.*

Resumo. *O tráfego aéreo tem crescido cada vez mais e a tecnologia ADS-B (Automatic Dependent Surveillance - Broadcast) se mostra uma importante ferramenta para auxiliar o funcionamento do setor. Entretanto, dispositivos proprietários de elevado custo são usados, limitando a adoção da tecnologia por usuários com menos recursos financeiros. Com o objetivo de fornecer uma solução de mais baixo custo, este trabalho desenvolveu e realizou a análise de desempenho de um sistema de código aberto para monitoramento aéreo baseado na tecnologia ADS-B, aplicado a hardware single board. Os resultados mostraram que o sistema single board apresentou bom desempenho quando comparado à plataforma de propósito geral usada para comparação.*

1. Introdução

O número de aeronaves tem crescido bastante nos últimos anos [Gössling e Humpe 2020]. Em decorrência disso e devido às limitações dos tradicionais sistemas de radares empregados para o monitoramento e controle do espaço aéreo, novas tecnologias, como a ADS-B (*Automatic Dependent Surveillance - Broadcast*) [Su et al. 2018], começaram a ser desenvolvidas para garantir e melhorar o funcionamento dos sistemas de transporte aéreo [Park e Tomlin 2014]. Apesar das vantagens, tanto a tecnologia ADS-B como os radares ainda são implementados em sistemas de alto custo com softwares proprietários. Tal cenário é viável para aeroportos médios e grandes, mas é inviável em aeroportos de pequeno porte (de cidades pequenas ou propriedades privadas), pelas limitações de recursos para sua implementação e manutenção [Feitosa et al. 2015]. Além disso, dificulta pesquisas que buscam melhorar a tecnologia ADS-B [Schäfer et al. 2014].

Recentemente, produz-se dispositivos cada vez menores, mais baratos e de maior poder computacional. Um exemplo é a plataforma Raspberry Pi [Vujović e Maksimović 2015], usada em sistemas embarcados ou de propósito geral. Com a tecnologia ADS-B para melhor controle do espaço aéreo e seu custo dificultando sua adoção, plataformas de menores custos vêm ganhando espaço.

Nesse contexto, alguns trabalhos vêm sendo desenvolvidos. Um deles é o de [Feitosa et al. 2015], que realiza a implementação e avaliação de um sistema coletor de mensagens ADS-B em dois tipos de plataforma, uma embarcada e outra de propósito geral. Entretanto, [Feitosa et al. 2015] utiliza uma API (*Application Programming Interface*) de terceiros para a decodificação das mensagens e as linguagens empregadas para o desenvolvimento do sistema exigem mais recursos da plataforma de *hardware*.

No trabalho de [Pahlevy et al. 2018], é desenvolvido um nano satélite para a captura de mensagens ADS-B, em Raspberry Pi. Todavia, a troca de mensagens com um computador local e as análises não verificam o consumo dos recursos de *hardware*. Já [Abdulaziz et al. 2015] simula a transmissão e recepção de mensagens ADS-B com dados previamente capturados e analisa a capacidade do sistema de detectar erros de paridade e mede taxas de erro de *bit* e de pacote sem investigar o consumo de recursos.

Em [Schäfer et al. 2014] é analisado o OpenSky, que permite a visualização de aeronaves em tempo real e fornece *software* e *hardware* coletor de mensagens ADS-B aos interessados em contribuir. O objetivo é fornecer uma rede de sensores aberta. O trabalho faz uma avaliação do sistema em dois anos de funcionamento, analisando a quantidade de mensagens recebidas, o sistema de multilateração empregado e apresentando métricas sobre o canal de comunicação. Todavia, o trabalho ignora o uso de recursos computacionais e, embora o acesso seja aberto, o sistema não é aberto.

Assim, este trabalho implementa um *software* de código aberto para monitoramento de aeronaves equipadas com a tecnologia ADS-B e analisa o desempenho de sua aplicação em plataforma *single board*, semelhante à Raspberry Pi. Foram realizados experimentos para avaliar a eficiência de um sistema de mais baixo custo. Os resultados mostraram que o sistema na plataforma *single board* apresenta resultados similares aos obtidos em plataformas de maior poder computacional, como um computador pessoal. Assim, espera-se contribuir para a melhoria do monitoramento do espaço aéreo, tornando mais fácil a utilização da tecnologia ADS-B por pessoas ou instituições de menor poder aquisitivo - como pequenos e médios aeroportos e outros interessados -, contribuindo para a comunidade com um *software* de código aberto.

Para atender aos objetivos do trabalho, duas principais etapas foram executadas: o desenvolvimento de um decodificador de mensagens ADS-B, em linguagem C, obtendo informações de identificação, velocidade e posicionamento das aeronaves; e a análise de desempenho comparando este *software* com um desenvolvido em linguagem Python numa etapa anterior, em duas plataformas de *hardware* diferentes: um computador pessoal e uma *single board*. Este trabalho tem como base trabalhos realizados no projeto "Atualização da Monitoração Aeronáutica e Auto-Sustentabilidade", desenvolvido por professores e alunos da Universidade Federal do Ceará (UFC) no Campus de Quixadá.

2. Fundamentação teórica

2.1. Sistemas de radares e tecnologia ADS-B

O monitoramento do espaço aéreo é realizado por diferentes sistemas de controle, tais como as Torres de Controle Aéreo, os Controles de Aproximação e os Controles de Área, de acordo com a etapa do voo em que cada aeronave se encontra. Para viabilizar as atividades de controle e monitoramento aéreo, adotou-se a tecnologia de radares, que se comunicam com os equipamentos das aeronaves, enviando requisições e capturando as respostas contendo identificação e altitude [Abdulaziz et al. 2015].

Entretanto, as limitações dos sistemas de radares - como os elevados custos de equipamentos, instalação, operação e manutenção, a dificuldade de fornecer cobertura em algumas regiões, uma baixa resolução para as informações de posicionamento, grande diminuição de desempenho na presença de ruídos e um longo período de atualização dos dados - motivaram organizações de todo o mundo a desenvolver uma nova tecnologia, dando origem ao ADS-B [Abdulaziz et al. 2015][Schäfer et al. 2014].

O ADS-B fornecer maior capacidade, eficiência e segurança no controle do tráfego aéreo [Park and Tomlin 2014]. Suas vantagens incluem: maior precisão no posicionamento das aeronaves, maior taxa de atualização dos dados, menores custos e melhor desempenho no monitoramento de baixas altitudes e possibilidade de obtenção de informações diretamente entre as aeronaves [Su et al. 2018].

Essa tecnologia é dita dependente, pois utiliza outros sistemas, como o GNSS (*Global Navigation Satellite System*) [Abdulaziz et al. 2015], para obter algumas informações. Além disso, é automática por não necessitar de intervenção para o envio de mensagens, compartilhando-as periodicamente ou por eventos [Schäfer et al. 2014]. Por fim, o *broadcast* vem da maneira como a transmissão de mensagens é realizada [Park and Tomlin 2014].

As aeronaves enviam suas informações através de mensagens do tipo ADS-B, que podem ser de 56 *bits* ou de 112 *bits*. Este trabalho utilizou um receptor ADS-B que captura sinais enviados na frequência de 1090 MHz. Recebida uma mensagem, os campos são decodificados conforme estrutura da Tabela 1. Cada tipo de mensagem provê uma informação (e.g., posição, velocidade e identificação).

Tabela 1. Campos que compõem uma mensagem ADS-B

Formato de Downlink (DF)	Capacidade (CA)	Endereço da Aeronave (ICAO)	Dados	Paridade (PI)

Fonte: Adaptado de Junzi (2018)

2.2. Plataformas *single board*

Recentemente, os computadores *single board*, definidos como computadores completos em uma única placa, têm se destacado devido ao seu baixo custo, ao pequeno consumo de energia e ao razoável poder computacional. As plataformas Raspberry Pi e Beaglebone Black [Johnston et al. 2018] são exemplos. Dentre as desvantagens estão a limitação de recursos, como processamento, memória e armazenamento [Johnston et al. 2018][Qureshi and Koubâa 2019]. Apesar disso, *single boards* conseguem executar uma variedade de sistemas operacionais, sendo o mais popular o Linux, e lidar com cargas de trabalho convencionais [Qureshi and Koubâa 2019].

Tabela 2. Especificações mais detalhadas do hardware single board

Single Board	Orange Pi PC Plus
CPU	H3 Quad-core ARM Cortex-A7 H.265/HEVC 4K
Memória	1GB DDR3 (compartilhada com GPU)
Armazenamento	TF card (Max. 32GB) / MMC card slot 8GB EMMC Flash
Sistema operacional	Linux Armbian Xenial 5.38 kernel 3.4.113 Orangepipcplus.

Fonte: Adaptado de Orange Pi (2016)

Este trabalho escolheu a plataforma Orange Pi Pc Plus, como *single board* para a execução do *software* de decodificação ADS-B empregando o Linux. Além disso, realizou-se a análise do desempenho dessa plataforma durante a execução do monitoramento. A Tabela 2 apresenta as principais configurações *single board* usada.

2.3. Análise de desempenho

A análise de desempenho consiste num conjunto de atividades para avaliar um sistema e entender os resultados, contribuindo para uma tomada de decisão. Ela possui diversos conceitos, tais como: métricas, fatores, níveis, carga de trabalho, parâmetros, serviços fornecidos e seus resultados [Jain, 1990]. Além disso, cada contexto gera a necessidade de uma análise de desempenho única e sua avaliação é sujeita a erros. Entretanto, um conjunto de etapas comuns, chamada Abordagem Sistemática [Jain 1990], pode ser empregado para evitar erros comuns.

Em uma análise de desempenho, considerar a diferença entre amostra e população. Uma amostra é um subconjunto da população e afirmações sobre uma amostra não podem ser generalizadas como afirmações definitivas sobre a população. Apesar disso, pode-se utilizar os resultados ao se avaliar amostras para a construção de definições probabilísticas sobre a população [Jain 1990][Cação 2010]. Na estatística inferencial, duas ferramentas são usadas para esse fim: testes de hipótese e intervalos de confiança. Os primeiros permitem verificar uma determinada suposição sobre a população baseando-se nas evidências obtidas a partir de amostras. Já os segundos permitem a criação de intervalos de estimadores para um determinado parâmetro populacional, segundo determinado nível de confiança [Montgomery and Runger 2016]. Além disso, podem-se adotar métodos paramétricos e não paramétricos, dependendo se os dados atendem ou não a certos critérios de normalidade [Cação 2010].

Neste trabalho empregaram-se testes de hipótese e intervalos de confiança para a análise dos experimentos. Testes de normalidade foram aplicados a cada conjunto de dados e técnicas não paramétricas foram abordadas, uma vez que a grande maioria dos dados apresentou distribuição não normal.

3. Metodologia

Para que os objetivos propostos neste trabalho fossem alcançados, um conjunto de etapas foi adotado: a implementação do coletor de mensagens ADS-B em C; a adaptação do coletor de mensagens ADS-B em Python; a realização de experimentos preliminares; e a análise de desempenho do sistema desenvolvido.

O *software* coletor foi desenvolvido em linguagem C por esta ser uma das mais utilizadas para a construção de sistemas embarcados [Sebesta 2009]. A construção do *software* ocorreu em quatro etapas: implementação da comunicação com o Receptor microADSB[1]; decodificação das mensagens; implementação das operações de banco de dados; e comunicação com o servidor remoto.

A Figura 4 apresenta o fluxograma do algoritmo desenvolvido. Foram criadas duas *threads*, sendo uma exclusivamente para a comunicação com o servidor. A interação entre elas é através da lista de informações a serem enviadas. Para que as

[1] http://www.anteni.net/adsb/index.html#!/microADSB-USB-receiver/p/15504142/category=3647494

informações da lista fossem enviadas, usou-se o formato JSON, com a biblioteca json-c. Já para a comunicação com o servidor remoto, empregou-se a biblioteca Libcurl.

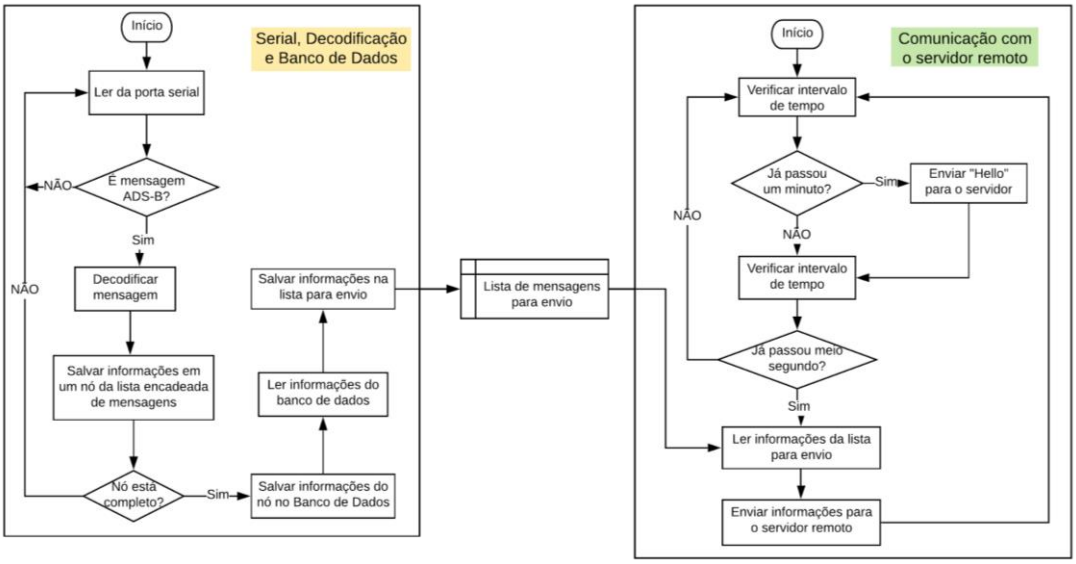

Figura 4. Estrutura do Coletor ADS-B desenvolvido em C.

O coletor em Python foi adaptado adicionando-se as operações de escrita e leitura em banco de dados, de maneira que ambos os Coletores apresentassem algoritmos semelhantes. Uma das principais diferenças é que, em C, sempre que uma nova mensagem ADS-B é recebida, ela é decodificada e suas informações são armazenadas em um nó de uma lista encadeada. Ao passo que o código em Python salva as mensagens recebidas e apenas as decodifica quando a instância da classe que as armazena é considerada completa. Para que um nó ou uma instância sejam considerados completos, eles precisam ter um conjunto mínimo de informações, que são: as mensagens de posição, a mensagem de identificação e a de velocidade.

Experimentos preliminares foram realizados para testar os equipamentos (quatro antenas, dois receptores microADSB e duas plataformas *single board*) a serem utilizados. Os resultados mostraram que um dos receptores estava com defeito. Como a obtenção de outro receptor em tempo hábil não foi viável, os experimentos de avaliação de desempenho foram realizados empregando-se apenas um receptor. Assim, cada experimento foi executado individualmente, com a coleta dos dados em um ambiente real, não sendo possível a utilização dos mesmos dados em cada coleta.

A análise de desempenho teve como objetivos a avaliação do uso de recursos computacionais durante a execução dos *softwares* decodificadores de mensagens ADS-B e a capacidade do sistema de continuar a fornecer serviços de acordo com a utilização desses recursos. Para isso, dois *hardwares* foram empregados, sendo um deles uma plataforma *single board* (Tabela 2) e o outro, um computador pessoal (Tabela 3). Dois *softwares* foram utilizados, um desenvolvido em linguagem C e o outro em Python.

Tabela 3. Especificações do computador pessoal.

CPU	Intel® Core™ i5-5200U CPU@ 2.20GHz×4
Memória	8GB, compartilhada com a placa de vídeo integrada
Armazenamento	Disco de 1 TB
Sistema operacional	Ubuntu 18.04.3 LTS

Como métricas para análise de desempenho, empregou-se o uso de processador e uso de memória durante a execução dos serviços; o tempo para o tratamento de cada mensagem; a quantidade total de mensagens recebidas por minuto; dessas, a proporção das que são do tipo ADS-B; e delas, a taxa de mensagens que o sistema consegue decodificar, já que ele não está tratando todos os tipos mensagens ADS-B.

Inicialmente, planejou-se a execução simultânea dos sistemas nas duas plataformas de *hardware*, para que eles tivessem a mesma amostra. Mas com o defeito em um dos receptores, usou-se uma plataforma por vez. Foram quatro cenários:

a) Cenário 1: hardware *single board* e Coletor em C.
b) Cenário 2: hardware *single board* e Coletor em Python.
c) Cenário 3: computador pessoal e Coletor em C.
d) Cenário 4: computador pessoal e Coletor em Python.

No total, foram realizados doze experimentos, cada um com duração mínima de 8 horas e máxima de 12 horas e 35 minutos, alternando-se entre os quatro tipos de cenários da seguinte maneira: primeiro o Cenário 1, depois o Cenário 3, a seguir o Cenário 2 e, por fim, o Cenário 4. Então, repetiu-se essa ordem, até completar os doze experimentos, obtendo-se um total de três capturas por cenário. Todas as capturas foram realizadas no mesmo local e a condição de parada foi já ter passado oito horas ou mais.

A análise adotou os passos da Figura 5: realizou-se a transformação dos dados, para facilitar sua visualização; fez-se a sua divisão em amostras menores; verificou-se que medida de tendência central utilizar para a estimativa populacional; usaram-se testes de normalidade para verificar que métodos estatísticos adotar; e realizou-se a análise dos dados conforme técnicas estatísticas mais adequadas para a distribuição.

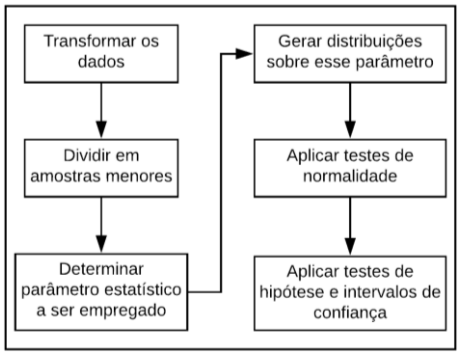

Figura 5. Fluxograma de análise dos dados

4. Resultados e discussões

O consumo de processador, assim como o uso de memória, fornecem um indicativo do esforço do *hardware* para cumprir as atividades requisitadas pelos *softwares*. Neste trabalho, realizou-se a medição da porcentagem de uso do processador e do uso de memória (em *megabytes*) nos quatro cenários descritos anteriormente. Os valores mensurados levam em consideração todos os processos sendo executados na plataforma e não apenas aqueles criados para a execução dos *softwares* de monitoramento aéreo. Exemplos de captura ao longo do tempo podem ser observados na Figura 6 (processador) e na Figura 7 (memória). Os dados de uso de processador e de memória foram registrados a cada cinco segundos. Já os tempos de tratamento de mensagens foram registrados a cada mensagem ADS-B decodificável recebida.

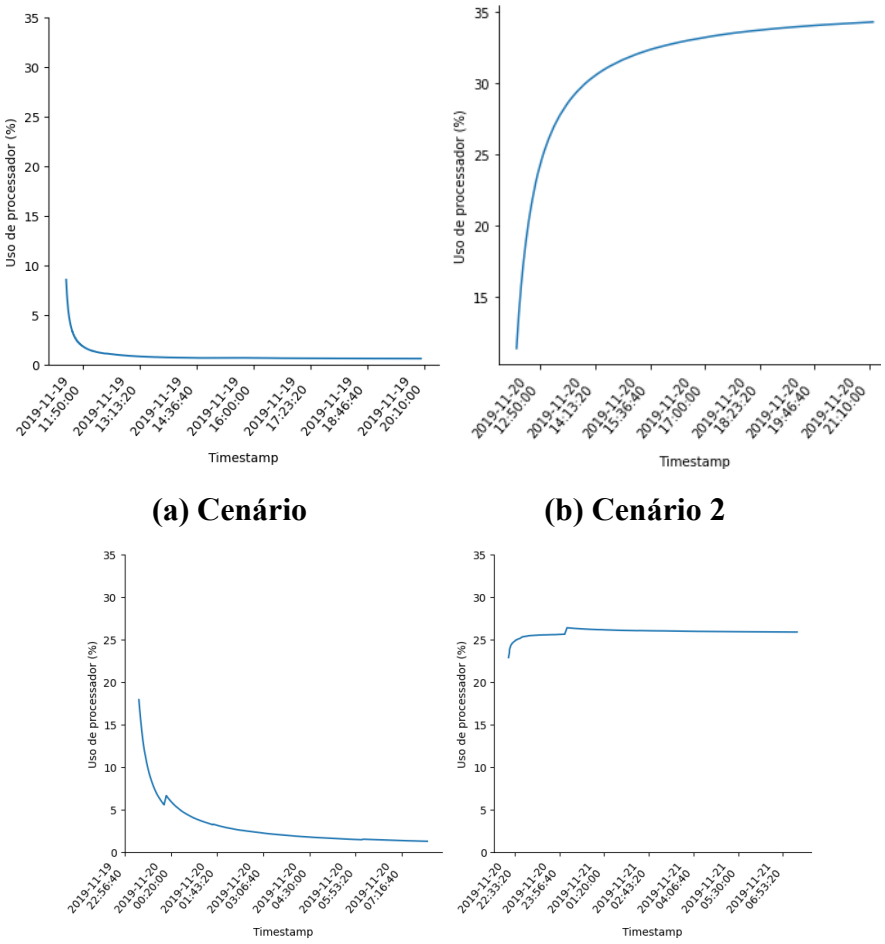

Figura 6. Uso do processador ao longo do tempo nos cenários 1 (a), 2 (b), 3 (c) e 4 (d).

Conforme se pode observar, nos cenários com o Coletor em C o uso do processador diminuiu com o decorrer do tempo, ao passo que aumentou nos cenários com o Coletor em Python. Isso sugere que, ao longo do tempo, os recursos de processamento exigidos pelo código em Python são maiores que aqueles impostos pelo código em C. Outra observação importante é que o consumo de processamento no Cenário 2 torna-se maior do que aquele do Cenário 4. Ou seja, embora ambos cresçam, os recursos exigidos do single board são maiores que os do computador pessoal.

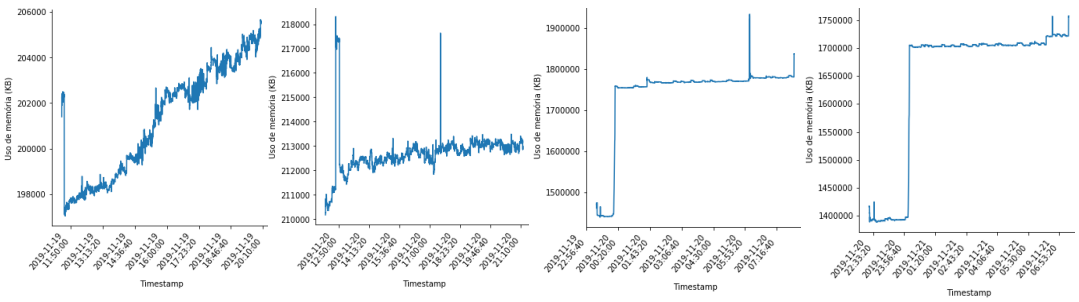

Figura 7. Consumo de memória ao longo do tempo nos cenários 1 (a), 2 (b), 3 (c) e 4 (d).

Em todos os cenários, o uso de memória aumenta ao longo do tempo. Entretanto, nos cenários 1 e 2, usando a plataforma *single board*, o uso de memória apresentou mais variações, conforme pode ser percebido pelas curvas mais ruidosas (Figura 7). Além disso, percebe-se que na maioria das vezes ele é maior no Cenário 2 que no Cenário 1. Nos cenários 3 e 4, os valores foram mais estáveis.

O tempo para tratamento das mensagens indica a eficiência do sistema na realização das atividades. Quanto maior esse tempo, maior a probabilidade de que as informações não sejam entregues em tempo hábil. Neste trabalho, considerou-se esse tempo como todo o intervalo para a decodificação das mensagens, as operações de banco de dados e o armazenamento das informações na lista de mensagens para envio ao servidor. A Figura 8 apresenta exemplos de captura para os quatro cenários.

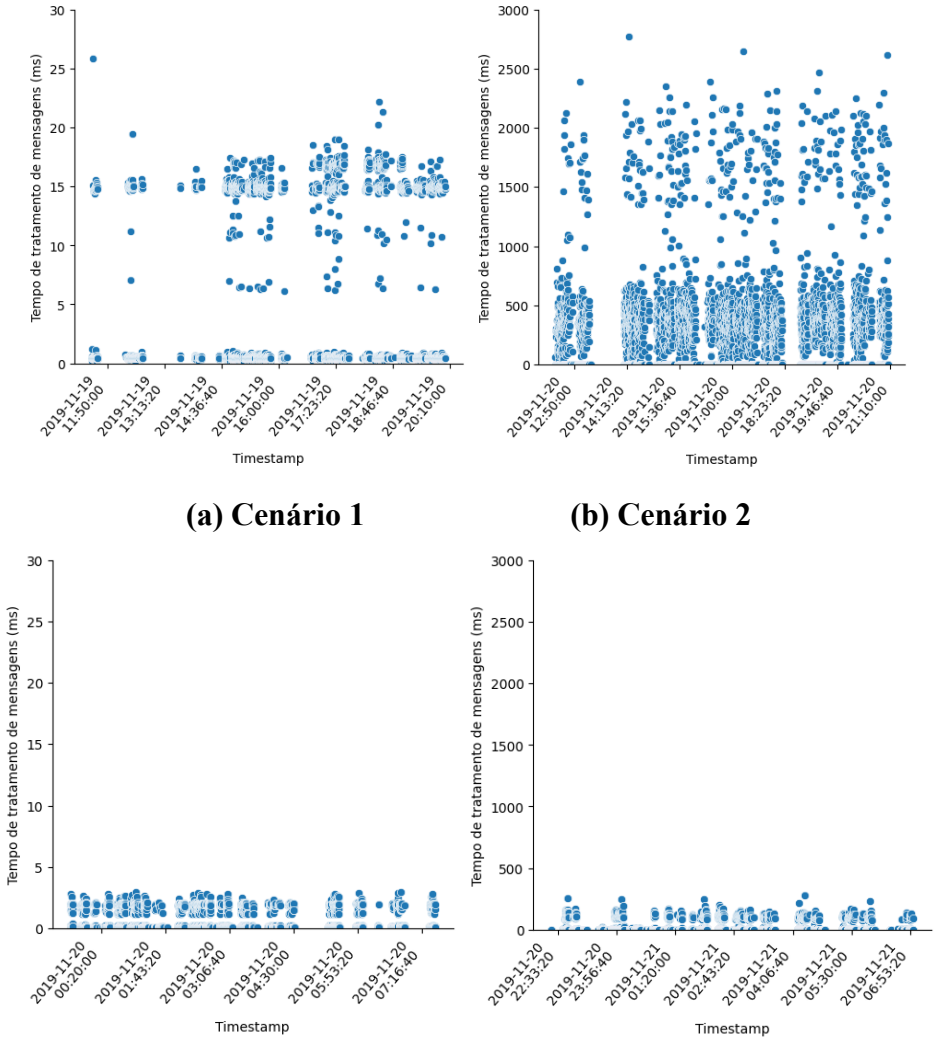

Figura 8. Tempo de tratamento das mensagens nos cenários 1 (a), 2 (b), 3 (c) e 4 (d).

De acordo com os gráficos, pode-se observar que os cenários em que o Coletor em C foi executado (1 e 3) apresentaram menores tempos de tratamento do que aqueles para o código em Python (2 e 4). Dessa forma, os dois melhores cenários - aqueles que apresentam os menores tempos de decodificação - foram o 1 e o 3, com o Cenário 3 melhor do que o Cenário 1; e o pior cenário foi o 2.

O próximo passo realizado para todas as três métricas descritas foi a construção de amostras menores, de tamanho 120, a partir da reamostragem dos dados em sequencia. Todos os dados foram usados, mudando somente o agrupamento. Para cada uma dessas amostras, foi aplicado o teste de normalidade de Anderson-Darling (o teste de Shapiro-Wilk não é capaz de tratar amostrar pequenas), com nível de confiança de 95%, verificando-se que a maioria delas apresentou distribuição não normal. Com esses resultados, escolheu-se a mediana como estimador populacional. Para cada métrica e cenário, construiu-se uma distribuição para as medianas das amostras geradas anteriormente. Para essas novas distribuições, aplicou-se novamente o teste de normalidade de Anderson-Darling, com 95% de confiança, mostrando que a maioria delas não se apresentou normal. Então, aplicou-se o teste de Kruskal-Wallis para se verificar a semelhança das distribuições, de acordo com a métrica, observando-se que existiam diferenças significativas entre elas. Ou seja, o uso de processador entre os cenários, assim como o uso de memória e o tempo para tratamento das mensagens, apresentou-se diferente, seja devido aos valores ou ao comportamento de uso.

Por fim, aplicou-se o teste de Dunn para se determinar quais distribuições apresentaram diferenças significativas. Foi verificado, para o uso de processador, que todas as distribuições apresentaram grandes diferenças. Já, para o uso de memória e para o tempo de tratamento de mensagens, os cenários 3 e 4 não apresentaram diferenças significativas, mostrando que os cenários em que os computadores pessoais foram empregados apresentaram comportamento semelhante, independentemente do coletor empregado. Além disso, calculou-se o intervalo de confiança para as medianas de cada métrica, para cada cenário, aplicando-se a técnica de *bootstrap*, uma vez que os dados apresentaram-se não paramétricos. Um exemplo de intervalos de confiança pode ser observado na Tabela 4, para o uso de processador.

Tabela 4. Intervalos de confiança para o uso de processador (%).

Cenário	Limite inferior	Mediana amostral	Limite superior
1	0,63925	0,6698	0,6812
2	34,13725	34,41418	34,71531
3	1,34735	1,54275	1,67565
4	25,88	25,90781	25,92478

Já com relação às mensagens recebidas, suas métricas fornecem indicativos sobre os serviços fornecidos pelos sistemas coletores de mensagens ADS-B. De acordo com a Tabela 5, pode-se observar que o cenário que teve maior tempo de captura foi o 2. Apesar disso, este cenário também foi o que teve acesso ao menor número de mensagens. Por sua vez, o Cenário 1 teve acesso à maior quantidade de mensagens.

Tabela 5. Quantidades totais de mensagens.

Cenário	Duração (Horas: Minutos)	Mensagens recebidas	Mensagens ADS-B	Mensagens ADS-B decodificadas	Mensagens ADS-B não decodificadas
1	25:59	81847	54552	48008	6544
2	32:51	18116	11564	10331	1233
3	25:36	38596	23885	21672	2213
4	30:22	55780	29753	26738	3015

Ao se observar a Tabela 6, nota-se que a quantidade de mensagens recebidas por minuto reflete o total de mensagens recebidas. Todavia, a possibilidade de os coletores terem passado certo tempo sem receber nenhuma mensagem e, em outros momentos, terem recebido uma grande quantidade, leva à necessidade de análises mais detalhadas, que observem intervalos menores de captura.

Ainda na Tabela 6, pode-se perceber que, mesmo tendo acesso a quantidades diferentes de mensagens, os cenários possuem uma taxa semelhante de mensagens ADS-B recebidas em relação ao total de mensagens. Assim, mesmo para conjuntos de diferentes tamanhos, a proporção de mensagens ADS-B dentro deles é aproximada. Ademais, observa-se que as taxas de mensagens que podem ser decodificadas em relação às mensagens ADS-B também são semelhantes entre os cenários.

Tabela 6. Taxa de mensagens.

Cenário	Mensagens recebidas por minuto	Mensagens ADS-B recebidas por minuto	Mensagens decodificadas por minuto	Mensagens ADS-B pelo total de mensagens	Mensagens decodificadas pelo total de ADS-B
1	52,4997	34,9917	30,7941	66,65 %	88 %
2	9,1913	5,8671	5,2415	63,83 %	89,34 %
3	25,1276	15,5501	14,1094	61,88 %	90,73 %
4	30,6147	16,3298	14,6751	53,34 %	89,87 %

Com essas informações, pôde-se verificar que, independentemente das populações a que tiveram acesso, todos os cenários apresentaram desempenhos semelhantes com relação à quantidade de mensagens que conseguiram decodificar. Além disso, para as populações consideradas, as quantidades de mensagens ADS-B presentes, independente das quantidades totais de mensagens recebidas, também foram aproximadas. Isso pode sugerir que tanto a plataforma de mais baixo custo (*single board*), como a de propósito geral, apresentaram desempenhos semelhantes, o que justifica a adoção de uma plataforma e menor custo sem perda de desempenho.

5. Considerações finais

Neste trabalho realizou-se a implementação e análise de desempenho de um sistema *single board* para monitoramento aéreo, baseado na tecnologia ADS-B. Um *software* em C foi desenvolvido e outro, em Python, foi adaptado. Além disso, empregou-se uma plataforma *single board* e outra de propósito geral comparar o desempenho de *hardware* com diferentes recursos computacionais para as atividades de monitoramento aéreo.

De acordo com o percentual de uso do processador, pôde-se verificar que os cenários apresentaram comportamentos de uso diferentes: aqueles em que o Coletor em Python foi empregado exigiram mais recursos ao longo do tempo. Já com relação ao uso de memória, todos os cenários apresentaram um consumo crescente, mas apenas os cenários em que o *hardware* foi o computador pessoal não apresentaram diferenças significativas com relação ao seu comportamento. Por fim, no que diz respeito ao tempo para tratamento das mensagens, os cenários em que o Coletor em C foi empregado exigiram menos tempo do que os do Coletor em Python, tendo este apresentado valores

em segundos. Para todas as métricas, o Coletor em Python demonstrou necessitar de mais recursos das duas plataformas de *hardware* do que o Coletor em C.

Por fim, levando-se em consideração as métricas calculadas sobre as mensagens recebidas, observou-se que os cenários tiveram acesso a quantidades diferentes de mensagens. Entretanto, foi possível observar que as taxas de mensagens ADS-B presentes nos conjuntos de mensagens aos quais os cenários tiveram acesso foram próximas umas das outras. Além disso, notou-se que as taxas de mensagens decodificadas em relação aos totais de mensagens ADS-B recebidas também foram aproximadas, indicando um desempenho semelhante entre os quatro cenários com respeito a essa métrica.

Portanto, pode-se sugerir que mesmo plataformas de *hardware* com menores poderes computacionais e mais baixos custos, como a plataforma *single board* aplicada neste trabalho, podem ser empregadas para atividades de monitoramento aéreo, apresentando resultados tão bons como os de um *hardware* com maiores poderes computacionais, como o computador pessoal adotado. Apesar disso, deve-se levar em consideração que, conforme a demanda por recursos aumenta, o desempenho do sistema pode piorar. Isso foi o que possivelmente ocorreu com o Cenário 2, em que o Coletor em Python foi executado na plataforma *single board*, exibindo as maiores taxas de uso de processador, de uso de memória, quando comparado ao Cenário 1 que também empregou a plataforma *single board* e os maiores tempos para o tratamento das mensagens.

Como trabalhos futuros, propõe-se a realização de experimentos em que todos os cenários tenham acesso à mesma população, de maneira que possa-se avaliar outros aspectos dos sistemas utilizados. Uma abordagem para isso seria empregar-se dados, já coletados, em uma simulação em que os mesmos dados seriam enviados, assim como feito em Abdulaziz et al. (2015). Além disso, métricas que levem em consideração outros aspectos dos sistemas também podem ser adotadas, como por exemplo, as que avaliam a comunicação com o servidor remoto. Por fim, pode-se realizar a análise do consumo de energia pelo sistema desenvolvido e a sua adaptação para ser capaz de se comunicar com qualquer tipo de equipamento de recepção, criando-se, por exemplo, um arquivo que mapeia o formato de mensagens para o tipo de receptor.

Referências

Abdulaziz, Abdulrazaq et al. Optimum receiver for decoding automatic dependent surveillance broadcast (ADS-B) signals. **American Journal of Signal Processing**, v. 5, n. 2, p. 23-31, 2015.

Cação, Rosário. **Testes estatísticos: testes paramétricos e não paramétricos**. [S. l.], 2010. 43 slides. Disponível em: https://pt.slideshare.net/rosariocacao/testes-parametricos-e-nao-parametricos-3396639. Acesso em: 02 mar. 2024.

Feitosa, Antônio Guilherme Estevão Soares. **Coletor de dados para monitoramento aéreo usando ADS-B e mini PC Android.** 2016. 37 f. TCC (graduação em Sistemas de Informação) - Universidade Federal do Ceará, Campus Quixadá, Quixadá, CE, 2016. Disponível em: http://www.repositoriobib.ufc.br/000028/00002825.pdf. Acesso em: 02 mar. 2024.

Gössling, Stefan and Humpe, Andreas. **The global scale, distribution and growth of aviation: Implications for climate change**. Global Environmental Change, v. 65, p. 102194, 2020.

Jain, Raj. **The art of computer systems performance analysis: techniques for experimental design, measurement, simulation, and modeling**. [New York]: John Wiley & Sons, 1990.

Johnston, Steven J. et al. Commodity single board computer clusters and their applications. **Future Generation Computer Systems**, v. 89, p. 201-212, 2018.

Junzi, Sun. **The 1090 MHz Riddle:** An open-access book about decoding Mode-S and ADS-B data. [S. l.], 2018. Disponível em: https://mode-s.org/decode/book-the_1090mhz _riddle-junzi_sun.pdf. Acesso em: 02 mar. 2024.

Montgomery, Douglas C. and Runger, George C. **Estatística aplicada e probabilidade para engenheiros**. 6. ed. Rio de Janeiro: LTC, 2016.

BBC News. **How do you track a plane?**. 17 mar. 2014. Disponível em: https://www.bbc.com/news/world-asia-pacific-26544554. Acesso em: 2 dez. 2019.

Pahlevy, Reza Noval et al. Nanosatellite ADS-B Receiver Prototype for Commercial Aircraft Detection. *In*: INTERNATIONAL CONFERENCE ON CONTROL, ELECTRONICS, RENEWABLE ENERGY AND COMMUNICATIONS (ICCEREC). Indonesia. **Anais** [...]. Indonesia: IEEE, 2018. p. 6-12.

Park, Pangun and Tomlin, Claire. Performance evaluation and optimization of communication infrastructure for the next generation air transportation system. **IEEE Transactions on Parallel and Distributed Systems**, v. 26, n. 4, p. 1106-1116, 2014.

Qureshi, Basit and Koubâa, Anis. On energy efficiency and performance evaluation of single board computer based clusters: a hadoop case study. **Electronics**, v. 8, n. 2, p. 182, 2019.

Schäfer, Matthias et al. Bringing up OpenSky: a large-scale ADS-B sensor network for research. *In*: INTERNATIONAL SYMPOSIUM ON INFORMATION PROCESSING IN SENSOR NETWORKS, 13., 2014. **Proceedings** [...]. Alemanha: IEEE Press, 2014. p. 83-94.

Sebesta, Robert W. **Conceitos de linguagens de programação.** 9.ed. Bookman, 2009.

Shah, Purva. **Reducing Air Traffic congestion with Automatic Dependent Surveillance Broadcast (ADS-B)**. 28 fev. 2017. Disponível em: https://www.einfochips.com/blog/reducing-air-traffic-congestion-with-automatic-dependent-surveillance-broadcast-ads-b/. Acesso em: 02 mar. 2024.

Su, Xiangjun et al. SDR reception and analysis of civil aviation ADS-B signals. *In*: INTERNATIONAL CONFERENCE OF SAFETY PRODUCE INFORMATIZATION (IICSPI). **Anais** [...]. China: IEEE, 2018. p. 893-896.

Vujović, Vladimir and Maksimović, Mirjana. Raspberry Pi as a sensor web node for home automation. **Computers & Electrical Engineering**, v. 44, p. 153-171, 2015.