# Um Framework de Coleta de Indicadores-Chave de Desempenho para Redes 5G O-RAN

Matheus Gomes, Murilo Silva, Victor Leite, João Bastos, Felipe Alves, André Riker e Antônio Abelém

***Resumo*— As rede de acesso via rádio aberta (Open RAN) desagregam as funcionalidades da Rede de Acesso via Rádio (RAN), permitindo mais abertura e interoperabilidade dos componentes da RAN. Em Open RAN, os documentos de padronização especificam e definem os Indicadores-Chave de Desempenho - *Key Performance Metric (KPM)* - como dados que contém informações sobre a infraestrutura de rede, por exemplo, taxas de transferência de dados nas UEs, blocos de recursos físicos e outras. O acesso aos KPMs por componentes do sistema é fundamental para soluções dinâmicas e inteligentes em redes Open RAN, porém as funcionalidades de coleta, armazenamento e tratamento desses dados são únicas a cada aplicação para os controladores inteligentes. Portanto, este artigo apresenta o desenvolvimento e implementação de um framework que visa coletar, tratar e armazenar KPMs em redes Open RAN. A implementação foi realizada em um ambiente emulado desenvolvido pela OAI (OpenAirInterface) composto pelo núcleo da rede 5G e a pilha da RAN, onde o *framework* pôde ser testado e validado obtendo as métricas de 8 UEs.**

***Palavras-Chave*— Open RAN; KPM; Indicadores-Chave de Desempenho; Redes 5G.**

***Abstract*— Open Radio Access Networks (Open RAN) disaggregate the functionalities of the Radio Access Network (RAN), increasing openness and interoperability of RAN components. In Open RAN, standardization documents specify and define Key Performance Metric (KPM) as data that contains information about the network infrastructure, for example, data transfer rates in user equipment, physical resource blocks and others.Access to KPMs by system components is fundamental for dynamic and intelligent solutions in Open RAN networks, but the functionalities for collecting, storing and processing this data are unique to each application for intelligent controllers. Therefore, this paper presents the development and implementation of a framework that aims to collect, process and store KPMs in Open RAN networks. The implementation was carried out in an emulated environment developed by OAI (OpenAirInterface) consisting of the core of the 5G network and the RAN stack, where the framework could be tested and validated by obtaining the metrics of 8 UEs.**

***Keywords*— Open RAN; KPM; 5G Networks.**

## I. Introdução

No contexto atual das redes móveis, é evidente o foco em pesquisas de novas soluções para redes Open RAN (*Open Radio Access Network*), que incorporam em sua essência princípios como a virtualização de funções de rede, fatiamento de rede, interfaces abertas e hardware personalizável [1]. Essas características são fundamentais para promover a interoperabilidade entre diferentes fornecedores e suas respectivas soluções tecnológicas. Assim como as redes Open RAN emergem como estruturas nativas a nuvem, orientadas a dados e adequadas ao uso de *Machine Learning* (ML) [2].

Esses benefícios podem ser aproveitados pelas aplicações dos controladores RIC (RAN Inteligent Controller) pela arquitetura O-RAN que especifica os indicadores-chave de desempenho (*Key Performance Metrics* - KPMs). A partir da coleta desses indicadores, as aplicações podem ser cruciais para realizar o monitoramento e a otimização da rede. Esses indicadores apresentam as métricas de desempenho como fluxo de qualidade de serviço (*Quality of Service* - QoS), utilização de recursos, latência, taxa de transferência, e entre outras informações dos dispositivos participantes de uma operação eficiente da rede O-RAN.

No entanto, não é possível utilizar soluções implementadas de coleta, tratamento e armazenamentos das KPMs que permitam diferentes casos de uso a partir do acesso aos dados, pois, normalmente, cada aplicação teve que desenvolver a coleta e manipulação de métricas de sua própria solução. Isso é um entrave para soluções dinâmicas e autônomas baseadas em ML, os quais podem contribuir com o monitoramento e otimização da rede.

Este artigo propõe um *framework* em forma de aplicação que procura solucionar essa lacuna. A proposta inclui o desenvolvimento de um *framework* em formato de uma xApp acoplada ao Near-RT RIC, em uma infraestrutura com núcleo da rede 5G e uma pilha da RAN com gNodeB monolítica, para realizar a coleta, processamento e armazenamento das KPMs. Portanto, permitindo que o utilizador da aplicação obtenha acesso aos dados necessários para as soluções que envolvam ML de forma mais robusta e eficaz.

O restante deste artigo está dividido da seguinte forma. Seção II apresenta o referencial teórico, abordando os principais conceitos e definições de O-RAN que estão relacionados com o *framework* proposto. Seção III descreve e discute os trabalhos relacionados mais relevantes encontrados na literatura. Seção IV apresenta o *framework* proposto. Seção V e VI introduzem o ambiente de experimentação para condução dos testes de desempenho da proposta e os resultados obtidos, respectivamente. Seção VII apresenta as conclusões e trabalhos futuros.

## II. Referencial Teórico

As redes Open RAN emergiram para tornar as redes de telecomunicações mais abertas, flexíveis e eficientes, sendo

Matheus Gomes, Murilo Silva, Victor Leite, João Bastos, Felipe Alves, André Riker e Antônio Abelém pertencem ao Programa de Pós-Graduação em Ciência da Computação (PPGCC), Federal University of Para, Belém-Pará, e-mail: matheus.cordovil@itec.ufpa.br, murilosilva@itec.ufpa.br, victor.leite@ig.ufpa.br, joao.bastos.junior@itec.ufpa.br, abelem@ufpa.br, ariker@ufpa.br.

projetadas para fazer uso de controladores inteligentes, componentes desagregados e interfaces abertas padronizadas pela ORAN Alliance e pelo 3GPP (*3rd Generation Partnership Project*) [3]. Assim, criam um ambiente a ser implantado nativamente na nuvem, orientado a dados e propício ao uso de técnicas e algoritmos de ML, que potencializam o controle inteligente da rede em um ambiente dinâmico.

Os principais componentes dessa arquitetura, ilustrados na Figura 1, são o SMO (*Service Management and Orchestration*), os RICs: Non-RT RIC (*Non Real Time RIC*) e Near-RT RIC (*Near Real Time RIC*), O-RU (*Open Radio Unit*), O-CU (*Open Centralized Unit*) e O-DU (*Open Distributed Unit*) [4]. O SMO é um *framework* que fornece a orquestração de ponta a ponta, além de hospedar o controlador Non-RT RIC, o qual fornece funcionalidades de controle inteligente da RAN e otimizações em escalas de tempo maiores que 1s. Acima dele são introduzidos rApps que utilizam das funções do SMO e do Non-RT RIC para fornecer funções adicionais com uso de Inteligência Artificial (IA) para otimizar e controlar a RAN. Por intermédio dessas aplicações, podem ser realizadas análises de dados e fornecer informações ao Near-RT RIC por meio da Interface A1.

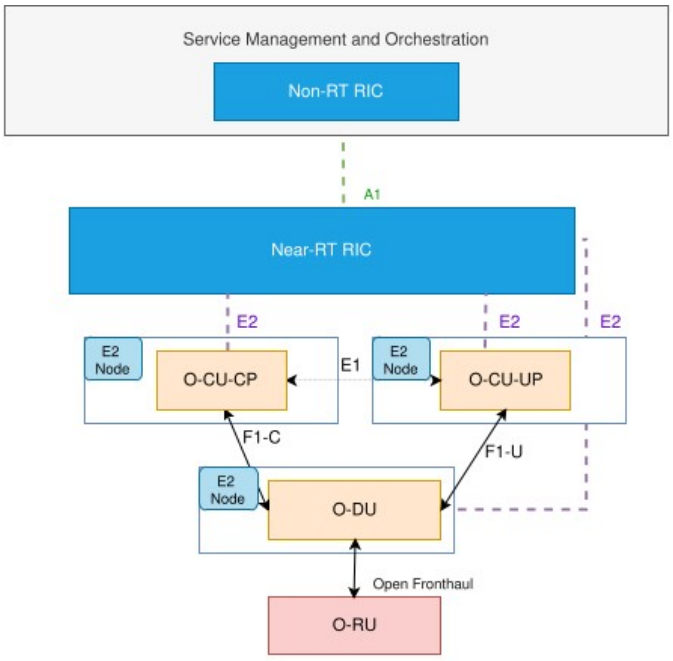


Fig. 1: Arquitetura Open RAN [1].

O outro RIC, Near-RT RIC, é um controlador que suporta operações quase em tempo real, na escala de 10ms a 1s. Nele podem ser hospedadas aplicações conhecidas como xApps, para incluir funcionalidades a rede, como: gerenciamento de recursos, gerenciamento de mobilidade e segurança. Essas aplicações podem funcionar com a implementação de IA/ML, usando o controlador para obter métricas da rede e aferir decisões inteligentes que reconfiguram os E2 Node, visando a otimização ou controle de recursos. As aplicações podem utilizar a interface A1 do Near-RT RIC para trocar informações com o Non-RT RIC, mas abaixo utiliza a interface E2, por meio da E2T (*E2 Termination*) e os modelos de serviço para que seja possível interagir com o E2 Node e haja troca de informações com os elementos da RAN. Esses modelos são padronizados pela ORAN Alliance, um exemplo é o E2SM KPM (*E2 Service Model Key Performance Measurements*).

Abaixo dos controladores, como pode ser observado na Figura 1, o componente CU divide-se em um plano de controle e um plano de usuário. Dessa forma, o O-CU-CP corresponde ao plano de controle do PDCP (*Packet Data Convergence Protocol*) e contém RRC (*Radio Resource Control*) responsável pelo controle de recursos da rádio. Ademais, o O-CU-CP corresponde ao plano do usuário do protocolo PDCP e o SDSP (*Service Data Adaptation Protocol*) responsável pela gerência do QoS dos fluxos de tráfego. O componente DU, cuida das funcionalidades referentes a camada física e das camadas de MAC (*Medium Access Control*) e RLC (*Radio Link Control*). Por último, o RU que é o componente responsável por hospedar a camada Low-PHY e processamento de RF com base em uma divisão funcional da camada inferior.

## III. Trabalhos Relacionados

Ao revisar a bibliografia atual sobre Open RAN estritamente relacionada ao tema de coleta de dados de indicadores de desempenho em redes Open RAN, é possível encontrar um número reduzido de artigos que abordam principalmente este tema. Nesta seção há a apresentação dos artigos considerados mais relevantes ao tema e que corroboraram com o desenvolvimento deste projeto.

Os autores Bonati et al. [5] propõe um *framework* chamado OpenRAN Gym, essa solução é aberta suportando (i) coleta de dados, (ii) design, (iii) prototipagem e (iv) teste de soluções de controle orientadas a dados para sistemas O-RAN de próxima geração. As ferramentas apresentadas possuem vários módulos de *software* para coleta de métricas de RAN, e um RIC adaptado para execução em plataformas sem fio experimentais. Este trabalho demonstra como coletar dados, projetar, treinar e testar xApps que usam de IA e ML em escala.

Villa et al. [6] apresentam o X5G, um ambiente de teste de rede 5G privado implantado na Universidade Northeastern em Boston. O X5G é aberto, programável e integra componentes de vários fornecedores. Essa implementação é baseada em uma combinação de componentes programáveis de código aberto desde a camada PHY até a rede *core* (CN). A plataforma, denominada de NVIDIA Aerial RAN CoLab (ARC), é implantada em uma infraestrutura dedicada de vários fornecedores com 8 servidores para CU e DU, 4 RUs que podem ser instaladas em um espaço de laboratório, *fronthaul* O-RAN 7.2 e *hardware* de temporização e um CN 5G dedicado.

Os autores Kouchaki e Marojevic [7] discutem o desenvolvimento ponta-a-ponta de uma xApp baseada em ML. Nesse processo são citados dois passos principais: (i) criar as funcionalidades básicas seguindo os requisitos do RIC e (ii) fazer e implantar a aplicação em um contêiner. Para o primeiro é possível utilizar *frameworks* como o ricxappframe 3.2.0 fornecido pelo PyPi, que engloba as funcionalidades básicas, como a comunicação com o *RIC Message Router*. O segundo passo engloba o desenvolvimento e implantação da

aplicação. Destaca-se ainda a necessidade de uma xApp que faça a coleta e processamento das métricas da rede, utilizadas pelos modelos de ML. Por fim o trabalho aborda em detalhes os modelos utilizados e a implantação da xApp.

O trabalho proposto pelos autores Polese et al. [8] abordam a implementação de DRL (D*eep Reinforcement Learning*) para controle de malha fechada visando à melhoria do QoS em fatias de rede eMBB, URLLC e MTC. Para isso, o trabalho destaca a importância na coleta de métricas significativas para representar bem o estado da rede. Além disso, o trabalho explora o *design* de xApps baseadas em DRL para controle em malha fechada em O-RAN, incluindo testes em uma plataforma experimental em grande escala, o Colosseum. Por fim, apresentam possíveis aplicações futuras e avanços possíveis pela adoção da DRL e a importância da seleção das métricas e do desenvolvimento dos agentes capazes de lidar com as variedades de configurações.

## IV. Framework de Coleta de KPMs

A quantidade crescente de dispositivos conectados à rede gera um volume massivo de dados de desempenho. Cada UE pode estar envolvida em diferentes tipos de tráfego, desde chamadas de voz e *streaming* de vídeo até comunicações de IoT (*Internet of Things*), por consequência, diferentes tipos de tráfego possuem requisitos distintos de qualidade de serviço (QoS). É com base nos dados coletados em diferentes cenários que o RIC pode realizar, em tempo real, o gerenciamento e o controle inteligente de RAN, utilizando esses dados para apoiar nas tomadas de decisão. A arquitetura desagregada de redes Open RAN adiciona uma camada de complexidade na coleta de métricas, integrar e correlacionar dados de desempenho de várias fontes em tempo real exige um *framework* sofisticado e bem coordenado.

Nesse sentido, a solução consiste no desenvolvimento de uma aplicação com *framework* para coleta, processamento e armazenamento de métricas de desempenho em um cenário de redes Open RAN. A aplicação foi desenvolvida na linguagem C++ utilizando um SDK e modelos de serviços disponibilizados pelo FlexRIC [9]. Além disso, o *framework* implantado por meio de uma xApp para rodar no Near-RT RIC fornecido pela solução, sendo assim capaz de realizar a coleta e o processamento das métricas em tempo real a cada ciclo do RIC.

A proposta se aproveita da arquitetura desagregada da gNodeB para permitir uma coleta mais granular e precisa das métricas. Além disso, a integração entre as métricas fornecidas pelas unidades da gNodeB (CU, DU e RU) possibilita uma visão unificada e detalhada do desempenho das UEs. O *framework* desenvolvido permite a coleta desses dados em tempo real e é compatível com diferentes implementações da gNodeB, como a solução fornecida pela OpenAirInterface (OAI) ou pelo software srsRAN, considerando UEs que podem ter diferentes tipos de tráfego e mobilidade, conforme os princípios do Open RAN.

Algumas KPMs que podem ser coletadas incluem: taxa de transferência no *downlink* e *uplink*, taxa de sucesso dos pacotes, taxa de perda de pacotes no *downlink*, quantidade de PRBs (*Physical Resource Blocks*) fornecidos, entre outras. No fim, após a fase de processamento a aplicação realiza o armazenamento dessas métricas utilizando como mecanismo de persistência um banco de dados SQLite3, como demonstrado na Figura 2, a qual também apresenta o fluxo de dados desde a coleta de métricas dos fluxos eMBB, mMTC e URLLC, pela xApp até o armazenamento e finalizando com a exposição dessas métricas para diferentes casos de uso.

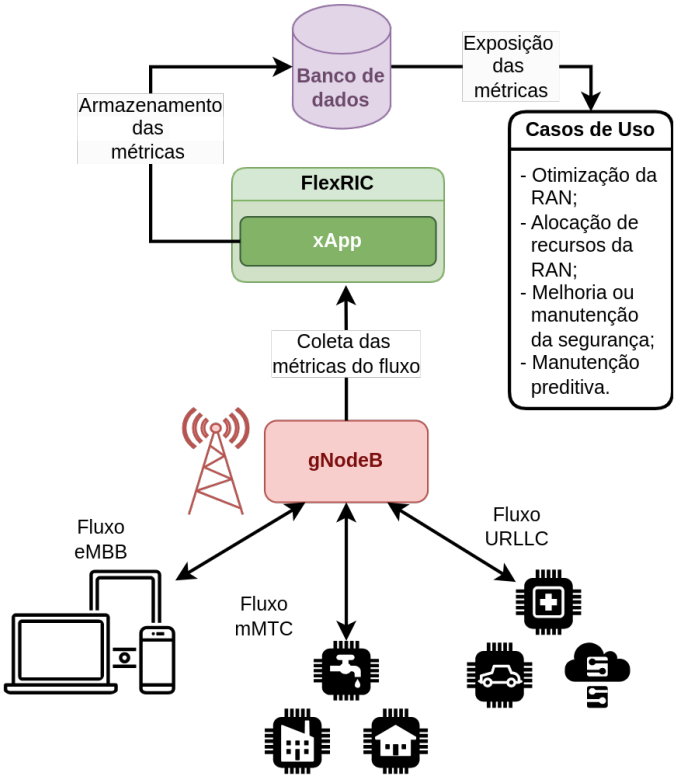


Fig. 2: Lógica de funcionamento da xApp.

## V. Ambiente de Experimentação

Para a elaboração da xApp foi desenvolvido um cenário de experimentação com as especificações O-RAN. Desse modo, foram implementados de forma segmentada os componentes: núcleo da rede 5G, pilha da RAN e controlador Near-RT RIC em 3 diferentes máquinas virtualizadas, cada uma hospedando um desses componentes. Estas VMs foram hospedadas em única máquina com a seguinte configuração:

- Sistema Operacional: Proxmox 8.2.2
- CPU: Intel(R) Core(TM) i9-12900KS 8 cores e 24 threads cores x86_64 @ 3.4 GHz
- Memória RAM: 64 GB
- SSD: 2 TB

Para chegar ao resultado apresentado na Figura 3, a VM dedicada ao núcleo da rede foi implementada por meio do projeto de código aberto OAI-5G Core Network (CN)[10], compatível com as especificações do 3GPP e que apresenta diversos conjuntos de recursos, como as seguintes Funções de Rede (NF): AMF (*Access and Mobility Management Function*), SMF (*Session Management Function*), UPF (*User Plane Function*), NRF (*Network Repository Function*), AUSF (*Authentication*

*Server Function*), UDM (*Unified Data Management*), UDR (*Unified Data Repository*) e NSSF (*Network Slice Selection Function*). Entretanto, para fim dos testes da xApp, a versão minimalista foi implementada com 4 NFs (AMF, UPF, SMF e NRF), estas funções de rede foram instaladas por meio de um tutorial fornecido pela OAI, onde o Docker Compose é utilizado para iniciar e executar os serviços de um arquivo em YAML, assim, criando e iniciando os contêineres das NFs com suas respectivas configurações de rede.

A pilha da RAN foi hospedada em outra máquina virtual por meio do projeto OAI 5G RAN [11], o qual segue os princípios de compatibilidade com o 3GPP e habilita implementação da gNodeB (gNB) e UE. A gNB suporta uma implementação modo *Non Stand-Alone*, onde gNB 5G se conecta ao núcleo 4G, e o modo *Stand-Alone* onde a infraestrutura é plenamente 5G, este mesmo suporte ocorre na UE da OAI. A instalação destes componentes se deu semelhantemente ao núcleo da rede utilizando versões conteinerizadas dos componentes para instalar as dependências e construir a gNB e UE. No caso do ambiente da Figura 3, foi implementado uma gNB monolítica, ou seja, sem a separação da CU e DU. Após realizar a inicialização da gNB, a mesma se conecta às NFs UPF e AMF, bem como após a inicialização das UEs, elas se conectam a gNodeB. Todo esse ambiente é levantado em Docker e os componentes são isolados em contêineres.

Por fim, a última VM hospedou o controlador Near-RT RIC, o FlexRIC, com o *framework* de coleta de indicadores de chave de desempenho. A instalação do controlador partiu da execução da adaptação de sua implementação e instalação de componentes para um contêiner, possibilitando uma maior agilidade e portabilidade do projeto para diferentes ambientes, nele estão presentes os modelos de serviços que são cruciais, pois servem para a coleta de métricas provindas da gNodeB. Este nó (gNodeB) conecta-se ao controlador habilitando as funções de gerência. Após isso, foi possível construir e inicializar a aplicação que aumenta as funcionalidades do controlador. Todo esse ambiente e as configurações das máquinas, podem ser observadas na Figura 3, bem como o posicionamento dos componentes e as conexões entre eles.

## VI. Resultados Obtidos

Após montar o ambiente de experimentação, ilustrado na Figura 3, foram feitos uma série de testes envolvendo o ambiente e a aplicação. Nestes testes, considerou-se a avaliação de 4 fatores principais, sendo eles: quantidade de UEs, sendo necessário para a verificação da escalabilidade da coleta para a aplicação. Taxas de *downlink* e *uplink*, para verificar a desempenho da infraestrutura com relação à quantidade de UEs suportadas, como também, a checagem do impacto do monitoramento do *framework* no tráfego enviado e recebido pelas UEs. Por fim, a latência de monitoramento, o qual é a medida do tempo que leva para transporte de uma mensagem de reporte de métricas enviada pela gNodeB até a xApp, sendo um fator importante para validar a escalabilidade e o tempo de resposta da aplicação. A média desses foram medidos por um tempo de 10 minutos com granularidade de 1s, sendo os valores obtidos apresentado na Tabela I.

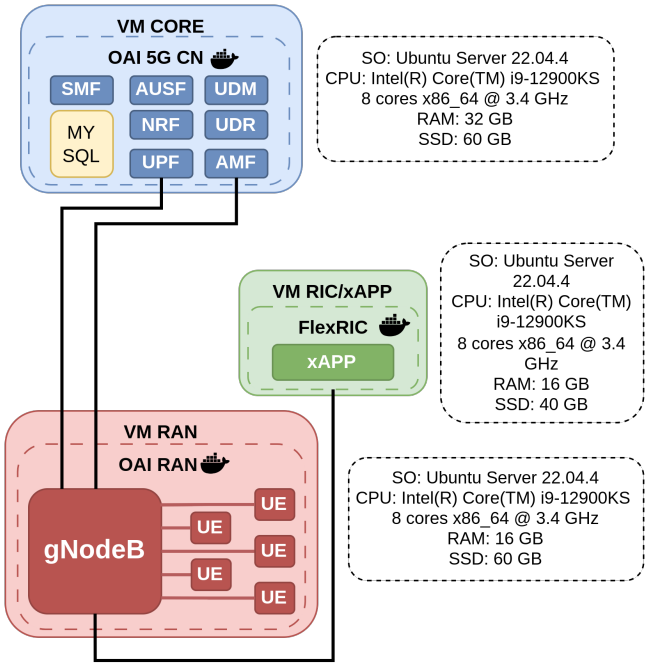


Fig. 3: Ambiente de experimentação implementado em Docker.

TABELA I: Métricas de performance colhidas durante a coleta.

| UEs | Downlink (Mbps) | Uplink (Mbps) | Latência ($\mu$s) |
|---|---|---|---|
| 1 | 195.06 | 105.95 | 2763 |
| 2 | 55.57 | 24.87 | 2868 |
| 4 | 30.76 | 12.65 | 3399 |
| 8 | 10.48 | 3.55 | 2153 |

Como visto nos resultados da Tabela I, a partir do aumento das UEs tanto o *downlink* quanto o *uplink* possuem a tendência de diminuir os valores de Mbps devido à divisão da banda larga e espera-se o mesmo comportamento para quantidades de UEs superiores a oito. Enquanto isso observa-se a tendência de aumento da latência com 1, 2 e 4 UEs, mas a mesma diminuiu com 8 UEs, portanto a latência não apresenta impactos negativos significativos devido aos valores baixos e a pequenas diferença entre eles. Esses resultados demonstraram que o ambiente de experimentação, com núcleo da rede somado a pilha da RAN da OAI e o FlexRIC, criaram um ambiente emulado 5G válido para os testes de validação da aplicação com o *framework*.

Nos casos dos testes de validação do *framework*, enquanto foram gerados os tráfegos para validação do ambiente de experimentação, a Figura 4 apresenta as métricas obtidas nos *logs* da aplicação, onde pode ser identificado o número de identificação da UE, para *downlink* e *uplink*: volume de dados chegando na camada de protocolo PDCP, a taxa de vazão e o PRB. Portanto, a aplicação pôde ser validada durante os testes de aumento de UEs, onde os valores referentes a cada UE foram adequados ao ambiente construído.

```
[kpmManager.cpp:133] UE ID type = gNB | amf_ue_ngap_id = 1
[kpmManager.cpp:138] ran_ue_id = 1
[kpmManager.cpp:73] DRB.PdcpSduVolumeDL = 0 [Mbits]
[kpmManager.cpp:77] DRB.PdcpSduVolumeUL = 73150 [Mbits]
[kpmManager.cpp:109] DRB.RlcSduDelayDl = 0.00 [µs]
[kpmManager.cpp:101] DRB.UEThpDl = 0.00 [Mbps]
[kpmManager.cpp:105] DRB.UEThpUl = 73.54 [Mbps]
[kpmManager.cpp:65] DRB.PrbTotDl = 2465 [PRBs]
[kpmManager.cpp:69] DRB.PrbTotUl = 162294 [PRBs]
[kpmManager.cpp:133] UE ID type = gNB | amf_ue_ngap_id = 2
[kpmManager.cpp:138] ran_ue_id = 2
[kpmManager.cpp:73] DRB.PdcpSduVolumeDL = 0 [Mbits]
[kpmManager.cpp:77] DRB.PdcpSduVolumeUL = 170652 [Mbits]
[kpmManager.cpp:109] DRB.RlcSduDelayDl = 0.00 [µs]
[kpmManager.cpp:101] DRB.UEThpDl = 0.01 [Mbps]
[kpmManager.cpp:105] DRB.UEThpUl = 173.92 [Mbps]
[kpmManager.cpp:65] DRB.PrbTotDl = 3315 [PRBs]
[kpmManager.cpp:69] DRB.PrbTotUl = 360291 [PRBs]
[kpmManager.cpp:133] UE ID type = gNB | amf_ue_ngap_id = 3
[kpmManager.cpp:138] ran_ue_id = 3
[kpmManager.cpp:73] DRB.PdcpSduVolumeDL = 0 [Mbits]
[kpmManager.cpp:77] DRB.PdcpSduVolumeUL = 18455 [Mbits]
[kpmManager.cpp:109] DRB.RlcSduDelayDl = 0.00 [µs]
[kpmManager.cpp:101] DRB.UEThpDl = 0.00 [Mbps]
[kpmManager.cpp:105] DRB.UEThpUl = 18.56 [Mbps]
[kpmManager.cpp:65] DRB.PrbTotDl = 1965 [PRBs]
[kpmManager.cpp:69] DRB.PrbTotUl = 50863 [PRBs]
[kpmManager.cpp:133] UE ID type = gNB | amf_ue_ngap_id = 4
[kpmManager.cpp:138] ran_ue_id = 4
[kpmManager.cpp:73] DRB.PdcpSduVolumeDL = 0 [Mbits]
[kpmManager.cpp:77] DRB.PdcpSduVolumeUL = 41440 [Mbits]
[kpmManager.cpp:109] DRB.RlcSduDelayDl = 0.00 [µs]
[kpmManager.cpp:101] DRB.UEThpDl = 0.00 [Mbps]
[kpmManager.cpp:105] DRB.UEThpUl = 41.67 [Mbps]
[kpmManager.cpp:65] DRB.PrbTotDl = 2175 [PRBs]
```

Fig. 4: Saída das métricas nos *logs* da aplicação.

## VII. Conclusões e Trabalhos Futuros

No cenário atual de Open RAN, as pesquisas tem focado no desenvolvimento de soluções que gerenciem e otimizem as redes O-RAN. Estas redes contam com uma arquitetura flexível e desagregada que facilitam a introdução de aplicações nos RIC para criação de novas soluções. Sendo assim, o caso de uso que essa proposta está associada é a coleta, tratamento e armazenamento das métricas das UEs para o uso de ML.

O *framework* de coleta de Indicadores-Chave de Desempenho para Open RAN apresentado neste trabalho demonstrou ser uma ferramenta essencial para a gestão eficiente e inteligente das redes de acesso via rádio. Através da coleta, processamento e armazenamento de KPMs em ambientes Open RAN, foi possível obter percepções valiosas sobre o desempenho da rede, permitindo casos de uso que exijam tomadas de decisão mais informadas e aprimorando a qualidade de serviço oferecida aos usuários.

Os resultados obtidos durante os testes de validação do *framework* evidenciaram a sua capacidade de lidar com diferentes cenários e quantidades de UEs, além de demonstrar a importância da coleta e análise adequada dos Indicadores-Chave de Desempenho para garantir a eficiência operacional e a escalabilidade das redes Open RAN.

Como trabalhos futuros, estima-se a utilização do *framework* para o treinamento de modelos de aprendizado de máquina para a realização de ações para a melhoria de QoS na rede O-RAN, bem como, melhor distribuição de recursos (e.g. PRBs) para cada UE. Além disso, estima-se o teste de outros tipos de banco de dados, para diminuição na latência da aplicação, bem como, o desenvolvimento de outros módulos extras dentro da xApp para lidar com *network slicing*.

## Agradecimentos

Este trabalho foi financiado com recursos da RNP - Rede Nacional de Ensino e Pesquisa, sendo este trabalho parte do GT-IQoS do programa OpenRAN@Brasil.

## Referências

[1] S. Marinova and A. Leon-Garcia, "Intelligent o-ran beyond 5g: Architecture, use cases, challenges, and opportunities," *IEEE Access*, vol. 12, pp. 27 088–27 114, 2024.

[2] M. K. Bahare, A. Gavras, M. Gramaglia, J. Cosmas, X. Li, Bulakci, A. Rahman, A. Kostopoulos, A. Mesodiakaki, D. Tsolkas, M. Ericson, M. Boldi, M. Uusitalo, M. Ghoraishi, and P. Rugeland, "The 6G Architecture Landscape - European perspective," Feb. 2023. [Online]. Available: https://doi.org/10.5281/zenodo.7313232

[3] M. Polese, L. Bonati, S. D'Oro, S. Basagni, and T. Melodia, "Understanding o-ran: Architecture, interfaces, algorithms, security, and research challenges," *IEEE Communications Surveys Tutorials*, vol. 25, no. 2, pp. 1376–1411, 2023.

[4] ORAN Alliance, "O-ran architecture overview — oran master documentation," https://docs.o-ran-sc.org/en/latest/architecture/architecture.html.

[5] L. Bonati, M. Polese, S. D'Oro, S. Basagni, and T. Melodia, "Openran gym: Ai/ml development, data collection, and testing for o-ran on pawr platforms," *Computer Networks*, vol. 220, p. 109502, 2023.

[6] D. Villa, I. Khan, F. Kaltenberger, N. Hedberg, R. S. da Silva, A. Kelkar, C. Dick, S. Basagni, J. M. Jornet, T. Melodia *et al.*, "An open, programmable, multi-vendor 5g o-ran testbed with nvidia arc and openairinterface," *arXiv preprint arXiv:2310.17062*, 2023.

[7] M. Kouchaki and V. Marojevic, "Actor-critic network for o-ran resource allocation: xapp design, deployment, and analysis," in *2022 IEEE Globecom Workshops (GC Wkshps)*, 2022, pp. 968–973.

[8] M. Polese, L. Bonati, S. D'Oro, S. Basagni, and T. Melodia, "Colo-ran: Developing machine learning-based xapps for open RAN closed-loop control on programmable experimental platforms," *CoRR*, vol. abs/2112.09559, 2021. [Online]. Available: https://arxiv.org/abs/2112.09559

[9] R. Schmidt, M. Irazabal, and N. Nikaein, "Flexric: an sdk for next-generation sd-rans," in *Proceedings of the 17th International Conference on Emerging Networking EXperiments and Technologies*, ser. CoNEXT '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 411–425. [Online]. Available: https://doi.org/10.1145/3485983.3494870

[10] OpenAirInterface, "OpenAirInterface 5G Core Network Project," 2024, acessado em: 2024-06-04. [Online]. Available: https://openairinterface.org/oai-5g-core-network-project/

[11] ——, "OpenAirInterface 5G Radio Access Network Project," 2024, acessado em: 2024-06-04. [Online]. Available: https://openairinterface.org/oai-5g-ran-project/