UNIVERSIDADE FEDERAL DE SANTA MARIA
CENTRO DE TECNOLOGIA
CURSO DE GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Pedro Melo Carvalho

TELEMETRIA PREDITIVA BASEADA NO COMPORTAMENTO DE FLUXOS DE REDE

Santa Maria, RS
2024

Pedro Melo Carvalho

TELEMETRIA PREDITIVA BASEADA NO COMPORTAMENTO DE FLUXOS DE REDE

Trabalho Final de Graduação apresentado ao Curso de Graduação em Ciência da computação da Universidade Federal de Santa Maria (UFSM, RS), como requisito parcial para obtenção do grau de **Bacharel em Ciência da computação**.

Orientador: Prof. Dr. Carlos R. Paula dos Santos

Santa Maria, RS
2024

Pedro Melo Carvalho

TELEMETRIA PREDITIVA BASEADA NO COMPORTAMENTO DE FLUXOS DE REDE

Trabalho Final de Graduação apresentado ao Curso de Graduação em Ciência da computação da Universidade Federal de Santa Maria (UFSM, RS), como requisito parcial para obtenção do grau de **Bacharel em Ciência da computação**.

Aprovado em 22 de julho de 2024:

_____

Carlos R. Paula dos Santos, Prof. Dr. (UFSM)
(Presidente/Orientador)

_____

Mateus Beck Rutzig, Prof. Dr. (UFSM)

_____

Raul Ceretta Nunes, Prof. Dr. (UFSM)

Santa Maria, RS
2024

DEDICATÓRIA

A todos que em qualquer momento e proporção me estenderam a mão para aliviar o fardo.

Todos esses que aí estão
Atravancando meu caminho,
Eles passarão...
Eu passarinho!

(Mario Quintana)

RESUMO

TELEMETRIA PREDITIVA BASEADA NO COMPORTAMENTO DE FLUXOS DE REDE

AUTOR: Pedro Melo Carvalho
Orientador: Carlos R. Paula dos Santos

Com o crescimento do volume de tráfego de rede, as tecnologias que garantem a segurança e confiabilidade são exploradas e reavaliadas para que se tornem mais eficientes. A telemetria em banda (INT) é uma técnica que reporta o estado da rede, proporcionando relatos de alta granularidade em tempo real. O principal problema da INT é o uso de recursos adicionais por conta do volume adicionado ao tráfego, o *overhead*. Diversos estudos foram conduzidos que investigam esse problema, mas eles têm a mesma limitação, apenas analisam e relatam o passado. O objetivo do presente trabalho é propor uma nova forma de reduzir o *overhead* da telemetria de rede, o que é feito através da redução da frequência de inserção de telemetria em fluxos que possuem comportamentos previsíveis, além disso, o algoritmo deve reportar uma predição de comportamento para fluxos conhecidos. A partir de comparações com outros algoritmos, foi possível notar uma grande redução no *overhead* com baixa perda na precisão das informações reportadas.

Palavras-chave: telemetria, redes de computadores, programabilidade no plano de dados

ABSTRACT

PREDICTIVE TELEMETRY BASED ON NETWORK FLOW BEHAVIOR

AUTHOR: Pedro Melo Carvalho
ADVISOR: Carlos R. Paula dos Santos

With the growth in network traffic volume, technologies that ensure security and reliability are explored and reassessed to become more efficient. In-band Network Telemetry (INT) is a technique that reports the network state, providing high-granularity real-time reports. The main issue with INT is the use of additional resources due to the increased traffic volume, known as overhead. Several studies have been conducted investigating this issue, but they share the same limitation, they only analyze and report the past. The objective of the present work is to propose a new way to reduce the overhead of network telemetry. This is achieved by reducing the frequency of telemetry insertion in network flows with predictable behavior. Additionally, the algorithm should report a behavior prediction for known network flows. Through comparisons with other algorithms, it was possible to observe a significant reduction in overhead with minimal loss in the accuracy of the reported information.

Keywords: telemetry, computer networks, data plane programmability

LISTA DE FIGURAS

SUMÁRIO

1 INTRODUÇÃO

A área de redes de computadores, responsável pela conexão de dispositivos para troca de dados e recursos, tem passado por uma evolução significativa. Inicialmente restritas a redes locais limitadas, expandiram-se com o avanço da tecnologia, incorporando redes sem fio, fibra ótica e protocolos de alta velocidade. Hoje, as redes são cruciais para operações comerciais, educacionais e pessoais, permitindo a transferência de dados em tempo real e suportando a computação em nuvem (SUNSHINE, 1989).

O monitoramento de rede buscar garantir a segurança, confiabilidade e eficiência das infraestruturas de rede, através, por exemplo, da detecção previa de problemas, como congestionamento de tráfego e ataques DDOS. Com o crescimento do acesso à internet, há um grande aumento no volume dos fluxos de dados, o que pode causar congestionamento da rede (QUEIROZ, 2021). Tendo em vista que as tecnologias de monitoramento agregam ao volume, esse aumento evidencia a necessidade da pesquisa de estratégias que as tornem mais eficientes.

As estratégias de monitoramento evoluíram muito com o passar dos anos, entre as mais antigas estão *pull*, em que o operador de rede pede um relatório do seu estado, e *push*, em que a informação é enviada para o operador automaticamente dependendo de alguma condição que pode ser um evento, ou um limite de tempo. Dentre as mais recentes, há a telemetria que reporta informações sobre a rede enviando pacotes na mesma via usada pelos fluxos, a evolução dela é a *In-Band Network Telemetry* (INT), que adiciona as informações reportadas em cabeçalhos inseridos nos pacotes dos fluxos, eles são removidos e entregues ao operador de rede (YU, 2019) (TAN et al., 2021).

Essas estratégias possuem limitações, *pull* e *push* tem problemas de escalabilidade, sincronização e latência. A telemetria tradicional, por sua vez, possui limitações como baixa granularidade e dificuldade de reportar informações em tempo real, e a INT tem como principal problema o *overhead*, volume adicionado ao fluxo pelos cabeçalhos de telemetria.

Múltiplos trabalhos foram publicados que investigam o *overhead*, como sINT (KIM; SUH; PACK, 2018) e DINT (BRUM; SANTOS; FERRETO, 2023a) (ver Seção 2.2). Utilizando-os como base teórica, este trabalho tem por objetivo apresentar uma nova estratégia para reduzir o *overhead* causado pela *In-Band Network Telemetry*. Isso é feito com base no conceito que os comportamento dos fluxos tendem a se repetir (KUMAR et al., 2004), portanto, é possível prever com confiança o comportamento futuro de um fluxo assim que ele for identificado, a fim de reduzir a frequência de inserção de telemetria de fluxos conhecidos e por meio disso, reduzir o *overhead* (KANDULA et al., 2009).

Dentre as contribuições presentes nesse trabalho estão uma forma nova de reduzir o *overhead* da telemetria de rede; uma estratégia que altera as informações reportadas

através de telemetria para uma predição do comportamento esperado do fluxo, logo no primeiro pacote; e uma forma de calcular e atualizar a métrica que guarda a confiança nessa predição. No contexto deste trabalho, o comportamento de um fluxo de rede é definido por duração, número de bytes e número de pacotes de um fluxo, conforme em (ALQAHTANI; ALANAZI; HAMDAOUI, 2020), contudo, cabe destacar que essa definição não é rígida, mas foi considerada por conta das limitações de tempo, apesar disso, outras métricas podem ser incluídas em futuros trabalhos.

Este trabalho está organizado da seguinte forma: no capítulo 2 é apresentada a fundamentação teórica dos conceitos de telemetria, programação no plano de dados, e também, uma revisão do estado da arte. O capítulo 3 contém a descrição da solução proposta deste trabalho, detalhando requisitos, arquitetura e implementação. No capítulo 4 é apresentada a avaliação de desempenho, contendo metodologia, resultados e uma discussão quanto aos resultados. Por fim, no capítulo 5, são discutidos os resultados obtidos a partir da proposta, e considerações finais sobre o trabalho, juntamente com propostas para trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

A primeira seção deste capítulo apresenta uma revisão sobre telemetria, seu propósito, evolução e limitações. Na segunda seção , são discutidos alguns trabalhos relacionados que investigam o *overhead*, e outros que fazem identificação de fluxos.

2.1 TELEMETRIA

Para realizar a manutenção do bem estar das redes de computadores, são necessárias estratégias que possibilitem reportar seu estado. As informações reportadas são utilizadas pelo controlador de rede para diferentes finalidades, como monitorar o desempenho, detectar ataques, ajustar dinamicamente o roteamento do tráfego, e lidar com tráfegos de comportamentos atípicos, como *microbursts* e fluxos elefantes.

Dentre os métodos tradicionais de reportar o estado de rede, vale destacar *pull* e *push*. No *pull*, o controlador de rede ativamente faz a requisição para os dispositivos de rede, já no método *push*, os dispositivos enviam informações de acordo com condições pre-estabelecidas, que podem ser o acontecimento de um evento ou a passagem de um período de tempo. Ambos os métodos possuem limitações, como a latência da coleta de dados do método *pull*, e sobrecarga de processamento do método *push*, ambos também sofrem de dificuldade de escalabilidade.

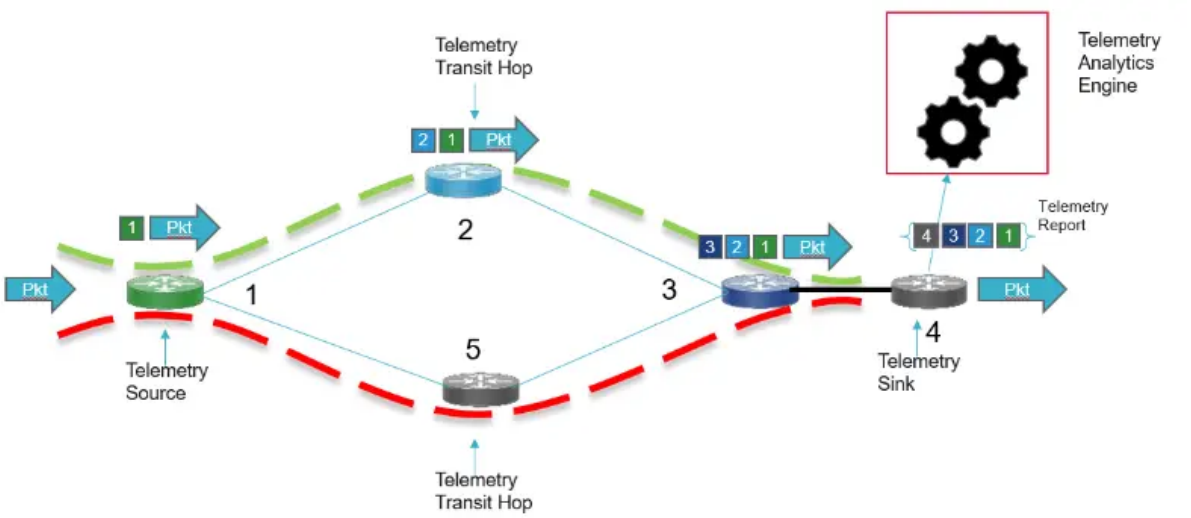
Esses métodos evoluíram para a telemetria, que envia informações da rede utilizando seus próprios dispositivos e vias para levar seus pacotes. Ela possui protocolos mais eficientes, como gRPC (*g Remote Procedure Call*), que permite comunicação entre instâncias de execução e chamada de métodos remotos (MACEDO, 2024), ou SNMP, que diminui o número e complexidade das funções de gerenciamento (CASE FEDOR, 1989). Em geral esses protocolos reduzem a sobrecarga e melhoram a escalabilidade (YU, 2019). Ainda assim, a telemetria tradicional sofre de algumas limitações, como baixa granularidade e dificuldade em reportar o estado da rede em tempo real.

Para combater essas limitações, surgiu a *In-Band Network Telemetry*, que é utilizada para reportar métricas mais específicas, podendo funcionar em periodos variáveis e analisando o fluxo de pacotes individuais. Dessa forma, a informação telemetrada é inserida como um cabeçalho adicional nos pacotes presentes nos fluxos, podem ser inseridos múltiplos cabeçalhos de roteadores diferentes no mesmo pacote. Os cabeçalhos são removidos antes dos pacotes chegarem ao destino e entregues ao controlador de rede (KIM et al., 2015).

A figura 1 mostra uma rede que utiliza INT, a qual possui alguns componentes. O roteador 1 é uma *telemetry source* (fonte de telemetria), que é uma entidade de rede de

confiança que insere cabeçalhos nos pacotes que envia, nesse exemplo, ele é a fonte para os fluxos vermelho e verde. O roteador 2 é um *telemetry hop* (salto de telemetria), ou seja, um nodo intermediário por onde os pacotes passam, esses nodos podem ou não estar programados para inserir mais dados nos pacotes. O roteador 4 é um *telemetry sink* (coletor de telemetria), que é um nodo responsável por extrair os cabeçalhos e enviar as informações telemetradas para o controlador de rede.

Figura 1 – Uma rede com capacidade de telemetria.



Fonte: Adaptado de (MANDAL, 2020).

A principal limitação da INT é o *overhead*, que, neste contexto, significa o custo extra causado pela inserção de cabeçalhos de telemetria de rede, medido em bytes. Diversos trabalhos foram publicados nos últimos anos que tem por finalidade minimizar o *overhead*, sem reduzir significativamente a precisão das informações telemetradas, a próxima seção fala sobre alguns desses trabalhos.

2.2 TRABALHOS RELACIONADOS

Nos últimos anos foram publicados diversos trabalhos que investigam o *overhead* causado pela telemetria de rede. Eles propuseram estratégias como dinamicamente avaliar e descartar informações menos valiosas; ou dividir os cabeçalhos, inserindo partes dele em múltiplos pacotes do fluxo, diminuindo o *overhead* por pacote; ou estratégias que alteram dinamicamente a frequência de inserção de telemetria. Estes e demais trabalhos estão detalhados abaixo.

No trabalho de (CHOWDHURY et al., 2014), é apresentado um framework chamado Payless, que fornece uma API para a coleta de estatísticas de fluxo em diferentes níveis de

agregação. Ele usa um algoritmo adaptativo de coleta de estatísticas que fornece informações em tempo real. A fim de reduzir o *overhead*, a frequência de inserção de telemetria é alterada de acordo com o volume de fluxos passando pela rede, conforme o volume aumenta, a frequência tende a, também, aumentar. O framework busca alcançar um equilíbrio entre precisão e saturação de rede, e traz consigo a capacidade de abstração da rede e uniformização dos pedidos de estatísticas. Ele conta com alta modularidade e customização a fim de ser uma tecnologia bem adaptável. Essa proposta apresenta algumas limitações, como a necessidade de que o algoritmo de extração de estatísticas tenha uma estimativa da quantidade de dados que passarão pela rede, a fim de alterar a frequência de coleta. Outra limitação é: por conta da frequência variar conforme o volume do tráfego, o método é vulnerável à saturação causada por fluxos elefante.

Em (KIM; SUH; PACK, 2018), é apresentado o sINT, que tem como proposta o ajuste dinâmico da frequência de inserção de telemetria. Ele utiliza a tupla-5 de cada fluxo como chave de uma tabela hash, essa tabela guarda a frequência de inserção específica de cada fluxo. A frequência pode ser alterada a cada vez que a telemetria é inserida no fluxo, caso haja mudança considerável nos valores reportados, a frequência aumenta, senão ela diminui. Uma desvantagem do sINT é, caso a rede se encontre instável, ou tenham aconteçam constantes mudanças de roteamento, a frequência tende a ficar muito alta.

(CHOWDHURY; BOUTABA; FRANÇOIS, 2021) apresenta um mecanismo de baixo custo, chamado LINT, que funciona no plano de dados, e tem por objetivo diminuir o *overhead*. A estratégia adotada é a de analisar e descartar estatísticas menos desejadas, diminuindo a saturação do plano de dados. Uma vantagem desse método é ele não necessitar de comunicação com o plano de controle. Mesmo assim, o método apresenta limitações, dentre elas, o LINT analisa e decide se deve enviar a informação de cada pacote individualmente, portanto em seu pior caso, ele pode decidir envias as estatísticas com muita frequência, causando grande custo para a rede. Da mesma forma ele pode decidir não inserir telemetria por períodos muito longos de tempo, o que faria com que o mecanismo de monitoração perdesse muita precisão.

O trabalho de (PAN et al., 2021) propõe PINT, um framework que visa dispersar o *overhead*, dividindo os cabeçalhos de telemetria em múltiplos pacotes, delimitando um limite de volume adicional por pacote, de forma a reduzir a largura de banda necessária. Isso é feito utilizando funções de hash globais que determinam quais pacotes devem ser escolhidos para carregar os bits de telemetria, além disso, usando probabilidade para estimar métricas referentes à ocupação da fila. Porém, o PINT é ineficiente para fluxos de curta duração e também é vulnerável a mudanças de roteamento.

Em (BRUM; SANTOS; FERRETO, 2023a) é apresentado o DINT, que traz uma proposta para alterar dinamicamente a frequência de inserção de telemetria. Essa estratégia propõe que a frequência seja proporcional à estabilidade da rede, ou seja, caso a vazão

de dados se mantenha relativamente constante, a frequência de inserção diminui, caso o comportamento da rede se torne errático, a frequência aumenta até um certo limite. Uma limitação que o DINT possui é, em fluxos de longa duração, os valores utilizados como limites da frequência são relativamente altos, o que gera considerável sobrecarga na rede.

Há alguns trabalhos publicados que utilizam formas de predição, embora eles tenham por objetivo identificar fluxos elefantes, que são fluxos de grande duração e volume que utilizam a maior parte dos recursos das redes (o que não faz parte do escopo desse trabalho), é valido utilizá-los como base, adaptando suas estratégias para o contexto da telemetria de rede.

Por conta do impacto que fluxos de longa duração e volume têm em redes de alta performance, faz-se necessário identificar esses fluxos, a fim de tratá-los. (SILVA et al., 2018) apresenta o mecanismo IDEAFIX que propõe identificar fluxos elefante em Ponto de Troca de Tráfego (*Internet Exchange Points*-IXPs) baseadas em P4. O trabalho propõe guardar o volume e duração dos fluxos em registradores indexados por chaves hash. Essas informações são extraídas a cada pacote e comparadas com limiares de classificação. Caso um fluxo seja identificado como elefante, uma notificação é enviada ao plano de controle. As ações que serão tomadas para tratar esse fluxo são predefinidas pelo operador de rede. Uma desvantagem dessa abordagem é que o fluxo só é identificado como elefante depois que uma quantidade suficientemente grande de pacotes passaram pelo roteador, o que faz com que o tempo de identificação seja relativamente demorado.

Em (SILVA et al., 2020) é proposta uma solução que através da predição do comportamento de fluxos de rede identifica fluxos elefante antes que eles excedam os limiares. O algoritmo realiza a predição do volume e da duração de fluxos, permitindo que o tratamento deles seja feito de forma mais rápida e eficiente. A predição é feita utilizando o modelo *Locally Weigthed Regression* (LWR) que infere os valores de variáveis independentes a partir de variáveis dependentes, utilizando ponderação localizada. As variáveis dependentes escolhidas foram volume e duração de fluxos, a variável independente escolhida foi o instante de tempo em que o fluxo chega no roteador (*timeStamp*). Uma desvantagem dessa estratégia é que ela necessita uma certa periodicidade padronizada do fluxo, ou seja se a frequência de ocorrência de um fluxo elefante for instável, o método não vai identificá-lo com precisão.

2.3 DISCUSSÃO

Na seção anterior foram discutidos alguns trabalhos que têm como objetivo reduzir o *overhead* causado pela telemetria de rede, eles utilizam diferentes estratégias que em sua maioria alteraram dinamicamente o tempo de inserção de telemetria. Uma limitação que todos têm em comum é que eles funcionam de forma reativa, ou seja, observam uma

janela de tempo para posteriormente decidir se a telemetria deve ser inserida, percebendo apenas mudanças momentâneas na rede, e mantêm este comportamento independente da natureza intrínseca do tráfego. Seria interessante que mecanismos de telemetria se adaptassem em função dessa previsibilidade, de forma a não apenas reduzir o *overhead* mas, além disso, alterar a informação que está sendo reportada para o monitoramento. Dessa forma, na próxima sessão é apresentada a solução proposta deste trabalho.

3 SOLUÇÃO PROPOSTA

Nesse capítulo é explicada a solução proposta por esse trabalho. Também são apresentadas as seções: requisitos, que fala sobre os objetivos que devem ser cumpridos pelo trabalho; arquitetura, que apresenta a estrutura do algoritmo, e as grandes funções dele; e por fim, implementação, que mostra alguns trechos mais relevantes do código.

3.1 SÍNTESE DA PROPOSTA

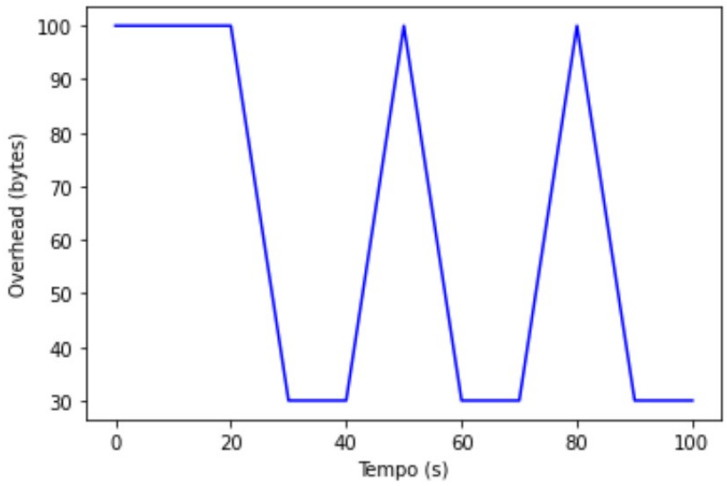
Utilizando alguns conceitos apresentados na seção 2.2, principalmente em (KIM; SUH; PACK, 2018) e em (BRUM; SANTOS; FERRETO, 2023a), esse trabalho propõe o *Recurrent INT* (RINT), uma alternativa para definir o tempo de inserção de telemetria. A proposta tem por base a premissa de que, comportamentos dos fluxos de rede tendem a se repetir (KUMAR et al., 2004), dessa forma, ela utiliza uma tabela de recorrência, que guarda informações dos fluxos de rede, sendo elas: duração, quantidade de *bytes*, e quantidade de pacotes. Além disso, possui uma métrica que guarda a confiança de que, quando o fluxo se repetir, ele apresentará o comportamento esperado. A partir disso, dado que a confiança no comportamento de um fluxo esteja alta o suficiente, o seu tempo de inserção de telemetria é aumentado, e a métrica reportada no seu primeiro pacote é alterada para informar seu comportamento esperado.

3.2 REQUISITOS

Dentro do escopo desse trabalho está o principal requisito de reduzir o *overhead* causado pela *In-Band Network Telemetry*, sem ocasionar em perda significativa na precisão das informações telemetradas quanto ao estado real da rede. Essas métricas de desempenho são consideradas as mais fundamentais no contexto da telemetria de rede, dado que a eficiência de um método de inserção é medida principalmente com a sua capacidade de reduzir o *overhead* sem perder precisão.

A figura 2 mostra a expectativa referente ao *overhead* em função do tempo, onde o algoritmo começa apenas coletando dados, e inserindo a telemetria de forma estática. Quando um fluxo tiver se repetido vezes o suficiente e a confiança em seu comportamento estiver acima do limiar, ele entra no modo de confiança, que aumenta consideravelmente o período de inserção de telemetria e, por consequência, o *overhead* diminui proporcionalmente. Os picos que acontecem após isso são momentos onde fluxos desconhecidos

Figura 2 – Overhead esperado.



Fonte: Autor.

passam pelo roteador. É importante destacar que, o método utilizado quando não se está tratando de um fluxo de confiança poderia ser outro, mas para esse trabalho o método escolhido foi a telemetria estática.

Outro requisito mínimo é alterar as métricas telemetradas no primeiro pacote de cada fluxo conhecido, a fim de reportar uma predição de seu comportamento futuro. A partir dessa predição, o controlador de rede poderá realizar ações preventivas de acordo com as métricas reportadas, como por exemplo, alterar o roteamento de determinado fluxo ou interromper sua propagação.

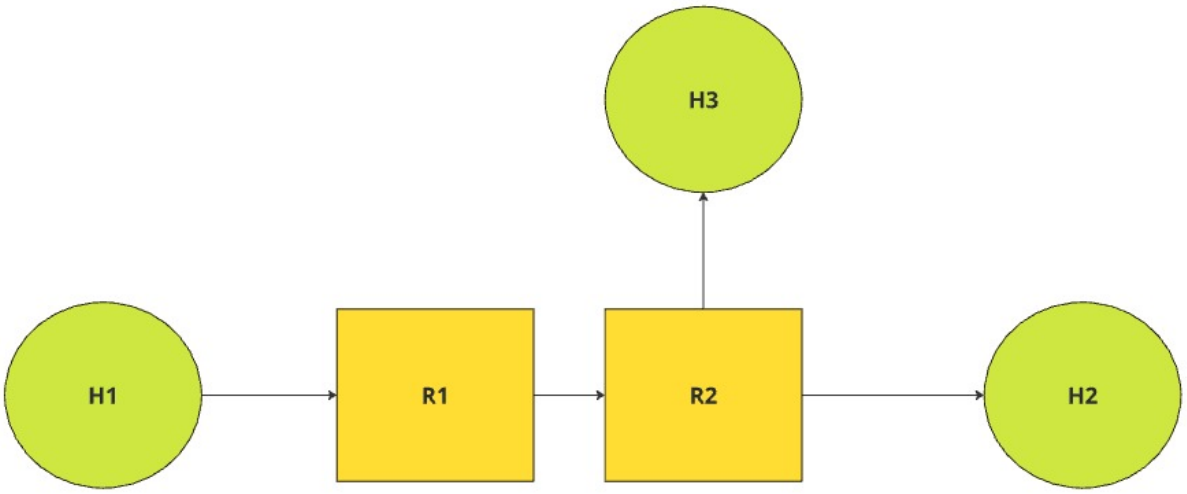
Por fim, é importante que o algoritmo não gere um aumento significativo no atraso de processamento dos fluxos de rede. Esse atraso deve ser calculado, e reduzido o quanto possível, a fim de medir se os ganhos com *overhead* e predição são significativos o suficiente para compensarem o atraso adicionado ao processamento.

3.3 ARQUITETURA

A topologia utilizada é composta por 3 *hosts*, e 2 roteadores, onde um *host* envia pacotes, enquanto os outros dois apenas recebem, e os roteadores inserem ou removem telemetria e repassam os pacotes. Como mostrado na figura 3, $H1$ envia pacotes que têm por destino $H2$, esses pacotes passam pelo roteador R1 que realiza a inserção de telemetria e envia-os ao próximo *switch*. O roteador $R2$ retira os cabeçalhos de telemetria e os envia ao *host* $H3$, e envia os pacotes para $H2$.

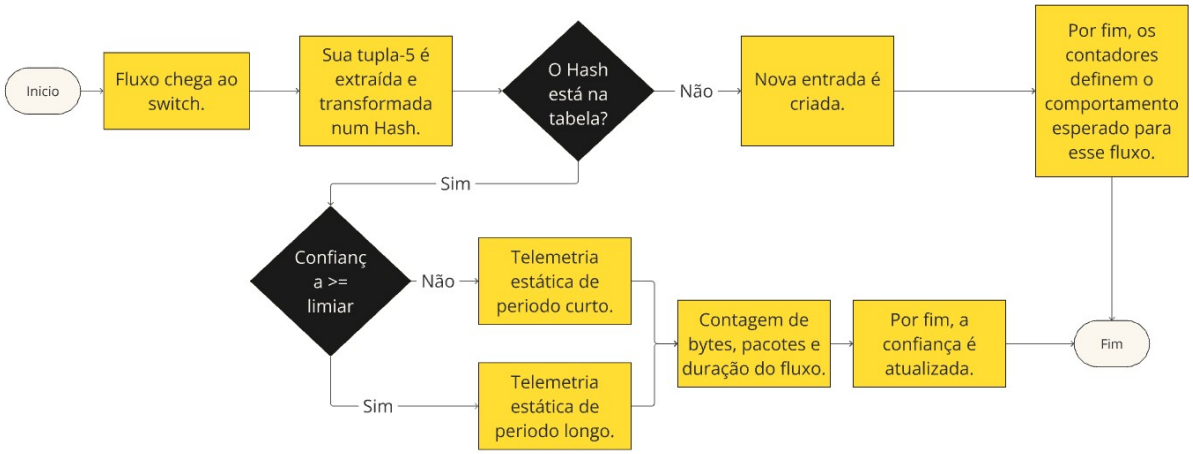
No roteador $R1$ (mostrado na figura 4), o algoritmo de inserção de telemetria é executado, primeiramente, tupla-5 é extraída (porta origem, porta destino, ip origem, ip

Figura 3 – Topologia utilizada.



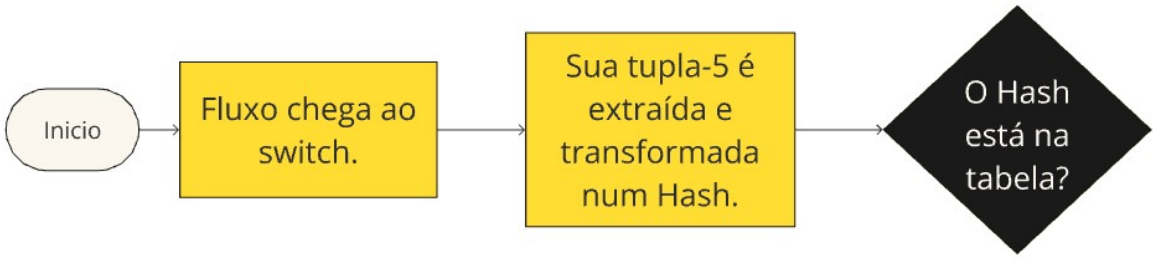
destino, protocolo), e convertida em uma chave *Hash*. Na sequência, é feita uma busca procurando essa chave na tabela de fluxos recorrentes, como visto na figura 5. Caso não haja uma célula na tabela para essa chave, uma nova entrada é criada e seus valores são inicializados. Então, a telemetria para esse fluxo será inserida com um período estático curto, sua duração, quantidade de pacotes e bytes são contados. Ao fim do fluxo, seu comportamento é atualizado a partir dos contadores, e sua confiança é inicializada, como mostrado na figura 6.

Figura 4 – Fluxograma do roteador 1



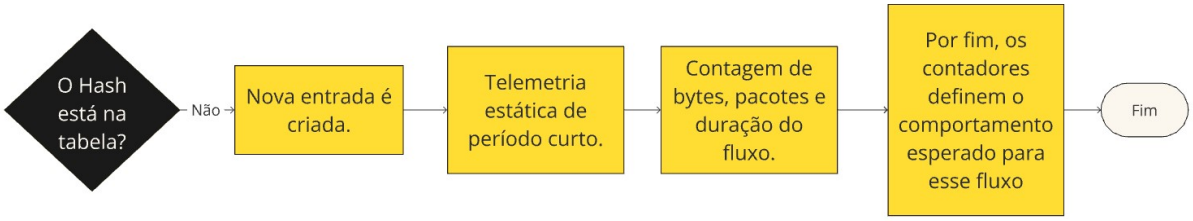
Fonte: Autor.

Figura 5 – Parte do fluxograma do roteador 1: A tupla-5 é pesquisada na tabela de fluxos recorrentes.



Fonte: Autor.

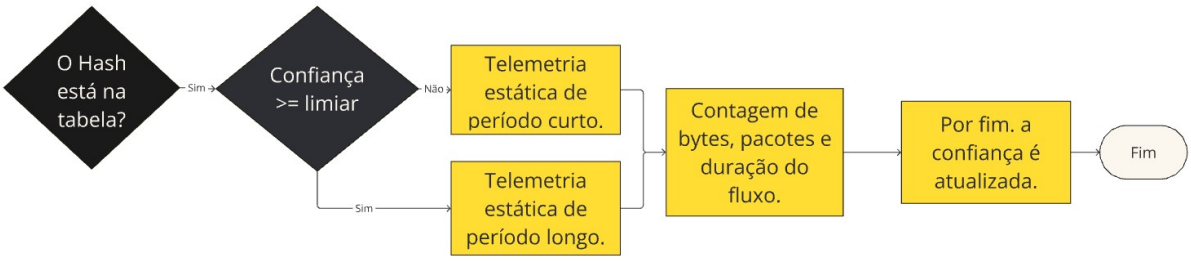
Figura 6 – Parte do fluxograma do roteador 1: Caso não haja entrada para a chave do fluxo na tabela de comportamentos.



Fonte: Autor.

Na figura 7 é mostrado o caso em que há uma entrada para a tupla-5 atual na tabela de fluxos recorrentes, então é verificado se a confiança para esse fluxo está acima de um limiar. Caso não esteja, a telemetria será inserida com um período curto, do contrário, a telemetria será inserida com uma frequência mais alta, e também, um cabeçalho que carrega o comportamento esperado para este fluxo é inserido no seu primeiro pacote. De qualquer modo, ao fim do fluxo, seu comportamento apresentado é comparado com o esperado e a confiança é atualizada.

Figura 7 – Parte do fluxograma do roteador 1: Caso haja entrada para a chave do fluxo na tabela de comportamentos.



Fonte: Autor.

3.4 IMPLEMENTAÇÃO

Essa seção aborda detalhes de implementação e funcionamento do algoritmo, comentando as ferramentas utilizadas e alguns trechos de código. A linguagem utilizada para esse trabalho foi *python3*, que possui a biblioteca *scapy* a qual permite a manipulação de pacotes de rede. O trabalho também faz uso da ferramenta *Mininet* que foi usada para emular uma rede de computadores, utilizando da topologia discutida na seção 3.3.

O cabeçalho de telemetria utilizado nesse trabalho possui quatro campos, dos quais, um carrega uma *flag* e os outros carregam informações sobre comportamento esperado ou observado: $current\_or\_expected$ carrega um bit que, quando 0, informa se o cabeçalho reporta a ultima janela de observação e, quando 1, carrega uma predição de comportamento; $packets\_counted$ carrega a contagem de pacotes da ultima janela de observação, ou a quantidade de pacotes esperada para o fluxo atual; $bytes\_counted$ carrega a contagem de bytes da ultima janela de observação, ou a predição da quantidade de bytes do fluxo atual; e $duration$ carrega a duração esperada para o fluxo atual, ou 0, caso $current\_or\_expected$ seja 0. O código desse cabeçalho é apresentado no Algoritmo 1.

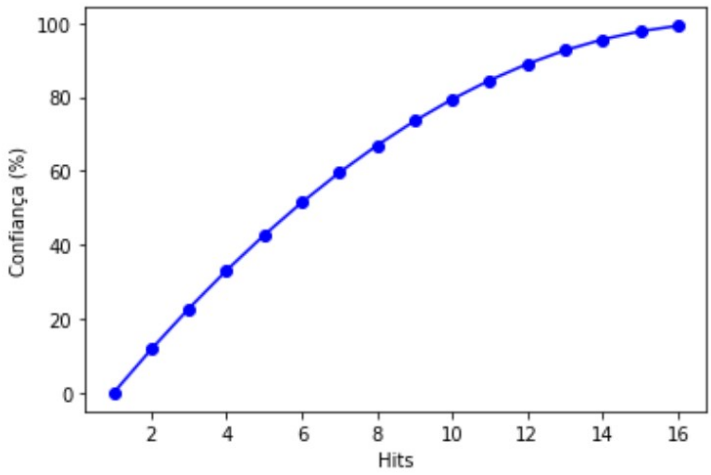
Algoritmo 1: Header

```
1 name = 'Telemetry'
2 fields_desc = [ bit("current_or_expected"),
3                 int("packets_counted"),
4                 int("bytes_counted"),
5                 float("duration")]
```

A confiança é guardada na forma de uma sequência de 16 bits, que guarda como '1' os *hits* (quando o fluxo apresentou um comportamento próximo ao esperado) e '0' os *misses* (quando o fluxo não apresentou o comportamento esperado). As novas entradas na sequência são inseridas realizando um *left shift* e inserindo o novo bit à sua direita. O valor da confiança é calculado de forma que as entradas mais antigas (à esquerda) tenham peso menor do que as mais novas (à direita), dá-se a cada bit um peso que é n/m, sendo n a sua posição na sequência começando em 1, e m sendo o somatório dos números entre 1 e 16, então é feita a soma desses valores e transformada em uma porcentagem.

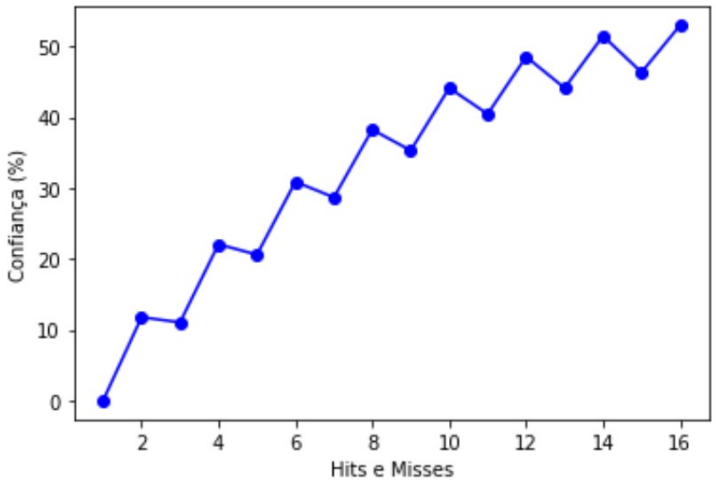
Para testar esse método do cálculo de confiança foram feitos alguns cenários, a Figura 8 mostra o primeiro cenário onde 16 *hits* são inseridos em sequência, a Figura 10 mostra o segundo cenário onde são inseridos 16 *misses*, a Figura 9 mostra o terceiro cenário, em que a sequência está inicializada com 16 *misses*, e então são inseridos *hits* e *misses* alternadamente, e a Figura 11 mostra o quarto cenário onde a sequência está inicializada com 16 *hits*, e são inseridos *misses* e *hits* alternadamente. A partir destes testes é possível verificar que o método toma uma forma que se assemelha a uma parábola, e que os pesos fazem com que a confiança tenda a 50%.

Figura 8 – Cálculo da certeza com a confiança inicializada em 0%, e então inserindo 16 *hits*.



Fonte: Autor.

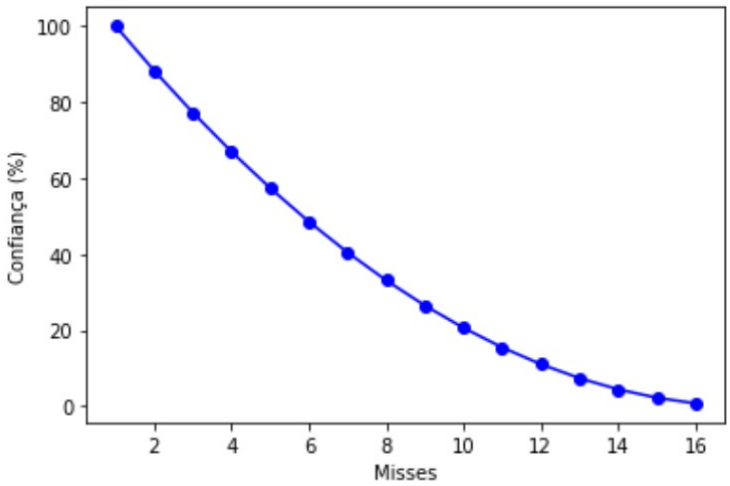
Figura 9 – Cálculo da certeza com a confiança inicializada em 0%, e então inserindo *hits* e *misses* alternados.



Fonte: Autor.

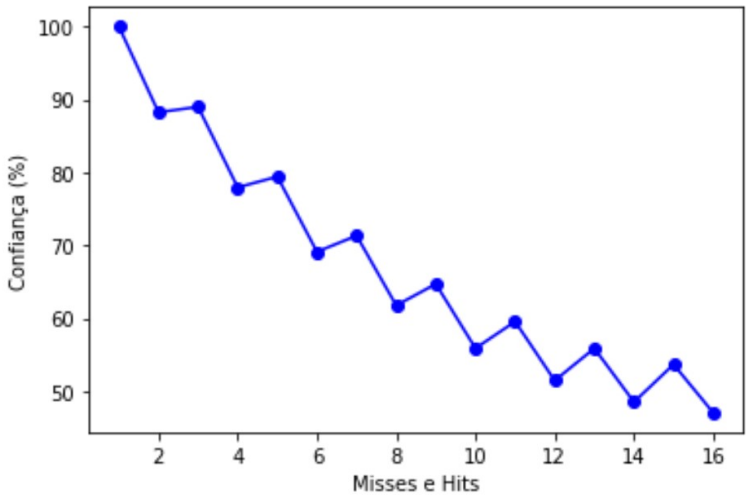
No Algoritmo 2 é mostrado o algoritmo do roteador 1, que é responsável por fazer a contagem dos pacotes e bytes que trafegam pelo roteador, e também por medir o comportamento do fluxo atual, a fim de passá-lo para a tabela de comportamentos recorrentes. Ele chama o método *checkTuple5* da tabela, que retorna um *char*, que serve para informar se o pacote atual marca o inicio, fim ou continuação de um fluxo. Caso seja o inicio, a confiança no comportamento desse fluxo é comparada com o limiar de confiança, se estiver acima do mínimo, o modo de confiança é alterado para 1, e o comportamento esperado para esse fluxo é inserido como um cabeçalho de telemetria. Carregando o valor 1 para $current\_or\_expected$, essa inserção não interfere no tempo de telemetria padrão, já que

Figura 10 – Cálculo da certeza com a confiança inicializada em 100%, e então inserindo 16 *misses*.



Fonte: Autor.

Figura 11 – Cálculo da certeza com a confiança sendo inciada em 100%, e então inserindo *misses* e *hits* alternados.



Fonte: Autor.

ela tem o objetivo de carregar uma predição ao invés de reportar a janela de observação. Caso seja o ultimo pacote de um fluxo, o modo de confiança é alterado para 0.

Em seguida, o algoritmo verifica se o tempo para a próximo inserção, o qual é dependente do $confidenceMode$, chegou ao tempo da janela de observação, caso tenha chegado, ele reinicia os contadores de tráfego (que são independentes dos contadores para o fluxo atual). Então é feita a verificação do tempo de telemetria, e a inserção do cabeçalho com a o valor 0 para $current\_or\_expected$.

O método $checkTuple5$ é mostrado no Algoritmo 3, que recebe um pacote e o comportamento apresentado pelo fluxo do qual faz parte. Então, ele extrai a tupla-5 do pacote e compara com a do pacote anterior, a fim de determinar se eles fazem parte do mesmo fluxo. Se as tuplas forem iguais, é feita uma verificação que determina se o pacote é o

Algoritmo 2: Router 1

```
1  pktCounter()
2  behavior = ('pktCount': currentFluxPktCount,
                'byteCount': currentFluxByteCount,
                'duration': currentTime - fluxTimeStart)
3  dataFlowCode = recTable.checkTuple5(pkt, behavior)
4  if dataFlowCode == 'I' and recTable.getConfidence(pkt) >= confidenceThreshold
     then
5      confidenceMode = 1
6      expectedBehavior = recTable.getExpectedBehavior(pkt)
7      pkt = insertHeader(pkt,1,expectedBehavior)
8  else if dataFlowCode == 'F' then
9      confidenceMode = 0
10 if timeUntilInsertion() == observationWindow then
11     restartFlowCounters()
12 if checkTelemTime() then
13     insertHeader(pkt,0,pktCount,byteCount)
```

último de seu fluxo, caso seja, o método $updateRecurrentTable$ é chamado. Esse método verifica se o fluxo já possui um comportamento esperado, ou atualizar sua sequência de confiança, então o algoritmo retorna 'F', indicando o fim do fluxo. Caso o pacote não seja o último de seu fluxo, o algoritmo retorna 'C', indicando que há continuação. Se as tuplas não são iguais, trata-se de um novo fluxo, então a variável que guarda a tupla anterior é atualizada, e o método $makeNewEntry$ é chamado, ele verifica se há entrada para esse fluxo na tabela, e se não houver, uma nova é criada, então o algoritmo retorna 'I', informando que um novo fluxo foi inciado.

Algoritmo 3: RecTable.checkTuple5

```
   Data: pkt,behavior
   Result: 'I','C','F'
1  t5 = getT5(pkt)
2  if t5 = previousT5 then
3      if lastPktFromFlow(pkt) then
4          updateRecurrentTable(t5,behavior)
5          return'F'
6      return'C'
7  else
8      previousT5 = t5
9      makeNewEntry(t5)
10     return'I'
```

O Algoritmo 4 recebe uma tupla-5 e um comportamento de fluxo. Então, ele gera uma chave *hash* a partir da tupla e verifica na tabela se há um comportamento atrelado a

essa chave, caso não haja ele coloca os valores do comportamento observado para serem guardados como comportamento esperado desse fluxo. Caso o fluxo já tenha um comportamento esperado, resta atualizar a sequência de confiança, isso é feito dividindo o comportamento observado pelo esperado e subtraindo 1. Em seguida, o módulo desse calculo é comparado com uma métrica de precisão, caso esteja acima, o novo bit da sequência será '1', senão, será '0'.

Algoritmo 4: RecTable.updateRecurrentTable

```
   Data: t5,behavior
1  hashed = getHashed(t5)
2  if behaviorTable[hashed]['behavior'] == NULL then
3      behaviorTable[hashed]['behavior'] = behavior
4  else
5      if absolute((observedBehavior/expectedBehavior) -1) <= precision then
6          behaviorTable[hashed]['confidence'].shifLeft('1')
7      else
8          behaviorTable[hashed]['confidence'].shifLeft('0')
```

4 ANÁLISE DE DESEMPENHO

Nesse capítulo são apresentados os métodos de teste usados para medir o desempenho do algoritmo desenvolvido, levando em conta as métricas apresentadas na seção 3.2, e sendo estruturado nas seguintes seções: Ambiente de teste e metodologia, que fala sobre as tecnologias usadas para simular o *hardware* necessário e o tráfego de rede, além disso, é apresentada a metodologia empregada nos testes escolhidos, e os algoritmos usados como comparativo ao que desenvolvido nesse trabalho; Resultados obtidos, onde são apresentados os dados gerados a partir dos experimentos realizados; e Discussão, onde os resultados são analisados e interpretados, juntamente com algumas implicações que podem ser concluídas a partir deles.

4.1 AMBIENTE DE TESTE E METODOLOGIA

O *software* utilizado para simular os *hosts*, roteadores e também o envio de pacotes de rede foi o *Mininet*. A topologia, explicada na seção 3.3, foi implementada utilizando 5 *hosts*, visto que, no *Mininet*, *switches* não podem executar *scripts*, e apenas repassam pacotes. Os *scripts* usados em cada um dos nodos da topologia foram feitos na linguagem Python3, e foi empregado o uso da biblioteca scapy, que permitiu a manipulação dos pacotes e inserção de telemetria.

Baseados nos testes realizados em (BRUM; SANTOS; FERRETO, 2023a), foram construídos três cenários : o primeiro, onde a vazão dos pacotes é constante; o segundo, onde a vazão começa alta e vai diminuindo até metade do período de teste, e aumenta até o mesmo valor que estava no inicio; e o terceiro, onde a vazão é errática, para o qual o tempo entre o envio de pacotes varia entre 0 e 0.5 segundos. Todos os cenários possuem duração de 300 segundos e em cada cenário há um fluxo de comportamento repetitivo, que é enviado a cada 320 pacotes. Esse fluxo possui duração média de 10 segundos, com tamanho em pacotes de 160 e tamanho em bytes de 8800.

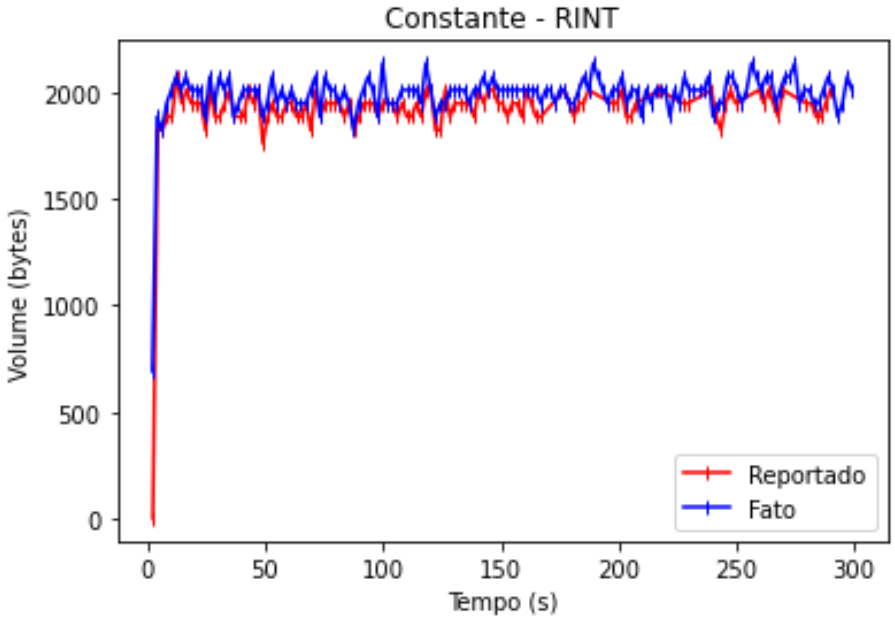
Esses cenários serão utilizados para testar três algoritmos: o primeiro com período de inserção de telemetria estático de 2 segundos, o RINT e o DINT de (BRUM; SANTOS; FERRETO, 2023a). Para o RINT os parâmetros utilizados foram: precisão 20% (valor utilizado quando compara-se o comportamento esperado e o observado), limiar de confiança 70%. Esses valores foram obtidos a partir de testes que buscavam parâmetros que geram o comportamento desejado para o algoritmo. O DINT possui como parâmetros T_min e T_max igual a 2 e 5 segundos, respectivamente, k igual a 8, e α e β com os valores 1.125 e 10. Esse parâmetros foram escolhidos conforme um dos casos de teste de (BRUM; SANTOS; FERRETO, 2023b).

Os testes guardam as quantidades e tamanhos dos pacotes que chegam ao *host* destino ($H2$), e as informações telemetradas enviadas para o *host* que simula o operador de rede ($H3$). Dessa forma, é possível comparar as informações telemetradas com o tráfego real da rede e, além disso, medir a precisão por meio do cálculo do erro quadrático médio (RMSE) e o atraso médio de processamento por pacote. Por fim, os três métodos vão ser comparados através da medição o volume de cabeçalhos de telemetria com relação ao tempo de cada um.

4.2 RESULTADOS OBTIDOS

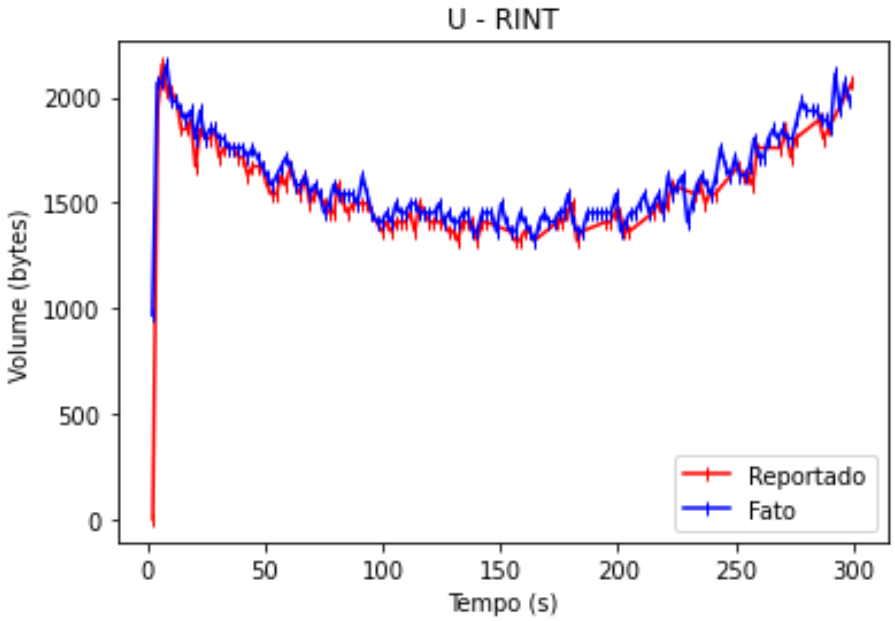
Os resultados obtidos para o algoritmo RINT nos três cenários discutidos na seção anterior são mostrados nas figuras abaixo. O azul representa o volume que chega ao $H2$, enquanto o vermelho representa as informações telemetradas que são entregues ao $H3$. A figura 12 mostra o cenário de teste em que o volume de pacotes que passam pela rede é constante. Na figura 13 é mostrado o cenário que tem alta vazão no inicio, a qual diminui até a metade da duração do teste, e então, aumenta até chegar a vazão do começo. O ultimo cenário é mostrado na figura 14, onde a vazão é errática, devido ao período de envio de pacotes ser aleatório.

Figura 12 – Cenário 1: RINT quando a vazão é constante.



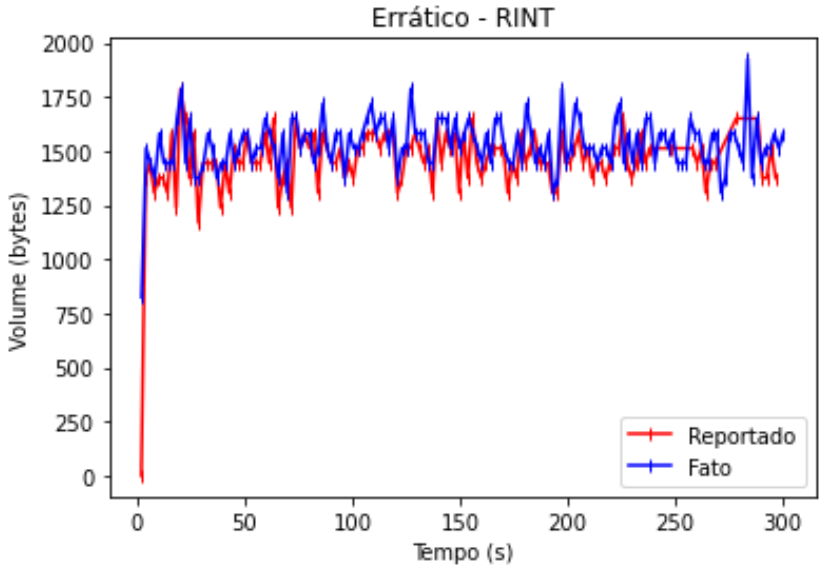
Fonte: Autor.

Figura 13 – Cenário 2: RINT quando a vazão é em formato de 'U'.



Fonte: Autor.

Figura 14 – Cenário 3: RINT quando a vazão é errática.



Fonte: Autor.

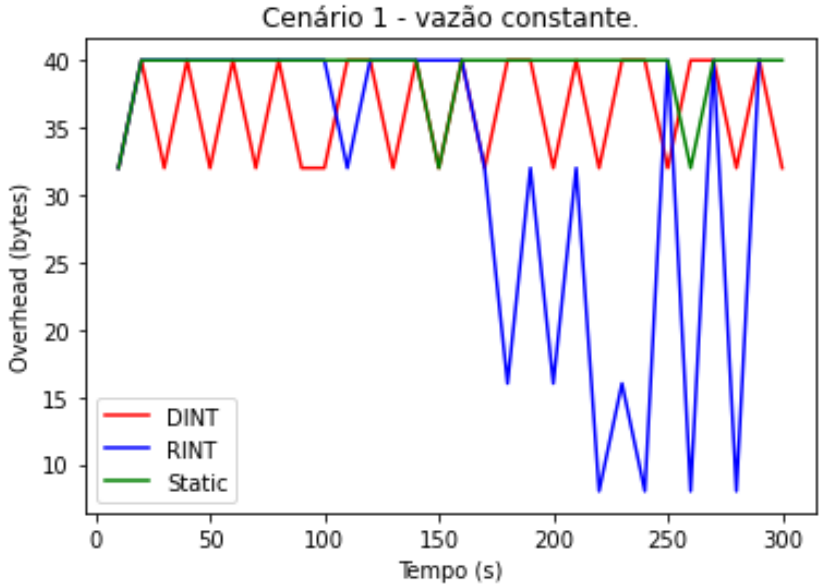
Os demais algoritmos foram testados nos mesmos cenários, a fim de medir precisão, e *overhead* em relação ao tempo, para fins de comparação dos mesmos com o RINT. Na figura 15, são mostrados os volumes agregados por inserção de cabeçalhos de telemetria para cada algoritmo quando a vazão é constante. O segundo cenário é mostrado na figura 16, onde o a vazão diminui e em seguida aumenta. Por fim, a figura 17, mostra o *overhead* dos algoritmos quando a vazão é errática.

A telemetria estática está representada pelo gráfico verde, que se mantém constante ao decorrer dos testes. A linha vermelha representa o DINT, que tem como seu T_min o mesmo valor do período da telemetria estática e por isso, em alguns instantes, tende a voltar ao *overhead* da mesma. Em azul, está representado o *overhead* do RINT, que passa por um período inicial de coleta de informações, evidenciado no inicio dos gráficos, e ao ganhar confiança, reduz a quantidade de cabeçalhos, no entanto, quando um fluxo desconhecido passa pelo roteador, o modo de confiança é desligado e, por consequência, o *overhead* sobe, causando picos ao final do gráfico.

Conforme visto nos gráficos das Figuras 15, 16 e 17, o RINT tem sucesso em reduzir o *overhead* para casos em que há fluxos com comportamentos repetitivos passando pelas fontes de telemetria. Isso acontece porque, no início do período de teste, o RINT está aprendendo e adicionando confiança aos fluxos e, portanto, ainda inserindo telemetria de forma estática, por esse motivo, os gráficos do RINT e do algoritmo estático se sobrepõem. Depois dessa fase inicial, o *overhead* diminui consideravelmente, voltando a subir apenas quando passarem fluxos desconhecidos ou de baixa confiança pelo *switch*.

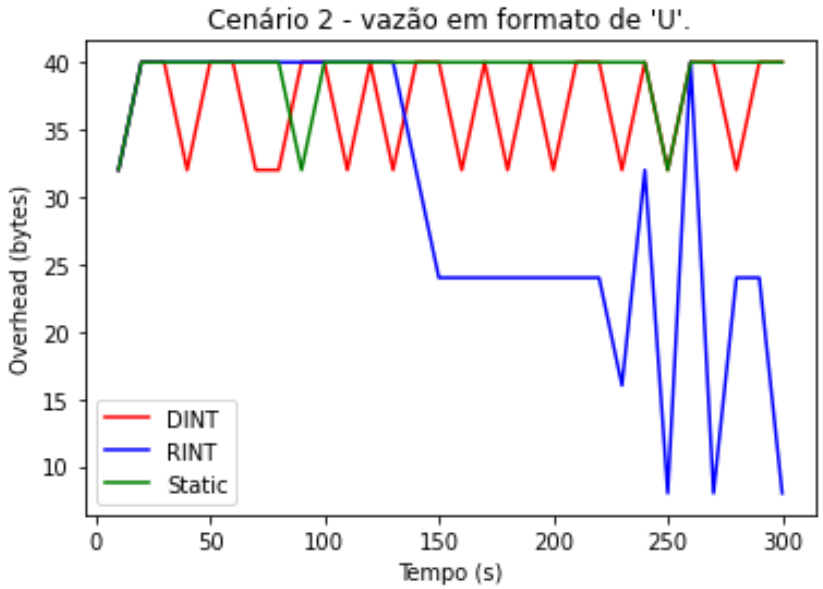
Quanto ao algoritmo estático, seu período de inserção não muda, portanto, o *overhead* não varia de acordo com o tempo. Já no DINT, como seu período varia de acordo com a estabilidade da rede, tendendo a voltar constantemente para seu valor mínimo, por consequência disso, o *overhead* não é reduzido tão consideravelmente quanto o RINT.

Figura 15 – Cenário 1: *overhead* por tempo quando a vazão é constante.



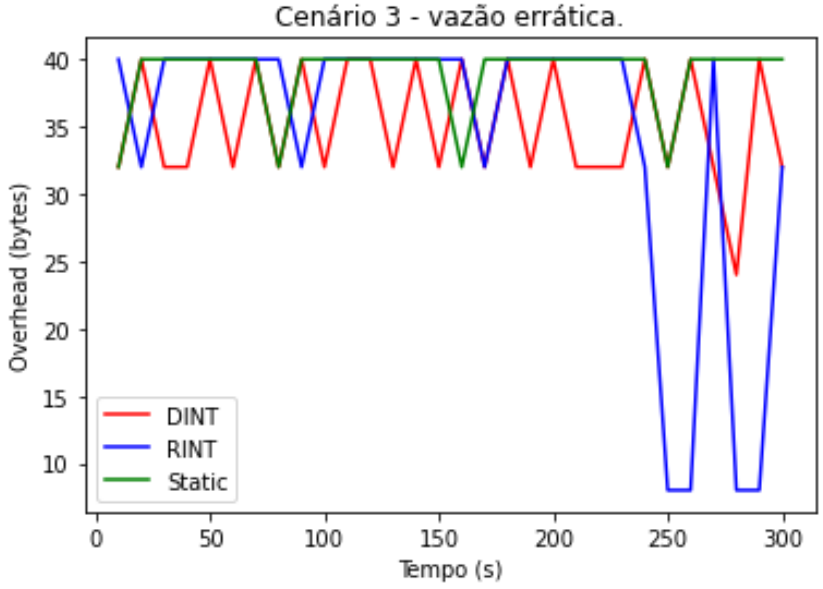
Fonte: Autor.

Figura 16 – Cenário 2: *overhead* por tempo quando a vazão é em formato de 'U'.



Fonte: Autor.

Figura 17 – Cenário 3: *overhead* por tempo quando a vazão é errática.

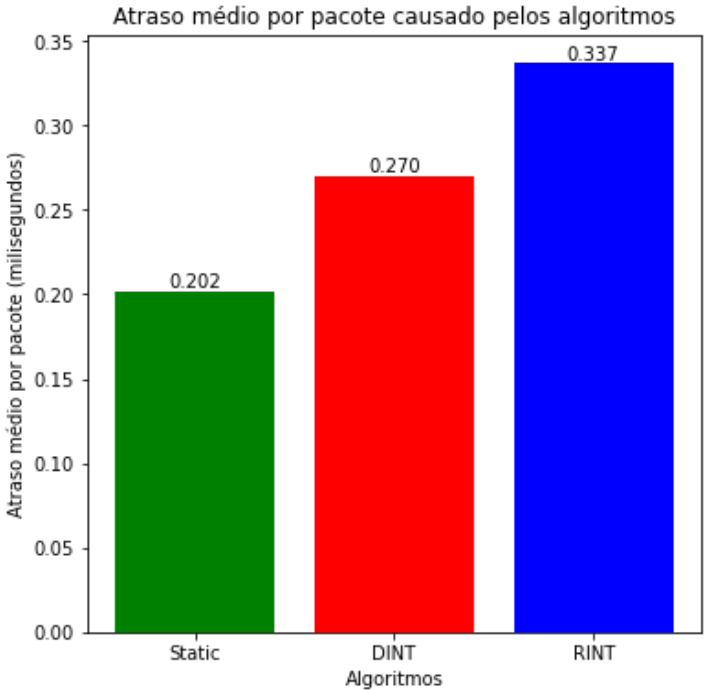


Fonte: Autor.

A partir dos testes feitos utilizando os diferentes cenários em cada algoritmo, foi possível calcular em média o atraso por pacote gerado pelo processamento. Conforme mostrado na figura 18, o atraso adicionado pelo processamento do RINT em relação aos outros algoritmos é considerável, mas não excessivo. Esse atraso adicional é causado pela complexidade do algoritmo em relação aos outros, visto que, o RINT busca na tabela de fluxos recorrentes a entrada referente a cada fluxo. É valido lembrar que, a implementação

foi feita utilizando *python3* e a biblioteca *scapy*, que são excelentes em versatilidade e facilidade de uso, mas deixam a desejar em desempenho, portanto, essas ferramentas, por si só, já geram um atraso considerável no funcionamento do algoritmo.

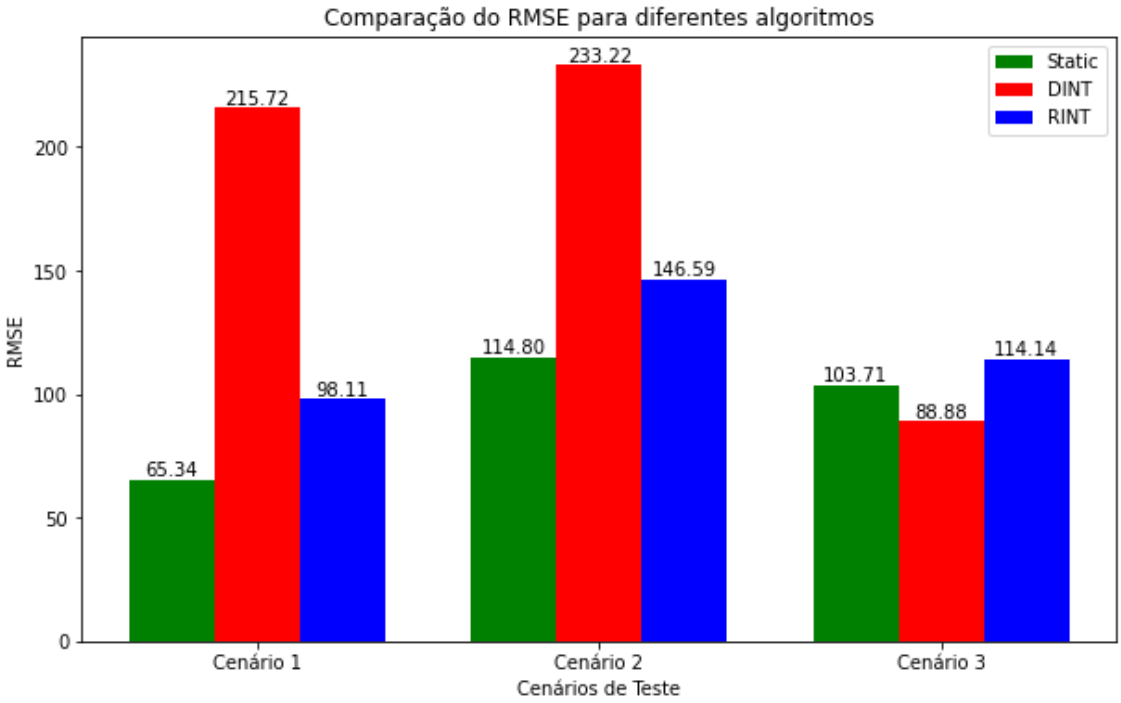
Figura 18 – Atraso médio por pacote causado pelos algoritmos.



Fonte: Autor.

A métrica de RMSE mede a acurácia das informações reportadas pela telemetria em comparação com o tráfego da rede. Dessa forma, o gráfico da Figura 19 evidencia que, dada a presença de fluxos com comportamentos recorrentes, o RINT mantém alta precisão na maioria dos casos quando comparado ao DINT que, possui valores de RMSE altos nos cenários em que a estabilidade da rede é alta, por conta de aumentar seu período de inserção de acordo com a mesma. Já o método estático possui a maior acurácia por conta de não aumentar seu período de inserção, ou seja, ele reporta a maior quantidade de vezes, gerando mais *overhead* mas perdendo o mínimo de precisão, mesmo assim, quando comparado ao método estático, o RINT não apresenta uma perda de acurácia muito considerável.

Figura 19 – Erro quadrático médio de cada algoritmo, separado por cenário.



Fonte: Autor.

4.3 DISCUSSÃO DOS RESULTADOS

A partir dos resultados mostrados na Seção 4.2, é possível perceber que enquanto o algoritmo dinâmico funciona bem em redes de vazão estável, o RINT reduz o *overhead* mais drasticamente após seu período de aprendizado. Considerando que as aplicações para o algoritmo analisem a redução de *overhead* a longo prazo, a diferença entre o RINT e os demais algoritmos tende a aumentar ainda mais significativamente.

Já que a perda de precisão do RINT é miníma e o seu potencial em reduzir *overhead* é bem significativo, o algoritmo cumpriu com sucesso os requisitos do trabalho (discutidos na Seção 3.2) e mostra-se viável para o uso de redes que tenham a característica de possuir fluxos com comportamentos repetitivos. Além disso, a predição de comportamento foi implementada com sucesso e cria inúmeras possibilidades de uso, podendo, por exemplo, ser utilizada para prevenir a rede quanto a fluxos que requerem algum tratamento especial.

5 CONCLUSÃO

Para garantir a confiabilidade e eficiência das redes de computadores, é necessário que o controlador de rede tenha informação do estado da mesma. Para esse fim, existem diversas técnicas que reportam esse estado, sendo a mais eficiente delas, a telemetria em banda que promove alta granularidade de informações reportadas em tempo real, cujo principal problema é o *overhead*.

Este trabalho teve por objetivo investigar esse problema, propondo o RINT, um algoritmo que utiliza a recorrência de comportamentos em fluxos de redes para alterar o período de inserção de telemetria. Para isso, utilizou-se uma métrica de confiança para predizer se fluxos conhecidos devem apresentar o comportamento esperado. A partir disso, aumenta-se o período de inserção, reduzindo o *overhead* sem causar perda significativa na precisão das informações reportadas. Portanto, os requisitos do trabalho foram a redução do *overhead*, baixa perda de acurácia das informações e alterar a métrica reportada, para que fluxos conhecidos carreguem uma predição de seu comportamento.

Os testes foram feitos em comparação com o DINT e um algoritmo estático, onde demonstraram que o RINT cumpre com sucesso os requisitos técnicos, mostrando-se funcional para redes que possuem fluxos com comportamentos recorrentes. Além disso, a predição de comportamento de fluxos é funcional e possui inúmeros usos para o operador de rede. Em contrapartida, os testes mostraram uma limitação da linguagem escolhida, pois os valores de atraso foram prejudicados por conta do *python3* e a biblioteca *scapy*, que por si só geram atraso de processamento.

Para trabalhos futuros, seria interessante alterar e testar diferentes métodos para substituir a inserção estática que é utilizada quando o RINT está inserindo cabeçalho no momento de aprendizagem, ou seja, em fluxos de baixa confiança. Outra mudança interessante seria testar métodos mais avançados de predição e alteração dos limiares de precisão e confiança utilizados nos testes. Além disso, sugere-se a implementação do algoritmo em linguagens de mais alto desempenho, por exemplo, em *P4*, que teria mais aplicabilidade prática, e testes de atraso mais precisos.

REFERÊNCIAS

ALQAHTANI, J.; ALANAZI, S.; HAMDAOUI, B. Traffic behavior in cloud data centers: A survey. In: IEEE. **2020 International Wireless Communications and Mobile Computing (IWCMC)**. Limassol, Cyprus, 2020. p. 2106–2111.

BRUM, H. B.; SANTOS, C. R. P. D.; FERRETO, T. C. Providing fine-grained network metrics for monitoring applications using in-band telemetry. In: IEEE. **2023 IEEE 9th International Conference on Network Softwarization (NetSoft)**. Madrid, Spain, 2023. p. 116–124.

____. Providing fine-grained network metrics for monitoring applications using in-band telemetry. In: IEEE. **2023 IEEE 9th International Conference on Network Softwarization (NetSoft)**. Madrid, Spain, 2023. p. 116–124.

CASE FEDOR, S. . D. **RFC 1098: Simple Network Management Protocol (SNMP) — rfc-editor.org**. 1989. https://www.rfc-editor.org/rfc/rfc1098#section-3. [Accessed 13-07-2024].

CHOWDHURY, S. R. et al. Payless: A low cost network monitoring framework for software defined networks. In: IEEE. **2014 IEEE Network Operations and Management Symposium (NOMS)**. Krakow, Poland, 2014. p. 1–9.

CHOWDHURY, S. R.; BOUTABA, R.; FRANÇOIS, J. Lint: Accuracy-adaptive and lightweight in-band network telemetry. In: IEEE. **2021 IFIP/IEEE International Symposium on Integrated Network Management (IM)**. Bordeaux, France, 2021. p. 349–357.

KANDULA, S. et al. The nature of data center traffic: Measurements & analysis. In: **Proceedings of the 9th ACM SIGCOMM Conference on Internet Measurement**. New York, NY, USA: Association for Computing Machinery, 2009. (IMC '09), p. 202208. ISBN 9781605587714. Disponível em: <https://doi.org/10.1145/1644893.1644918>.

KIM, C. et al. In-band network telemetry via programmable dataplanes. In: ACM SIGCOMM. 2015. Disponível em: <https://api.semanticscholar.org/CorpusID:15782087>.

KIM, Y.; SUH, D.; PACK, S. Selective in-band network telemetry for overhead reduction. In: IEEE. **2018 IEEE 7th International Conference on Cloud Networking (CloudNet)**. Tokyo, Japan, 2018. p. 1–3.

KUMAR, A. et al. Data streaming algorithms for efficient and accurate estimation of flow size distribution. In: **Proceedings of the Joint International Conference on Measurement and Modeling of Computer Systems**. New York, NY, USA: Association for Computing Machinery, 2004. (SIGMETRICS '04/Performance '04), p. 177188. ISBN 1581138733. Disponível em: <https://doi.org/10.1145/1005686.1005709>.

MACEDO, M. W. M. d. **Universidade Federal do Rio Grande do Norte: Estudo inicial sobre o uso do protocolo gRPC na comunicação interna entre serviços do TCE/RN repositorio.ufrn.br**. 2024. https://repositorio.ufrn.br/handle/123456789/57871. [Accessed 13-07-2024].

MANDAL, K. **In-band Network TelemetryMore Insight into the Network**. 2020. https://www.keysight.com/blogs/en/tech/traf-gen/2020/11/18/in-band-network-telemetrymore-insight-into-the-network. [Accessed 09-08-2024].

PAN, J. et al. Orchestrating probabilistic in-band network telemetry for network monitoring. In: IEEE. **2021 7th International Conference on Computer and Communications (ICCC)**. Chengdu, China, 2021. p. 441–446.

QUEIROZ, L. **Tráfego de Internet no Brasil cresce 43novo recorde - Capital Digital — capitaldigital.com.br**. 2021. https://capitaldigital.com.br/trafego-de-internet-no-brasil-cresce-43-e-bate-novo-recorde/. [Accessed 03-06-2024].

SILVA, M. et al. Predicting elephant flows in internet exchange point programmable networks. In: AINA. Springer, Cham, 2020. p. 485–497. ISBN 978-3-319-98284-7.

SILVA, M. V. B. da et al. Ideafix: Identifying elephant flows in p4-based ixp networks. In: IEEE. **2018 IEEE Global Communications Conference (GLOBECOM)**. Abu Dhabi, United Arab Emirates, 2018. p. 1–6.

SUNSHINE, C. A. A brief history of computer networking. **Computer Network Architectures and Protocols**, Springer, p. 3–6, 1989.

TAN, L. et al. In-band network telemetry: A survey. **Computer Networks**, v. 186, p. 107763, 2021. ISSN 1389-1286. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1389128620313396>.

YU, M. Network telemetry: Towards a top-down approach. **SIGCOMM Comput. Commun. Rev.**, Association for Computing Machinery, New York, NY, USA, v. 49, n. 1, p. 1117, feb 2019. ISSN 0146-4833. Disponível em: <https://doi.org/10.1145/3314212.3314215>.