UNIVERSIDADE FEDERAL DO PAMPA

Rumenigue Hohemberger

Orquestração Escalável da Coleta de Dados de Telemetria *In-band* em Planos de Dados Programáveis

Alegrete
2022

Rumenigue Hohemberger

Orquestração Escalável da Coleta de Dados de Telemetria *In-band* em Planos de Dados Programáveis

Dissertação de Mestrado apresentada ao Curso de Mestrado Profissional em Engenharia de Software da Universidade Federal do Pampa como requisito parcial para a obtenção do título de Mestre em Engenharia de Software.

Orientador: Prof. Dr. Marcelo Caggiani Luizelli

Coorientador: Prof. Dr. Fábio Diniz Rossi

Alegrete
2022

Ficha catalográfica elaborada automaticamente com os dados fornecidos
pelo(a) autor(a) através do Módulo de Biblioteca do
Sistema GURI (Gestão Unificada de Recursos Institucionais) .

H718o Hohemberger, Rumenigue
Orquestração escalável da coleta de dados de telemetria in-band em planos de dados programáveis / Rumenigue Hohemberger.
97 p.

Dissertação(Mestrado)-- Universidade Federal do Pampa, MESTRADO EM ENGENHARIA DE SOFTWARE, 2022.
"Orientação: Marcelo Caggiani Luizelli".

1. Monitoramento de Rede. 2. Telemetria de Rede in-band. 3. Problema do Plano de Orquestração de Telemetria. I. Título.

RUMENIGUE HOHEMBERGER

ORQUESTRAÇÃO ESCALÁVEL DA COLETA DE DADOS DE TELEMETRIA IN-BAND EM PLANOS DE DADOS PROGRAMÁVEIS

Dissertação apresentada ao Programa de Pós-Graduação em Engenharia de Software da Universidade Federal do Pampa, como requisito parcial para obtenção do Título de Mestre em Engenharia de Software.

Dissertação defendida e aprovada em: 16 de fevereiro de 2022.

Banca examinadora:

_____

Prof. Dr. Marcelo Caggiani Luizelli
Orientador
UNIPAMPA

_____

Prof. Dr. Fábio Diniz Rossi
Corientador
IFFAR

_____

Prof. Dr. Guilherme da Cunha Rodrigues
IFSUL

Prof. Dr. Arthur Francisco Lorenzon

UNIPAMPA

Assinado eletronicamente por **MARCELO CAGGIANI LUIZELLI**, **PROFESSOR DO MAGISTERIO SUPERIOR**, em 18/03/2022, às 10:11, conforme horário oficial de Brasília, de acordo com as normativas legais aplicáveis.

Assinado eletronicamente por **Fábio Diniz Rossi**, **Usuário Externo**, em 18/03/2022, às 11:37, conforme horário oficial de Brasília, de acordo com as normativas legais aplicáveis.

Assinado eletronicamente por **Guilherme da Cunha Rodrigues**, **Usuário Externo**, em 18/03/2022, às 11:53, conforme horário oficial de Brasília, de acordo com as normativas legais aplicáveis.

Assinado eletronicamente por **ARTHUR FRANCISCO LORENZON**, **PROFESSOR DO MAGISTERIO SUPERIOR**, em 18/03/2022, às 12:56, conforme horário oficial de Brasília, de acordo com as normativas legais aplicáveis.

A autenticidade deste documento pode ser conferida no site https://sei.unipampa.edu.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **0757699** e o código CRC **AD47D89E**.

Este trabalho é dedicado a minha família, pelo incentivo e apoio.

AGRADECIMENTOS

Inicialmente agradeço ao meus pais, Ivo e Lenir, pelos ensinamentos e pelo exemplo. Além do incentivo constante pela busca de conhecimento e formação. Aos meus irmãos, pela parceria e pelo compartilhamento de ideias, mesmo que por muitas vezes à distância.

A minha esposa Luana, obrigado pela paciência, apoio e companheirismo durante estes 2 anos, juntamente com as minhas filhas Manuela e Mariana, sempre estiveram ao meu lado. Vocês fazem parte desta conquista.

Gostaria de agradecer ao meu orientador Prof. Dr. Marcelo Caggiani Luizelli e coorientador Prof. Dr. Fábio Diniz Rossi, por terem me orientado e me conduzido no decorrer deste trabalho. Fico imensamente grato pelo compartilhamento de conhecimentos, pelo apoio e pela paciência. Também agradeço os professores Dr. Arthur Francisco Lorenzon e Dr. Guilherme da Cunha Rodrigues, pelas contribuições a este trabalho.

Aos colegas do Laboratório de Otimização de Sistemas (LOS) e colegas do Programa de Pós-Graduação em Engenharia de Software (PPGES), agradeço pelo companheirismo, pelas discussões e pelas ajudas durante o mestrado.

Por fim, agradeço a todos que de alguma forma contribuíram diretamente ou indiretamente no desenvolvimento e na concretização deste trabalho.

"A single packet history can be the “smoking gun” that reveals why, how, and where a network failed"(HANDIGOL et al., 2014)

RESUMO

A telemetria de rede *in band* (INT) é um paradigma emergente de monitoramento de rede. Ao coletar itens de telemetria de baixo nível em tempo real, a INT pode aumentar substancialmente a visibilidade de toda a rede - permitindo, por exemplo, detecção de eventos transientes como *micro-bursts*. Estudos recentes têm focado em *(i)* desenvolver mecanismos INT para aumentar a visibilidade em toda a rede; e *(ii)* projetar novas soluções de monitoramento. No entanto, pouco foi feito para coordenar o processo de coleta de itens de telemetria neste novo paradigma. Isso é particularmente desafiador, dependendo de quais itens de telemetria de rede são coletados, isso pode degradar a visibilidade de toda a rede em termos de consistência/*freshness*. Pesquisas anteriores se concentraram em orquestrar de forma otimizada a coleta de estatísticas de telemetria *in-band* da rede, fornecendo pouca ou nenhuma escalabilidade para ser aplicada em ambientes realistas. Neste trabalho, *(i)* formalizamos teoricamente o Problema do Plano de Orquestração de Telemetria de Rede *In-band* e propomos um modelo de orquestração baseado em aprendizado de máquina e *(ii)* abordamos a limitação de escalabilidade das abordagens existentes ao propor um procedimento de busca local. Os resultados mostram que nossa abordagem de modelo de orquestração baseada em aprendizado de máquina supera as heurísticas do estado da arte por um fator de até 8x em relação ao número de anomalias de rede identificadas, por exemplo. Finalmente, nosso algoritmo supera as soluções de orquestração do estado da arte - (`Gather` e `Balance`) por um fator de 2 em relação ao número de requisitos de aplicativos de monitoramento atendidos, enquanto mantém a solução perto do ideal (menos de 5%) - exigindo alguns segundos para executar.

Palavras-chave: Monitoramento de Rede. Telemetria de Rede *in-band*. Problema do Plano de Orquestração de Telemetria.

ABSTRACT

In-band network telemetry (INT) is an emerging network monitoring paradigm. By collecting low-level telemetry items in real time, INT can substantially enhance network-wide visibility - allowing, for example, timely detection problems such as micro-burst. Recent studies have focused on *(i)* developing INT mechanisms to increase network-wide visibility; and *(ii)* to design new monitoring solutions. However, little has been done to coordinate the process of collecting telemetry items in this new paradigm. This is particularly challenging because depending on which network telemetry items are collected, it might degrade network-wide visibility in terms of consistency/freshness. Previous researches have focused on optimally orchestrating the collection of in-band network telemetry statistics from the network, providing little or none scalability to be applied in realistic environments. In this work, we *(i)* theoretically formalize the In-band Network Telemetry Orchestration Plan Problem and propose a machine learning based orchestration model and *(ii)* tackled the scalability limitation of existing approaches by proposing an iterated local search procedure. Results show that our machine learning based orchestration model approach outperforms state-of-the-art heuristics by up a factor of 8x with respect to the number of network anomalies identified, for instance. At least, our algorithm outperforms state-of-the-art orchestration solutions by a factor of 2 with respect to the number monitoring applications requirements satisfied, while keeping the solution close to the optimum (less than 5%) – demanding a few seconds to execute.

Key-words: Network Monitoring. In-band Network Telemetry. Telemetry Orchestration Plan Problem.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS

RndFit Random-Fit

LISTA DE SIGLAS

API *Application Programming Interface*

BGP *Border Gateway Protocol*

DDQN *Double Deep Q-Network*

DFS *Depth-first Search*

FDC Função de Distribuição Cumulativa

FPGA *Field-programmable Gate Array*

GRASP *Greedy Randomized Adaptive Search Procedure*

INT *In-band Network Telemetry*

INTO *In-band Network Telemetry Orchestration*

IP *Internet Protocol*

IPFIX *IP Flow Information eXport*

LCR Lista de Candidatos Restritos

LRC *Least-Recently Collected*

MTU *Maximum Transmission Unit*

ONF *Open Networking Foundation*

ONOS *Open Network Operating System*

P4 *Programming Protocol-independent Packet Processors*

PISA *Switches Protocol-Independent Switch Architecture*

PL *Programação Linear*

PLI *Programação Linear Inteira*

PLIM *Programação Linear Inteira Mista*

P^{2}INT *Probe Planning for In-Band Network Telemetry*

RFC *Request for Comments*

RR Round-robin

SDN *Software defined networking*

SNMP *Simple Network Management Protocol*

SYN *synchronize*

TCP *Transmission Control Protocol*

LISTA DE SÍMBOLOS

α, β e ρ Parâmetros de ajuste espacial/temporal

$\mathcal{P}t$ Caminho de rede simples

D Dispositivos de encaminhamento programáveis

F Conjunto de fluxos de rede ativos

G Infraestrutura de rede

K_f Quantidade de espaço disponível para incorporar itens de telemetria em um fluxo de pacote

L Conjunto de dispositivos interconectados

M Conjunto de aplicações de monitoramento

P Item coletado

R_m^t Dependências temporais

R_m Subconjunto de itens de telemetria

V Conjunto de itens de telemetria

W Período de tempo

SUMÁRIO

1 INTRODUÇÃO

1.1 Contexto e Motivação

A telemetria de rede *in-band* (INT) é uma técnica de monitoramento de plano de dados programáveis que fornece maior visibilidade do estado dos dispositivos de rede para operadores (LIU et al., 2018). Esta maior visibilidade ajuda os operadores de infraestruturas a identificar e solucionar problemas de curta duração (por exemplo: contenção de fluxo de rede, *micro-burst* e desequilíbrio de carga (TAMMANA; AGARWAL; LEE, 2018).

Exemplos de informações que fornecem esta maior visibilidade incluem estados internos dos dispositivos de encaminhamento, como níveis de ocupação das filas e métricas de desempenho da rede/dispositivo como o tempo de processamento de um determinado pacote no plano de dados. Os pacotes de produção ou pacotes de sondagem são responsáveis por conduzir a coleta desses dados de telemetria à medida que são roteados junto a infraestrutura de rede. Esses pacotes carregam instruções específicas de telemetria, que são interpretadas por dispositivos de rede, configurando dispositivos de encaminhamento para coletar os estados de rede necessários (LIU et al., 2018). Em algum local da rede, os dados de telemetria coletados são extraídos e enviados para aplicações de monitoramento, que processam e, finalmente, reagem a eventos de rede espúrios. Por exemplo, suponha que um determinada aplicação de monitoramento esteja incumbida de prever o congestionamento da rede, exigindo a coleta de ocupação da fila e tempo de processamento dos dispositivos de encaminhamento (ou seja, requisitos de monitoramento).

O ecossistema de rede com o emprego da INT apresenta novas oportunidades e desafios ao propor o incremento de visibilidade da rede. Como o próprio nome se refere, a visibilidade está relacionada a visualização da rede, permitindo ao administrador ou gerente um olhar onipresente, visualizando cada evento que ocorre com cada pacote e em cada dispositivo (HANDIGOL et al., 2014), onde cada evento é transformado em um item de telemetria e coletado. Apesar da possibilidade da coleta destes itens, existem limitações quanto a coleta e ao seu processamento, sendo: i) A necessidade de fluxos de produção e capacidade residual para o transporte destes itens. Em redes de computadores a quantidade de fluxos é limitada, bem como a capacidade disponível no fluxo de transportar itens; ii) O *overhead* imposto aos dispositivos de encaminhamento, devido a grande quantidade de itens de telemetria e a sua manipulação; iii) A coleta destes itens pode competir com recursos empregados no encaminhamento de pacotes e assim penalizar estes fluxos. Desta forma, a coleta de itens de telemetria de interesse para aplicações de monitoramento surge como uma alternativa.

A coleta de itens de telemetria se dá em função das necessidades de monitoramento da rede, com o paradigma SDN, aplicações de monitoramento apontam quais itens de telemetria são de interesse e quais restrições devem ser respeitados. Estas aplicação estão

localizadas no topo do ecossistema de rede, assim, temos uma abordagem *top-down* do monitoramento, ou seja de cima para baixo obedecendo a um primeiro momento aos requisitos de monitoramento das aplicações. Estes itens de telemetria são tomados como requisitos de monitoramento das aplicações, uma vez que são necessários para que uma determinada aplicação de monitoramento faça a tomada de decisões e sua inferência.

Aplicações de monitoramento possuem diferentes requisitos de monitoramento, onde cada requisito é pertencente a uma dependência, sendo elas a dependência espacial ou temporal. A primeira remente a obrigatoriedade da coleta de itens de telemetria em um único dispositivo de encaminhamento. Por outro lado, a dependência temporal coleta de itens de telemetria em uma determinada faixa de tempo ou prazo. Para a abordagem do problema, em um primeiro momento modelamos a solução através da utilização da Programação Linear Inteira (PLI) com o objetivo de maximizar as dependências satisfeitas. Em um segundo momento, empregamos aprendizado de máquina para a clusterização de dependências com K-*Means* e de pesos arbitrários auto-atualizáveis afim de instrumentar de forma dinâmica coleta de itens de telemetria.

Outra preocupação existente leva em conta a complexidade do ecossistema de rede. A complexidade está associada com a quantidade de dispositivos de rede, fluxos, aplicações de monitoramento e requisitos de monitoramento. Estas características podem mudar rapidamente, podemos citar como exemplo o incremento repentino de tráfego. Assim, faz-se necessário que a coleta de itens de telemetria de interesse seja capaz de acompanhar estas variações, ou seja, escalável. Para atacar a complexidade propomos uma abordagem utilizando a Meta-Heurística *Greedy Randomized Adaptive Search Procedure* (GRASP).

O conceito de INT foi criado por iniciativas da indústria [1] e acadêmicas, principalmente devido aos rápidos avanços em dispositivos de rede programáveis, onde podemos citar como exemplo, *SmartNICs* e linguagem de domínio específico de descrição de rede, neste caso a linguagem P4 (BOSSHART et al., 2014a). Esforços de pesquisa recentes (PAN et al., 2019; HOHEMBERGER et al., 2019; LIU et al., 2018; MARQUES et al., 2019) deram os primeiros esforços para orquestrar a coleta de dados de telemetria de rede *in-band*. Por exemplo, Liu et al. (LIU et al., 2018), Pan et al. (PAN et al., 2019) e Castro et al. (CASTRO et al., 2021) concentram seus esforços para realizar a telemetria de rede usando pacotes de sondagem (isto é: *probing packets*), enquanto Marques et al. (MARQUES et al., 2019) e Hohemberger et al. (HOHEMBERGER et al., 2019) concentram na incorporação de dados de telemetria em pacotes de rede de produção. Outros autores propõe otimizações relacionadas ao emprego de fluxos de produção como Song et al. (SONG et al., 2020) e Chowdhury, Boutaba e François (CHOWDHURY; BOUTABA; FRANçOIS, 2021)

A orquestração da telemetria de rede *in-band* é um problema desafiador por dois motivos. Primeiro, dependendo de quais itens de telemetria de rede são coletados, isso

[1] Por exemplo: Barefoot Advanced Network Telemetry e Broadcom Trident 3

pode degradar a visibilidade da infraestrutura obtida em termos de cobertura e atualização (MARQUES et al., 2019). Em segundo lugar, e mais importante, dependendo de como os dados de telemetria são coletados, o processo de coleta pode afetar o desempenho de aplicações de monitoramento de rede (HOHEMBERGER et al., 2019).

Desta forma, este trabalho dá os primeiros passos na direção de incrementar a visibilidade de infraestruturas de rede por meio da telemetria *in-band*. Para isso, emprega-se o tráfego de produção na coleta de informações de telemetria que satisfaçam os requisitos das aplicações de monitoramento em infraestruturas de rede. Estes requisitos podem variar de acordo com as objetivos de monitoramento de cada aplicação e com o comportamento da própria rede.

1.2 Objetivos e Contribuições

Apesar dos esforços citados acima, a maioria das soluções de orquestração negligenciam os requisitos existentes (espaciais e temporais) por aplicações de monitoramento e oferecem pouca capacidade de atender o crescimento da rede, ou seja de promover escalabilidade para ser aplicada em ambientes de produção. Desta forma, este trabalho tem como objetivo:

> *Orquestrar de forma escalável e dinâmica a aquisição de dados de telemetria in-band em infraestruturas programáveis.*

As principais contribuições deste trabalho se desdobram em:

- Definição e modelagem do problema de orquestração da coleta de dados de telemetria *in-band*(HOHEMBERGER et al., 2019).
  - Introduziu-se neste trabalho o problema de orquestração da coleta de dados de telemetria *in-band* baseado em aprendizado de máquina. A ideia principal consiste em dinamicamente orientar a aquisição de dados por meio de um mecanismo de aprendizagem. Para resolver este problema, formalizou-se o problema através de Programação Linear Inteira Mista (PLIM).
- Projeto de heurística escalável para o problema (HOHEMBERGER et al., 2020).
  - Introduziu-se uma heurística baseada em pesquisa local iterada para resolver o problema de forma a produzir soluções de alta qualidade para infraestruturas de rede em larga escala.

Ambas contribuições publicadas estão disponíveis nos apêndices A e B, respectivamente.

1.3 Organização

Este trabalho está organizado como segue. No Capítulo 2, apresenta-se a fundamentação teórica sobre monitoramento de infraestruturas redes considerando a programabilidade e monitoramento de infraestruturas, conceitos relacionados além de fornecer uma visão geral da literatura relacionada. No Capítulo 3, introduz-se o problema de orquestração de telemetria com sua respectiva modelagem matemática, apresenta-se a construção da solução ótima e solução escalável. Por fim, no Capítulo 4, concluí-se este trabalho com indicações de trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA E TRABALHOS RELACIONADOS

Neste capítulo são apresentados conceitos chave, no escopo desta dissertação, relacionados a programabilidade de infraestruturas de rede e ao monitoramento baseado em telemetria *in-band* em planos de dados programáveis. Posteriormente, apresenta-se uma visão geral dos trabalhos que compõem o estado da arte relacionados ao problema de orquestração da coletados de informações de telemetria *in-band*.

2.1 Fundamentação Teórica

2.1.1 Programabilidade em Infraestruturas de Rede

Apesar da expansão e popularização da Internet, as tecnologias empregadas nas infraestruturas de rede ainda são rígidas e limitadas à soluções *ad hoc*. Isso se dá, em parte, pela heterogeneidade de equipamentos, soluções e de fabricantes empregados nas infraestruturas. As infraestruturas de rede atuais são em grande parte compostas por um conjunto de caixas pretas interligadas – o que limita a operação e o desenvolvimento das mesmas devido a funcionalidades pré-estabelecidas. Essas limitações contribuem para a ossificação da Internet, isto é, a dificuldade de inovação no núcleo das infraestruturas. A programabilidade de infraestrutura de redes se apresenta, como uma alternativa para endereçar as limitações mencionadas e promover o maior avanço tecnológico. Neste contexto, o paradigma de redes baseadas *Software Defined Networking* (SDN) surgiu como um alternativa para prover maior programabilidade às infraestruturas.

O paradigma SDN baseia-se em quatro pilares (KREUTZ et al., 2014): (i) desacoplamento dos planos de dados e de controle; (ii) unidade de controle centralizada e externa, responsável por prover a abstração da rede; (iii) decisões de encaminhamento com base nos fluxos; (iv) rede programável de acordo com as proposições do controlador e das aplicações. O desacoplamento permite ao operador da rede uma independência do hardware e consequentemente do fabricante. Isto ocorre devido a possibilidade de programar o plano de dados de acordo com as diferentes necessidades dos serviços suportados ou do comportamento esperado da rede. Já o controle de uma série de dispositivos pode ser efetuado por uma única estrutura centralizada. Para além disso, é possível realizar as escolhas do sistema operacional de rede ou controlador SDN, como *Open Network Operating System* (ONOS), Beacon, RYU e POX. Isto também é possível em termos de protocolo a ser utilizado na comunicação entre o controlador e os dispositivos do plano de dados. Dentre as soluções existentes, pode-se mencionar o Openflow (MCKEOWN et al., 2008), sendo este um protocolo aberto. Muitas destas soluções são fomentadas por consórcios de empresas atuantes no ramo de telecomunicações e tecnologias, tendo o intuito propor a padronização e de disseminar estas soluções, como a *Open Networking Foundation* (ONF) e *Linux Foundation*.

A Figura 1 ilustra a arquitetura de infraestruturas baseadas em SDN. A arquite-

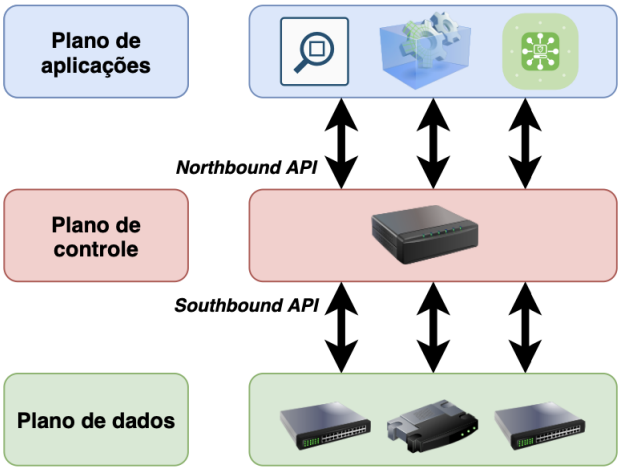


Figura 1 – Arquitetura de rede SDN. Fonte: adaptado de (ALVES, 2017) e (MOURA, 2015)

tura é composta por três camadas ou planos distintos: (i) plano de aplicações, composta pelas aplicações que utilizam os serviços da infraestrutura SDN. Esta camada é interligada ao plano de controle pela *Application Programming Interface* (API) aberta *Northbound*; (ii) plano de controle, responsável pelo controle centralizado da infraestrutura. Esta camada é responsável por implementar a inteligência da infraestrutura e também por oferecer as APIs abertas *Northbound* e *Southbound*; (iii) plano de dados, concentra no substrato físico da infraestrutura (por exemplo, *switches*, roteadores, *SmartNICs*).

Inicialmente, os dispositivos presentes no plano de dados apresentavam somente suporte ao protocolo OpenFlow (ou outra interface *southbound*) para permitir a interação entre as aplicações do plano de controle e do plano de dados. Apesar de permitir a programabilidade de infraestruturas, a programabilidade até então estava limitada às primitivas implementadas pelos planos de dados e suportado pelo protocolo OpenFlow. Por exemplo, para um desenvolvedor implementar um novo protocolo de enlace (por exemplo, um novo *Ethernet*) e avaliá-lo em uma infraestrutura de rede, havia a necessidade de se projetar equipamentos de encaminhamentos com suporte ao novo protocolo, além de estender a própria definição do protocolo OpenFlow. Como é possível observar, apesar da programabilidade fornecida pelo protocolo OpenFlow, esta ainda estava limitada às funcionalidades implementadas nos dispositivos de encaminhamento.

Para permitir a programabilidade do plano de dados e quebrar a dependência de hardwares específicos (e da interface *southbound*), nos últimos anos surgiram vários esforços para prover abstrações para permitir a rápida programação e prototipação de planos de dados. Um desses esforços recentes que tem recebido atenção da indústria e da academia é a linguagem P4 (BOSSHART et al., 2014b). P4 é um acrônimo para *Programming Protocol-independent Packet Processors* e é uma linguagem de programação utilizada para programar o plano de dados de dispositivos de encaminhamento (P4 Language Consor-

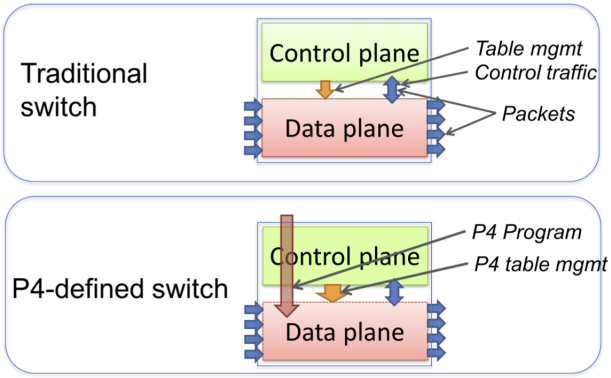


Figura 2 – Switch tradicional vs switch programável Fonte: (P4 Language Consortium, 2020)

tium, 2020) como, por exemplo, SmartNICs, *Field Programmable Gate Array* (FPGA), e roteadores. Em um dispositivo de encaminhamento tradicional, o plano de dados é definido de acordo com as premissas propostas pelo fabricante. Já o plano de controle controla o plano de dados gerenciando as entradas nas tabelas de encaminhamento. Por outro lado, dispositivos programáveis possuem duas características básicas: (i) o plano de dados é definido no momento em que o código P4 compilado é carregado no dispositivo (por exemplo, por meio de um *firmware*) e está representado pela seta vermelha longa na Figura 2; e (ii) o plano de controle se comunica com o plano de dados utilizando o mesmo canal, porém o programa P4 podem ser modificados a qualquer momento pelo controlador. As modificações necessárias são aplicadas através do compilador P4 integrado ao controlador através de uma API. Estas características e comparações podem ser visualizadas na Figura 2, contendo um switch tradicional e um programável.

2.1.2 Monitoramento de Infraestruturas

2.1.2.1 Monitoramento Passivo e Ativo

A partir do monitoramento de infraestruturas de rede é possível a identificação de problemas e anomalias e, desta forma, garantir a continuidade de recursos a serviços de rede. O monitoramento consiste em identificar, quantificar e reportar eventos ocorridos promovendo a visibilidade da rede. O monitoramento de redes tradicionais consiste em aplicações distribuídas do tipo gerente-agente. Estas aplicações tem a finalidade de coletar métricas dos agentes e concentrá-las em um gerente (SILVA et al., 2018). O gerente pode ser visto como parte de uma ferramenta de gerenciamento, assim o gerente é responsável pela coleta de dados e comunicação com os agentes dos dispositivos monitorados. A comunicação é realizada de acordo com as definições de objetos ou eventos monitorados. Para objetos, as consulta são feitas através de *polling*. O *polling* consiste no envio de requisição sobre o estado destes objetos de tempos em tempos. Já para eventos, o agente toma a iniciativa de reportar ao gerente a ocorrência de um determinado evento. Uma

visão desta estrutura é apresentada na Figura 3. Para que ocorra a comunicação entre gerente-agente é necessária a utilização de um protocolo, de modo a padronizar o formato e as trocas de mensagens. Exemplos de protocolos amplamente utilizados incluem, por exemplo, o protocolo SNMP[1].

Administradores ou operadores de rede contam com um arcabouço de soluções a serem utilizadas de acordo com a infraestrutura a ser monitorada, construindo uma base de conhecimento a ser empregada na solução de eventos indesejados. Além da atuação de forma reativa, também é de interesse do operador atuar da foma pró-ativa com o intuito de mitigar ou evitar danos ao funcionamento da rede, e desta forma o monitoramento realizado de forma eficiente subsidia as tomadas de decisões (OLIVEIRA et al., 2017). O monitoramento de rede pode ser dividido basicamente em duas grupos: passivo e ativo.

O monitoramento passivo consiste no processo de observação de pacotes e fluxos, sem interferência na estrutura ou encaminhamento dos mesmos. Os dados gerados são baseados na observação e leitura de estados, servindo como entrada para a aplicação de monitoramento. Os pontos de coletas podem variar de acordo com os dispositivos disponíveis. Além disso, o dispositivo deve ser capaz de executar a captura dos estados desejados. Problemas relacionados a esta abordagem estão ligados a quantidade e a granularidade de informações capturadas em infraestruturas de grande porte e ao acompanhamento de fluxos de pacotes em um determinado percurso.

No monitoramento ativo temos a injeção de tráfego especializado junto aos fluxos de produção, de forma a percorrer um determinado caminho. Os *probes* ou sondas são pacotes inseridos e transmitidos na infraestrutura de maneira proposital de forma a trafegar entre dois pontos da infraestrutura. A intenção desses pacotes consiste em extrair métricas de desempenho como latência, *jitter* e taxas de perdas de pacotes. As sondas possuem como vantagem a verificação rápida de um caminho entre dois pontos, passando eventualmente por diversos domínios. Entretanto, a utilização do monitoramento ativo pode ocasionar a degradação do desempenho da infraestrutura rede, já que os *probes* podem competir por recursos destinados aos fluxos de produção (como por exemplo, filas de encaminhamento e processamento) e até mesmo ocasionando sobrecarga das ferramentas de monitoramento. Por outro lado a utilização de forma insuficiente pode resultar na captura de apenas uma fração dos estados presentes na rede em um determinado momento, revelando apenas uma visão superficial do estado da infraestrutura.

No monitoramento passivo, o roteador realiza a coleta de dados e os armazena de forma temporária, sendo limitado a um intervalo máximo de 15 segundos de inatividade, por um período máximo de 30 minutos de atividade no caso de fluxos com grande duração ou no momento em que um fluxo é extinto por uma aplicação. Assim, há necessidade de exportar os dados armazenados para um coletor. Caso contrário, estes dados podem ser perdidos ou competir por recursos do dispositivo, como memória e processamento. Outros

[1] Definido pela RFC 1157

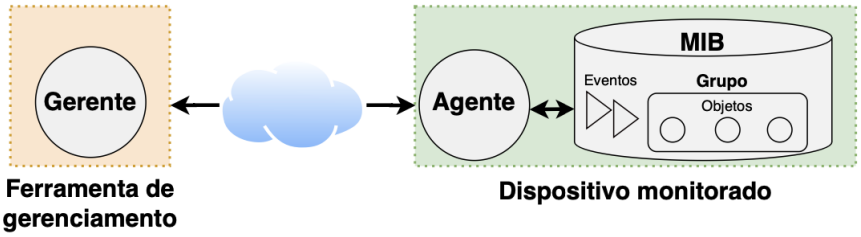


Figura 3 – Interação gerente-agente Fonte: adaptado de (SANTOS, 2015)

dados também são exportados juntamente com estas informações, como *timestamps* de incio e final de cada fluxo, quantidade de *bytes* e protocolos do cabeçalho IP. Através destes dados é possível obter, por exemplo, o *troughput* em uma determinada interface em um espaço de tempo, ou até mesmo os protocolos mais utilizados. Estas informações ajudam na caracterização de tráfego e consequentemente numa melhor visualização da rede. O processo de coleta descrito acima é empregado pelo NetFlow, originalmente implementado pela CISCO *Systems*. O NetFlow, apesar de conseguir coletar métricas importantes para a visibilidade da rede, pode tornar-se computacionalmente caro para o dispositivo, especialmente em situações onde o trafego é intenso. Para evitar estes problemas causados pela exaustão do roteador, a Cisco propôs uma varição denominada "*Sampled NetFlow*". Esta variação permite que ao invés de capturar todo o tráfego que passa pelo dispositivo ou pela interface, sejam capturadas apenas algumas amostras. A seleção destas amostras pode ser feita de duas formas: através de filtros pré-determinados como portas e protocolos, ou através de fatias de tempo. Desta forma, as métricas colhidas são apenas uma estimativa do fluxo real. Apesar das melhorias implementadas, a definição correta da taxa de amostragem é considerada um desafio, uma taxa muito esparsa pode se afastar da representação do estado real dos fluxos ou no caso de recursos disponíveis, pode-se desperdiçar recursos por não serem utilizados.

Posteriormente, para o melhoramentos dos mecanismos existentes de monitoramento, definiu-se o *IP Flow Information eXport* (IPFIX). O objetivo do IPFIX consiste em melhorar a forma da coleta de pacotes e dados, proporcionado uma maior flexibilidade. Além disto, o protocolo passou a ser aberto, eliminando incompatibilidades entre diferentes fabricantes. Uma das principais diferenças em relação ao NetFlow está na flexibilização do conceito relacionado ao fluxo, sendo definido como uma sequencia de pacotes com propriedades comuns que passam através de um dispositivo de rede em um determinado espaço de tempo (IETF, 2013). Isto permite agrupar pacotes de acordo com especifidades definidas pelo administrador de rede, como um campo do cabeçalho, aplicação, todos os pacotes em uma determinada faixa de tempo ou de um determinado IP e porta. Outra mudança proposta está na utilização de "*templates*" definidos para a troca de mensagens entre os processos. Cada mensagem enviada para o processo seguinte tem a capacidade de carregar um novo "*template*" ou conjunto de dados. A Figura 4 ilustra a

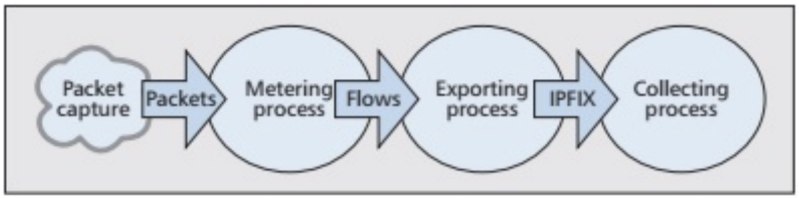


Figura 4 – Arranjo dos processos na estrutura IPFIX Fonte: (Trammell; Boschi, 2011)

arquitetura do IPFIX em que é possível observar os três tipos de processos: (i) *metering process*, o (ii) *exporting process*; e o (iii) *collecting process*.

Em paralelo ao IPFIX e ao NetFlow, definiu-se o sFlow. Este protocolo obtém estatísticas da rede utilizando a coleta de amostras de pacotes. Posteriormente, os dados relevantes são enviados a um servidor responsável para análise e armazenamento. A estrutura do sFlow envolve dois componentes: (i) Agente sFlow – dispositivos que coletam as informações dos pacotes e encaminhar para um coletor, e (ii) Coletor sFlow – dispositivo central que captura as informações oriundas dos agentes. O agente sFlow pode operar de duas formas, de acordo com a quantidade de pacotes comutados e unidades de tempo. Na primeira é definida o filtro a ser aplicado para selecionar o fluxo que se deseja monitorar, a partir daí são coletados pacotes de acordo com a taxa de amostragem desejada. Para os pacotes capturados são extraídos os cabeçalhos, a fim de coletar dados como *flags* de controles, endereço de destino e origem. Na segunda forma são definidas fatias de tempo onde deseja-se monitorar as métricas de uma determinada interface, por exemplo, taxa de erros e taxa de ocupação.

2.1.2.2 Monitoramento baseado em Telemetria *In-Band*

A telemetria *in-band* surgiu recentemente a partir de linguagens de programação para o plano de dados como, por exemplo, P4 e da adoção de *hardware* programáveis. O conceito de monitoramento baseado em INT está relacionado com a utilização do fluxo de produção para o transporte de informações de telemetria ou metadados dos dispositivos de encaminhamento ao passar pelos dispositivos. A Figura 5 apresenta como ocorre a inserção de elementos e extração INT em um pacote no momento em que trafega pela rede. O dispositivo programado coleta o dado de interesse, como ocupação da fila ou latência, insere em um campo pré-definido no pacote, por exemplo, *payload* ou campos não utilizados, e esta operação é repetida por diversos dispositivos enquanto o pacote trafega pela rede, sendo extraída em um determinado momento. Após a extração dos dados são enviados para a ferramenta de gerenciamento ou para um coletor INT. Além da utilização do tráfego de produção, também é possível a utilização de pacotes específicos para percorrer um determinado caminho em uma rede com instruções sobre métricas a serem coletadas. Esta técnica é similar à utilização de *probes* ativos.

Através do emprego do monitoramento baseado em INT, pode-se garantir uma

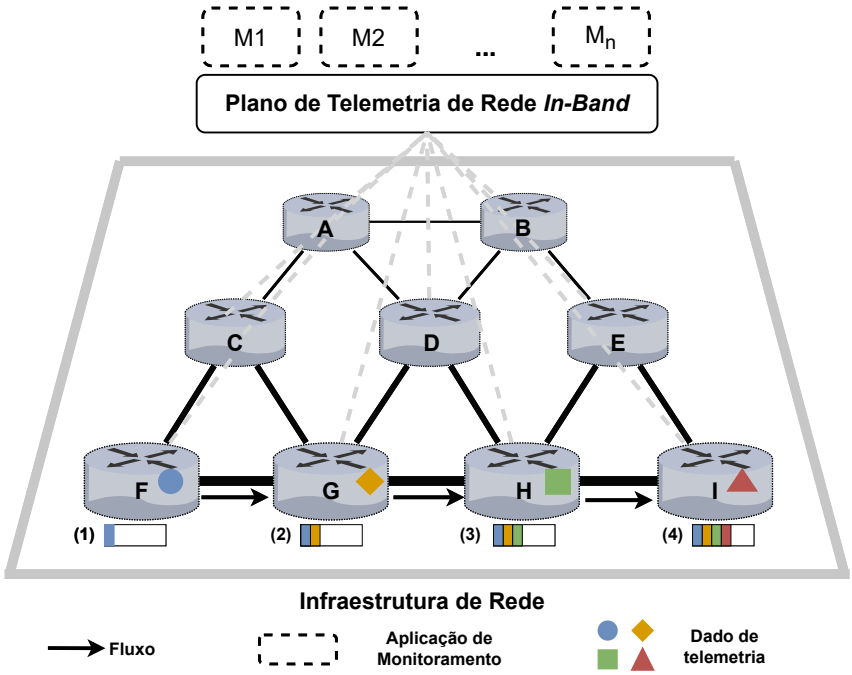


Figura 5 – Arquitetura de coleta de itens de telemetria Fonte: Do autor

maior atualização das informações coletadas do plano de dados, uma vez que o dado é extraído no exato momento em que um determinado estado está a ocorrer, favorecendo a detecção de eventos transientes e de curta duração como *microbursts* – além de permitir a resposta imediata ao evento anômalo. Desta forma, a telemetria *in-band* apresenta vantagens sobre métodos tradicionais de monitoramento.

2.2 Programação Linear Inteira

A Programação Linear Inteira (PLI) é uma técnica de modelagem amplamente difundida na área de pesquisa operacional, principalmente pela aplicabilidade nos mais diferentes problemas de otimização. O modelo de programação linear inteiro é representado por um conjunto de equações lineares (MARINS, 2011) e uma determinada função objetivo – a qual maximiza ou minimiza o valor dado as restrições postas pelo conjunto de equações lineares inteiras.

De acordo com Barboza et al. (2015), um problema de PLI é composta por:

1. A função objetivo apresentada como uma função linear inteira representada pelas variáveis de decisão, refletindo o problema que pretende-se resolver;

2. Restrições do problema, descritas como interdependências entre as variáveis de decisão e demais limitadores, podendo ser representadas por equações e/ou inequações.

3. Variáveis de decisão, que compõe tanto a função objetivo quanto as restrições, e assumem valores inteiros.

A formalização matemática do problema, também conhecida como forma padrão é definida como segue:

Minimizar ou maximizar $\mathrm{z} = f(x_1,x_2,\ldots,x_n) = \mathrm{c}_1x_1 + c_2x_2 + \cdots + c_nx_n$

Sujeito a:

$$\begin{array}{llll} a_{11}x_1 & +a_{12}x_2 & +\cdots+a_{1n}x_n & \{\leq,=,\geq\}b_1 \\ a_{21}x_1 & +a_{22}x_2 & +\cdots+a_{2n}x_n & \{\leq,=,\geq\}b_2 \\ \vdots & \vdots & \vdots & \vdots \\ a_{m1}x_1 & +a_{m2}x_2 & +\cdots+a_{mn}x_n & \{\leq,=,\geq\}b_m \\ & & & x_1,x_2,\ldots,x_n \geq 0 \end{array}$$

Onde:

- f é a função objetivo;
- x_i são as variáveis de decisão, $i = 1, 2...n$;
- a_{ij} é a constante ou coeficiente da i-ésima restrição da j-ésima variável, $i = 1, 2, ..., m, j = 1, 2, ..., n$;
- b_i é o termo independente ou quantidade de recursos disponíveis da restrição, $i = 1, 2, ..., m$;
- c_i é a constante ou coeficiente da j-ésima variável da função objetivo, $i = 1, 2, ..., n$;

A definição de restrições e variáveis de decisão tem grande importância, uma vez que elas delimitam a região admissível de busca. Nesta região concentram-se as soluções viáveis, e entre elas está a solução ótima da função objetivo (MARINS, 2011).

Os valores assumidos pelas variáveis podem indicar a qual sub classificação o problema pertence. Desta forma, quando todas as variáveis do problema assumem valores inteiros, temos um problema (PLI), caso contrário há um problema de Programação Linear Inteira Mista (PLIM). Ainda, quando todos os valores das variáveis são reais, temos um Programa Linear (PL).

O Método Simplex é um método iterativo que parte de uma solução factível inicial buscando a cada iteração uma nova solução factível, também denominada solução básica factível adjacente (BELFIORE; FÁVERO, 2013). Se a solução encontrada possui um melhor custo em razão da função objetivo, toma-se como nova solução. Cabe ressaltar, que para ser considerada solução básica factível adjacente há a necessidade de atender as restrições do problema e não negatividades. Abaixo apresentamos a sequência de passos para a solução de problemas de maximização proposto por Belfiore e Fávero (2013):

1. Realizar a transformação do problema para a forma padrão, ou seja transformar inequações em equações utilizando uma variável de folga (como por exemplo, $x_1 \leq 8$) transformada em $x_1 + x_2 = 8$ através da inserção da variável x_2);

2. Encontrar uma solução básica factível inicial, podendo ser escolhida entre as várias soluções da região factível;

3. Testar a otimalidade da solução - leva em conta se há uma solução adjacente que incremente o valor da função z, ou seja melhor.

   De modo iterativo, determinar uma solução básica factível adjacente melhor considerando z. Para isto: i) Determinar a variável não básica que passará para o conjunto de variáveis básicas. Deve se escolhida a variável de maior incremento no valor de z; ii) Escolher a variável básica que passará para o conjunto de variável não básica. Deve ser removida a variável que limita o crescimento da variável não básica selecionada no passo anterior a entrar na base; iii) Aplicar o Método de Eliminação Gauss-Jordan. Com o novo sistema de equações cada equação deve possuir apenas uma variável básica com coeficiente igual a 1, cada variável básica deve aparecer em apenas uma equação, e a função objetivo deve ser escrita em função das variáveis não básicas, de forma que os valores das novas variáveis básicas e da função objetivo z podem ser obtidos diretamente e o teste de otimalidade possa ser verificado. Resolver o sistema de equações recalculando os valores da nova solução básica adjacente.

2.3 K-*Means*

Coleções de dados podem ser classificados ou organizados de acordo com características semelhantes ou dissemelhantes, originando agrupamentos. Esta técnica é denominada Análise de Agrupamentos ou Análise de *clusters* (SILVA et al., 2020).

Dentro da Análise de Grupamentos podemos destacar a técnica K-médias ou K-*means*. Esta técnica emprega a estratégia de agrupamento por partição, permitindo o agrupamento de dados em K agrupamentos ou *clusters*, cada um ao redor de um centroide de acordo com uma determinada características (TAN; STEINBACH; KUMAR, 2013).

Como primeiro passo, deve-se definir quantos *clusters* serão criados, sendo atribuído este valor a k. Então, serão escolhidos k centroides e ligados a eles as demais instâncias de acordo com uma característica semelhante pré-definida, como por exemplo a distância euclidiana. Desta forma teremos a formação de *clusters*. Conforme são acrescentados novas instâncias aos *clusters*, são efetuadas novas adequações de qual instância possui um melhor posicionamento (centro) dentro do *cluster* considerando as instancias que o rodeiam. Esta etapa consiste em um processo iterativo sendo delimitado pela quantidade de iterações ou até que seja atingida um estabilidade na solução obtida - inexistência de migração de instâncias de um centroide para outro, por exemplo (WITTEN et al., 2005).

2.4 Trabalhos Relacionados

Nesta seção é apresentada uma revisão do estado da arte sobre monitoramento de infraestruturas de rede utilizando INT. Ressalta-se as abordagens utilizadas, ferramentas e heurísticas empregados durante o monitoramento de rede. Um compilado destas soluções é apresentado na Tabela 1.

Zhu et al. (2015), apresenta um sistema de monitoramento para *datacenters* denominado Everflow. Para rastrear pacotes pelas rede, usa-se *"match and mirror* ". Esta abordagem pode ser implementada até mesmo em switches de "prateleira"desde que consigam aplicar um determinado conjunto de filtros para a análise de tráfego tanto de cabeçalhos como *payload*. O tráfego selecionado é espelhado para servidores dedicados a busca de padrões, diminuindo o *overhead* no *switch*. Para captura de tráfego foram definidas 3 regras: i) Captura de um conjunto de pacotes durante um determinado percurso; ii) Captura de pacotes marcados com o bit de *debug* no cabeçalho. iii) Captura de pacotes de protocolo de tráfego, como por exemplo *Border Gateway Protocol* (BGP). Os servidores responsáveis pela análise recebem o tráfego espelhado através de um balanceador de carga, que além de controlar o fluxo de entrada para o servidor também é responsável por encaminhar os pacotes capturados de um fluxo para o mesmo servidor. Cada servidor possui dois tipos de estados: tracejado de pacote e contador. A primeira se detém a manter em um tabela um identificador formado pela tupla de identificação de um fluxo, campo de identificação do datagrama IP e demais itens contendo informações, como endereços de IP do *switch* onde foi efetuada a captura do pacote e *timestamp*. Apesar da grande quantidade de pacotes capturados, são armazenados apenas aqueles onde são encontradas anomalias, como *loop* e *drop*. No estado contador são contabilizados dados tanto de carga de link com latência. Outro recurso disponível no Everflow é a função de utilizar *probes* para a validação e coleta de métricas.

Liu et al. (2016) propõe a solução denominada UnivMon, com o objetivo de promover o monitoramento de fluxos de rede mantendo características como generalidade e precisão, as quais são mencionadas como antagônicas, onde ocorre o ganho de uma em detrimento de outra. A generalidade se refere a cobertura das diferentes métricas a serem quantificadas de acordo com as solicitações das aplicações de monitoramento. Já a precisão está ligada a qualidade das informações quantificadas. Assim, são propostos 3 três requerimentos a serem satisfeitos pela solução: i) Fidelidade pra um grande conjunto de aplicações, mesmo sem saber quais métricas serão de interesse da aplicação, deve ser capaz de manter um nível elevado de precisão; ii) Abstração de um grande switch para monitoramento, o UnivMon deve prover uma abstração geral da rede as aplicações dispostas sobre o plano de controle; iii) Implementação viável, a abordagem deve ser possível de ser utilizada em switches programáveis através da Utilização de P4. A Figura 6 apresenta uma visão da organização e da interação entre o plano de dados e o plano de controle. No plano de dados temos em execução uma primitiva de propósito geral, responsável pelo

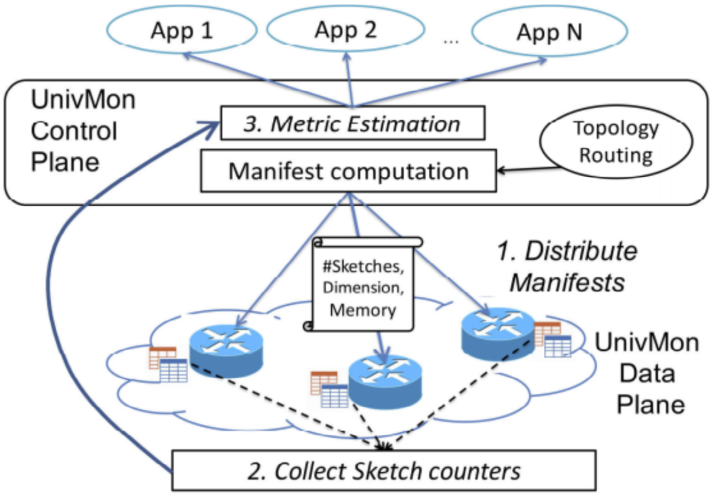


Figura 6 – Arquitetura do *Framework* UnivMon Fonte:(LIU et al., 2016)

processamento de fluxos de pacotes, coleta de estatísticas e armazenamento destes dados em estruturas associadas a cada fluxo. O Plano de controle coleta as estatísticas do plano de dados e estima as métricas de interesse das aplicações. Desta forma, o processo de monitoramento consiste de dois estágios respectivamente, um online e outro *offline*.

Tammana, Agarwal e Lee (2016) propõe PathDump, um depurador de rede com design minimalista, focado em um conjunto de problemas onde a ideia central consiste em rastrear a trajetória de pacotes para identificar anomalias de rede utilizando *switches* de prateleira com pequenas modificações, como operações condicionais e capacidade de inserir dados no cabeçalho de um pacote. Outras características do funcionamento são a captura do pacote e a extração dos dados para alimentar uma base de dados utilizada por um mecanismo de consultas. A extração dos dados de telemetria é feita por um dispositivo de borda, onde armazena um conjunto de informações como, uma lista de fluxos de entrada, um mapa contendo todos os dispositivos e seus respectivos identificadores, e por último, arquivos de configuração da rede contendo as políticas de encaminhamento. O espaço disponível no cabeçalho do pacote é um fator limitante, já que a capacidade é totalmente preenchida com 8 saltos, armazenando apenas o código identificador do *switch*. Ao chegar ao destino é necessário o encaminhamento da informação diretamente ao ponto de coleta.

Sivaraman et al. (2017) apresenta HashPipe, um algoritmo para a detecção de *Heavy hitters* em plano de dados programáveis. HashPipe implementa uma *pipeline* de tabelas *hash* para armazenar contadores para fluxos pesados. Para isto, são mantidos os identificadores de fluxo e contadores de fluxos pesados no *switch*, em um pipeline de tabelas *hash*. Quando um pacote entra no primeiro estágio do *pipeline*, seu contador é atualizado ou inicializado se há um *hit* ou um slot vazio. Se houver um *miss*, uma nova chave é inserida na tabela. Em todos os estágios seguintes, a chave e o contador acompanham o pacote. Devido a grande quantidade de informações a serem armazenadas e a quantidade limitada de memória, chaves com contadores leves são removidas para

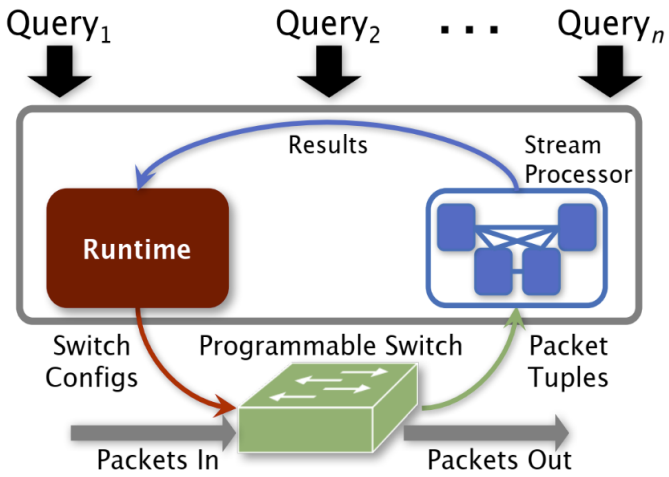


Figura 7 – Arquitetura Sonata Fonte: (GUPTA et al., 2017)

liberar espaço de armazenamento a ser ocupada com contadores pesados. Experimentos realizados em um enlace de *backbone*, apresentam menos de 5% de falsos negativos e 0.001% de falsos positivos enquanto reportou cerca de 300 fluxos pesados com apenas 4500 contadores (menos de 80 KBytes) de um universo composto por 400000 fluxos, mostrando-se eficiente detecção de *Heavy hitters* e utilização de recursos disponíveis no *switch*.

Gupta et al. (2017) apresenta Sonata, apresenta uma interface de alto nível para efetuar consultas sobre itens de telemetria. Para tornar possível a execução das consultas, são feitos ajustes quanto a utilização dos recursos disponíveis no plano de dados juntamente com alterações no processamento de fluxos, aliado a isto iterativas melhorias nos subconjunto de tráfego que satisfazem a consulta, desta forma torna-se possível uma melhor utilizações dos recursos disponíveis. Uma vez escolhida a consulta, a mesma é compilada e enviada para os *switches* da rede. Afim de diminuir o tempo de resposta das consultas, elas são divididas e executadas de forma dinâmica, podendo ser parte no switch ou no processador de fluxo, esta função é de responsabilidade do planejador de consultas, a arquitetura utilizada por ser vista na Figura 7. Muitas das restrições impostas ao particionamento e execução das consultas estão relacionadas a limitação dos recursos disponíveis nos *switches Protocol-Independent Switch Architecture* (PISA), como *parsing* de campos de cabeçalhos, realização de ações em pacotes, armazenamento de estados em memória e a execução destas operações em um número limitado de estágios. O custo incrementa com o número de campos a serem extraídos do pacote e de acordo com a profundidade da árvore.

Tammana, Agarwal e Lee (2018) vê a utilização dos dispositivos finais como uma opção para realizar o monitoramento de rede, uma vez que estes dispositivos possuem disponibilidade de recursos, como memória e processamento. A solução proposta denominada SwitchPointer, visa a integração de parte do processamento efetuado no dispositivo

final ao *pipeline* de processamento de pacotes do *switch*, por mais que isto possa geral um *overhead* adicional. A possibilidade de realizar o processamento está relacionada, a programação das rotinas independentemente de um hardware específico. Ao contrário de algumas soluções que armazenam as informações de tráfego para diagnosticar problemas, SwitchPointer armazena ponteiros para os dispositivos finais onde está armazenado os dados de telemetria. Desta forma, ao detectar uma anomalia, é possível consultar os dados armazenados em diferentes dispositivos e recuperar dados para depurar o evento. É necessário incluir duas informações no cabeçalho dos pacotes, a primeira é a trajetória que é percorrida fim-a-fim, composto por um conjunto de *switches* (como por exemplo identificador do switch), a segunda é a informação de época, quando um pacote atravessa estes *switches*. Para agilizar tráfego das informações entre os switches de rede e dispositivos finais, pode-se utilizar enlaces dedicados para o envio de informações.

NetVision é uma solução proposta por Liu et al. (2018) como uma alternativa ao *overhead* imposto a rede, resultante da coleta de itens de telemetria utilizando fluxo de produção, além da capacidade residual escassa para a alocação de itens. Para resolver este problema, são utilizadas *probes* com a finalidade de coletar itens de telemetria de interesse. O caminho é a ser percorrido é inserido no cabeçalho bem como os itens coletados. Outros objetivos propostos são a escalabilidade e cobertura. A primeira se dá com o disparo de *probes* de acordo com a necessidade de itens de monitoramento. Já a cobertura, ocorre pela inclusão de itens de encaminhamento no plano de monitoramento.

Yang et al. (2018) apresenta Elastic Sketch, uma solução genérica voltada para o monitoramentos de rede e adaptação de acordo com as características de tráfego. Elastic Sketch é composto por duas partes: uma pesada e outra leve. Esta nomenclatura se deve ao tratamento aplicado a fluxos levando em consideração o seu tamanho, assim, fluxos grandes ou *elephant flows* são tratados pela parte pesada e fluxos pequenos ou *mouse flows* pela parte leve. Para tornar Elastic Sketch elástico são empregadas 3 filosofias: i) Para se tornar adaptativo a quantidade de largura de banda disponível, os *sketches* podem ser comprimidos e fundidos; ii) Quando a quantidade de pacotes aumenta consideravelmente, a parte leve auxilia o processando de fluxos pesados, deixando de lado fluxos leve; iii) Conforme o o número de fluxos pesados varia e o avanço é desconhecido, a o tamanho da memória disponível para a parte pesada pode ser aumentado. Para tornar a solução genérica, são descartadas informações sobre pacotes oriundos de pequenos fluxos por serem inúteis e ocuparem memória. Ainda, é independente em termos de plataforma, pois está disponível tanto em software quanto em hardware.

Pan et al. (2019) apresenta um *framework* denominado INT-*path* com o objetivo de extrair de forma eficiente o estado do tráfego na rede. Esta motivação está ligada a dois fatores sendo: i) permitir ao controlador uma visão global da rede e ii) o gerenciamento de rede automatizado (como por exemplo aprendizado de máquina) requer dados e *feedback* de forma oportuna. O problema é atacado usando a filosofia de dividir para conquistar,

sendo uma parte desacoplando a solução em um mecanismo de roteamento e, outra a política de geração de caminho de roteamento. Baseado no mecanismo de roteamento, são utilizadas duas políticas de planejamento para gerar e distribuir múltiplos caminhos não sobrepostos afim de cobrir toda a rede. O primeiro é baseado em *Depth-first Search* (DFS), o segundo, baseia-se no caminho de Euler. A abordagem proposta codifica o estado do tráfego de rede em uma séria de imagens de bitmap, a qual permite utilizar técnicas avançadas de reconhecimento de padrões, ajudando a automatizar a rede no que diz respeito a monitoramento e resolução de problemas. Os agentes INT não conseguem controlar o roteamento das *probes*, ficando a cargo das tabelas de roteamento. Todos os dispositivos de rede são consultados periodicamente durante o monitoramento do tráfego, isto pode acarretar *overhead* sobe a rede: i) as *probes* ocupam uma fração da largura de banda, ii) Agentes INT devem ser criados pra a gerar e coletar *probes* do mais diferentes caminhos, iii) quanto maior a quantidade de agentes INT necessários, maiores serão as penalidades de desempenho sofridas pelo controlador. Outro fator importante é tamanho do caminho a ser percorrido por uma *probes*, caso o caminho seja muito longo, haverá a degradação em função do tempo considerando caminhos curtos em relação a caminhos longos.

A otimização dos recursos de rede para INT é problema abordado por Marques et al. (2019), também descrito como *In-band Network Telemetry Orchestration* (INTO). São propostas duas heurísticas, *Gather* e *Balance* para minimizar o *overhead* enquanto mantém-se a qualidade na obtenção dos dados de monitoramento. A heurística *Gather*, tem como estratégia minimizar o número de fluxos usados para transportar dados de telemetria. O pior caso de complexidade computacional é dado por $O(n^2 \cdot m + n \cdot m^2)$, onde n é o número de interfaces em I e m é o número de fluxos em F. Portanto, o algoritmo é executado em tempo polinomial para o número de interfaces e fluxos na rede.

A heurística *Balance* por sua vez se esforça para minimizar o número máximo de itens de telemetria a serem transportados por qualquer fluxo único. No pior caso a complexidade desta a complexidade é dada por $O(n^2 + m^2)$, onde n é o número de interfaces em I e m é o número de fluxos em F.

Castro et al. (2020) utiliza *probes* para a coleta de itens de telemetria, estes fluxos são roteados pela rede por um caminho pré-definido, organizado em ciclos. Entretanto, caso um ou mais dispositivos venham a sofrer uma falha, ocorrerá a interrupção de um ou mais caminhos de coleta utilizados, implicando no comprometimento do ciclo. Desta forma, o autor propõe *Patcher*, um planejador tolerante a falhas. Para manter a integridade da coleta de itens de telemetria pelas *probes*, e consequentemente dos ciclos de monitoramento, é efetuado o rearranjo das *probes*. Por exemplo, caso um dispositivo sofra uma falha catastrófica, acarretará no comprometimento de um ou mais caminhos de roteamento. É, portanto, necessário a readequação dos fluxos que realizam a coleta de itens de telemetria. Caso não exista um fluxo vizinho pré-existente capaz de atender a esta

demanda, um novo *probe* é criado e os itens de telemetria associados a ele. São preteridos fluxos com o menor caminho possível. Também são levados em consideração critérios de cobertura, capacidade de transporte dos fluxos, capacidade do ciclo e conectividade.

Fast-INT é um *framework* leve proposto por Yang et al. (2020), atua no plano de dados e tem com o objetivo de coletar comportamentos anômalos que venham a ocorrer rede. A escolha dos itens de telemetria a serem coletados é feita através do algoritmo *Double Deep Q-Network* (DDQN) levando em consideração as modificações de rede. A estrutura de decisão recebe como entrada um o estado da rede e retorna uma ação a ser aplicada no ambiente. DQN utiliza redes neurais multi-*layers* para revisar a função estado-ação em *Q-learning*. A inclusão do agente se dá em decorrência de funções a serem executadas, como aplicar ações, checá-las e por fim atualizar parâmetros do modelo. Outra ferramenta utilizada é um inspetor de pacotes INT, com a função de coletar informações em um determinado ponto e focar diretamente nos sintomas e eventos, assistindo a predição de rede. A informação recolhida é enviada para o módulo de recompensas que avalia os metadados INT retornado recompensas e ajudando a manter o modelo atualizado.

Para Suh et al. (2020) a linguagem P4 surge como possibilidade para a coleta de itens de telemetria baseado em amostragem. São destacadas duas estratégias: baseada em eventos e taxas. A primeira tem como finalidade coletar métricas sobre eventos que venham a ocorrer em um dispositivo, como por exemplo o tamanho da fila de encaminhamento. Assim que a fila atinge um valor pré-determinado, são inseridas informações nos pacotes de fluxo. Já a estratégia baseada em taxas, insere itens INT e um determinado conjunto de pacotes que é roteado por este nó.

Song et al. (2020) aborda o problema causado pela grande quantidade de dados de telemetria gerados e coletados, causando *overhead* sobre os dispositivos de rede e também ao *southbound* - quando enviados ao controlador. O foco do trabalho está em reduzir a coleta de itens de telemetria desnecessários e contribuir para o decremento da quantidade de itens coletados. Para isto, é proposto INT-*filter*, um arquitetura ligada ao plano de dados e plano de controle com o intuito de selecionar e monitorar partes da rede com base em um mecanismo de predição. Os mecanismos de predição se baseiam em métodos de regressão linear e não linear, de acordo com o histórico da janela de dados. Outra característica está na possibilidade de trabalhar em conjunto com outros métodos como (Pan et al., 2019) e (SUH et al., 2020), orientando o mecanismo de aquisição de itens INT.

Castro et al. (2021) aborda a utilização de *probes* para a coleta de itens de telemetria. Entretanto, o seu emprego necessita de coordenação, para que a ocorra a cobertura da rede e a coleta de dados. Como solução pra este problema, é apresentado *Probe Planning for In-Band Network Telemetry* (P^2INT). A evolução da solução pode ser separa em dois momentos, sendo: formalização do problema como um modelo de Programa-

ção Linear Inteira (PLI), consistindo na generalização de dois problemas de otimização, sendo Problema Roteamento em Arcos Capacitado e Problema de Empacotamento. E, por último a introdução de heurística para guiar o modelo PLI. Assim, são maximizadas a utilização de *probes* empregadas para realizar a coleta de itens de telemetria em um determinado ciclo, gerando uma solução ótima.

Chen et al. (2021) aborda o problema do *overhead* gerado pela inserção de dados usados pela INT e emprego da *Segment Routing*, já que ambos utilizam o cabeçalho do fluxo de produção armazenento temporário de dados. Outra preocupação está relacionada a cobertura dos fluxos de rede e dos dispositivos de encaminhamento de interesse. Para atacar este problema é proposto inicialmente uma abordagem através da modelagem do problema com o uso de Programação Linear Inteira. Já a complexidade inicial do problema mitigada pela proposição de um algoritmo recursivo baseado em caminhos ranqueados, capaz de resolver em tempo polinomial.

Buscando o equilíbrio entre visibilidade e *overhead* imposto a rede pela utilização de INT como forma de monitoramento, Chowdhury, Boutaba e François (2021) propõe um mecanismo denominado *Lightweight* INT. O seu objetivo é identificar e filtrar itens de telemetria de acordo com a sua importância. Para isto, sua execução é feita junto ao plano de dados juntamente com uma função predictiva que auxilia na escolha de quais itens devem ser coletados. Esta diminuição na coleta de itens de telemetria leva a uma menor necessidade de processamento por parte dos dispositivos que realizam o encaminhamento e o processamento de pacotes.

Analisando os trabalhos descritos acima, é possível notar a problemática em torno da coleta de itens INT, vindo a ser abordada na literatura por apresentar-se como um problema de relevância para prover visibilidade da rede. Para uma melhor compreensão, a Tabela 1 faz um comparativo entre as características dos trabalhos. A linguagem P4 e o emprego de dispositivos programáveis possibilitaram alternativas na formas de monitoramento tradicionais, proporcionando uma resposta rápida – muito próximo ao ao tempo real, a anomalias.

Alguns denominadores são motivo de preocupação quanto a utilização de INT, como grande quantidade de dados e o *overhead* causado sobre os dispositivos de encaminhamento, roteamento e controlador((SONG et al., 2020); (MARQUES et al., 2019);(LIU et al., 2018); (ZHU et al., 2015)), além de requerer infraestrutura robusta para o tratamento destes dados (ZHU et al., 2015) e até mesmo a utilização de dispositivos finais (TAMMANA; AGARWAL; LEE, 2018).

Buscando diminuir o volume de dados coletadas e direcionando os esforços para identificar quais dados de fato são interessantes a serem coletados, são empregados mecanismos para tal tarefa, como *Match and mirror* (ZHU et al., 2015), interfaces para consultas específicas (GUPTA et al., 2017), telemetria como serviço (LIU et al., 2018), adaptação da ferramenta aos fluxos (YANG et al., 2018), redes neurais (YANG et al.,

2020), amostragem (SUH et al., 2020) e heurísticas especializadas (MARQUES et al., 2019) e Chen et al. (2021).

Outras preocupações estão relacionadas a cobertura da rede, ou seja, garantir que todos os itens de interesse espalhados pela infraestrutura de rede sejam coletados por um fluxo. Os fluxos podem ser especializados como (*probes*) ou de produção, onde utiliza-se o a capacidade residual para armazenamos dos dados. Trabalhos como Castro et al. (2021) e Castro et al. (2020) contribuem na otimização da utilização de *probes* e tolerância a falhas, no caso de interrupções em dispositivos de encaminhamento. Por outro lado, existem esforços para a otimização e orquestração da coleta de dados utilizando fluxos de produção ((Pan et al., 2019); (LIU et al., 2018); (MARQUES et al., 2019); Chen et al. (2021) e Chowdhury, Boutaba e François (2021)). Tal motivação para o emprego está relacionada a dois fatores principais: i) fluxos de produção são resultantes da comunicação da própria rede; ii) a coleta do dado pelo fluxo proporciona uma maior *freshness* em relação a um evento que está a ocorrer no momento em que é coletado. Tomando estes fatores anteriores como ponto de partida, este trabalho busca uma abordagem guiada pelos requisitos das aplicações de monitoramento, evitando assim a coleta de itens de telemetria desnecessários, uma vez que a coleta indiscriminada contribui para a deterioração dos recursos disponíveis. Outros característica que se destaca em relação aos demais trabalhos citados, são as dependências espaciais e temporais dos requisitos de monitoramento. Estes requisitos permitem as aplicações de monitoramento coletar itens de telemetria de domínios específicos, como limitados por um fração de tempo ou espaço. Além disso, contempla a escalabilidade da rede, uma vez que o ambiente dinâmico, onde a quantidade de fluxos, itens de telemetria, requisitos de monitoramento, aplicações de monitoramento e dispositivos de encaminhamento se altera com de acordo com a sua utilização. Este comportamento reforça esta necessidade da da solução abarcar esta característica.

Tabela 1 – Características analisadas nos trabalhos sobre monitoramento de redes utilizando telemetria

Trabalho	*Probes*	Fluxo de produção	Contadores	Característica
Zhu et al. (2015)	x	x		*Match and mirror*
Liu et al. (2016)			x	Online e offline
Tammana, Agarwal e Lee (2016)		x		Offline
Sivaraman et al. (2017)			x	Tabela Hash
Gupta et al. (2017)		x	x	Interface de consulta
Tammana, Agarwal e Lee (2018)		x	x	Dispositivos finais
Liu et al. (2018)	x			Telemetria como serviço
Yang et al. (2018)			x	Adaptação aos fluxos
Pan et al. (2019)		x	x	Imagens de *bitmap*
Marques et al. (2019)		x		Heurísticas
Castro et al. (2020)	x			Tolerância a falhas
Yang et al. (2020)		x		Redes Neurais
Suh et al. (2020)		x		Amostragem flexível
Song et al. (2020)	x			Otimização com predição
Castro et al. (2021)	x			Otimização de *Probes*
Chen et al. (2021)		x		Otimização da cobertura de fluxos
Chowdhury, Boutaba e François (2021)		x		Otimização dos itens coletados
Este trabalho		x		Requisitos de aplicações e escalabilidade

3 ORQUESTRAÇÃO ESCALÁVEL DA COLETA DE DADOS DE TELEMETRIA *IN-BAND*

Neste capítulo é apresentado o problema de orquestração de telemetria *in-band*, modelo e a solução escalável. Mais precisamente na Seção 3.1 apresenta-se uma visão geral do problema, enquanto que na Seção 3.2 modela-se o problema através de Programação Linear Inteira e é apresentada a Solução Ótima com sua avaliação perante as demais soluções do estado da arte. Por fim, na Seção 3.3, apresenta-se a proposta de Heurística GRASP aplicado ao problema e avaliação da solução proposta. Ressalta-se que seção 3.2 e 3.3 são baseadas em baseadas em Hohemberger et al. (HOHEMBERGER et al., 2019), disponível no Apêndice A e Hohemberger et al. (HOHEMBERGER et al., 2020), disponível no Apêndice B, respectivamente.

3.1 Visão geral do problema

A telemetria de rede *in-band* consiste em incorporar informações de telemetria em pacotes de fluxo de produção. A Figura 8 ilustra uma dada infraestrutura de rede com quatro fluxos de rede ativos (denominados, f_1, f_2, f_3 e f_4) sendo roteados por meio de um conjunto de dispositivos de encaminhamento - variando de A a I. O fluxo de rede f_1 pode coletar itens de telemetria de seus dispositivos de encaminhamento $\{A, B, E, I\}$. No topo da rede, existe um conjunto de aplicações de monitoramento especializados $\{M_1, M_2, ..., M_n\}$. Essas aplicações são responsáveis por fazer inferências sobre problemas ou comportamentos específicos da rede (como, identificar congestionamentos de rede ou ataques maliciosos) e eventualmente reagir a eles, exigindo diferentes informações da infraestrutura. Por exemplo, vamos considerar que a aplicação M_1 visa prever e identificar o congestionamento (transiente) da rede. Ele requer fluxos de rede para coletar dos dispositivos de encaminhamento o número de fluxos de rede ativos ($\triangle$), ocupação da fila ($\bigcirc$) e tempo de processamento diretamente dos planos de dados ($\square$). Por sua vez, M_2 identifica ataques de inundação SYN. Para isso, é necessário monitorar constantemente o número de fluxos de rede ativos ($\triangle$) e o número de *handshakes Transmission Control Protocol* (TCP) incompletos ($\lozenge$).

Observe que essas aplicações de monitoramento não estão limitados aos exemplos dados e podem exigir que diferentes subconjuntos de itens de telemetria (cobertura) sejam coletados continuamente em uma determinada taxa (atualização) para que a precisão da inferência possa ser atendida ao longo do tempo.

O Problema do Plano de Telemetria de Rede *In-band* consiste em orquestrar dinamicamente o processo de coleta de informações de rede para maximizar o desempenho da aplicação de monitoramento de rede sem, no entanto, penalizar os fluxos de rede de produção. O problema está longe de ser resolvido de maneira trivial. Isso é devido a

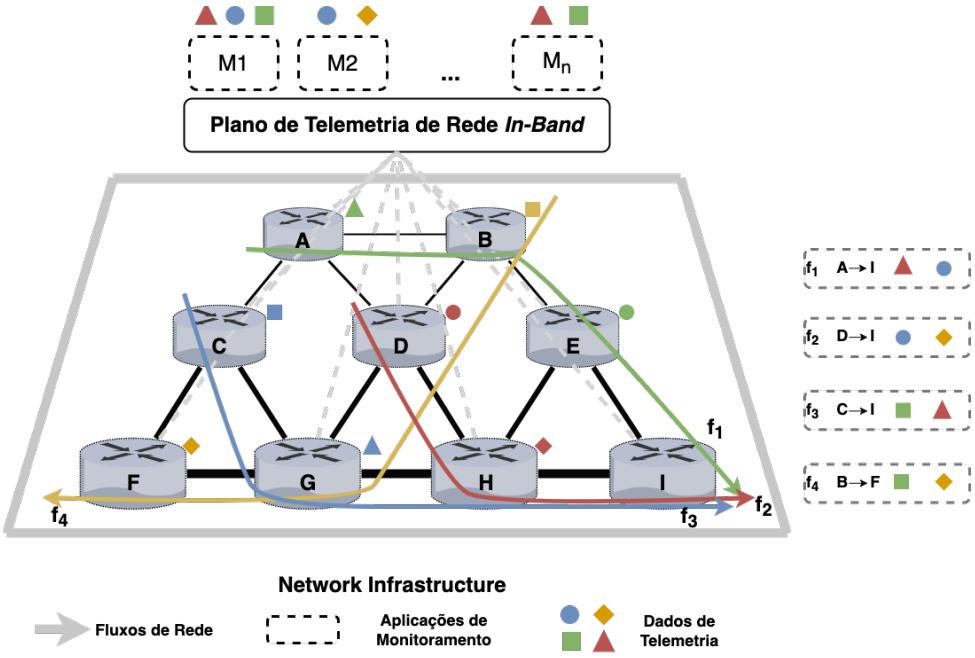


Figura 8 – Exemplo de planejamento de telemetria de rede *in-band*. O exemplo ilustra um instante onde quatro fluxos de rede ativos (f_1, f_2, f_3, f_4) coletam dados de telemetria de dispositivos de encaminhamento. Os dados de telemetria são então enviados para aplicações de monitoramento de acordo com suas necessidades.

(i) aplicações de monitoramento podem ter requisitos completamente diferentes - o que implica que alguns itens de telemetria podem ser coletados com mais frequência do que outros; e *(ii)* os pacotes de rede têm recursos não utilizados limitados (em geral, limitado pelo *Maximum Transmission Unit* (MTU) do link de dados) - e, portanto, é inviável coletar todos os itens na mesma unidade de tempo. Para ilustrar o quão desafiador é esse problema, considere o cenário da Figura 8. Vamos supor que os fluxos de rede podem coletar e transportar uma quantidade máxima fixa de itens de telemetria em uma unidade de tempo específica (por exemplo, dois itens em uma unidade de tempo). Existem algumas alternativas para orquestrar a aquisição de itens INT. Em primeiro lugar, uma solução ingênua consiste em coletar todos os itens de telemetria em cada unidade de tempo. Para construir essa solução, relaxamos a suposição de que os pacotes/quadros de rede são limitados pelo espaço. Uma segunda estratégia (e mais realista) leva em consideração um limite superior na quantidade de itens de telemetria coletados. Portanto, o número de itens de telemetria coletados por fluxos de rede são bastante restritos do que a estratégia ingênua. Na verdade, verifica-se que a introdução deste conjunto de restrições torna o problema NP-difícil (pois é uma generalização do problema de empacotamento (GAREY; JOHNSON, 1979)). Apesar de haver restrições rígidas no número de itens a serem coletados em uma única unidade de tempo, uma solução relaxada pode ser construído com Round-robin (`RR`). Embora isso possa representar uma solução viável ao longo do tempo, ela perde um aspecto fundamental da telemetria de rede, identificar quais itens de tele-

metria são de fato importantes para serem coletados. Essas informações são mportantes para garantir a visibilidade da infraestrutura de rede em toda a rede com precisão.

3.2 Orquestração ótima da coleta de dados de telemetria in-band

3.2.1 Modelo de orquestração ótimo baseado em Programação Linear Inteira

3.2.2 Descrição e notação do Modelo

O modelo de otimização que propomos para resolver o problema de orquestração definido acima considera uma infraestrutura de rede física $G = (D, L)$, um conjunto de fluxos de rede ativos F, um conjunto de itens de telemetria V e um conjunto de aplicações de monitoramento M. O Conjunto D na rede G representa os dispositivos de encaminhamento programáveis $D = \{1, ..., |D|\}$, enquanto o conjunto L interconectando o par de dispositivos $(d_1, d_2) \in (D \times D)$. Cada dispositivo de encaminhamento $d \in D$ é capaz de incorporar um subconjunto de itens $V_d \subset V$ em pacotes de fluxo $f \in F$. Cada item de telemetria $v \in V$ tem seu tamanho definido pela função $S : V \to \mathbb{N}^+$. Os fluxos de rede F são usados para coletar dados de telemetria de dispositivos de encaminhamento D. Um fluxo $f \in F$ tem dois pontos finais, ou seja, origem e destino e é roteado através da infraestrutura de rede G usando um caminho simples $\mathcal{P}t$. Denotamos o caminho percorrido pelo fluxo f como função $\mathcal{P}t : F \to \{D_1 \times ... \times D_{|D|}\}$. Os fluxos de rede F são encapsulados em um protocolo de encaminhamento e a quantidade de espaço disponível para incorporar itens de telemetria em um fluxo de pacote é limitada por um constante $K_f \in \mathbb{N}^+$. Observe que um único pacote de fluxo $f \in F$ pode coletar no máximo $\sum_{\forall d \in \mathcal{P}t(f)} V_d$ itens de telemetria da infraestrutura de rede G em um dado prazo. A menos que K_f assuma um valor grande o suficiente (como, $K_f \geq \sum_{\forall d \ in \mathcal{P}t(f)} |V_d|$), não é possível coletar todos os itens de todos os dispositivos de encaminhamento no caminho roteado $\mathcal{P}t(f)$.

A aplicação de monitoramento $m \in M$ requer um subconjunto de itens de telemetria $R_m \subset V$ para operar. Os itens de telemetria podem ter dependências espaciais e temporais. Dizemos que dois ou mais itens de telemetria têm dependência espacial, pois devem ser coletados do mesmo dispositivo de encaminhamento $d \in D$. Representamos as dependências espaciais pelo conjunto de conjuntos $R^s_m \subset \mathscr{P}(R_m)$. Por sua vez, dizemos que um conjunto de itens de telemetria possuem dependência temporal, desde que devam ser coletados em um determinado prazo. As dependências temporais são representadas pelo conjunto de conjuntos $R^t_m \subset R^s_m$, e o prazo requerido é expresso como uma função $T^f : R^t_m \to \mathbb{N}^+$. O modelo mantém registro da última unidade de tempo em que um item $P \in R^t_m$ foi coletado por meio de uma função $H^f : R^t_m \to \mathbb{N}^+$. Quando $(\forall P \in R^t_m) : H^f(P) > T^f(P)$, o item P está desatualizado, isto é, o prazo foi expirado.

Dada uma infraestrutura de rede G, um conjunto de fluxos de rede F, um conjunto de aplicações de monitoramento M, um conjunto de itens de telemetria V e uma constante

K_f, o problema de otimização busca uma solução viável que maximize o número de dependências espaciais e temporais. O modelo de saída é denotado por uma 3-tupla $\chi = \{Y, S^b, T^b\}$. Variáveis de $Y = \{\, y_{d,v,f}\,, \forall\, d \in D, v \in V, f \in F\}$ indica que um dispositivo de encaminhamento d insere item de telemetria v em pacotes de fluxo de rede f. Variáveis de $S^b = \{\, s^b_{m,d,p}\,, \forall m \in M, d \in D, p \in P \in R^s_m\}$ e $T^b = \{\, t^b_{m,p}\,, \forall m \in M, p \in P \in R^t_m\}$ são usadas para acompanhar as dependências espaciais e temporais satisfeitas pelo modelo. O conjunto de variáveis $s^b_{m,d,p}$ indica que a dependência espacial p da aplicação de monitoramento m é satisfeita no dispositivo de rede d. Inversamente, o conjunto de variáveis $t^b_{m,p}$ indica que a dependência temporal p da aplicação de monitoramento m é satisfeita. A seguir, descrevemos a formulação de programação linear inteira mista para o problema.

$$\textbf{Maximizar} \qquad \sum_{m \in M} \sum_{p \in P \in R^s_m} \sum_{d \in D} s^b_{m,d,p} + t^b_{m,p} \tag{1}$$

Sujeito a:

$$\sum_{d \in \mathcal{P}t(f)} \sum_{v \in V_d} y_{d,v,f} \cdot S(v) \leq K_f \qquad \forall f \in F \tag{2}$$

$$\sum_{f \in F} y_{d,v,f} \leq 1 \qquad \forall d \in D, v \in V_d, \tag{3}$$

$$s_{m,d,p} = \sum_{v \in P} \sum_{f \in F} y_{d,v,f} \qquad \forall m \in M, p \in P \in R^s_m, d \in D \tag{4}$$

$$t_{m,p} = \sum_{v \in P} \sum_{d \in D} \sum_{f \in F} y_{d,v,f} \qquad \forall p \in P \in R^t_m : H^f(p) > T^f(p) \tag{5}$$

$$s^b_{m,d,p} \leq \frac{s_{m,d,p}}{|P|} \qquad \forall m \in M, p \in P \in R^s_m, d \in D \tag{6}$$

$$t^b_{m,p}. \leq \frac{t_{m,p}}{|P|} \qquad \forall m \in M, p \in P \in R^t_m \tag{7}$$

$$y_{d,v,f} \in \{0, 1\} \qquad \forall d \in D, v \in V, f \in F \tag{8}$$

$$s_{m,d,p} \in \mathbb{N}^+ \qquad \forall m \in M, d \in D, p \in P \in R^s_m \tag{9}$$

$$s^b_{m,d,p} \in \{0,1\} \qquad \forall m \in M, d \in D, p \in P \in R^s_m \quad (10)$$

$$t_{m,p} \in \mathbb{N}^+ \qquad \forall m \in M, p \in P \in R^t_m \quad (11)$$

$$t^b_{m,p} \in \{0,1\} \qquad \forall m \in M, p \in P \in R^t_m \quad (12)$$

O conjunto de restrições (2) garante que um fluxo de rede $f \in F$ não exceda sua capacidade K_f, enquanto o conjunto de restrições (3) garante que um determinado item de telemetria v seja coletado no máximo por um único fluxo de rede $f \in F$ em um determinado dispositivo d. Os conjuntos de restrições (4) e (6) consideram se uma dependência espacial é atendida ou não. Para uma determinada dependência espacial, o conjunto de restrições (4) conta o número de itens de telemetria coletados em um dispositivo d, enquanto o conjunto de restrições (6) verifica se a dependência é atendida ou não, isto é, verificando se todos itens foram coletados. Da mesma forma, os conjuntos de restrições (5) e (7) contam o número de dependências temporais que são satisfeitas. Observe que os conjuntos de variáveis $s_{m,d,p}$ e $t_{m,p}$ são variáveis auxiliares utilizadas para contar o número de itens de telemetria coletados por dependências de monitoramento. Por último, os conjuntos de restrições (8)-(12) definem os domínios das variáveis.

3.2.3 Modelo de orquestração baseado em aprendizado de máquina

O modelo de orquestração proposto nas subseções anteriores é capaz de maximizar o número de itens de telemetria coletados. No entanto, ainda falta a questão fundamental da orquestração: *quais itens de telemetria são de fato importantes para serem coletados?* Modelos semelhantes na literatura como Cohen et al. (COHEN; KATZIR; RAZ, 2006) trabalham com a importância de um determinado item por meios de pesos arbitrários, que precisam ser ajustados manualmente de tempos em tempos. Para preencher essa lacuna, inserimos em nosso modelo um mecanismo de aprendizagem a fim de instrumentar de forma dinâmica (e inteligente) a coleta de itens de telemetria da rede.

A importância de um item de telemetria depende dos requisitos de aplicações de monitoramento, que mudam dinamicamente ao longo do tempo de acordo com o comportamento da rede. Assumimos que em uma determinada unidade de tempo t, o modelo de aprendizagem proposto conhece um subconjunto de itens de telemetria, representados por $V^t \subseteq \sum_{t-W}^{t} \sum_{m \in M} R^s_m$, onde $W \in \mathbb{N}^+$ representa uma determinada janela de tempo. Um elemento $P \in V^t$ representa um $|P|$ tupla de espaço dimensional $P = (v_1, v_2, \cdots v_{|P|})$, onde cada dimensão representa uma informação de telemetria coletada da infraestrutura,

como $v_{|S|} \in \mathbb{R}^+$. Dado um conjunto de V^t de dados de telemetria coletados nas últimas W unidades de tempo, modelamos a camada do Plano de Telemetria de Rede *In-Band* (como mostrado na Figura 8) como vários modelos de agrupamento *online*. A ideia consiste em agrupar comportamentos de rede com base em itens de telemetria que compartilham as mesmas dependências. O modelo de orquestração, portanto, pode coordenar quais itens de telemetria são importantes para serem coletados a seguir (por exemplo, itens de telemetria observando comportamentos incomuns). Cada *cluster* mantém o controle de itens com as mesmas dimensões $|P|$. Particionamos $V^t = \{V^{t,1}, V^{t,2}, ..., V^{t,\mathscr{P}(V^t)}\}$ para que cada subconjunto possui itens com as mesmas dimensões $|P|$. Em seguida, agrupamos os subconjuntos $V^{t,i} \in V^t$ em K_i partições exclusivas, como $(\forall i \in \{1, ..., \mathscr{P}(V^t)\})$:$V_1^{t,i}, V_2^{t,i}, ..., V_{K_i}^{t,i}$. O centroide de um subconjunto $V_k^{t,i}$ é definido como $C(V_k^{t,i})$. Um elemento $P \in V^{t,i}$ é atribuído a um *cluster* $V_k^{t,i}$ se a distância de v a $(V_k^{t,i})$ é mínimo. No entanto, é importante mencionar que os problemas de agrupamento são problemas NP-difíceis bem conhecidos (MAHAJAN; NIMBHORKAR; VARADARAJAN, 2012) - mesmo para gráficos planares.

Em seguida, definimos como o modelo mantém a consistência e *freshness* do *cluster* ao longo do tempo. Para capturar a dinâmica da rede, propomos uma função de *fading* que estima por quanto tempo o modelo mantém os dados de telemetria nele. Os dados de telemetria medidos com maior variação em um período de tempo W tendem a ser abrangidos por um número menor de unidades de tempo do que aqueles com menor variação. A variância de um item de telemetria (ou tupla - P) é definida de acordo com seu índice de dispersão. Denotamos o índice de dispersão como $I(v) = \frac{\sigma_v^2}{\mu}$. O número de unidades de tempo que um item de telemetria é mantido em V^t é definido de acordo com uma função por peça $T(P) : V^t \to \mathbb{N}^+$, definido como segue :

$$T(P) = \begin{cases} W & I(P) \leq 1 \\ W \cdot (1 - \rho) & I(P) > 1 \end{cases} \tag{13}$$

Os pontos de dados de telemetria $P \in V^t$ que estão sub dispersos (ou seja, $I(P) \leq 1$) são mantidos no agrupamento por pelo menos W unidades de tempo. Por sua vez, os pontos de dados super dispersos (como, $I(P) > 1$) tendem a estar no *cluster* por menos unidades de tempo (W é decaído em relação a $\rho \in [0, 1]$). Observe que esta abordagem permite que o modelo capture eventos transitórios (como, filas ocupadas por um curto espaço de tempo) e também eventos persistentes que perduram por unidades de tempo mais longas (como, interrupção física em um dispositivo de encaminhamento). Denotamos por $V^+ \subset V^t$ e $V^- \subset V^t$ o subconjunto de itens de telemetria que estão super e sub dispersos, respectivamente. Itens super dispersos com dependências espaciais são representados pelo conjunto $R_m^{+s} = \{P \in R_m^{+s} | P \in R_m^s \wedge (P \cap V^+ \neq \emptyset)\}$. Da mesma forma, itens sub dispersos com dependências espaciais pelo conjunto $R_m^{-s} = \{P \in R_m^{-s} | P \in R_m^s \wedge (P \cap V^- \neq \emptyset)\}$.

Estamos interessados em otimizar como evoluímos ao longo do tempo as informações que conhecemos sobre a infraestrutura G, mantendo *cluster* $V^{t,i} \in V^t$ ao longo do tempo. Presumimos que na maioria das vezes, o valor dos pontos de telemetria evolui suavemente. Mudanças abruptas acontecem no estado da rede, mas com baixa probabilidade. Essa suposição é realista para telemetria de rede *in-band*. Nosso principal objetivo é reduzir a aquisição de dados de telemetria irrelevantes e, ao mesmo tempo, manter a precisão dos dados de telemetria adquiridos. O modelo de orquestração baseado em aprendizado de máquina substitui a Equação (1) pela Equação (14).

$$\begin{aligned} \textbf{Maximize} \quad & \alpha \cdot \sum_{m \in M} \sum_{p \in P \in R_m^+} \sum_{d \in D} s_{m,d,p}^b + t_{m,p}^b \\ & + \beta \cdot \sum_{m \in M} \sum_{p \in P \in R_m^-} \sum_{d \in D} s_{m,d,p}^b + t_{m,p}^b \end{aligned} \tag{14}$$

A função objetivo combinada tenta maximizar sabiamente o número de itens super dispersos e sub dispersos coletados de G. Observe que a função objetivo prioriza a coleta de itens sobre dispersos e sub dispersos de acordo com os parâmetros α e β. Isso se explica porque, em operação normal da rede, o índice de dispersão tende a diminuir enquanto não houver um evento suspeito. No caso de um evento suspeito, o índice tenderia a aumentar e, portanto, haverá grandes chances de ser coletado.

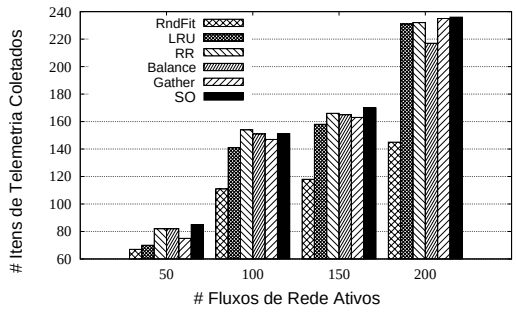
3.2.4 Avaliação

3.2.5 Carga de Trabalho

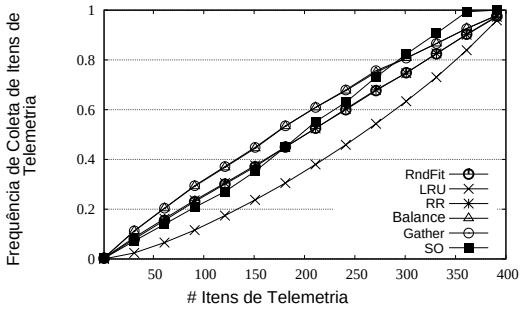
Executamos o modelo proposto usando IBM *CPLEX Optimization Studio* 12.9 para obter soluções ideais e implementamos a versão online do K-*Means* (LIBERTY; SRIHARSHA; SVIRIDENKO, 2016) usando a linguagem Java para ponderar dinamicamente o importância dos itens de telemetria [1]. Os experimentos foram realizados em uma máquina com quatro processadores Intel Xeon E5-2670 e 56 GB de RAM, usando o Ubuntu *Server* 11.10. Consideramos diferentes instâncias de rede física que foram geradas com Brite (Medina et al., 2001), seguindo o modelo Barabasi-Albert (ALBERT; BARABÁSI, 2000). Usamos infraestruturas físicas consistindo de 50 dispositivos de encaminhamento e, em média, 200 links físicos. No topo de cada instância de infraestrutura, havia um conjunto F de fluxos ativos. Variamos a quantidade de fluxos de rede de 50 a 200. Cada fluxo $f \in F$ interconectou dois pontos finais aleatórios na infraestrutura e foi roteado usando o algoritmo de caminho mais curto. Consideramos fluxos de rede baseados em IP e, portanto, variamos uniformemente a constante K_f de 10-30 bytes. Além disso, assumimos que os dispositivos de encaminhamento têm 8 itens de telemetria possíveis para exportar[2], variando $S(v)$ de 2 a 20 bytes (PAN et al., 2019). Consideramos um

[1] Material reproduzível disponível em https://github.com/mcluizelli/comml-int19

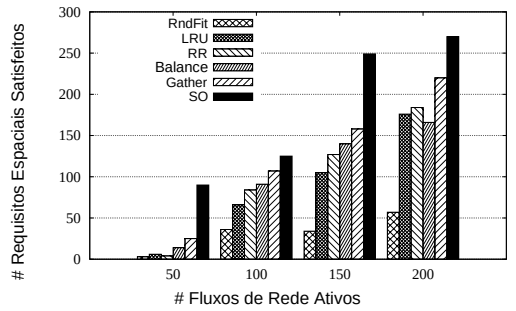
[2] Telemetria de rede in-band: <https://p4.org/assets/INT-current-spec.pdf>



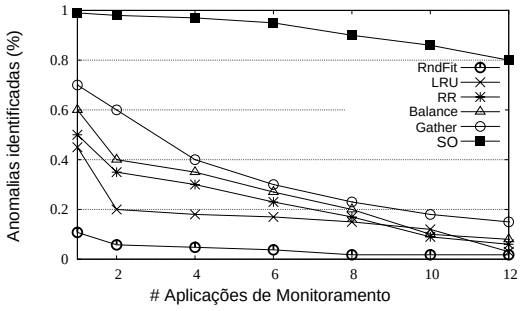
(a) Itens de telemetria coletados.



(b) Número de vezes que os itens de telemetria foram coletados (200 fluxos de rede).



(c) Requerimentos espaciais satisfeitos.



(d) Eventos anômalos detectados (200 fluxos de rede).

Figura 9 – Métricas de performance em orquestração de itens INT.

conjunto de aplicações de monitoramento, variando de 2 a 12. Cada aplicação requer no máximo 4 dependências espaciais, cada uma com pelo menos 2 itens (como, ID do switch + métrica monitorada) e no máximo 4. Consideramos $\alpha = 1$, $\beta = 1$, $W = 5$ e $\rho = 0,5$, escolhidos de forma empírica. Esses parâmetros podem ser ajustados para priorizar a coleta de dependências espaciais sobre as temporais e para ajustar as funções de *fading*. Nosso experimento foi executado em unidades de tempo de 200. Em cada unidade de tempo, nosso modelo e algoritmos são executados. Cada experimento é repetido 30 vezes para garantir um nível de confiança de 95% ou superior.

3.2.6 Heurísticas avaliadas

Comparamos nosso modelo proposto denominado SO com *(i)* duas heurísticas de orquestração de última geração propostas por (MARQUES et al., 2019), a saber `Gather` e `Balance`, e *(ii)* três heurísticas: Round-Robin (`RR`), *Least-Recently Collected* (`LRC`), Random-Fit (`RndFit`). A heurística `RR` se esforça para atribuir itens de telemetria aos fluxos de rede de forma round-robin. `LRC` prioriza a coleta de itens de telemetria que não foram coletados recentemente. Finalmente, `RndFit` escolhe itens de telemetria aleatoriamente e tenta atribuir ao fluxo de rede aleatório.

3.2.7 Resultados

Focamos nossa análise em dois aspectos principais do monitoramento de rede: *(i)* visibilidade de toda a rede (em termos de cobertura) e *(ii)* desempenho da aplicação de monitoramento de rede. Para isso, avaliamos a cobertura de telemetria de rede (ou seja, o número de itens de telemetria coletados), uma série de dependências espaciais satisfeitas e a capacidade de identificar anomalias de rede. A Figura 9(a) ilustra a quantidade média de itens de telemetria coletados ao longo do tempo para um número crescente de fluxos de rede ativos. Observamos que *(i)* número de fluxos de rede impacta diretamente na visibilidade da rede (como, quanto mais fluxos de rede ativos, maior a cobertura da rede) e *(ii)* o modelo de orquestração proposto supera em média as heurísticas avaliadas em um fator de 1,7x quando comparado ao `RndFit`. Esse comportamento se deve à capacidade de atribuir itens de telemetria de maneira ideal aos fluxos de rede. A seguir, avaliamos o número de vezes(frequência) que os itens de telemetria são coletados, ilustrados na Figura 9(b) por meio de uma Função de Distribuição Cumulativa (FDC). Apesar de mostrar na Figura 9(a) que o número de itens coletados pelo nosso modelo e pelas heurísticas avaliadas são semelhantes, observamos na Figura 9(b) que cada método prioriza as coleções de diferentes subconjuntos de itens de telemetria, como por exemplo a LRC, que prioriza a coleta de itens de telemetria não coletados em um tempo anterior. Esta diferença ocorre devido à *(i)* política de atribuição heurística e *(ii)* ao número de fluxos de rede ativos disponíveis para atribuir itens de telemetria. O último ponto pode ser explicado pela presença de dispositivos de encaminhamento altamente interconectados nas topologias de rede - em que itens de telemetria são potencialmente coletados com mais frequência do que outros. Esse fato é ainda mais exacerbado à medida que os fluxos de rede ativa são roteados usando os caminhos mais curtos. Observe que nosso modelo de orquestração, diferentemente das heurísticas avaliadas, prioriza a coleta de um subconjunto de itens de telemetria - em média, 25% dos itens de telemetria são coletados de forma mais intensa (mais de 80% das unidades de tempo). Em contraste, até 40% dos itens de telemetria são coletados com menos frequência (menos de 40% das unidades de tempo). A Figura 9(c) ilustra o número médio de dependências espaciais satisfeitas ao longo do tempo. Conforme observado, nosso modelo supera as heurísticas avaliadas por um fator de 2,5x (para 150 fluxos de rede ativos). O número de dependências espaciais satisfeitas é uma informação de entrada essencial para os modelos de aprendizagem, que exigem informações de telemetria acopladas - caso contrário, os dados de telemetria são inúteis para o processo de aprendizagem.

Por último, a Figura 9(d) ilustra o número médio de anomalias de rede identificadas. Para este experimento, consideramos um número crescente de aplicações de monitoramento executando em paralelo e, portanto, aumentando a necessidade de um número maior de dependências espaciais. Além disso, assumimos que os valores dos itens de telemetria seguem uma distribuição normal, definida por uma média e desvio padrão

(por exemplo, ocupação da fila dada pela média de 100 e desvio de 20). Em seguida, injetamos anomalias nesses itens (isto é, valores fora do intervalo esperado), durando 5 vezes unidades (em média) em (no máximo) 10% dos dispositivos de encaminhamento. As anomalias foram injetadas por uma função geradora no fluxo de produção. Quando o número de aplicações de monitoramento é relativamente pequeno (até 2 aplicações de monitoramento), o mecanismo de aprendizado de máquina pode identificar até 50-60% das anomalias da rede com a entrada fornecida pelos dados de telemetria coletados pelas abordagens avaliadas. No entanto, à medida que aumenta o número de aplicações de monitoramento (e, portanto, o número de dependências espaciais), a capacidade de identificar anomalias de rede diminui consideravelmente, chegando a até 5-10% das anomalias identificadas. O motivo está relacionado à frequência com que os itens de telemetria são coletados da rede. Os algoritmos avaliados se esforçam para coletar itens de telemetria de forma justa (por exemplo, round-robin) - o que reduz a frequência geral com que os itens de telemetria são coletados e, portanto, aumenta as chances de perder a coleta de eventos anômalos. Em contraste, nosso modelo pode lidar com o número crescente de aplicações de monitoramento. A entrada fornecida por nosso modelo pode alimentar modelos de mecanismos de aprendizado de máquina para identificar até 97% das anomalias da rede (até 4 aplicações de monitoramento). De 4 a 12 aplicações de monitoramento de rede, a precisão de identificação de anomalias de rede diminui em 20% (isto é, 8x maior do que as heurísticas avaliadas).

3.2.8 Considerações

O modelo proposto nas subseções anteriores é capaz de maximizar o número de itens de telemetria coletados utilizando-se da abordagem proposta pela PLIM e do Método Simplex. Também é capaz de oferecer uma solução ótima, levando em consideração restrições impostas pelo modelo, como exemplo a capacidade residual do fluxo para a coleta de itens de telemetria. Outra contribuição proposta para incrementar a visibilidade está na qualidade dos itens de telemetria coletados, ou seja, coletar itens de telemetria que de fato atendam aos requisitos das aplicações de monitoramento e em um determinada janela de tempo. Para isso, dividiu-se em duas etapas a qualificação destes dados, sendo i) definição da importância de determinado item por meio de pesos arbitrários e ajustáveis e ii) agrupamento de itens de telemetria levando em consideração comportamentos comuns empregando o método k-*Means online*. Esta fatia de tempo está relacionada com a *freshness* dos dados coletados e consumidos pelas aplicações de monitoramento.

O incremento à visibilidade da rede fica evidente quando comparada a solução proposta com as demais soluções existentes, seja pelo melhor aproveitamento dos fluxos de rede existentes, coleta de itens de telemetria, satisfação de dependências espaciais e número médio de anomalias de rede identificadas, por exemplo. Apesar do sucesso da solução, uma questão a ser considerada é complexidade da solução, em especial o

método Simplex, que neste caso é limitada superiormente por uma função exponencial (KLEE; MINTY, 1972), assim, quanto maior a infraestrutura de rede, como por exemplo, dispositivos de encaminhamento, fluxos de rede e aplicativos de monitoramento, maior será o tempo necessário para se obter uma solução ótima. Dessa forma, a eficiência da proposta apresentada fica comprometida em decorrência de um futuro aumento da rede, e comprometendo consequente a sua escalabilidade e afetando diretamente a proposta do emprego da INT, que visa uma rápida resposta a eventos que ocorrem na rede. No próximo capitulo apresentamos uma solução heurística para resolver este problema de forma eficiente.

3.3 Algoritmo Meta-Heurístico Proposto

Para atacar a complexidade do problema e obter soluções com a alta qualidade, nesta seção introduzimos um algoritmo heurístico para o problema. Nosso algoritmo é baseado em *Greedy Randomized Adaptive Search Procedure* (GRASP) (FEO; RESENDE, 1995), técnicas que tem sido aplicada com sucesso para resolver problemas de otimização complexos.

3.3.1 GRASP aplicado ao problema de orquestração de telemetria *in-band* escalável

A ideia central consiste em construir iterativamente uma solução χ (linha 3), seguida por um procedimento de busca local (linha 5) de modo a evitar máximos locais através do emprego de soluções aleatórias gananciosas (linhas 2-11). O algoritmo é executado até que um critério de parada seja atendido, que pode ser um limite de tempo global ou uma série de iterações sem melhorias.

Cada vez que o algoritmo fica preso em algum máximo local (linha 10), o procedimento é reiniciado, ou seja, uma nova solução é construída do zero. No entanto, acompanhamos a melhor solução encontrada até agora (χ^{best}). O Algoritmo 1 consolida a abordagem proposta.

3.3.2 Heurística construtiva

A heurística proposta para resolver o problema de telemetria de rede *in-band* utiliza uma construção passo a passo de uma solução candidata usando um procedimento de construção guloso probabilístico. O Algoritmo 2 é detalhado a seguir.

A heurística começa a construindo uma Lista de Candidatos Restritos (LCR) $\mathcal{L}$, que consiste em possíveis atribuições de dependências espaciais/temporais para dispositivos de encaminhamento em potencial. Para todas as dependências de monitoramento P, ou seja, $(m \in M) : P \in R_m^s$, o conjunto de dispositivos de encaminhamento em potencial D' consiste no subconjunto $D' \subset D$ que satisfaça as dependências espaciais/temporais P,

Algorithm 1 Abordagem heurística baseada em GRASP

Input: $G = (D, L)$: infraestrutura de rede
Input: α: ganância da heurística de construção
1: $\chi^{best} \leftarrow \emptyset$
2: **while** os critérios de parada não são atendidos **do**
3: $\chi \leftarrow$ gerar solução aleatória gananciosa, com parâmetro α (Algorithm 2)
4: **repeat**
5: $\chi^{'} \leftarrow$ buscaLocal(χ) (Algorithm 3)
6: Aplica-se Alg. 2 na solução atual $\chi^{'}$ para dependências mais satisfeitas
7: **if** $cost(\chi^{'}) > cost(\chi^{best})$ **then**
8: $\chi^{best} \leftarrow \chi^{'}$
9: **end if**
10: **until** a solução $\chi^{'}$ não melhorou
11: **end while**
12: **return** χ^{best}

isto é, $D' = \{d \in D' \mid P \subset V_d\}$. Então, o LCR criado $\mathcal{L}$ é classificado em ordem crescente em relação à soma dos itens de telemetria em P, isto é, $\sum_{v \in P} S(v)$). Linhas 2-3 descrevem este procedimento.

O algoritmo itera até que não aja nenhuma dependência espacial/temporal desmarcada (linhas 4-20). Como a função objetivo consiste em maximizar a quantidade de dependências espaciais/temporais satisfeitas, nossa heurística tenta primeiro satisfazer as dependências de menor tamanho. Em vez de construir uma solução avidamente, nossa estratégia escolhe a próxima dependência espacial/temporal aleatoriamente entre as melhores $\alpha \in [0, 1]$ candidatos em $\mathcal{L}$. Isto é, se o parâmetro $\alpha = 0.3$, então os algoritmos restringiriam o LCR $\mathcal{L}$ para os 30% melhores candidatos. Dentre eles, a escolha é feita de forma randômica. Observe que os valores mais baixos de α fazem com que o algoritmo se comporte avidamente, enquanto valores mais altos fazem o procedimento agir de forma aleatória. Este procedimento é ilustrado nas linhas 5-6.

Depois de escolher o candidato espacial/temporal (P, d), o algoritmo tenta iterativamente atribuir um item de telemetria $v \in P$ para um fluxo de rede disponível $f \in F$ (linhas 9-15). Observe que os possíveis fluxos de rede são calculados antecipadamente na linha 7. Um item de telemetria $v \in P$ é atribuído ao fluxo da rede f *sse* seu espaço disponível K_f é maior que o tamanho do item v, ou seja, $S(v) \leq K_f$ (linha 10). No caso de um item de telemetria $v \in P$ não é atribuído a um fluxo de rede disponível, todas as atribuições anteriores de (P, d) são desfeitas (linhas 16-18), e o procedimento é reiniciado. Por último, o algoritmo retorna uma solução viável χ (linha 21). A complexidade do algoritmo é limitada por $O(|M| \cdot |R^s_m| \cdot |D| \cdot |V| \cdot |F|)$. Observe que, na prática, o número de dependências espaciais/temporais R^s_m, e o número máximo de itens de telemetria V devem ser limitados por uma constante fixa.

Algorithm 2 Procedimento Heurístico Construtivo

Input: α: ganância da heurística de construção

1: $\chi \leftarrow \emptyset$
2: $\mathcal{L} \leftarrow \{(P,d) \in \mathcal{L} : P \in R_m^s \wedge P \subseteq V_d\}$, gerar lista de candidatos restritos
3: ordenar $\mathcal{L}$ em ordem ascendente considerando o tamanho da dependência P, isso é, $\sum_{p \in P} S(p)$
4: **while** há dependência espacial/temporal em $\mathcal{L}$ **do**
5: $(P,d) \leftarrow$ Random($\mathcal{L}$, α)
6: $\mathcal{L} \leftarrow \mathcal{L} \setminus (P,d)$
7: $\mathcal{F} \leftarrow$ fluxos de rede possíveis , ou seja, aqueles roteados por meio de dispositivo de encaminhamento d
8: **for** cada item de telemetria $v \in P$ **do**
9: **for** cada fluxo de rede candidato $f \in \mathcal{F}$ **do**
10: **if** $S(v) \leq K_f$ **then**
11: Atribui o item de telemetria v para o fluxo de rede f na solução χ
12: $K_f \leftarrow K_f - S(v)$
13: break
14: **end if**
15: **end for**
16: **if** item de telemetria $v \in P$ não foi atribuído **then**
17: Desfaz atribuição anterior de dependência espacial/temporal (P,d) in χ
18: **end if**
19: **end for**
20: **end while**
21: **return** χ

3.3.3 Heurística de melhoramento

Depois de construir uma solução estabelecida χ, aplicamos um procedimento de busca local para liberar espaço nos fluxos de rede utilizados $\mathcal{F}$. A ideia é realocar atribuições anteriores (P,d) em uma tentativa de eventualmente, permitir que dependências espaciais/temporais bloqueadas sejam satisfeitas. O algoritmo 3 apresenta uma visão geral do procedimento proposto.

O algoritmo verifica todos os dispositivos de rede $d \in D$, e todos os pares de fluxos de rede roteados por d, ou seja, pares (f_1, f_2) (linhas 5-7), e tenta reatribuir itens de telemetria de dependência espacial/temporal (P,d) de um determinado fluxo de rede f_1 para o fluxo da rede f_2 (linha 9). Uma reatribuição só é válida *sse* (i) existe espaço disponível no fluxo da rede f_2, ou seja, $K_{f_2} - S(v) \geq 0$; e (ii) a reatribuição não foi realizada anteriormente (linha 10). O algoritmo acompanha as reatribuições realizadas usando o conjunto $\mathcal{N}_{lst}$. Observe que quando uma reatribuição $\mathcal{N}(v, f_1, f_2)$ é feita, sua contraparte $\mathcal{N}(v, f_2, f_1)$ também é adicionado a $\mathcal{N}_{lst}$ (linha 13). Essa condição evita que o algoritmo alterne entre as reatribuições. Os algoritmos de busca local adotam o primeira estratégia de melhoria, isto é, sobre todos os movimentos válidos, ele sempre executa o primeiro que maximiza o espaço em um determinado fluxo de rede (linhas 18-20). Sempre

que uma reatribuição é feita, o procedimento é reiniciado.

Algorithm 3 Procedimento de Busca Local – Realocação de itens de telemetria

Input: $G = (D, L)$: infraestrutura de rede
Input: χ: solução atual
1: $\mathcal{N} \neq \emptyset$ ▷ movimento atual
2: $\mathcal{N}_{lst} \leftarrow \emptyset$ ▷ lista de todos os movimentos realizados
3: **while** solução χ foi melhorada ($\mathcal{N} \neq \emptyset$) **do**
4: $\mathcal{N} \leftarrow \emptyset$
5: **for** cada dispositivo de encaminhamento $d \in D$ **do**
6: $\mathcal{F} \leftarrow$ possíveis fluxos de rede, aqueles roteados por meio de dispositivo de encaminhamento d em χ
7: **for** cada par de fluxos de rede $(f_1, f_2) \in \mathcal{F}$ **do**
8: **for** cada item de telemetria v atribuído ao fluxo de rede f_1 **do**
9: $\mathcal{N} \leftarrow (v, f_1, f_2)$ ▷ Reatribuir o item v do fluxo de rede f_1 to f_2
10: **if** $\mathcal{N} \notin \mathcal{N}_{lst} \wedge \left(\sum_{d' \in \mathcal{P}t(f)} \sum_{u \in V_{d'}} y_{d',u,f} \cdot S(u)\right) + S(v) \leq K_{f_2}$ **then**
11: Executar movimento $\mathcal{N}$ na solução χ
12: $\mathcal{N}' \leftarrow (v, f_2, f_1)$
13: $\mathcal{N}_{lst} \leftarrow \mathcal{N}_{lst} + \mathcal{N} + \mathcal{N}'$
14: Reiniciar a pesquisa local, vá para linha 3
15: **end if**
16: **end for**
17: **end for**
18: **end for**
19: **end while**
20: **return** χ, φ

3.4 Avaliação experimental

Nesta seção, avaliamos a eficácia de nossa abordagem baseada em heurística, comparando-a com concorrentes de última geração - `Gather` e `Balance`. Todos os experimentos foram realizados em uma máquina com quatro processadores Intel Xeon E5-2670 e 56 GB de RAM, utilizando o sistema operacional Ubuntu GNU / Linux Server 11.10 x86 64bits.

3.4.1 Carga de trabalho

Para avaliar o problema de telemetria da rede *in-band*, consideramos diferentes instâncias de rede física que foram geradas com Brite (Medina et al., 2001), seguindo o modelo Barabasi-Albert (ALBERT; BARABÁSI, 2000). Nós consideramos infraestruturas de rede física que variam de pequena escala (50 dispositivos de encaminhamento) a grandes (até 600 dispositivos de encaminhamento).

Acima de cada instância de infraestrutura, há um conjunto F de fluxos de rede. Variamos o tamanho de $|F|$ de 50 a 600. Cada fluxo $f \in F$ interconectou dois pontos finais

aleatórios em G usando o caminho mais curto. Da mesma forma que (HOHEMBERGER et al., 2019), consideramos a constante de limite superior K_f variando uniformemente de 10-30 Bytes, enquanto os dispositivos de encaminhamento têm 8 possíveis itens de telemetria para exportar[3], variando $S(v)$ de 2 a 20 Bytes (PAN et al., 2019).

Além disso, consideramos um número crescente de aplicativos de monitoramento - variando de 2 a 20. Cada aplicativo requer no máximo 4 dependências espaciais, cada uma com pelo menos 2 itens (por exemplo, identificador do *switch* + métrica monitorada) e no máximo 4. O algoritmo proposto foi definido para considerar $\alpha = 0.3$, uma vez que empiricamente mostrou ser o valor mais eficaz, e um número de iterações sem melhorias igual a 10, como critério de parada. Cada experimento foi repetido 30 vezes para garantir um nível de confiança de 95% ou superior.

3.4.2 Heurísticas avaliadas

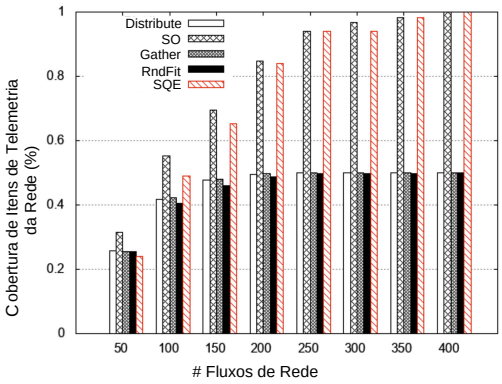
Comparamos nossa heurística proposta denominada SQE com a *(i)* solução ótima, SO proposta (HOHEMBERGER et al., 2019); *(ii)*duas heurísticas de orquestração atuais propostas por (MARQUES et al., 2019), demominadas `Gather` e `Balance`, e *(iii)* para uma heurística de pior caso, a heurística Random-Fit (`RndFit`). A heurística `RndFit` escolhe aleatoriamente os itens de telemetria e tenta atribuir ao fluxo de rede disponível aleatório.

3.5 Resultados

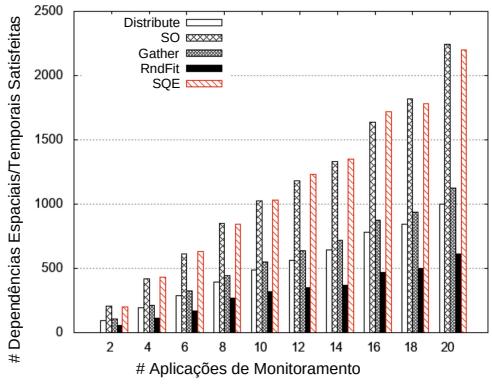
Primeiro, analisamos nossa abordagem proposta denominada SQE em comparação com a solução do estado da arte em relação às métricas de desempenho de telemetria de rede *in-band*, considerando infraestruturas de rede de pequena escala. Em seguida, nos concentramos em mostrar que nossa solução pode lidar em tempo hábil com infraestruturas de rede em grande escala.

Começamos a analisar a visibilidade de toda a rede em termos de cobertura de telemetria de rede (ou seja, a porcentagem de itens de telemetria coletados). A Figura 10(a) mostra a cobertura média de telemetria de rede de itens de telemetria coletados para um número crescente de fluxos de rede ativos. Observamos que com o incremento do número de fluxos há o impacta direto na cobertura da rede e consequentemente na visibilidade. Notamos também que a heurística proposta em média supera as heurísticas existentes por um fator de 2. Observe que o número crescente de fluxos de rede não afeta a cobertura obtida para heurísticas atuais. Isso acontece porque esses algoritmos fazem escolhas gananciosas, levando a uma solução sub ótima. Mais importante, podemos observar que nossa heurística pode cobrir a telemetria de rede de forma semelhante à solução ótima (em média, a solução está a menos de 5% da ótima).

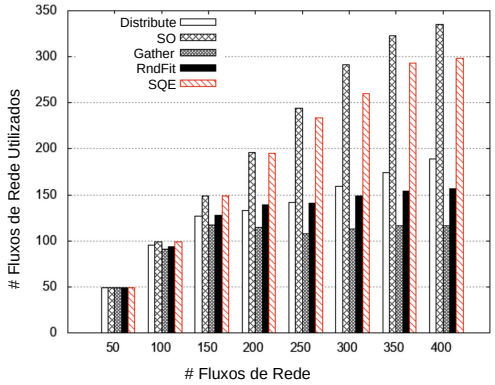
[3] In-band Network Telemetry: <https://p4.org/assets/INT-current-spec.pdf>



(a) Cobertura dos itens de telemetria.



(b) Número de dependências espacial e temporal satisfeitas



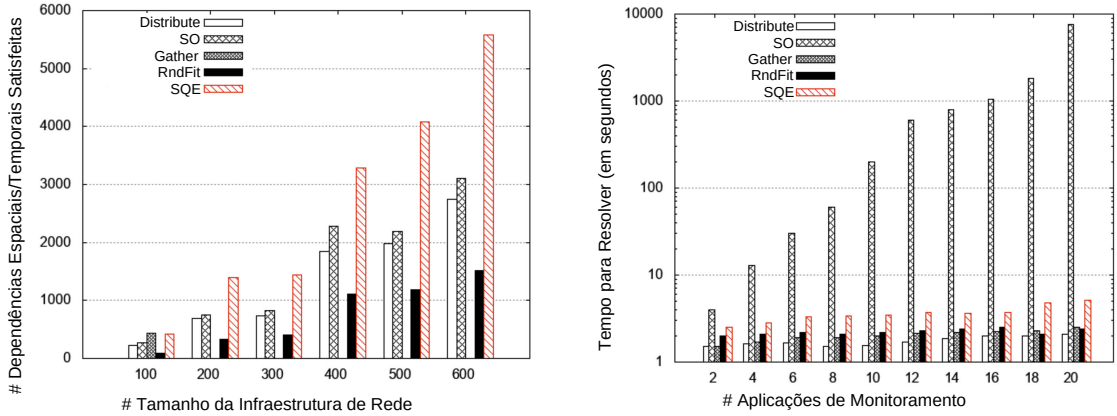
(c) Número de fluxos de redes utilizados por soluções

Figura 10 – Métricas de desempenho de orquestração INT aplicadas a pequenas instâncias (50 dispositivos de encaminhamento).

Em seguida, avaliamos a quantidade de dependências espaciais/temporais que são satisfeitas, dado um conjunto crescente de aplicativos de monitoramento (variando de 2 a 20), ilustrado na Figura 10(b). Para este experimento, definimos o número de fluxos em 400. Podemos observar que (i) o número de dependências espaciais/temporais aumenta com o aumento das aplicações de monitoramento e (ii) que nossa solução heurística está próxima da ótima. Além disso, nossa heurística produz soluções até 3x melhores do que os concorrentes e no máximo 7% longe do ótimo (em relação ao número de dependências espaciais/temporais satisfeitas). Além disso, observamos que as soluções atuais lutam para lidar com dependências espaciais/temporais, especialmente quando o número de aplicativos de monitoramento aumenta. A razão para esse comportamento é porque os concorrentes avaliados tomam decisões locais sem considerar o monitoramento dos requisitos do aplicativo. Por exemplo, o algoritmo de linha de base `Balance` se esforça para coletar itens de telemetria de maneira justa (por exemplo, round-robin nos fluxos de rede) - o que diminui a chance geral de satisfazer os requisitos de monitoramento. Vale ressaltar que o número de dependências espaciais satisfeitas é uma informação de

entrada essencial para um aplicativo de monitoramento baseado em aprendizado de máquina, que requer informações de telemetria acopladas para serem utilizadas no processo de aprendizado/inferência.

Para mostrar as razões do baixo desempenho dos concorrentes avaliados, ilustramos na Figura 10(c) a quantidade de fluxos de rede utilizados por cada solução. No eixo x, apresentamos o número de fluxos de rede disponíveis (ou seja, de 50 a 400), enquanto no eixo y o número de fluxos usados. Idealmente, uma solução usaria tantos fluxos de rede quanto possível, uma vez que o objetivo é maximizar o número de dependências espaciais/temporais. No entanto, como podemos observar na figura, o número de fluxos de rede disponíveis geralmente não são utilizados na totalidade. Isso ocorre por três motivos: (i) a capacidade dos fluxos da rede K_f varia, eventualmente proibindo sua utilização por um item de telemetria de grande porte; (ii) os fluxos de rede são roteados através do caminho mais curto, o que pode levar a uma distribuição desigual de fluxos de rede por dispositivos de encaminhamento (ou seja, dispositivos de encaminhamento altamente interconectados teriam muito mais fluxos de rede para incorporar dados de telemetria); e (iii) a decisão tomada pelos algoritmos, conforme ilustrado na figura.



(a) Número de dependências espacial e temporal satisfeitas.

(b) Tempo requerido para solucionar o problema (em escala de logarítmica).

Figura 11 – Métricas de desempenho de orquestração INT aplicadas a grandes instâncias.

Por último, avaliamos nossa proposta heurística em infraestruturas de rede de grande escala, ou seja, variando o tamanho da rede de 100 a 600 em dispositivos de encaminhamento. Definimos o número de aplicativos de monitoramento como 20 e o número de fluxos como 600. A Figura 11(a) ilustra a quantidade de dependências espaço/temporais que são satisfeitas. Observe que para esta configuração, a solução ótima só foi possível executar para um tamanho de rede pequeno de 100 (e demandou mais de 2 horas), para instâncias maiores o quantidade de memória disponível foi o fator limitante . Observamos que, mesmo para instâncias grandes, nossa abordagem supera as abordagens atuais, até 2x melhor para um tamanho de rede de 600.

Para mostrar que a solução do problema usando nossas escalas de procedimento em comparação com a solução ótima, a Figura 11(b) ilustra o tempo necessário para resolver o problema, dado um conjunto crescente de aplicações de monitoramento. Consideramos um tamanho de rede de 100 para podermos comparar com o valor ideal. Observe que o tempo necessário para resolver o problema de forma otimizada cresce exponencialmente - até 7.500 segundos ao considerar 20 aplicativos de monitoramento. No entanto, observamos que o tempo consumido por nossa heurística é até 2 segundos maior do que as abordagens atuais, principalmente devido ao procedimento iterativo de busca local.

4 CONCLUSÃO

O monitoramento de redes de computadores não é uma tarefa trivial, uma vez que complexidade pode crescer rapidamente (com o aumento de dispositivos, serviços, enlaces e requisitos de qualidade). Entretanto, as limitações impostas pelos modelos atuais de monitoramento incentivaram pesquisadores e desenvolvedores a explorar novas formas de monitoramento de infraestrutura de redes. Assim, a Telemetria *In-band*, surge como uma alternativa para prover a visibilidade de rede.

Neste trabalho, em um primeiro momento formalizamos o Plano de Orquestração de Telemetria de Rede *In-band*. A ideia principal consiste em guiar dinamicamente o processo de aquisição de dados de telemetria utilizando mecanismos de aprendizagem. Mostramos que nosso modelo proposto pode coletar com eficácia os itens de telemetria mais importantes e fornecer visibilidade precisa de toda a rede para aplicativos de monitoramento. Nosso modelo supera as heurísticas de última geração por um fator de 2,5x em relação ao número de dependências espaciais satisfeitas e por um fator de 8x ao comparar as anomalias de número identificadas.

Em um segundo momento abordamos a limitação de escalabilidade do problema do plano de orquestração de telemetria de rede *In-band*. Projetamos uma abordagem heurística baseada no procedimento GRASP, incluindo procedimentos construtivos personalizados e de busca local. Mostramos que nossa abordagem pode coletar dados de telemetria de rede com eficácia e satisfazer dependências espaciais/temporais de aplicativos de monitoramento - até 2x melhor do que abordagens de última geração e até 5% pior do que a solução ideal - enquanto exige alguns segundos para executar.

Como trabalhos futuros, pretendemos avaliar a heurística proposta em uma bancada de teste programável realista, integrando-a em controladores SDN e a *smartNICS* de rede. Também analisar o emprego técnica de *path-relinking* que permite melhorar a qualidade das soluções obtidas. Para isso, são exploradas a utilização de soluções elite encontradas através de busca tabu ou *scatter search* (ALVARENGA; ROCHA, 2006).

4.1 Publicações Científicas

1. HOHEMBERGER, R. et al. Orchestrating in-band data plane telemetry with machinelearning.IEEE Communications Letters, v. 23, n. 12, p. 2247–2251, Dec 2019. ISSN2373-7891.

2. HOHEMBERGER, R. et al. A heuristic approach for large-scale orchestration of thein-band data plane telemetry problem. In: BAROLLI, L. et al. (Ed.).Advanced Information Networking and Applications. Cham: Springer InternationalPublishing, 2020. p. 381–392. ISBN 978-3-030-44041-1

REFERÊNCIAS

ALBERT, R.; BARABÁSI, A.-L. Topology of evolving networks: Local events and universality. **Physical Review Letters**, American Physical Society, v. 85, p. 5234 – 5237, Dec 2000. Citado 2 vezes nas páginas 55 e 62.

ALVARENGA, F.; ROCHA, M. L. Melhorando o desempenho da metaheurística grasp utilizando a técnica path-relinking: Uma aplicação para o problema da árvore geradora de custo mínimo com grupamentos. **XXXVIII Simpósio Brasileiro de Pesquisa Operacional**, 2006. Citado na página 67.

ALVES, A. R. Homenetrescue: um serviço sdn para detecção e solução de problemas em redes domésticas. Universidade Federal de Minas Gerais, 2017. Citado 2 vezes nas páginas 15 e 32.

BARBOZA, A. O. et al. Programação linear inteira mista e algoritmo genético aplicados ao problema de transferência e estocagem de produtos em uma indústria petrolífera. **Sistemas & Gestão**, v. 10, n. 4, p. 561–74, 2015. Citado na página 37.

BELFIORE, P.; FÁVERO, L. P. **Pesquisa Operacional para cursos de Engenharia**. [S.l.]: Elsevier Brasil, 2013. v. 1. Citado na página 38.

BOSSHART, P. et al. P4: Programming protocol-independent packet processors. **ACM SIGCOMM 14**, ACM, New York, NY, USA, v. 44, n. 3, p. 87–95, jul. 2014. ISSN 0146-4833. Citado na página 28.

BOSSHART, P. et al. P4: Programming protocol-independent packet processors. **SIGCOMM Comput. Commun. Rev.**, Association for Computing Machinery, New York, NY, USA, v. 44, n. 3, p. 87–95, jul. 2014. ISSN 0146-4833. Disponível em: <https://doi.org/10.1145/2656877.2656890>. Citado na página 32.

CASTRO, A. G. et al. Patcher: Towards fault - tolerant probing planning for in-band network telemetry. In: **2020 IEEE Latin-American Conference on Communications (LATINCOM)**. [S.l.: s.n.], 2020. p. 1–6. Citado 2 vezes nas páginas 44 e 47.

CASTRO, A. G. et al. Near-optimal probing planning for in-band network telemetry. **IEEE Communications Letters**, p. 1–1, 2021. Citado 3 vezes nas páginas 28, 45 e 47.

CHEN, B. et al. Orchestrating segment routing with int for low-overhead and adaptive network monitoring. 2021. Citado 2 vezes nas páginas 46 e 47.

CHOWDHURY, S. R.; BOUTABA, R.; FRANçOIS, J. Lint: Accuracy-adaptive and lightweight in-band network telemetry. In: **2021 IFIP/IEEE International Symposium on Integrated Network Management (IM)**. [S.l.: s.n.], 2021. p. 349–357. Citado 3 vezes nas páginas 28, 46 e 47.

COHEN, R.; KATZIR, L.; RAZ, D. An efficient approximation for the generalized assignment problem. **Information Processing Letters**, v. 100, n. 4, p. 162 – 166, 2006. ISSN 0020-0190. Citado na página 53.

FEO, T. A.; RESENDE, M. G. C. Greedy randomized adaptive search procedures. **Journal of Global Optimization**, v. 6, n. 2, p. 109–133, Mar 1995. ISSN 1573-2916. Citado na página 59.

GAREY, M. R.; JOHNSON, D. S. **Computers and Intractability: A Guide to the Theory of NP-Completeness**. New York, NY, USA: W. H. Freeman & Co., 1979. ISBN 0716710447. Citado na página 50.

GUPTA, A. et al. **Sonata: Query-Driven Network Telemetry**. 2017. Citado 4 vezes nas páginas 15, 42, 46 e 47.

HANDIGOL, N. et al. I know what your packet did last hop: Using packet histories to troubleshoot networks. In: **11th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 14)**. [S.l.: s.n.], 2014. p. 71–85. Citado 2 vezes nas páginas 9 e 27.

HOHEMBERGER, R. et al. Orchestrating in-band data plane telemetry with machine learning. **IEEE Communications Letters**, v. 23, n. 12, p. 2247–2251, Dec 2019. ISSN 2373-7891. Citado 4 vezes nas páginas 28, 29, 49 e 63.

HOHEMBERGER, R. et al. A heuristic approach for large-scale orchestration of the in-band data plane telemetry problem. In: BAROLLI, L. et al. (Ed.). **Advanced Information Networking and Applications**. Cham: Springer International Publishing, 2020. p. 381–392. ISBN 978-3-030-44041-1. Citado 2 vezes nas páginas 29 e 49.

IETF. **Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of Flow Information**. 2013. Disponível em: <https://tools.ietf.org/html/rfc7011>. Citado na página 35.

KLEE, V.; MINTY, G. J. How good is the simplex algorithm. **Inequalities**, New York, v. 3, n. 3, p. 159–175, 1972. Citado na página 59.

KREUTZ, D. et al. Software-defined networking: A comprehensive survey. **Proceedings of the IEEE**, Ieee, v. 103, n. 1, p. 14–76, 2014. Citado na página 31.

LIBERTY, E.; SRIHARSHA, R.; SVIRIDENKO, M. An algorithm for online k-means clustering. In: SIAM. **2016 Proceedings of the eighteenth workshop on algorithm engineering and experiments (ALENEX)**. [S.l.], 2016. p. 81–89. Citado na página 55.

LIU, Z. et al. Netvision: Towards network telemetry as a service. In: **2018 IEEE 26th International Conference on Network Protocols (ICNP)**. [S.l.: s.n.], 2018. p. 247–248. ISSN 1092-1648. Citado 5 vezes nas páginas 27, 28, 43, 46 e 47.

LIU, Z. et al. One sketch to rule them all: Rethinking network flow monitoring with univmon. In: **Proceedings of the 2016 ACM SIGCOMM Conference**. New York, NY, USA: Association for Computing Machinery, 2016. (SIGCOMM '16), p. 101–114. ISBN 9781450341936. Disponível em: <https://doi.org/10.1145/2934872.2934906>. Citado 4 vezes nas páginas 15, 40, 41 e 47.

MAHAJAN, M.; NIMBHORKAR, P.; VARADARAJAN, K. The planar k-means problem is np-hard. **Theoretical Computer Science**, v. 442, p. 13 – 21, 2012. ISSN 0304-3975. Citado na página 54.

MARINS, F. A. S. Introdução à pesquisa operacional. **São Paulo: Cultura Acadêmica: Universidade Estadual Paulista**, 2011. Citado 2 vezes nas páginas 37 e 38.

MARQUES, J. A. et al. An optimization-based approach for efficient network monitoring using in-band network telemetry. **Journal of Internet Services and Applications**, v. 10, n. 1, p. 16, Jun 2019. Citado 7 vezes nas páginas 28, 29, 44, 46, 47, 56 e 63.

MCKEOWN, N. et al. Openflow: Enabling innovation in campus networks. **SIGCOMM Comput. Commun. Rev.**, Association for Computing Machinery, New York, NY, USA, v. 38, n. 2, p. 69–74, mar. 2008. ISSN 0146-4833. Disponível em: <https://doi.org/10.1145/1355734.1355746>. Citado na página 31.

Medina, A. et al. Brite: an approach to universal topology generation. In: **Proceedings of the IEEE MASCOTS**. [S.l.: s.n.], 2001. p. 346–353. ISSN 1526-7639. Citado 2 vezes nas páginas 55 e 62.

MOURA, H. D. Ethanol: uma plataforma sdn para redes wi-fi. Universidade Federal de Minas Gerais, 2015. Citado 2 vezes nas páginas 15 e 32.

OLIVEIRA, H. T. et al. Estudo sobre características de administradores de redes de computadores no brasil para identificação e elaboração de personas. In: SBC. **Anais Principais do XXXV Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos**. [S.l.], 2017. Citado na página 34.

P4 Language Consortium. **P416 Language Specification, version 1.2.1**. 2020. Disponível em: <https://p4.org/p4-spec/docs/P4-16-v1.2.1.htmll>. Citado 2 vezes nas páginas 15 e 33.

PAN, T. et al. Int-path: Towards optimal path planning for in-band network-wide telemetry. In: **IEEE INFOCOM 19**. [S.l.: s.n.], 2019. p. 1–9. Citado 2 vezes nas páginas 28 e 63.

Pan, T. et al. Int-path: Towards optimal path planning for in-band network-wide telemetry. In: **IEEE INFOCOM 2019 - IEEE Conference on Computer Communications**. [S.l.: s.n.], 2019. p. 487–495. Citado 3 vezes nas páginas 43, 45 e 47.

PAN, T. et al. Int-path: Towards optimal path planning for in-band network-wide telemetry. In: **IEEE INFOCOM 19**. [S.l.: s.n.], 2019. p. 1–9. Citado na página 55.

SANTOS, M. Gerência de redes de computadores. RNP/ESR, 2015. Citado 2 vezes nas páginas 15 e 35.

SILVA, C. E. d. et al. Coordenação de múltiplos veículos autônomos de entrega usando k-means e algoritmos bio-inspirados. Universidade Federal de Uberlândia, 2020. Citado na página 39.

SILVA, N. C. B. d. et al. Scpnet-uma arquitetura de monitoramento de rede baseada em políticas. Universidade Federal de Santa Maria, 2018. Citado na página 33.

SIVARAMAN, V. et al. Heavy-hitter detection entirely in the data plane. In: **Proceedings of the Symposium on SDN Research**. New York, NY, USA: Association for Computing Machinery, 2017. (SOSR '17), p. 164–176. ISBN 9781450349475. Disponível em: <https://doi.org/10.1145/3050220.3063772>. Citado 2 vezes nas páginas 41 e 47.

SONG, E. et al. Int-filter: Mitigating data collection overhead for high-resolution in-band network telemetry. In: **GLOBECOM 2020 - 2020 IEEE Global Communications Conference**. [S.l.: s.n.], 2020. p. 1–6. Citado 4 vezes nas páginas 28, 45, 46 e 47.

SUH, D. et al. Flexible sampling-based in-band network telemetry in programmable data plane. **ICT Express**, v. 6, n. 1, p. 62–65, 2020. ISSN 2405-9595. Disponível em: <https://www.sciencedirect.com/science/article/pii/S2405959519302358>. Citado 2 vezes nas páginas 45 e 47.

TAMMANA, P.; AGARWAL, R.; LEE, M. Simplifying datacenter network debugging with pathdump. In: **12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)**. Savannah, GA: USENIX Association, 2016. p. 233–248. ISBN 978-1-931971-33-1. Disponível em: <https://www.usenix.org/conference/osdi16/technical-sessions/presentation/tammana>. Citado 2 vezes nas páginas 41 e 47.

TAMMANA, P.; AGARWAL, R.; LEE, M. Distributed network monitoring and debugging with switchpointer. In: **15th USENIX NSDI 18**. Renton, WA: [s.n.], 2018. p. 453–456. ISBN 978-1-931971-43-0. Citado 4 vezes nas páginas 27, 42, 46 e 47.

TAN, P.-N.; STEINBACH, M.; KUMAR, V. Data mining cluster analysis: basic concepts and algorithms. **Introduction to data mining**, Pearson Education India, p. 487–533, 2013. Citado na página 39.

Trammell, B.; Boschi, E. An introduction to ip flow information export (ipfix). **IEEE Communications Magazine**, v. 49, n. 4, p. 89–95, 2011. Citado 2 vezes nas páginas 15 e 36.

WITTEN, I. H. et al. Practical machine learning tools and techniques. In: **DATA MINING**. [S.l.: s.n.], 2005. v. 2, p. 4. Citado na página 39.

YANG, F. et al. Fast-int: Light-weight and efficient in-band network telemetry in programmable data plane. In: **2020 IEEE 92nd Vehicular Technology Conference (VTC2020-Fall)**. [S.l.: s.n.], 2020. p. 1–5. Citado 2 vezes nas páginas 45 e 47.

YANG, T. et al. Elastic sketch: Adaptive and fast network-wide measurements. In: **Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication**. New York, NY, USA: Association for Computing Machinery, 2018. (SIGCOMM '18), p. 561–575. ISBN 9781450355674. Disponível em: <https://doi.org/10.1145/3230543.3230544>. Citado 3 vezes nas páginas 43, 46 e 47.

ZHU, Y. et al. Packet-level telemetry in large datacenter networks. In: **Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication**. [S.l.: s.n.], 2015. p. 479–491. Citado 3 vezes nas páginas 40, 46 e 47.

Apêndices

APÊNDICE A – ORCHESTRATING IN-BAND DATA PLANE TELEMETRY WITH MACHINE LEARNING

Orchestrating In-Band Data Plane Telemetry with Machine Learning

Rumenigue Hohemberger, Ariel G. Castro, Francisco G. Vogt, Rodrigo B. Mansilha, Arthur F. Lorenzon, Fabio D. Rossi, Marcelo C. Luizelli

***Abstract*—In-band network telemetry (INT) is an emerging network monitoring paradigm. By collecting low-level telemetry items in real time, INT can substantially enhance network-wide visibility - allowing, for example, timely detection problems such as micro-burst. Recent studies have focused on *(i)* developing INT mechanisms to increase network-wide visibility; and *(ii)* to design new monitoring solutions. However, little has been done to coordinate the process of collecting telemetry items in this new paradigm. This is particularly challenging because depending on which network telemetry items are collected, it might degrade network-wide visibility in terms of consistency/freshness. In this letter, we theoretically formalize the In-band Network Telemetry Orchestration Plan Problem and propose a machine learning based orchestration model. Results show that our approach outperforms state-of-the-art heuristics by up a factor of 8x with respect to the number of network anomalies identified, for instance.**

I. Introduction

In-band network telemetry has recently emerged as a promising monitoring alternative to provide higher network-wide visibility to network operators [1]. This finer-grained data plane monitoring mechanism allows to cope with short-lived problems, such as network flow contention, micro-burst, and load imbalance – just to name a few [2], [3]. Yet, data plane telemetry is paramount to the success of real-time network applications with stringent responsiveness requirements, such as virtual/augmented reality [4] and self-driving cars [5]. Much of the progress in the field has been enabled by recent advances in programmable network devices and high-level domain-specific networking description and query languages [6].

By using programmable devices, in-band data plane telemetry allows to collect and encapsulate low-level telemetry information into production network traffic whenever possible. Packets contain header fields that are interpreted as telemetry instructions by network devices. These instructions instrument these devices to collect and write into the packet network states [1]. This telemetry information might encompass switch-internal states (*e.g.*, queue occupancy and switchID) and network performance metrics (*e.g.*, data plane processing time and latency), for instance. In the process of in-band telemetry, the collected information is carried into a packet along its routing path and, at some point in the network, it is extracted and reported to a monitoring application. Then, the monitoring applications process and eventually react to spurious networks events.

Recent approaches [1], [2], [3], [7], [8] have striven to deliver the best out of the in-band telemetry to improve network-wide visibility. PathDump [7] and SwitchPointer [3] combines in-network programmability and the available end-host resources to collect and monitor telemetry data in order to debug networks events. More recently, Liu et al.[1] and

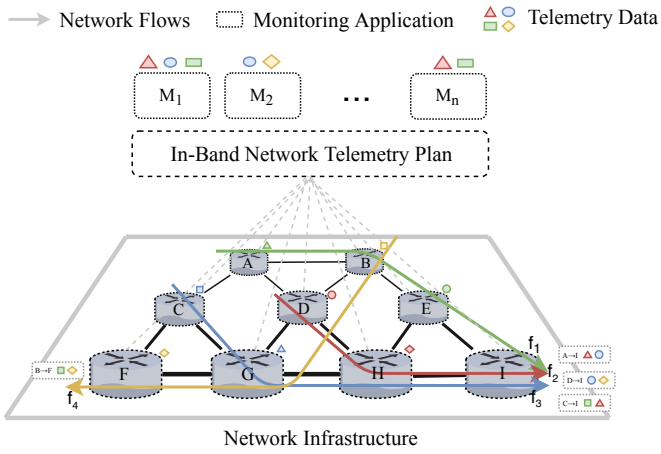


Fig. 1. Example of in-band network telemetry planning. The example illustrates a snapshot where four active network flows (f_1, f_2, f_3, f_4) collect telemetry data from forwarding devices. Telemetry data are then sent to monitoring applications according to their needs.

proposed NetVision, an attempt to provide network telemetry as a service, while Marques et al. [9] provided the first effort towards in-band network orchestration. Despite these efforts to make real-time in-band telemetry a reality in programmable networks, little has yet been done to dynamically coordinate how to collect network information in this new paradigm. This is particularly challenging mainly for two reasons. First, depending on which network telemetry items are collected, it might degrade network-wide visibility in terms of coverage, consistency, and freshness [9]. Second, depending on how network telemetry is collected, it might impact the network monitoring application's performance.

In this letter, we introduce the In-band Network Telemetry Orchestration Plan Problem as a machine learning aided optimization model. The main idea consists of dynamically (and wisely) guiding the in-band data acquisition by means of a learning mechanism. To tackle this problem, we theoretically formalize it as a Mixed Integer Linear Programming (MILP) model. The model can be seen as a generalization of the well-known Bin Packing problem [10] and, therefore, it is an NP-hard problem. Despite the scalability limitations on solving NP-hard problems, this exact formulation represents an optimal bound for future in-band network telemetry approximations. To the best of our knowledge, this is the first attempt to formulate this problem. Results show that the proposed model outperforms [9] by up a factor of 8x with respect to the number of network anomalies identified, increasing

the number of telemetry data collected, and maintaining the consistency and freshness of network visibility.

II. The In-band Network Telemetry Orchestration Plan Problem

A. Problem Overview

The in-band network telemetry consists of embedding telemetry information into production flow packets. Fig. 1 illustrates a given network infrastructure with four active network flows (namely, f_1, f_2, f_3, and f_4) being routed through a set of forwarding devices – ranging from A to I. Network flow f_1 can collect telemetry items from its forwarding devices $\{A, B, E, I\}$. On top of the network, there exists a set of specialized monitoring application $\{M_1, M_2, ..., M_n\}$. These applications are in charge of making inference about specific network problems or (mis)behaviors (*e.g.*, identify network congestion or malicious network attacks) and eventually react to them, demanding different pieces of information from the infrastructure. For instance, let us consider that application M_1 aims to predict and identify (transient) network congestion. It requires network flows to collect from forwarding devices the number of active network flows ($\triangle$), queue occupancy ($\bigcirc$), and processing time directly from data planes ($\square$). In turn, M_2 identifies SYN flood attacks. For that, it requires to constantly monitor the number of active network flows ($\triangle$), and the number of incomplete TCP handshakes ($\Diamond$). Observe that these monitoring applications are not limited to the given examples and may demand different subsets of telemetry items (coverage) to be continuously collected in a given rate (freshness) so that the inference accuracy can be met over time.

The In-band Network Telemetry Plan Problem consists of dynamically orchestrating the process of collecting network information to maximize network monitoring applications performance without, however, penalizing production network flows. The problem is far from being trivially solved. This is due to *(i)* monitoring applications might have completely different requirements – which implies that some telemetry items might be collected more frequently than others; and *(ii)* network packets have limited unused resources (in general, up to the MTU data link limit) – and, therefore, it is unfeasible to collect all items within the same time unit. To illustrate how challenging this problem is, consider the scenario from Fig. 1. Let us assume network flows can collect and carry a maximum fixed amount of telemetry items at a specific time unit (*e.g.*, two items at a time unit). There are a few alternatives to orchestrate the acquisition of INT items. First, a naïve solution consists of collecting all telemetry items in every single time unit. For building such a solution, we relax the assumption that network packets/frames are space-bounded. A second (and more realistic) strategy takes into account an upper-bound limit on the amount of collected telemetry items. Therefore, the number of telemetry items collected by network flows are rather constrained than the naïve strategy. In fact, it turns out that introducing this set of constraint makes the problem NP-hard (as it is a generalization of the well-studied Bin Packing problem [10]). In spite of having hard constraints on the number of items to be collected in a single time unit, a relaxed solution (with respect to time) might be built by round-robining the collection of telemetry items over time. Although this might represent a feasible (but relaxed) solution over time, it misses one fundamental aspect of network telemetry, namely to identify which telemetry items are in fact important to be collected. This information is rather important in order to ensure network-wide visibility of the network infrastructure accurately.

B. Model description and notation

The optimization model we propose to solve the orchestration problem defined above considers a physical network infrastructure $G = (D, L)$, a set of active network flows F, a set of telemetry items V, and a set of monitoring applications M. Set D in network G represents programmable forwarding devices $D = \{1, ..., |D|\}$, while set L links interconnecting pair of devices $(d_1, d_2) \in (D \times D)$. Each forwarding device $d \in D$ is able to embed a subset of items $V_d \subseteq V$ into packets of flow $f \in F$. Each telemetry item $v \in V$ has its size defined by function $S : V \to \mathbb{N}^+$.

Network flows F are used to collect real-time telemetry data from forwarding devices D. A flow $f \in F$ has two endpoints (*i.e.*, ingress and egress forwarding devices) and is routed through the network infrastructure G using a simple path $\mathcal{P}t$. We denote the path taken by flow f as function $\mathcal{P}t : F \to \{D_1 \times ... \times D_{|D|}\}$. Network flows F are encapsulated in a forwarding protocol (*e.g.*, NSH[1], IPv4). Therefore, the amount of available space to embed telemetry items in packets is bounded by a constant $K_f \in \mathbb{N}^+$. Observe that a single packet of flow $f \in F$ can collect at most $\sum_{\forall d \in \mathcal{P}t(f)} V_d$ telemetry items from the network infrastructure G at a given time frame. Unless K_f assumes a sufficient large value (*i.e.*, $K_f \geq \sum_{\forall d \in \mathcal{P}t(f)} |V_d|$), it is not possible to collect all items from all forwarding devices in the routed path $\mathcal{P}t(f)$.

Monitoring application $m \in M$ requires a subset of telemetry items $R_m \subseteq V$ to operate properly. These telemetry items might have spatial and temporal dependencies. We say two or more telemetry items have spatial dependency *iff* they must be collected from the same forwarding device $d \in D$. Spatial dependencies are represented by the set of sets $R^s_m \subseteq \mathscr{P}(R_m)$. In turn, we say that a set of telemetry items have temporal dependency *iff* they must be collected within a given deadline. Temporal dependencies are represented by the set of sets $R^t_m \subseteq R^s_m$, and the required deadline is expressed as a function $T^f : R^t_m \to \mathbb{N}^+$. The model keeps track of the last time unit an item $P \in R^t_m$ was collected by means of a function $H^f : R^t_m \to \mathbb{N}^+$. When $(\forall P \in R^t_m) : H^f(P) > T^f(P)$, item P is out of date (*i.e.*, the deadline has expired).

C. Naïve Orchestration Model

Given a network infrastructure G, a set of network flows F, a set of monitoring applications M, a set of telemetry items V and a constant K_f, the optimization problem seeks a feasible solution that maximizes the number of spatial and

[1]https://tools.ietf.org/html/rfc8300

temporal dependencies. The model output is denoted by a 3-tuple $\chi = \{Y, S^b, T^b\}$. Variables from $Y = \{\, y_{d,v,f}\,, \forall\, d \in D, v \in V, f \in F\}$ indicate that a forwarding device d embed telemetry item v into packets of network flow f. Variables from $S^b = \{\, s^b_{m,d,P}\,, \forall m \in M, d \in D, P \in R^s_m\}$ and $T^b = \{\, t^b_{m,P}\,, \forall m \in M, P \in R^t_m\}$ are used to keep track of spatial and temporal dependencies satisfied by the model. Next, we describe the MILP formulation for this orchestration problem.

$$\textbf{Maximize} \qquad \sum_{m \in M} \sum_{p \in P \in R^s_m} \sum_{d \in D} s^b_{m,d,p} + t^b_{m,p} \tag{1}$$

Subject to:

$$\sum_{d \in \mathcal{P}t(f)} \sum_{v \in V_d} y_{d,v,f} \cdot S(v) \leq K_f \qquad \forall f \in F \tag{2}$$

$$\sum_{f \in F} y_{d,v,f} \leq 1 \qquad \forall d \in D, v \in V_d, \tag{3}$$

$$s_{m,d,p} = \sum_{v \in P} \sum_{f \in F} y_{d,v,f} \qquad \forall m \in M, p \in P \in R^s_m, d \in D \tag{4}$$

$$t_{m,p} = \sum_{v \in P} \sum_{d \in D} \sum_{f \in F} y_{d,v,f} \qquad \forall p \in P \in R^t_m : H^f(p) > T^f(p) \tag{5}$$

$$s^b_{m,d,p} \leq \frac{s_{m,d,p}}{|P|} \qquad \forall m \in M, p \in P \in R^s_m, d \in D \tag{6}$$

$$t^b_{m,p} \cdot \leq \frac{t_{m,p}}{|P|} \qquad \forall m \in M, p \in P \in R^t_m \tag{7}$$

$$y_{d,v,f} \in \{0,1\} \qquad \forall d \in D, v \in V, f \in F \tag{8}$$

$$s_{m,d,p} \in \mathbb{N}^+ \qquad \forall m \in M, d \in D, p \in P \in R^s_m \tag{9}$$

$$s^b_{m,d,p} \in \{0,1\} \qquad \forall m \in M, d \in D, p \in P \in R^s_m \tag{10}$$

$$t_{m,p} \in \mathbb{N}^+ \qquad \forall m \in M, p \in P \in R^t_m \tag{11}$$

$$t^b_{m,p} \in \{0,1\} \qquad \forall m \in M, p \in P \in R^t_m \tag{12}$$

Constraint set (2) ensures that a network flow $f \in F$ does not exceed its capacity (*i.e.*, K_f). Constraint set (3) ensures that a single telemetry item is collected at most by a single network flow $f \in F$ in a given device d. Constraints sets (4) and (6) aim to account whether or not a spatial dependency is met. Given a spatial dependency, constraint set (4) count the number of telemetry items collected in a device d, while constraint set (6) verify whether or not the dependency is met (*i.e.*, checking if all items were collected). Similarly, constraints sets (5) and (7) count the number of temporal dependencies that are satisfied. Last, constraints sets (8)-(12) define the domains of output variables.

D. Machine Learning based Orchestration Model

The orchestration model proposed in the previous subsection is able to maximize the number of collected telemetry items. However, it still misses the fundamental orchestration question: *which telemetry items are in fact important to be collected?* Similar models in the literature (*e.g.* [11]) handle the importance of a given item by means of arbitrary weights, which needs to be manually fine-tuning from time to time, in a static fashion. To fill in this gap, we plug in our model a learning mechanism in order to dynamically (and wisely) instrument the collection of telemetry items from the network.

The importance of a telemetry item depends on the requirements of monitoring applications, which changes dynamically over time according to the network behavior. We assume that in a given time unit t, the proposed learning model knows a subset of telemetry items, represented by $V^t \subseteq \sum_{t-W}^{t} \sum_{m \in M} R^s_m$, where $W \in \mathbb{N}^+$ represents a given time window. An element $P \in V^t$ represents a $|P|$-dimensional space tuple $P = (v_1, v_2, \cdots v_{|P|})$, where each dimension represents a telemetry information collected from the infrastructure, *i.e.* $v_{|S|} \in \mathbb{R}^+$. Given a set V^t of telemetry data collected within the last W time units, we model the In-Band Network Telemetry Plan layer (as shown in Fig. 1) as multiple *online* clustering models. The idea consists of clustering network behaviors based on telemetry items that share the same dependencies. The orchestration model, therefore, can coordinate which telemetry items are important to be collected next (*e.g.*, telemetry items observing unusual behaviors). Each cluster keeps track of items having the same $|P|$ dimensions. We partition $V^t = \{V^{t,1}, V^{t,2}, ..., V^{t,\mathscr{P}(V^t)}\}$ so that each subset has items of the same $|P|$ dimensions. We then cluster subsets $V^{t,i} \in V^t$ into K_i exclusive partitions, *i.e.* $(\forall i \in \{1, ..., \mathscr{P}(V^t)\}) : V^{t,i}_1, V^{t,i}_2, ..., V^{t,i}_{K_i}$. The centroid of a subset $V^{t,i}_k$ is defined as $C(V^{t,i}_k)$. An element $P \in V^{t,i}$ is assigned to a cluster $V^{t,i}_k$ if the distance of v to $(V^{t,i}_k)$ is minimal. We omit the definition of the clustering problem (as an optimization problem) due to space constraints. However, it is important to mention that clustering problems are well-known NP-hard problems [12] – even for planar graphs.

Next, we define how the model keeps cluster consistency and freshness over time. In order to capture the network dynamics, we propose a fading function that estimates how long the model keeps telemetry data in it. Telemetry data measured with higher variance in a time frame W tends to be spanned through a lower number of time units than those with lower variance. The variance of a telemetry item (or a P-tuple) is defined according to its dispersion index. We denote the dispersion index as $I(v) = \frac{\sigma^2_v}{\mu}$. The number of time units a telemetry item is kept in V^t is defined according to a piecewise function $T(P) : V^t \rightarrow \mathbb{N}^+$, defined as follow:

$$T(P) = \begin{cases} W & I(P) \leq 1 \\ W \cdot (1 - \rho) & I(P) > 1 \end{cases} \tag{13}$$

Telemetry data points $P \in V^t$ that are under-dispersed (i.e, $I(P) \leq 1$) are kept in the clustering for at least W time units. In turn, over-dispersed data points (*i.e.*, $I(P) > 1$) tends to be in the cluster for fewer time units (W is decayed as with respect to $\rho \in [0, 1]$). Observe that this approach allows the model to capture transient events (*e.g.*, short-time queue occupancy) and also persist events the endure to longer time units (*e.g.*, physical disruption in a forwarding device). We denote by $V^+ \subseteq V^t$ and $V^- \subseteq V^t$ the subset of telemetry items that are over- and under-dispersed, respectively. Over-

dispersed items with spatial dependencies are represented by set $R_m^{+s} = \{P \in R_m^{+s} | P \in R_m^s \wedge (P \cap V^+ \neq \emptyset)\}$. Similarly, under-dispersed items with spatial dependencies by set $R_m^{-s} = \{P \in R_m^{-s} | P \in R_m^s \wedge (P \cap V^- \neq \emptyset)\}$. We are interested in optimizing how we evolve over time the information we know about the infrastructure G, maintaining $V^{t,i} \in V^t$ clusters over time. We assume most of the time, the value of telemetry points evolves smoothly. Abrupt changes happen to the network state, but with low probability. This assumption is realistic to in-band network telemetry. Our main goal is to reduce the acquisition of irrelevant telemetry data and, at the same time, maintaining the accuracy of acquired telemetry data. The machine learning based orchestration model replaces Equation (1) by Equation (14).

$$\textbf{Maximize} \quad \alpha \cdot \sum_{m \in M} \sum_{p \in P \in R_m^+} \sum_{d \in D} s_{m,d,p}^b + t_{m,p}^b + \beta \cdot \sum_{m \in M} \sum_{p \in P \in R_m^-} \sum_{d \in D} s_{m,d,p}^b + t_{m,p}^b \qquad (14)$$

The combined objective function tries to wisely maximize the number of collected over-dispersed and under-dispersed items from G. Observe that the objective function prioritizes the collection of over- and under-dispersed items according to parameters α and β. That is explained because, in regular network operation, the dispersion index tends to lower while there is not a suspicious event. On the event of a suspicious event, the index would tend to increase and therefore there will high the chances to be collected.

III. Evaluation

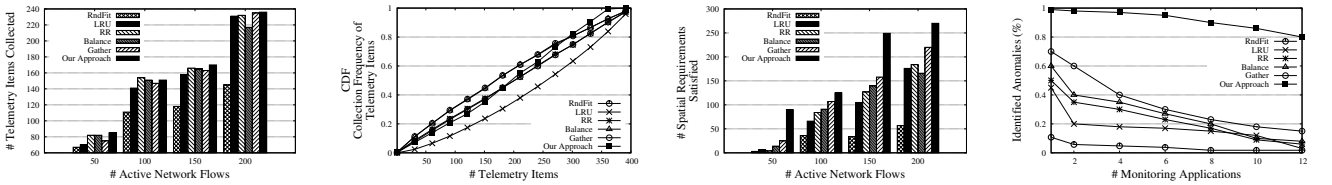
Setup. We ran the proposed model using IBM *CPLEX Optimization Studio* 12.9 to obtain optimum solutions and implemented the online version of K-Means [13] using Java language to dynamically ponderate the importance of telemetry items[2]. Experiments were performed on a machine with four Intel Xeon E5-2670 processors and 56 GB of RAM, using the Ubuntu Server 11.10. We considered different physical network instances that were generated with Brite [14], following the Barabasi-Albert model [15]. We used physical infrastructures consisting of 50 forwarding devices and, on average, 200 physical links. On top of each infrastructure instance, there was a set F of active flows. We varied the amount of network flows from 50 to 200. Each flow $f \in F$ interconnected two random endpoints in the infrastructure and, was routed using the shortest path algorithm. We consider IP-based network flows and, therefore, we vary uniformly the constant K_f from 10-30 Bytes. Further, we assume that forwarding devices have 8 possible telemetry items to export[3], varying $S(v)$ from 2 to 20 Bytes [16]. We consider a set of monitoring applications, ranging from 2 to 12. Each application requires at most 4 spatial dependencies, each having at least 2 items (*i.e.*, switch ID + monitored metric) and at most 4. We consider $\alpha = 1$, $\beta = 1$, $W = 5$, and $\rho = 0.5$. These parameters can be fine-adjusted to prioritizes the collection of spatial dependencies over temporal ones and to adjust the fading functions. Our experiment ran in 200 time units. At each time unit, our model and algorithms are run. Each experiment is repeated 30 times to ensure a confidence level of 95% or higher.

Baseline. We compare our proposed model against *(i)* two state-of-the-art orchestration heuristics proposed by [9], namely `Gather` and `Balance`, and *(ii)* three heuristics: Round-Robin (`RR`), Least-Recently Collected (`LRC`), Random-Fit (`RndFit`). The heuristic `RR` strives to assign telemetry items to network flows in a round-robin way. `LRC` prioritizes the collection of telemetry items that have not been collected recently. Finally, `RndFit` chooses randomly telemetry items and tries to assign to random network flow.

Results. We focus our analysis on two key aspects of network monitoring: *(i)* network-wide visibility (in terms of coverage), and *(ii)* network monitoring applications performance. For that, we evaluate network telemetry coverage (*i.e.*, the number of telemetry items collected), a number of spatial dependencies satisfied, and the ability to identify network anomalies. Fig. 2(a) illustrates the average amount of telemetry items collected over time for an increasing number of active network flows. We observe that *(i)* number of network flows directly impacts on the network visibility (*i.e.*, the more active network flows, the higher the network coverage) and *(ii)* the proposed orchestration model on average outperform evaluated heuristics by up a factor of 1.7x when comparing to `RndFit`. This behavior is due to the ability to optimally assign telemetry items to network flows. In contrast, evaluated heuristics make local decisions on which item assign (*e.g.*, least recently collected) to each flow, leading to sub-optimal solutions. On average, our proposed orchestration model covers 60% of telemetry items in each time unit (considering 400 telemetry items in total). Next, we evaluate the number of times (frequency) telemetry items are collected, illustrated in Fig. 2(b) by means of a CDF (Cumulative Distribution Function). Despite showing in Fig. 2(a) that the number of items collected by our model and by the baselines are similar, we observe in Fig 2(b) that each method prioritizes the collections of different subsets of telemetry items. This difference occurs due to *(i)* the heuristic assignment policy and *(ii)* the number of active network flows available to assign telemetry items. The latter point can be explained by the presence of highly interconnected forwarding devices in the network topologies – in which telemetry items are potentially collected more frequently than others. This fact is further exacerbated as active network flows are routed using shortest paths. Observe that our orchestration model, differently from evaluated heuristics, prioritizes the collection of a subset of telemetry items – on average, 25% of telemetry items are collected more intensely (more than 80% of the time units). In contrast, up to 40% of telemetry items are collected less frequently (less than 40% of the time units). Fig. 2(c) illustrates the average number of spatial dependencies satisfied over time. As observed, our model outperforms evaluated heuristics by up a factor of 2.5x (for 150 active network flows). The number of spatial dependencies satisfied is essential input information to

[2]Reproducible material available on https://github.com/mcluizelli/comml-int19

[3]In-band Network Telemetry: https://p4.org/assets/INT-current-spec.pdf



(a) Telemetry Items collected.

(b) Number of times telemetry items are collected (200 network flows).

(c) Spatial requirements satisfied.

(d) Anomaly events detected (200 network flows).

Fig. 2. In-band telemetry orchestration performance metrics.

learning models, which required coupled telemetry information – otherwise, telemetry data are useless for the learning process. Last, Fig. 2(d) illustrates the average number of network anomalies identified. For this experiment, we consider an increasing number of monitoring applications running in parallel and, therefore increasing the need for a higher number of spatial dependencies. Further, we assume that telemetry items values follow a normal distribution, defined by an average and standard deviation (*e.g.*, queue occupancy given by the average of 100 and deviation of 20). We then inject anomalies to these items (*i.e.*, values out from the expected range), lasting for 5 times units (on average) on (at most) 10% of forwarding devices. When the number of monitoring applications is relatively small (up to 2 monitoring applications), machine learning mechanism can identify up to 50-60% of network anomalies with the input provided by telemetry data collected by the evaluated approaches. However, as the number of monitoring applications increases (and, therefore, the number of spatial dependencies), the ability to identify network anomalies decreases considerably, reaching up to 5-10% of identified anomalies. The reason is correlated with the frequency with that telemetry items are collected from the network. Baseline algorithms strive to collect telemetry items in a fairly way (e.g., round-robin) – which lowers the overall frequency that telemetry items are collected and therefore higher the chances to miss the collection of anomalous events. In contrast, our model can cope with the increasing number of monitoring applications. The input provided by our model can feed machine learning mechanisms models in order to identify up to 97% of network anomalies (up to 4 monitoring applications). From 4-12 network monitoring applications, the accuracy of identifying network anomalies decreases by 20% (*i.e.*, 8x higher than the baselines).

we intend to investigate the design of dynamic and scalable heuristic/approximation algorithms, derive upper-bounds to the proposed model and to integrate to commercial offerings.

IV. Final Remarks

In this letter, we formalize the In-band Network Telemetry Orchestration Plan by means of a MILP model. The main idea consists of dynamically guiding the acquisition process of telemetry data using learning mechanisms. We showed that our proposed model can effectively collect the most important telemetry items and provide accurate network-wide visibility to monitoring applications. Our model outperforms state-of-the-art heuristics by a factor of 2.5x with respect to the number of spatial dependencies satisfied and by a factor of 8x when comparing the number anomalies identified. As future work,

References

[1] Z. Liu, J. Bi, Y. Zhou, Y. Wang, and Y. Lin, "Netvision: Towards network telemetry as a service," in *2018 IEEE 26th International Conference on Network Protocols (ICNP)*, Sep. 2018, pp. 247–248.

[2] B. Arzani, S. Ciraci, L. Chamon, Y. Zhu, H. H. Liu, J. Padhye, B. T. Loo, and G. Outhred, "007: Democratically finding the cause of packet drops," in *15th USENIX NSDI 18*. Renton, WA: USENIX Association, 2018, pp. 419–435.

[3] P. Tammana, R. Agarwal, and M. Lee, "Distributed network monitoring and debugging with switchpointer," in *15th USENIX NSDI 18*, Renton, WA, 2018, pp. 453–456.

[4] R. I. T. da Costa Filho, M. C. Luizelli, M. T. Vega, J. van der Hooft, S. Petrangeli, T. Wauters, F. De Turck, and L. P. Gaspary, "Predicting the performance of virtual reality video streaming in mobile networks," in *ACM MMSys '18*. New York, NY, USA: ACM, 2018, pp. 270–283.

[5] G. Ananthanarayanan, P. Bahl, P. Bodk, K. Chintalapudi, M. Philipose, L. Ravindranath, and S. Sinha, "Real-time video analytics: The killer app for edge computing," *Computer*, vol. 50, no. 10, pp. 58–67, 2017.

[6] P. Bosshart, D. Daly, G. Gibb, M. Izzard, N. McKeown, J. Rexford, C. Schlesinger, D. Talayco, A. Vahdat, G. Varghese, and D. Walker, "P4: Programming protocol-independent packet processors," *ACM SIGCOMM 14*, vol. 44, no. 3, pp. 87–95, Jul. 2014.

[7] P. Tammana, R. Agarwal, and M. Lee, "Simplifying datacenter network debugging with pathdump," in *12th USENIX OSDI 16)*, Savannah, GA, 2016, pp. 233–248.

[8] Y. Zhu, N. Kang, J. Cao, A. Greenberg, G. Lu, R. Mahajan, D. Maltz, L. Yuan, M. Zhang, B. Y. Zhao, and H. Zheng, "Packet-level telemetry in large datacenter networks," in *ACM SIGCOMM 15*. New York, NY, USA: ACM, 2015, pp. 479–491.

[9] J. A. Marques, M. C. Luizelli, R. I. T. Da Costa, and L. P. Gaspary, "An optimization-based approach for efficient network monitoring using in-band network telemetry," *Journal of Internet Services and Applications*, vol. 10, no. 1, p. 16, Jun 2019.

[10] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. New York, NY, USA: W. H. Freeman & Co., 1979.

[11] R. Cohen, L. Katzir, and D. Raz, "An efficient approximation for the generalized assignment problem," *Information Processing Letters*, vol. 100, no. 4, pp. 162 – 166, 2006.

[12] M. Mahajan, P. Nimbhorkar, and K. Varadarajan, "The planar k-means problem is np-hard," *Theoretical Computer Science*, vol. 442, pp. 13 – 21, 2012.

[13] E. Liberty, R. Sriharsha, and M. Sviridenko, "An algorithm for online k-means clustering," in *Proceedings of the Workshop on Algorithm Engineering and Experiments (ALENEX)*, pp. 81–89.

[14] A. Medina, A. Lakhina, I. Matta, and J. Byers, "Brite: an approach to universal topology generation," in *Proceedings of the IEEE MASCOTS*, Aug 2001, pp. 346–353.

[15] R. Albert and A.-L. Barabási, "Topology of evolving networks: Local events and universality," *Physical Review Letters*, vol. 85, pp. 5234 – 5237, Dec 2000.

[16] T. Pan, E. Song, Z. Bian, X. Lin, X. Peng, J. Zhang, T. Huang, B. Liu, and Y. Liu, "Int-path: Towards optimal path planning for in-band network-wide telemetry," in *IEEE INFOCOM 19*, Apr 2019, pp. 1–9.

APÊNDICE B – A HEURISTIC APPROACH FOR LARGE-SCALE ORCHESTRATION OF THE IN-BAND DATAPLANE TELEMETRY PROBLEM

A Heuristic Approach for Large-Scale Orchestration of the In-Band Data Plane Telemetry Problem

Rumenigue Hohemberger, Arthur Francisco Lorenzon, Fábio Diniz Rossi, Marcelo Caggiani Luizelli

Abstract

In-band network telemetry (INT) is an emerging network monitoring paradigm aimed to collect data plane telemetry statistics. INT is able to improve network-wide visibility considerably, providing timely discovery of issues such as micro-burst. Previous researches have focused on optimally orchestrating the collection of in-band network telemetry statistics from the network, providing little or none scalability to be applied in realistic environments. In this work, we tackled the scalability limitation of existing approaches by proposing an iterated local search procedure. Results show that our algorithm outperforms state-of-the-art orchestration solutions by a factor of 2 with respect to the number monitoring applications requirements satisfied, while keeping the solution close to the optimum (less than 5%) – demanding a few seconds to execute.

1 Introduction

In-band network telemetry (INT) is a mainstream data plane monitoring technique aimed to provide comprehensive, in-depth network-wide visibility to network operators [7]. This higher network-wide visibility will ultimately help network operators to troubleshoot short-lived network events promptly (e.g., network flow contention, micro-burst, and load imbalance [13]). In short, the INT concept allows collecting and encapsulating low-level data plane telemetry information into network traffic packets.

Examples of such low-level information include switch-internal states (e.g., queue occupancy and switch ID) and network performance metrics (e.g., data plane processing time and latency). Productions packets (or specially-crafted probing packets) are in charge of driving the collection of these telemetry data as they are being routed along with the network infrastructure. These packets carry specific telemetry instructions, which are interpreted by network

Marcelo C. Luizelli, Arthur F. Lorenzon
Federal University of Pampa (UNIPAMPA)
Alegrete, Brazil, e-mail: {marceloluizelli,arthurlorenzon}@unipampa.edu.br

Rumenigue Hohemberger, Fabio D. Rossi
Federal Institute Farroupilha (IFFAR)
Alegrete, Brazil, e-mail: {rumenigue.hohemberger,fabio.rossi}@iffarroupilha.edu.br

devices, setting forwarding devices to collect the required network states [7]. At some location in the network, the collected telemetry data is extracted and sent to monitoring applications, which process and finally react to spurious network events. For example, suppose a given monitoring application is in place to predict network congestion, demanding the collection of queue occupancy and processing time from forwarding devices (i.e., monitoring requirements).

The INT concepts have been fostered by industry[1] and academic initiatives, mainly due to the rapid advances in programmable network devices (e.g., SmartNICs) and domain-specific networking description languages (e.g., P4 language). Recent research efforts [10, 5, 7, 8] have made the first efforts towards the orchestration of in-band network telemetry data collection to improve network-wide visibility. For instance, Liu et al. [7] and Pan et al. [10] have focused on performing network telemetry using probing packets, while Marques et al. [8] and Hohemberger et al. [5] have focused on the embedding of telemetry data into production network packets. The orchestration of in-band network telemetry is challenging mostly for two reasons. First, depending on which network telemetry items are collected, it might degrade network-wide visibility in terms of coverage and freshness [8]. Second, and more importantly, depending on how network telemetry is collected, it might affect the network monitoring application's performance [5].

Despite these efforts, most of the orchestration solutions have neglected monitoring application requirements and provide little scalability to be applied in real environments. To fill in this gap, in this paper, we address the scalability of the in-band network telemetry orchestration plan problem considering monitoring requirements. We propose an iterated local search based heuristic to solve the problem so as to produce high-quality solutions for large scale network infrastructures. Our proposed heuristic strives to satisfy the higher number of monitoring applications requirements. Results show that the proposed algorithm outperforms state-of-the-art orchestration solutions [5, 8] by a factor of 2x with respect to the number monitoring applications satisfied, while keeping the solution close to the optimum (on average, less than 5%). More importantly, we show that our heuristic can timely cope with large-scale network infrastructures.

2 Related Work

To the best of our knowledge, the in-band network telemetry plan problem has not been investigated before the inception of programmable data planes. Recent advances in forwarding devices (e.g.,[2]) have enabled to continuously push telemetry information to data collectors – known as Model-Driven Telemetry (MDT). In this context, Putina et al. [11] proposed a mechanism for real-time detection of BGP anomalies, relying on machine-learning techniques and MDT-based telemetry data streaming. In turn, Mazieres et al. [6] introduced the seminal work on in-band network telemetry, introduc-

[1] For instance: Barefoot Advanced Network Telemetry and Broadcom Trident 3

ing the concept of tiny packet programs. Everflow [14] extended the proposed INT concept by exploring the "match-and-mirror" functionality of commodity switches. Everflow utilizes the INT concept to filter out packets that satisfy a given pattern and send them to multiple data analyzers.

More recently, Gupta et al. [4] designed SONATA, a high-level interface to express telemetry queries. Based on data plane constraints, monitoring queries are partitioned and processed in multiple devices – ensuring that queries and the packet forwarding operate at line rate. Other studies have proposed to execute specific telemetry operations (e.g., heavy hitters) directly by the data plane. Tammana et al. proposed PathDump [12], a mechanism designed to identify and debug anomalous behaviors in programmable network infrastructure. PathDump keeps track of the packet's route for subsequent analysis of it, employing INT instruction in the forwarding devices. SwitchPointer [13] further advances that approach, proposing to collect end-host telemetry information to enhance debugging capabilities.

Last, Liu et al. [7] and Pan et al. [10] have focused on performing network telemetry through active INT probe packets. Both strategies have relied on *Euler Circuits* to indicate the path to be taken by active probe packets. In turn, Marques et al. [8] and Hohemberger et al. [5] have focused on the embedding of telemetry data into production network packets. Marques et al. [8] designed heuristic approaches to orchestrate how live packets collect network telemetry data, while Hohemberger et al. [5] designed a machine learning-based model that wisely collects telemetry data based on its importance.

As can be observed, most of the research efforts related to the in-band telemetry are still restricted to mechanisms that mostly utilize collected telemetry data for new monitoring solutions [6, 14, 12, 13]. The studies introduced by [10, 5, 7, 8] represent the first steps towards the orchestrating of in-band network telemetry. However, these studies have mainly focused on maximizing the amount of telemetry data collected from the network infrastructure without considering monitoring applications' requirements. This leads to sub-optimal solutions (as later shown in our experiments) and limits the operationalization of the proposed strategies in a production environment. As a first step towards a scalable orchestration solution, this work advances the state-of-the-art by proposing a heuristic-based approach to timely solves the in-band network telemetry problem.

3 The In-band Network Telemetry Orchestration Plan Problem

3.1 Problem Overview

The in-band network telemetry plan problem is an NP-hard problem [5] and it consists of orchestrating the process of collecting network telemetry data in order to maximize the number of network monitoring requirements satisfied. To illustrate the problem, consider the scenario from Fig. 1. In the problem, there is a set of monitoring applications (M_1, M_2, M_3, and M_4) demanding potentially different subsets of telemetry items and a set of network flows

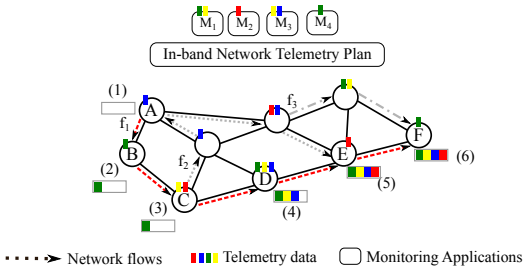


Fig. 1 Example of in-band network telemetry planning.

(f_1, f_2, and f_3) that are used collect and route subsets of items to network monitoring collectors. Network flow packets have limited unused resources, and, therefore, it is unfeasible to collect and embed all items in a single packet.

Let us assume network flows can collect and carry a maximum fixed amount of four telemetry items at a specific time frame. In (1) the network flow packet f_1 starts being routed from A to F. Then, the network flow collects the green telemetry item in (2) – satisfying the requirement of monitoring application M_4. Then, in (3), the packet is routed through device C and, in (4), it collects the yellow and blue items from the same forwarding device, satisfying the requirement of the monitoring application M_3. Observe that, despite collecting the green and yellow telemetry items (as demanded by monitoring application M_1), they were not collected from the same device and, therefore, the monitoring requirement was not satisfied. Last, the packet is routed to forwarding device E, and it collects the red telemetry item in (5) – satisfying the requirement of the monitoring application M_2. Finally, it reaches device F, where all these telemetry items are extracted and reported to an INT collector before sending the original packet to its final destination. Observe that this solution is optimal concerning the maximum number of monitoring requirements satisfied.

3.2 Problem Formulation

We now formally define the in-band network telemetry plan problem as an optimization model. We adopt a revised version of the model proposed by [5], which captures the monitoring aspects we are currently interested in. First, we describe both the input and output of the model and establish a supporting notation. Then, we formally present the optimization problem.

Input. The optimization model takes as input a physical network infrastructure $G = (D, L)$, a set of active network flows F, a set of telemetry items V, and a set of monitoring applications M. Set D in network G represents programmable forwarding devices $D = \{1, ..., |D|\}$, while set L links interconnecting pair of devices $(d_1, d_2) \in (D \times D)$. Each forwarding device $d \in D$ is able to embed a subset of items $V_d \subseteq V$ into packets of flow $f \in F$. Each telemetry item $v \in V$ has its size defined by function $S : V \rightarrow \mathbb{N}^+$. Network

flows F are used to collect telemetry data from forwarding devices D. A flow $f \in F$ has two endpoints (*i.e.*, source and destination) and is routed through the network infrastructure G using a simple path $\mathcal{P}t$. We denote the path taken by flow f as function $\mathcal{P}t : F \rightarrow \{D_1 \times ... \times D_{|D|}\}$. Network flows F are encapsulated in a forwarding protocol and the amount of available space to embed telemetry items in a packet flow is bounded by a constant $K_f \in \mathbb{N}^+$.

Monitoring requirements. Monitoring application $m \in M$ requires a subset of telemetry items $R_m \subseteq V$ to operate. Telemetry items might have spatial and temporal dependencies. Telemetry items have spatial dependency *iff* they must be collected from the same forwarding device $d \in D$. We represent spatial dependencies by the set of sets $R^s_m \subseteq \mathscr{P}(R_m)$. In turn, telemetry items have temporal dependency *iff* they must be collected within a given deadline. Temporal dependencies are represented by the set of sets $R^t_m \subseteq R^s_m$, and the required deadline is expressed as a function $T^f : R^t_m \rightarrow \mathbb{N}^+$. The model keeps track of the last time unit an item $P \in R^t_m$ was collected by means of a function $H^f : R^t_m \rightarrow \mathbb{N}^+$. When $(\forall P \in R^t_m) : H^f(P) > T^f(P)$, item P is out of date (*i.e.*, the deadline has expired).

Output. The model output is denoted by a 3-tuple $\chi = \{Y, S^b, T^b\}$. Variables from $Y = \{\, y_{d,v,f}\,, \forall\, d \in D, v \in V, f \in F\}$ indicate that a forwarding device d embed telemetry item v into packets of network flow f. Variables from $S^b = \{\, s^b_{m,d,p}\,, \forall m \in M, d \in D, p \in P \in R^s_m\}$ and $T^b = \{\, t^b_{m,p}\,, \forall m \in M, p \in P \in R^t_m\}$ are used to keep track of spatial and temporal dependencies satisfied by the model. Variable set $s^b_{m,d,p}$ indicates that a spatial dependency p from monitoring application m is satisfied in network device d. Conversely, variable set $t^b_{m,p}$ indicates that temporal dependency p from monitoring application m is satisfied. Next, we describe the mixed-integer linear programming formulation for the problem.

$$\textbf{Maximize} \quad \sum_{m \in M} \sum_{p \in P \in R^s_m} \sum_{d \in D} s^b_{m,d,p} + t^b_{m,p} \tag{1}$$

Subject to:

$$\sum_{d \in \mathcal{P}t(f)} \sum_{v \in V_d} y_{d,v,f} \cdot S(v) \leq K_f \qquad \forall f \in F \tag{2}$$

$$\sum_{f \in F} y_{d,v,f} \leq 1 \qquad \forall d \in D, v \in V_d, \tag{3}$$

$$s_{m,d,p} = \sum_{v \in P} \sum_{f \in F} y_{d,v,f} \qquad \forall m \in M, p \in P \in R^s_m, d \in D \tag{4}$$

$$t_{m,p} = \sum_{v \in P} \sum_{d \in D} \sum_{f \in F} y_{d,v,f} \qquad \forall p \in P \in R^t_m : H^f(p) > T^f(p) \tag{5}$$

$$s^b_{m,d,p} \leq \frac{s_{m,d,p}}{|P|} \qquad \forall m \in M, p \in P \in R^s_m, d \in D \tag{6}$$

$$t^b_{m,p} \cdot \leq \frac{t_{m,p}}{|P|} \qquad \forall m \in M, p \in P \in R^t_m \tag{7}$$

$$y_{d,v,f} \in \{0, 1\} \qquad \forall d \in D, v \in V, f \in F \tag{8}$$

$$s_{m,d,p} \in \mathbb{N}^+ \qquad \forall m \in M, d \in D, p \in P \in R^s_m \tag{9}$$

Algorithm 1 GRASP-based heuristic Approach

Input: $G = (D, L)$: network infrastructure
Input: α: greediness of the construction heuristic
1: $\chi^{best} \leftarrow \emptyset$
2: **while** the stopping criteria is not met **do**
3: $\chi \leftarrow$ generete greedy-randomized solution, with parameter α (Algorithm 2)
4: **repeat**
5: $\chi^{'} \leftarrow$ localSearch(χ) (Algorithm 3)
6: Applies Alg. 2 on incumbent solution $\chi^{'}$ to higher satisfied dependencies
7: **if** $cost(\chi^{'}) > cost(\chi^{best})$ **then**
8: $\chi^{best} \leftarrow \chi^{'}$
9: **end if**
10: **until** the solution $\chi^{'}$ is not improved
11: **end while**
12: **return** χ^{best}

$$s^b_{m,d,p} \in \{0,1\} \qquad \forall m \in M, d \in D, p \in P \in R^s_m \quad (10)$$

$$t_{m,p} \in \mathbb{N}^+ \qquad \forall m \in M, p \in P \in R^t_m \quad (11)$$

$$t^b_{m,p} \in \{0,1\} \qquad \forall m \in M, p \in P \in R^t_m \quad (12)$$

Constraint set (2) ensures that a network flow $f \in F$ does not exceed its capacity K_f, while constraint set (3) ensures that a given telemetry item v is collected at most by a single network flow $f \in F$ in a given device d. Constraints sets (4) and (6) account whether or not a spatial dependency is met. For a given a spatial dependency, constraint set (4) count the number of telemetry items collected in a device d, while constraint set (6) verifies whether or not the dependency is met (*i.e.*, checking if all items were collected). Similarly, constraints sets (5) and (7) count the number of temporal dependencies that are satisfied. Note that variable sets $s_{m,d,p}$ and $t_{m,p}$ are auxiliary variables utilized to count the number of telemetry items collected by monitoring dependencies. Last, constraints sets (8)-(12) define the domains of the variables.

4 Proposed Heuristic Approach

To tackle the complexity and come up with high-quality solutions, we introduce a custom-made constructive algorithm and a local-search procedure. Our algorithm is based on Greedy Randomized Adaptive Search Procedure (GRASP) [3], techniques that have been successfully applied to solve hard optimization problems. The core idea consists of iteratively build a solution χ (line 3), followed by a local search procedure (line 5) so as to avoid local maximums (lines 2-11). The algorithm runs until a stopping criterion is met, which can be a global time limit or a number of iteration without improvements. Every time the algorithm gets stuck in some local maximum (line 10), the procedure is restarted – i.e., a new solution is built from scratch. However, we keep track of the best solution found so far (χ^{best}). Algorithm 1 consolidates the proposed approach.

4.1 Greedy-randomized Constructive Heuristic

The proposed heuristic to solve the in-band network telemetry problem utilizes a step-wise construction of a candidate solution using a probabilistic greedy construction procedure. Algorithm 2 is detailed next.

The heuristic starts building a Restricted Candidate List (RCL) $\mathcal{L}$, which consists of possible assignments of spatial/temporal dependencies to potential forwarding devices. For all monitoring dependencies P, i.e. $(m \in M) : P \in R^s_m$, the set of potential forwarding devices D' consists of the subset $D' \subseteq D$ that satisfies the spatial/temporal dependencies P, i.e. $D' = \{d \in D' \mid P \subseteq V_d\}$. Then, the created RCL $\mathcal{L}$ is sorted in ascending order with respect to the sum of telemetry items in P (i.e., $\sum_{v \in P} S(v)$). Lines 2-3 describe this procedure.

The algorithm iterates until there is no spatial/temporal dependency left unchecked (lines 4-20). As the objective function consists of maximizing the amount of spatial/temporal dependencies satisfied, our heuristic tries to satisfy the smallest-size dependencies first. Instead of building a solution greedily, our strategy picks the next spatial/temporal dependence randomly amongst the best $\alpha \in [0, 1]$ candidates in $\mathcal{L}$. For instance, if parameter $\alpha = 0.3$, then the algorithms would restrict the RCL $\mathcal{L}$ to the 30% better candidates. Amongst them, the choice is made randomly. Observe that lower values of α lead the algorithm to behave greedily, while higher values make the procedure to act randomly. This procedure is illustrated in lines 5-6.

After picking the spatial/temporal candidate (P, d), the algorithm iteratively tries to assign a telemetry item $v \in P$ to an available network flow $f \in F$ (lines 9-15). Observe that possible network flows are computed in advance in line 7. A telemetry item $v \in P$ is assigned to network flow f *iff* its available space K_f is greater than the size of item v (i.e., $S(v) \leq K_f$) (line 10). In case a telemetry item $v \in P$ is not assigned to an available network flow, all previous assignment of (P, d) is undone (lines 16-18), and the procedure is restarted. Last, the algorithm returns a feasible solution χ (line 21). The algorithm's complexity is upper-bounded by $O(|M| \cdot |R^s_m| \cdot |D| \cdot |V| \cdot |F|)$. Observe that, in practice, the number of spatial/temporal dependencies R^s_m, and the maximum number of telemetry items V are expected to be bounded by a fixed constant.

4.2 Local Search Heuristic

After building an incumbent solution χ, we applied a local search procedure in order to free up space on utilized network flows $\mathcal{F}$. The idea is to reallocate previous assignments (P, d) in an attempt to eventually allow blocked spatial/temporal dependencies to be satisfied. Algorithm 3 overviews the proposed procedure.

The algorithm scans all network devices $d \in D$, and all pairs of network flows routed through d (i.e. pairs (f_1, f_2) (lines 5-7), and tries to reassign telemetry items of spatial/temporal dependency (P, d) from a given network flow f_1 to network flow f_2 (line 9). A reassignment is only valid *iff* (i) there

Algorithm 2 Constructive Heuristic Procedure

Input: α: greediness of the construction heuristic
1: $\chi \leftarrow \emptyset$
2: $\mathcal{L} \leftarrow \{(P, d) \in \mathcal{L} : P \in R^s_m \wedge P \subseteq V_d\}$, generate restricted candidate list
3: sort $\mathcal{L}$ in ascending order w.r.t to the size of dependency P, i.e. $\sum_{p \in P} S(p)$
4: **while** there is spatial/temporal dependency in $\mathcal{L}$ **do**
5: $(P, d) \leftarrow$ Random($\mathcal{L}$, α)
6: $\mathcal{L} \leftarrow \mathcal{L} \setminus (P, d)$
7: $\mathcal{F} \leftarrow$ possible network flows, i.e. those routed through forwarding device d
8: **for** each telemetry item $v \in P$ **do**
9: **for** every candidate network flow $f \in \mathcal{F}$ **do**
10: **if** $S(v) \leq K_f$ **then**
11: Assign telemetry item v to network flow f in solution χ
12: $K_f \leftarrow K_f - S(v)$
13: break
14: **end if**
15: **end for**
16: **if** telemetry item $v \in P$ has not been assigned **then**
17: Undo previous assignment of spatial/temporal dependency (P, d) in χ
18: **end if**
19: **end for**
20: **end while**
21: **return** χ

exist available space in network flow f_2, i.e. $K_{f_2} - S(v) \geq 0$; and (ii) the reassignment has not been performed previously (line 10). The algorithm keeps track of performed reassignments using the set $\mathcal{N}_{lst}$. Observe that when a reassignment $\mathcal{N}(v, f_1, f_2)$ is made, its counterpart $\mathcal{N}(v, f_2, f_1)$ is also added to $\mathcal{N}_{lst}$ (line 13). This condition avoid the algorithm to cycle amongst reassignments. The local search algorithms adopts the first improvement strategy, i.e. over all valid movements it always performs the first that maximizes the space in a given network flow (lines 18-20). Whenever a reassignment is made, the procedure is restarted.

5 Evaluation

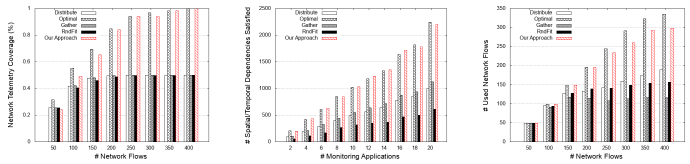
In this section, we evaluate the effectiveness of our heuristic-based approach, comparing it with state-of-the-art competitors. All experiments were performed on a machine with four Intel Xeon E5-2670 processors and 56 GB of RAM, using the Ubuntu GNU/Linux Server 11.10 x86 64 operating system.

5.1 Workload

To evaluate the in-band network telemetry problem, we considered different physical network instances that were generated with Brite [9], following the Barabasi-Albert model [1]. We considered physical network infrastructures ranging from small-scale (50 forwarding devices) to large-scale ones (up to 600 forwarding devices). On top of each infrastructure instance, there is a set F of network flows. We varied the size of $|F|$ from 50 to 600. Each flow $f \in F$ interconnected two random endpoints in G using the shortest path. Similarly to [5], we consider the upper-bound constant K_f varying uniformly

Algorithm 3 Local Search Procedure – Reallocating Telemetry Items

Input: $G = (D, L)$: network infrastructure
Input: χ: incumbent solution
1: $\mathcal{N} \neq \emptyset$ ▷ current move
2: $\mathcal{N}_{lst} \leftarrow \emptyset$ ▷ list of all performed moves
3: **while** solution χ has been improved ($\mathcal{N} \neq \emptyset$) **do**
4: $\quad\mathcal{N} \leftarrow \emptyset$
5: $\quad$**for** each forwarding device $d \in D$ **do**
6: $\quad\quad\mathcal{F} \leftarrow$ possible network flows, those routed through forwarding device d in χ
7: $\quad\quad$**for** each pair of network flows $(f_1, f_2) \in \mathcal{F}$ **do**
8: $\quad\quad\quad$**for** each telemetry item v assigned to network flow f_1 **do**
9: $\quad\quad\quad\quad\mathcal{N} \leftarrow (v, f_1, f_2)$ ▷ Reassign item v from network flow f_1 to f_2
10: $\quad\quad\quad\quad$**if** $\mathcal{N} \notin \mathcal{N}_{lst} \wedge \left(\sum_{d' \in \mathcal{P}t(f)} \sum_{u \in V_{d'}} y_{d',u,f} \cdot S(u)\right) + S(v) \leq K_{f_2}$ **then**
11: $\quad\quad\quad\quad\quad$Perform move $\mathcal{N}$ on solution χ
12: $\quad\quad\quad\quad\quad\mathcal{N}' \leftarrow (v, f_2, f_1)$
13: $\quad\quad\quad\quad\quad\mathcal{N}_{lst} \leftarrow \mathcal{N}_{lst} + \mathcal{N} + \mathcal{N}'$
14: $\quad\quad\quad\quad\quad$Restart local search, go to line 3
15: $\quad\quad\quad\quad$**end if**
16: $\quad\quad\quad$**end for**
17: $\quad\quad$**end for**
18: $\quad$**end for**
19: **end while**
20: **return** χ, φ



(a) Coverage of Telemetry Items.

(b) Number of spatial and temporal dependencies satisfied.

(c) Number of network flows used by solutions.

Fig. 2 In-band telemetry orchestration performance metrics applied to small instances (50 forwarding devices).

from 10-30 Bytes, while forwarding devices have 8 possible telemetry items to export[2], varying $S(v)$ from 2 to 20 Bytes [10]. Further, we consider an increasing number of monitoring applications – ranging from 2 to 20. Each application requires at most 4 spatial dependencies, each having at least 2 items (*i.e.*, switch ID + monitored metric) and at most 4. Our proposed heuristic algorithm was set to consider $\alpha = 0.3$, since it has empirically shown to be the most effective value, and a number of iteration without improvements equal to 10, as a stopping criterion. Each experiment is repeated 30 times to ensure a confidence level of 95% or higher.

Baseline. We compare our proposed heuristic against *(i)* the optimal solution proposed by [5]; *(ii)* against two state-of-the-art orchestration heuristics proposed by [8], namely `Gather` and `Balance`, and *(iii)* to a worst-case heuristic, the Random-Fit heuristic (`RndFit`). The `RndFit` heuristic chooses

[2] In-band Network Telemetry: `https://p4.org/assets/INT-current-spec.pdf`

randomly a telemetry items and tries to assign to random available network flow.

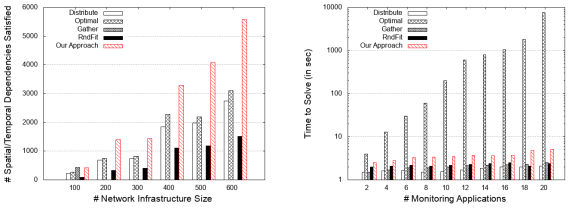
5.2 Results

First, we analyze our proposed approach in comparison to the state-of-the-art solution regarding in-band network telemetry performance metrics considering small-scale network infrastructures. Then, we focus on showing that our solution can timely cope with large-scale network infrastructures.

We start analyzing the network-wide visibility in terms of network telemetry coverage (*i.e.*, the percentage of telemetry items collected). Fig. 2(a) shows the average network telemetry coverage of telemetry items collected for an increasing number of active network flows. We observe that the number of network flows impacts on the network-wide visibility directly (*i.e.*, the more network flows, the higher the network coverage). We also noted that the proposed heuristic on average outperforms state-of-the-art heuristics by up a factor of 2. Observe that the increasing number of network flows does not affect the obtained coverage for state-of-the-art heuristics. That happens because these algorithms make greedy choices, leading to a sub-optimal solution. More importantly, we can observe that our heuristic can cover network telemetry similarly to the optimal solution (on average, the solution is less than 5% from the optimum).

Next, we evaluate the amount of spatial/temporal dependencies that are satisfied, given an increasing set of monitoring application (varying from 2 to 20), depicted in Fig. 2(b). For this experiment, we set the number of flows to 400. We can observe that (i) the number of spatial/temporal dependencies increases as the monitoring applications do and (ii) that our heuristic solution is close to the optimum one. Additionally, our heuristic produces solutions up to 3x better than competitors and at most 7% away from the optimum (regarding the number of spatial/temporal dependencies satisfied). Further, we observe that state-of-the-art solutions struggle to cope with spatial/temporal dependencies, especially when the number of monitoring applications increases. The reason for that behavior is because evaluated competitors make local decisions without considering monitoring application requirements. For instance, the baseline algorithm `Balance` strives to collect telemetry items in a fairly way (e.g., round-robin on the network flows) – which lowers the overall chance to satisfy monitoring requirements. It is worth mentioning that the number of spatial dependencies satisfied is a piece of essential input information to a machine-learning-based monitoring application, which requires coupled telemetry information to be used in the learning/inference process.

In order to show the reasons behind the low performance of evaluated competitors, we illustrate in Fig 2.(c) the amount of network flows used by each solution. In the x-axis, we present the number of available network flows (i.e. from 50 to 400), while in y-axis the number of used ones. Ideally, one solution would use as much network flows as possible since the objective goal is to maximize the number of spatial/temporal dependencies. However, as we



(a) Number of spatial and temporal dependencies satisfied.

(b) Required time to solve the problem (in log scale).

Fig. 3 In-band telemetry orchestration performance metrics applied to large instances.

can observe in the figure, the number of available network flows are usually not used in totality. That happens for three reasons: (i) network flows capacity K_f varies, forbidding eventually to be used by a large-size telemetry item; (ii) network flows are routed through the shortest-path, which might lead to an uneven distribution of network flows per forwarding devices (i.e., highly interconnected forwarding devices would have much more network flows to embed telemetry data); and (iii) the decision made by the algorithms, as illustrated in the figure.

Last, we evaluate our proposed heuristic in large-scale network infrastructures, i.e. ranging the network size from 100 to 600 forwarding devices. We set the number of monitoring application to 20 and the number of flows to 600. Fig. 3(a) illustrates the amount of spatial/temporal dependencies that are satisfied. Observe that for this setting, the optimum solution was only been possible to run for a small network size of 100 (and it demanded more than 2 hours). We observe that, even for large instances, our approach outperforms state-of-the-art approaches, up to 2x better to network size of 600. In order to show that solving the problem using our procedure scales in comparison to the optimal solution, Fig. 3(b) illustrates the amount of time required to solve the problem, given an increasing set of monitoring applications. We consider a network size of 100 in order to be able to compare with the optimal value. Observe that the time required to solve the problem optimally grows exponentially – up to 7500 seconds when considering 20 monitoring applications. However, we observe that the time consumed by our heuristic is up to 2 seconds higher than state-of-the-art approaches, mainly due to the iterated local search procedure.

6 Final Remarks

In this paper, we tackled the scalability limitation of the in-band network telemetry orchestration plan problem. We designed a heuristic approach based on the GRASP procedure, including a custom-made constructive and local-search procedures. We showed that our approach can effectively collect network telemetry data and satisfy spatial/temporal dependencies of moni-

toring applications – up to 2x better than state-of-the-art approaches and up to 5% worse than the optimum solution – while demanding a few seconds to execute. As future work, we intend to evaluate the proposed heuristic on a realistic programmable testbed, integrating it into SDN controllers.

Acknowledgements

This work was partially funded by National Council for Scientific and Technological Development (CNPq 2018/427814), Foundation for Research of the State of Sao Paulo (FAPESP 2018/23092-1), and Foundation for Research of the State of Rio Grande do Sul (19/2551-0001224-1,19/2551-0001266-7).

References

1. Albert, R., Barabási, A.L.: Topology of evolving networks: Local events and universality. Physical Review Letters **85**, 5234 – 5237 (Dec 2000)
2. Cisco: (2018), `https://www.cisco.com/c/en/us/solutions/service-provider/cloud-scale-networking-solutions/model-driven-telemetry.html`
3. Feo, T.A., Resende, M.G.C.: Greedy randomized adaptive search procedures. Journal of Global Optimization **6**(2), 109–133 (Mar 1995)
4. Gupta, A., Harrison, R., Canini, M., Feamster, N., Rexford, J., Willinger, W.: Sonata: Query-driven streaming network telemetry. In: Proceedings of ACM SIGCOMM'18. ACM, New York, NY, USA (2018)
5. Hohemberger, R., Castro, A.G., Vogt, F.G., Mansilha, R.B., Lorenzon, A.F., Rossi, F.D., Luizelli, M.C.: Orchestrating in-band data plane telemetry with machine learning. IEEE Communications Letters **23**(12), 2247–2251 (Dec 2019)
6. Jeyakumar, V., Alizadeh, M., Geng, Y., Kim, C., Mazières, D.: Millions of little minions: Using packets for low latency network programming and visibility. In: Proceedings of the ACM SIGCOMM '14. pp. 3–14. ACM, New York, NY, USA (2014)
7. Liu, Z., Bi, J., Zhou, Y., Wang, Y., Lin, Y.: Netvision: Towards network telemetry as a service. In: 2018 IEEE 26th International Conference on Network Protocols (ICNP). pp. 247–248 (Sep 2018)
8. Marques, J.A., Luizelli, M.C., Da Costa, R.I.T., Gaspary, L.P.: An optimization-based approach for efficient network monitoring using in-band network telemetry. Journal of Internet Services and Applications **10**(1), 16 (Jun 2019)
9. Medina, A., Lakhina, A., Matta, I., Byers, J.: Brite: an approach to universal topology generation. In: Proceedings of the IEEE MASCOTS. pp. 346–353 (Aug 2001)
10. Pan, T., Song, E., Bian, Z., Lin, X., Peng, X., Zhang, J., Huang, T., Liu, B., Liu, Y.: Int-path: Towards optimal path planning for in-band network-wide telemetry. In: IEEE INFOCOM 19. pp. 1–9 (Apr 2019)
11. Putina, A., Rossi, D., Bifet, A., Barth, S., Pletcher, D., Precup, C., Nivaggioli, P.: Telemetry-based stream-learning of bgp anomalies. In: Proceedings of the ACM Workshop Big-DAMA'18. pp. 15–20. ACM, New York, NY, USA (2018)
12. Tammana, P., Agarwal, R., Lee, M.: Simplifying datacenter network debugging with pathdump. In: 12th USENIX OSDI 16). pp. 233–248. Savannah, GA (2016)
13. Tammana, P., Agarwal, R., Lee, M.: Distributed network monitoring and debugging with switchpointer. In: 15th USENIX NSDI 18. pp. 453–456. Renton, WA (2018)
14. Zhu, Y., Kang, N., Cao, J., Greenberg, A., Lu, G., Mahajan, R., Maltz, D., Yuan, L., Zhang, M., Zhao, B.Y., Zheng, H.: Packet-level telemetry in large datacenter networks. In: Proceedings of the ACM SIGCOMM '15. pp. 479–491. ACM, New York, NY, USA (2015)

ÍNDICE