Brasil 6G

Projeto Brasil 6G
Fase III

Atividade 4.3 – Avaliação de desempenho da rede em condições reais - Parte 1

18/07/2025

Histórico de Atualizações:

Versão	Data	Autor(es)	Notas
1	30/06/2025	Anderson Reis Rufino Marins Juliano Silveira Ferreira Luiz Gustavo Barros Guedes	Elaboração de conteúdo
2	18/07/2025	Luciano Leonel Mendes Luiz Gustavo Barros Guedes	Revisão de texto

Lista de Figuras

Lista de Tabelas

Acrônimos

6G Sexta Geração de Rede Móvel Celular

AD Analógico-Digital

ACM *Adaptive Coding and Modulation*

AGC *Automatic Gain Control*

BER *Bit Error Rate*

BS *Base Station*

CFM *Cubic Feet per Minute*

CPU *Central Process Unit*

DA Digital-Analógico

GFDM *Generalized Frequency Division Multiplexing*

GPS *Global Positioning System*

GPU *Graphics Processing Unit*

IP *Internet Protocol*

LNA *Low Noise Amplifier*

MAC *Media Access Control*

MIMO *Multiple-Input and Multiple-Output*

MCS *Modulation and Coding Scheme*

NVMe *Non-Volatile Memory Express*

PA *Power Amplifier*

PCIe *Peripheral Component Interconnect Express*

PHY *Physical Layer*

RF *Radio Frequency*

SATA *Serial Advanced Technology Attachment*

SDR *Software Defined Radio*

SNR *Signal-to-Noise Ratio*

SSDs *Solid-State Drivers*

UE *User Equipment*

Sumário

1 Introdução

As discussões em torno da próxima geração de redes de comunicação móvel, o Sexta Geração de Rede Móvel Celular (6G), abrangem uma ampla gama de cenários de uso e aplicações emergentes, muitas das quais exigirão avanços significativos em termos de conectividade, latência, cobertura e capacidade computacional. Nesse contexto, o projeto Brasil 6G busca contribuir de forma concreta para o desenvolvimento dessa nova geração de redes, com ênfase especial em soluções adaptadas às necessidades do Brasil e de outras nações que enfrentam desafios semelhantes, especialmente no que diz respeito à falta de conectividade em áreas remotas e rurais.

A concepção e o desenvolvimento de uma rede de acesso voltada para tais contextos impõem uma série de desafios técnicos. Entre eles, destacam-se a criação e a integração de algoritmos de processamento digital de sinais, sua adaptação para execução em tempo real ou em sincronia com elementos de *hardware*, além da montagem de protótipos funcionais por meio da integração de módulos e dispositivos.

Este relatório apresenta os resultados obtidos no âmbito da atividade 4.3 da Fase III do projeto Brasil 6G, cujo objetivo principal é a avaliação do desempenho da rede de acesso sob condições operacionais realistas. Inicialmente, foram realizadas otimizações nos algoritmos base do transceptor, seguidas de testes e ajustes conduzidos em ambiente laboratorial. Para tornar esses testes mais representativos das condições de operação em campo, foram montados protótipos de transceptores que permitiram a execução de experimentos sob diferentes cenários e ambientes.

Como continuidade desse trabalho, estão previstas atividades de testes complementares em condições de campo, as quais visam identificar possíveis necessidades de melhorias adicionais e, assim, promover o avanço na maturação da solução tecnológica.

Com o intuito de organizar e apresentar os resultados de forma clara e estruturada, este relatório está dividido da seguinte forma: a Seção 2 descreve o processo de montagem dos protótipos utilizados para os testes laboratoriais; a Seção 3 detalha as otimizações aplicadas e os principais testes realizados, incluindo análises de desempenho e estabilidade; por fim, a Seção 4 apresenta as considerações finais e as conclusões extraídas a partir das atividades conduzidas.

2 Montagens para realização de testes laboratoriais

Conforme mencionado, a execução da atividade 4.3 do projeto Brasil 6G contempla a realização de testes de algoritmos da rede de acesso em condições realistas. Para viabilizar a realização de tais testes, foi realizada a montagem de novos protótipos dos transceptores que compõe a referida rede de acesso. A montagem dos protótipos envolveu a integração dos dispositivos de *hardware* que foram especificados na fase II do projeto Brasil 6G conforme detalhado em [1]. Vale destacar que foi adotado para a implementação do transceptor o conceito de *Software Defined Radio* (SDR), fazendo uso de computadores de uso geral associado a *front-ends* de RF [2]. O diagrama de *hardware* do transceptor Brasil 6G está apresentado na Figura 1.

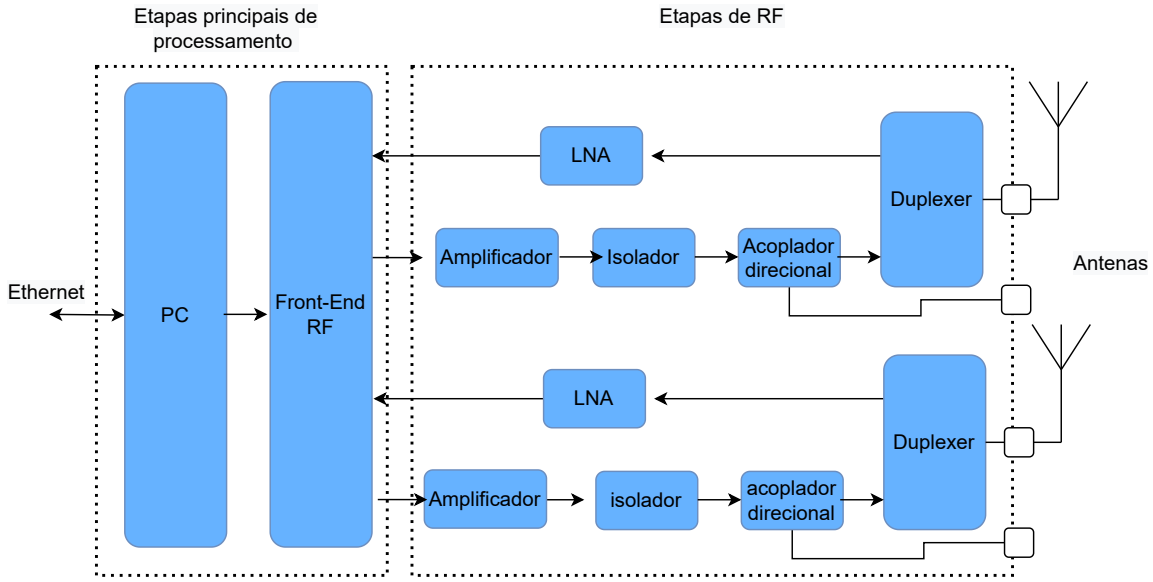


Figura 1: Diagrama macro de *hardware* do transceptor Brasil 6G.

A arquitetura de *hardware* do transceptor, apresentada na Figura 1, é formada por duas etapas: etapa de processamentos principais e etapas de RF. A seguir uma breve descrição dos principais dispositivos que compõe cada etapa mencionada, assim como sua função delas no transceptor:

- O PC é um computador de uso geral utilizado para a execução dos algoritmos de processamento de sinais da rede de acesso, que incluem gerenciamento dos dados do usuário, alocação de recursos de transmissão, modulação e demodulação digital, codificação e decodificação de canal, estimação e equalização de canal, sincronismo, dentre outros;
- O Front-end de RF integra os conversores Digital-Analógico (DA) e Analógico-Digital (AD), assim como os dispositivos de RF para conversão do sinal de banda base para canal e vice-versa;
- O amplificador de RF ou *Power Amplifier* (PA) é o dispositivo responsável por amplificar o sinal de RF gerado pelo front-end de RF;

- O isolador é um dispositivo de proteção que permite o acoplamento de sinal em uma única direção de propagação;
- O acoplador direcional viabiliza a realização de medidas de amostras do sinal de transmissão sem ser necessário desligar as antenas;
- O amplificador de baixo ruído ou *Low Noise Amplifier* (LNA) é responsável por amplificar o sinal de recepção;
- O Duplexer viabiliza a transmissão e recepção de sinais de maneira simultânea em uma mesma antena, uma vez, que atua como um filtro de separação dos sinais da banda de *uplink* e de *downlink*;
- Antenas: responsáveis por viabilizar a irradiação do sinal de RF, destacando que o sistema prevê a operação com duas antenas que operam simulâneamente para a transmissão e recepção de sinais, ou seja, com esquema *Multiple-Input and Multiple-Output* (MIMO) 2 x 2.

A Figura 2 apresenta uma foto do protótipo do transceptor montado, na qual é possível visualizar os principais dispositivos de *hardware* que o compõem, conforme descrito anteriormente. Estão destacados na figura o computador, o *front-end* de RF e o módulo de RF.

Figura 2: Protótipo montado do transceptor.

A Figura 3 mostra na parte a) um dos set-ups utilizados para a realização de testes via cabo de RF, enquanto a parte b) mostra um dos set-ups de testes utilizando transmissão

via antenas. Nas duas partes é possível observar dois transceptores ou terminais, que são os terminais *Base Station* (BS) e um terminal *User Equipment* (UE) que estão em avaliação. O set-up de transmissão via cabo foi utilizado para avaliações inicias dos algoritmos base do transceptor, de forma de ruídos e interferências externas pudessem ser desconsideradas. Este cenário é importante para a avaliação e validação inicial de certos algoritmos. Embora não haja influência de interferências externas, a comunicação estabelecida entre os terminais em teste é via RF e etapas como aquelas associadas ao sincronismo, por exemplo, são consideradas nos testes, bem como os efeitos da conversão de sinais de analógico para digital e vice-versa. O set-up de transmissão via antena, por sua vez, permite a realização de testes com um caráter ainda mais realista, uma vez que o sistema também recebe interferências e ruídos presentes no espectro e não há uma conexão física entre os transceptores.

(a) Testes via cabo.

(b) Testes via antena.

Figura 3: Diferentes set-ups de avaliação dos transceptores.

Vale destacar que, com relação aos algoritmos de processamento digital de sinais que compõem o transceptor Brasil 6G, que os mesmos foram implementados utilizando o *framework* GNU Radio [3], que é uma ferramenta *open-source* de desenvolvimento de soluções de radiocomunicação e rádio definido por *software*. Os testes realizados permitiram avaliar tais algoritmos e definir as otimizações necessárias para os mesmos.

Os set-ups apresentados foram utilizados para a realização de diferentes testes e avaliações, assim como otimizações de hardware e de software, conforme melhor detalhado ao longo deste relatório.

3 Testes laboratoriais

O desenvolvimento dos transceptores da rede de acesso Brasil 6G teve como ponto de partida os algoritmos criados em fases anteriores. Visando a maturação da solução proposta e a conquista de maior estabilidade de operação, foram realizadas otimizações em tais algoritmos, incluindo correções de *bugs*, ajustes relacionados à inicialização do sistema, além de utras otimizações otimizações. Tais ações foram fundamentais para consolidar o comportamento robusto dos protótipos ao longo do tempo, permitindo a obtenção de desempenho confiável no ambiente de teste.

A seguir, descrevem-se as principais otimizações aplicadas aos algoritmos base da rede de acesso:

- **Melhorias na estabilidade de operação e no processo de inicialização:** realizaram-se ajustes estruturais no fluxo de inicialização do sistema, com o intuito de minimizar condições de falha na inicialização dos transceptores e alcançar maior previsibilidade no estado inicial dos módulos de *software* e *hardware*;
- **Otimização da alocação de memória:** reformulações no gerenciamento de memória permitiram estabilizar a quantidade de memória alocada ao longo do tempo de execução, evitando flutuações ou vazamentos que poderiam comprometer o desempenho em testes prolongados;
- **Aprimoramento da sincronização do terminal do usuário:** foram implementados ajustes nos algoritmos de detecção e sincronização da UE utilizando como referência o sinal de preâmbulo transmitido pela BS, aumentando a confiabilidade do processo de sincronismo temporal;
- **Refinamento do valor de MCS:** incorporaram-se melhorias no algoritmo de estimativa da relação sinal-ruído ou *Signal-to-Noise Ratio* (SNR), com definição mais precisa das faixas de valores correspondentes a cada esquema de modulação e codificação ou MCS, contribuindo para uma seleção mais adequada e adaptativa do enlace de comunicação;
- **Melhorias no sistema de arrefecimento dos protótipos:** com base nas observações de instabilidades térmicas durante os testes, realizaram-se intervenções no sistema de resfriamento, incluindo a substituição dos ventiladores originais, a instalação de dissipadores de calor adicionais sobre o processador, sobre os dispositivos de armazenamento não-volátil e no próprio gabinete dos equipamentos. Essas medidas contribuíram significativamente para a mitigação do aquecimento excessivo, especialmente durante ciclos intensivos de operação.

Na sequência, são apresentados os principais testes laboratoriais conduzidos durante a presente etapa do projeto, com ênfase nos métodos, parâmetros avaliados e conclusões extraídas a partir dos dados coletados.

3.1 Criação de logs para análises de operação

Para o devido monitoramento das operações dos transceptores da rede de acesso Brasil 6G, com o objetivo de identificar pontos de melhoria e garantir condições de estabilidade na operação dos sistemas, desenvolveram-se *scripts* específicos para explorar ferramentas de análise de

desempenho dos computadores envolvidos nos testes de longa duração. Considerando que tais testes podem demandar uso intensivo e contínuo de recursos do sistema, torna-se imprescindível o acompanhamento sistemático de variáveis críticas, como taxa de transferência de dados provida pelo enlace, temperatura, uso da unidade central de processamento ou *Central Process Unit* (CPU) e memória. Os *scripts* desenvolvidos não apenas viabilizaram o monitoramento em tempo real como também permitiram a correlação temporal entre comportamentos anômalos observados durante os testes e os respectivos estados dos recursos do sistema no instante da ocorrência. Destacam-se, a seguir, três *scripts* fundamentais elaborados para essa finalidade:

- **Análise da taxa de transferência de dados com a ferramenta de *software* `iperf3`:** desenvolveu-se um *script* para execução periódica da ferramenta `iperf3`, amplamente utilizada para avaliação de desempenho em redes de protocolo internet ou *Internet Protocol* (IP), com capacidade de medir a taxa de transferência (*throughput*) entre um cliente e um servidor. O `iperf3` fornece métricas essenciais, como largura de banda efetiva, *jitter* e perda de pacotes, sendo indispensável para a caracterização do enlace de comunicação e identificação de gargalos. No entanto, observou-se que sua execução contínua resultava em algumas instabilidades, uma vez que a ferramenta não foi projetada para uso ininterrupto por longos períodos. A fim de contornar essa limitação, o *script* foi ajustado para realizar execuções com duração de 1 hora, seguidas de pausas de 10 segundos, reiniciando o processo de forma cíclica por meio de um laço de repetição. Essa abordagem aumentou significativamente a robustez dos testes de longa duração, permitindo identificar precisamente os instantes e intervalos associados à degradação ou interrupção da taxa de transferência, fornecendo assim uma base sólida para análise cruzada com os demais parâmetros do sistema.

- **Monitoramento de temperatura com a a ferramenta de *software* `sensors`:** visando analisar o comportamento térmico dos sistemas sob teste, desenvolveu-se um *script* para coleta e arquivamento periódico dos dados fornecidos pela ferramenta `sensors`, integrante do pacote `lm-sensors`. Essa ferramenta permite o monitoramento de temperaturas de diversos componentes, especialmente CPU e unidade de processamento gráfico ou *Graphics Processing Unit* (GPU), exibindo informações como temperatura atual, limiar (*high*) e valor crítico (*critical*). A análise desses parâmetros é crucial, visto que temperaturas elevadas podem induzir falhas operacionais, redução de desempenho e riscos físicos ao *hardware*. Em logs iniciais, observou-se que uma ou mais unidades da CPU ultrapassaram o limiar térmico, acionando automaticamente mecanismos de proteção, como a redução da frequência de operação (CPU *throttling*). Embora a ultrapassagem do limiar não implique risco imediato, ela indica operação fora da zona recomendada, sendo desejável o aprimoramento do sistema de resfriamento. Ultrapassagens do valor crítico, por sua vez, representam riscos reais de dano físico ao sistema. Constatou-se que picos térmicos estavam diretamente associados à queda abrupta de desempenho, como a zeragem da taxa de transferência capturada pelo `iperf3`. Para reforçar essa análise, foi utilizado o comando `dmesg`, que acessa o *buffer* de mensagens do *kernel*, revelando alertas e eventos de limitação térmica emitidos pelo sistema. Essas informações foram fundamentais para identificar e compreender os episódios de *thermal throttling*. Como resposta, adotaram-se medidas como verificação do funcionamento das ventoinhas, otimização do fluxo de ar no gabinete, ajustes na BIOS/UEFI para controle térmico, e instalação de ventoinhas com maior fluxo de ar ou *Cubic Feet per Minute* (CFM). A integração dos *scripts* permitiu

identificar com precisão a coincidência temporal entre falhas operacionais e anomalias térmicas, consolidando uma metodologia robusta para análise e prevenção de instabilidades.

- **Monitoramento do uso de recursos com a ferramenta de *software* `top`:** implementou-se um *script* para a execução periódica da ferramenta `top`, que fornece uma visão abrangente e em tempo real do uso dos principais recursos do sistema operacional, como carga da CPU, uso de memória física e de *swap*, e atividades dos processos em execução. Essa ferramenta mostrou-se particularmente útil para detectar padrões de degradação de desempenho, vazamentos de memória e sobrecarga de processos ao longo de experimentos de longa duração. O *script* foi configurado para registrar automaticamente os dados do `top` a cada 5 minutos, em uma janela temporal inicial de 12 horas, posteriormente estendida conforme a ampliação da duração dos testes. O intervalo de 5 minutos representa um compromisso entre resolução temporal e volume de dados, permitindo identificar tendências progressivas no uso de recursos sem sobrecarregar o armazenamento. Os registros permitiram observar comportamentos como o aumento gradual no consumo de memória, estabilizações ou flutuações repentinas, os quais foram correlacionados com eventos identificados por outras ferramentas, contribuindo para uma análise mais precisa das causas de instabilidades.

A Figura 4 apresenta a execução de cada uma das ferramentas supracitadas, empregadas nos *scripts* durante um teste prático com o transceptor. As janelas numeradas como 1, 2 e 3 correspondem, respectivamente, às ferramentas `iperf3` (monitoramento do tráfego de rede), `sensors` (verificação da temperatura dos componentes) e `top` (acompanhamento do uso da CPU em tempo real).

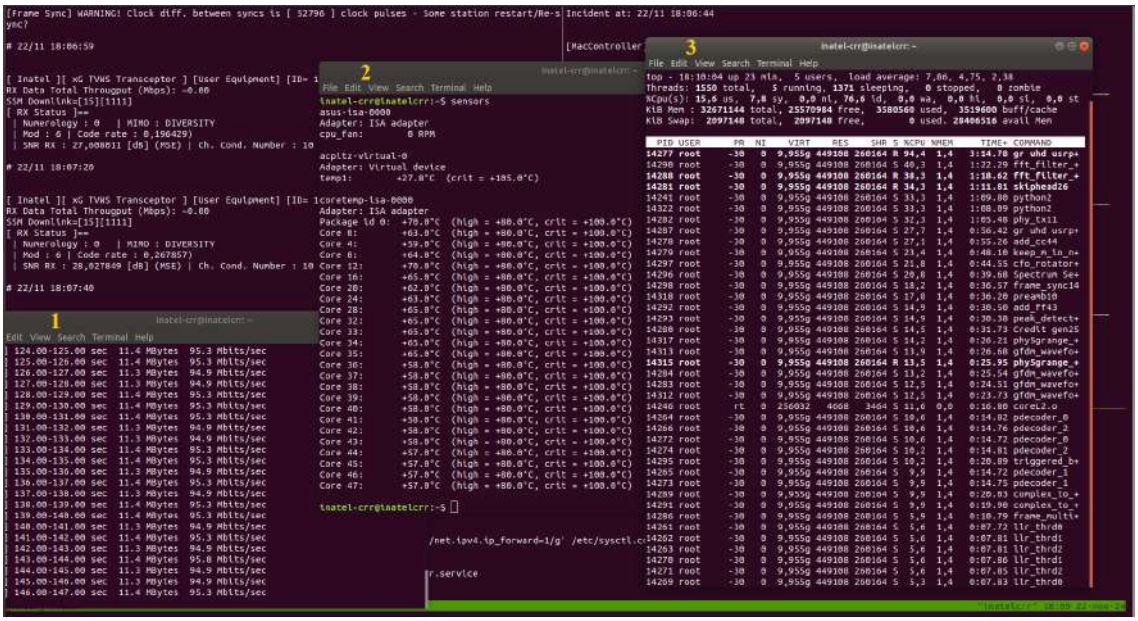


Figura 4: Monitoramento de tráfego de rede em tempo real, temperatura e uso de CPU durante testes com o transceptor.

3.2 Análise de Desempenho de Rede e Comportamento Térmico dos Processadores

Durante a execução de testes de desempenho de rede com a ferramenta de *software* `iperf3`, foram observadas anomalias na taxa de comunicação entre o terminal BS e um terminal UE, incluindo quedas bruscas e momentos em que não houve comunicação. Dado o comportamento atípico, levantou-se a hipótese de que problemas de origem térmica nos processadores pudessem estar interferindo no desempenho do sistema. Essa suposição foi motivada por observações prévias dos resultados dos *scripts*, que apontavam temperaturas elevadas nos processadores das máquinas envolvidas no sistema.

Para investigar a hipótese de aquecimento excessivo dos componentes e seus impactos, foram utilizados os seguintes comandos no terminal:

- `sudo dmesg -T | grep -i "thermal"`: para identificar mensagens relacionadas a eventos térmicos registrados pelo *kernel*;
- `sudo dmesg -T | grep -i "throttle"`: para verificar se ocorreu *thermal throttling*, ou seja, limitação do desempenho do processador devido ao superaquecimento.

Além disso, ao final de cada execução dos testes com `iperf3`, foram coletadas as seguintes informações para posterior análise:

- Por meio do *script*, verificou-se a saída do comando `sensors` para aferição direta da temperatura dos núcleos;
- Registro completo, desde o último *boot*, do log do *kernel* salvo em um arquivo de texto com o comando: `dmesg -T » dmesg_log.txt`

Destaca-se que essas coletas foram realizadas em ambas as máquinas participantes do teste de comunicação (cliente: UE 1; e o servidor: BS), garantindo que o diagnóstico abrangesse tanto o transmissor quanto o receptor dos dados.

Dispondo os dados coletados em uma tabela, realizou-se um cruzamento entre as informações obtidas (temperatura, mensagem de log, taxa de transmissão e eventos de sincronismo) com foco em três comportamentos principais:

1. Início da queda da taxa de transmissão;
2. Momentos de taxa significativamente baixa;
3. Interrupção da comunicação.

Durante essa correlação, buscou-se, sobretudo, entender os seguintes pontos: o que ocorre com a temperatura quando a taxa de transmissão atinge valores mínimos ou zero; quais mensagens aparecem nos subsistemas de camada física ou *Physical Layer* (PHY) e de controle de acesso ao meio ou *Media Access Control* (MAC) no `dmesg` nesses instantes; se a taxa de transmissão retorna após zerar ou permanece inoperante; a existência de quedas intermediárias e seus possíveis gatilhos térmicos ou de rede.

Em diversos testes, constatou-se que o aumento da temperatura precedia eventos de *throttling*, conforme mensagens explícitas no `dmesg`. Quando o *throttling* era ativado, observou-se aumento na latência, redução na taxa de transferência de dados e eventos de sincronismo interrompido, especialmente visíveis na camada PHY. Em alguns momentos, o log do MAC

evidenciou falhas na negociação de parâmetros, o que coincidiu não apenas com *retrieves*, associados às tentativas de retransmissão do sistema, como também às mais frequentes e acentuadas quedas de taxa. A frequência de *retrieves* aumentava significativamente quando a taxa caía, sugerindo uma possível relação causal: ou o aumento nos *retrieves* sobrecarregava o sistema, levando à queda de taxa, ou o problema térmico induzia instabilidades que causavam ambos os sintomas.

A análise combinada dos dados evidencia que os eventos de aquecimento excessivo dos processadores têm impacto direto no desempenho da comunicação de rede, seja por meio do *throttling* ativado automaticamente pelo sistema, seja por efeitos indiretos sobre as camadas física e de enlace da pilha de comunicação. A sincronização instável, aliada a quedas sucessivas de taxa e aumento de retransmissões, aponta para um cenário de degradação progressiva conforme a temperatura se eleva.

Com isso, reforçou-se a necessidade de controle térmico eficaz (melhor ventilação, por exemplo) para tender ao desempenho ideal. Por meio dessas melhorias, houve uma devida estabilidade das temperaturas em níveis aceitáveis e seguros para o sistema. A Figura 5 destaca uma mudança principal de hardware que foi a inserção de um novo *cooler* com refrigeração a ar para melhor resfriamento da CPU, assim como a instalação de um *cooler* de maior CFM para o próprio gabinete do computador.

Figura 5: Novo *cooler* para o processador do sistema.

3.2.1 Substituição do `dmesg` por `journalctl` e monitoramento térmico de SSDs

Dando continuidade à análise de estabilidade durante os testes de desempenho de rede, estendeu-se a investigação para abranger os dispositivos de armazenamento, em especial as unidades de estado sólido ou *Solid-State Drivers* (SSDs) do tipo memória não-volátil expressa ou *Non-Volatile Memory Express* (NVMe), cujas temperaturas operacionais passaram a ser monitoradas. Essa extensão se deu após a constatação de que, além da temperatura da CPU, outros componentes críticos como SSDs podem sofrer degradação de desempenho térmica, afetando, inclusive, processos de leitura e escrita intensivos, comuns durante transferências de dados em alta taxa.

Durante a investigação, optou-se por substituir o uso do comando `dmesg` pelo `journalctl`, mais especificamente por meio de `sudo journalctl -k » journalctl_log.txt`. A motivação para essa mudança inclui:

- Persistência dos logs: Ao contrário do `dmesg`, que mostra apenas os logs do *kernel* desde o último *boot*, o `journalctl` permite acesso aos logs históricos, inclusive através de filtros temporais e critérios de severidade;
- Melhor organização e legibilidade, com possibilidade de busca por mensagens específicas de subsistemas como NVMe, *thermal*, MAC, PHY, entre outros;
- Análise contextualizada de eventos de *hardware* em paralelo com processos de usuário e do sistema.

Com essa abordagem, ampliou-se a capacidade de análise dos testes, facilitando a correlação de eventos ao longo do tempo, uma vez que se aprimorou a profundidade e abrangência dos logs analisados, permitindo um histórico contínuo e filtrável dos eventos do sistema, fator fundamental para investigações com múltiplas fases de teste, seja de curta ou longa duração. Um exemplo bastante específico de aplicação consiste no cenário em que o teste seja interrompido por alguma possível queda de energia e, com isso, um novo *boot* se fizesse necessário. As informações referentes ao teste até o momento de sua interrupção seriam desconsideradas pelo `dmesg` e adequadamente identificadas pelo `journalctl`.

Adicionou-se também o monitoramento da temperatura dos dispositivos de armazenamento por meio das seguintes ferramentas:

- Para SSDs do tipo *Serial Advanced Technology Attachment* (SATA) convencionais, foi utilizado o comando `sensors` em conjunto com a análise de logs;
- Para SSDs NVMe, foi instalado e utilizado o pacote `nvme-cli` que é capaz de fornecer parâmetros de diagnóstico do dispositivo.

Observou-se que os SSDs NVMe operam tipicamente a temperaturas mais elevadas quando comparados aos SSDs SATA, devido à maior densidade de transferência de dados, maior proximidade física com a CPU e uso de barramentos do tipo *Peripheral Component Interconnect Express* (PCIe). Em contextos de carga elevada, ou seja, de grandes transferências via `iperf3`, acesso simultâneo a arquivos, entre outros, os NVMe apresentaram picos térmicos próximos aos limites operacionais recomendados pelo fabricante (geralmente entre 70 e 85°C).

Também foi utilizado o comando `journalctl -k | grep nvme` para extrair do log do *kernel* mensagens específicas relacionadas ao funcionamento e eventuais alertas emitidos pelo subsistema NVMe.

3.3 Testes relacionados à alocação de memória

Durante os testes iniciais de estabilidade de longa duração, identificou-se um comportamento crítico no sistema: falhas associadas ao esgotamento de memória. Devido à complexidade dos algoritmos envolvidos e à alta densidade de operações em execução simultânea, tornou-se um desafio identificar qual módulo específico seria o responsável pelo vazamento de memória. Diante disso, optou-se pelo uso da ferramenta `heaptrack` [4], uma solução especializada para diagnóstico de problemas de uso de memória em aplicações C/C++.

A ferramenta `heaptrack` permite monitorar e registrar, em tempo real, todas as alocações dinâmicas realizadas na *heap*, associando cada requisição de memória às funções e estruturas de código que a originaram. Além de identificar os pontos de alocação excessiva, a ferramenta possibilita mensurar o volume total de memória alocada por função ao longo do tempo, o

que é essencial para detectar *memory leaks*, fragmentação ou crescimento descontrolado de *buffers*. Contudo, durante a análise, observou-se que, embora o `heaptrack` fosse eficiente para apontar funções com alocações frequentes ou volumosas (em termos absolutos), ele não indicava diretamente quais alocações estavam crescendo progressivamente com o tempo, ou seja, não distinguia alocações constantes (e esperadas) daquelas potencialmente associadas a vazamentos.

Para contornar essa limitação, adotou-se uma estratégia iterativa: primeiramente, fez-se necessário identificar as alocações constantes e legítimas, que foram sendo eliminadas da análise. Em seguida, o foco passou a recair sobre as alocações variáveis e suspeitas, que apresentavam comportamento não estacionário ao longo da execução. Com essa abordagem, foi possível isolar um elemento crítico do sistema relacionado à *tags*, que se revelou responsável por uma quantidade significativa de alocações de memória não liberadas adequadamente. A Figura 6 apresenta a interface do `heaptrack`, evidenciando a quantidade de memória alocada (coluna *Allocated*) por diferentes funções, bem como a respectiva quantidade de memória alocada e não desalocada (coluna *Leaked*) por cada uma delas. Destaca-se, de forma significativa, o volume de memória associada a mecanismos relacionados ao processo de propagação de *tags*, os quais estão realçados em amarelo na referida figura.

As *tags* são metadados que podem ser gerados pelos blocos do GNU Radio e que podem ser associados à amostras específicas do fluxo de dados. Elas permitem transmitir informações adicionais junto com os dados, sem alterar o conteúdo principal do fluxo. Este tipo de recursos é útil para sinalizar eventos, delimitar pacotes ou transferir parâmetros dinamicamente durante o processamento do fluxo de dados.

Allocations	Temporary	Peak	Leaked	Allocated	Location
121366626	4363519	873,4 MB	873,5 MB	11,7 GB	gr::buffer::add_item_tag(gr::tag_t const&) in ?? (libgnuradio-runtime.so.3.7.11)
9246588	8269	14,8 kB	15,0 kB	887,7 MB	gr::block_detail::add_item_tag(unsigned int, gr::tag_t const&) in ?? (libgnuradio-runtime.so.3.7.11)
4730878	0	13,8 kB	14,0 kB	454,2 MB	std::vector<long, std::allocator<long> >::~vector() in stl_vector.h:434 (libgnuradio-inatel5g_range-1.0.0git.so.0.0.0)
4730...	0	13,8 kB	14,0 kB	454,2 MB	__base_dtor in tags.h:91 (libgnuradio-inatel5g_range-1.0.0git.so.0.0.0)
473...	0	13,8 kB	14,0 kB	454,2 MB	gr::block::add_item_tag(unsigned int, unsigned long, boost::intrusive_ptr<pmt::pmt_base> const&, boost::intrusive_ptr<pmt::pmt_b...
47...	0	13,8 kB	14,0 kB	454,2 MB	boost::intrusive_ptr<pmt::pmt_base>::~intrusive_ptr() in intrusive_ptr.hpp:98 (libgnuradio-inatel5g_range-1.0.0git.so.0.0.0)
1...	0	5,8 kB	5,8 kB	106,7 MB	general_work in frame_multiplexer_impl.cc:235 (libgnuradio-inatel5g_range-1.0.0git.so.0.0.0)
1...	0	5,8 kB	5,8 kB	106,7 MB	general_work in frame_multiplexer_impl.cc:236 (libgnuradio-inatel5g_range-1.0.0git.so.0.0.0)
1...	0	288 B	384 B	106,7 MB	work in phy5grange_impl.cc:604 (libgnuradio-inatel5g_range-1.0.0git.so.0.0.0)
1...	0	288 B	384 B	106,7 MB	work in phy5grange_impl.cc:605 (libgnuradio-inatel5g_range-1.0.0git.so.0.0.0)
3...	0	288 B	288 B	3,4 MB	general_work in frame_multiplexer_impl.cc:386 (libgnuradio-inatel5g_range-1.0.0git.so.0.0.0)
3...	0	288 B	288 B	3,4 MB	general_work in frame_multiplexer_impl.cc:387 (libgnuradio-inatel5g_range-1.0.0git.so.0.0.0)
3...	0	192 B	192 B	3,4 MB	general_work in frame_multiplexer_impl.cc:372 (libgnuradio-inatel5g_range-1.0.0git.so.0.0.0)
3...	0	192 B	192 B	3,4 MB	general_work in frame_multiplexer_impl.cc:373 (libgnuradio-inatel5g_range-1.0.0git.so.0.0.0)
3...	0	192 B	192 B	3,4 MB	general_work in frame_multiplexer_impl.cc:378 (libgnuradio-inatel5g_range-1.0.0git.so.0.0.0)
3...	0	192 B	192 B	3,4 MB	general_work in frame_multiplexer_impl.cc:379 (libgnuradio-inatel5g_range-1.0.0git.so.0.0.0)
3...	0	192 B	192 B	3,4 MB	general_work in frame_multiplexer_impl.cc:382 (libgnuradio-inatel5g_range-1.0.0git.so.0.0.0)
3...	0	192 B	192 B	3,4 MB	general_work in frame_multiplexer_impl.cc:383 (libgnuradio-inatel5g_range-1.0.0git.so.0.0.0)
4515710	8269	960 B	960 B	433,5 MB	gr::block::add_item_tag(unsigned int, unsigned long, boost::intrusive_ptr<pmt::pmt_base> const&, boost::intrusive_ptr<pmt::pmt_b...
2222...	0	480 B	480 B	213,3 MB	general_work in frame_sync_impl.cc:247 (libgnuradio-inatel5g_range-1.0.0git.so.0.0.0)
2222...	8269	480 B	480 B	213,3 MB	general_work in frame_sync_impl.cc:248 (libgnuradio-inatel5g_range-1.0.0git.so.0.0.0)
35839	0	0 B	0 B	3,4 MB	general_work in frame_sync_impl.cc:215 (libgnuradio-inatel5g_range-1.0.0git.so.0.0.0)
35839	0	0 B	0 B	3,4 MB	general_work in frame_sync_impl.cc:216 (libgnuradio-inatel5g_range-1.0.0git.so.0.0.0)

Figura 6: Evidência de alocação crescente de memória ocasionado pelo mecanismo de propagação de *tags*.

Uma vez removida a geração de *tags* nos módulos de processamento identificados, observou-se uma melhora substancial na estabilidade do sistema, com redução expressiva no uso de memória e estabilização da quantidade de memória alocada. Ao longo da realização dos testes pode-se observar que o consumo de memória alocada para a execução dos algoritmos do transceptor permaneceu estável mesmo com a operação do sistema de maneira ininterrupta ao longo de dias, conforme evidenciado na Secção 3.4. Desta forma, a `heaptrack` pode ser considerada como sendo uma ferramenta essencial no processo de depuração e otimização do código dos transceptores, com relação a análise da alocação de memória, permitindo diagnósticos precisos em um ambiente de desenvolvimento C/C++ de alta complexidade e criticidade.

3.4 Testes de longa duração

Visando avaliar a estabilidade de operação dos algoritmos dos transceptores por um período de tempo maior e sem interrupção, foram realizados testes com duração de dias, que foram classificados de testes de longa duração. Os testes iniciais envolvendo 1 terminal BS e um terminal UE foram realizados considerando o ambiente de laboratório e a comunicação entre os terminais estabelecida via cabo de *Radio Frequency* (RF), a fim de reduzir as variáveis envolvidas e isolar o sistema de interferências externas. Desta forma, os testes iniciais envolveram apenas os sinais gerados e recebidos pelo próprio sistema. Vale destacar que os testes de longa duração foram realizados com a transmissão de dados da BS para a UE, ou seja, enlace de *downlink*, com taxa da ordem de 95 Mbps. A seguir estão descritos os principais testes e análises realizadas neste contexto.

Inicialmente foram realizados testes com duração de 3 a 5 dias, sendo identificados problemas principais relacionados a *bugs* de operação e instabilidade de operação. Os *logs* de operação descritos na Seção 3.1 foram fundamentais para identificação de problemas e possíveis causas e orientar o processo de otimização. Dentre os principais otimizações realizadas, pode-se destacar aquelas relacionadas à etapa de sincronização do terminal UE.

Conforme detalhado no relatório [5], o quadro de transmissão concebido para a rede de acesso Brasil 6G tem como unidade básica o símbolo *Generalized Frequency Division Multiplexing* (GFDM), já que o GFDM é a forma de onda adotada para a rede de acesso [6, 7]. Os símbolos GFDM são agrupados nos chamados subquadros e tem duração de 4,6 ms. Os subquadros, por sua vez, são agrupados em quadros. A Figura 7 mostra um exemplo do quadro e do subquadro para as numerologias 0 e 1 da rede de acesso, que foram detalhadas em [5]. Inicialmente foi adotada que o quadro tem duração total de 32 subquadros, o que equivale a duração de $147,2$ ms. O quadro de transmissão possui em seu início um período de silêncio e um preâmbulo. O período de silêncio é previsto para que os terminais possam realizar o sensoriamento espectral dentro da banda de operação da rede, enquanto que o preâmbulo é empregado como referência principal para a sincronização temporal do quadro. Este preâmbulo é formado por uma sequência de dados fixa e conhecida pelo receptor. No caso dos terminais UE, o preâmbulo é também utilizado para definição do instante de geração de seu quadro de transmissão.

Durante a realização dos testes de longa duração, pode-se verificar que houve falha no processo de sincronização do quadro de transmissão da UE a partir do quadro da BS, o que impactava na paralisação da comunicação entre os terminais. A partir de ajustes e otimizações, pode-se identificar a ocorrência da perda de sincronização e restabelecer a comunicação entre os terminais a partir da re-sincronização. Esta perda de sincronismo ocorre devido a dois fatores principais: diferença acumulativa de erros de frequência entre os osciladores de cada terminal e também acúmulo de latência de processamento na CPU, que pode impactar no atraso nos instantes de recepção e geração do quadro de transmissão. Os testes foram realizados sem o emprego de *Global Positioning System* (GPS) para sincronizar os osciladores de cada terminal, o que torna o testes mais exigente para os algoritmos de sincronização. Porém, considera-se o uso de GPS para a realização de testes posteriores. Com relação a redução de tempo de processamento, foram otimizadas algumas funções identificadas.

Uma vez que os testes com duração de 3 a 5 dias foram finalizados e os problema identificados foram mitigados, foi iniciada uma nova bateria de teste, nas mesmas condições citadas anteriormente, porém, com maior duração: 10 dias ininterruptos. As Figuras 8 e 9 mostram registros realizados na BS no primeiro e no décimo dia de teste, respectivamente. As figuras

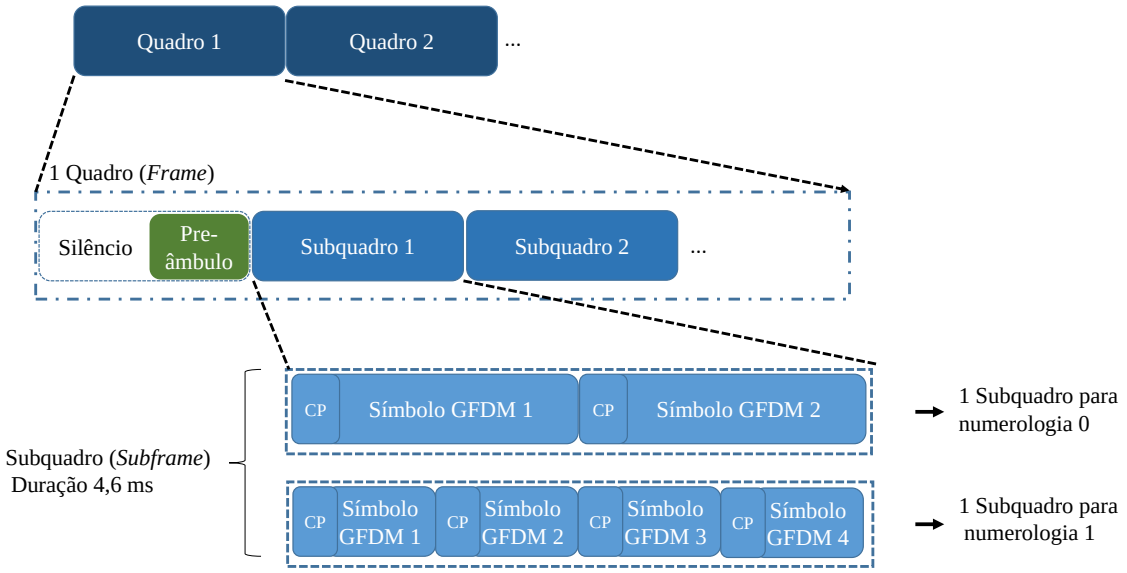


Figura 7: Organização do quadro de transmissão: símbolo GFDM, subquadro e quadro.

exibem a interface do transceptor, ao fundo, sobreposta por algumas janelas com análises e medidas: o resultado do comando `sensors`, identificada pelo número 1 e que exibe medidas de temperaturas do processador; e o resultado do comando `top`, identificada pelo número número 2, que exibe informações referentes a alocação de memória, processamento e processos em execução.

A Figura 8, apresenta o resultado das medidas para o primeiro dia do teste. Vinte e cinco minutos após o início, conforme mostrado na subjanela 2, os resultados do comando `top`, indicam um baixo consumo de memória, limitado a apenas $1,2\%$ dado que o computador possui 32 GB de RAM. Simultaneamente, na subjanela 1, as temperaturas dos processadores se mantêm dentro de níveis aceitáveis, abaixo do limiar considerado elevado (a citar pela temperatura média do conjunto completo dos *cores* dentro de um mesmo processador físico, *Package id 0*, sendo em torno de +70.0°C. É importante enfatizar que todos os elementos do sistema se encontravam instalados em temperatura ambiente.

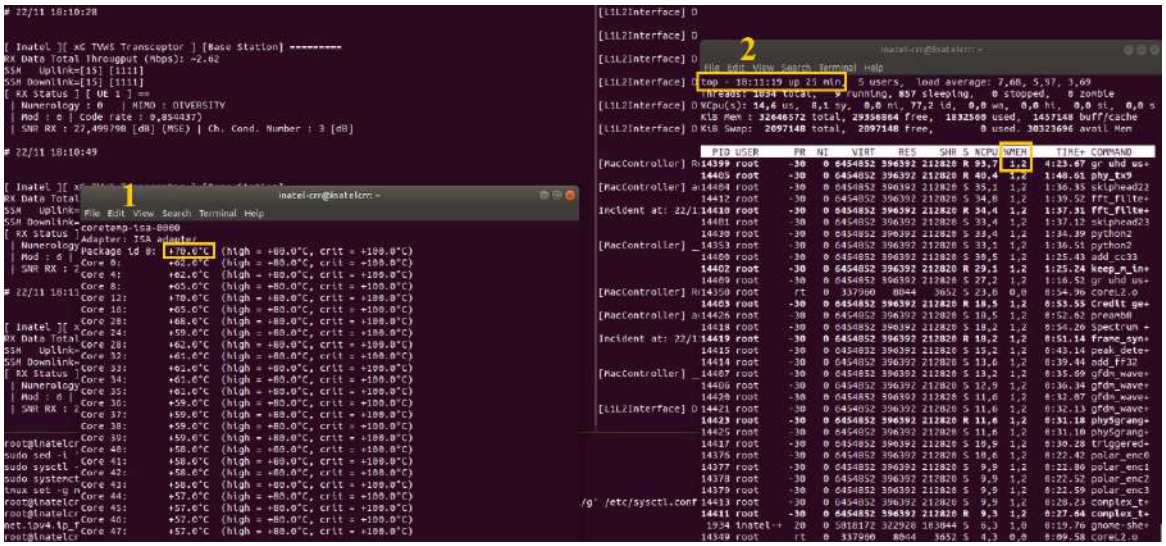


Figura 8: Resultado de medidas realizadas no 1º dia de testes do terminal BS.

A partir da análise das mesmas medidas, porém, tomadas no décimo dia de teste, conforme mostra a Figura 9, pode-se verificar que a temperatura dos processadores se manteve adequada, +68.0°C, como mostra a subjanela 1 da figura, assim como o consumo de memória do sistema que se manteve estável e idêntico ao valor alocado no primeiro dia, como pode ser visualizado na subjanela 2. No topo da subjanela 2 é possível visualizar que o sistema está em execução por 10 dias (termo: *"up to 10 days"*).

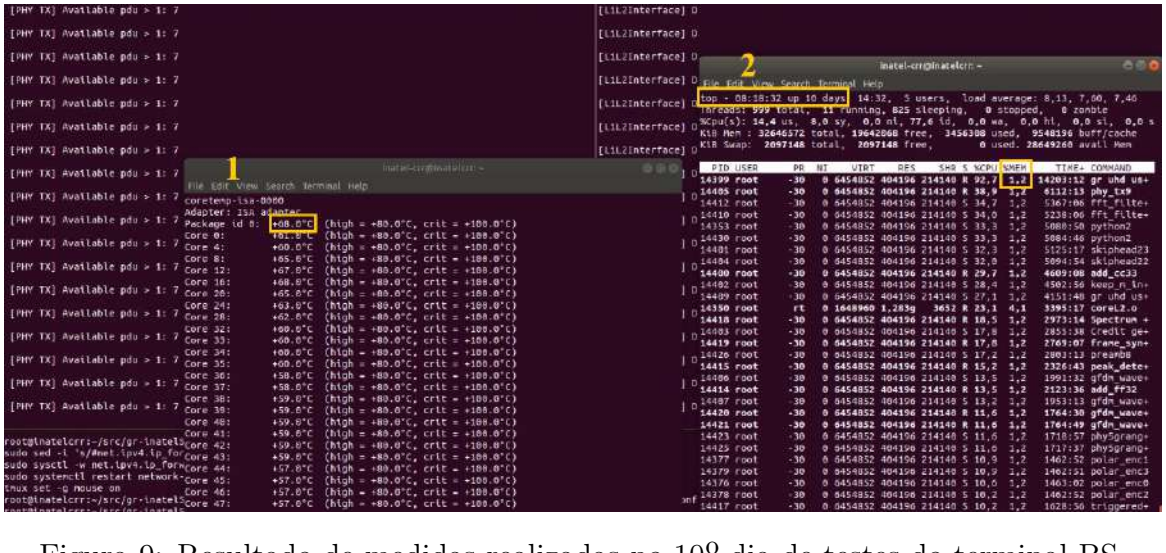


Figura 9: Resultado de medidas realizadas no 10º dia de testes do terminal BS.

As medidas do teste relacionadas ao terminal UE encontram-se apresentadas nas Figuras 10 e 11 e estas mostram para o primeiro e o décimo dia, respectivamente. As figuras mostram resultados de medidas e testes realizados com as ferramentas `iperf3`, exibida na subjanelas 1, `sensors`, exibida na subjanelas 2 e, por fim, `top`, mostrada na subjanelas 3.

Com relação as medidas de temperatura e memória observados no primeiro dia de teste, Figura 10, pode-se perceber valores medidos próximos àqueles associados à BS, com temperatura de +71.0°C e memória alocada em torno de 1.4% para a mesma quantidade de memória

RAM que a BS. A Figura mostra também a medida do `iperf3`, na subjanela 3, com a taxa de comunicação em torno de 95 Mbits/s estabelecida com a BS.

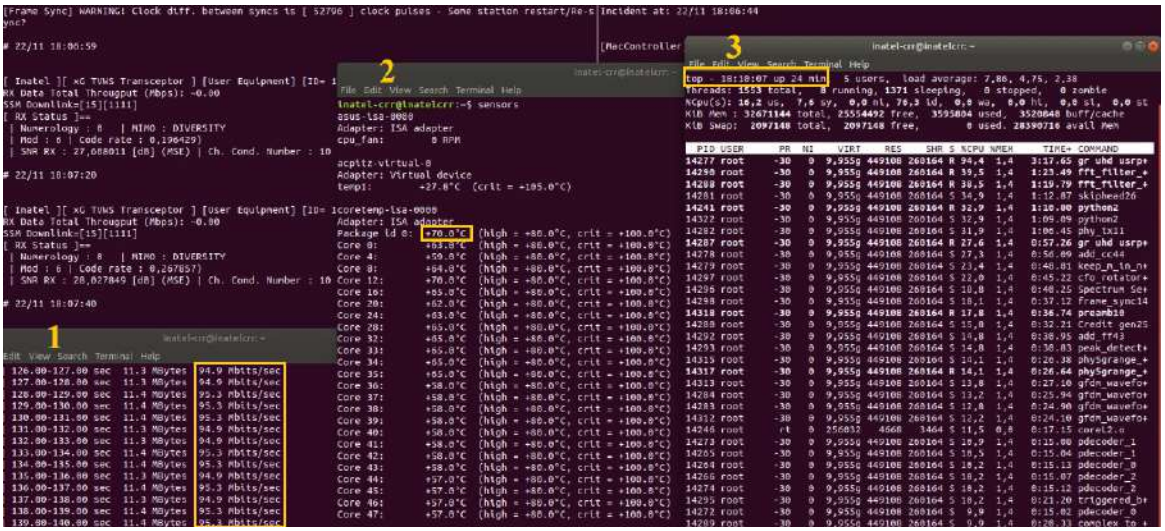


Figura 10: Resultado de medidas realizadas no 1º dia de teste do terminal UE.

Conforme mencionado, as medidas correspondentes ao décimo dia de teste do terminal UE estão apresentados na Figura 11. A figura em questão apresenta as informações organizadas de maneira idêntica à figura do primeiro dia, Figura 10, ou seja, o resultados de medidas com a ferramenta `iperf3` são apresentados na subjanelas 1, da ferramenta `sensors` estão na subjanelas 2 e, por fim, da ferramenta `top` estão mostradas na subjanelas 3. Pode-se observar a partir dos resultados mostrados, que os níveis de consumo de memória e temperatura se mantiveram estáveis e com valores similares aos medidos no primeiro dia de teste e a taxa de comunicação permaneceu inalterada.

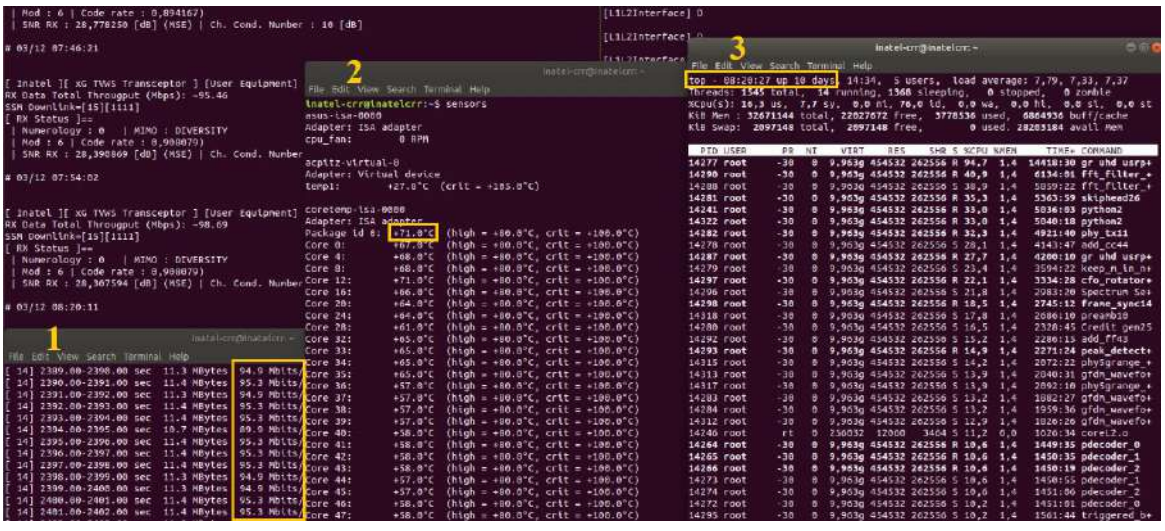


Figura 11: Resultado de medidas realizadas no 10º dia de teste do terminal UE.

Desta forma, pode-se verificar a partir de testes laboratoriais que as otimizações realizadas proporcionaram maior estabilidade de operação em relação aos quesitos avaliados. Vale destacar que testes complementares devem ser realizados em condições diferentes para consolidar a

avaliação. Além disso, testes adicionais, envolvendo outras questões e funcionalidades, devem ainda ser realizados ao longo da execução do projeto.

3.5 Testes e ajustes do MCS de operação

O transceptor Brasil 6G opera com *Adaptive Coding and Modulation* (ACM), que consiste na otimização do enlace de comunicação por meio do ajuste dinâmico da taxa de codificação de canal e esquema de modulação, em função das condições instantâneas do canal. Especificamente, o transceptor altera de forma dinâmica os parâmetros da camada física com base na qualidade do sinal recebido, avaliada por meio de métricas como a SNR. O objetivo do ACM, portanto, é definir uma configuração que otimize a eficiência espectral do enlace, assegurando tanto altas taxas de transmissão quanto robustez e confiabilidade na comunicação.

A implementação de ACM no transceptor Brasil 6G envolve as seguintes etapas principais:

1. Avaliação das condições do canal, baseada na estimativa da SNR de cada terminal, obtida a partir do sinal recebido;
2. Realimentação do sistema de controle, que ocorre de forma centralizada na camada MAC da BS, exigindo, dessa forma, o envio da estimativa de SNR pela UE à BS;
3. Definição, pela camada MAC da BS, de qual valor de MCS a ser utilizado para o enlace de *uplink* e *downlink* para o terminal e instante em questão;
4. Envio à UE da informação sobre qual MCS de *uplink* deve ser utilizado.

Para refinar o valor da SNR requerida por cada MCS, foi realizado um teste prático conforme descrito a seguir. Definiu-se a geração e a transmissão periódica de um pacote de dados conhecido tanto pelo terminal BS quanto pelo terminal UE. Após o processo de demodulação e decodificação do pacote de dados recebido, verifica-se a quantidade de bits errados presentes nele, permitindo, assim, calcular a medida de *Bit Error Rate* (BER) do enlace. Uma medida de BER foi feita de forma acumulativa, sendo exibida a cada 10 s e sendo reinicializada a cada 60 s. Esse tipo de medida viabiliza a rápida identificação de flutuações ou degradações na qualidade do enlace, facilitando também a verificação de padrões de erro relacionados a fatores temporais, como interferências periódicas ou variações ambientais. Implementou-se uma segunda BER acumulativa que é exibida e reinicializada a cada 3×10^9 bits de dados recebidos. Ao se realizar a medida de BER em uma quantidade maior de bits, obtém-se uma estimativa de BER com maior precisão estatística, facilita algumas comparações e análises. Porém, o tempo necessário para processar 3×10^9 bits pode variar conforme a taxa de transmissão, o que pode impactar a frequência das medições.

O refinamento inicial da definição do MCS envolveu a realização de testes práticos de transmissão e recepção de sinais via antena, com o objetivo de considerar condições de recepção mais realistas, ainda que em ambiente laboratorial. A Figura 12 ilustra esse cenário mostrando as estações BS e UE com seus respectivos pares de antenas, Ant 1 e Ant 2, já que o teste considerou o uso de MIMO 2×2. A distância do enlace em questão é de $4,8$ m. O ajuste inicial deverá ser posteriormente refinado, considerando condições reais de operação em campo, além de outros fatores que serão discutidos adiante. Os testes para definição do valor de SNR requerido por cada MCS foram realizados com o *Automatic Gain Control* (AGC) desativado, ou seja, sem ajuste automático do ganho da potência do sinal recebido, a fim de reduzir as variáveis envolvidas nos testes. Devido à curta distância do teste, o ganho de recepção no *front-end* de RF

foi configurado em 0 dB, e nenhum amplificador externo, como o LNA, foi utilizado. Ademais, o teste em questão considerou a análise de comunicação no enlace de *downlink*.

Figura 12: *Setup* utilizado para ajustes iniciais do MCS.

Os testes para definição do valor de SNR requerido por cada MCS foi realizado considerando as seguintes etapas:

- Fixação, tanto na BS quanto na UE, dos parâmetros de operação associados ao MCS (modulação e taxa de codificacão) a ser avaliado;
- Ajuste do ganho de transmissão do terminal BS até observar uma BER igual a 0 por 20 a 30 s. Eventualmente, fez-se necessário adicionar atenuadores na entrada da recepção do terminal UE;
- Uma vez que o passo anterior foi realizado com sucesso, observava-se a BER por, ao menos, 3 ciclos de medidas, cada um com um total de 3×10^9 dados. Caso os valores obtidos da BER em 3 medidas consecutivas fossem iguais a 0, o valor de SNR estimado pelo terminal UE seria o adotado como sendo o valor de SNR requerido para operação utilizando o MCS em questão.

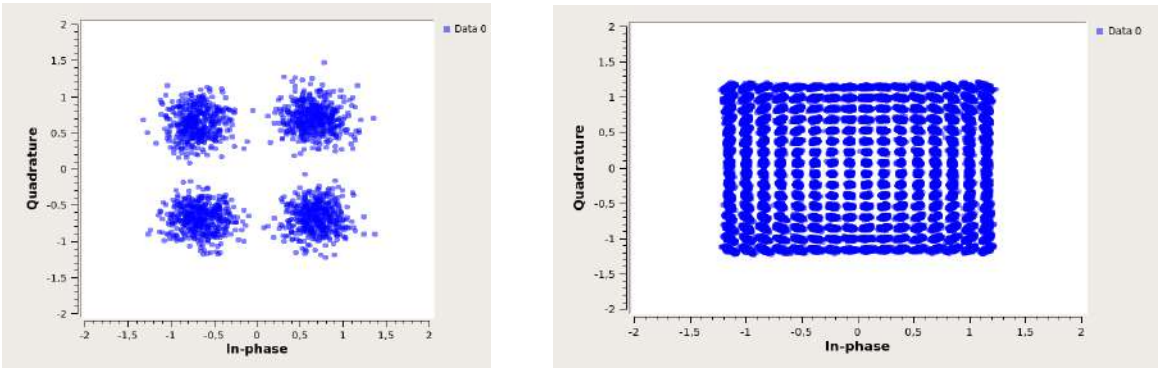
A Tabela 1 mostra o resumo da versão inicial dos valores de SNR requeridos por cada MCS obtido de acordo com a metodologia acima descrita, mostrando um índice associado a cada MCS, bem como a correspondente modulação e taxa de codificação. Pode-se ver também também o valor de SNR médio identificado como requerido para o MCS em questão. Conforme mencionado, tais valores devem ser refinados a partir de testes em condições de campo, considerando também o emprego do AGC e do LNA.

Conforme mostra a Tabela 1, os testes foram realizados para condições de recepção consideradas ruins, com SNR próxima de 12 dB, assim como condições favoráveis, como SNR acima de 33 dB. As Figuras 13a e 13b mostram, respectivamente, uma visão da constelação de recepção

Tabela 1: Versão inicial para a definição de MCS.

MCS Índice	Mapeamento	Taxa de codificação	SNR média (dB)
1	QPSK	0,08333	<12
2		0,16667	12
3		0,375	13
4		0,66667	14
5	16-QAM	0,45833	18,5
6		0,58333	20,5
7		0,625	22,5
8		0,70833	23,5
9	64-QAM	0,58333	25
10		0,66667	26
11		0,70833	27
12		0,79167	29
13		0,875	31
14		0,91667	32
15	256-QAM	0,79167	33
16		0,83333	33,2
17		0,91667	33,5

para esses dois casos extremos do teste, que são associados à operação com MCS com índice 2 e 17. Os valores de taxa de comunicação útil, medida a partir da execução da ferramenta `iperf`, para os respectivos MCSs foram de $4,8$ Mbps e 122 Mbps.



(a) Teste do MCS 2.

(b) Teste do MCS 17.

Figura 13: Constelações obtidas durante os testes de MCS.

4 Conclusão

A realização da Atividade 4.3 da Fase III do projeto Brasil 6G permitiu avanços significativos na avaliação de desempenho da rede de acesso proposta, considerando condições laboratoriais controladas e cenários mais próximos da realidade de uso. A partir da montagem de protótipos funcionais dos transceptores, foi possível implementar e validar otimizações nos algoritmos base do transceptor, garantindo maior estabilidade e eficiência no processamento dos sinais em tempo real.

Os testes conduzidos até o momento forneceram evidências relevantes sobre o comportamento da rede e indicaram pontos que poderão ser ajustados nas etapas seguintes do projeto. A abordagem adotada permitiu não apenas validar o funcionamento dos módulos desenvolvidos, como também antecipar requisitos para a operação da rede em ambientes adversos ou com restrições específicas, como os encontrados em áreas remotas.

Como desdobramento, os resultados obtidos servirão de base para os testes de campo que estão previstos nas próximas fases do projeto, os quais serão essenciais para identificar eventuais limitações operacionais e para consolidar melhorias na arquitetura proposta. Dessa forma, os esforços realizados até aqui representam um passo importante rumo à maturação da solução tecnológica voltada à conectividade em áreas remotas e à consolidação de uma infraestrutura nacional alinhada com os objetivos estratégicos do 6G.

Referências

[1] Centro de Referência em Radiocomunicações (CRR) Inatel, "Atividade 2.2 - Definição dos componenetes de hardware da plataforma," *Projeto Brasil 6G*, Jul. 2022.

[2] ——, "Atividade 2.1 - Definição das arquiteturas física e lógica da rede protótipo," *Projeto Brasil 6G*, Jul. 2022.

[3] GNU Radio, "GNU Radio - The Free & Open Software Radio Ecosustem," 2025, acessado em 26 de junho de 2025. [Online]. Available: http://www.gnuradio.org

[4] KDE Community and contributors, "Heaptrack: a heap memory profiler for linux," https://github.com/KDE/heaptrack, KDE, 2025.

[5] Centro de Referência em Radiocomunicações (CRR) Inatel, "Atividade 3.6 - Formatação de Quadro e Integração," *Projeto Brasil 6G*, Jul. 2023.

[6] F. Conceição, M. Gomes, V. Silva, R. Dinis, A. Silva, and D. Castanheira, "A survey of candidate waveforms for beyond 5G systems," *Electronics*, vol. 10, no. 1, 2021. [Online]. Available: https://www.mdpi.com/2079-9292/10/1/21

[7] N. Michailow, M. Matthé, I. S. Gaspar, A. N. Caldevilla, L. L. Mendes, A. Festag, and G. Fettweis, "Generalized frequency division multiplexing for 5th generation cellular networks," *IEEE Transactions on Communications*, vol. 62, no. 9, pp. 3045–3061, 2014.