Ponto de Troca de Internet do Futuro (FIXP) - Habilitando Interconexão de Arquiteturas de Internet do Futuro

Juliano Coelho Gonçalves de Melo

Universidade Federal de Uberlândia
Faculdade de Computação
Programa de Pós-Graduação em Ciência da Computação

Uberlândia
2026

Juliano Coelho Gonçalves de Melo

Ponto de Troca de Internet do Futuro (FIXP) - Habilitando Interconexão de Arquiteturas de Internet do Futuro

Tese de doutorado apresentada ao Programa de Pós-graduação da Faculdade de Computação da Universidade Federal de Uberlândia como parte dos requisitos para a obtenção do título de Doutor em Ciência da Computação.

Área de concentração: Ciência da Computação

Orientador: Prof. Flávio de Oliveira Silva, Ph.D.

Uberlândia
2026

Dados Internacionais de Catalogação na Publicação (CIP)
Sistema de Bibliotecas da UFU, MG, Brasil.

M528p
2026

Melo, Juliano Coelho Gonçalves de, 1979-
Ponto de Troca de Internet do Futuro (FIXP) [recurso eletrônico] : habilitando interconexão de arquiteturas de internet do futuro / Juliano Coelho Gonçalves de Melo. - 2026.

Orientador: Flávio de Oliveira Silva.
Tese (Doutorado) - Universidade Federal de Uberlândia, Programa de Pós-graduação em Ciência da Computação.
Modo de acesso: Internet.
Disponível em: http://doi.org/10.14393/ufu.te.2026.5504
Inclui bibliografia.
Inclui ilustrações.

1. Computação. I. Silva, Flávio de Oliveira, 1970-, (Orient.). II. Universidade Federal de Uberlândia. Programa de Pós-graduação em Ciência da Computação. III. Título.

CDU: 681.03

André Carlos Francisco
Bibliotecário-Documentalista - CRB-6/3408

UNIVERSIDADE FEDERAL DE UBERLÂNDIA
Coordenação do Programa de Pós-Graduação em Computação
Av. João Naves de Ávila, 2121, Bloco 1A, Sala 243 - Bairro Santa Mônica, Uberlândia-MG, CEP 38400-902
Telefone: (34) 3239-4470 - www.ppgco.facom.ufu.br - cpgfacom@ufu.br

ATA DE DEFESA - PÓS-GRADUAÇÃO

Programa de Pós-Graduação em:	Ciência da Computação				
Defesa de:	Tese, 01/2026, PPGCO				
Data:	27 de Janeiro de 2026	Hora de início:	09:30	Hora de encerramento:	14:22
Matrícula do Discente:	11913CCP004				
Nome do Discente:	Juliano Coelho Gonçalves de Melo				
Título do Trabalho:	Ponto de Troca de Internet do Futuro (FIXP) - Habilitando Interconexão de Arquiteturas de Internet do Futuro				
Área de concentração:	Ciência da Computação				
Linha de pesquisa:	Sistemas de Computação				
Projeto de Pesquisa de vinculação:	----------				

Reuniu-se por videoconferência, a Banca Examinadora, designada pelo Colegiado do Programa de Pós-graduação em Ciência da Computação, assim composta: Professores Doutores: Rafael Pasquini - FACOM/UFU, Silvio Ereno Quincozes - Unipampa, Daniel Nunes Corujo - Universidade de Aveiro, António Luis Duarte Costa - Universidade do Minho e Flávio de Oliveira Silva - Universidade do Minho, orientador do(a) candidato(a).

Os examinadores participaram desde as seguintes localidades: Flávio de Oliveira Silva - Braga/Portugal, Silvio Ereno Quincozes - Alegrete/RS, Daniel Nunes Corujo - Aveiro/Portugal, António Luis Duarte Costa - Braga/Portugal. O aluno Juliano Coelho Gonçalves de Melo participou de Guajará-Mirim/RO e somente o Rafael Pasquini participou da cidade de Uberlândia.

Iniciando os trabalhos o presidente da mesa, Prof. Dr. Flávio de Oliveira Silva, apresentou a Comissão Examinadora e o(á) candidato(a), agradeceu a presença do público, e concedeu ao(á) Discente a palavra para a exposição do seu trabalho. A duração da apresentação do(a) Discente e o tempo de arguição e resposta foram conforme as normas do Programa.

A seguir o senhor presidente concedeu a palavra, pela ordem sucessivamente, aos examinadores, que passaram a arguir o(á) candidato(a). Ultimada a arguição, que se desenvolveu dentro dos termos regimentais, a Banca, em sessão secreta, atribuiu o resultado final, considerando o candidato(a):

Aprovado

Esta defesa faz parte dos requisitos necessários à obtenção do título de Doutor.

O competente diploma será expedido após cumprimento dos demais requisitos, conforme as normas do Programa, a legislação pertinente e a regulamentação interna da UFU.

Nada mais havendo a tratar foram encerrados os trabalhos. Foi lavrada a presente ata que após lida e achada conforme foi assinada pela Banca Examinadora.

Documento assinado eletronicamente por **Daniel Nunes Corujo**, **Usuário Externo**, em 06/02/2026, às 11:47, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **Flávio de Oliveira Silva**, **Usuário Externo**, em 06/02/2026, às 11:57, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **Rafael Pasquini**, **Professor(a) do Magistério Superior**, em 06/02/2026, às 15:09, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **Silvio Ereno Quincozes**, **Usuário Externo**, em 06/02/2026, às 15:41, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **António Luis Duarte Costa**, **Usuário Externo**, em 07/02/2026, às 16:52, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

A autenticidade deste documento pode ser conferida no site https://www.sei.ufu.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **6975526** e o código CRC **9D8E905B**.

Referência: Processo nº 23117.000774/2026-76 SEI nº 6975526

Dedico este trabalho a todos os meus familiares, principalmente minha mãe Rosângela, vovó Archidamia e vovô Borgico, que são diretamente responsáveis pelo homem que me transformei.

Agradecimentos

Ao longo dessa jornada que se chama vida, nascemos parte de uma família, encontramos amigos que seguem conosco até o fim do percurso e construímos, no matrimônio, um suporte que nos abastece de sorrisos, lágrimas, instantes eternos e aprendizado. Aprendemos a amar porque, ao longo dessa estrada, encontramos pessoas que nos fazem querer viver para enxergar o sol do outro dia, que nos emocionam com suas estórias, que, no meio da escuridão, conseguem acender a luz nos corações só com um simples sorriso ou gesto de generosidade. Aprendemos a amar porque encontramos pessoas que conhecem esse sentimento, algo muito raro nos dias atuais. São essas, sim, pessoas extraordinárias. Por isso, são dignas não só de agradecimentos, mas também de gratidão.

Agradeço aos meus familiares, principalmente à minha mãe, Rosângela, e aos meus avós, Archidamia e Borgico, pelo suporte e pelo amor incondicional. Aprendemos no seio familiar qual o maior tesouro que existe: é o sentimento que acalanta a alma, a mente, que traz paz e felicidade; não a felicidade ideal que todos imaginam existir, mas instantes alegres que valem a pena todo o esforço, todo o suor que exalamos para sobreviver. Eu acredito nessa alegria momentânea, mas também duradoura; acredito que seja esse o único sentimento compatível com a dignidade humana.

Agradeço à minha esposa, Elaine, e à filha, Vitória, pelo amor sem fim e pela compreensão de todas as minhas ausências decorrentes do tempo dedicado a este trabalho. Acompanharei vocês duas, de mãos dadas, até onde a vista não consegue alcançar, além, além, além, e... mais além.

Agradeço aos colegas do grupo MEHAR pela troca de experiências e pelo aprendizado; um agradecimento especial ao meu amigo Pedro Frosi Rosa, cuja paciência nas discussões sem fim é infinita; obrigado pelos ensinamentos técnicos e pelos conselhos sábios que me deu ao longo dessa labuta.

Não poderia deixar de fora um grande colega de profissão, Natal Vieira de Souza Neto. Sua generosidade profissional é ímpar e devo deixar registrado que aprendi muito e tenho certeza de que continuarei aprendendo com você.

Agradeço aos docentes e técnicos da Faculdade de Computação pelo ensinamento e

pelo suporte prestados nos últimos anos. É uma honra dizer que, após esses anos, faço parte de uma comunidade tão importante.

Deixo um forte abraço para o Erisvaldo, ex-integrante da secretaria de pós-graduação da Faculdade de Computação; sempre se mostrou gentil, competente e pronto para resolver quaisquer problemas; e foram muitos. É um grande amigo que fiz e que levarei por toda a vida. Um grande abraço também à sua substituta, Carol. Apesar de não nos conhecermos pessoalmente, obrigado pela preocupação e pelas notificações, para que não perdêssemos nenhum prazo estabelecido pelas normas e pelo estatuto da instituição.

Gostaria de deixar um agradecimento especial ao meu amigo e orientador Flávio de Oliveira Silva, cuja perseverança e espírito de trabalho nos influenciam a sermos profissionais cada vez melhores. Em qualquer instituição de ensino, esses valores são fundamentais. No decorrer desse trabalho, passei por momentos difíceis e você sempre me ajudou a sair deles, com generosidade, paciência e estratégia. Levo você como amigo, daqueles com quem, como dizia minha vó, a gente conta nos dedos da mão. Obrigado por tudo, meu amigo! Espero que no fim dessa jornada, ainda possamos aprender um com o outro e trabalhar juntos em novos desafios de pesquisa.

Agradeço à Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) pelo apoio financeiro e a todos que acreditaram e apoiaram, de alguma forma, este trabalho.

Para finalizar, mando um forte abraço e agradecimentos especiais aos membros da banca, que dedicaram tempo para ler e avaliar esta tese de doutorado.

Pensar é o trabalho mais difícil que existe. Talvez por isso tão poucos se dediquem a ele.
(Henry Ford)

A imaginação é mais importante que a ciência, porque a ciência é limitada, ao passo que a imaginação abrange o mundo inteiro
(Albert Einstein)

Resumo

A Internet foi projetada para atender a necessidades diferentes das atuais. As novas demandas de aplicações atuais às quais a rede é submetida não são adequadamente atendidas devido a limitações conhecidas. Uma das dificuldades da Internet é a falta de flexibilidade para implementar novos serviços na camada de rede, o que limita sua evolução. Esse cenário motivou o desenvolvimento de diversas arquiteturas de Internet do Futuro (*Future Internet Architectures (FIAs)*). Trata-se de arquiteturas disjuntas que, por meio de seus designs, propõem mecanismos que avançam diversos aspectos da comunicação, tais como flexibilidade na camada de rede, mobilidade, multicasting, qualidade de serviço (*Quality of Service (QoS)*), conectividade, qualidade de experiência (*Quality of Experience (QoE)*), segurança, entre outros. Apesar da demanda, a falta de implementação dessas novas arquiteturas em redes de alto desempenho dificulta avaliar a maturidade de suas características de comunicação em cenários e aplicativos específicos. Por outro lado, devido a onipresença da Internet tradicional, uma substituição da arquitetura TCP/IP é impraticável. Uma rede única de comunicação ou um ponto de troca lógico que integra as FIAs e a arquitetura legada TCP/IP, sem a perda de seus princípios arquitetônicos (recursos e mecanismos), é uma ideia promissora para atender às demandas das aplicações atuais e futuras. Este trabalho apresenta o Ponto de Troca de Internet do Futuro (*Future Internet eXchange Point (FIXP)*) e coloca-o como um candidato dessa rede única ou ponto de troca lógico que permite a coexistência de diversas arquiteturas de Internet; também realiza análises qualitativa e quantitativa do FIXP: capacidade de evolução e incorporação de novas arquiteturas, viabilidade de implantação do FIXP na infraestrutura de rede atual e desempenho da comunicação de aplicações das arquiteturas IP e ETArch na infraestrutura FIXP, levando em conta a qualidade de experiência do usuário. Dessa forma, demonstra-se, por meio de casos de uso selecionados, a coexistência eficiente dessas duas arquiteturas em um único substrato físico de comunicação. Além disso, o trabalho apresenta os detalhes de integração das arquiteturas ETArch e IP no FIXP, visando facilitar, futuramente, a incorporação de outras arquiteturas de Internet nessa infraestrutura. Em relação aos trabalhos relacionados, o acervo coletado possibilita

a comparação entre esses trabalhos e o FIXP, posicionando-o no estado da arte como a única infraestrutura que não sobrecarrega os dispositivos de comutação por meio de processos de interoperabilidade e/ou de desenvolvimento de pilhas de protocolos. Além disso, seus princípios arquitetônicos também diferenciam-se desses trabalhos correlatos, e por conseguinte, o nível de dificuldade da integração de novas FIAs no FIXP é baixo em relação a eles. Esses dois fatos, ausência de processos adicionais de sobrecarregamento e nível de dificuldade baixo/médio de integração de novas FIAS, facilitam a implantação do FIXP nas redes atuais. O trabalho também apresenta a visão que o FIXP possui em relação a aplicativos multiarquitetura, aqueles que possuem diversos serviços, sendo que cada um deles utiliza uma arquitetura de Internet distinta. Os casos de uso que demonstram o suporte do FIXP a esses aplicativos simulam a execução de diferentes microsserviços, que atuam em uma mesma aplicação e/ou máquina virtual, fato esse que é bastante frequente nos dias atuais. A diferença é que os microsserviços de uma aplicação multiarquitetura podem utilizar *frameworks* de diferentes FIAs.

Palavras-chave: *Arquitetura de Internet do Futuro*, *ETArch*, *Ponto de Troca de Internet do Futuro*, *FIXP*, *IXP*, *Ponto de Troca de Internet*, *NovaGenesis*.

Abstract

The Internet was designed to meet needs different from those of today. Current applications impose new demands on the network that it cannot meet, given its limitations. One of the difficulties of the Internet is the lack of flexibility to deploy new network-layer services, which limits its evolution. This scenario motivated the development of several Future Internet Architectures (*FIAs*). These are disjoint architectures that propose, through their designs, to offer mechanisms that advance various aspects of communication, such as flexibility in the network layer, mobility, multicasting, quality of service (*QoS*), connectivity, quality of experience (*QoE*), security, among others. Despite the demand, the lack of implementation of these new architectures in high-performance networks makes it difficult to assess the maturity of their communication characteristics for specific scenarios and applications. On the other hand, due to the ubiquity of the traditional Internet, replacing the TCP/IP architecture is impractical. A single communication network or logical exchange point that integrates FIAs and the TCP/IP architecture without compromising its architectural principles (resources and mechanisms) is a promising approach to meeting the demands of current and future applications. This work presents the Future Internet Exchange Point (*FIXP*) and positions it as a candidate for this single network or logical exchange point that allows the coexistence of different Internet architectures; it also provides qualitative and quantitative analysis of FIXP: its capacity for evolution and incorporation of new architectures, the feasibility of implementing FIXP in the current networks, and the performance of application communication of IP and ETArch architectures in the FIXP infrastructure, taking into account the quality of user experience. Evaluations demonstrate the FIXP architecture conceptually by executing selected use cases in which applications from different FIAs coexist in the same communication environment. In addition, the work presents the details of the integration of the ETArch and IP architectures in the FIXP network, aiming to facilitate, in the future, the incorporation of other Internet architectures in this infrastructure. Regarding related work, the collection allows comparison with these works and with FIXP, positioning it as the state of the art, as the only infrastructure that does not add overhead to switching devices

through interoperability processes and/or protocol stack development. Furthermore, its architectural principles differ from those of related work; therefore, the difficulty of integrating new FIAs into FIXP is low to medium relative to that of related work. These two facts, the lack of data plan overhead and the low or medium difficulty level of integrating new FIAs, facilitate the implementation of FIXP in current networks. The work also presents FIXP's view of multi-architecture applications. These applications use different Internet architectures for their services. The use cases that demonstrate FIXP's support for these applications simulate the execution of microservices within the same application or virtual machine, which is common nowadays. The difference is that microservices in a multi-architecture application can use *frameworks* from different *FIAs*.

Keywords: *Future Internet Architecture*, *ETArch*, *Future Internet eXchange Point*, *FIXP*, *IXP*, Internet eXchange Point, *NovaGenesis*.

Lista de ilustrações

Figura 1 – Cenário Motivacional. Coexistência de FIAs heterogêneas. 36
Figura 2 – Organização geral da tese de doutorado. 43
Figura 3 – Mapeamento da localização dos objetivos específicos e das respostas às questões de pesquisa na estrutura da tese. 45
Figura 4 – Arquitetura de uma Rede Definida por Software. Fonte: (UFRJ, 2015). 49
Figura 5 – Arquitetura SDN-OpenFlow. Adaptado de (MCKEOWN et al., 2008). 51
Figura 6 – Modelo Sintético de fluxo do plano de dados programável P4. Fonte: (KAUR; KUMAR; AGGARWAL, 2021). 53
Figura 7 – Implementação das interfaces de um programa P4. Fonte: (KAUR; KUMAR; AGGARWAL, 2021). 54
Figura 8 – Evolução da arquitetura de rede tradicional para o modelo *Software-Defined Networking (SDN)* com plano de dados programável. Fonte: (KAUR; KUMAR; AGGARWAL, 2021). 57
Figura 9 – Camadas da *ETArch*. Fonte: (SILVA; KOFUJI; ROSA, 2013). 61
Figura 10 – Representação do *Domain Title Service (DTS)* 62
Figura 11 – Um exemplo de topologia *ETArch*. 64
Figura 12 – Modelo de camadas de NovaGenesis com seus principais componentes e processos. Fonte: (TELECOMUNICAçõES, 2021b). 67
Figura 13 – Exemplo de encapsulamento de uma mensagem do NovaGenesis. Fonte: (TELECOMUNICAçõES, 2021b). 68
Figura 14 – Executando *3GPP Common API Framework (CAPIF)* na arquitetura Camara. Fonte: (ORDONEZ-LUCENA; DSOUZA, 2022). 81
Figura 15 – Visão geral da arquitetura FIXP. Ilustração da comunicação interna entre os componentes funcionais. Adaptado de (GAVAZZA et al., 2020; SILVA et al., 2022) 96
Figura 16 – Programando o plano de dados na rede FIXP. Adaptado de (LANGUAGE, 2023) 100
Figura 17 – Implantação da rede FIXP nas redes atuais. 103

Figura 18 – Executando aplicativos ETArch na rede *FIXP*. 108
Figura 19 – Executando aplicativos IP na rede *FIXP*. 111
Figura 20 – Fluxograma do FIXP e o controlador NGCA. Fonte: (SILVA et al., 2022). 114
Figura 21 – Cenário de avaliação. 136

Lista de tabelas

Tabela 1 – *Application Programming Interface (API)* do protocolo *Entity Title Control Protocol (ETCP).* 66
Tabela 2 – Coleção de trabalhos relacionados. Características encontradas nos *papers*. 85
Tabela 3 – Formato de primitivas do protocolo FIXP. Apresentando alguns exemplos. 97
Tabela 4 – Características das FIAs implantadas no FIXP atualmente. 106
Tabela 5 – Tabela comparativa que exibe os mecanismos utilizados por cada meta-arquitetura que suporta diversas FIAs 121
Tabela 6 – Tabela comparativa das características das FIAs implantadas em cada meta-arquitetura. 122
Tabela 7 – Lista de casos de uso proposta nesta tese. 131
Tabela 8 – Lista de métricas propostas para a avaliação qualitativa. 133
Tabela 9 – Ferramentas tecnológicas utilizadas em cada demanda da análise qualitativa. 137
Tabela 10 – Medições realizadas nas aplicações "*Chat IP*" e "*Chat ETArch*", por caso de uso, na rede FIXP. 138
Tabela 11 – Medições realizadas nas aplicações "*Streaming de Vídeo On Demand IP*" e "*Streaming de Vídeo On Demand ETArch*", por caso de uso, na rede FIXP. 142
Tabela 12 – Medições realizadas nos serviços de *"streaming de vídeo on demand"* e "*Chat*" executados por uma aplicação multiarquitetura que utiliza frameworks da arquitetura ETArch e IP. 145

Lista de siglas

ACLs Access Control Lists

ANTS Active Networks

API Application Programming Interface

APIs Application Programming Interfaces

AS Autonomous System

ASIC Application Specific Integrated Circuit

CAPEX Capital expenditure

CAPIF 3GPP Common API Framework

CDC Camada de Convergência

CNG Camada Nova Genesis

DAF Distributed Application Facility

DDoS Distributed Denial-Of-Service

DHID Destination Host Identifier

DHT Distributed Hash Table

DHTs Distributed Hash Tables

DIF Distributed IPC Facility

DIFs Distributed IPC Facilites

DTS Domain Title Service

DTSA Domain Title Service Agent

DTSAs Domain Title Service Agents

MDTSA Domain Title Service Agent

DTSCP Domain Title Service Control Protocol

ENDN Enhanced Named Data Networking

ETCP Entity Title Control Protocol

FAL FIXP Abstraction Layer

FCL FIXP Control Layer

FFM FIXP FlowMod

FIA Future Internet Architecture

FIAs Future Internet Architectures

FICs Future Internet Controllers

FIFu Future Internet Fusion

FIXP Future Internet eXchange Point

FRHS FIXP Rule Handle Service

FIXPs Future Internet eXchange Points

FPGA Field-programmable Gate Array

FPGAs Field-programmable Gate Arrays

FPI FIXP Packet-in

FPL FIXP Physcical Layer

FPO FIXP Packet-out

5GS 5G system

GSMA Global System for Mobile communication Association

GW Gateway

GWs Gateways

GIRS Generic Indirection Resolution System

GNRS Global Name Resolution Service

GUID Globally Unique Identifier

GUIDs Globally Unique Identifiers

HT Hash Table

HTS Hash Table Service

IaaS Infrastructure as a Service

IEEE Institute of Electrical and Electronic Engineers

IETF Internet Engineering Task Force

ICN Information-Centric Networking

IP-NaaS IP network as a Service

IPC Inter-Process Communication

IPCs Inter-Process Communications

ISPs Internet Service Providers

IS-IS Intermediate-System-to-Intermediate-System

IXP Internet eXchange Point

IXPs Internet eXchange Points

Kib kibibit

KiB kibibyte

KPIs key performance indicator

LLDP Link Layer Discovery Protocol

MAC Media Access Control

MF MobilityFirst

MiB Mebibyte

MPEG-4 Moving Pictures Expert Group 4

MPLS Multi Protocol Label Switching

ms microssegundos

MTU Maximum Transmission Unit

NaaS Network as a service

NB Name Binding

NBs Name Bindings

NDN Named Data Networking

NEs Network Elemens

NFV Network Functions Virtualization

NG NovaGenesis

NGMN Next Generation Mobile Network

NGCA NovaGenesis Control Agent

NIC Network Interface Controller

NLNs Natural Language Namings

NLSR Named-data Link State Routing

NPU Neural Processing Unit

NPUs Neural Processing Units

NRNCS Name Resolution and Networking Cache System

NTP Network Time Protocol

ONF Open Networking Foundation

OPEX Operating expense

OSPF Open-Shortest Path First

OXM Openflow eXtended Match

P4 Programming Protocol-independent Packet Processors

PG Proxy/Gateway

PGCS Proxy/Gateway Controller Service

PISA Protocol Independent Switch Architecture

PoP Point of Presence

PoPs Points of Presence

PSS Publish/Subscribe Service

PURSUIT Publish-Subscribe Internet Technologies

QoE Quality of Experience

QoS Quality of Service

RBAC Role Based Access Control

RINA Recursive Inter-Network Architecture

RIP Routing Information Protocol

RTEU Response Time Expected by User

SAL Service Access Layer

SCION Scalability, Control, and Isolation On Next-Generation Networks

SCN Service-Centric Networking

SDN Software-Defined Networking

SDX Software Defined Internet eXchange

SID Service Identifier

SOA Service-Oriented Architecture

SHM Shared Memory

SLA Service Level Agreement

SLAs Service Level Agreements

SON Self-organising Networks

SPs Service Providers

SVN Self-Verifying Name

SVNs Self-Verifying Names

TCAM Ternary Content Addressable Memory

TIC Tecnologia de Informação e Comunicação

TTPs Table Type Patterns

UPF User Plane Function

URLLC ultra-reliable and low latency communications

UTC Coordinated Universal Time

VIM Virtualized Infrastructure Managers

VLAN Virtual Local Area Network

VLANs Virtual Local Area Networks

VNFs Virtualized Network Functions

VPLS Virtual Private LAN Service

VPN Virtual private network

VXLAN Virtual Extensible LAN

XIA eXpressive Internet Architecture

XIP eXpressive Internet Protocol

Sumário

1 INTRODUÇÃO 33
1.1 Motivação 35
1.2 Justificativa 36
1.3 Objetivo geral 37
1.4 Objetivo específico 37
1.5 Hipótese 38
1.6 Desafios e questões de pesquisa 38
1.7 Contribuições 39
1.8 Organização da tese 42
1.8.1 Organização Geral 42
1.8.2 Organização relacionada aos objetivos específicos e às questões de pesquisa 46

2 FUNDAMENTAÇÃO TEÓRICA E TRABALHOS RELACIONADOS 47
2.1 Fundamentação teórica 47
2.1.1 Um breve história de Redes Definidas por Software 49
2.1.2 *OpenFlow* 51
2.1.3 *Linguagem Programming Protocol-independent Packet Processors (P4)* . 52
2.1.4 Diferenças entre OpenFlow e *P4* 55
2.1.5 Direções futuras de redes SDN 58
2.1.6 Visão geral da arquitetura *ETArch* e seus conceitos principais 60
2.1.7 Visão geral da arquitetura NovaGenesis e seus conceitos principais . . . 66
2.1.8 Uma breve descrição de outras arquiteturas de Internet do Futuro . . . 73
2.1.9 Uma breve reflexão sobre as redes do futuro. 77
2.2 Trabalhos Relacionados 84
2.2.1 Trabalhos relacionados às redes ativas 86
2.2.2 Trabalhos relacionados ao paradigma SDN 86
2.2.3 Trabalhos relacionados à implementação de pilhas de protocolos 87

2.2.4 Trabalhos relacionados à interoperabilidade 88
2.2.5 Trabalhos relacionados que possuem abordagens híbridas 88
2.2.6 Uma breve discussão dos trabalhos relacionados 89

3 FIXP: HABILITANDO A COEXISTÊNCIA DE ARQUITETURAS DE INTERNET DISTINTAS 93
3.1 Definição conceitual do FIXP 94
3.2 Visão geral da arquitetura FIXP 95
3.3 Comunicação interna dos componentes funcionais da arquitetura FIXP 96
3.4 Serviços e formatos das primitivas do protocolo FIXP 99
3.5 *Switch FIXP* 100
3.6 Implantação do FIXP nas redes atuais 103
3.7 Integrando arquiteturas de Internet distintas à rede *FIXP* . . . 105
3.7.1 Breve descrição das arquiteturas selecionadas 105
3.7.2 Integração da ETArch no *FIXP* 108
3.7.3 Integração da arquitetura IP no *FIXP* 110
3.7.4 Integração da arquitetura *NovaGenesis* no FIXP 112
3.8 Uma breve discussão sobre o FIXP e seus objetivos arquitetônicos 115

4 AVALIAÇÕES E ANÁLISE DOS RESULTADOS OBTIDOS . 117
4.1 Método para avaliação 117
4.2 Avaliações 120
4.2.1 Avaliação das características do FIXP em relação aos trabalhos relacionados 121
4.2.2 Avaliação da arquitetura FIXP 125
4.2.3 Avaliação da implementação e implantação do FIXP nas redes atuais . . 126
4.2.4 Avaliação de performance 129
4.3 Resposta das questões de pesquisa 146
4.4 Considerações finais 151
4.4.1 Uma breve discussão que envolve as *FIAs*, redes corporativas e o *FIXP*. 152
4.4.2 Uma breve discussão sobre o FIXP e as infraestruturas de redes do futuro 153

5 CONCLUSÃO 157
5.1 Considerações finais 157
5.2 Contribuições em produções bibliográficas 160
5.3 Trabalhos Futuros 162

REFERÊNCIAS 167

APÊNDICES **179**

APÊNDICE A – APÊNDICE A 181
A.1 Sobre a programabilidade do plano de dados do *switch* FIXP . 181
A.1.1 Considerações iniciais 181
A.1.2 Declaração de dados 182
A.1.3 Estágio de processamento Parser 183
A.1.4 Estágio de processamento Match-Action 184
A.1.5 Estágio de processamento Deparser 189
A.1.6 Considerações finais 190

Capítulo 1

Introdução

Os alicerces conceituais da rede mundial de computadores, hoje representada pela arquitetura TCP/IP, foram concebidos na década de 70 (BARAN, 1964). Desde então, a Internet passou por uma evolução inimaginável. O aumento do processamento de hardware previsto pela Lei de Moore (LEINER et al., 2009) ampliou a largura de banda e a vazão de enlaces a baixo custo. A malha de elementos de rede se estendeu e o número de hosts aumentou drasticamente. Novas tecnologias de transmissão, tanto wired quanto wireless, foram criadas; bem como foram introduzidos muitos protocolos na camada física.

Esse contexto foi de extrema importância para o desenvolvimento de novas aplicações e serviços, bem como para o surgimento de diversos dispositivos, tais como smartphones, phablets, chips M2M, tablets, PCS, Smart TVs, entre outros. É bom salientar que o aparecimento de novas aplicações, juntamente com essa evolução tecnológica, traz basicamente duas implicações: diversificação das tecnologias de transmissão: GSM, 3G, 4G, 5G, WiMAX, Wi-Fi, Ethernet, LORA, Sigfox, NB-IoT, LTE-M, etc.; e modificação dos padrões de tráfego das redes de computadores.

As últimas previsões da CISCO corroboram esses fatos. Segundo o último relatório anual da Internet (2018-2023) (CISCO, 2020), a previsão é de que o número de dispositivos conectados à rede IP ultrapasse 29,3 bilhões de unidades até 2023. Esse número representa mais do que três vezes a população mundial e um aumento de 59,23% em relação a 2018. Estima-se que esses dispositivos realizem 14,7 bilhões de conexões M2M e 5,7 bilhões de conexões móveis. Grande parte desses dispositivos advém da Internet das Coisas (IoT), que se tornou o sistema predominante (CISCO, 2020). Isso deve-se ao aumento das implantações de aplicativos, tais como telemedicina e sistemas de navegação de carros inteligentes, que exigem maior largura de banda e menor latência (CISCO, 2020). Outros aplicativos IoT que merecem destaque são os domésticos, que oferecem serviços de automação e segurança residenciais, vigilância por vídeo e rastreamento. Outro ponto a ser considerado é que, em 2017, o tráfego de vídeo IP correspondia a cerca de 82% do tráfego na rede de computadores, com a expectativa de atingir 84% em 2022 (CISCO, 2018). Toda essa movimentação na mudança de fluxo e de conexões provocou um aumento

na análise de segurança. Estima-se que os ataques DDoS tenham dobrado de 7,9 milhões em 2018 para 15,4 milhões em 2023 (CISCO, 2020).

A tendência é que os aplicativos da próxima geração tenham requisitos cada vez mais complexos, e essa será a norma dos próximos anos (CISCO, 2020). São aplicativos que utilizam inteligência artificial, Deep Learning, análises preditivas e descobrem insights através de históricos massivos de dados e preveem eventos futuros.

A dificuldade da evolução da Internet perante o surgimento dessas novas demandas é bastante conhecida e passa por dois pontos cruciais (PAPASTERGIOU et al., 2017) (TURNER; TAYLOR, 2005): a camada de rede é engessada, o que impossibilita a inclusão de novos serviços; e o custo elevado de implantá-los na infraestrutura física da rede. No primeiro caso, pode-se citar, como exemplo, o RTP (SCHULZRINNE et al., 2003), protocolo de aplicação que desempenha funções de transporte fim a fim para aplicações que transmitem dados em tempo real, como áudio ou vídeo. Isto aumenta a complexidade da arquitetura e gera overhead de comunicação decorrente de funções sobrepostas. No segundo caso, pode-se citar o IPV6, proposto em 1995, e ainda não completamente implantado (PAPASTERGIOU et al., 2017) (TURNER; TAYLOR, 2005).

Esse cenário chamou a atenção da comunidade científica e várias iniciativas tais como NewArch (SCIENCE; INSTITUTE; DIVISION, 2000), Future Internet Design (FIND) (ROBERTS, 2009), Future Internet Architecture (FIA) (FOUNDATION, 2010), Future Internet Architecture Next Phase (FIA-NP) (FOUNDATION, 2013); e, os programas-quadro (framework programs) FP6 (COMMISSION, 2002), FP7 (COMMISSION, 2007) e H2020 (COMMISSION, 2014) da Next Generation Internet (NGI) indicam que este assunto está em voga desde a década de 90 nos EUA e na Europa. São projetos que incentivam a criação das chamadas FIAs, que têm como foco lidar com as limitações da arquitetura TCP/IP.

Alguns exemplos de FIAs clean-slate (ROBERTS, 2009) projetadas neste período são ETArch (GONÇALVES et al., 2014) (SILVA et al., 2014), NovaGenesis (NG) (ALBERTI et al., 2019), *Recursive Inter-Network Architecture (RINA)*(VRIJDERS et al., 2014), MobilityFirst (MF) (VENKATARAMANI et al., 2014), XIA (HAN et al., 2012) e NDN (ZHANG et al., 2014); cada qual com suas características peculiares, difere em vários aspectos.

Cada arquitetura FIA foi desenvolvida de forma diferente e atende a demandas distintas. Cada uma possui suas orientações de design, paradigmas de comunicação, endereçamento de rede, identificadores e localizadores das entidades comunicantes, processo de resolução de nomes, entre outras decisões de projeto. Em suma, cada uma pode propor alterações relevantes que vão desde a camada de rede até a camada de aplicação.

1.1 Motivação

Nesse cenário, ainda não há como apontar uma arquitetura que atenda a todas as demandas da Internet. O ideal é que essa pluralidade de FIAs coexista em um substrato físico compartilhado (TURNER; TAYLOR, 2005), no qual cada uma dessas FIAs possa atender a demandas pontuais de comunicação. Dessa forma, essas arquiteturas podem ser progressivamente avaliadas, e se for o caso, podem ser gradualmente integradas nas redes de telecomunicações. É possível que uma das FIAs possa emergir como a principal arquitetura da Internet, embora também a coexistência diversificada possa ser a melhor abordagem (FISHER, 2014); pois possibilita a criação de uma multiplicidade de aplicativos de FIAs diferentes, onde cada FIA pode atender as demandas de rede existentes e/ou melhorar as métricas dos requisitos atendidos atualmente.

Este trabalho pressupõe que a busca por uma arquitetura de Internet única, capaz de oferecer a capacidade de rede necessária para atender a todos os requisitos de comunicação, é uma corrida viável, embora menos provável. O cenário atual mostra que as arquiteturas são disjuntas, possuem projetos com objetivos específicos e em diferentes estágios de implementação. Portanto, esta tese considera que a melhor abordagem é criar as condições para que aplicações que utilizam recursos de diferentes arquiteturas de Internet possam coexistir em um mesmo ambiente de comunicação.

No entanto, a rigidez das redes atuais em admitir novos serviços motiva, há décadas, pesquisadores a discutirem a noção de uma arquitetura de metarede, que evolui para acomodar casos de uso não previstos. Um exemplo é o Softnet (ZANDER; FORCHHEIMER, 1983), desenvolvido na década de 80. Trata-se de uma rede programável que utiliza métodos próximos aos de uma rede ativa ANTS (WETHERALL, 1998) para controlar o comportamento dos nós da rede.

Fundamentalmente, essas meta-arquiteturas oferecem interfaces que possibilitam a programabilidade do comportamento dos dispositivos de rede, permitindo, assim, a incorporação de novos serviços no nível de rede; e até, em alguns casos, diversidade arquitetônica.

Grande parte desses trabalhos foram possíveis devido ao auxílio de alguns habilitadores tecnológicos. *SDN* (KAUR; KUMAR; AGGARWAL, 2021) oferece flexibilidade, programabilidade, automação de rede, padronização na manutenção de dispositivos distintos, entre outros benefícios. A linguagem *P4* é uma instanciação de SDN que eleva ainda mais a programabilidade em relação a protocolos anteriores, como o OpenFlow (MCKEOWN et al., 2008), pois permite a programação do plano de dados, facilitando, assim, a incorporação de novos serviços na rede. Outros mecanismos de rede também são utilizados, como interoperabilidade, virtualização de rede, redes sobrepostas, tunelamento, redes ativas, entre outros. A tabela 2 do Capítulo 2 apresenta um resumo dos mecanismos utilizados e as respectivas referências de cada trabalho.

Os trabalhos relacionados descritos no capítulo 2 demonstram que o desenvolvimento

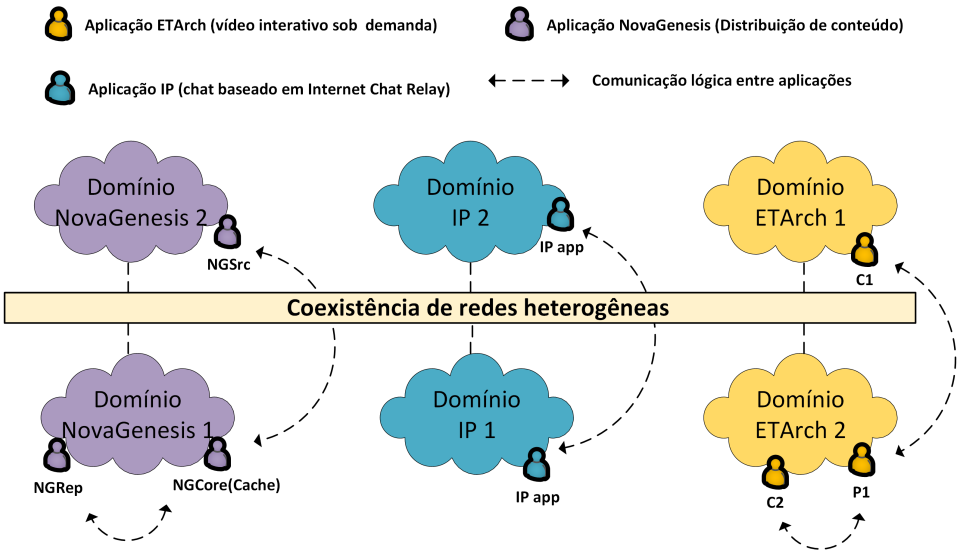

Figura 1 – Cenário Motivacional. Coexistência de FIAs heterogêneas.

de meta-arquiteturas é um tema que movimenta e impulsiona discussões sobre as características das redes do futuro diante das novas demandas das aplicações atuais. No fundo, esses trabalhos discutem como uma rede pode suportar serviços com diferentes princípios arquitetônicos, que lhes conferem capacidades de rede distintas. Este assunto também motiva este trabalho.

A Figura 1 apresenta o cenário motivacional. Trata-se de uma rede heterogênea, em que vários domínios de FIAs distintos comunicam-se entre si da seguinte forma: aplicações ETArch comunicam-se com aplicações ETArch, aplicações IP comunicam-se com aplicações IP, aplicações NovaGenesis comunicam-se com aplicações NovaGenesis, e assim por diante.

A solução que esta tese apresenta permite a construção desse cenário, que é bastante complexo nos dias atuais. Uma arquitetura SDN orientada a workspace (ETArch), uma arquitetura orientada a serviços com princípios de ICN (NovaGenesis) e uma arquitetura orientada a host (IP) coexistem em uma rede única que possibilita a comunicação entre seus aplicativos. Dessa forma, aplicações atuais podem escolher a arquitetura que melhor atenda aos requisitos para uma boa qualidade de comunicação.

1.2 Justificativa

De maneira geral, as soluções encontradas no estado da arte possuem alguns problemas: sobrecarregamento dos equipamentos de rede por motivos variados; alto nível de dificuldade e ineficiência (possível perda de funcionalidades) no processo de integração de novas FIAs à meta-arquitetura; dificuldade de implantação das meta-arquiteturas em ambientes corporativos; alguns trabalhos analisam as possibilidades de uma rede programável, padronizam a comunicação para entidades distintas, mas não atingem a complexidade

da coexistência de diversas FIAs.

Por isso,o princípio arquitetônico da solução que este trabalho propõe difere de todos os trabalhos relacionados e, assim, utiliza mecanismos operacionais distintos que possibilitam um ambiente de coexistência que não apresenta os problemas relatados.

1.3 Objetivo geral

O objetivo geral deste trabalho é propor, desenvolver, analisar e avaliar um conjunto de princípios arquitetônicos que definem uma infraestrutura de comunicação única capaz de suportar a coexistência de arquiteturas de Internet distintas, possibilitando a comunicação entre suas aplicações de forma transparente; de tal forma que esses princípios proveem flexibilidade na inclusão de novos serviços na camada de rede, mitigam o nível de dificuldade de inclusão de novas FIAs na infraestrutura proposta e não adicionam processos que oneram os dispositivos de comunicação tais como interoperabilidade e acúmulo de pilhas de protocolos.

A proposta avança o estado da arte nos planos conceitual e funcional.

No plano conceitual, alguns trabalhos investigam a programabilidade dos planos de dados e o controle das metaredes (ZANDER; FORCHHEIMER, 1983; WETHERALL, 1998; GUO et al., 2021; SIGNORELLO et al., 2016; KARRAKCHOU; SAMAAN; KARMOUCH, 2020; GIMENEZ; GRASA; BUNCH, 2020), mas não propõem um modelo conceitual que suporta a interconexão entre FIAs distintas. Outros trabalhos possuem e atingem os mesmos objetivos arquitetônicos (MACHADO; DOUCETTE; BYERS, 2015; GUIMARãES et al., 2019) , porém, possuem conceitos arquiteturais distintos que geram problemas de comunicação.

Aliás, o modelo arquitetônico, distinto dos trabalhos relacionados, na prática, fundamenta o desenvolvimento de mecanismos alternativos para a solução desses problemas, elevando também o estado da arte no plano funcional. A infraestrutura proposta padroniza a comunicação e possibilita a coexistência de Internets distintas, sem que haja processos adicionais que onerem os elementos de rede quanto a armazenamento, processamento e memória. Além disso, o modelo mitiga três níveis de dificuldade encontrados no estado da arte: implantação dos princípios arquitetônicos em ambiente corporativo, integração das novas FIAs nas infraestruturas de comunicação propostas e ineficiência dos mecanismos que possibilitam a coexistência de FIAs distintas, que, em alguns casos, podem perder algumas de suas funcionalidades.

1.4 Objetivo específico

A fim de alcançar o objetivo geral da tese de forma concreta, definem-se alguns objetivos específicos e, posteriormente, tenta-se granjeá-los de forma sistemática.

Seguem abaixo os objetivos específicos da proposta.

1. Operar a separação entre os planos de dados e de controle, a fim de flexibilizar a infraestrutura de comunicação proposta.

2. Desenvolver um protocolo que permite às FIAs integrar suas funcionalidades na rede, reconhecer os seus serviços nos dispositivos de comutação e gerenciar seus fluxos de controle e dados.

3. Estruturar mecanismos que permitem a programabilidade dos planos de dados e de controle pelos participantes da comunicação de tal forma que não comprometem os elementos de rede em relação aos sobrecarregamentos de armazenamento, memória e processamento; que não dificultam a integração de novas FIAs na infraestrutura proposta; e, que possibilitam uma integração eficiente, ou seja, sem perda de funcionalidades.

4. Avaliar qualitativamente os princípios arquitetônicos da solução em relação às limitações da atual arquitetura de Internet; às necessidades das arquiteturas de Internet do futuro; e aos trabalhos relacionados, posicionando-a, assim, no estado da arte.

5. Avaliar qualitativamente a viabilidade da implementação e da implantação dos princípios arquitetônicos nas redes atuais.

6. Avaliar quantitativamente o desempenho dos aplicativos das FIAs que participam da comunicação, considerando a experiência do usuário.

1.5 Hipótese

Uma infraestrutura de rede baseada em SDN, com planos de dados e controle programáveis (FIXP), viabiliza a coexistência de múltiplas arquiteturas de Internet (FIAs) sem exceder os limites operacionais de processamento e memória dos dispositivos de comutação atuais, mantendo uma complexidade de implementação inferior à de soluções que utilizam mecanismos de interoperabilidade e/ou implementação de pilhas de protocolos.

1.6 Desafios e questões de pesquisa

Os desafios de pesquisa podem ser descritos com base nas características fundamentais da proposta desta tese.

Poucos são os trabalhos da literatura correlata que possibilitam a coexistência de FIAs distintas, e por conta disso, não estabelecem, de forma definitiva, princípios arquiteturais que solucionam os problemas desta comunicação.

Por outro lado, cada FIA possui características próprias: dispositivos de comutação (switches), funcionalidades, orientações de design (orientadas a host, conteúdo, workspace, etc.), paradigmas de comunicação (push, pull), esquemas de endereçamento (identificadores e localizadores de recursos), protocolos, entre outras.

A elaboração de um modelo conceitual novo que oferece uma infraestrutura de comunicação que fornece interfaces que padronizam a comunicação entre entidades robustas e heterogêneas, como é o caso de arquiteturas de Internet, e, simultaneamente, soluciona os problemas de comunicação existentes nos trabalhos correlatos tais como dificuldade de integração de novas FIAS, ineficiência (perda de funcionalidades) no processo de integração, sobrecarregamento dos elementos de rede, entre outros; é um desafio que gera várias questões de pesquisa.

***Questão 1**: Se cada FIA possui seus próprios switches, como adequá-los à outra rede de comunicação?*

***Questão 2**: Se cada FIA possui diferentes funcionalidades, paradigmas de comunicação e orientações de design; como adaptá-las à infraestrutura de comunicação?*

***Questão 3**: Como reconhecer a semântica e a sintaxe de diferentes protocolos de rede que possuem diferentes esquemas de endereçamento?*

***Questão 4**: Como suportar a evolução dos protocolos, das funcionalidades e dos princípios arquiteturais de uma FIA que já se encontra integrada à rede?*

***Questão 5**: Como instalar diversos protocolos e funcionalidades de FIAs distintas sem sobrecarregar os dispositivos de comutação do plano de dados?*

***Questão 6**: Como mitigar a dificuldade de integração de novas FIAs em relação aos trabalhos correlatos?*

***Questão 7**: Como fazer a implantação de FIAs de maneira eficiente, ou seja, sem perda de funcionalidades?*

***Questão 8**: Como viabilizar a implantação da meta-arquitetura proposta no ambiente corporativo, em escala mundial?*

1.7 Contribuições

A evolução contínua da camada de aplicação, juntamente com as limitações da Internet, está aumentando os desafios das infraestruturas de rede nas empresas de telecomunicações. Nas redes atuais, cada tipo de serviço de rede é suportado por uma capacidade de infraestrutura específica (BOUBENDIR; BERTIN; SIMONI, 2016), o que engessa a solução, pois, se a demanda muda, os protocolos de comunicação podem se modificar, o que pode exigir a inclusão de novos elementos na infraestrutura física da rede ou até a substituição dos existentes. Isso torna as arquiteturas atuais das empresas de telecomunicações inadequadas para atender dinamicamente aos requisitos das aplicações (BOUBENDIR;

BERTIN; SIMONI, 2016), e por conseguinte, isso pode causar uma lacuna entre as necessidades da aplicação e as capacidades de rede.

Essa discussão prévia é para mostrar que as contribuições do *FIXP* podem estar diretamente ligadas ao objetivo geral desse trabalho, mas também podem extrapolá-lo em um futuro próximo, pois os princípios arquitetônicos do FIXP têm uma gênese que pode contribuir significativamente para as tendências de redes futuras.

Um exemplo de extrapolação é que o *FIXP*, na ânsia de criar uma arquitetura em que coexistam várias *FIAs*, também possibilita uma infraestrutura que pode, no futuro, fornecer uma rede como serviço (*Network as a service (NaaS)*) (BOUBENDIR; BERTIN; SIMONI, 2016; AFOLABI et al., 2018). Do ponto de vista conceitual, pode-se dizer que se tratará de uma rede mais completa do que aquelas oferecidas por arquiteturas da próxima geração, como, por exemplo, a arquitetura 5G (AHMADI, 2019; BARAKABITZE et al., 2020). A arquitetura em questão é capaz de oferecer redes lógicas dinâmicas, com elasticidade suficiente para atender em tempo de execução a manutenção da qualidade de serviço de uma comunicação (BARAKABITZE et al., 2020; AHMADI, 2019; AFOLABI et al., 2018), mas não são capazes ainda de oferecer uma nova arquitetura de Internet para casos de uso específicos. O oferecimento dessas redes lógicas tradicionais, no 5G, herda as limitações da rede TCP/IP, enquanto uma possível rede oferecida pelo *FIXP* apresenta uma nova arquitetura de Internet, adequada às necessidades do usuário. Essa nova rede *FIXP* pode tratar, de forma nativa, questões que limitam a Internet atual, como por exemplo, multicasting, mobilidade e inclusão de novos serviços de rede sem que seja necessário modificar a sua infraestrutura física, que é configurável, extensível e programável.

Um segundo exemplo pode ser citado. Outro requisito das redes da próxima geração é aproximar os usuários das funções de rede, permitindo que as aplicações consigam consumir recursos da operadora conforme suas necessidades, por meio de interfaces programáticas (*Application Programming Interfaces (APIs)*) (ORDONEZ-LUCENA; DSOUZA, 2022). Para que isso seja possível, as redes têm, necessariamente, que separar a lógica dos serviços de rede da infraestrutura física. O projeto Camara (ORDONEZ-LUCENA; DSOUZA, 2022), por exemplo, é uma iniciativa que propõe a produção em escala industrial de *APIs* para promover essa aproximação. No entanto, devido às limitações do TCP/IP, essa aproximação entre o usuário e os serviços de rede é realizada pela camada de aplicação. O problema disso é que a complexidade da comunicação é controlada pela própria aplicação, não é oferecida de forma transparente pelos serviços de rede. No caso do *FIXP*, ele permite essa aproximação por meio dos controladores implantados. Isso permite que a rede abstraia a complexidade de execução do serviço oferecido, oferecendo-o de forma transparente à aplicação. Um exemplo é que o usuário solicita ao controlador ETArch serviços de *multicast* por meio de protocolos da ETArch que estão no nível de rede, e não no nível de aplicação. O controlador *ETArch* então oferece esse canal de comunica-

ção ao usuário de forma transparente, abstraindo toda a complexidade da comunicação. No caso da arquitetura *NovaGenesis*, o controlador publica e permite a assinatura de serviços pelos usuários/aplicações, garantindo, inclusive, acordos de comunicação (*Service Level Agreement (SLA)*). Nesse caso, a arquitetura *NovaGenesis* também oferece seus serviços em nível de rede, não em nível de aplicação, abstraindo a complexidade da comunicação. Se as operadoras desejarem, na infraestrutura FIXP, fornecer seus serviços de rede para aplicações/usuários por meio de *APIs* de serviço desenvolvidas, por exemplo, no projeto CAMARA (ORDONEZ-LUCENA; DSOUZA, 2022), elas devem desenvolver seus próprios controladores-FIXP para oferecer esse recurso. Aliás, esses controladores devem ser distribuídos para que o serviço possa ser oferecido entre domínios.

Seguem abaixo as contribuições desse trabalho: aquelas diretamente relacionadas ao objetivo geral e aquelas que extrapolam os objetivos arquitetônicos, mas podem ser essenciais como contribuições diretas para as redes da próxima geração.

1. O trabalho oferece um modelo conceitual e lógico de princípios arquitetônicos, até então inéditos, que possui a capacidade de suportar a coexistência de aplicações que utilizam arquiteturas de Internet distintas e que, além disso, mitiga e/ou soluciona os problemas de comunicação encontrados nos trabalhos relacionados, quais sejam: dificuldade de implementação e implantação de novas *FIAs*; adicionamento de processos que oneram a capacidade dos elementos de rede; integração ineficiente das *FIAs*, com possível perda de funcionalidades; dificuldade, em alguns casos, de implantação da meta-arquitetura nas redes atuais.

2. O trabalho propõe uma infraestrutura de rede programável, extensível e configurável nos planos de dados e controle. Isso permite desacoplar a lógica dos serviços do substrato físico, eliminando a necessidade de expansão ou substituição da infraestrutura física a cada nova demanda do usuário. Na rede FIXP, novos requisitos são atendidos via incorporação de novos serviços ou atualização dos existentes; mantendo-se o aparato de hardware inalterado, salvo em cenários que exigem expansão de escalabilidade.

3. Os princípios arquitetônicos do *FIXP* possibilitam a materialização de uma rede sob demanda que, embora não disponha dos recursos de uma rede nativa em nuvem, apresenta uma proposta conceitual superior às redes lógicas oferecidas pelo 5G. Isso ocorre porque a rede oferecida pelo FIXP estabelece uma nova arquitetura de Internet sob demanda, transcendendo as limitações herdadas do modelo TCP/IP que ainda restringem as soluções de redes lógicas contemporâneas.

4. O protocolo *FIXP* padroniza a comunicação entre o plano de dados programável e os controladores independentemente dos fornecedores de elementos de rede; o que ainda é um problema nas redes tradicionais.

5. Os princípios arquitetônicos do FIXP são avaliados quantitativamente, demonstrando, assim, experimentalmente, uma prova de conceito da implantação das *FIAs* na meta-arquitetura proposta. Além disso, 22 casos de uso distintos são realizados experimentalmente para verificar o comportamento dos switches FIXP, o sobrecarregamento da infraestrutura de rede e o desempenho de aplicações que geram diferentes naturezas de tráfego: Chat (textos pequenos) e Streaming de Vídeo (transmissão de vídeos contínuos sob demanda). Vinte e dois casos de uso, com diferentes aplicações e variáveis de controle, estressam a infraestrutura de comunicação de tal forma a fazer várias medições que verificam as limitações e os benefícios técnicos da tecnologia escolhida. A avaliação quantitativa deixa um rastro de dados organizados e de métodos de teste validados para a comunidade científica.

6. O trabalho desenvolve a primeira versão nativa do controlador ETArch.Openflow na linguagem *P4*.

7. Os resultados da pesquisa são publicados em conferências e periódicos de alta relevância nacionais e internacionais. Todos os conhecimentos adquiridos são disponibilizados para comunidade, visando a evolução da ciência em relação ao tema estudado.

1.8 Organização da tese

Esta seção apresenta a organização da tese de doutorado em relação aos itens textuais que devem estar presentes em um trabalho dessa natureza (Subseção 1.8.1) e aos objetivos específicos, que vão sendo cumpridos ao longo do trabalho (Subseção 1.8.2). Essa última subseção também aponta a localização, na estrutura apresentada, das respostas aos desafios de pesquisa.

1.8.1 Organização Geral

A organização geral da tese foca nos procedimentos adotados ao longo do desenvolvimento deste trabalho para cumprir os objetivos gerais (ver Seção 1.3) e específicos (ver Seção 1.4), bem como para demonstrar a hipótese formulada na Seção 1.5. A sequência de atividades realizadas, de forma customizada, é: estudo da bibliografia, formulação do problema, proposta de uma solução, avaliação dessa proposta diante de soluções já existentes e considerações finais sobre a relevância do trabalho.

A figura 2 apresenta a organização geral desta tese de doutorado.

Segue abaixo, uma descrição customizada de cada um dos capítulos.

1. O Capítulo 1 propõe um objeto de estudo (coexistência de *FIAs* nas redes atuais); descreve o contexto atual em relação ao tema; identifica o problema; explana

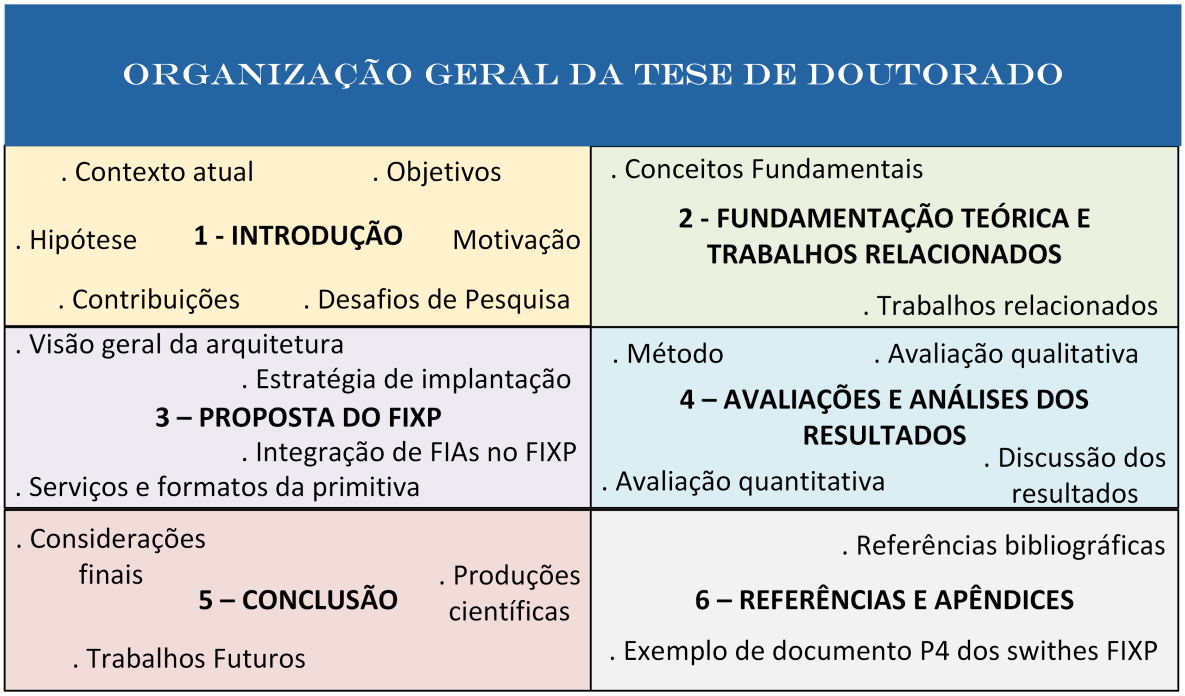

Figura 2 – Organização geral da tese de doutorado.

as dificuldades expressas no problema em termos de desafios de pesquisa; elabora hipóteses ou conjecturas que podem, a priori, solucionar parcialmente ou inteiramente os problemas identificados; descreve objetivos que devem ser alcançados para superar os problemas encontrados e evoluir o objeto de estudo no estado da arte. Esse capítulo é diretamente responsável pela justificação da pesquisa, estabelecendo, assim, o objeto de estudo como relevante para uma tese de doutorado.

2. O Capítulo 2 apresenta duas seções: Fundamentação teórica e Trabalhos relacionados. A seção "Fundamentação teórica" descreve os princípios arquitetônicos do SDN, paradigma utilizado na solução desta tese, apresentando abordagens diferentes, tais como *P4* e *Openflow*. A diferença entre essas abordagens justifica as decisões de projeto do *FIXP*. Esse capítulo também descreve várias arquiteturas de Internet do Futuro que são integradas ao *FIXP* ou a trabalhos correlatos. Por último, faz-se uma reflexão sobre as infraestruturas de rede do futuro, descrevendo suas tendências conceituais e tecnológicas. O objetivo é promover uma discussão posterior sobre as contribuições da solução FIXP para as redes da próxima geração. A seção "Trabalhos relacionados" apresenta os trabalhos correlatos, diretamente ou indiretamente relacionados ao *FIXP*. Alguns desses trabalhos não possuem exatamente o mesmo objetivo arquitetônico do *FIXP*, mas ainda sim são relacionados porque utilizam mecanismos de redes programáveis. Outros trabalhos possuem o mesmo objetivo arquitetônico, mas os mecanismos de atingi-lo são distintos do *FIXP*. O capítulo produz uma tabela comparativa das características de todos os trabalhos relacionados, que servirá de fonte para as avaliações qualitativas realizadas em capítulos

posteriores.

3. O Capítulo 3 percorre uma longa jornada pelo *FIXP*, detalhando a proposta da tese. Primeiramente, faz uma definição conceitual da arquitetura, para que não aja dúvidas sobre os termos que são utilizados no decorrer do trabalho. Logo após descreve a arquitetura: visão geral, comunicação interna dos componentes, serviços e formato das primitivas do protocolo; e, blocos funcionais fixos e programáveis dos *switches FIXP* utilizados nos experimentos. Além disso, propõe estratégias de implantação do FIXP nas redes atuais de forma gradual, até alcançar escala global. Outro ponto importante do capítulo é a estratégia de integração de novas FIAs no FIXP, bem como a apresentação de diagramas de sequência que mostram as trocas de mensagens entre as FIAs implantadas e a rede FIXP, detalhando, assim, o funcionamento lógico dos blocos funcionais da arquitetura.

4. O Capítulo 4 realiza as avaliações da arquitetura proposta no Capítulo 3. Esse capítulo apresenta três etapas de avaliações qualitativas que visam avaliar as características do FIXP em relação aos trabalhos correlatos, posicionando-o no estado da arte; avaliar a relevância dos princípios arquitetônicos do *FIXP* em relação ao TCP/IP e aos trabalhos correlatos; e, avaliar a eficiência da implementação e implantação desses princípios nas redes atuais. A quarta etapa avaliativa é quantitativa, são executados 22 casos de uso selecionados que tem três objetivos principais: provar, conceitualmente, a implementação do *FIXP* e a coexistência de *FIAs* em um mesmo ambiente de comunicação; avaliar a performance de aplicações ETArch e IP que participam da comunicação e trafegam diferentes naturezas de tráfego: vídeo e texto; avaliar o sobrecarregamento da infraestrutura *FIXP* no momento da execução dos casos de uso mencionados. O método utilizado para realizar as experimentações é descrito, detalhadamente, logo no início do capítulo. Ao longo das avaliações, têm-se dados suficientes para responder às questões de pesquisa, fazer interpretações dos resultados adquiridos e discutir o cumprimento dos objetivos da tese. Além disso, no final do capítulo são realizadas duas reflexões que extrapolam o objetivo geral do trabalho, porém aprofundam o conhecimento da arquitetura FIXP frente à evolução das redes do futuro: a primeira reflexão é sobre as contribuições do FIXP para incorporar arquiteturas de Internet do Futuro nas redes corporativas; a segunda reflexão é sobre as contribuições da solução FIXP para as redes da próxima geração.

5. O Capítulo 5 sintetiza todas as discussões realizadas nos Capítulos 2, 3 e 4; descrevendo a relevância do *FIXP* no estado da arte. Essas discussões estão em uma subseção intitulada "*Considerações finais*". Em seguida, o capítulo cita as produções bibliográficas que foram aprovadas ou submetidas atualmente (ainda sem resposta). Por último, o capítulo descreve um conjunto de problemas ainda não tratados, cujas

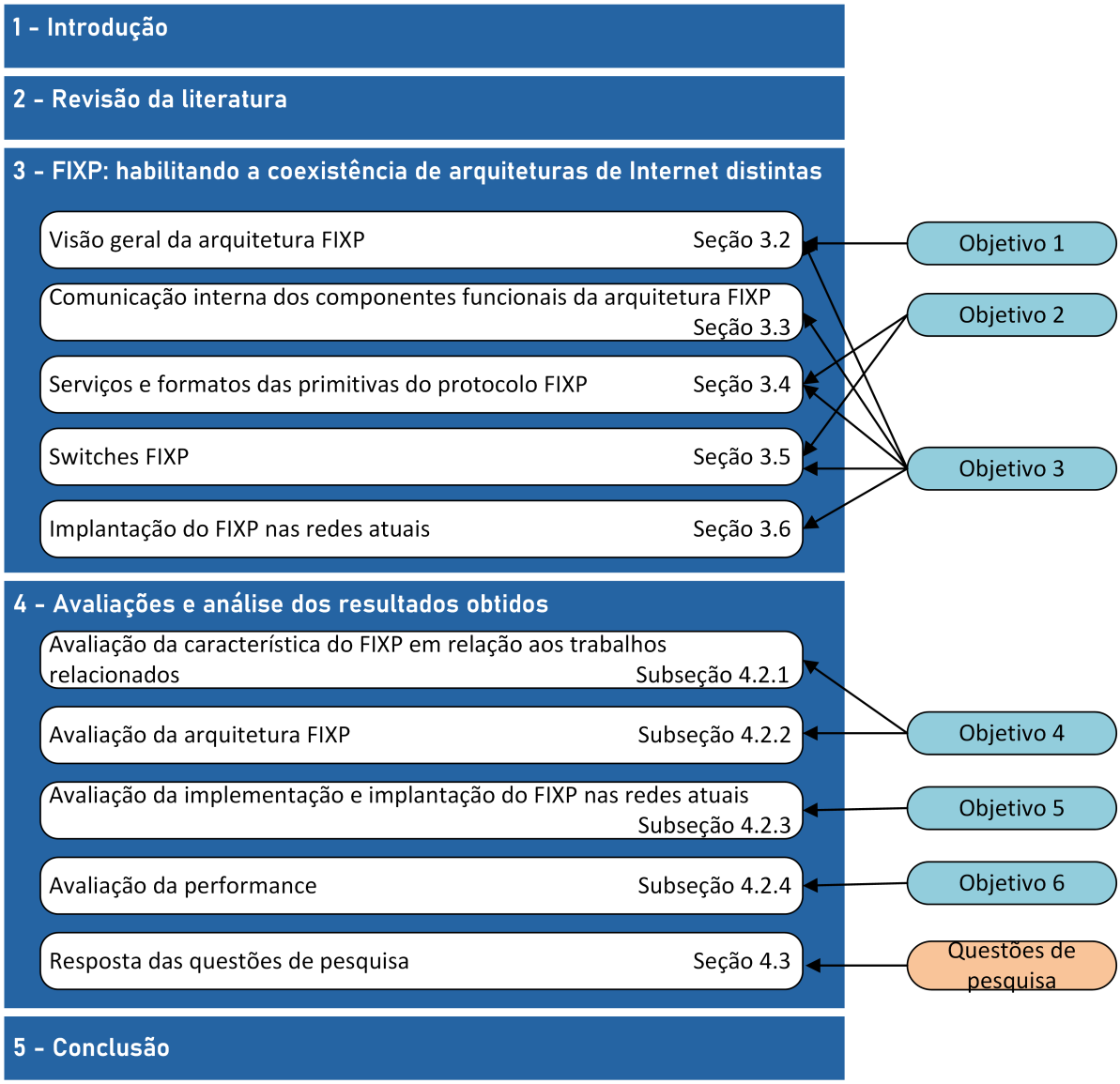

Figura 3 – Mapeamento da localização dos objetivos específicos e das respostas às questões de pesquisa na estrutura da tese.

soluções, certamente, vão evoluir o *FIXP*, caso sejam implementadas futuramente na arquitetura.

6. O Apêndice A apresenta, parcialmente, o documento *P4* (programa *P4*), que especifica o plano de dados das *FIAs* implantadas na infraestrutura FIXP. O intuito desse apêndice é demonstrar que os algoritmos são descritivos, sequenciais e simples, de tal maneira que a execução deles em tempo real não sobrecarrega os dispositivos de rede; já que suas funcionalidades são padronizadas para qualquer *switch* de comutação, seja ele programável ou tradicional.

1.8.2 Organização relacionada aos objetivos específicos e às questões de pesquisa

O cumprimento dos objetivos específicos e a resposta aos desafios de pesquisa são parte obrigatória de uma tese de doutorado. A Figura 3 apresenta o mapeamento da localização dos objetivos específicos (descritos em 1.4) e das respostas às questões de pesquisa (descritas em 1.6) na estrutura da tese.

Capítulo 2

Fundamentação teórica e trabalhos relacionados

Este capítulo apresenta duas seções principais.

A Seção 2.1 descreve os principais embasamentos filosóficos, conceituais e arquiteturais que formam a base referencial da construção do FIXP. Os principais fundamentos são apresentados de forma detalhada, como também as discussões que envolvem seu entorno. O entendimento dessa seção é pré-requisito para o aprendizado dos princípios arquitetônicos da solução proposta neste trabalho.

A Seção 2.2 descreve trabalhos que possuem similaridade com o proposto. A discussão de cada um desses trabalhos se aprofunda, na medida que a seção avalia cada um desses trabalhos qualitativamente, mostrando os principais mecanismos utilizados por cada um deles e as consequências que essas escolhas de projeto determinam na comunicação. Essas análises possibilitam a criação de uma tabela comparativa de características que, posteriormente, auxilia a posicionar o FIXP no estado da arte.

2.1 Fundamentação teórica

As tecnologias emergentes no domínio da Tecnologia de Informação e Comunicação (TIC) evoluíram, estão cada vez mais exigentes e demandam cada vez mais requisitos da rede. Por um lado, arquiteturas de rede tais como 5G, Internet das Coisas (IoT), computação em nuvem, sistemas autônomos tais como redes baseadas em intenção e mecanismos de *Self-healing* precisam cada vez mais de altas taxas de transferência, acessibilidade onipresente e gerenciamento dinâmico da rede (KAUR; KUMAR; AGGARWAL, 2021). Por outro lado, esse contexto evolutivo modifica o perfil das aplicações e serviços que começam a ser executadas por dispositivos variados tais como *laptops*, *tablets* e *smartphones*. O aparecimento dessas novas aplicações juntamente com a evolução tecnológica impõem sobre a Internet a necessidade de suportar novos requisitos, como QoS, QoE, mobilidade, segurança, tempo real (*real-time*), *multicasting*, entre outros.

No entanto, a arquitetura de rede tradicional não é capaz de atender a essas demandas devido ao engessamento das camadas TCP/IP que dificulta a inclusão de novos serviços (KAUR; KUMAR; AGGARWAL, 2021). Além disso, os elementos de comutação da infraestrutura física também são ossificados; não conseguem atender as demandas devido ao forte acoplamento entre seus sistemas operacionais e o plano de dados (KAUR; KUMAR; AGGARWAL, 2021). Historicamente, os dispositivos de encaminhamento tradicionais são inflexíveis e suportam um conjunto fixo de protocolos devido às características intrínsecas dos seus chips (ZANNA; RADCLIFFE; CHAVEZ, 2019; MCKEOWN; REXFORD, 2016). Geralmente, os protocolos implantados são aqueles definidos por organizações de padronização internacionais, tais como Internet Engineering Task Force (IETF) (FORCE, 2025) e Institute of Electrical and Electronic Engineers (IEEE) (ELECTRICAL; ENGINEERS, 2025).

O montante dessas dificuldades citadas é a maior barreira para implementação de novos serviços: um bom exemplo é o IPv6, proposto em 1995 (BRADNER; MANKIN, 1995) e ainda não completamente implantado (PAPASTERGIOU et al., 2017; TURNER; TAYLOR, 2005). Outra consequência dos gargalos citados é o gerenciamento da rede, que se torna difícil e demorado, pois a topologia dos dispositivos apresenta interfaces de rede que variam entre vários fornecedores (KAUR; KUMAR; AGGARWAL, 2021).

Uma forma de resolver parte desses problemas é desacoplar o sistema operacional (plano de controle) do elemento de comutação da rede (plano de dados). Teoricamente, essa separação flexibiliza o desenvolvimento de funcionalidades de rede que hoje estão encapsuladas em switches proprietários; e, por consequência, facilita e sintetiza o gerenciamento da rede. Pode-se pensar na ideia de que um protocolo padronize a comunicação entre os planos de dados e controle; e, que controladores, que armazenam funcionalidades de controle e gerenciamento, consigam configurar a comunicação através de um conjunto de aplicativos centralizados; pode-se pensar também em flexibilizar o comportamento do switch através da programabilidade do plano de dados. São dois mecanismos distintos que estão sendo discutidos: o primeiro é a possibilidade de flexibilizar o comportamento do fluxo de dados a partir do plano de controle, o segundo é a possibilidade de flexibilizar o comportamento do switch (especificação de cabeçalhos, tabelas de encaminhamento, etc.) através da programabilidade do plano de dados. Enfim, são duas ideias que podem resolver grande parte dos gargalos; pois esses dois mecanismos podem facilitar a inclusão de novos serviços em uma rede de computadores, proporcionando, dessa forma, o desenvolvimento de protocolos que utilizam diferentes princípios arquitetônicos. Enfim, a possibilidade da separação do plano de controle do plano de dados é promissora e oferece possibilidades que até então não existiam na arquitetura tradicional.

Esta subseção apresenta, primeiramente, o embasamento conceitual e tecnológico do projeto FIXP (Subseções 2.1.1 à 2.1.5). Além disso, apresenta, detalhadamente, as arquiteturas de Internet do Futuro que são integradas na rede FIXP: ETArch (Subseção

2.1.6) e NovaGeneis (Subseção 2.1.7). Em seguida, descreve, brevemente, de forma geral, outras FIAs que são implementadas e implantadas nos projetos que envolvem os trabalhos relacionados (Subseção 2.1.8). Por último, faz-se uma reflexão sobre como será as infraestruturas de rede do futuro (2.1.9), descrevendo as tendências conceituais que essas redes vão implementar a fim de se tornarem mais dinâmicas, e por conseguinte, mais propensas a atenderem requisitos de aplicações atuais e futuras. O objetivo dessa reflexão é discutir posteriormente quais as contribuições que a solução FIXP oferece para as redes da próxima geração.

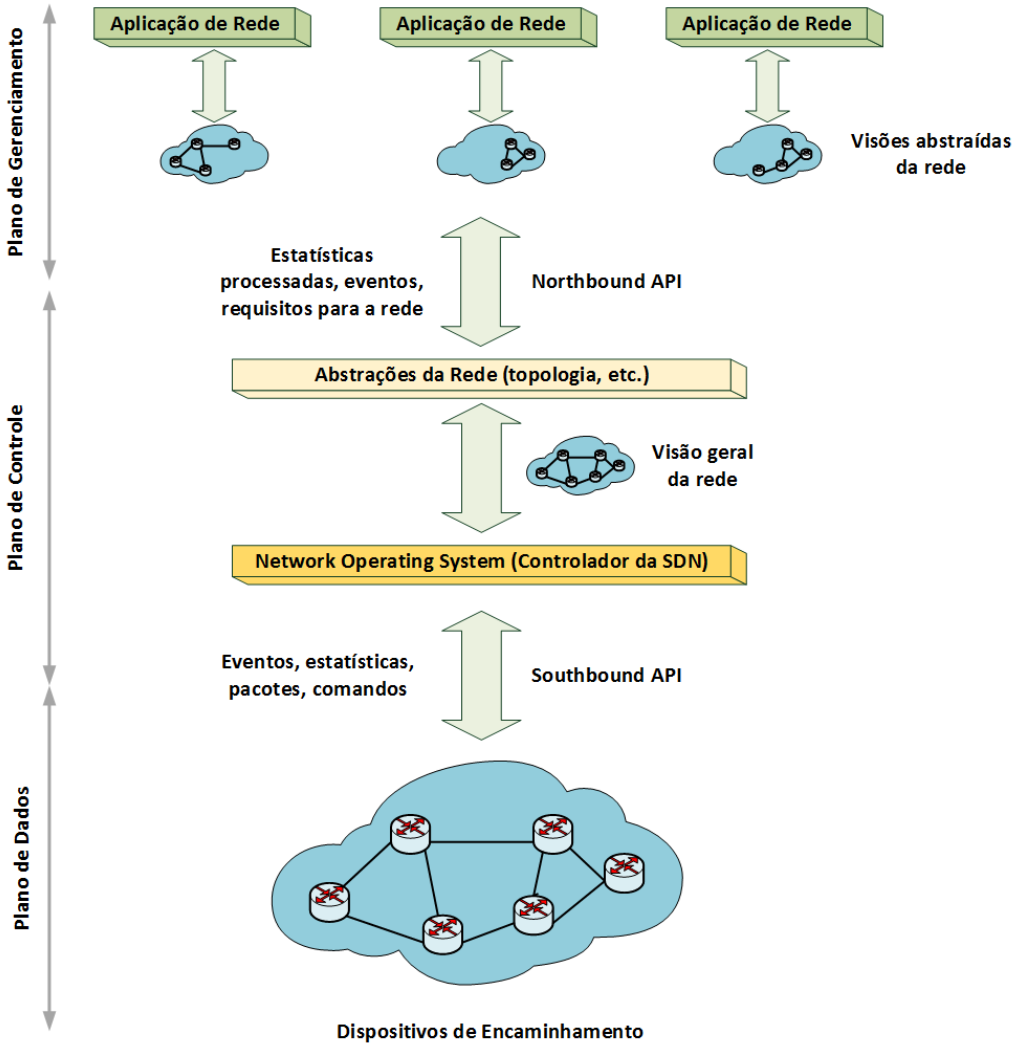

Figura 4 – Arquitetura de uma Rede Definida por Software. Fonte: (UFRJ, 2015).

2.1.1 Um breve história de Redes Definidas por Software

Open Networking Foundation (ONF) (FOUNDATION, 2025a) é um consórcio sem fins lucrativos que promove e democratiza a inovação em *SDN* (COX et al., 2017; KREUTZ et al., 2015) e cria padronizações de protocolos e tecnologias relacionadas.

SDN é um paradigma que repensa a rede quanto à forma de gerenciamento e operação. Uma das principais características é a separação do plano de controle dos elementos de comutação do plano de dados(KAUR; KUMAR; AGGARWAL, 2021). A ideia é retirar dos *switches* o monopólio de criação de funcionalidades de rede e colocá-lo em uma instância de software externa logicamente centralizada. Essa nova concepção introduz uma nova infraestrutura de rede que consiste em três camadas: camada de infraestrutura, camada de controle e camada da aplicação. A Figura 4 exibe uma visão geral da arquitetura SDN.

A camada de infraestrutura (XIE et al., 2015) compreende elementos de encaminhamento físicos ou virtuais tais como *switches*, roteadores, pontos de acesso, entre outros. Essa camada representa a infraestrutura física subjacente e é responsável direta pelo encaminhamento de primitivas das arquiteturas de rede. Por isso compõe o que chamamos de plano de dados.

A camada de controle (XIE et al., 2015) representa o software que controla a rede. Quando esse software é implementado recebe o nome de controlador SDN. Ele pode fornecer duas interfaces importantes. A primeira se chama Interface *SouthBound*, atua entre a camada de infraestrutura física e a camada de controle, abstrai a complexidade do plano de dados e facilita, dessa forma, a execução de operações nos dispositivos de comutação, como por exemplo, a manutenção dos registros das tabelas de encaminhamento. A segunda se chama Interface *NorthBound*, atua entre a camada de controle e as aplicações de rede, e oferece abstrações que facilitam o gerenciamento da rede, como por exemplo, a visão geral dos elementos de comutação na topologia subjacente. A camada de controle é um *broker* que possibilita a comunicação de aplicações de alto nível com os elementos de encaminhamento de rede. Por isso compõe o que chamamos de plano de controle.

A camada de aplicação (XIE et al., 2015) contém programas que gerenciam a rede através das interfaces que o controlador oferece.Essa camada representa a inteligência da rede, é onde estão as abstrações das suas funcionalidades. Através das aplicações, os gerenciadores de rede podem desenvolver mecanismos de roteamento, *firewall*, balanceadores de carga, monitoramento de tráfego, detecção de ataque, agregação e desagregação de pacotes, entre outros (KAUR; KUMAR; AGGARWAL, 2021; JOUET; CZIVA; PEZAROS, 2015). Essa camada compõe o que chamamos de plano de gerenciamento.

SDN possibilita a introdução de novas abstrações, simplifica o gerenciamento, facilita a evolução e inovação da rede (JOUET; CZIVA; PEZAROS, 2015). Por conta disso, é um paradigma que ganhou atenção tanto da indústria quanto do mundo acadêmico. Empresas como *Google*, *AT&T*, *Microsoft* e *Cisco* migraram para *SDN* para reduzir o custo da rede e acelerar a entrega de serviços orientados ao cliente (KAUR; KUMAR; AGGARWAL, 2021). Outros benefícios que podem ser apontados são (KAUR; KUMAR; AGGARWAL, 2021): protocolo padrão para configuração dos dispositivos de vários fornecedores; facilitação de desenvolvimento de mecanismos de automação de rede; custo reduzido de CAPEX e OPEX.

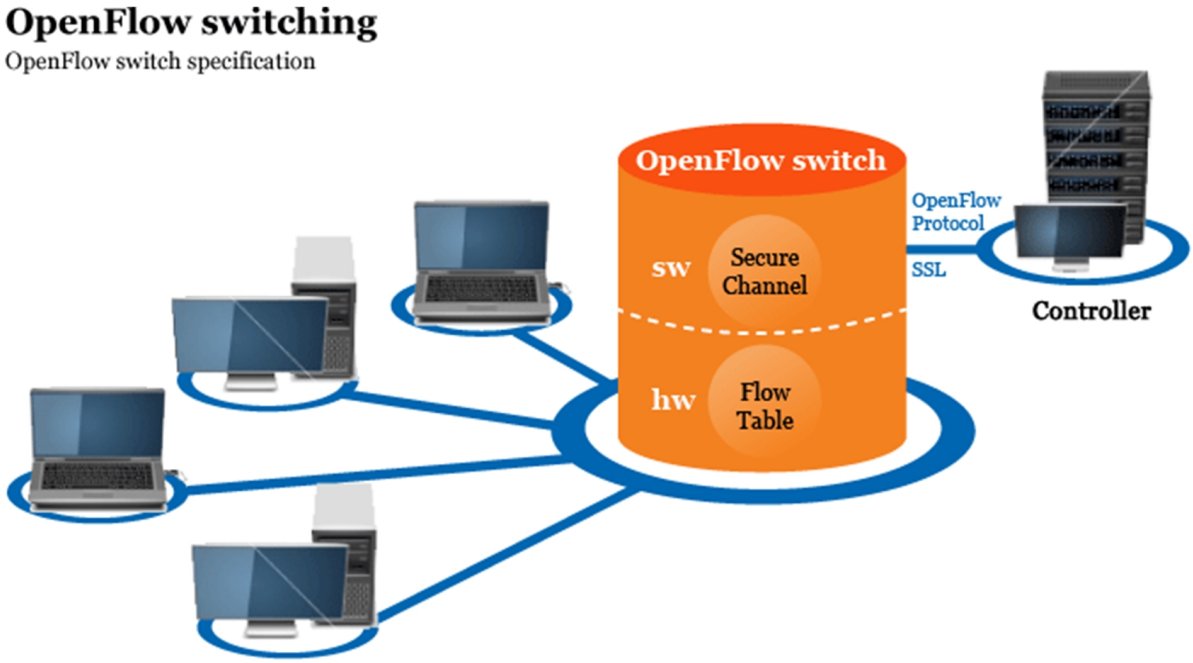

Figura 5 – Arquitetura SDN-OpenFlow. Adaptado de (MCKEOWN et al., 2008).

2.1.2 *OpenFlow*

OpenFlow (MCKEOWN et al., 2008) é uma instanciação do paradigma SDN, consolidada e amplamente utilizada por pesquisadores da área acadêmica e fabricantes de dispositivos de rede (SOFTWARE, 2019). Trata-se de um protocolo que materializa o ambiente SDN. Conforme pode ser visto na Figura 5, esse ambiente é formado basicamente por switches e controladores que suportam o protocolo *OpenFlow*; e esses dispositivos são peças essenciais para a separação dos planos de controle e dados. Cada um desses elementos de rede possuem funções diferentes.

O *switch Openflow* pertence ao plano de dados, e por isso, é responsável pelo encaminhamento das primitivas de dados e controle da arquitetura de rede que está realizando a comunicação. Para isso, o *switch* contém uma ou mais tabelas de fluxo que auxiliam essa tarefa. Basicamente, se a primitiva de dados possui uma correspondência nas tabelas de fluxo, ela sofre uma ação; que poderia estar relacionada, por exemplo, ao encaminhamento dessa primitiva para uma porta de egresso do switch. Caso não haja correspondência, ela sofre uma outra ação, que poderia ser, por exemplo, o envio dessa primitiva ao controlador para que o mesmo analise a situação e tome as providências necessárias. Em relação às primitivas de controle, geralmente, são enviadas diretamente ao controlador.

O protocolo *Openflow* realiza a comunicação entre *switch* e controlador (HU; HAO; BAO, 2014) através de um canal de comunicação SSL/TLS (LARA; KOLASANI; RAMAMURTHY, 2014; FOUNDATION, 2015). Por isso, diz-se que *OpenFlow* implementa a *API Southbound*, que abstrai a complexidade da camada de infraestrutura física subjacente e facilita a comunicação do controlador com os dispositivos de rede. Dessa forma,

o controlador pode movimentar as tabelas de fluxo do switch através de comandos que incluem, excluem ou modificam as suas linhas de entrada; e, permitir, o gerenciamento da rede através da camada de aplicação.

O controlador utiliza o protocolo OpenFlow para modificar o perfil do encaminhamento das primitivas de dados, e também oferece uma interface de serviços para que as aplicações possam gerenciar a rede. Quanto maior a abstração da interface, mais facilmente as aplicações conseguem desenvolver as funcionalidades de rede. Essa interface denomina-se *API Northbound*. Nem todos os controladores *SDN* possuem essa interface. O controlador Open Network Operation System (ONOS) (FOUNDATION, 2025b) possui as APIs *Northbound* e *Southbound* bem definidas e por conta disso é bastante utilizado no desenvolvimento de projetos *SDN* (FOUNDATION, 2025b).

2.1.3 *Linguagem P4*

P4 (LANGUAGE, 2023) é uma linguagem que define o comportamento de *switches SDN* que se fundamentam em arquiteturas que dão suporte à programabilidade do plano de dados. Trata-se de uma linguagem de alto nível que define o comportamento do hardware sem que seja preciso entender os detalhes do dispositivo físico tais como portas lógicas ou registradores (KAUR; KUMAR; AGGARWAL, 2021). Embora o *P4* tenha sido projetado para programar *switches* físicos, seu escopo foi ampliado para abranger uma ampla variedade de dispositivos tais como placas de interface de rede, *Neural Processing Unit (NPU)*, *Field-programmable Gate Array (FPGA)* e *Application Specific Integrated Circuit (ASIC)* (LANGUAGE, 2023; KAUR; KUMAR; AGGARWAL, 2021). Por conta dessa extensão, o termo genérico *"target"* é utilizado para representar qualquer um desses dispositivos *P4* (LANGUAGE, 2023).

O fabricante de *targets P4* tem que fornecer pelo menos dois elementos que são essenciais para o seu funcionamento (KAUR; KUMAR; AGGARWAL, 2021; LANGUAGE, 2023): uma definição de arquitetura programável (modelo da arquitetura) e um compilador.

Um *target* é projetado a partir de um modelo de arquitetura, que pode ser pensada como se fosse um contrato entre o programa *P4* e o dispositivo (WORKING, 2024). Esse contrato é um conjunto de interfaces que o programa *P4* implementa a fim de configurar os blocos funcionais programáveis que compõem o pipeline do plano de dados, gerenciando, dessa forma, o comportamento do dispositivo. Nessa fase, o operador da rede, que pode ser, por exemplo, um controlador, pode configurar o reconhecimento dos seus serviços de controle, especificar tabelas de fluxo e ações; e, criar estruturas de dados temporárias ou permanentes que registram informações da comunicação.

A compilação de um conjunto de programas *P4* produz dois artefatos (LANGUAGE, 2023; KAUR; KUMAR; AGGARWAL, 2021): um arquivo, por exemplo, binário, que descreve a configuração dos componentes que compõem o plano de dados; e, uma *API*

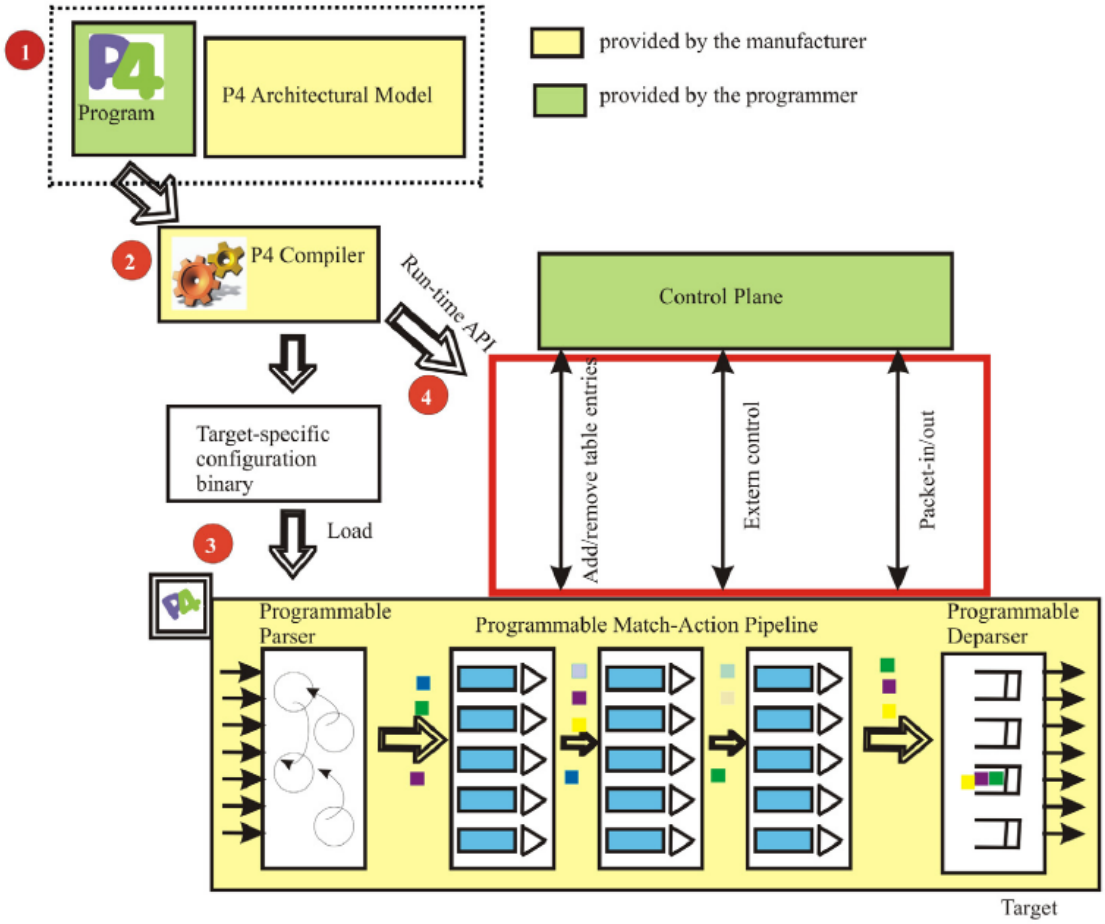

Figura 6 – Modelo Sintético de fluxo do plano de dados programável P4. Fonte: (KAUR; KUMAR; AGGARWAL, 2021).

P4Runtime (WORKING, 2024) para gerenciar o estado dos objetos do plano de dados por um controlador que se localiza dentro ou fora do target. O arquivo binário é utilizado no momento que o target é ligado, configurando, dessa forma, em tempo de execução, todos os blocos funcionais que fazem parte do pipeline de dados.

A Figura 6 mostra um modelo genérico e sintético de fluxo de trabalho do plano de dados programável *P4*.

A Figura 7 apresenta um programa *P4* que implementa as interfaces *Parser Logic*, *Match+Action Tables* e *Deparser Logic*; que configura seus respectivos blocos funcionais. Além disso, o programa define uma estrutura de dados denominada ethernet_t, que auxilia no reconhecimento do cabeçalho do quadro *Ethernet*.

Em comparação com sistemas de processamento de primitivas baseados na escrita de microcódigo em hardware personalizado, o *P4* oferece vantagens significativas (LANGUAGE, 2023).

Flexibilidade: as políticas de encaminhamento são programáveis, em contraste com os mecanismos de encaminhamento dos dispositivos tradicionais.

Expressividade: *P4* expressa algoritmos de processamento de pacotes que são indepen-

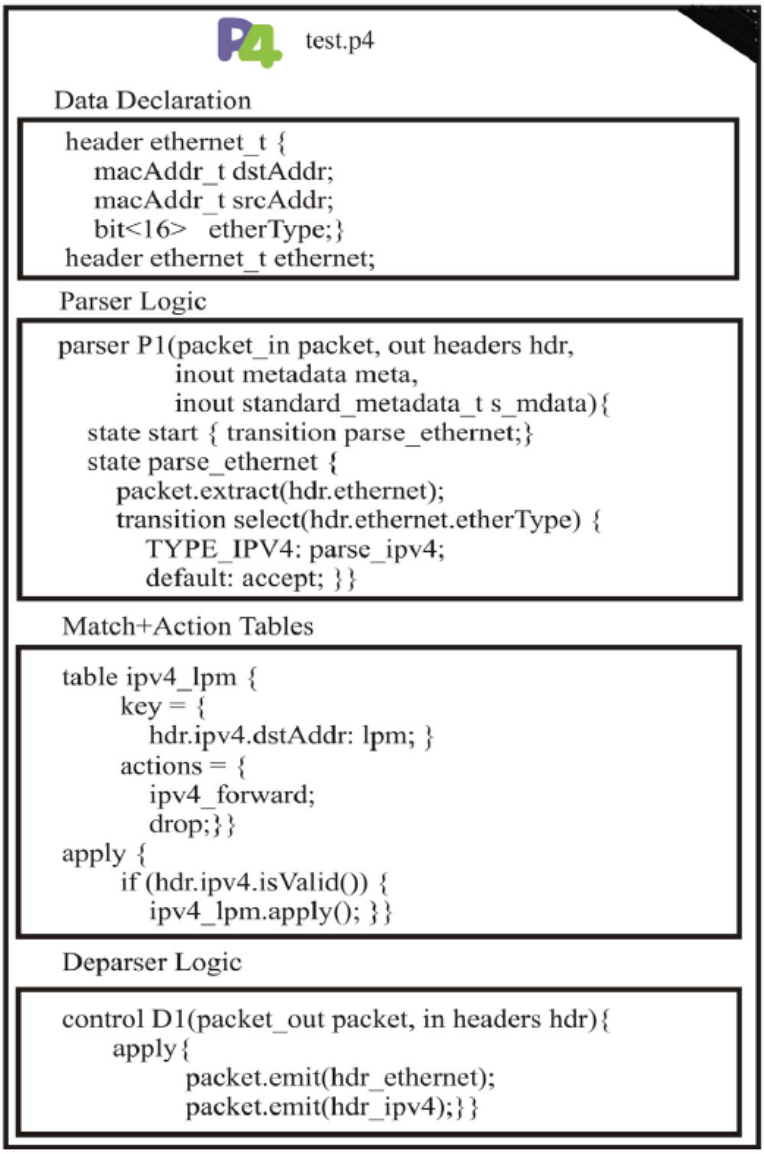

Figura 7 – Implementação das interfaces de um programa P4. Fonte: (KAUR; KUMAR; AGGARWAL, 2021).

dentes de hardware; pois utilizam apenas operações de uso geral e pesquisas em tabelas. Esses programas são portáteis entre hardwares que possuem a mesma arquitetura.

Abstração da complexidade: *P4* é uma linguagem de alto nível que abstrai a complexidade do hardware adjacente e mapeia, por exemplo, os recursos físicos de armazenamento disponíveis em componentes abstratos de software definidos pelo usuário. Dessa forma, o compilador pode gerenciar recursos físicos, como por exemplo, alocar um dado armazenado em variável específica.

Engenharia de software. Programas *P4* oferecem recursos de software interessantes tais como verificação de tipos e reutilização de software.

Bibliotecas de componentes. *P4* permite que o fabricante desenvolva bibliotecas específicas de funções fixas e programáveis. As programáveis são oferecidas através de uma

interface que é posteriormente implementada por um programa *P4*.

2.1.4 Diferenças entre OpenFlow e *P4*

Openflow é um protocolo que pressupõe que os *switches* tenham um comportamento fixo e bem conhecido (COX et al., 2017; MCKEOWN; REXFORD, 2016; KAUR; KUMAR; AGGARWAL, 2021). Normalmente, a folha de dados (*datasheet*) de um *switch ASIC* detalha esses comportamentos. Os chips de *switches* tradicionais de alto desempenho suportam um número limitado de protocolos padrões definidos por órgãos que regulam a Internet tais como *IEEE* e *IETF* (MCKEOWN; REXFORD, 2016). *Openflow* foi projetado para utilizar componentes de hardware de *switches* tradicionais, por isso, a primeira versão desse protocolo; apesar de permitir o preenchimento de tabelas de encaminhamento, reconhece apenas quatro protocolos comuns(*Ethernet*, *Virtual Local Area Networks (VLANs)*, *IPv4*, *Access Control Lists (ACLs)*), 12 campos de correspondência, máximo de 12 campos aos quais um único fluxo pode corresponder e máximo de 264 bits que uma entrada de fluxo pode conter (JOUET; CZIVA; PEZAROS, 2015; MCKEOWN; REXFORD, 2016; KAUR; KUMAR; AGGARWAL, 2021). À medida que o interesse cresceu, novas versões sugiram, e agora o *OpenFlow 1.5* (FOUNDATION, 2015), em sua última versão, suporta 44 campos de cabeçalhos (KAUR; KUMAR; AGGARWAL, 2021). A complexidade do protocolo *Openflow* está aumentando, mas ainda não oferece flexibilidade para estender o número de campos de cabeçalhos e ações personalizadas. O *Openflow* ainda é um protocolo fixo, com recursos limitados, que não oferece agilidade na adição de novos campos de correspondência; o que limita a quantidade de cabeçalhos e ações que ele reconhece.

Se por um lado o *Openflow* possui esta dificuldade, por outro, o número de campos de cabeçalhos novos que estão surgindo para atender requisitos de novas aplicações está crescendo com o surgimento de novos protocolos *clean-slate* (KARRAKCHOU; SAMAAN; KARMOUCH, 2020; GAVAZZA et al., 2020; FENG; TAN; JIN, 2019; GUO et al., 2021; SIGNORELLO et al., 2016).

Uma das alternativas para mitigação desses problemas foi a introdução do *Openflow eXtended Match (OXM)*; que flexibiliza o reconhecimento de campos de correspondência; porém os *switches Openflow* são formados por placas ASIC de função fixa; e, são necessários cerca de 04 anos para incrementar o suporte de um novo protocolo nesse tipo de dispositivo (MCKEOWN; REXFORD, 2016). Outras limitações referem-se, por exemplo, à dificuldade do *Openflow* em fazer correspondência através de um intervalo de endereços (KAUR; KUMAR; AGGARWAL, 2021). É importante salientar que nem todos os *switches SDN* suportam todos os cabeçalhos *OpenFlow*, o que é outro problema. Nesse caso, o dispositivo pode informar ao controlador quais os recursos que ele suporta através do padrão *Table Type Patterns (TTPs)* (FOUNDATION, 2014).

Esse cenário motivou empresas como *Google*, *Microsoft*, *Intel*, *Stanford*, *Princeton* e *Barefoot* a definir a linguagem *P4* (KAUR; KUMAR; AGGARWAL, 2021). A grande novidade dessa linguagem, que potencializa a diferença com o protocolo *OpenFlow*, é a programabilidade do plano de dados. Por um lado, tanto *P4* quanto *OpenFlow* permitem a separação dos planos de dados e de controle. Essa diferença entre esses dois paradigmas *SDN* permite que o *P4* possa dividir o seu plano de controle de tal forma que parte dessa inteligência pode ser colocada diretamente no *switch* e parte dessa inteligência em um dispositivo logicamente centralizado. Várias pesquisas, no estado da arte, estão utilizando esse potencial do *P4*, de programabilidade do plano de dados, para fazer experimentos de balanceamento de carga, monitoramento de fluxo, detecção de ataques (*Distributed Denial-Of-Service (DDoS)*) e agregação e desagregação de pacotes (KAUR; KUMAR; AGGARWAL, 2021; COX et al., 2017).

Tanto o *P4* quanto o *OpenFlow* conseguem controlar o comportamento do fluxo; mas *P4* é mais flexível e programável, e por conta disso consegue se adaptar a qualquer protocolo tradicional ou *clean-slate* (ROBERTS, 2009). Somente o *P4* consegue influir no comportamento do *switch*, ou seja, só o *P4* consegue programar o plano de dados de tal forma que ele possa definir tabelas de encaminhamento, personalizar ações, armazenar dados em estruturas/registradores pré-definidos pela linguagem, entre outros.

Algumas perguntas tornam-se relevantes nessa discussão. Por que o *Openflow* não possui mecanismos de programabilidade do plano de dados? Por que ele não foi desenvolvido para chips de função programável? Porque os chips de função fixa não reconhecem um número maior de protocolos?

O número de protocolos reconhecidos em *switches ASIC-OpenFlow* de função fixa é baixo porque quanto maior o número de campos que o *OpenFlow* reconhece, maior tem que ser o tamanho da memória *Ternary Content Addressable Memory (TCAM)* instalada no *switch* para que o número de fluxos armazenados instantaneamente seja o mesmo (JOUET; CZIVA; PEZAROS, 2015). Um *switch Openflow Top-of-Rack* tal como o *Pronto 3290 (FireBolt 3 switching ASIC)*, que suporta *OpenFlow* 1.0, possui cerca de 2000 fluxos que podem ser inseridos simultaneamente. No caso desse mesmo *switch* suportar o *OpenFlow 1.5*, apenas 700 entradas poderiam ser acomodadas naquele instante (JOUET; CZIVA; PEZAROS, 2015). Quanto à utilização de ASICs de função fixa para desenvolvimento do protocolo *OpenFlow*, uma das razões é porque, até bem pouco tempo, esses chips possuíam desempenho de cerca de 10 a 100 vezes maior do que um chip programável (MCKEOWN; REXFORD, 2016). Hoje já existem arquiteturas *P4* padrões que tornam os chips de função programável com desempenhos tão eficientes quanto os chips de função fixa (MCKEOWN; REXFORD, 2016). Um exemplo dessas arquiteturas é a *Protocol Independent Switch Architecture (PISA)* (BLEULER et al., 2003) (MCKEOWN; REXFORD, 2016). Agora, se podemos programar *switches* rápidos com *P4*, nada impede que dispositivos mais lentos tais como *Neural Processing Units (NPUs)*, *Field-programmable Gate*

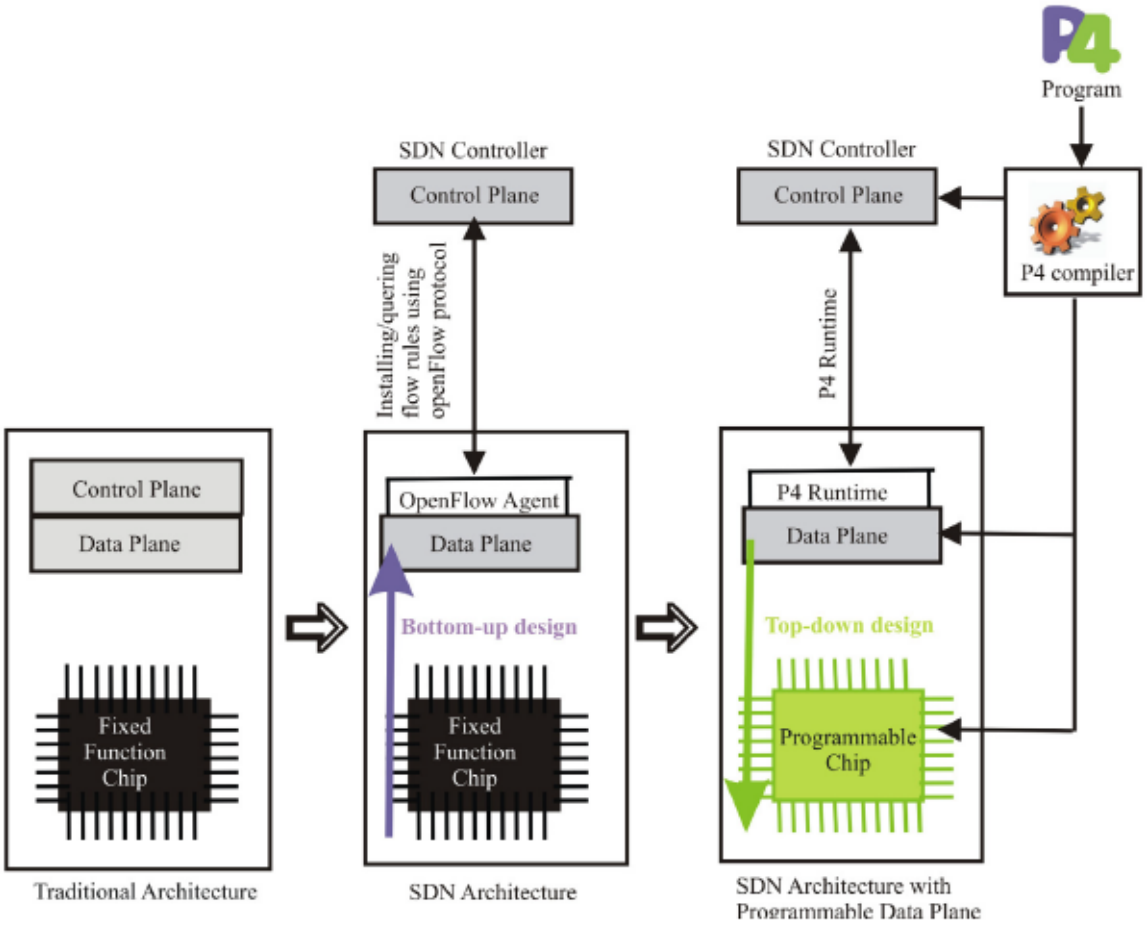

Figura 8 – Evolução da arquitetura de rede tradicional para o modelo *SDN* com plano de dados programável. Fonte: (KAUR; KUMAR; AGGARWAL, 2021).

Arrays (FPGAs) e *switches* de software também sejam programados pela mesma linguagem. Pode-se pensar até em descrever *switches* tradicionais utilizando *P4*. Dessa forma, a interoperabilidade entre dispositivos ficaria mais fácil, e as especificações dos *switches* menos ambíguas; já que poderiam ser descritas pela sintaxe *P4* ao invés de linguagem natural (MCKEOWN; REXFORD, 2016).

Conforme se apresenta as características do *P4*, pode-se pensar que o protocolo *OpenFlow* torna-se obsoleto; porém ainda existem muito *switches* de função fixa que suportam o *OpenFlow* (MCKEOWN; REXFORD, 2016). Vale lembrar que tanto *Openflow* quando *P4* possuem uma abordagem top-down quando se pensa em gerenciamento de rede, ou seja, ambos permitem que o controlador gerencie através de aplicações de alto nível a infraestrutura física subjacente. No entanto, apenas o *P4* possui uma abordagem top-down quando se pensa na programabilidade do plano de dados. Um programa *P4* configura o pipeline de dados, definindo dessa forma o seu comportamento. Desse ponto de vista, pode-se considerar o OpenFlow um subconjunto do *P4*; ou seja, pode-se considerar que os recursos do *OpenFlow* podem ser programados pela linguagem *P4*. Dessa perspectiva, *OpenFlow* seria um programa *P4* (MCKEOWN; REXFORD, 2016). Aliás, já existe

um programa *P4* chamado *OpenFlow.P4* que configura um *chip PISA* para suportar *OpenFlow* (MCKEOWN; REXFORD, 2016). As diferenças do *P4* e *Openflow* quanto à programabilidade do plano de dados podem ser vistas na Figura 8.

2.1.5 Direções futuras de redes SDN

SDN é uma abordagem promissora que fornece flexibilidade, programabilidade, automação de rede, protocolo padrão para configuração de dispositivos de vários fornecedores, gerenciamento de rede logicamente centralizado, entre outros (KAUR; KUMAR; AGGARWAL, 2021).

Apesar das vantagens, a arquitetura *SDN* possui problemas de escalabilidade e desempenho devido ao sobrecarregamento do controlador. Um exemplo é a falta de inteligência em *switches OpenFlow* para processamento de pacotes com estado (KAUR; KUMAR; AGGARWAL, 2021). Para casos de uso que dependem dessa característica, de manutenção de estado, o plano de dados depende fortemente de um controlador. Embora o *switch OpenFlow* forneça suporte parcial para processamento de pacotes com estado com auxílio de contadores, eles não armazenam as informações (KAUR; KUMAR; AGGARWAL, 2021). Outra questão já discutida é a limitação de campos reconhecidos por *switches OpenFlow*, o que impede a interação desses *switches* com novos protocolos (COX et al., 2017).

Nesse contexto, a programabilidade do plano de dados, em redes SDN, oferece a possibilidade de gerenciar não só o repasse do fluxo de dados, mas também o comportamento do próprio switch. Essa funcionalidade permite a configuração de blocos funcionais do dispositivo responsáveis pelo seu comportamento. Através dessas estruturas, é possível que um operador de rede estabeleça tabelas de encaminhamento, chaves de correspondência, ações personalizadas, entre outros.

Esse salto aumenta as possibilidades de redes *SDN* e prepara o ambiente para novas aplicações.

Alguns aplicativos, como balanceadores de cargas, utilizam inteligência no plano de dados para processamento de pacotes com estado (COX et al., 2017). Algumas soluções como OpenState (COX et al., 2017) oferece recursos semelhantes, porém herdam as limitações do OpenFlow. Oferecer um plano de dados altamente programável permite que ele faça a transferência das requisições ao servidor adequado sem a presença de um controlador. Antes do pipeline de dados programável, todas as funcionalidades de rede eram executadas por hardwares ou servidores dedicados com alto custo geral (high overall coast) e baixo desempenho (COX et al., 2017). O paper (MOLERO; VISSICCHIO; VANBEVER, 2018) observa que soluções modernas de hardware na indústria de rede executam operações complexas sem influenciar o rendimento e são altamente programáveis (COX et al., 2017). Esse avanço de hardwares de rede juntamente com o surgimento de chips de funções programáveis tão eficientes quanto chips de funções fixas (em termos de encami-

nhamento) (MCKEOWN; REXFORD, 2016) podem influenciar projetos de redes futuras (COX et al., 2017) (MOLERO; VISSICCHIO; VANBEVER, 2018). Aliada ao avanço de hardware, a programabilidade do plano de dados soluciona o problema de limitação de campos de correspondência do OpenFlow.

Recentemente, pesquisadores de rede estão utilizando *P4* para investigar algumas áreas importantes para a Internet tais como monitoramento de redes; detecção de DDoS; balanceamento de carga; agregação e desagregação de pacotes; aplicações de engenharia de tráfego; gerenciamento dinâmico de filas nos dispositivos de rede; entre outros (KAUR; KUMAR; AGGARWAL, 2021; COX et al., 2017). Por exemplo, (HE et al., 2018) propõe um mecanismo que remove entradas de fluxo quando existem flags de encerramento de sessão em conexões TCP ao invés de utilizar um contador de limite de tempo, que é o mecanismo padrão do OpenFlow. A utilização dessa inteligência no plano de dados otimiza a utilização da *TCAM* e diminui a sobrecarga do controlador.

Há uma necessidade de revisitar casos de uso como os citados; e investigar o impacto que a programabilidade do plano de dados ocasiona em diversos setores da Internet. Além do mais, a linguagem *P4* trouxe um novo olhar sobre a rede, é promissora e tem levantado enorme interesse à crescente demanda da indústria de telecomunicações (KAUR; KUMAR; AGGARWAL, 2021).

Um passo bastante promissor em direção à adoção generalizada do *P4* é a plataforma Aether (CASCONE, 2021). Essa plataforma reúne vários projetos *ONF* para criar uma solução de conectividade móvel privada 4G/5G. O plano de controle estende o ONOS (FOUNDATION, 2025b) para aproveitar as vantagens de uma arquitetura de microserviços (CASCONE, 2021); e, o plano de dados possui um roteador *P4* que lida com o tráfego que o usuário envia e recebe das estações base de rádio (radio base stations) (CASCONE, 2021). O roteador P4, na plataforma *Aether*, chama-se *User Plane Function (UPF)* e consegue atender aos requisitos de baixa latência e alta taxa de transferência que os casos de uso corporativos da Indústria 4.0 demandam. O *P4-UPF* está operacional e está implantado em todo o mundo como parte da rede Aether.

Ademais, intrinsecamente, a rede *SDN-P4* possui características de metarede, o que possibilita a absorção de funcionalidades de serviços tradicionais ou clean-slate. Por conta disso, as arquiteturas de Internet do Futuro estão migrando para a rede *SDN-P4*, tendo como justificativa principal a busca de uma rede onde possam coexistir com o IP. Muitos trabalhos seguem essa linha (KARRAKCHOU; SAMAAN; KARMOUCH, 2020; GUO et al., 2021; SIGNORELLO et al., 2016; FENG; TAN; JIN, 2019; GIMENEZ; GRASA; BUNCH, 2020), com diferentes níveis de implementação. (GIMENEZ; GRASA; BUNCH, 2020), por exemplo, faz uma implementação parcial da *RINA* (VRIJDERS et al., 2014) em redes *SDN-P4* com o objetivo de demonstrar a coexistência dessa *Future Internet Architecture (FIA)* com a arquitetura IP no mesmo ambiente de comunicação.

2.1.6 Visão geral da arquitetura *ETArch* e seus conceitos principais

As várias limitações da tecnologia Internet, algumas delas especificadas nas seções anteriores, levaram pesquisadores do mundo inteiro a pensar em novos modelos que fomentam o desenvolvimento de novas arquiteturas de Internet. Um desses modelos nasceu em 2009 e foi denominado Modelo de Títulos (PEREIRA; KOFUJI; ROSA, 2010).

Os principais objetivos desse modelo são: criar uma nova estratégia de endereçamento horizontal que resolva o problema de ambiguidade de identificação e localização; atenuar a necessidade de vários endereços, sendo que esse modelo prevê apenas um endereço unificado para comunicação entre as entidades; suporte semântico das necessidades de comunicação por uma camada de serviço; realizar injeção de requisitos novos nas camadas intermediárias sem que aumente a complexidade de operação e manutenção.

ETArch (SILVA et al., 2012) é a materialização do Modelo de Títulos. A proposta é que sejam resolvidos, naturalmente, pelas camadas inferiores, problemas que atualmente são solucionadas pela camada de aplicação da arquitetura Internet. *ETArch*, atualmente, oferece requisitos de Internet do Futuro, tais como: *multicast* (SILVA et al., 2012), mobilidade (GUIMARãES et al., 2014), segurança (DE, 2017) , *QoS* (LEMA et al., 2014), *QoE* (LEMA et al., 2014), dentre outros.

O roteamento das primitivas *ETArch* (NETO et al., 2015) não é realizado entre dois *hosts*, como na arquitetura Internet. Trata-se um roteamento orientado a *workspace*, que é um canal de comunicação lógico naturalmente *multicast*. A definição da rota considera os requisitos de todas as entidades que compõem ou estão anexadas a esse *workspace*, de tal forma que cada caminho configurado fornece os recursos de rede necessários para uma boa qualidade da comunicação.

O cálculo da rota é feito pelo controlador *SDN-OpenFlow* (NETO et al., 2015), que executa esse procedimento antes que a entidade de comunicação receba o conteúdo requisitado, ou seja, o comportamento do controlador, no caso do algoritmo de roteamento, é proativo. Dessa forma, algumas vantagens podem ser citadas: o algoritmo é flexível; não haverá custo de roteamento por parte dos elementos de rede, porque depois que o controlador calcula a rota de comunicação de cada uma das entidades do workspace, a função do elemento de rede resume-se apenas ao repasse das primitivas (NETO et al., 2015).

As subseções subsequentes tratam, brevemente, os aspectos conceituais da *ETArch*.

2.1.6.1 Camadas da *ETArch*

A Figura 9 exibe as camadas da ETArch, que apesar da abordagem *clean-slate*, apenas reexamina as camadas de rede e transporte da arquitetura Internet. A camada física e a camada de aplicação, que sofreram maiores evoluções com o crescimento em escala da

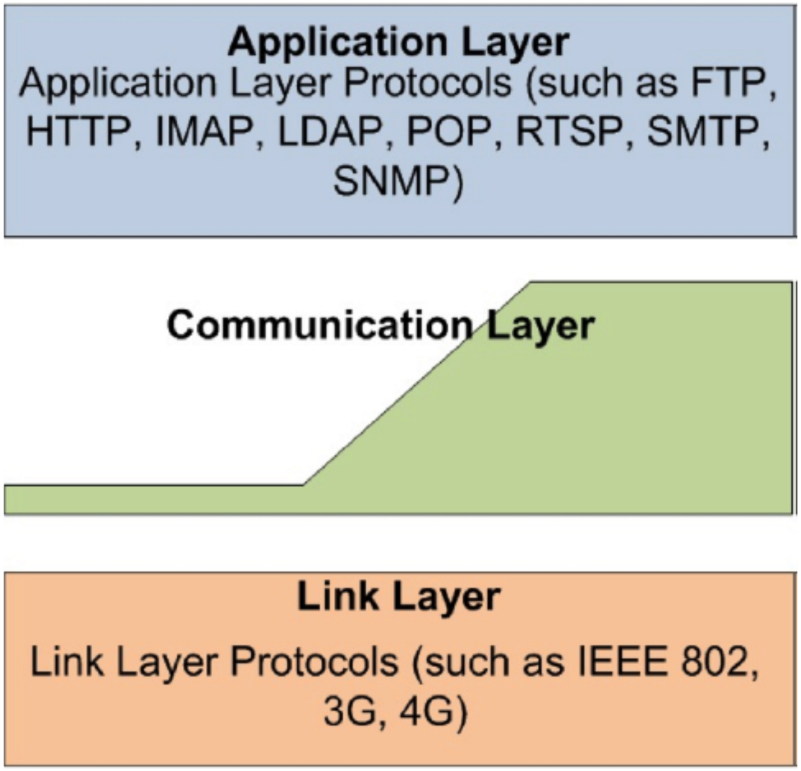

Figura 9 – Camadas da *ETArch*. Fonte: (SILVA; KOFUJI; ROSA, 2013).

Internet, são preservadas. Na arquitetura TCP/IP, a pilha de comunicação da arquitetura Internet depende das camadas de transporte (TCP/UDP/SCTP) e rede (IP). Na arquitetura *ETArch*, essas camadas são substituídas pela camada de comunicação que possui altura flexível.

O formato não usual da camada de comunicação indica a flexibilidade da arquitetura em se adaptar aos requisitos da aplicação. Aplicações com mais requisitos utilizam mais módulos e funções de rede, enquanto aplicações sem nenhum requisito, estabelece comunicação "quase" direta com a camada física.

2.1.6.2 Entidade

Entidade é o elemento fundamental da *ETArch*. Pode ser definido como qualquer coisa que tenha capacidade de se comunicar, como um *host*, *smartphone*, *Network Elemens (NEs)*. Uma entidade possui pelo menos um título e um conjunto de requisitos (PEREIRA; KOFUJI; ROSA, 2010).

Um requisito (SILVA; KOFUJI; ROSA, 2013) é uma demanda que a entidade necessita. Antes de iniciar a comunicação, uma entidade passa às camadas inferiores os requisitos que precisa para que a experiência do usuário seja satisfatória. A rede deve ser capaz de atender os requisitos dessa comunicação.

Capacidades (SILVA; KOFUJI; ROSA, 2013) têm os mesmos elementos que os requisitos, mudando a maneira como são manipulados.

Requisitos são vistos como um recurso que uma entidade solicita da rede para seu bom funcionamento, enquanto capacidade são recursos que a rede oferece para um contexto de comunicação.

2.1.6.3 Título

Título é definido como um identificador único, não ambíguo e independente da topologia da rede subjacente (ROBERTS, 2009). Uma entidade possui um ou mais títulos, dependendo da necessidade. Na prática, qualquer entidade comunicante possui pelo menos um nome válido. O identificador da entidade é independente da sua localização, facilitando, dessa forma, sua mobilidade. O título tem papel fundamental na implementação do endereçamento horizontal.

Nessa arquitetura, o endereço de destino (localização das entidades) das primitivas de dados é o título do *workspace*. Os switches reconhecem o título e replicam a primitiva para todas as entidades da comunicação.

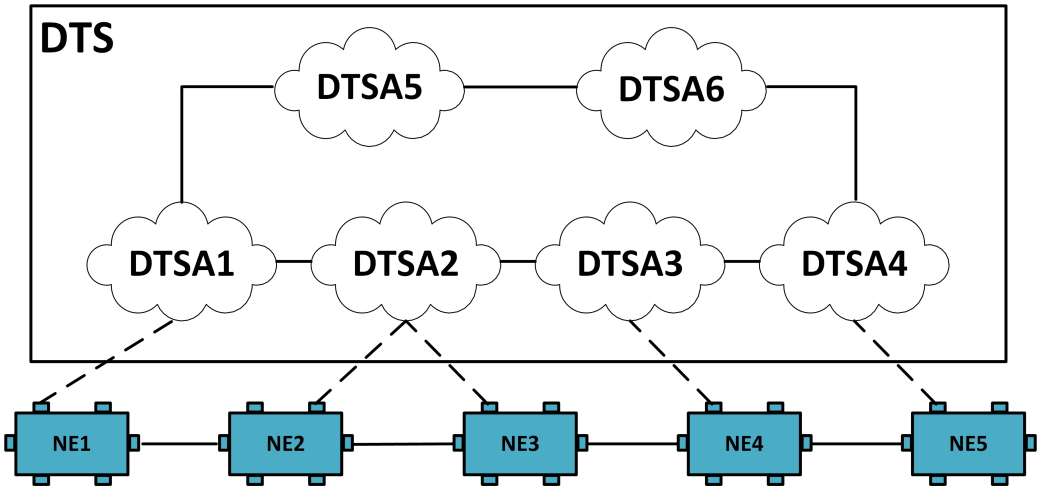

Figura 10 – Representação do *DTS*

2.1.6.4 *DTS*

O *DTS* (SILVA; KOFUJI; ROSA, 2013) representa o conjunto universo de todo o plano de controle existente na arquitetura *ETArch*. Dentro desse conjunto, há vários agentes, denominados *Domain Title Service Agents (DTSAs)*, que viabilizam a divisão da rede em subconjuntos gerenciáveis. Cada um desses subconjuntos pode ser visto como um domínio distinto da rede, cada qual gerenciado por agentes do *DTS*. O *DTS*, como conjunto universo ou domínio maior, tem o papel de gerenciar todos os subdomínios, realizando dessa forma, a orquestração total das funcionalidades de controle.

Dentre essas funções, podemos citar: resolução de títulos de *workspace* para viabilizar a comunicação multicast; atribuição de títulos para as entidades comunicantes; gerencia-

mento do ciclo de vida de todas as entidades pertencentes ao *DTS*; cálculo da melhor rota para a extensão do *workspace*; oferecer qualidade de serviço para as entidades comunicantes. O *DTS* é um ambiente distribuído que gerencia todas as funcionalidades de controle e torna possível a realização da comunicação entre entidades na arquitetura *ETArch*.

A Figura 10 representa o conjunto universo do *DTS*. A lista de agentes (DTSA1 à DTSA6) constitui os elementos do plano de controle. A lista de elementos de rede (NE1 à NE5) representa a infraestrutura de comunicação do plano de dados. O plano de controle é a inteligência que gerencia os elementos de rede: fornece *workspace* para as entidades comunicantes e gerencia o ciclo de vida de cada um desses canais de comunicação. O plano de dados apenas faz o encaminhamento das primitivas para as entidades que participam da comunicação.

2.1.6.5 *Workspace*

Workspace (SILVA; KOFUJI; ROSA, 2013) é um conceito que permite que duas ou mais entidades troquem primitivas em uma instância de comunicação específica. Ele é um barramento lógico, independente da topologia e natureza da rede subjacente, no qual entidades podem se conectar para consumir o recurso fornecido por esse contexto de comunicação. Quanto aos recursos, cita-se como exemplo, streaming de vídeo ou tráfego de textos de aplicações de chat. Quanto ao contexto de comunicação, entende-se que o *workspace* consegue fornecer a capacidade de rede necessária para cada caso de uso específico.

O plano de controle utiliza um tipo específico de *workspace*, denominado *workspace* de controle, onde são transmitidos apenas primitivas relacionadas ao controle da rede. Essa comunicação especial de controle é estabelecida entre entidades e *DTSAs* através do protocolo *ETCP* ou entre *DTSAs* através do protocolo *Domain Title Service Control Protocol (DTSCP)*.

No caso do *workspace* de dados, criado pelo plano de controle para prover comunicação entre as entidades, os recursos desse *workspace* são consumidos apenas por entidades anexadas a esse canal de comunicação.

A Figura 11 exibe um exemplo de topologia da arquitetura *ETArch*. O *DTS* é o conjunto universo que detém três domínios maiores, controlados pelo MASTERDTSA1, MASTERDTSA2 e MASTERDTSA3. O MASTERDTSA1, por sua vez, possui 3 subdomínios, compostos pelo DTSA1, DTSA2 e DTSA3. O MASTERDTSA2 possui dois subdomínios, compostos por DTSA4 e DTSA5. O MASTERDTSA3 não possui subdomínio.

O *DTS* é o espaço formado por todos esses domínios/subdomínios e é responsável por gerenciá-los corretamente a fim de manter a comunicação de todas as entidades envolvidas. Para auxiliá-lo, ele possui duas entidades especiais: o *Domain Title Service Agent (DTSA)* e o *Domain Title Service Agent (MDTSA)*. Esses dois agentes mantêm o

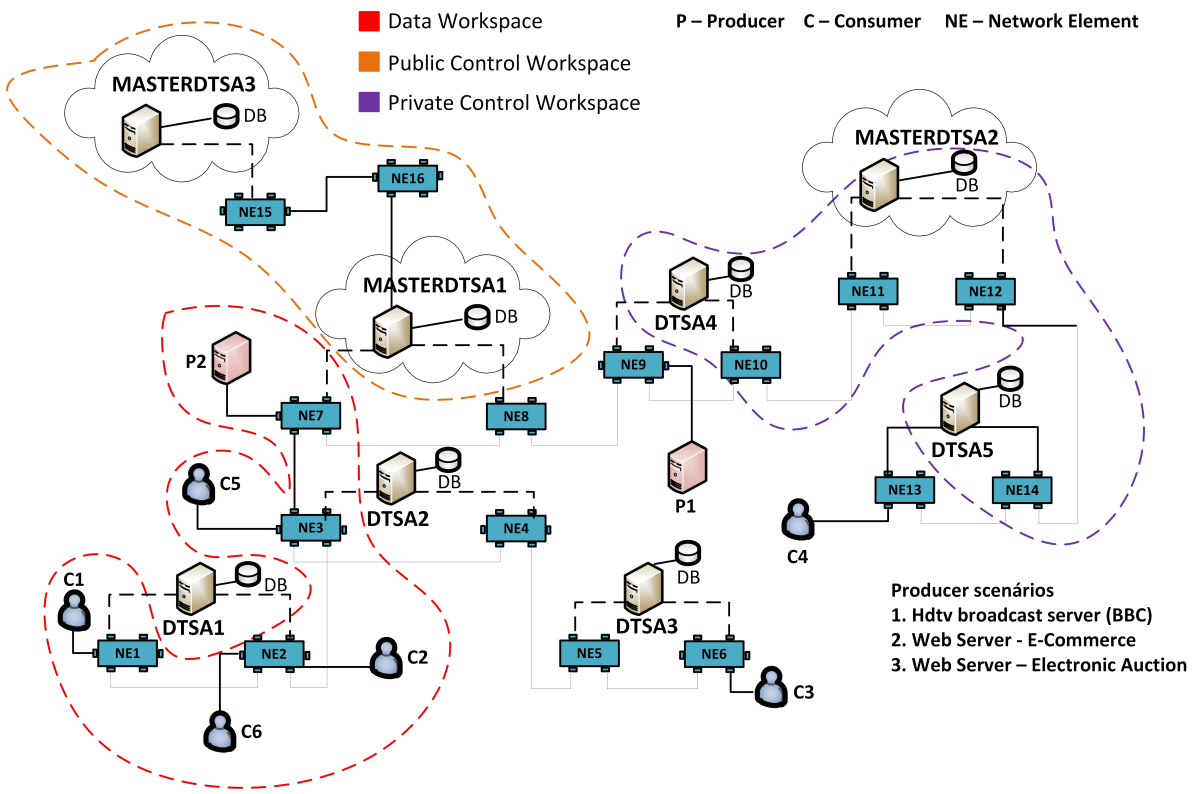

Figura 11 – Um exemplo de topologia *ETArch*.

gerenciamento dos três *workspace*s representados na figura: *workspace* de dados, tracejada em vermelho; *workspace* de controle público, tracejada em laranja; *workspace* de controle privado, tracejado em roxo.

O termo 'Público' na nomeação do *workspace* não quer dizer acesso irrestrito, mas tem analogia com as redes Carriers em telecomunicações, onde esta característica é empregada no suporte de interações de controle inter-domínios.

Quando uma entidade quer se anexar ao *workspace*, ela requisita o 'attach' através de uma primitiva de controle, utilizando para isso o título do *workspace*. Se o processo de solicitação for realizado com sucesso, a entidade começa a receber os serviços disponibilizados pelo *workspace*. Todos os consumidores do *workspace* de dados da Figura 11 (C1, C2, C5, C6) recebem os serviços disponibilizados pelo produtor P2, que pode ser por exemplo, uma transmissão de vídeo HDTV.

2.1.6.6 *DTSA*

O *DTSA* é uma extensão de um controlador *SDN-OpenFlow* (SILVA; KOFUJI; ROSA, 2013). Ele possui as funcionalidades de controle da arquitetura *ETArch*, e através delas, gerencia o ciclo de vida das entidades que pertencem ao seu domínio. Na Figura 11: C1, C2, C6, NE1 e NE2 são entidades que pertencem ao domínio do DTSA1. Por este motivo, o DTSA1 conhece a topologia de seu domínio, os *workspace*s criados por suas entidades ou estendidos de outros domínios. P2 cria o *workspace* de dados no domínio

do MASTERDTSA1, que também desempenha, nesse caso, o papel de *DTSA*. Esse *workspace* também está presente em outros dois domínios: os domínios do DTSA2 e do DTSA1. Dessa forma, tanto o DTSA1 quanto o DTSA2 possuem em seus registros informações desse *workspace* de dados criado por P2. Dizemos que essas informações são de um *workspace* estendido, pois não foi criado nem pelo DTSA1, nem pelo DTSA2, e sim, foi estendido do domínio pertencente ao MDTSA1.

Um exemplo concreto de funcionalidade de um *DTSA* é o cálculo de rotas para extensão do *workspace*. Quando uma entidade quer entrar na comunicação, ela requisita essa operação através de uma primitiva de controle. C2 faz a requisição, o DTSA1 recebe essa primitiva de controle, registra essa entidade em seu repositório, e logo depois, estende o *workspace* até C2, como mostrado na figura. Nesse caso, o DTSA1 já detinha a informação desse *workspace* em seus registros, pois já o havia estendido anteriormente para C1 e C6. Dessa forma, ele mesmo (DTSA1) estende o *workspace* através de um algoritmo de roteamento intra-domínio. Caso o DTSA1 não reconheça o *workspace* de dados solicitado por C2, o processo de extensão é outro; mas esse caso não faz parte do escopo desse trabalho. Os experimentos realizados, neste trabalho, referentes à arquitetura *ETArch*, consideram apenas 1 domínio de ação.

2.1.6.7 *MDTSA*

O *MDTSA* (SILVA; KOFUJI; ROSA, 2013) é uma extensão do DTSA, ou seja, um *MDTSA* pode desempenhar função de DTSA, mas o contrário não se consuma. Esses dois agentes desempenham apenas funções de controle e preparam a rede para o plano de dados. A diferença crucial é que o *MDTSA* controla domínios de *DTSAs*, portanto, possui uma visão mais global da rede. Dessa forma, o *MDTSA* tem que externalizar solicitações de controle interna e internalizar solicitações de controle externa, gerenciando, dessa forma, uma comunicação entre domínios.

2.1.6.8 Protocolos de controle da ETArch

O *DTSA* e o *MDTSA* executam suas operações através dos protocolos de controle da ETArch: *ETCP* e *DTSCP* (SILVA; KOFUJI; ROSA, 2013), respectivamente. O primeiro faz a comunicação entre entidades e *DTSAs*, enquanto o segundo realiza a comunicação entre *DTSAs*.

O protocolo *DTSCP* não faz parte do escopo desse trabalho, já que os casos de uso que envolvem a arquitetura ETArch são realizadas apenas em um único domínio.

Segue abaixo, na Tabela 1, os serviços do protocolo *ETCP* (SILVA; KOFUJI; ROSA, 2013).

Tabela 1 – *API* do protocolo *ETCP*.

Método	Descrição
register	Registra a entidade no *DTSA* do seu domínio.
unregister	Remove o registro de uma entidade no *DTSA* do seu domínio.
createWorkspace	Cria um *workspace* com a finalidade de oferecer um recurso.
attachWorkspace	Vincula uma entidade a um *workspace* existente.
detachWorkspace	Desvincula uma entidade do workspace correspondente.
deleteWorkspace	Deleta um workspace e todos os registros de vinculação de suas entidades.
send	Envia o recurso para todas as entidades que estão vinculadas ao workspace.
receive	Recebe o recurso do *workspace*.

2.1.7 Visão geral da arquitetura NovaGenesis e seus conceitos principais

NovaGenesis (NG) é uma arquitetura *clean-slate* que se preocupa principalmente em flexibilizar a pilha de rede para integrar sem maiores desgastes as funcionalidades das aplicações atuais e futuras (ALBERTI et al., 2019). Por conta disso, possui uma infraestrutura auto-organizada, que permite alocação de recursos dinâmicos por contexto de comunicação (ALBERTI et al., 2018).

O *NG* segue as premissas do *Service-Oriented Architecture (SOA)* e enxerga a rede como um conglomerado de serviços que se comunicam a partir da demanda e oferta de recursos. Cada serviço, no *NG*, é uma entidade virtual capaz de solicitar ou executar determinadas ações tais como processamento, troca ou armazenamento de informações (ALBERTI et al., 2019). Dessa forma, a entidade que tem a capacidade de executar uma tarefa deve oferecê-la à rede. A entidade que possui uma demanda, deve tentar descobrir se alguma outra entidade de rede provê o recurso, e em caso positivo; deve solicitá-lo.

Assim sendo, *NG* promove um ambiente distribuído de comunicação entre serviços que produzem e consomem recursos da rede. Cabe ao *NG* orquestrar essa comunicação e promover o encontro entre essas entidades. Para isso, possui serviços padrões de controle que auxiliam na descoberta de recursos através de associações de nomes (*Name Bindings (NBs)*); negociam acordos de comunicação (*SLA*) que dependem do contexto; possibilitam o roteamento, orientado a nome, das primitivas da arquitetura; registram publicações e assinaturas em Tabelas de Hash Distribuídas (Distributed Hash Tables (DHTs)) com intuito de coordenar o ciclo de vida de entidades e comunicações *NG*; entre outros (SILVA, 2021).

Um dos conceitos importantes da arquitetura diz respeito à nomeação das entidades comunicantes. A nomenclatura é genérica e permite a criação de qualquer nome possível em qualquer convenção, seja ela hierárquica ou plana. Associação de nomes (ALBERTI et al., 2019; SILVA, 2021) é um mapeamento, em forma de vetor (tuplas), entre dois ou mais nomes. Através dos *NBs*, *NG* define o identificador e a localização de uma determinada

entidade comunicante. O identificador de uma entidade dentro de um escopo/domínio é único e representa um nome exclusivo. O localizador é importante para os mecanismos de roteamento; e, por isso, o *NG* vincula o identificador da entidade à sua localização. Vale lembrar que o identificador e a localização representam conceitos distintos, e portanto possuem nomeações diferentes. Isso permite que uma entidade mude de localização sem que seja necessário a modificação de seu identificador. A tupla <Processo-07, Domínio-01, Subrede-01> representa que a entidade de identificador "Processo-07" localiza-se no *namespace* "Subrede-01.Dominio-01". Outra interpretação poderia ser dada a essa tupla: a entidade de identificador "Dominio-01.Processo-07" localiza-se no *namespace* "Subrede-01". No *NG*, as interpretações e as composições de *NBs* são ilimitadas. O conceito de nomeações é importante porque tanto as publicações quanto as assinaturas de recursos de rede são feitas através de *NBs*.

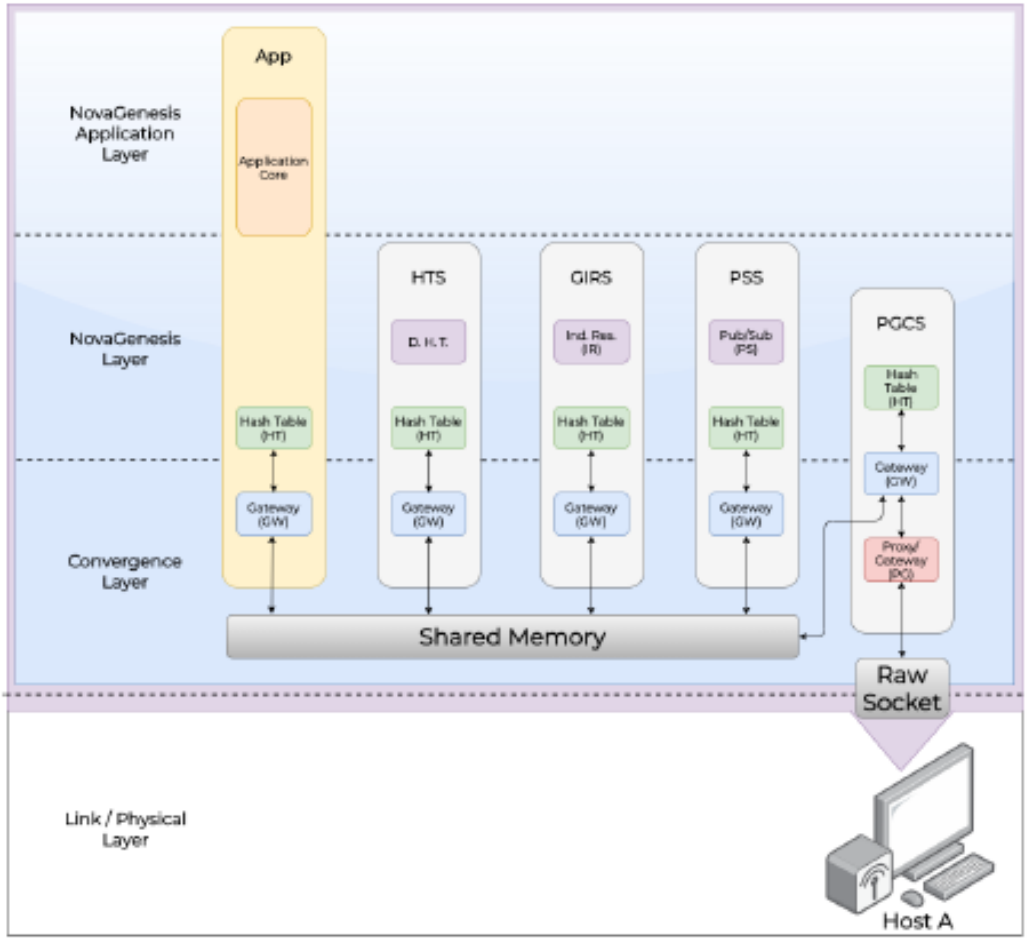

Figura 12 – Modelo de camadas de NovaGenesis com seus principais componentes e processos. Fonte: (TELECOMUNICAçõES, 2021b).

A Figura 12 apresenta um *host NG*. Este *host* possui os principais componentes e serviços da arquitetura *NG*: *Hash Table Service (HTS)*, *Generic Indirection Resolution System (GIRS)*, *Publish/Subscribe Service (PSS)* e *Proxy/Gateway Controller Service (PGCS)*. Cada um desses serviços possuem um conjunto de blocos funcionais.

Um exemplo é o serviço PGCS, que possui três blocos funcionais: *Hash Table (HT)*, *Gateway (GW)* e *Proxy/Gateway (PG)*. A arquitetura prevê três camadas genéricas que abrangem todos os serviços *NG*: Camada de Convergência, Camada NovaGenesis e Camada de aplicação. Essas camadas são abstratas e cada uma possui suas próprias metas. Outra informação da figura é que cada serviço ultrapassa as funcionalidades de uma única camada. *PGCS*, por exemplo, possui funcionalidades da Camada de Convergência e da Camada NovaGenesis; mais especificamente, os blocos funcionais GW e PG executam funções provenientes da Camada de Convergência, já o bloco HT executa funções que pertencem à Camada NovaGenesis. Neste *host*, em especial, a camada de enlace utiliza a interface *Raw Socket* para encaminhar um frame Ethernet para o meio físico. A memória compartilhada(*Shared Memory (SHM)*) é um mecanismo disponível nos sistemas operacionais que permitem a comunicação entre processos através da memória RAM do *host*. *NG* utiliza esse mecanismo para realizar troca de mensagens entre processos que estão sendo executados dentro do mesmo sistema operacional (ALBERTI et al., 2019). Em relação às mensagens trocadas entre os blocos funcionais que estão dentro do mesmo componente, o *NG* utiliza mecanismos relacionados a eventos e callback de ações que o processo realiza (SILVA, 2021).

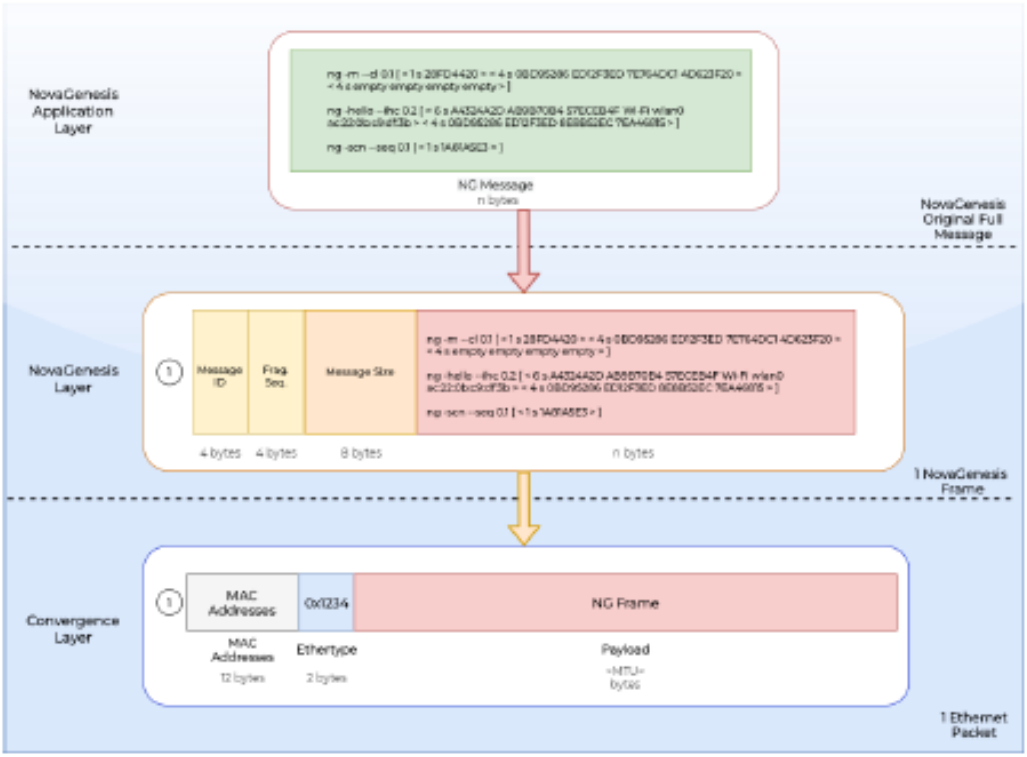

Figura 13 – Exemplo de encapsulamento de uma mensagem do NovaGenesis. Fonte: (TELECOMUNICAçõES, 2021b).

A Figura 13 (SILVA, 2021) apresenta as três camadas do *NG* trabalhando juntas no envio de uma mensagem à rede. Um aplicativo *NG*, da camada de aplicação, envia

uma mensagem de controle para rede. As mensagens de controle tem o objetivo de requisitar ou publicar recursos; e, de estabelecer a conexão entre os pares. No caso da figura, a mensagem de controle possui linhas de comandos (ações) que visam configurar o comportamento das entidades comunicantes.

Como exemplo de ação, a linha de comando "hello"(*NG* -hello ...) configura se o tipo de comunicação é *intra-host* ou *inter-host*; expõe os nomes de autoverificação (*Self-Verifying Name (SVN)*) de componentes internos do *PGCS* para a outra entidade comunicante; define a camada de enlace da comunicação (*Ethernet*, *Wi-Fi*, etc.); informa o endereço MAC da placa; entre outros (SILVA, 2021). De um modo resumido, a ação *Hello* negocia os parâmetros da comunicação.

A linha de comando de encaminhamento/roteamento (*NG* -m -cl ...) passa informações importantes ao par comunicante, tais como o identificador do host(HID), do sistema operacional(OSID), do processo(PID) e do bloco (BID) onde o serviço está sendo executado. Essas informações compõem o endereço de origem (do remetente) e destino (do receptor) das entidades que participam da comunicação; e por conta disso, são indispensáveis para o mecanismo de roteamento das primitivas (SILVA, 2021).

A primitiva de controle, que contém as ações, é enviada à *Camada Nova Genesis (CNG)*. *CNG* encapsula a mensagem da aplicação em uma ou mais primitivas *NG*. Para isso, cria três cabeçalhos de controle: Identificador da Mensagem, Sequência de Fragmentação e tamanho da mensagem (SILVA, 2021). A Figura 13 exibe o formato da primitiva *NG* (SILVA, 2021).

. Identificador da Mensagem (4 bytes): é um número que representa o identificador de uma mensagem *NG*. No caso da mensagem ultrapassar a Unidade Máxima de Transmissão (*Maximum Transmission Unit (MTU)*), *NG* fragmenta a mensagem em várias primitivas que possuem o mesmo identificador. Supondo que a mensagem seja enviada em três primitivas diferentes, cada uma das três primitivas terão o mesmo identificador. O identificador é exclusivo da mensagem, não da primitiva.

. Sequência da fragmentação (4 bytes): é um número que representa a sequência dos fragmentos de uma mensagem. *NG* pode fragmentar uma mensagem em várias primitivas caso o tamanho dessa mensagem ultrapasse o *MTU* da camada de enlace. Caso isso aconteça, cada uma das primitivas que carrega a mensagem tem um número de sequência diferente. A primeira primitiva tem valor zero, a segunda valor um, a terceira valor dois, e assim por diante. Essa sequência é importante nos sistemas finais, onde é feita a remontagem da mensagem *NG* fragmentada.

. Tamanho da mensagem(8 bytes): é um número que representa o tamanho da mensagem *NG* que a aplicação está passando para a rede. Essa informação é importante por dois motivos: primeiramente, as primitivas *NG* tem tamanho variável e esse campo auxilia na extração do payload; em segundo lugar, é um campo fundamental no processo de remontagem de mensagens fragmentadas, já que aponta o tamanho dessas mensagens.

A partir dessa informação, o *NG* consegue perceber se todas as primitivas de uma mensagem fragmentada chegaram ao ponto final (*endpoint*), e a partir daí, remonta a mensagem utilizando os dois campos citados anteriormente: Identificador da Mensagem e Sequência de Fragmentação.

Voltando à Figura 13 (SILVA, 2021), a *CNG* cria a primitiva *NG* e envia-a para a Camada de Convergência (CDC). A partir daí, CDC encapsula a primitiva *NG* na tecnologia de enlace solicitada pela aplicação através da ação "Hello". Um dos parâmetros que a ação "Hello" negocia com o par comunicante é a tecnologia de enlace da comunicação. No caso da figura, a aplicação solicita uma conexão *Ethernet*. *CDC*, então, encapsula a primitiva *NG* em um quadro *Ethernet* e envia-a para rede através da interface *Raw Socket*.

As funcionalidades de cada camada, componente e bloco funcional do *NG* são apresentas abaixo. A Figura 12 mostra como os componentes e os blocos funcionais padrões da arquitetura estão acomodados nas camadas abstratas do *NG*.

. Camada de convergência (CDC): *Raw Socket* é uma interface que recebe os dados diretamente da camada de enlace (*layer link*) e repassa-os à Camada de Convergência. Cada entidade *NG* possui um *PGCS* e cada *PGCS* representa um único domínio (SILVA, 2021). Nesse contexto, a Camada de Convergência tem o papel de adaptar mensagens *NG* que uma aplicação envia para tecnologias de enlace tais como Ethernet, Wi-Fi, ZigBee; e vice-versa (SILVA, 2021). Através dessa camada, os serviços/processos internos existentes dentro de um domínio comunicam-se entre si através de uma memória compartilhada (Shared Memory); como também, pode haver comunicação entre serviços que estão em domínios diferentes. No caso das figuras 12 e 13, a CDC encapsula a primitiva *NG* em um frame Ethernet através da API Raw Socket para realizar uma comunicação interdomínio(a opção –ihc do comando *hello* indica *inter-host communication*); e, também recebe mensagens das tecnologias de enlace (no caso, Ethernet) e adapta-as para o próximo bloco do *PGCS* (no caso, o Gateway). A comunicação entre blocos funcionais dentro de um mesmo serviço também é parte integrante das funcionalidades dessa camada. Como exemplo, a figura exibe a comunicação entre o Proxy/Gateway(PG) e Gateway(GW). Essa comunicação interna utiliza mecanismos de *callback* orientados à eventos de ações de serviço (SILVA, 2021).

. Camada NovaGenesis(C*NG*): essa camada encapsula os dados ou requisições de controle de aplicações *NG* em primitivas *NG* para, posteriormente, enviá-las à rede. O inverso também é válido, ou seja, desencapsula as mensagens de controle ou os dados das primitivas *NG* e entrega-os ao serviço/aplicação correspondente. No processo de encapsulamento, a camada *NG* elabora primitivas com cabeçalhos de controle que vão permitir, futuramente, a execução de alguns recursos da arquitetura tais como identificação, fragmentação e remontagem de mensagens *NG* (ver Figura 13)(SILVA, 2021). Essas primitivas podem ser de controle, e por conta disso carregam em seu payload ações necessárias (linhas de comando) que configuram o comportamento das entidades comunicantes.

A primitiva também pode ser de dados. Nesse caso, elas carregam em seus payloads dados da aplicação. CNG permite a comunicação entre blocos funcionais que situam-se em um mesmo serviço. Por exemplo, o Gateway(GW) e Hash Table (HT) do serviço *PGCS* comunicam-se entre si nessa camada (Ver Figura 12). Essa comunicação interna utiliza mecanismos de callback orientados à eventos de ações de serviço (SILVA, 2021).

. Camada de Aplicação: os aplicativos *NG*, que estão na camada de aplicação, expõem os serviços que oferece ou requisita serviços da rede a partir da publicação ou consulta de NBs. NBs podem representar as entidades (serviços virtuais) comunicantes que ofertam ou requisitam recursos da rede, as localizações dessas entidades, os identificadores dos componentes internos que fazem a comunicação (componentes do *PGCS* da aplicação), etc. As informações são publicadas por editores (*publishers*) e posteriormente são consultadas por assinantes (*subscribers*) através de um serviço de resolução de nomes que é crucial para o funcionamento da infraestrutura. Tanto as publicações quanto as consultas são feitas através da *API PSS*. A partir dessa API, os aplicativos também podem assinar, através de NBs, recursos oferecidos por outros aplicativos.

. Proxy/Gateway (PG): bloco funcional do *PGCS* que implementa o encaminhamento de mensagens para outros *PGCS*s pares que se encontram em outros domínios (ALBERTI et al., 2019). Nesse sentido, PG realiza a comunicação entre domínios na arquitetura *NG*. Além disso, como remetente, encapsula a primitiva *NG* na tecnologia de enlace que a aplicação requisita antes de enviá-la à rede; como receptor, faz o contrário; desencapsula a primitiva *NG* de acordo com a tecnologia de enlace da comunicação e envia-a para o Gateway (GW). Sockets são as interfaces que possibilitam o encapsulamento e desencapsulamento dessas primitivas.

. Gateway (GW): bloco funcional do *PGCS* que trabalha principalmente como um gateway de aplicação. Ele realiza comunicação entre serviços que estão no mesmo domínio através de uma memória compartilhada (Shared Memory). Este componente é responsável direto pelo encapsulamento das requisições de controle ou dados que chegam das aplicações em primitivas *NG* e, também, pelo desencapsulamento de mensagens *NG* que chegam da rede a partir do PG. Esses dados brutos são identificados e colocados em um buffer de memória para um serviço específico. A Figura 12 (SILVA, 2021) mostra que os serviços se comunicam através desse componente. Um exemplo é que o *PGCS* pode se comunicar com a aplicação (*Application core*) ou com o *PSS* através da memória compartilhada. Nesses casos, o *GW* do *PGCS* disponibiliza informação para o GW do PSS ou GW da aplicação; e vice-versa. O gateway de cada um dos serviços intermedia a comunicação dentro do domínio (*intra-domain*). Além disso, a figura também mostra que todos os blocos funcionais *GW* comunicam-se internamente com outros blocos funcionais internos denominados Hash Table (HT). Além da comunicação entre serviços, GW também realiza comunicação interna através de mecanismos de *callback* ativados por eventos de ações dos próprios serviços (SILVA, 2021).

. Hash Table(HT): bloco funcional que tem a responsabilidade de armazenar os nomes de autoverificação (*Self-Verifying Names (SVNs)*) e os nomes de linguagem natural (*Natural Language Namings (NLNs)*) das entidades comunicantes; como também os NBs que representam a vinculação desses nomes a outros nomes com o intuito de representar localizações e identificações dessas entidades. Eles podem armazenar, por exemplo, os SVNs de cada bloco interno dos serviços que estão sendo executados em um domínio específico. Dessa forma, esses registros colaboram, por exemplo, com o bloco funcional GW de cada serviço, que encaminha mensagens dentro do domínio (ALBERTI et al., 2019; SILVA, 2021).

. Proxy/gateway/controller service (PGCS) (ALBERTI et al., 2019; SILVA, 2021): este serviço engloba várias funcionalidades. Como proxy/gateway, fornece a comunicação interdomínio entre entidades *NG* e atua como um representante das aplicações que participam da rede *NG*. Isso quer dizer que o *PGCS* é capaz de receber uma primitiva de uma aplicação *NG*, reconhecê-la e enviá-la ao seu destino. Esse recurso de comunicação possibilita a criação de um ecossistema de serviço inteligente, onde os pares de entidades comunicantes podem expor seus serviços como também consumir recursos da rede. *PGCS* encapsula mensagens de aplicações *NG* em quadros (frames) de tecnologias da camada de enlace conhecidas tais como Wi-Fi, Ethernet, ZigBee, etc. e envia-as à rede (SILVA, 2021). Quando a mensagem *NG* chega ao destino, *PGCS* realiza o processo inverso: desencapsula as mensagens *NG* das tecnologias de enlace citadas e envia-as ao bloco funcional GW (ver Figura 12). Como Gateway, oferece a comunicação intradomínio (*intra-domain*) entre processos/serviços que são executados dentro do mesmo SO através de uma memória compartilhada. Cada serviço possui um GW, que intermedia a comunicação intradomínio entre os serviços (aplicações, *HTS*, *GIRS*, *PSS* e *PGCS*). Nesse ponto, o GW recebe a mensagem *NG* do serviço, reconhece-a e envia-a para o destino adequado (dentro do domínio). Para realizar essa comunicação, os *Gateways (GWs)* utilizam os IDs padrões da *NG* tais como identificadores de processos, sistema operacional, *host*, entre outros. Nessa fase de comunicação, o *GW* também realiza o processo de fragmentação/desfragmentação de mensagens que extrapolam a *MTU* da primitiva de enlace. Como controlador, o *PGCS* gerencia os recursos físicos e virtuais programáveis (SILVA, 2021). Algumas metas do *PGCS* corroboram essa afirmação: esse serviço orquestra o ciclo de vida das entidades e da comunicação; gerencia credenciais de acesso e os termos de acordo da comunicação através de *Service Level Agreements (SLAs)*; pode ser visto como um serviço de roteamento de software que coordena o envio e recebimento de mensagens *NG*; inicializa os componentes de um domínio, o que permite a descoberta e publicação dos serviços oferecidos.

. Resolução de nomes e serviços de cache de rede (*Name Resolution and Networking Cache System (NRNCS)*): este serviço implementa cache de rede de conteúdo através de resolução de associação de nomes (*name bindings*); promovendo a descentralização do

conteúdo e aumentando a performance da troca de dados ao diminuir a distância entre o assinante e o recurso solicitado. Ele possui uma tabela de *hash* distribuída (*Distributed Hash Table (DHT)*) que armazena *NBs* publicadas e assinadas. Cada NB publicada possui uma lista de serviços autorizados a assinar essas *NBs*; portanto, só serviços autorizados podem acessar os conteúdos armazenados. Quando um serviço inicializa, ele publica NBs que associam identificadores às localizações (*NLNs* e *SVNs*), expondo seus serviços a outros serviços da rede.

NRNCS engloba outros três serviços descritos abaixo:

. Publicação/Assinatura (*PSS*): Os serviços acessam essa API por meio dos seus identificadores. A sua principal finalidade é fornecer uma interface que descobre, acessa e publica serviços de acordo com suas NBs (identificações e localizações de entidades NG) (ALBERTI et al., 2019). Para realizar essas ações, o PSS movimenta (insere, pesquisa, deleta) um repositório distribuído de NBs através de um serviço denominado Hash Table Service (HTS). Esse repositório é imprescindível para o gerenciamento dos recursos da rede. A partir do PSS, é possível publicar NBs (com dados associados, se houver); publicar NBs/dados e notificar os recursos publicados a outros serviços; assinar NBs/dados; assinar NBs/dados e notificar essa assinatura a outros serviços; entregar conteúdo assinado para a entidade adequada; revogar publicação de NB/dados (ALBERTI et al., 2019); etc.

. Hash Table Service (*HTS*): é um serviço que implementa um sistema *DHT* baseado em nomes de autoverificação (*Service-Centric Networking (SCN)*) (ALBERTI et al., 2019; TELECOMUNICAçõES, 2021a). Dessa forma, *HTS* armazena *NBs* e conteúdos relacionados de forma distribuída, o que garante desempenho adequado na execução de pesquisas referentes a recursos de rede (ALBERTI et al., 2019).

. Serviço Genérico de Resolução Indireta (*GIRS*): este serviço intermedeia a comunicação entre *PSS* e *HTS* no *NRNCS*. Ele calcula, a partir da chave NB, em qual instância *HTS* o *Name Binding (NB)* vai ser armazenado. A partir do resultado, *GIRS* seleciona a instância *HTS* adequada e armazena a informação (ALBERTI et al., 2019). O *GIRS* fornece a interface necessária ao *PSS* tanto para armazenar os valores-chave das *NBs* quanto para recuperar *NBs* e conteúdos já publicados (CASCONE, 2021).

2.1.8 Uma breve descrição de outras arquiteturas de Internet do Futuro

Os princípios fundamentais da *eXpressive Internet Architecture (XIA)* é a capacidade de evoluir seus paradigmas de comunicação e criar mecanismos para que a interação entre entidades comunicantes seja segura (ANAND et al., 2011). Quanto à evolução dos paradigmas de comunicação, ele deve ser capaz de englobar funcionalidades diferentes de comunicação, como por exemplo, de redes centradas em informação (*Information-Centric Networking (ICN)*) e de redes centradas em serviço (*SCN*). O intuito é que a

arquitetura seja flexível e que a rede *XIA* possa oferecer suporte nativo para quaisquer tipos de comunicação existentes ou que venha a existir devido a demandas futuras. Em relação à segurança, *XIA* estabelece que cada comunicação seja intrinsecamente segura; ou seja, as entidades participantes são capazes de validar a comunicação sem precisar acessar informações ou configurações externas (ANAND et al., 2011). A lógica que define a comunicação da *XIA* parte de dois componentes: o *eXpressive Internet Protocol (XIP)* e os *Principals* (HAN et al., 2012).

ZFilter (JOKELA et al., 2009) é uma arquitetura *multicast* que fundamenta sua comunicação na extensão do paradigma *Publish/Subscribe*. Baseia-se em uma abordagem recursiva, onde as camadas da arquitetura (*Rendezvous*, *Topology* e *Forwarding*) são aplicadas recursivamente no topo de si mesmas e onde as funcionalidades das camadas superiores utilizam as funcionalidades das camadas inferiores. Isso tem um efeito colateral: a primitiva *ZFilter* pode, em alguns casos, possuir mais de um cabeçalho; onde cada cabeçalho fornece uma abstração completa de conectividade (JOKELA et al., 2009). Um dos casos onde isso acontece é na implementação de encaminhamento inter-domínios (inter-domains). Em relação ao mecanismo de encaminhamento, baseia-se na identificação de links ao invés de nós. O endereço de destino, que é uma árvore unicast/*multicast* de links físicos ou virtuais (JOKELA et al., 2009), descreve o caminho da primitiva *ZFilter* até o(s) *endpoint*(s). Dessa forma, o encaminhamento de primitivas se dá no estilo de roteamento na origem (*source routing*), permitindo encaminhamento de primitivas sem dependência de endereçamento fim a fim.

O *Named Data Networking (NDN)* é uma arquitetura de rede baseada em *ICN* (SAXENA et al., 2016). A rede *NDN* deve ser escalável, fornecendo suporte aos serviços da Internet a uma grande quantidade de nomenclaturas de nomes; deve ser resiliente, detectando falhas na entrega de pacotes e recuperando-os a uma taxa de transferência aceitável; deve ser eficiente, suportando encaminhamento de primitivas para vários caminhos e cache de conteúdo para disseminação eficiente de dados; deve ser seguro, oferecendo mecanismos de integridade, autenticação de origem, entre outros (SAXENA et al., 2016). Quanto ao encaminhamento, existem dois formatos de primitivas (YI et al., 2013): Interesse (Ipkt) e dados (Dpkt). A comunicação é iniciada pelo consumidor (quem requisita o conteúdo) através de um Ipkt. O dado mais importante do Ipkt é o nome do conteúdo que deve ser exclusivo no escopo da comunicação. Quando um Ipkt chega no produtor (quem produz a informação) ou no provedor (quem hospeda o conteúdo), eles emitem uma primitiva de dados (Dpkt) para cada primitiva de intenção (Ipkt) recebida. O Dpkt faz o caminho inverso do Ipkt através de roteamento simétrico.

RINA (UNIVERSITY, 2013; ISHAKIAN et al., 2012; TROUVA et al., 2012) baseia-se no princípio fundamental de que a rede é uma comunicação entre processos (*Inter-Process Communication (IPC)*). A premissa básica da arquitetura é que uma rede não é um conjunto de camadas de funções diferentes, mas uma única camada distribuída de

comunicação entre processos que se repete em diferentes escopos (UNIVERSITY, 2013). Cada instância dessa camada pode estar instalada em dispositivos diferentes e implementa os mesmos mecanismos, mas as políticas são ajustadas para operar diferentes faixas de espaço de desempenho (capacidade, atraso, perda, etc.) (UNIVERSITY, 2013). Cada camada representa um escopo e possui um conjunto de processos de aplicação *IPC* que fornece serviços distribuídos para processos de aplicação das camadas superiores; por exemplo, comunicação. Os processos de aplicação das camadas superiores, eles próprios podem ser *Inter-Process Communications (IPCs)* de processos de aplicação das suas camadas superiores, e assim por diante. Essa sobreposição de camadas, onde a camada inferior é um substrato de recursos da camada superior configura a recursividade (UNIVERSITY, 2013). Cada camada, em *RINA*, é denominada Instalação de *IPC* Distribuído (*Distributed IPC Facility (DIF)*). Trata-se de uma estrutura organizadora que define um escopo de serviços através de um conjunto de processos distribuídos de aplicação IPC. O *DIF* é responsável pela funcionalidade da rede (TROUVA et al., 2011). Uma forma de implementar uma comunicação é ter várias *Distributed IPC Facilites (DIFs)*, onde cada uma tem uma função específica: segurança, controle de fluxo, fragmentação, etc. Por outro lado, uma camada pode reunir vários recursos para executar uma tarefa específica. Cada DIF é independente e cria um substrato que fornece recursos específicos para processos de aplicação de camadas superiores (BODDAPATI et al., 2012). De forma mais geral, Instalação de Aplicação Distribuída (*Distributed Application Facility (DAF)*) é um conjunto de processos de aplicação que cooperam em executar determinada função (UNIVERSITY, 2013). DIF é um DAF específico formados por tipos de processos IPCs que se limitam à comunicação (UNIVERSITY, 2013). Um processo de aplicação distribuída importante em *RINA* é o Inter-DIF DIRETORY (IDD). São processos de aplicação que formam DAFs de gerenciamento responsáveis pela gestão de nomes, e consequentemente, pela busca de entidades fora do escopo de um DIF (UNIVERSITY, 2013; TROUVA et al., 2012). Um IDD procura pelo objeto requisitado fazendo buscas nos IDDs dos vizinhos até encontrá-lo ou até atender uma condição pré-estabelecida de término (TROUVA et al., 2012). Quando encontrado, o IDD pode estabelecer uma conexão entre origem e destino através da extensão de um DIF existente ou por meio da criação de um *DIF* dedicado (ISHAKIAN et al., 2012; UNIVERSITY, 2013). O roteamento é realizado através dos *DIFs* subjacentes, sendo que os endereços são nomes que representam os *IPCs* (nós que garantem o encaminhamento das mensagens).

MobilityFirst (MF) (RAYCHAUDHURI; NAGARAJA; VENKATARAMANI, 2012; VENKATARAMANI et al., 2014) oferece uma nova camada de rede (*GUID Service layer*) que disponibiliza uma série de serviços que são essenciais para o problema da mobilidade. Ela utiliza o conceito de Identificadores Globais Exclusivos (*Globally Unique Identifiers (GUIDs)*) para habilitar serviços presentes nos dispositivos móveis e também para ser um ponto crucial de segurança da arquitetura (RAYCHAUDHURI; NAGARAJA; VEN-

KATARAMANI, 2012). Os *GUIDs* são identificadores planos (*flat*) e de nível de rede de longa duração, que acompanham as entidades por todo o seu ciclo de vida. As primitivas MF possuem *Globally Unique Identifier (GUID)* de origem e destino, identificador do serviço (*Service Identifier (SID)*) e uma lista de endereços de rede(Network Adresses - NAs) em seu cabeçalho (RAYCHAUDHURI; NAGARAJA; VENKATARAMANI, 2012). A entidade remetente MF solicita um serviço da rede através do *GUID* da entidade de destino e através do *SID* que especifica o serviço de comunicação. O modo de entrega da comunicação é passado como parâmetro para a API do serviço (RAYCHAUDHURI; NAGARAJA; VENKATARAMANI, 2012): unicast(padrão), *multicast*, multi-homed, recuperação de conteúdo ou mensagem baseada em contexto. O roteamento do MF (protocolo GSTAR) baseia-se em uma associação de nomes diferentes (híbrida), que vincula o *GUID* a uma lista de NAs. O Serviço de Resolução Global de Nomes (*Global Name Resolution Service (GNRS)*) realiza essa vinculação salto a salto (*hop-by-hop*) e por isso, o roteamento é dinâmico e a vinculação é tardia (late-binding) (RAYCHAUDHURI; NAGARAJA; VENKATARAMANI, 2012). O *GNRS* é um recurso central da arquitetura centrada em mobilidade, permitindo a vinculação instantânea de nomes a endereços roteáveis para cenários dinâmicos de desconexão, multi-homed, migração de serviços, entre outros (RAYCHAUDHURI; NAGARAJA; VENKATARAMANI, 2012). Alguns outros recursos do protocolo de roteamento do MF são importantes: reconhecimento de rede de borda (roteamento interdomínio) e armazenamento (recurso do roteamento intradomínio) (RAYCHAUDHURI; NAGARAJA; VENKATARAMANI, 2012). O primeiro garante a entrega de dados de forma eficiente para redes que possuem propriedades diferentes. O segundo recurso permite que a rede possa gerenciar armazenamentos de dados/primitivas para superar flutuações na qualidade do link e desconexões geradas por mobilidade. O armazenamento permite que a rede MF retransmita as informações depois que a rede se estabilize. Essa funcionalidade torna a rede mais robusta.

Publish-Subscribe Internet Technologies (PURSUIT) (FOTIOU et al., 2012; TROSSEN; PARISIS, 2012) é uma arquitetura de rede centrada em informação (*ICN*). Através de um conjunto de métodos, as aplicações consomem e produzem recursos na rede. Os métodos permitem publicar/cancelar e assinar/cancelar item (unidade) ou escopo (conjunto) de informação. Através do paradigma de Publicação/Assinatura, *PURSUIT* suporta de forma nativa tipos de comunicação como *multicast*, *anycast* e multihoming (XYLOMENOS et al., 2012) e proporciona segurança em nível de pacote (FOTIOU et al., 2012). *PURSUIT* possui três funções que disseminam informação (TROSSEN; PARISIS, 2012): *Rendezvous* possui a localização das entidades comunicantes e do conteúdo, por isso promove o encontro entre os pares; *Topology Formation* recebe a informação de *Rendezvous* e elabora a topologia de rede subjacente, também calcula caminhos para entrega da informação; (*Forwarding*) realiza a transferência das informações solicitadas, encaminhando-as para as interfaces de rede ou para o proxy local.

Serval (NORDSTRÖM et al., 2012) é uma arquitetura de sobreposição da camada IP baseada em Redes Centradas em Serviços (*SCN*). As funcionalidades de sobreposição do *Serval* fazem parte de uma nova camada de acesso a serviço (*Service Access Layer (SAL)*), que se localiza entre as camadas IP e TCP tradicionais; e, possibilita que entidades comuniquem-se diretamente através de nomes de serviços. A arquitetura possui três identificadores: serviços, fluxos e localizadores. A comunicação entre entidades é baseada em identificadores de serviços (*serviceIds*), que representa a entidade comunicante. A identificação de fluxos (*flowIds*) representa o contexto da comunicação. Os identificadores de localização representam o local ou o *host* onde o serviço está sendo executado. O roteamento se torna possível quando o serviço de resolução de nomes do *Serval* vincula *serviceIds* aos localizadores das entidades que participam da comunicação. Em relação aos *flowIds*, eles são importantes no momento que a primitiva *Serval* chega ao receptor. Somente através desse identificador, a *SAL* é capaz de desencapsular a informação e enviá-la à aplicação correspondente.

2.1.9 Uma breve reflexão sobre as redes do futuro.

Alguns pontos são fundamentais para se discutir as redes do futuro. Espera-se que esta rede possa resolver os problemas das limitações atuais da Internet, como por exemplo, a dificuldade de atender os requisitos das novas aplicações, que estão cada vez mais exigentes. A evolução contínua da camada de aplicação têm modificado o padrão de fluxo da Internet nos últimos anos (CISCO, 2020). Um exemplo são os serviços Over-The-Top (SUJATA et al., 2015), cujas aplicações esperam acesso ubíquo aos serviços de rede, de qualquer dispositivo, por meio de qualquer rede de acesso e com requisitos específicos de qualidade de serviço (*QoS*) (BOUBENDIR; BERTIN; SIMONI, 2016). Outros pontos que essa rede do futuro deve considerar referem-se a questões de gerenciamento e auto-organização (*Self-organising Networks (SON)*) (WANG; YAN, 2016; CANINI et al., 2017; RAMIREZ-PEREZ; RAMOS, 2016).

A arquitetura TCP/IP tem sido cada vez mais coadjuvante no processo de comunicação. De fato, o que tem acontecido nos agentes de telecomunicações é que os requisitos cada vez mais exigentes das aplicações têm levado a um forte acoplamento entre serviços e infraestruturas de rede (BOUBENDIR; BERTIN; SIMONI, 2016). Na ausência do suporte de rede TCP/IP, esse acoplamento produz um modelo estático de fornecimento de serviços de rede, no qual cada tipo de serviço de rede é suportado por uma capacidade de infraestrutura específica (BOUBENDIR; BERTIN; SIMONI, 2016). Esse modelo produz dois problemas: (i) o agente de telecomunicações fornece uma solução única, monolítica e fechada para os requisitos de determinada aplicação, não permitindo, dessa forma, que essa solução adapte-se às necessidades dinâmicas do usuário; (ii) se novos requisitos surgem na camada de aplicação, pode ser necessário a implantação de outra infraestrutura que tenha novas capacidades que solucionem as novas exigências do usuário, proporcio-

nando gastos adicionais com operações (*Operating expense (OPEX)*) e capital (*Capital expenditure (CAPEX)*).

Nesse contexto, é vital que as redes de telecomunicações do futuro desacoplem as lógicas dos serviços das infraestruturas físicas e aproximem as aplicações das capacidades que ela oferece (BOUBENDIR; BERTIN; SIMONI, 2016; AHMADI, 2019; DUAN; YAN; VASILAKOS, 2012). Para isso, as arquiteturas devem aproveitar os benefícios da virtualização de rede, que é base principal de softwarização e fatiamento de rede (AFOLABI et al., 2018). Para materializar esses conceitos, redes 5G (AFOLABI et al., 2018; BARAKABITZE et al., 2020) e várias outras soluções de arquiteturas tais como (BOUBENDIR; BERTIN; SIMONI, 2016) têm demonstrado que os paradigmas *SDN* e *Network Functions Virtualization (NFV)* são peças fundamentais para definir um modelo *NaaS* que oferece redes (serviços de rede) de acordo com os casos de uso específicos da aplicação.

2.1.9.1 Alguns conceitos são importantes para as Redes do Futuro.

A virtualização de rede é a representação baseada em software de recursos de hardware e software, considerando as funcionalidades dos planos de controle e de dados (AFOLABI et al., 2018). A virtualização de rede na Internet permite que redes virtuais heterogêneas de diferentes operadoras de serviço coexistam no mesmo ambiente de comunicação, porém isoladas uma das outras (DUAN; YAN; VASILAKOS, 2012). A composição dessas redes lógicas interdomínos permitem a implantação de serviços fim a fim personalizados, pois conseguem compartilhar e utilizar recursos de redes subjacentes fornecidos por provedores de infraestrutura (DUAN; YAN; VASILAKOS, 2012). Além disso, a virtualização de rede possibilita a separação entre o provisionamento de serviços de rede dos mecanismos de transporte de dados, dividindo o papel dos *Internet Service Providers (ISPs)* em duas entidades: Provedores de Infraestrutura (InPs), que gerenciam a infraestrutura física e provedores de serviços de rede (*Service Providers (SPs)*), que criam redes virtuais para oferecer serviços de rede fim a fim, utilizando os recursos obtidos das InPs (DUAN; YAN; VASILAKOS, 2012). A infraestrutura de rede física, composta por links e nós, é virtualizada e disponibilizada para redes virtuais, que podem ser configuradas e desmontadas dinamicamente por *SPs*, de acordo com as necessidades do cliente (DUAN; YAN; VASILAKOS, 2012).

O conceito de fatiamento de rede (*network slicing*) e softwarização estão intrinsecamente ligados ao conceito de virtualização.

O conceito de fatiamento pode ser rastreado até a ideia de Infraestrutura como Serviço (Infrastructure as a Service (IaaS)), em que diferentes locatários compartilham recursos de computação, rede e armazenamento, criando redes virtuais e funcionais em uma infraestrutura física comum (AFOLABI et al., 2018). No contexto do 5G, introduzido pela *Next Generation Mobile Network (NGMN)* (NGMN, 2025), o fatiamento de rede facilita várias redes lógicas independentes sob uma plataforma de infraestrutura física comum,

consumindo recursos físicos/lógicos de rede e nuvem para oferecer inovações técnicas (serviços) e de negócios em um ambiente multilocatório programável e orientado a software. O 3GPP define fatiamento de rede como uma tecnologia que "permite que a operadora crie redes personalizadas para fornecer soluções otimizadas para diferentes cenários de mercado que exigem diversos requisitos, tais como desempenho e isolamento (AFOLABI et al., 2018). O ITU-T descreve o fatiamento de rede como partições de rede isoladas e lógicas, compostas por múltiplos recursos virtuais, isolados e equipados com planos de dados e controle programáveis (AFOLABI et al., 2018).

Softwarização de rede é uma abordagem que envolve o uso de programação e software para projetar, arquitetar, implantar e gerenciar componentes de rede (AFOLABI et al., 2018; BARAKABITZE et al., 2020).

2.1.9.2 As redes móveis já estão se adaptando às necessidades das redes do futuro

O sistema de quinta geração de telefonia móvel (5G system (5GS)) (3GPP, 2022), por exemplo, materializa conceitos de softwarização e fatiamento de rede através de tecnologias tais como *SDN* e *NFV* (AHMADI, 2019). A arquitetura 5G é orientada a serviços com serviços de rede modularizados e foi construída com base nos princípios de que sistemas 5G devem suportar uma variedade de aplicações com diferentes características e requisitos de desempenho (AHMADI, 2019). Essa abordagem aproxima a infraestrutura de rede das aplicações, que conseguem se comunicar através de *APIs*. Essas interfaces de serviços modularizados permitem que a rede 5G exponha suas capacidades de rede e que provedoras de serviços descubram, adicionem e atualizem serviços de rede de acordo com as necessidades. Um dos conceitos mais inovadores incorporados dentro das redes das próximas gerações é a separação das funções do plano de usuário (plano de dados) e do plano de controle, que permite implantação de redes centralizadas ou distribuídas, individualmente escaláveis e flexíveis (AHMADI, 2019). O desacoplamento dos serviços de rede do plano de dados permite que fatiamentos de rede possam executar conjuntos distintos de recursos de computação, armazenamento e redes em uma única infraestrutura física, atendendo, dessa forma, diversas necessidades de diferentes casos de uso. Uma fatia de rede pode ser vista como um perfil exclusivo para uma aplicação, para uma categoria específica de serviços (como por exemplo, tipos de tráfego específicos), ou também pode ser oferecido para um cliente sob demanda através de uma abordagem *NaaS* (AHMADI, 2019). Além disso, a separação entre lógica de serviços e infraestrutura possibilita que os serviços sejam implantados em posições estratégicas da rede, mais perto do usuário; na borda da rede de acesso (Mobile Edge Computing - MEC). Alguns casos de uso, tais como computação tátil, que necessitam de baixa latência (*ultra-reliable and low latency communications (URLLC)*) se beneficiam dessa solução.

2.1.9.3 Oferecimento de redes como serviço - NaaS

O trabalho (BOUBENDIR; BERTIN; SIMONI, 2016) demonstra que a utilização de *NFV* e *SDN* são elementos essenciais para definir um modelo arquitetônico *NaaS*, que permite que agentes de telecomunicações ofereçam seus serviços de rede de forma dinâmica a terceiros. Para isso a arquitetura possui duas camadas. A Camada de Exposição de Rede aproxima a visibilidade dos recursos de rede das aplicações, permitindo, por exemplo, que as aplicações descubram, solicitem e selecionem os serviços desejados. Além disso, essa camada é responsável pela composição das funções de rede. A camada de Infraestrutura de Rede de Software compreende um conjunto de recursos virtualizados de computação, armazenamento e rede, controlados pelo Gerenciador de Infraestrutura Virtualizada (*Virtualized Infrastructure Managers (VIM)*). O trabalho implanta esses serviços na borda da rede, na forma de datacenters distribuídos. Os controladores *SDN* dentro da infraestrutura centraliza o gerenciamento e desempenha o papel no encadeamento e conectividade de *Virtualized Network Functions (VNFs)*. Esse trabalho, através da arquitetura proposta, fornece serviços de rede de forma dinâmica e consegue automatizar o gerenciamento dos seus ativos através de um controle logicamente centralizado. Ao mesmo tempo, esses serviços são oferecidos na borda da rede para melhorar o desempenho das aplicações.

Percebe-se que oferecer a rede como serviço é fundamental para as redes do futuro, e para que isso seja possível, a realização de um requisito é fundamental: aproximar os usuários das operadoras de rede, ou seja, expor os recursos de rede dos agentes de telecomunicações de forma programática, por meio de *APIs*, oferecendo-lhes uma experiência semelhante ao consumidor, com opções, escalabilidade, visibilidade e controle. Esse recurso, certamente, adiciona alguns mecanismos operacionais: o provedor deve controlar o acesso dessas aplicações (*Role Based Access Control (RBAC)*); deve registrar os logs das atividades e monitorar se a comunicação está condizente com o *SLA* (ORDONEZ-LUCENA; DSOUZA, 2022).

2.1.9.4 O projeto Camara

O projeto Camara (FOUNDATION, 2023; ORDONEZ-LUCENA; DSOUZA, 2022) é uma iniciativa entre a *Global System for Mobile communication Association (GSMA)* e a Linux Foundation: visa garantir uma grande adoção de *NaaS*, impulsionando a padronização e desenvolvimento de *APIs* de telecomunicações que tenham os seguintes princípios: aberta (código fonte da API tem licença Apache 2.0); alcance global (são oferecidas por diferentes operadoras de telecomunicações) e facilidade de uso (abstrai a semântica das *APIs* de rede, ocultando a complexidade das telecomunicações) (ORDONEZ-LUCENA; DSOUZA, 2022). Essas *APIs* pertencem aos domínios de redes móveis, de linha fixa, nuvem e serviços de suporte como autenticação.

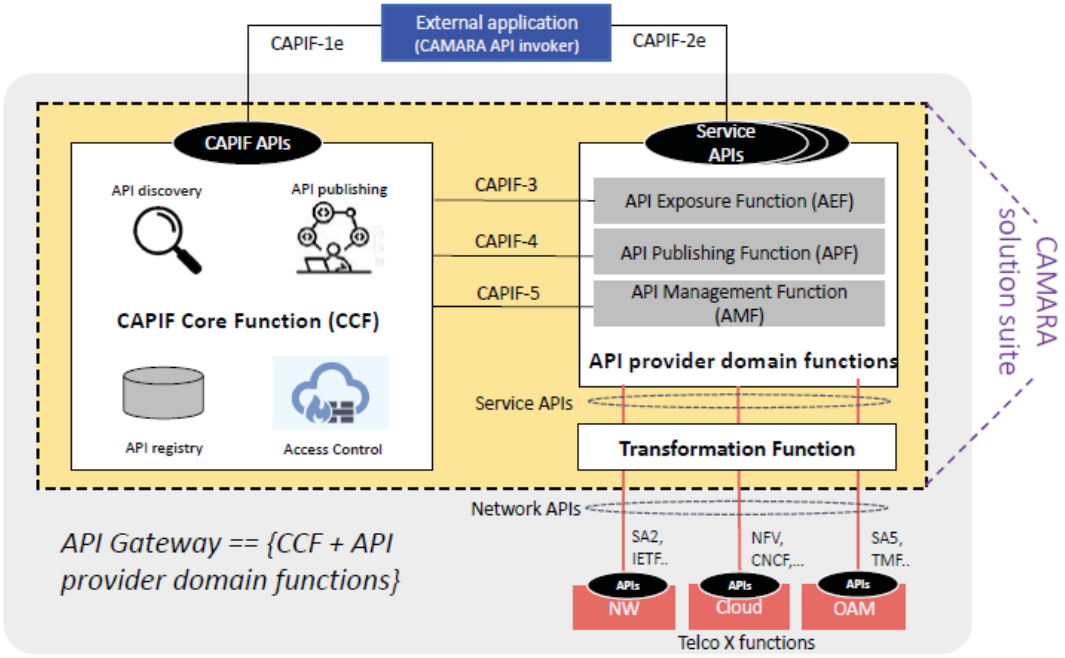

Figura 14 – Executando *CAPIF* na arquitetura Camara. Fonte: (ORDONEZ-LUCENA; DSOUZA, 2022).

A fim de padronizar o desenvolvimento de *APIs* de tal forma que operadoras diferentes possam utilizá-las em seus ambientes de produção, o projeto Camara propõe uma arquitetura de referência, que é apresentada na Figura 14. Algumas definições precisam ser explanadas.

APIs de rede (ORDONEZ-LUCENA; DSOUZA, 2022) são ativos de telecomunicações, incluindo recursos de rede (funções de núcleo, acesso e transporte), recursos de nuvem (infraestruturas de hospedagem de carga de trabalho virtualizadas), recursos de TI (OSS e ferramentas de orquestração), entre outros. Essas *APIs* advém de organizações de regulação tais como 3GPP, ETSI ou TMForum; e estão diretamente relacionadas às tecnologias subjacentes.

APIs de serviço (ORDONEZ-LUCENA; DSOUZA, 2022) são *APIs* disponibilizadas para consumo de terceiros, como por exemplo, aplicativos externos. Essas *APIs* são o foco do projeto Camara, abstraem a complexidade das funcionalidades de rede e são oferecidas para consumo dos usuários.

A arquitetura de referência possui dois componentes arquitetônicos (ORDONEZ-LUCENA; DSOUZA, 2022): o gateway de API, que é responsável pelo controle de acesso dos aplicativos externos, monitoramento das invocações das *APIs* de serviço, fornecimento dos endpoints das *APIs* de serviço, autorização de invocações de *APIs*, registros das atividades em arquivos de log, exposição/publicação dos serviços de rede para torná-los detectáveis pelos aplicativos, entre outros; e, a função de transformação, que mantém informações sobre a correspondência entre *APIs* de serviços e *APIs* de rede, permitindo

fazer o mapeamento de *APIs* de rede em *APIs* de serviço e vice-versa. O projeto Camara propõe que o gateway da arquitetura seja implementado pelo *CAPIF* (3GPP, 2023), basicamente por dois motivos (ORDONEZ-LUCENA; DSOUZA, 2022): embora especificado pelo 3GPP, pode ser utilizado como gateway de qualquer *API*; e, trata-se de uma solução com ampla aceitação pela indústria.

As infraestruturas de rede do futuro caminham na direção de aproximar aplicações dos recursos de rede e o projeto Camara é uma iniciativa que possibilita a produção, em nível industrial, de *APIs* de serviços que abstraem a complexidade de funcionalidades de rede, e que posteriormente, depois de amplamente testadas pelas diretrizes formais do projeto, atingem o status de estável e são publicadas para consumo de aplicações de terceiros. Esse projeto tem a participação de parceiros de peso (PROJECT, 2025) tais como Huawei, Cisco, Juniper Networks, Verizon Communications INCC, Comcast Corp., AT&T Inc., AMD, Intel, British Telecom, China Mobile, Deutsche Telekom AG, Ericsson, *GSMA*, Nokia, TM Fórum, IEEE, entre outros; provando a importância desse tema para as redes atuais e futuras.

2.1.9.5 Gerenciamento das redes futuras

O gerenciamento de redes executa operações de provisionamento, controle e monitoramento de recursos de redes computacionais (DING, 2016). O modelo OSI de gerenciamento segue 05 etapas: gerenciamento de falhas, configuração, contabilização, desempenho e segurança (ITU-T, 1992).

O gerenciamento de rede tradicional é realizado por protocolos de monitoramento e configuração tais como SNMP/MIB (FEDOR et al., 1990; CASE et al., 1993; ROSE, 1990; MCCLOGHRIE; ROSE, 1991) e NETCONF/RESTCONF/YANG (ENNS, 2006; ENNS et al., 2011; BIERMAN; BJöRKLUND; WATSEN, 2017). Esses protocolos possuem algumas limitações que podem torná-los obsoletos quanto as suas utilizações em redes futuras (RUOHONEN, 2016). Alguns exemplos podem ser citados. Apesar da padronização do modelo de dados oferecidos pelo MIB e YANG, os fabricantes de dispositivo podem estender esses modelos, criar novas informações de monitoramento e novos modos de configuração, dificultando, dessa forma, a automação de gerenciamento para equipamentos de rede de diferentes fabricantes. Além disso, a semântica das informações registradas nesses modelos de dados pode destoar entre fornecedores, o que dificulta, por exemplo, geração de relatórios que agregam dados de monitoramento. Outro problema é que esses protocolos levam muito tempo para implantar novos serviços devido ao engessamento dos elementos de rede. Embora o NETCONF/RESTCONF sirvam como um facilitador para *SDN*, ele não está totalmente alinhado com a visão mais ampla desse paradigma, que é uma rede totalmente programável e sem lógica de controle incorporada aos dispositivos do plano de dados (RUOHONEN, 2016).

As medições estáticas atuais de desempenho de redes e aplicações não são extensíveis

às dinamicidades das redes do futuro, nem capazes de criar mecanismos de automação que gerem comportamentos autoadaptáveis (AHMADI, 2019). O ambiente dessas novas redes exigem, por exemplo, análise de medições em tempo real, dados de telemetria, informações baseadas em fluxo e implantação de novos serviços de gerenciamento na medida em que há novas demandas de monitoramento ou configuração (AHMADI, 2019). Mecanismos de processamento de dados massivos coletados da rede são imprescindíveis para que mecanismos de inteligência artificial consigam suportar redes auto-organizáveis, auto-otimizadas e autorrecuperáveis (AHMADI, 2019).

Uma forma de atenuar esses problemas é a utilização do paradigma *SDN* para gerenciamento das redes futuras: a operação centraliza-se logicamente e os planos de dados e controle programáveis, extensíveis e configuráveis são mais adequados ao dinamismo dessas redes. Porém, é preciso mencionar que as redes do futuro abrem novos pontos de preocupação, porque há o incremento de novos componentes tais como controladores e possíveis novos serviços, como por exemplo, implementações de fatiamento de redes. Como exemplo, pode-se pegar um dos critérios de gerenciamento, que são as falhas. Considera-se como falha de rede tradicional um problema que torna a rede inoperável, como por exemplo, uma rota que apresenta falha devido a indisponibilidade de um link. No caso de uma fatia de rede, mesmo que esse link esteja disponível, essa fatia que fornece conectividade para um serviço de *streaming* pode estar falhando devido às altas latências de transmissão (NETO et al., 2021). É válido lembrar que a complexidade do gerenciamento aumenta quando a implantação da fatia de rede envolve vários domínios administrativos. Nesse caso, o gerenciamento deve atingir vários sistemas autônomos com várias infraestruturas de rede distintas, e deve, por exemplo, ter mecanismos interdomínios para manter a qualidade de comunicação fim a fim e outros requisito que possam ter sido acordados em SLAs (MOREIRA et al., 2021). No caso dos controladores, uma falha precisa ser isolada, pois pode culminar em *livelocks* ou *deadlocks* (FONSECA; MOTA, 2017; NETO et al., 2021).

2.1.9.6 Considerações finais sobre as redes do futuro

A virtualização de rede é o conceito que vai fundamentar as redes do futuro e as tecnologias *SDN* e *NFV* são peças indispensáveis da sua implementação. A virtualização permite o gerenciamento da rede através de software, a criação de redes lógicas em uma mesma infraestrutura física, a separação lógica do software das topologias subjacentes, proporcionando elasticidade, escalabilidade e gerenciamento independente dos planos de dados e controle. A virtualização de redes promove maior capacidade de gerenciamento, flexibilidade e diversidade. Uma Internet diversificada possibilita um ambiente rico para inovações, estimulando de forma mais simples o desenvolvimento e a implantação de novos serviços. Além disso, a virtualização de redes facilita a entrega de serviços fim a fim em infraestruturas de rede de diferentes domínios, proporcionando entrega de qualidade de

serviço de ponta a ponta. Toda essa dinamicidade proporciona a entrega de redes como serviço, aproximando, dessa forma, as aplicações das capacidades de redes através de interfaces programáveis.

O projeto Camara é um exemplo de iniciativa que visa criar um acervo de *APIs* que caminham nessa direção. Como exemplo, um aplicativo móvel de uma instituição financeira pode aumentar sua camada de autenticação utilizando serviços de prevenção e fraude das operadoras. Antes de completar uma operação de *login*, o aplicativo pode, por exemplo, verificar se os dados do cartão SIM do dispositivo estão em conformidade com os dados do cliente na operadora ou podem constatar se o dispositivo está ativo no momento da solicitação de login. Outro exemplo é apresentado em (ORDONEZ-LUCENA; DSOUZA, 2022), onde é demonstrado a utilização da *API* de qualidade sob demanda (*Quality on Demand API*). No caso de uso apresentado, equipamentos móveis de usuário utilizam essa *API* para melhorar a qualidade de comunicação de serviços de jogos em nuvem em uma rede congestionada.

Outro ponto importante é que redes baseadas em softwarização (SDN) possibilitam a inclusão de serviços e arquiteturas de gerenciamento capazes de tornar a rede autônoma, criando mecanismos de auto-organização (*SON*) e autogerenciamento (AHMADI, 2019). Vários trabalhos caminham nessa direção (WANG; YAN, 2016; CANINI et al., 2017; RAMIREZ-PEREZ; RAMOS, 2016).

2.2 Trabalhos Relacionados

A adoção, implementação e implantação de novas arquiteturas de rede na Internet atual tem se mostrado um processo lento e custoso. O IPv6, por exemplo, tido como candidato a sucessor do IPv4, ainda está sendo implementado, de forma incremental, após vários anos da sua proposta inicial (BRADNER; MANKIN, 1995; CZYZ et al., 2014). Essa dificuldade estende-se para serviços *clean-slate*, que possuem diferentes orientações de design, esquemas de endereçamento de rede, paradigmas de comunicação, entre outros.

A consequência imediata disso é que o engessamento da Internet torna-se um obstáculo para a evolução das redes atuais (PAPASTERGIOU et al., 2017). De um lado, aplicativos estão cada vez mais complexos e demandam requisitos cada vez mais exigentes da rede (CISCO, 2020). Por outro, as redes atuais não conseguem inserir em sua infraestrutura novas arquiteturas de Internet que possuem diferentes princípios arquitetônicos do TCP/IP e que visam enfrentar problemas pontuais existentes na comunicação atual, tais como *multicasting*, mobilidade, segurança, *QoS*, *QoE*, entre outros.

Esse cenário não inspirou apenas o desenvolvimento deste trabalho; pelo contrário, incentivou a criação de uma linha de pesquisa que pensa a rede do futuro como um ambiente de comunicação capaz de admitir arquiteturas de Internet heterogêneas, com diferentes orientações de design (*host-oriented*, *data-oriented*, *'qualquer coisa'-oriented*);

Tabela 2 – Coleção de trabalhos relacionados. Características encontradas nos *papers*.

INT - Interoperability. **SDN** - Software Defined Networking. **VIR** - Network Virtualization.
OVE - Network Overlay. **TUN** - Tunneling. **ACT** - Active Networks.
PST - Implementation of the protocol stack. **P4** - P4 language
PRO - Additional processing into the switches. **STA** -Additional deployment of protocol stacks into the switches.
LEV - Level of difficulty in the deployment of new FIAs.
IAs - Internet Architectures integrated into the network.
x - It does not have the functionality. red - Represents negative side effects.
Active networks. SDN and/or Network virtualization. Protocol stack development.
Interoperability. Hybrids (Interoperability, SDN, P4, among others).

Related works.	**Mechanisms used in each related work.**								**Side effects of the mechanisms used.**			
	INT	**SDN**	**VIR**	**OVE**	**TUN**	**ACT**	**PST**	**P4**	**PRO**	**STA**	**LEV**	**IAs**
ANTS				v	v	v	v		Yes	Yes	x	(i) IP. (ii) ANTS.
SOFTNET						v	v		Yes	Yes	x	(i) SOFTNET Protocol.
ENDN.P4[(1)]		v						v	No.	No.	x	(i) Enhanced NDN.
NDN.P4[(1)]		v						v	No	No	x	(i) NDN.
NDN.P4[(2)]		v						v	Yes	No	x	(i) NDN.
RINA.P4		v						v	No	No	x	(i) RINA EFCP protocol. (ii) IP.
RouteFlow and Cardigan		v	v	v					No	No	x	(i) IP.
SDX		v	v	v					No	No	x	(i) IP.
Linux XIA							v		Yes	Yes	High.	(i) Serval. (ii) zFilter. (iii) IP.
Plutarch	v						v		Yes	Yes	x	The *paper* presents a conceptual model.
FIFu	v	v					v		Yes.	Yes.	High.	(i) PURSUIT. (ii) IP. (iii) NDN.
ICN.P4		v		v	v			v	No	No	x	(i) ICN Protocol. (ii) IP.
SCIONLAB		v		v	v				No	No	x	(i) SCION. (ii) IP.
SCN.P4		v		v				v	No	No	x	(i) SCN protocol. (ii) IP.

paradigmas de comunicação(*Push*, *Pull*, outros); esquemas de endereçamento; e, diferentes protocolos.

A ideia de uma arquitetura de metarede, que consegue absorver as diferentes características de uma comunicação, é promissora e existe há décadas. Há diferentes abordagens para solucionar esse problema. A Tabela 2 apresenta um acervo de trabalhos que cami-

nham nessa direção.

2.2.1 Trabalhos relacionados às redes ativas

Esta subseção apresenta meta-arquiteturas que orientam seus *designs* através de redes ativas (TENNENHOUSE et al., 1997). Essas soluções buscam redes programáveis sem segmentar a rede em regiões heterogênas; a rede é única, e as entidades compartilham os recursos disponíveis.

SOFTNET (ZANDER; FORCHHEIMER, 1983) é a arquitetura de metarede mais antiga da Tabela 2, pensada na década de 80. Trata-se de redes de enlace de rádio programáveis através de pacotes (primitivas da arquitetura). Dessa forma, a aplicação ou o usuário pode controlar o comportamento dos nós da rede a partir de códigos de linguagem de programação que são transportados pelas primitivas SOFTNET. O intuito é flexibilizar a criação de serviços de alto nível (como chamadas virtuais) e gerenciar as operações de uma rede de rádio *multi-hop*. Apenas protocolos SOFNET trafegam nessa rede.

ANTS (WETHERALL, 1998) também é uma rede programável que transporta programas ou chamadas de função através das cápsulas (primitivas da arquitetura). A arquitetura possui um protocolo de distribuição automática de código por demanda que diminui a carga de informação na rede e aumenta a performance de execução das rotinas encaminhadas. Ela admite a coexistência da arquitetura IP e dos protocolos ANTS.

2.2.2 Trabalhos relacionados ao paradigma SDN

Esta subseção apresenta arquiteturas que buscam redes programáveis (metaredes) através de *SDN*. Alguns desses trabalhos mostram a tendência de FIAs em migrarem suas funcionalidades para o paradigma *SDN-P4*. O propósito dessas migrações é integrá-las, gradualmente, com outras arquiteturas de Internet, e, acelerar suas implantações em escala industrial (GUO et al., 2021; GIMENEZ; GRASA; BUNCH, 2020). No entanto, os trabalhos estão em diferentes estágios de implementação. Alguns, apresentam casos de uso que mostram a coexistência de suas funcionalidades *clean-slate* com a arquitetura TCP/IP (GIMENEZ; GRASA; BUNCH, 2020). Outros, ainda estão desenvolvendo o plano de dados, testando as limitações e os benefícios do paradigma *SDN-P4* em relação à implementação de seus serviços nesse ambiente (GUO et al., 2021; SIGNORELLO et al., 2016). No entanto, todos esses trabalhos enxergam o plano de dados programável como característica importante para a rede da próxima geração e o que fundamenta essa visão é a possibilidade de coexistência de diferentes serviços de rede em um mesmo ambiente de comunicação.

Software Defined Internet eXchange (SDX) (GUPTA et al., 2014), RouteFlow (NASCIMENTO et al., 2011) e Cardigan (STRINGER et al., 2014) promovem um ponto de

troca de Internet, gerenciado por software, para melhorar a qualidade e flexibilidade do roteamento IP. *SDX* estende as funcionalidades do BGP para ter mais controle de encaminhamento do tráfego pela Internet; além disso, permite que os participantes do *Internet eXchange Point (IXP)* programem suas próprias políticas de roteamento através de switches *SDN* virtuais. Esses switches virtuais abstraem a complexidade da topologia física, isolam o tráfego e simulam a ilusão de que o participante é o único que está utilizando os recursos da rede. Para escrever políticas de roteamento, os usuários utilizam a linguagem *Pyretic*. RouteFlow e Cardigan não estende as funcionalidades do BGP, mas fornecem roteamento IP como serviço de nuvem, e por conseguinte, o *IXP* é lógico e pode ser distribuído para vários locais. Essas três arquiteturas são dependentes do BGP, oferecem uma rede programável e admitem apenas as funcionalidades da arquitetura IP.

NDN.P4$^{(1)}$ (GUO et al., 2021), *NDN.P4*$^{(2)}$ (SIGNORELLO et al., 2016) e *Enhanced Named Data Networking (ENDN)*.P4 (KARRAKCHOU; SAMAAN; KARMOUCH, 2020) são trabalhos que possuem em comum os mesmos objetivos: facilitar a integração de arquiteturas *ICN* em redes de produção; separar os planos de controle e dados; e, fornecer programabilidade no pipeline dos elementos de comutação. *NDN.P4*$^{(1)}$, *NDN.P4*$^{(2)}$ e *ENDN.P4* promovem a implantação de *NDN* em switches *P4*, cada qual com implementações independentes. NDN.*P4*$^{(1)}$ melhora o desempenho do protocolo de roteamento *Named-data Link State Routing (NLSR)*, porém, não possui o componente de cache do *NDN* nos swithces *P4*. *NDN.P4*$^{(2)}$ desenvolve o reconhecimento de pacotes *NDN* no plano de dados, mas não implementa o plano de controle. *ENDN.P4* produz uma versão modificada do *NDN*, melhorando o plano de dados do *NDN* original e oferecendo um portfólio de serviços de rede que não estão presentes no *NDN* tradicional, como por exemplo, conexões Push e Publish/Subscribe. É importante salientar que todas essas soluções de rede admitem apenas a arquitetura de Internet *NDN*.

RINA.P4 (GIMENEZ; GRASA; BUNCH, 2020) implementa parcialmente as funcionalidades da arquitetura *RINA* (VRIJDERS et al., 2014). O trabalho propõe a implantação de um protótipo do roteador interno *RINA* baseado em linguagem *P4*. Esse protótipo não executa algoritmos de escalonamento de pacotes para multiplexação de tráfego nem aplica mecanismos de segurança nas primitivas processadas. Inicialmente, esse roteador reconhece apenas duas arquiteturas: IP e *RINA*.

2.2.3 Trabalhos relacionados à implementação de pilhas de protocolos

Esta subseção apresenta meta-arquiteturas que buscam a coexistência de arquiteturas de Internet distintas em uma rede única, sem fragmentos ou regiões. Para isso, esses projetos criam meta-arquiteturas com conceitos arquitetônicos abstratos que possuem capacidade de evoluir e absorver as funcionalidades de outras arquiteturas de Internet.

Linux *XIA* (MACHADO; DOUCETTE; BYERS, 2015) é uma meta-arquitetura que desenvolve uma pilha de protocolos para cada *FIA* integrada à sua rede. Dessa forma, fornece um substrato físico de rede que pode ser compartilhado entre diferentes *FIAs*. Qualquer *FIA* nova da rede tem que ser anteriormente interpretada para os princípios arquitetônicos da arquitetura *XIA* (HAN et al., 2012). Todas as pilhas de protocolos, que representam novas *FIA*s, devem estar presentes nos elementos de rede da arquitetura, o que pode causar sobrecarregamento dos dispositivos de comutação. O trabalho mencionado possui casos de uso de três arquiteturas diferentes: Serval (NORDSTRÖM et al., 2012), zFilter (JOKELA et al., 2009) e IP.

2.2.4 Trabalhos relacionados à interoperabilidade

Esta subseção apresenta meta-arquiteturas que buscam a coexistência transparente entre *FIAs* através do processo de interoperação. Para isso, essas meta-arquiteturas segmentam a rede em várias regiões, onde cada uma delas representa a rede de uma única arquitetura de Internet. Depois, criam pontes que traduzem os protocolos de comunicação de cada região em ambas as direções (*internetworking*). O dispositivo responsável pela tradução faz o papel de gateway da comunicação entre arquiteturas. Dessa forma, um servidor *WEB ETArch* (SILVA et al., 2014) pode, por exemplo, receber requisições de um *browser NovaGenesis* (ALBERTI et al., 2019).

Plutarch (CROWCROFT et al., 2003) é uma arquitetura de interconexão e se concentra na conectividade entre redes radicalmente heterogêneas. O processo de interoperabilidade é possível devido a descrições de contexto e funções intersticiais. As descrições de contexto descrevem as propriedades de cada região (rede). As funções intersticiais são canais lógicos entre contextos (regiões) e são responsáveis pelo mecanismo de tradução de protocolos, primitivas, identificadores, serviços, entre outros; de um contexto para outro. Plutarch ainda é um modelo teórico, e os casos de uso apresentados não abordam a complexidade da conexão entre FIAs heterogêneas.

2.2.5 Trabalhos relacionados que possuem abordagens híbridas

Esta seção apresenta as meta-arquiteturas que apresentam uma abordagem híbrida, ou seja, podem ter mecanismos de interoperabilidade, virtualização de redes, SDN, sobreposição de redes, tunelamento e desenvolvimento de pilhas de protocolos.

ICN.P4 (FENG; TAN; JIN, 2019) tem uma solução híbrida, pois além de *switches P4*, utiliza mecanismos de sobreposição de redes (*overlay networks*), tunelamento e interoperabilidade. A solução passa por algumas fases: proposta de um novo formato de primitiva que transporta cabeçalhos TCP/IP e *ICN*; programação do plano de dados para reconhecer primitivas *ICN.P4*; desenvolvimento de uma aplicação que possui funções de gateway e transforma primitivas HTTP/TCP/IP em *ICN.P4* e vice-versa. Através desses

mecanismos, *ICN.P4* possibilita a transmissão de solicitações HTTP/TCP/IP em redes *ICN*, e por consequência, a coexistência de funcionalidades dessas arquiteturas em uma rede única de comunicação.

SCN.P4 (BAKTIR; OZGOVDE; ERSOY, 2018) tem uma solução híbrida, pois utiliza programabilidade do plano de dados e mecanismos de sobreposição de redes (*overlay networks*). A rede identifica requisição de serviços e microsserviços do protocolo *SCN*. Para isso, o switch *P4* vincula, dinamicamente, o identificador do serviço a um endereço (IP), que localiza a instância. Por esse motivo, *SCN.P4* depende do IP. O trabalho não desenvolve o plano de controle da solução. No entanto, o plano de dados compartilha funcionalidades de *SCN* sem alterar a pilha de protocolos TCP/IP; permitindo a coexistência dessas arquiteturas em uma rede única de comunicação.

SCIONLAB (KWON et al., 2020) é um testbed que possibilita construir cenários reais para testar, em escala global, diferentes características de Internet do Futuro da arquitetura *Scalability, Control, and Isolation On Next-Generation Networks (SCION)* (ZHANG et al., 2011). Esse projeto é importante pois permite testar o *SCION* em uma rede que está bem próxima da Internet atual, possibilitando experimentos que demonstram a viabilidade e a maturidade das suas funcionalidades em cenários específicos. SCIONLAB separa os planos de aplicação, dados e controle; e, utiliza mecanismos de sobreposição de rede para realizar roteamentos entre domínios. Esses mecanismos permitem, de certa forma, a coexistência entre a arquitetura IP e *SCION* (ZHANG et al., 2011).

Future Internet Fusion (FIFu) (GUIMARãES et al., 2019) segmenta a rede em regiões (*FIAs*) e promove pontes (*bridges*) que traduzem a comunicação entre essas arquiteturas em ambas as direções (*internetworking*). O que possibilita a tradução entre protocolos de regiões distintas é a descrição de contextos, que contém o detalhamento das propriedades de cada região. O componente FIXP$^{(2)}$ pertence ao plano de dados e é o gateway que realiza a tradução. *Future Internet Controllers (FICs)* são dispositivos de controle, que funcionam como um sistema de diretório distribuído; também controlam e auxiliam as operações dos *Future Internet eXchange Points (FIXPs)*$^{(2)}$ (plano de dados). A arquitetura tem o objetivo de permitir a coexistência de *FIAs* heterogêneas que possuem diferentes princípios arquitetônicos. As *FIAs* atuais que a arquitetura admite são: *PURSUIT* (TROSSEN; PARISIS, 2012), IP e *NDN* (ZHANG et al., 2014).

2.2.6 Uma breve discussão dos trabalhos relacionados

A Tabela 2 apresenta os mecanismos utilizados por cada trabalho relacionado descrito nas subseções anteriores e os efeitos colaterais que essas decisões de projeto causam na rede. Esta seção realiza uma discussão das consequências que alguns paradigmas e/ou mecanismos de rede tais como Interoperabilidade, Redes Ativas, implantação de diversas pilhas de protocolos, entre outros; estabelecem na comunicação. Os argumentos utilizados nesta subseção auxiliam, posteriormente, o posicionamento do *FIXP* no estado da arte.

As redes ativas fornecem uma rede programável no plano de dados que capacitam os *switches* a reconhecerem diversos tipos de comunicação. No entanto, esses nós ativos interpretam códigos de linguagem de programação que são transportados através das primitivas da arquitetura. Esse fato gera algumas consequências negativas: aumenta o tráfego da rede e sobrecarrega o processamento e a memória dos elementos de comutação. Além disso, o paradigma pressupõe que para transmitir menos primitivas na rede, a meta-arquitetura tem que instalar mais funcionalidades (códigos de serviços) nos *switches*; o que pode sobrecarregar a capacidade de armazenamento do equipamento e dificultar as atualizações de serviços na rede. Até onde se sabe, nenhuma das arquiteturas que utilizam esse paradigma elevam seus casos de uso para a complexidade de comunicação entre *FIAs* distintas. O ponto positivo é que a rede é única e padroniza a comunicação de serviços que possuem diferentes abordagens.

Os trabalhos que desenvolvem pilhas de protocolos para cada *FIA* que faz parte da meta-arquitetura possui alguns problemas. Primeiramente, traduzir as funcionalidades de uma arquitetura para os princípios de uma meta-arquitetura tem um nível de dificuldade elevado. O autor dessa implementação tem que conhecer profundamente as arquiteturas envolvidas em termos semânticos (interpretação dos dados, reduzindo ambiguidade), sintáticos (formato dos protocolos e formas de comunicação: XML, Protobuf, entre outros.), pragmáticos (como por exemplo, SLAs) e técnicos (diferentes protocolos de comunicação). Em segundo lugar, o processo de implantação pode ser ineficiente, causando perdas de funcionalidades, quando por exemplo, a arquitetura de Internet integrada possuir funcionalidades que não estão presentes na meta-arquitetura. Em terceiro lugar, cada implementação de *FIA* pressupõe uma nova pilha de protocolos, o que pode sobrecarregar os elementos de rede em termos de processamento, memória e armazenamento. Além disso, quanto maior a quantidade de *FIAs* que participam dessa meta-arquitetura, maior a quantidade de pilhas de protocolos, e por consequência, maior é a dificuldade de atualização desses serviços. O ponto positivo é que a rede é única e pode ser compartilhada por várias *FIAs*.

Os trabalhos que utilizam interoperação segmentam a rede em regiões, onde cada região obedece as regras de uma *FIA* específica. Por um lado, essa rede segmentada é mais complexa, pois cada região obedece regras de comunicação distintas, com seus próprios elementos de comutação. Se uma operadora implementar esse tipo de meta-arquitetura; para cada *FIA* que participa da comunicação, tem que existir um domínio (região) e/ou núcleo de rede específico que a suporte; pois, no pior dos casos, cada *FIA* possui seus próprios *switches*. Isso dificulta a instalação dessas regiões (*FIAs*) em escala global. Nesse tipo de meta-arquitetura, o que interliga cada região são gateways, que fazem as pontes lógicas de comunicação. Cada gateway possui entidades que realizam a transformação de um contexto arquitetural em outro. A implementação desses gateways que traduzem pares de arquiteturas é complexa. O autor da implementação tem que

possuir um alto grau de conhecimento das arquiteturas envolvidas. As pilhas de protocolos utilizadas no processo de comunicação devem estar devidamente instaladas nos gateways, o que, juntamente com a execução on-the-fly dos processos de interoperação, podem sobrecarregar o equipamento em termos de memória, processamento e armazenamento. Em casos específicos, como por exemplo, *streaming* de vídeo, a codificação/decodificação de formatos de vídeo podem fazer parte do processo de transformação e sobrecarregar ainda mais os equipamentos. Esses casos acontecem quando o par de arquiteturas utilizam padrões de *streaming* distintos. Em outros casos, é preciso inspecionar *payloads* em tempo de execução para transformar identificadores de recursos em formatos entendidos pelo par de arquiteturas. Além de todos esses problemas, a eficiência do processo também gera dificuldades, porque pode haver perda de funcionalidades pelos mesmos motivos das meta-arquiteturas que desenvolvem pilhas de protocolos, ou seja, quando uma arquitetura não possui a funcionalidade da outra. A interoperação é um processo árduo, onera a rede e pode ser ineficiente. O ponto positivo é que os trabalhos que empregam interoperabilidade dão suporte à coexistência de várias arquitetura de Internets (com todas as ressalvas anteriores) sem precisar modificar a infraestrutura física de cada FIA; ou seja, cada região permanece com suas topologias, dispositivos e protocolos.

Capítulo 3

FIXP: habilitando a coexistência de arquiteturas de Internet distintas

Este capítulo apresenta, minuciosamente, a arquitetura *FIXP*. Ele descreve a estruturação, especificação, configuração funcional, princípios conceituais e procedimentos operacionais dos protocolos, blocos funcionais e dispositivos que suportam a interconexão entre os clientes da rede. Os clientes são aplicações que consomem serviços *clean-slate* (REXFORD; DOVROLIS, 2010), evolucionários (REXFORD; DOVROLIS, 2010) ou legados, que possuem diferentes abordagens que variam desde a camada de enlace até a camada de aplicação. Essas aplicações consomem *frameworks* de *FIAs* distintas e disjuntas, com diferentes orientações de design (orientado a *host*, dados, 'qualquer coisa'); paradigmas de comunicação(*Push*, *Pull*); esquemas de endereçamento; e, diferentes protocolos.

A Seção 3.1 define, formalmente, o *FIXP*. A Seção 3.2 apresenta a visão geral do *FIXP*, mostrando a disposição das camadas, blocos funcionais, serviços e dispositivos dentro da arquitetura *FIXP*. A Seção 3.3 aprofunda a discussão em relação ao conhecimento dessa estrutura, e descreve, detalhadamente, a comunicação interna entre os componentes funcionais do *FIXP*. A Seção 3.4 mostra o formato das principais primitivas do protocolo *FIXP*, como também detalha suas funcionalidades. A Seção 3.5 discute como o *FIXP* realiza a programação do plano de dados, descrevendo os blocos funcionais programáveis e fixos, como também a lógica de funcionamento do *pipeline* de dados dos *switches FIXP*. A Seção 3.6 apresenta a estratégia de implantação gradual do FIXP em escala global. A Seção 3.7 explana, detalhadamente, a integração das arquiteturas ETArch, IP e NovaGenesis na rede *FIXP*. A integração da NovaGenesis não é objeto de pesquisa desta tese de doutorado; por conta disso, este trabalho referencia a dissertação proponente da incorporação dessa FIA na infraestrutura *FIXP*. A descrição ou o detalhamento da integração da NovaGenesis no FIXP, mesmo que referenciado, ou mesmo que não seja objeto ou objetivo desse trabalho, é importante porque demonstra ainda mais a flexibilidade do *FIXP* em receber diversos paradigmas, inclusive aqueles que representam conceitos *ICN*. Para finalizar, a Seção 3.8 faz uma sintetização sobre os temas discutidos, refletindo sobre como

os princípios arquitetônicos do *FIXP* descritos neste capítulo alcançam o cumprimento de boa parte do objetivo geral desse trabalho.

3.1 Definição conceitual do FIXP

Do ponto de vista tecnológico e holístico, considerando a implantação em escala mundial, o FIXP é uma infraestrutura de rede programável que padroniza a comunicação e possibilita a coexistência de aplicações que utilizam bibliotecas (*frameworks*) de diferentes arquiteturas de Internet com o intuito de diversificar as capacidades da rede em atender requisitos de casos de usos específicos atuais e futuros. Entende-se, por infraestrutura de rede, o conjunto de recursos de hardware e software que permitem a conectividade, as operações e o gerenciamento da rede FIXP. Da perspectiva do artefato 'ambiente de comunicação distribuída que permite a coexistência de aplicações que utilizam diferentes Arquiteturas de Internet', a existência da infraestrutura de rede FIXP é necessária e suficiente para que esse ambiente de comunicação se materialize. Por isso, neste trabalho, as expressões 'infraestutura de rede FIXP', 'rede FIXP' e 'infraestrutura FIXP' são sinônimas e representam o mesmo ente de comunicação.

Do ponto de vista tecnológico e restrito, considerando a implantação do FIXP apenas em uma operadora de telecomunicações, *IXP* ou *Point of Presence (PoP)*; o FIXP também é uma infraestrutura ou rede, como definida anteriormente, porém, local. Na operadora X, ela poderia ser chamada de infraestrutura de rede local FIXP da operadora X, ou rede local FIXP da operadora X, ou ponto de troca lógico FIXP da operadora X. Essas expressões, neste trabalho, representam o mesmo ente de comunicação. A função de uma infraestrutura local ou rede local FIXP implantada como uma ilha ou um silo tecnológico é suportar diversos clientes (aplicações) de FIAs distintas, porém, esse silo, isolado, não possibilita a implantação do FIXP em escala mundial; apenas pode funcionar como um ponto de troca lógico entre os domínios administrativos dessas aplicações. O adjetivo "lógico" significa que essa infraestrutura ou rede local FIXP não é fixa, e, por conseguinte, pode ser distribuída para qualquer lugar. Essa locomoção deve-se ao fato de que a gerência da rede é feita por software, sendo que o FIXP proporciona uma separação da lógica dos serviços de controle e gerenciamento da infraestrutura física da rede. Dessa forma, o FIXP pode ser implantado de maneira gradual através desses pontos de troca lógicos, e a conexão desses pontos lógicos ou infraestruturas de rede locais possibilita a comunicação em escala mundial, formando uma rede ou infraestrutura maior, que denomina-se infraestrutura ou rede FIXP (não mais em escala local, mas mundial). Para sintetizar, não se diferencia neste trabalho, a rede/infraestrutura do FIXP da rede/infraestrutura local do FIXP. Nesta tese, os casos de uso são locais, mas ela refere-se ao FIXP apenas como "infraestrutura FIXP", "rede FIXP" ou "ponto lógico FIXP". Essas expressões são sinônimas e representam o mesmo ente de comunicação. Também, em

alguns casos, chama-se de "infraestrutura física do FIXP" o conjunto de *switches* dessa infraestrutura, ou seja, essa expressão refere-se ao plano de dados da rede FIXP.

Do ponto de vista do modelo conceitual e lógico, o FIXP é uma arquitetura de rede programável. A programabilidade do FIXP estende-se aos planos de controle, dados e gerenciamento, e por conta disso, possibilita a coexistência de diversas FIAs no mesmo meio de comunicação, absorvendo suas funcionalidades. Esse fato coloca o FIXP como sendo também uma meta-arquitetura ou uma arquitetura de metarede, pois permite a padronização da comunicação de tal forma que arquiteturas de Internet distintas executam suas aplicações de forma transparente, como se estivessem em suas próprias redes.

3.2 Visão geral da arquitetura FIXP

A Figura 15 mostra a visão geral da arquitetura *FIXP*.

O *FIXP* é baseado em *SDN-P4* (GAVAZZA et al., 2020), e, por conta disso, a arquitetura separa os planos de controle e dados. Três camadas aparecem com essa divisão: Camada de Controle do *FIXP*(*FIXP Control Layer (FCL)*), Camada de Abstração do *FIXP*(*FIXP Abstraction Layer (FAL)*) e Camada Física do *FIXP*(*FIXP Physcical Layer (FPL)*).

FCL contém todos os controladores das *FIAs* que participam da comunicação *FIXP*. É a forma como a arquitetura consegue implantar as funcionalidades de qualquer FIA. Cada controlador é uma peça de software externa que possui aplicações capazes de gerenciar o comportamento dos switches e dos fluxos que atravessam a rede. O protocolo *FIXP* permite o gerenciamento do comportamento do switch através de um programa *P4* feito pelo usuário. Dessa forma, cada controlador define suas próprias tabelas de encaminhamento, ações e registradores que armazenam estados de comunicação. Além disso, o protocolo *FIXP* permite que esses controladores gerenciem os fluxos através da sua interface *South-Bound*, que abstrai a complexidade da camada de infraestrutura física presente no plano de dados, oferecendo uma API capaz de movimentar as tabelas de encaminhamento. Essa camada pertence ao plano de controle.

FPL contém a topologia de *swiches FIXP*. A Figura 15 apresenta apenas 1 switch para fins didáticos. A arquitetura *FIXP* não possui limitações para a quantidade de *switches* existentes na rede. Caso o *switch FIXP* reconheça a primitiva de ingresso, ele vai repassá-la para a interface de saída conforme a configuração das tabelas de encaminhamento. No caso da primitiva ainda ser desconhecida, uma das ações que pode ser tomada pelo *switch FIXP* é repassá-la para o controlador correspondente. Essa camada pertence ao plano de dados.

FAL tem a função de *broker*, ou seja, uma camada que intermedia a comunicação entre dispositivos de naturezas distintas. Esse *broker* é a própria interface *Southbound* do protocolo *FIXP*. Através dessa interface é possível fazer a comunicação em ambas

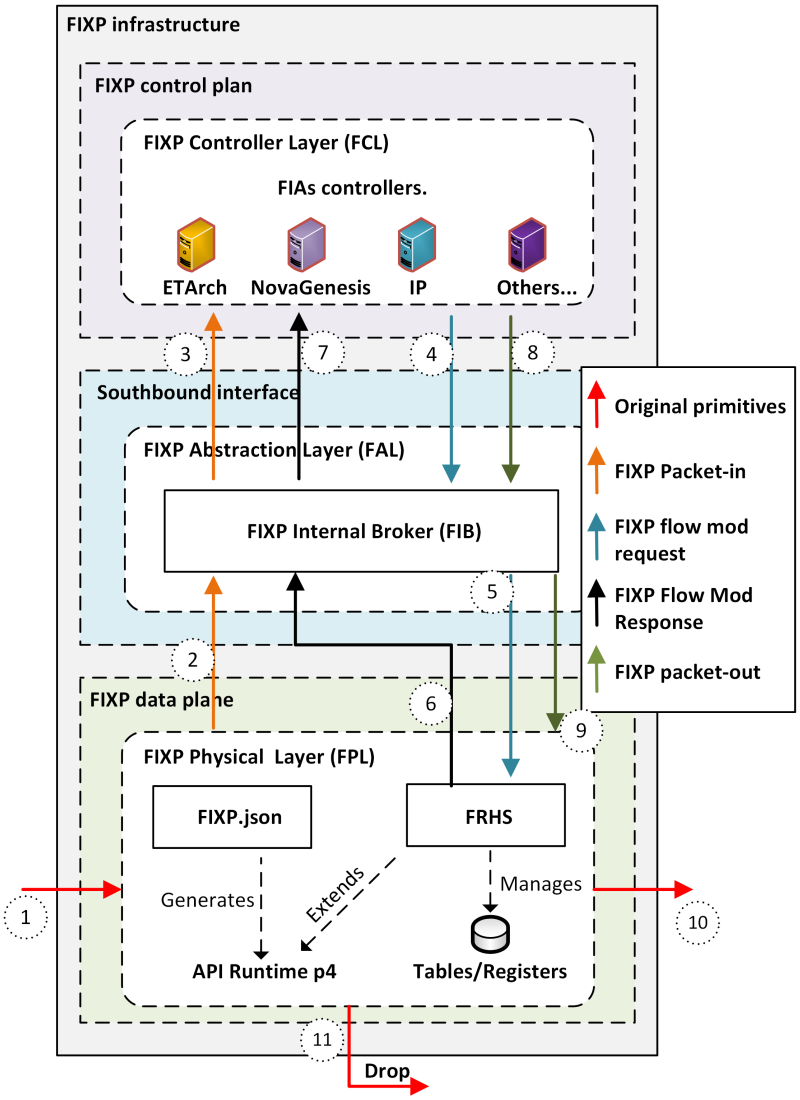

Figura 15 – Visão geral da arquitetura FIXP. Ilustração da comunicação interna entre os componentes funcionais. Adaptado de (GAVAZZA et al., 2020; SILVA et al., 2022)

as direções, ou seja, o controlador consegue requisitar serviços para o switch, como por exemplo, incluir uma entrada em uma tabela de encaminhamento; mas também o switch consegue se comunicar com o controlador, como por exemplo, enviar uma primitiva de resposta (primitiva *FIXP ACK/NAK*) ao controlador.

3.3 Comunicação interna dos componentes funcionais da arquitetura FIXP

A comunicação dos componentes internos da arquitetura *FIXP* se dá da seguinte forma (ver Figura 15).

A primitiva da *FIA* chega à *FPL* (1). Essa primitiva pode transportar dados/recursos

Tabela 3 – Formato de primitivas do protocolo FIXP. Apresentando alguns exemplos.

Header 1 - The Ethernet frame is present in all FIXP protocol primitives.		
Fields	**Content**	**Description**
Destination Address	FIXP	The primitive's destination address is the FIXP network.
Source Address	DTSA	The primitive's source address is DTSA, controller of the ETArch architecture.
Type	0900	Type allows the FIXP network to recognize its services. 0900 is the hexadecimal identifier of the FIXP protocol.
Data	FIXP services	The payload contains the Ethernet protocol data. This protocol carries one or more FIXP services (requests or responses). Each header listed below represents one of the FIXP protocol services. The contents of the "source address" and "destination address" fields determine whether the encapsulated primitive is a response or a request. In the current example, the DTSA requests a service from the FIXP network. If the contents of these fields were invertible, this would indicate a response from the FIXP network to the DTSA.
Header 2 - Protocolo FIXP-FORWARDING-TABLES-FLOW-MOD-REQUEST.		
Fields	**Content**	**Description**
Architecture ID. (1*)	0880	Identifies the architecture requesting a FIXP service. 0880 is the hexadecimal identifier of the ETArch.
Request ID. (2*)	1	Identifies the request for the primitive. Each request/response pair has the same identifier.
Switch (3*)	s01	Identifies the FIXP switch that will execute the request.
Thrift-port (4*)	9090	Identifies the port where the FIXP switch is running. A virtual machine can configure multiple FIXP switches simultaneously on different thrift-ports.
Command ID. (5*)	1	Identifies the operation that the FIXP switch performs. (1) The FIXP switch moves forwarding table entries.
Table name	etarch_forward	Name of the table where the FIXP switch maintains the forwarding entries.
Match key list	["CC:65:00:12:34:34"]	Each item in the list defines a match key in the forwarding table entry.
Action list	[["etarch_SetSpec_Group"]]	Each item in the list defines an action in the forwarding table entry.
Parameter array	[[1]]	Each item in the array represents a list of parameters for a given action. In this example, 1 is the parameter of the "etarch_SetSpec_Group" action.
Operation status	0	(0) Adding an entry to the forwarding table. (1) Modifying an entry in the forwarding table.
Header 3 - Protocolo FIXP-MULTICAST-GROUP-FLOW-MOD-REQUEST.		
Fields	**Content**	**Description**
Architecture ID.	0880	Similar to the field (1*).
Request ID.	1	Similar to the field (2*).
Switch	s01	Similar to the field (3*).
Thrift-port	9090	Similar to the field (4*).
Command ID. (6*)	2	Identifies the operation that the FIXP switch performs. (2) The FIXP switch creates or modifies multicast groups.
Group ID.	-1	Identifies the group number. If it is different from -1, the FIXP protocol uses this field to create the group identifier. Otherwise, the FIXP protocol creates a number for the group and sends a response to the controller, revealing the created identifier.
Group data array (7*)	[[-1, [2,3,4], -1, -1]]	Field format: [[rid, port list, node handle, handle associate]]. The FIXP protocol binds the egress ports [2, 3, 4] to the identifier of the created group. Once the switch activates the group, it sends a response that the controller uses to update the "rid" and "node handle". "Handle associate" is no longer used.
Operation status	0	(0) Adding a multicast group. (1) Modifying a multicast group.
Header 4 - Protocolo FIXP-FORWARDING-TABLES-FLOW-MOD-RESPONSE		
Fields	**Content**	**Description**
Architecture ID.	0880	Similar to the field (1*).
Response ID.	1	Similar to the field (2*).
Switch	s01	Similar to the field (3*).
Thrift-port	9090	Similar to the field (4*).
Command ID.	1	Similar to the field (5*).
Handle list.	[1]	Each item in the list represents the return ID (handle) of the forwarding table entry that the FIXP switch adds or modifies.
Response status	0	Identifies the response to a maintenance request on the forwarding table entries. (=0) success. (≠0) failure.
Header 5 - Protocolo FIXP-MULTICAST-GROUP-FLOW-MOD-RESPONSE		
Fields	**Content**	**Description**
Architecture ID.	0880	Similar to the field (1*).
Response ID.	1	Similar to the field (2*).
Switch	s01	Similar to the field (3*).
Thrift-port	9090	Similar to the field (4*).
Command ID.	2	Similar to the field (6*).
Group data array	[[1, 2]]	[[rid, handle node]]. Each item in the list is the return of the "rid"/"handle node" of a group that the switch creates. See more at (7*).
Group handle	1	Returns the identifier (handle) of the group that the FIXP switch creates.
Response status	0	Identifies the multicast group maintenance response. (=0) success. (≠0) failure.

(primitiva de dados) ou ser uma sinalização de controle (primitiva de controle).

FPL, no primeiro momento, tenta reconhecer a primitiva. Se a primitiva pertence a uma *FIA* que participa da comunicação *FIXP*, ela pode seguir dois caminhos: *FPL* encapsula a primitiva (*FIXP packet-in*) e envia-a para o controlador (2) ou repassa-a para seu destino final (10). No primeiro caso, *FPL* não encontra correspondência nas tabelas de encaminhamento ao pesquisar os identificadores de localização da primitiva. No segundo caso, *FPL* encontra essa correspondência e executa a ação correspondente. Quando a primitiva não pertence a uma *FIA* conhecida pelo *FIXP*, o *switch* descarta-a (11).

A primitiva chega em FAL (2). Só participantes devidamente registrados podem participar da comunicação *FIXP*. Caso o identificador da entidade esteja cadastrado, FAL envia a primitiva para o controlador correspondente (3), que desencapsula e analisa a primitiva original. Cada controlador de cada *FIA* realiza suas próprias ações. Geralmente, controladores proativos, enviam uma ou mais requisições para movimentar (incluir/deletar/alterar) as entradas das suas tabelas de encaminhamento nos *switches* da *FPL* (4). Nesses casos, o controlador encapsula essas requisições através da interface *Southbound* do *FIXP*. O serviço do *FIXP* que realiza essa operação é *FIXP Forwarding Tables Flow-Mod Request* (ver Tabela 3).

FAL analisa se a requisição pertence a um controlador que está registrado na rede. Em caso afirmativo, *FAL* envia a requisição para um ou mais *switches* da topologia (5). Um algoritmo de roteamento, por exemplo, pode requisitar a modificação de tabelas de encaminhamento de vários *switches* presentes no domínio administrativo.

O módulo *FIXP Rule Handle Service (FRHS)* é um serviço do *FIXP* que estende a *API RuntimeP4* (WORKING, 2024) e possibilita que controladores gerenciem as suas tabelas de encaminhamento. Ele recebe a requisição (5), desencapsula-a e executa a tarefa determinada. Logo após, envia uma primitiva de resposta (*ACK/NAK*) para o controlador (6, 7). O serviço do FIXP que realiza essa operação é *FIXP Forwarding Tables Flow-Mod Response* (ver Tabela 3).

Se *FRHS* realiza a operação com sucesso, novamente, cada controlador toma suas próprias decisões. Se a primitiva original é de controle, trata-se de uma solicitação enviada ao controlador por alguma entidade comunicante. O controlador, então, geralmente, responde essa solicitação (8, 9), encapsulando-a em uma primitiva *FIXP* (*packet-out*), que possui três informações adicionais: arquitetura encapsulada; nome do *switch* e número da porta de egresso para onde a resposta está sendo enviada (ver Tabela 3). O *switch FIXP* desencapsula a primitiva original e envia-a para o destino correspondente (10). Essa operação de resposta pode representar, por exemplo, o *handshake* feito por algumas arquiteturas que estabelecem uma conexão de controle antes da comunicação de dados.

Voltando um pouco: se *FRHS* realiza com sucesso as operações nas tabelas de encaminhamento, mas a primitiva original é de dados, o controlador tem que reinserir essa

primitiva novamente na rede. De forma análoga ao descrito anteriormente, o controlador encapsula a primitiva de dados (*FIXP packet-out*) e envia-a para o *switch* correspondente (8, 9), que consegue reinseri-la na rede e enviá-la para o seu destino final (10). Diferentemente da situação anterior, não se trata de um *handshake*, mas de enviar a primitiva de dados original para o receptor da informação.

Caso o *FRHS* não consiga realizar com sucesso as operações nas tabelas de encaminhamento, cada controlador tem a liberdade de tomar suas próprias ações, como por exemplo, reenviar a solicitação novamente para a rede *FIXP*.

3.4 Serviços e formatos das primitivas do protocolo FIXP

O protocolo de controle do *FIXP* possui diversas primitivas diferentes, cada uma com uma função específica. O conjunto desses serviços visa fornecer uma interface *Southbound* para que controladores de *FIAs* e switches *FIXP* comuniquem-se entre si sem se preocupar com a complexidade que essa comunicação envolve. Existem blocos funcionais do *FIXP* dentro dos switches e dentro da *FAL* que reconhecem as primitivas do *FIXP* e desempenham as funções determinadas, possibilitando que controladores determinem o comportamento do switch e dos fluxos referentes às suas primitivas.

A Tabela 3 apresenta o formato das principais primitivas e faz uma breve discussão de cada um de seus campos. Abaixo, faz-se uma descrição geral das funcionalidades de cada uma dessas mensagens, como também apresenta outras que aumentam as possibilidades de comunicação dos controladores com suas entidades comunicantes.

FIXP-PACKET-IN. Quando a primitiva chega no switch, uma das decisões que esse dispositivo pode tomar é enviá-la ao controlador correspondente. O *FIXP* realiza esse repasse através da primitiva de controle *FIXP-PACKET-IN*, que encapsula a primitiva original da *FIA* e envia-a para o controlador. Os cabeçalhos de *FIXP-PACKET-IN* contém metadados da comunicação e depende da primitiva encapsulada. Alguns dos metadados possíveis são: porta de ingresso da primitiva, tamanho da primitiva e data/hora de ingresso.

FIXP-PACKET-OUT. Quando o controlador recebe um *FIXP-PACKET-IN*, pode tratar-se de uma requisição enviada por uma das entidades das *FIAs* participantes. Ele responde a essa requisição através de um *FIXP-PACKET-OUT*, que encapsula uma primitiva de resposta, no caso de uma operação de *handshake*, ou encapsula uma primitiva de dados para reinseri-la na rede *FIXP*. Os cabeçalhos de *FIXP-PACKET-OUT* possuem três informações adicionais: *FIA* encapsulada, *switch* que recebe a resposta e a porta de egresso desse *switch*. Quando o *switch* recebe o *FIXP-PACKET-OUT*, ele desencapsula a mensagem de resposta e envia-a para o destino correspondente.

FIXP-FLOW-MOD-REQUEST. Quando o controlador tem que gerenciar o fluxo das suas mensagens, ele utiliza essa primitiva para configurar as entradas das tabelas de encaminhamento e os grupos *multicast*. A configuração desses grupos fornecem canais lógicos de comunicação que distribuem o conteúdo para várias entidades simultaneamente. Existem duas variações dessa primitiva: *FIXP-FORWARDING-TABLES-FLOW-MOD-REQUEST*, que gerencia as tabelas de encaminhamento e *FIXP-MULTICAST-GROUP-MOD-REQUEST*, que gerencia os grupos mencionados. O formato dessas primitivas estão na Tabela 3.

FIXP-FLOW-MOD-RESPONSE. Quando o *switch* executa a requisição *FIXP-FLOW-MOD-REQUEST*, ele responde-a através de uma primitiva *FIXP-FLOW-MOD-RESPONSE*. A resposta possui campos que identificam o sucesso ou a falha da operação. No caso de sucesso, a primitiva também possui informações que revelam os identificadores dos objetos que foram criados ou alterados dentro do *switch*. Existem duas variações dessa primitiva: *FIXP-FORWARDING-TABLES-FLOW-MOD-RESPONSE* e *FIXP-MULTICAST-GROUP-FLOW-MOD-RESPONSE*. A primeira é uma resposta à manutenção nas tabelas de encaminhamento, a segunda é uma resposta à manutenção de grupos *multicast*. O formato dessas primitivas estão na Tabela 3.

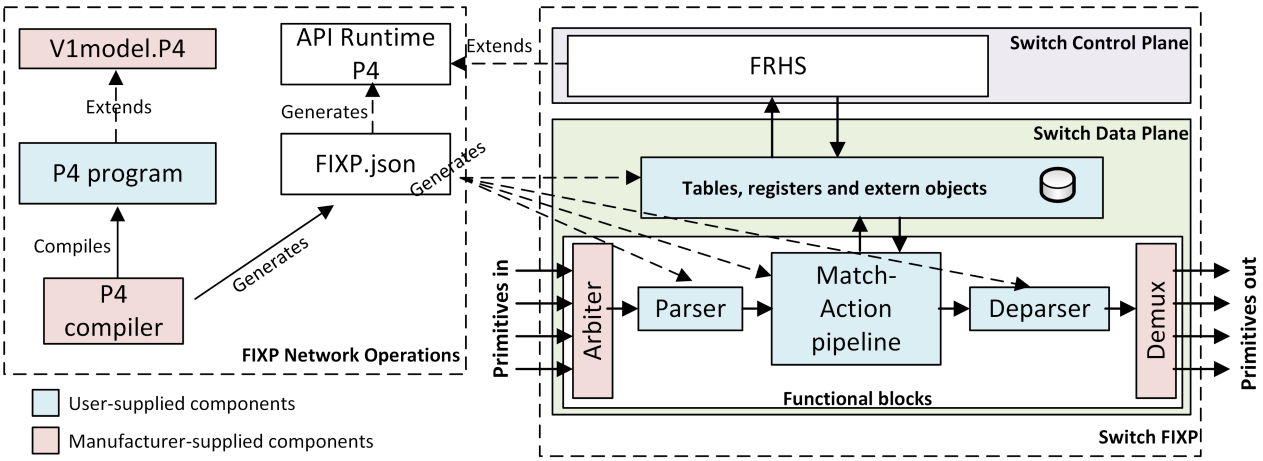

Figura 16 – Programando o plano de dados na rede FIXP. Adaptado de (LANGUAGE, 2023)

3.5 *Switch FIXP*

A Figura 16 mostra a flexibilidade do switch *FIXP* quanto à programação dos seus blocos funcionais. Isso permite à rede *FIXP* fornecer serviços que possibilitam a programação do *pipeline* de dados desse switch. Esses serviços permitem que cada *FIA* especifique, através de um programa *P4*, seus próprios objetos de controle no *pipeline* de dados do *switch* tais como tabelas de encaminhamento, ações e registradores que armazenam o estado da comunicação.

A compilação desses programas *P4*, de cada *FIA*, gera artefatos importantes: um deles é o arquivo *FIXP.json*. Através desse arquivo, o *switch FIXP* cria os objetos de controle e estabelece o funcionamento lógico do pipeline de dados que cada *FIA* necessita para a comunicação; e, além disso, cria uma *API RuntimeP4* personalizada que o módulo *FRHS*, presente no plano de controle do switch, estende.

FRHS é o módulo que possibilita aos controladores gerenciar o fluxo de suas primitivas através da manutenção das tabelas de encaminhamento e da criação de grupos de comunicação *multicast*. A Tabela 3 e a Seção 3.4 descrevem as primitivas que esse módulo reconhece.

Segue abaixo a descrição dos principais blocos funcionais do switch *FIXP* (LANGUAGE, 2023).

Arbiter block (bloco de função fixa). Recebe uma primitiva de uma porta *Ethernet*, da unidade do plano de controle do *switch* (CPU) ou da porta especial *Recirculate*. Ele possui um algoritmo de arbitragem interno que gerencia o enfileiramento nas portas de entrada do *switch*. Caso não tenha espaço nas filas para receber a primitiva, ela pode ser descartada. Outra função importante desse bloco é atribuir o valor da porta de ingresso ao metadado da primitiva.

Parser block(bloco programável). A rede *FIXP* permite que cada controlador configure, nesse bloco funcional, como o *switch* reconhece os cabeçalhos das suas primitivas, identificando, dessa forma, seus protocolos. O modelo de programação é de uma máquina finita, em que cada estado representa a extração de um conjunto de campos que compõe o cabeçalho de apenas um protocolo. A transição de estado continua até que o *Parser* termine a extração de todos os protocolos e envie as informações coletadas para o bloco *Match-action pipeline*. *Parser* também preenche alguns metadados da primitiva, como por exemplo, o tamanho do *payload* restante, que acaba sendo, nesse caso, o tamanho da mensagem/dados brutos.

Match-Action pipeline block(bloco programável). Através desse bloco, a rede *FIXP* permite que cada controlador crie e configure suas próprias tabelas de encaminhamento, chaves de correspondência e ações. Nessa etapa do pipeline, esse bloco já possui as informações enviadas pelo *Parser*, como o conteúdo dos cabeçalhos e alguns metadados. Nesse momento, o *switch*, de acordo com a programação feita no bloco, pode, por exemplo, pesquisar se o identificador do endereço de destino da primitiva está na tabela de encaminhamento. Caso esteja, ele executa a ação configurada; como por exemplo, enviar a primitiva para a porta de egresso correspondente. Caso não esteja, ele pode executar uma outra ação, como por exemplo, enviar a primitiva ao controlador correspondente. O módulo *FRHS* do *FIXP* fornece as interfaces necessárias para que cada controlador gerencie o fluxo de suas primitivas, permitindo manutenção das entradas das tabelas de encaminhamento e configuração de grupos *multicast*.

Deparser block(bloco programável). A rede *FIXP* permite que o controlador programe

esse bloco, especificando uma sequência de cabeçalhos que o *switch FIXP* utiliza, posteriormente, no bloco *Demux*, para montar a primitiva novamente. Este bloco programável pode, se for necessário, retirar ou adicionar cabeçalhos da primitiva original.

Demux block(bloco de função fixa). Esse bloco recebe a sequência de cabeçalhos de *Deparser* e a carga útil do *Parser*, monta uma nova primitiva e envia-a para a porta de saída correta. Essa porta depende do valor de alguns metadados atribuídos à primitiva pela programação do bloco Match-Action, tais como porta de egresso ou grupo *multicast*. Se a porta de egresso possui valor 1, *Demux* envia a primitiva para porta de egresso 1. Se o valor do grupo *multicast* é 2, e esse grupo está vinculado às portas 1, 2 e 3; então *Demux* replica a primitiva e envia-as para as portas correspondentes. Em alguns casos, *Demux* pode descartar a primitiva; quando por exemplo, a fila de interface de saída estiver cheia ou o número da porta de egresso for inválida.

Dois comentários sobre o switch FIXP merecem destaque.

O primeiro comentário é que os casos de uso dessa tese, apresentados no Capítulo 4, elaboram experimentos, em que cada máquina virtual (representante do switch FIXP físico - hardware) instancia apenas 1 switch FIXP virtual (extensão do BMv2 Simple Switch target - software) (LANGUAGE, 2024). Uma outra forma de pensar a infraestrutura física do FIXP está relacionada à criação de múltiplas instâncias de switch virtual em um único switch físico, de tal forma que as redes lógicas estabelecidas fossem isoladas e independentes. Dessa forma, cada rede lógica da topologia poderia ser consumida por uma determinada FIA, como também poderia ter planos de controle e dados independentes. Essa forma de pensar a infraestrutura FIXP como um conjunto de redes lógicas executadas sobre uma infraestrutura física que possui planos de dados e controle independentes, programáveis, extensíveis e configuráveis; é um ideia promissora para as redes da próxima geração. As consequências dessa rede lógica é objeto de um estudo futuro, e está além do escopo dessa tese. A atualização do pipeline de dados, por exemplo, com o switch em execução; poderia ser facilitada com a utilização de uma rede lógica para cada FIA implantada.

O segundo comentário é o seguinte. A tecnologia de enlace da infraestrutura *FIXP* é Ethernet, porém, o *FIXP* não utiliza a semântica dos campos *Media Access Control (MAC)* da maneira tradicional. Os conteúdos dos campos *MAC* de origem e destino não representam endereços de interfaces de rede (ver Tabela 3). *FIXP* utiliza uma *API* de comunicação que abstrai o esquema de endereçamento da tecnologia de transmissão, e encaminha suas primitivas utilizando um label qualquer, um nome que representa a interface de rede (*Network Interface Controller (NIC)*). Dessa maneira, a comunicação *FIXP* não depende de protocolos tradicionais para realizar sua comunicação, podendo modificar essa tecnologia para outra qualquer que esteja disponível nos dispositivos de comutação.

Findados os comentários, é bom salientar que as partes principais do documento ou

programa *P4* que especificam parcialmente o plano de dados da ETArch encontram-se no Apêndice A. O intuito desse apêndice é apresentar um pouco dos recursos da linguagem descritiva P4; e demonstrar que o código que promove o gerenciamento do pipeline de dados do *switch FIXP* não sobrecarrega o *switch* em termos de processamento, armazenamento e utilização de memória *RAM*. Na verdade, trata-se de um programa que descreve o funcionamento de operações que existem em quaisquer dispositivos de comutação, sendo eles programáveis ou tradicionais.

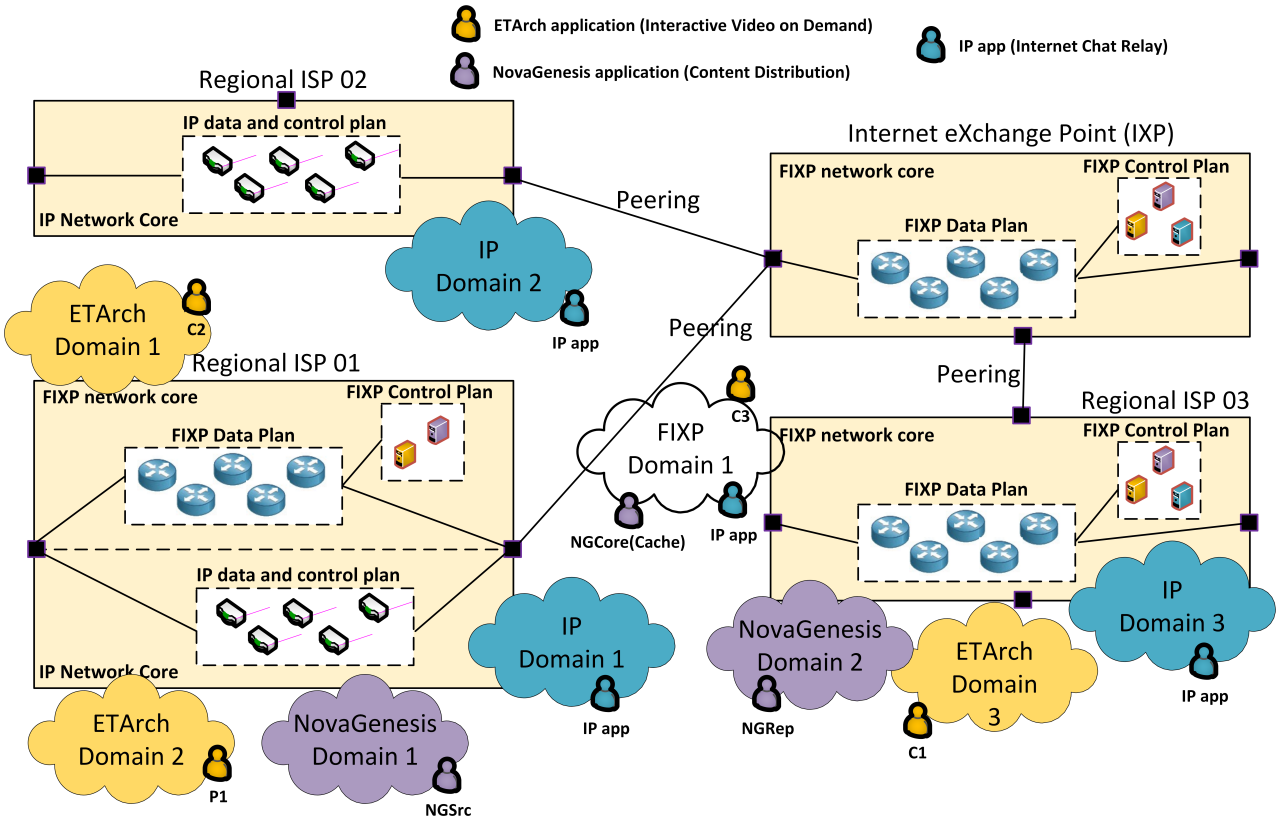

Figura 17 – Implantação da rede FIXP nas redes atuais.

3.6 Implantação do FIXP nas redes atuais

A Figura 17 representa a implantação do *FIXP* nas redes atuais. Trata-se de uma rede heterogênea, onde vários domínios de *FIAs* distintas comunicam-se entre si da seguinte forma: aplicações ETArch comunicam-se com aplicações ETArch, aplicações IP comunicam-se com aplicações IP, aplicações *NovaGenesis* comunicam-se com aplicações *NovaGenesis*, aplicações multiarquitetura comunicam-se com aplicações multiarquitetura, e assim por diante.

FIXP é o ponto de troca lógico ou a rede única que permite essa comunicação. *ISPs*, *Internet eXchange Points (IXPs)*, *Points of Presence (PoPs)*, provedores de conteúdo, redes privadas, entre outros; podem implantá-lo em suas dependências para promover

comunicação de *FIAs* distintas em escala global. Na figura, um *IXP* estabelece o *peering* em três *ISPs* distintos: *ISPs* Regionais 01, 02 e 03.

Outro ponto importante na figura são os domínios. Os domínios ETArch 1, 2 e 3 possuem topologias de *switches* ETArch, ou seja, *switches SDN-OpenFlow*. Os domínios IP 1, 2 e 3 possuem topologias de *switches* tradicionais IP. Os domínios *NovaGenesis* 1, 2 e 3 possuem serviços que fazem o roteamento de primitivas *NG*. O domínio *FIXP* possui topologia de *switches FIXP*, ou seja, *switches* que estendem *targets SDN-P4*. Cada domínio executa suas próprias aplicações e possui seus próprios serviços e dispositivos de comunicação, ou seja, domínios ETArch executam aplicações ETArch, domínios IP executam aplicações IP, e assim por diante. O domínio *FIXP* é o único que consegue executar aplicações de arquiteturas de Internet distintas. No caso da Figura 17, o domínio *FIXP* 1 executa aplicações ETArch, *NovaGenesis* e IP.

A aplicação P1 da ETArch, que está no domínio ETArch 2, é um produtor de *streaming* de vídeo *on demand*. Ele fornece vídeo para todos os consumidores que fazem parte do canal lógico de comunicação da ETArch, denominado *workspace*. Os consumidores C1, C2 e C3 da ETArch, presentes nos domínios ETArch 3, ETArch 1 e *FIXP* 1, respectivamente; são participantes da comunicação e recebem o vídeo de P1. Para participar de uma comunicação da ETArch, é preciso solicitar o recurso. A comunicação é *multicast* e as primitivas da ETArch replicam-se, no núcleo da rede, para vários destinos.

A aplicação de distribuição de conteúdo do *NovaGenesis* está utilizando os dois domínios *NovaGenesis* e o domínio *FIXP*. O produtor de conteúdo *NG* Source (NGSrc) descobre um *NG* Repository (NGRep) e um *NG* core *NRNCS* (NGCore) para armazenar fotos. NGRep é a entidade que vai assinar o conteúdo que NGSrc produz. NGCore é uma entidade de cache da rede que armazena temporiamente o conteúdo, representando um conceito *ICN*. O paradigma de comunicação dessa aplicação é *Publish/Subscribe*.

A aplicação de chat do IP simula o protocolo Internet Relay Chat (OIKARINEN, 1993). Quando um dos aplicativos IP envia a mensagem, todos os outros participantes que pertencem ao canal de comunicação enxergam-na. Os apps estão sendo executados no domínio *FIXP* 1 e nos domínios IP 1, 2 e 3. A comunicação é *unicast*, portanto, para que uma entidade IP envie mensagens para os outros participantes, a replicação das mensagens ocorre no sistema final (*endpoint*).

Do ponto de vista da implantação, a figura exibe quatro agentes que oferecem serviços de Internet. *IXP* possui um ponto lógico de troca ou uma rede local FIXP instalada em sua dependência, com três controladores SDN; que suportam as arquiteturas ETArch, *NovaGenesis* e IP. Dessa forma, o *IXP* consegue repassar as mensagens dessas arquiteturas de Internet distintas para os ISPs 01, 02 e 03. O ISP 01 também possui a implantação do FIXP, mas com apenas 02 controladores: ETArch e *NovaGenesis*. Isso quer dizer que esse ISP trafega mensagens da ETArch e *NovaGenesis* através do núcleo FIXP e mensagens IP através do plano de dados tradicional. O ISP 02 não possui o FIXP instalado em suas

dependências. Isso quer dizer que trata-se de um ISP tradicional, que reconhece apenas a arquitetura de Internet legada. O ISP 03 possui apenas um núcleo FIXP, com três controladores: ETArch, *NovaGenesis* e IP. Isso quer dizer que o FIXP, nesse provedor, já é responsável por todo o tráfego realizado. A avaliação da implantação do FIXP ocorre no Capítulo 4, porém, esta figura já apresenta, detalhadamente, como a rede FIXP possibilita a comunicação das FIAs integradas sem que os domínios administrativos das suas respectivas arquiteturas contenham a infraestrutura FIXP em suas dependências. Em resumo, o FIXP permite que a implantação da sua infraestrutura seja gradual, até que a rede se torne única e possibilite a comunicação em escala mundial.

Do ponto de vista da comunicação, esse cenário é bastante complexo para os dias atuais. Uma arquitetura *SDN* orientada a workspace (ETArch), uma arquitetura orientada a serviços com princípios de *ICN* (*NovaGenesis*) e uma arquitetura orientada a host (IP) coexistem dentro de uma rede única que possibilita a comunicação entre seus aplicativos. Dessa forma, aplicações atuais podem escolher a arquitetura que melhor atende os requisitos necessários para uma boa qualidade de comunicação.

3.7 Integrando arquiteturas de Internet distintas à rede *FIXP*

Esta seção explana, de forma detalhada, a integração das arquiteturas IP, ETArch e *NovaGenesis* na infraestrutura de rede do *FIXP*.

A Subseção 3.7.1 apresenta, brevemente, as diferenças arquitetônicas das arquiteturas selecionadas. A Subseções 3.7.2, 3.7.3 e 3.7.4 descrevem os procedimentos operacionais e a cronologia das trocas de mensagens que as arquiteturas ETArch, IP e *NovaGenesis*; respectivamente, realizam quando se integram ao *FIXP*.

Apenas uma ressalva: a integração das arquiteturas ETArch e IP são objetos de pesquisa dessa tese de doutorado. A integração da *NovaGenesis* é realizada por outro trabalho, que é devidamente referenciado na seção correspondente. A inclusão dessa integração, nesse trabalho, é importante, pois é um exemplo da incorporação, na rede FIXP, de serviços *clean-slate* que utilizam princípios de *ICN*. Esse fato torna-se importante, na medida que essa tese busca, no capítulo 4, a demonstração de que os princípios arquitetônicos do FIXP são flexíveis e possibilitam a absorção de funcionalidades de FIAs heterogêneas e disjuntas.

3.7.1 Breve descrição das arquiteturas selecionadas

As Subseções 2.1.6 e 2.1.7 apresentam a visão geral da ETArch e *NovaGenesis*, respectivamente. A Tabela 4 exibe, de forma detalhada, os princípios arquitetônicos dessas duas FIAs e do IP, já que representam as arquiteturas que se integram ao FIXP (plano de

Tabela 4 – Características das FIAs implantadas no FIXP atualmente.

Meta network architecture	Integrated architectures	Features
FIXP	ETArch	**Design orientation:** SDN-OpenFlow; Worspace-oriented communication. **Communication Paradigm:** Proactive (ETArch Controller Forwarding Rules Policy); Pull/Push. **Network Adresses:** Flat; Fixed lenght; Separation of Identifiers/Locators. **Network Protocols:** ETArch's protocols (ETCP e DTSCP). **Routing features:** Workspace-based multicast routing. **Switching element:** switch Openflow. **Security:** Identifiers are self-certifying.
	IP	**Design orientation:** Host-oriented. **Communication Paradigm:** Push. **Network Adresses:** Hierarquical identifier; Exclusive; Fixed length. **Network Protocols:** IP. **Routing features:** Supports host-based unicast communication. **Switching element:** IP-Dedicated Switch. **Security:** No security.
	NovaGenesis	**Design orientation:** SCN and Service-Oriented Architecture (SOA), Principles of ICN and Active Networks. **Communication Paradigm:** Push(Publish/Subscribe). **Network Adresses:** Flat (optional); Hierarchical (Optional); Variable length; Name Buildings (NB); Network addressing can have multiple interpretations; Natural Language Naming (Optional); Self-Verifying Name (SVN) via hash. **Network Protocols:** NovaGenesis Protocol. **Routing features:** NB-based routing; Stateful routing (in fragmentation scenarios). **Switching element:** It is a software (service) called PGCS. **Security:** Self-Verifying Name via hash.
	XIA(Prototype)	**Design orientation:** Principal-oriented (meta architecture (MACHADO; DOUCETTE; BYERS, 2015)). **Communication Paradigm:** Each principal has its communication paradigm. **Network Adresses:** Flat (optional); Hierarchical (optional); Variable length. **Network Protocols:** XIA protocol: eXpressive Internet Protocol (XIP) **Routing features:** Principals-based routing. **Switching element:** XIA-dedicated Switch. **Security:** Intrinsic, the identifier is self-certified.

dados e controle) até o momento da escrita desta tese. XIA está integrada parcialmente, no plano de dados; mas ainda não possui um controlador (SILVA, 2020). Por conta disso, essa integração não faz parte do escopo deste trabalho.

Esta subseção discute, brevemente, as características de cada uma dessas arquiteturas, apresentando algumas diferenças arquitetônicas e justificando-as, dessa forma, como arquiteturas selecionadas. As diferenças são cruciais para demonstrar, posteriormente, a flexibilidade do FIXP em aceitar novos protocolos e serviços de rede.

ETArch (GONÇALVES et al., 2014; SILVA et al., 2014) é uma arquitetura *SDN-OpenFlow* orientada a *workspace*, que é um canal lógico que oferece, intrinsecamente, uma comunicação multicast. A arquitetura separa os planos de dados e controle. *DTSA* é uma extensão de um controlador *SDN-OpenFlow* que executa os serviços de controle da ETArch. Cada *DTSA* gerencia um domínio de rede ETArch, e portanto, é capaz de gerenciar um conjunto de switches *OpenFlow* que faz parte do plano de dados desse domínio. Esses controladores são proativos, ou seja, configuram os switches *OpenFlow* antes que a entidade de comunicação receba o recurso solicitado. O paradigma de comunicação da ETArch é *Pull/Push*, dependendo da aplicação. As entidades da ETArch precisam enviar uma solicitação de controle para solicitar o recurso desejado. Como funciona? O produtor do recurso cria um *workspace* de comunicação que oferece, por exemplo, um *streaming* de vídeo interativo *on demand*. Para que uma entidade específica receba esse recurso de rede, ela tem que solicitar ao *DTSA* a sua admissão nesse *workspace*. Caso o *DTSA* decida pela admissão dessa entidade, ela começa a receber o recurso continuamente. O *DTSA* aloca um canal seguro de comunicação que fornece o streaming solicitado para a entidade em

questão. Se 10 entidades participam desse *workspace*, as 10 entidades recebem o mesmo conteúdo, o que torna o canal de comunicação intrinsecamente *multicast*. O protocolo ETCP possui diversos serviços e realiza a comunicação entre a entidade comunicante e o *DTSA*. O roteamento é baseado no *workspace*, que é o destino final da arquitetura. Esse processo de roteamento replica as primitivas no núcleo da rede para evitar sobrecarga de fluxos. O endereçamento ETArch é plano (*flat*), fixo (48 bits); e, separa identificadores de recursos das suas localizações.

A arquitetura IP (AGENCY, 1981; BRADNER; MANKIN, 1995; KUROSE, 2017) é orientada a *host* e o paradigma de comunicação é *Push*: a informação é enviada para um destino qualquer, sem solicitação prévia. O endereçamento de rede é hierárquico, exclusivo e contempla identificadores e localizadores no mesmo código IP, que é fixo, formado por 32 ou 128 bits, dependendo da versão. A comunicação intrínseca do IP é *unicast*, fim a fim (*end-to-end*). Caso o IP queira distribuir primitivas para várias entidades ao mesmo tempo (*multicasting*) utilizando *switches* padrões que não possuem o recurso IP *multicast* (DEERING; CHERITON, 1990), ele tem que replicar as primitivas na origem, podendo causar congestionamento no fluxo da rede. O roteamento é baseado no código IP, orientado a host; e, conta com auxílio de protocolos tais como *Open-Shortest Path First (OSPF)*, *Routing Information Protocol (RIP)* e *Intermediate-System-to-Intermediate-System (IS-IS)*.

NovaGenesis (ALBERTI et al., 2019) é uma rede baseada em serviços que se comunicam a partir de oferta e demanda de recursos, contendo princípios de *ICN* e *SOA*. Além disso, as primitivas carregam ações, baseadas em linhas de comandos *NG*, que influenciam a configuração dos elementos de rede. Por isso, também contém princípios de redes *Active Networks (ANTS)*. A comunicação do *NovaGenesis* ocorre graças a um de seus principais serviços: o *PSS*. Através dele, o *NG* assina e publica recursos de rede por meio do paradigma de comunicação *Push*(*Publish/Subscribe*). Quanto ao endereçamento de rede, há a separação entre localizadores e identificadores de recursos. Os endereços são compostos por tuplas de nomes (*NBs*) que vinculam identificadores de recursos às suas localizações. O perfil desses endereços depende da entidade comunicante: pode ter caráter plano (*flat*), hierárquico, tamanho fixo ou variável; pode tratar-se de *NLNs* ou *SVNs*. Isso interfere, diretamente, no tamanho das primitivas NG, que é variável. O roteamento do *NG* é feito através de *software*, por intermédio do serviço *PGCS*, e tem algumas características peculiares: é baseado em *NBs* e armazena estado (*stateful*) em situações de fragmentação. Algumas aplicações do NG, como a apresentada nessa tese, integra o serviço *NRNCS*, que é um sistema de cache de rede, o qual armazena temporariamente a informação. O protocolo de rede do NG possui vários serviços, como por exemplo, *PGCS*, *NRNCS*, *PSS*, *GIRS* e *HTS*.

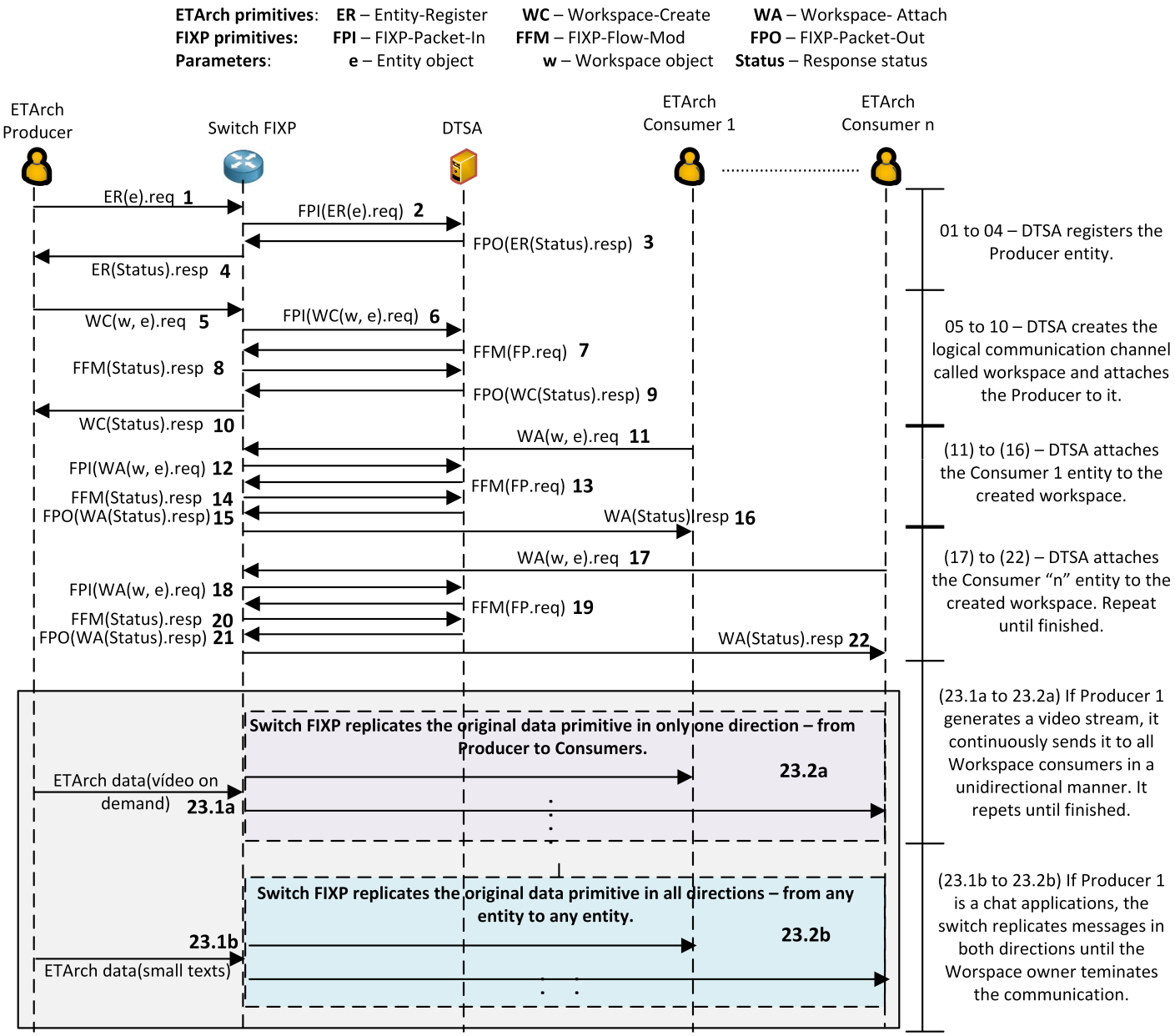

Figura 18 – Executando aplicativos ETArch na rede *FIXP*.

3.7.2 Integração da ETArch no *FIXP*

A arquitetura ETArch possui um controlador *SDN*, denominado *DTSA*, que gerencia a rede através do protocolo *OpenFlow*. O nível de dificuldade para integrar esse tipo de arquitetura na rede FIXP é baixo (ver Capítulo 4) e depende de duas ações: adequar a utilização da interface *Southbound* do *DTSA*, de *OpenFlow* para *FIXP*; e, descrever, através da linguagem *P4*, a lógica do pipeline de dados que define o comportamento do switch *FIXP*. Na primeira ação, o *DTSA* utiliza o framework da interface *Southbound* para se comunicar com a infraestrutura física da rede através das primitivas do protocolo *FIXP* que a Tabela 3 descreve. Esse procedimento possibilita ao controlador gerenciar os fluxos da primitiva ETArch. Na segunda ação, o autor da implantação configura o

funcionamento dos blocos funcionais do switch *FIXP* através de um programa *P4*, que define tabelas de encaminhamento, ações, chaves, registradores e outros objetos necessários para a comunicação dos aplicativos da arquitetura. Esse procedimento possibilita ao controlador ETArch programar o plano de dados e gerenciar o comportamento do switch *FIXP*, facilitando, por exemplo, a inclusão de novos serviços na camada de rede.

A Figura 18 exibe as trocas de mensagens entre aplicativos da arquitetura ETArch e a rede *FIXP*.

A ETArch é uma arquitetura proativa. Dessa forma, a entidade ETArch e o *DTSA* estabelecem a conexão através de um *handshake* (1 até 22.). Só depois desse processo, os aplicativos ETArch comunicam-se entre si. O *handshake* é padrão para qualquer caso de uso da ETArch, ou seja, as etapas 1-22 sempre ocorrem independentemente do aplicativo executado.

O estabelecimento da conexão se dá através de 03 etapas: registro da entidade proprietária do *workspace*; criação do *workspace* e vinculação da entidade proprietária a ele; vinculação das entidades consumidoras ao *workspace* criado. Nessa tese, o produtor representa a mesma entidade que cria o *workspace*, é quem produz o recurso. O consumidor utiliza os recursos da comunicação.

. *Registro da entidade proprietária do* workspace. O produtor envia uma mensagem ER.req para a rede *FIXP*, com o intuito de registrar-se como proprietário de um novo *workspace* (1). O switch *FIXP* recebe a mensagem, encapsula-a na primitiva *FIXP Packet-in (FPI)* e envia-a para o *DTSA*(2). *FPI* possui alguns metadados dessa primitiva, como por exemplo, a porta de ingresso e o switch que recebeu a requisição. O *DTSA* desencapsula a primitiva ER.req, reconhece-a e realiza o registro da entidade solicitante. Logo após, encapsula uma resposta, na primitiva *FIXP Packet-out (FPO)*, para a requisição recebida e envia-a para o *switch* adequado (3). *FPO* possui alguns metadados dessa comunicação, como por exemplo, o *switch* que recebe a resposta e a porta de egresso onde está a entidade solicitante. O *switch FIXP* recebe essa mensagem, desencapsula-a e envia a resposta da operação de registro para a entidade solicitante (4).

. *Criação do* workspace *e vinculação da entidade proprietária a ele.* O produtor já está registrado, então ele envia uma requisição WC.req para criar um novo *workspace* e vincular-se como proprietário desse canal recém-criado (5). O *switch FIXP* recebe a mensagem, encapsula-a na primitiva *FPI* e envia-a para o *DTSA*(6). O *DTSA* desencapsula a primitiva WC.req, reconhece-a; cria o *workspace* e vincula o produtor a esse canal. Para consolidar a criação do *workspace*, o *DTSA* envia uma requisição *FIXP FlowMod (FFM)* para que o *switch* movimente as tabelas de encaminhamento e os grupos de comunicação *multicast* que configuram esse canal de comunicação (7). O parâmetro FP.req, em *FFM*, representa um dos serviços/requisições/primitivas do *FIXP* que a Tabela 3 descreve. O *switch* responde a essa solicitação do *DTSA* (8), enviando como parâmetro o status da operação. O sucesso da operação faz com que o *DTSA* encapsule uma resposta para a

entidade solicitante na primitiva *FPO* (9). O *switch* recebe essa resposta, desencapsula-a e envia-a para a porta de egresso adequada, onde está a entidade solicitante (10).

. *Vinculação das entidades consumidoras ao* workspace *criado.* Similar à etapa anterior. A única diferença é que esta etapa apenas vincula entidades ao *workspace* criado. A primitiva WA possui essa função. Na figura, de 11 até 16, o *DTSA* vincula o Consumidor ETArch 1 ao *workspace* criado; de 17 até 22, o *DTSA* vincula "n" entidades ao *workspace* criado. Essa última operação representa a capacidade que o *DTSA* tem de vincular um número de entidades ilimitado à comunicação oferecida pelo canal. Depois que o *DTSA* conclui essas operações, todas as entidades ligadas ao *workspace* recebem o recurso da comunicação, que pode ser por exemplo, um streaming de vídeo ou um grupo de conversação similar a um chat.

Depois que a ETArch estabelece a conexão através das primitivas de controle, a comunicação de dados entre as entidades já pode acontecer. A figura revela dois casos de uso distintos. Se o produtor é um aplicativo de vídeo interativo *on demand*, ele envia esse tráfego para o *switch FIXP* (23.1a). O *switch* replica esse tráfego para todas as entidades que participam da comunicação (23.2a). Se o *workspace* possui 10 entidades, as 10 entidades recebem o vídeo do Produtor. Esse caso de uso replica as mensagens em apenas uma direção, do Produtor para os Consumidores. Se o produtor é um chat que possui um grupo de várias pessoas, cada pessoa é uma entidade. Nesse caso, o produtor envia uma mensagem (23.1b) e o *switch FIXP* replica essa mensagem para todos os participantes do grupo (23.2b). No caso do chat, o *switch* replica a mensagem em todas as direções, do Produtor para os Consumidores e dos Consumidores para o Produtor. Se o Consumidor 1 escreve uma mensagem, todas as outras entidades do grupo recebem-na, inclusive o Produtor. Esse envio não está presente no diagrama, pois ele é reduzido para fins didáticos.

3.7.3 Integração da arquitetura IP no *FIXP*

A arquitetura IP não possui um controlador *SDN*, portanto o nível de dificuldade para integrá-la ao *FIXP* é médio (ver Capítulo 4) e passa por três ações: fornecer um controlador *SDN* que possui as funcionalidades da arquitetura IP; adequar esse novo controlador à interface *Southbound* do protocolo *FIXP*; e, configurar o plano de dados programável e extensível da rede *FIXP* para reconhecer os serviços da arquitetura IP. Em relação à primeira ação, o autor da implementação tem que desenvolver aplicações de alto nível que refletem as funcionalidades da arquitetura IP, como por exemplo, desenvolver os algoritmos de roteamento que configuram as tabelas de encaminhamento do IP no domínio (*Autonomous System (AS)*). O controlador IP, que integra os casos de uso dessa tese, simula o algoritmo *intra*-AS *OSPF* (MOY, 1998). Em relação à segunda ação, o autor da implementação precisa integrar o controlador às interfaces que o *FIXP* oferece para se comunicar com a infraestrutura física da rede. A Tabela 3 descreve as principais

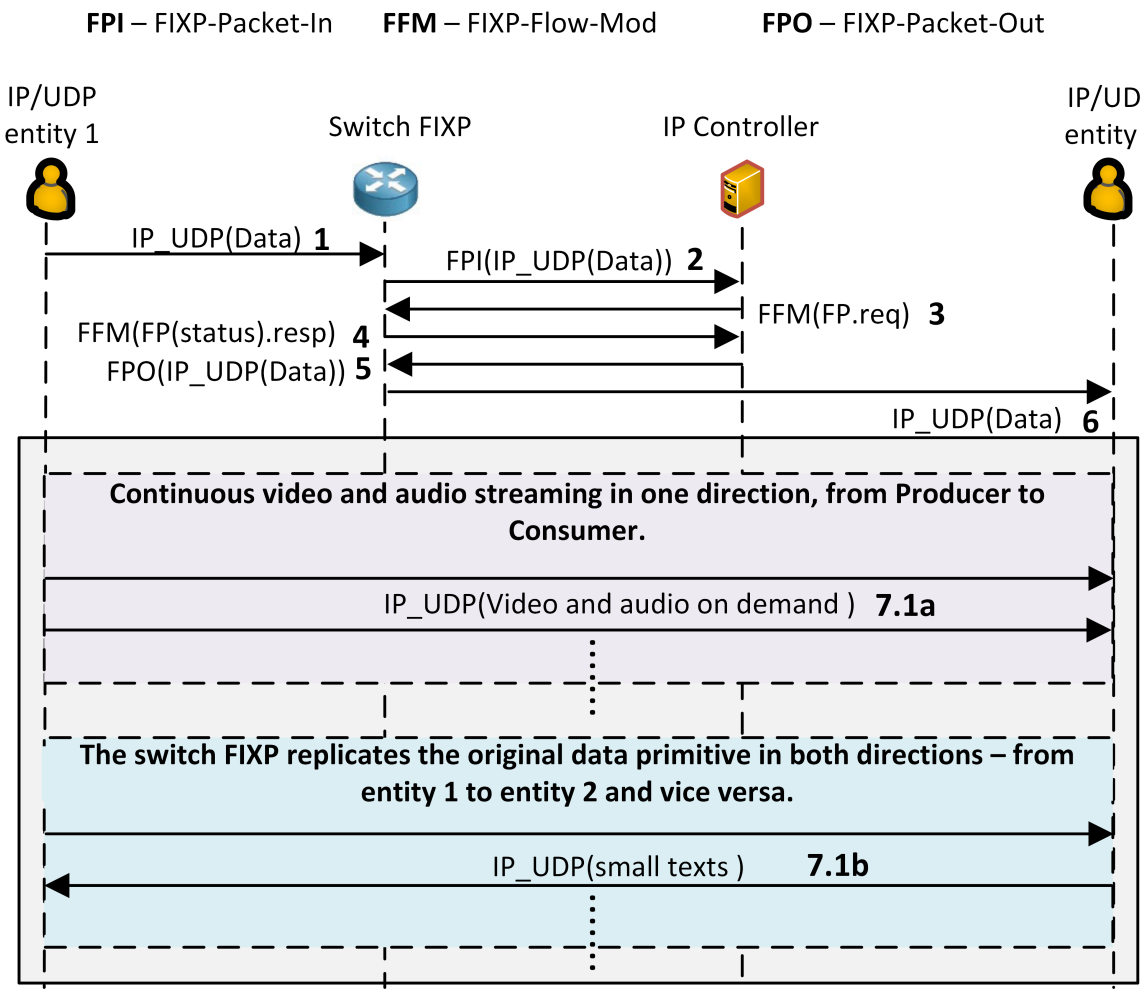

Figura 19 – Executando aplicativos IP na rede *FIXP*.

primitivas. Quanto à terceira ação, o autor da implementação tem que configurar os blocos funcionais programáveis do *switch FIXP* através de um programa *P4*. A rede *FIXP* possibilita a especificação da lógica de funcionamento do pipeline de dados pelo operador de rede. Nesse momento, cria-se tabelas de encaminhamento, chaves, ações, registradores, sequência de processamento; ou seja, a lógica de execução do processamento das primitivas e os objetos necessários para que a comunicação ocorra entre os aplicativos da arquitetura IP.

A Figura 19 descreve as trocas de mensagens entre aplicativos da arquitetura IP e a rede *FIXP*.

A entidade IP/UDP 1 envia um pacote de dados para a entidade IP/UDP 2 (1). Os casos de uso dessa tese utilizam o protocolo de transporte UDP para envio das informações. O *switch FIXP* recebe a mensagem (1), não reconhece o IP de destino da primitiva, e por conta disso, encapsula-a em *FPI* e envia-a ao controlador IP (2). O controlador executa o algoritmo *intra*-AS OSPF para determinar o menor caminho entre as entidades comunicantes. Logo após, envia a primitiva *FFM* para configurar as tabelas de encaminhamento

(3). O parâmetro FP.req em *FFM* é uma das requisições/primitiva/serviços que a Tabela 3 descreve. O *switch FIXP* realiza as operações na tabela de encaminhamento e envia a resposta da requisição atendida ao controlador IP (4). O parâmetro *Status* da resposta representa o sucesso ou a falha da operação. Com o sucesso da operação, o controlador IP encapsula a primitiva original, enviada anteriormente em (1), em *FPO* e envia-a para o *switch FIXP* (5). *FPO* contém os metadados da reinserção dessa primitiva de dados na rede, tais como o *switch* que realiza essa reinserção e a porta de egresso onde o receptor está. O *switch FIXP* recebe a requisição de reinserção (5), desencapsula a mensagem original e envia-a para a porta de egresso adequada, para o seu destino final (6).

O diagrama apresenta dois casos de uso. No primeiro (7.1a), a entidade IP/UDP 1 é um produtor de streaming de vídeo. Nesse caso, a entidade IP/UDP 1 envia tráfego de vídeo continuamente para a entidade IP/UDP 2, até que o vídeo acabe. A comunicação é unidirecional, da entidade 1 (produtor) para a entidade 2 (consumidor). Diferentemente da ETArch, o IP é intrinsecamente *unicast*, e por conta disso, o aplicativo de streaming só envia a primitiva de dados para uma entidade. Dessa forma, não há replicação das primitivas no núcleo da rede *FIXP*. No segundo caso (7.1b), as entidades IP/UDP 1 e IP/UDP 2 participam de um canal de chat que possui apenas 2 participantes. Quando a entidade 1 envia uma mensagem de texto, ela chega à entidade 2. O inverso também ocorre. Nesse caso, a comunicação é bidirecional e acontece até que o canal seja encerrado.

3.7.4 Integração da arquitetura *NovaGenesis* no FIXP

Esta seção apresenta a integração do *NovaGenesis* na rede *FIXP*, proposta realizada pela dissertação de mestrado (SILVA, 2021) e publicada em (SILVA et al., 2022).

A arquitetura *NovaGenesis* (ALBERTI et al., 2019; TELECOMUNICAçõES, 2021a) não tem um controlador *SDN* que possui as suas funcionalidades de controle e gerenciamento de rede. Nesse caso, o nível de dificuldade de integrar essa arquitetura na rede *FIXP* é médio (ver Capítulo 4) e depende de três ações: fornecer um controlador *SDN* que possui as funcionalidades de controle e gerenciamento da *NG*; adequar esse novo controlador à interface *Southbound* do protocolo *FIXP*; e, configurar o plano de dados programável e extensível do *FIXP* para reconhecer os protocolos, primitivas e serviços da *NG*.

Segue abaixo o detalhamento de cada uma dessas ações (SILVA, 2021; SILVA et al., 2022).

. *Fornecimento de um novo controlador* SDN-FIXP. Os autores da implementação desenvolvem o controlador *SDN-FIXP NovaGenesis Control Agent (NGCA)*, que estende as funcionalidades do *PGCS* (ver Subseção 2.1.7). É importante salientar que o serviço *PGCS* já existe na arquitetura tradicional (ALBERTI et al., 2019), ele é apenas transferido para um controlador *SDN-FIXP* (SILVA, 2021; SILVA et al., 2022). *NGCA* é um ponto de

comunicação que cria um ambiente de serviços distribuído capaz de promover encontros entre demanda e fornecimento de recursos.

. *Adequar o* NGCA *à interface* Southbound *do protocolo* FIXP. Os autores da implementação vinculam os aplicativos de controle e gerenciamento da *NG* à interface *Southbound* do *FIXP*. A partir dessa interface, o controlador *NGCA* utiliza o protocolo *FIXP* para coordenar a infraestrutura física do plano de dados em tempo de execução.

. *Configurar o plano de dados programável e extensível do* FIXP *para reconhecer protocolos, primitivas e serviços do NG*. A rede *FIXP* permite que o operador de rede, que pode ser o controlador *NG*, especifique o comportamento dos *switches* no plano de dados, definindo, no pipeline de dados do *switch FIXP*, a sequência lógica de processamento das primitivas da arquitetura. Nessa fase, há a criação de objetos que são necessários para comunicação tais como tabelas de encaminhamento, ações, chaves de encaminhamento, métodos de pesquisa, registradores de informação, entre outros. No caso da *NG*, as primitivas possuem tamanhos variados, possuem diferentes cabeçalhos, e nem todas carregam o endereço de destino (*Destination Host Identifier (DHID)*) da mensagem enviada. Em relação aos diferentes cabeçalhos, *NG* define dois cabeçalhos através de um programa P4: o primeiro cabeçalho, *novagenesis_hasDHID_t*, possui *DHID* e o segundo cabeçalho, *novagenesis_hasntDHID_t*, não possui *DHID*. Com relação a diferentes tamanhos de cabeçalhos, as primitivas *NG* possuem um campo denominado *MSG size*, que revela o tamanho total da mensagem original. *NG*, através do mesmo programa P4, consegue calcular, através desse campo e de delimitadores na primitiva, a localização de cada campo do cabeçalho e dos dados de controle que se encontram na mensagem (*payload*), como por exemplo, o endereço de destino (*DHID*). Em relação ao endereço de destino (*DHID*), no caso de fragmentação da mensagem, só o primeiro fragmento possui esta informação. Para resolver esse problema, a rede *FIXP* permite que o *NG* crie registradores próprios no *switch FIXP* que armazenam o estado da comunicação (roteamento *stateful*); de tal forma que essas informações são descartadas somente após o encerramento da conexão. Dessa forma, o *NG* é capaz de transferir suas funcionalidades de controle para uma peça de software externa (plano de controle programável), como também configurar o reconhecimento e a lógica de processamento do pipeline de dados das suas primitivas (plano de dados programável), possibilitando o reconhecimento de todos os seus serviços e o gerenciamento do fluxo das suas primitivas; sem que o controlador conheça a complexidade da infraestrutura física dos dispositivos de comutação.

A Figura 20 mostra a interação entre as primitivas *NG*, o *FIXP* e o *NGCA*.

As aplicações *NG* enviam suas primitivas para a rede (1). Tanto as primitivas de dados quanto as primitivas de controle contém o *DHID*. As únicas primitivas *NG* que não contém essa informação são as mensagens fragmentadas que o *FIXP* reconhece como se fosse o segundo cabeçalho (*novagenesis_hasntDHID_t*), ou seja, as mensagens fragmentadas a partir da sequência dois. A primeira análise do *switch FIXP* diz respeito à

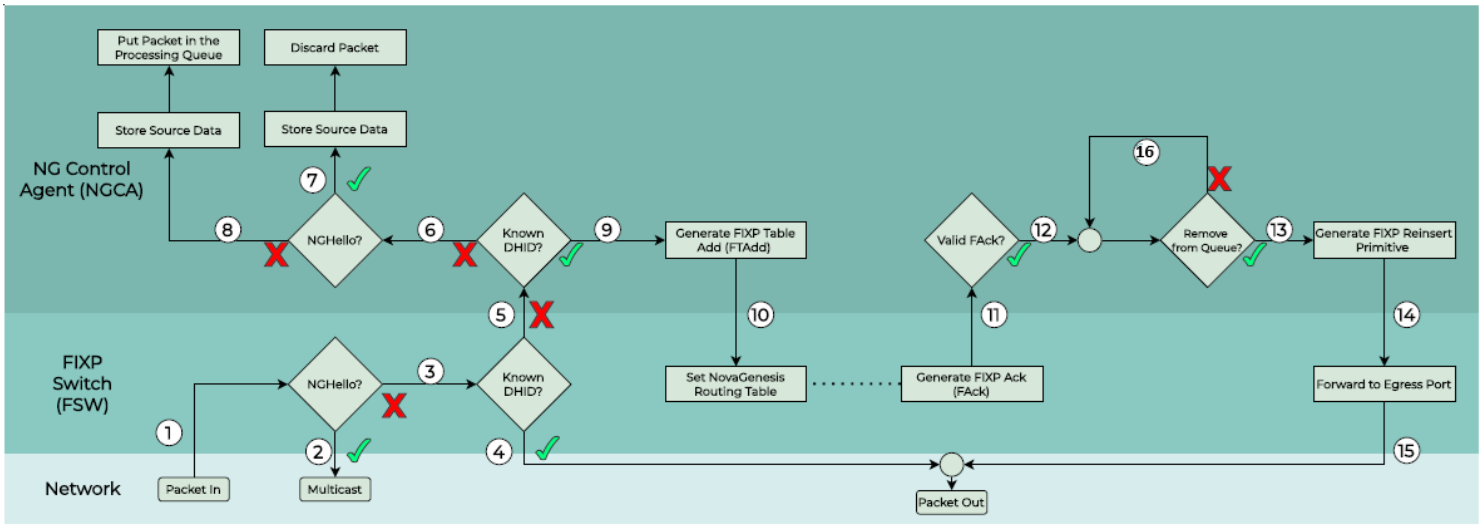

Figura 20 – Fluxograma do FIXP e o controlador NGCA. Fonte: (SILVA et al., 2022).

identificação do cabeçalho *NG*. Se o *switch FIXP* identifica o formato do primeiro cabeçalho (*novagenesis_hasDHID_t*), quer dizer que este cabeçalho possui *DHID*. Se o *DHID* possuir o endereço "FFFFFF" (primitiva '*NGHello*'), o *switch FIXP* faz um *multicasting* dessa mensagem (2) para todas as entidades *NG* conhecidas .A primitiva '*NGHello*' faz parte do processo de inicialização do domínio *NG* e é importante para promover o encontro entre os serviços distribuídos. Caso a mensagem não seja '*NGHello*' (3), o *switch FIXP* verifica se o *DHID* possui entradas na tabela de encaminhamento. Caso possua (4), o *switch FIXP* encaminha a primitiva para a porta correspondente. Caso o *DHID* não esteja inserida nas tabelas de encaminhamento (5), *switch FIXP* envia a primitiva ao controlador (5). Voltando um pouco; caso o *switch FIXP* identifique que o formato da primitiva é o segundo cabeçalho(*novagenesis_hasntDHID_t*), quer dizer que ela não contém o *DHID*. Isso quer dizer que a primitiva que está sendo analisada faz parte de uma fragmentação de sequência maior ou igual a 2. Nesse caso, o *switch FIXP* investiga o estado da conexão, ou seja, busca o *DHID* através do *MessageId* e envia a primitiva para a porta correspondente (4). Seguindo o raciocínio, a partir do passo (5).

A primeira análise do *NGCA* é consultar se o *DHID* da primitiva é conhecido. Para decidir sobre isso, ele pesquisa seu banco de dados. Se a primitiva possui *DHID* e se esse *DHID* está no banco de dados do *NGCA* (9), o controlador adiciona as regras necessárias nas tabelas de encaminhamento (10) e espera por uma resposta do *switch FIXP* (11). *NGCA* analisa a resposta e caso seja satisfatória (12), quer dizer que a inclusão de entradas nas tabelas de encaminhamento foi concluída com sucesso. Nesse ponto, *NGCA* coloca a primitiva em um *buffer* de envio, onde cada primitiva que recebeu um ACK satisfatório vai ser enviada para a porta de egresso adequada (14, 15) através da interface de reinserção da primitiva *FIXP* (13). Se a primitiva é reinserida no plano de dados (13, 14, 15), ela é removida do *buffer* de envio (16). Por outro lado, voltando um pouco, caso o *DHID*

não esteja no banco de dados do *NGCA*(6), quer dizer que o *NGCA* não reconhece o *DHID*, então ainda não sabe quais as regras deve acrescentar às tabelas de roteamento para realizar o envio da primitiva. Nesse caso, o *NGCA* verifica se o *DHID* é 'FFFFFF' e caso seja, identifica a primitiva *NGHello* (7). O controlador, nesse ponto, registra em seu banco de dados as informações do remetente e descarta a primitiva *NGHello*. Porque essa primitiva '*NGHello*' estaria no *NGCA* se ela já foi analisada no passo 1 do *switch FIXP*? Porque ela foi descartada? A resposta da primeira pergunta é que o *NGHello*, quando reconhecido pelo *switch FIXP*(2), ele fez um *multicasting*(2), então essa primitiva também é enviada ao controlador. O descarte é porque o *NGCA* não precisa inserir através do *FIXP* essas regras de grupo *multicasting* nas tabelas de encaminhamento, pois elas foram colocadas manualmente, logo após a inicialização dos switches FIXP, por uma decisão de projeto. É importante que essa primitiva chegue ao controlador devido ao enriquecimento do seu banco de dados, que cadastra as informações do remetente (7). Caso a primitiva não seja *NGHello* (8), o *NGCA* armazena as informações do remetente no banco de dados e coloca essa primitiva na fila para ser enviada posteriormente; quando o *NGCA* conhecer o seu *DHID*. Quando o *NG* reconhece o *DHID* das primitivas em espera, o processo é o mesmo que foi citado anteriormente. O *NGCA* configura as tabelas de encaminhamento através do *FIXP* (9, 10), espera o ack do *switch FIXP* (11), e caso o ACK seja favorável (12), isso quer dizer que as configurações das tabelas foram realizadas com sucesso. Depois disso. o *NGCA* reenvia as primitivas que estão no buffer de espera para a rede (14, 15) através da interface de reinserção do *FIXP* (13). Para cada primitiva reenviada (13, 14, 15), o *NGCA* a remove da fila de espera (16).

3.8 Uma breve discussão sobre o FIXP e seus objetivos arquitetônicos

Este capítulo apresenta a arquitetura *FIXP* e descreve conceitualmente cada bloco funcional, mostrando o comportamento lógico de cada um deles quando recebem as primitivas de controle e dados das arquiteturas de Internet.

O *FIXP* é uma rede única ou ponto lógico que possibilita a coexistência de aplicações de FIAs distintas; cuja comunicação possui algumas características que são inovadoras em relação às meta-arquiteturas apresentadas e amplamente discutidas nessa tese.

Em primeiro lugar, o *FIXP* permite a implantação gradual da sua infraestrutura em *IXPs*, *ISPs*, *PoPs*, entre outros agentes de telecomunicações. Nesse ínterim, o *FIXP* funciona como um ponto de troca lógico, que intercomunica FIAs distintas, mesmo que seus domínios administrativos tenham suas próprias infraestruturas de rede. A ideia é que a implantação desses pontos lógicos multipliquem-se nesses agentes e nos domínios administrativos das FIAs; até que se forme uma rede única, sem segmentação, que padro-

niza a comunicação entre entes comunicantes, que são heterogêneos e muitas das vezes, disjuntos.

Para cumprir o objetivo arquitetônico, *FIXP* propõe planos de dados e controle programáveis, extensíveis e configuráveis. Isso é um avanço frente aos trabalhos correlatos, pois possibilita e prepara a rede para novos serviços e aplicações, sem que os mecanismos utilizados causem problemas tais como sobrecarga dos dispositivos de rede em relação a processamento, armazenamento e memória; ineficiência da implantação de novas FIAS, que em alguns casos, causam perda de funcionalidades; e, dificuldade de implantação da solução (da meta-arquitetura) nas redes atuais. Esse ponto, dos mecanismos do FIXP não causar os mesmos problemas existentes nos trabalhos correlatos que possuem o mesmo objetivo arquitetônico do FIXP, é analisado profundamente no capítulo posterior.

No mais, o capítulo explana a integração de três arquiteturas de Internet distintas no *FIXP*: ETArch, *NovaGenesis* e IP; demonstrando, ainda de forma descritiva, que os princípios arquitetônicos do *FIXP* alcançam o objetivo geral dessa tese. Cada uma dessas arquiteturas possui suas próprias orientações de design, paradigmas de comunicação, esquemas de endereçamento, entre outros.

O Capítulo 4 realiza avaliações qualitativas que possui alguns objetivos claros: diferenciar as características do FIXP em relação aos trabalhos correlatos, posicionado-o no estado da arte; analisar a relevância dos princípios arquitetônicos do *FIXP* em relação às redes atuais e trabalhos relacionados; e, verificar a eficiência da implementação e implantação desses princípios arquitetônicos nas redes atuais. Além disso, faz a avaliação de performance da comunicação em relação a dois aspectos principais: quanto à infraestrutura *FIXP*: sobrecarregamento de processamento e memória; e, quanto às aplicações de FIAs que participam da comunicação, levando em conta a experiência do usuário. A partir dessas análises, esta tese começa a responder as suas questões de pesquisa, propostas na Subseção 1.6.

Capítulo 4

Avaliações e análise dos resultados obtidos

Este capítulo avalia o *FIXP* em quatro níveis (Seção 4.2). A primeira etapa, qualitativa e de cunho mais genérico, avalia as diferenças das características do FIXP em relação aos trabalhos correlatos que possuem o mesmo objetivo arquitetônico, posicionando-o no estado da arte (Subseção 4.2.1). A segunda etapa, qualitativa e singular, avalia especificamente a arquitetura: a relevância dos princípios arquitetônicos do *FIXP* em relação às necessidades de arquiteturas de Internet atuais e futuras (Subseção 4.2.2). A terceira etapa, qualitativa e singular, avalia especificamente a eficiência da implementação e implantação desses princípios arquitetônicos nas redes atuais (Subseção 4.2.3). A quarta etapa é quantitativa e mede a eficiência da comunicação no plano de dados (Subseção 4.2.4), levando em conta métricas que informam o sobrecarregamento da infraestrutura *FIXP* e a performance das aplicações que se comunicam nessa rede. Cada uma dessas avaliações possui seu próprio método (Seção 4.1) e visam demonstrar a hipótese e o cumprimento dos objetivos geral e específico propostos, respectivamente, nas Seções 1.5, 1.3 e 1.4. Logo após as avaliações, espera-se ter as informações necessárias para responder as questões de pesquisa (Seção 4.3) e para fazer algumas considerações finais sobre a arquitetura FIXP (Seção 4.4). Essas considerações visam discutir as contribuições do FIXP para as tendências das redes da próxima geração.

4.1 Método para avaliação

Esta tese executa duas abordagens de avaliação distintas: qualitativa e quantitativa. Em ambas as abordagens, é necessário classificar, ordenar e organizar os dados em classes ou categorias para serem analisados (MARCONI; LAKATOS, 2012). A organização desses dados está diretamente ligada às questões de pesquisa, hipóteses e objetivos do trabalho, que devem ser verificadas de forma analítica (MARCONI; LAKATOS, 2012).

A análise qualitativa é baseada na presença ou ausência de certa característica, e

também, na classificação de tipos diferentes de propriedades (MARCONI; LAKATOS, 2012). A análise quantitativa foca em termos de grandezas simbolizadas por um valor numérico e unidade de medida, como também podem ser expressas apenas por números, indicando quantidade de "algo" em valores absolutos (MARCONI; LAKATOS, 2012).

Essas avaliações, quantitativas e qualitativas, se complementam; na medida que são necessárias para responder e analisar perguntas que possuem naturezas distintas. As medidas quantitativas respondem a pergunta "quanto", enquanto as qualitativas respondem a pergunta "como" (MARCONI; LAKATOS, 2012). Vale abordar, que neste trabalho, a interpretação das medições também fazem parte da abordagem qualitativa, na medida em que a opinião do pesquisador está integrada na compreensão dos fatos.

A primeira, segunda e terceira avaliações realizadas nesse trabalho nas Subseções 4.2.1, 4.2.2 e 4.2.3 são qualitativas, e respondem, respectivamente, as seguintes perguntas: "Analisando as características gerais dos trabalhos relacionados e do FIXP, como as diferenças dos mecanismos operacionais do *FIXP* podem solucionar os problemas de comunicação identificados nos trabalhos que estão presentes no estado da arte?", "Como os princípios arquitetônicos da arquitetura *FIXP* podem ser relevantes para as redes atuais e futuras?" e "Como implementar e implantar de forma eficiente esses princípios arquitetônicos nas redes atuais?".

Para responder essas perguntas, este trabalho classifica uma série de informações que estão presentes nas tabelas 2, 5 e 6; que auxiliam na compreensão de que há efeitos colaterais (positivos e negativos) nas decisões de projeto/mecanismos utilizados por cada trabalho relacionado (ver Subseção 2.2.6); e, que agora, neste capítulo, auxilia na realização de uma análise qualitativa que responde as três perguntas anteriores. O fato é que tanto os trabalhos relacionados quanto as redes atuais possuem problemas significativos na comunicação, e esses problemas estão diretamente ligados aos mecanismos utilizados por cada solução.

Essas informações (mecanismos utilizados e efeitos colaterais) estão categorizadas, ordenadas e organizadas nas tabelas 2 e 5. Em cada uma das categorias que referem-se aos mecanismos utilizados; a tabela armazena apenas a ausência ou presença do mecanismo em cada *paper* pesquisado. As categorias que representam os mecanimos utilizados são: *Interoperability (INT)*, *Software Defined Networking (SDN)*, *Network Virtualization (VIR)*, *Network Overlay (OVE)*, *Tunneling (TUN)*, *Active Networks (ACT)*, *Implementation or semantic and Syntactic description of the protocol stack (PST)*, *P4 language (P4)*. Em cada uma das categorias que representam os efeitos colaterais; a tabela armazena a presença ou ausência do efeito colateral ou descreve sua característica para cada paper pesquisado. São elas: *Additional processing into the switches (PRO)*, *Additional deployment of protocol stacks into the switches (STA)* e *Level of difficulty in the deployment of new FIAs (LEV)* e *Internet Architectures integrated into the meta-network (IAs)*.

A Tabela 6, igualmente importante, registra os princípios arquitetônicos e os mecanis-

mos operacionais de cada FIA integrada em cada uma das meta-arquiteturas que possuem o mesmo objetivo arquitetônico do *FIXP*. Essas informações mostram a flexibilidade das meta-arquiteturas em absorver funcionalidades de serviços que pertencem a diferentes arquiteturas de Internet; e, são categorizadas, por arquitetura de Internet, da seguinte forma: *design orientation*, *communication paradigm*, *network addresses*, *network protocol*, *routing features*, *switching element* e *security*.

De um lado, a Tabela 6 demonstra a flexibilidade das meta-arquiteturas, cujos objetivos arquitetônicos são os mesmos do FIXP; de outro, as tabelas 2 e 5 informam que os mecanismos utilizados por cada uma dessas meta-arquiteturas são diferentes e que estão diretamente ligados aos efeitos colaterais da comunicação. Essas informações, em conjunto, fornecem a base que permite as três análises qualitativas realizadas no trabalho, e, por conseguinte, representam o embasamento de uma investigação criteriosa e interpretativa dos fatos observados; fatos esses que não podem ser mensurados quantitativamente. A relação existente entre mecanismos utilizados e os efeitos colaterais que esses mecanismos causam na comunicação - como também a quantidade de serviços evolucionários, *clean-slate* e tradicionais que cada meta-arquitetura suporta atualmente - representam a base lógica de argumentos que sustentam essa investigação.

A quarta avaliação realizada nesse trabalho, na Subseção 4.2.4, é quantitativa e tem três finalidades principais: demonstração conceitual da arquitetura FIXP, análise do sobrecarregamento da infraestrutura *FIXP* e análise da performance das aplicações que participam dessa rede. A primeira finalidade é obrigatória, pois faz parte do objetivo geral desta tese, as duas últimas são análises extras que o trabalho realiza para inferir o comportamento da infraestrutura FIXP no momento da execução dos casos de uso propostos.

A avaliação quantitativa demanda a implementação da arquitetura *FIXP*, o planejamento dos casos de uso, a categorização dos dados coletados/medições realizadas que são suficientes para cumprir os objetivos empíricos, a realização dos experimentos, a execução de técnicas estatísticas, a interpretação dos dados que demonstram ou refutam a hipótese e objetivos do trabalho.

A descrição de cada uma dessas demandas encontra-se na Subseção 4.2.4.

Em relação à implementação da arquitetura e de todos os componentes necessários para a sua materialização, o código está disponível em <https://github.com/julianoco/Future-Internet-eXchange-Point---FIXP>.

Em relação ao planejamento dos casos de uso e organização das informações, são elaborados 22 casos de uso que além de provar experimentalmente os objetivos arquitetônicos do FIXP, também testam a performance de aplicações de Chat, de *streaming* de vídeo sob demanda; e, de aplicações multiarquitetura, aquelas que possuem os dois serviços simultaneamente, sendo que um deles pertence à arquitetura ETArch e o outro à arquitetura IP; ou vice-versa. Cada caso de uso testa uma determinada aplicação de uma

FIA específica e possui suas próprias características tais como tipo de conexão, número de mensagens enviadas pela aplicação, número de mensagens processadas pelo switch, tamanho das mensagens e números de entidades que participam da comunicação. Essa variação é importante porque aumenta o número de contextos que são experimentados dentro da rede *FIXP*, possibilitando uma análise mais apurada dos acontecimentos.

Cada caso de uso é experimentado 10 vezes; ou seja, para cada caso de uso é feito 10 experimentos. Isso dá o montante de 220 experimentos e mais de 5 milhões de registros analisados.

Em relação à classificação/categorização das informações coletadas, ela é realizada através de métricas que avaliam a performance do *FIXP* e das aplicações. Essas métricas são coletadas de acordo com a aplicação que está sendo executada; pois cada aplicação possui seus próprios requisitos para uma boa experiência do usuário. Ao todo são 10 métricas escolhidas para medição da performance: *Response Time Expected by User (RTEU)*, *Delay*, *Jitter*, *Required BandWidth*, *Loss Rate*, *Error Rate*, *Memory*, *Processing*, *Data Sending Rate* e *Tranfer*.

O intervalo das métricas *RTEU*, *Delay*, *Jitter*, *Required BandWidth*, *Loss Rate*, *Error Rate* e *Data Sending Rate* possui 95% de confiança, com 09 graus de liberdade; calculada através da tabela *t-Student* (SPIEGEL; SCHILLER; SRINIVASAN, 2016). Mais detalhes estão na Subseção 4.2.4.4

Os resultados puramente quantitativos dão uma boa ideia do que acontece na comunicação de aplicações que trafegam naturezas distintas de tráfego: vídeo e pequenos textos de chat; porém os valores das grandezas coletadas relacionam-se entre si, produzindo efeitos na comunicação que só podem ser explicados através de uma interpretação qualitativa desses dados, onde a opinião do pesquisador está integrada intrinsecamente na análise que valida ou refuta a hipótese e os objetivos da tese; e que também, responde às questões de pesquisa. Essa análise é criteriosa e fundamentada, logicamente, em argumentos já conhecidos pela comunidade científica e pela observação dos fenômenos investigados.

4.2 Avaliações

Esta seção avalia o *FIXP* em quatro níveis. A primeira etapa é qualitativa e avalia as características gerais do FIXP, tendo como foco descrever as diferenças entre o FIXP e os trabalhos correlatos. O intuito é destacar os benefícios que os mecanismos operacionais utilizados pelo FIXP proporcionam na comunicação; posicionando-o, dessa forma, no estado da arte (Subseção 4.2.1). A segunda etapa é qualitativa e possui um caráter mais restrito. Ela avalia, especificamente, a relevância dos princípios arquitetônicos do FIXP em relação às necessidades de arquiteturas de Internet atuais e futuras (Subseção 4.2.2). A terceira etapa, também qualitativa, restringe-se à avaliação da eficiência de implementação e implantação desses princípios arquitetônicos do *FIXP* nas redes atuais

(Subseção 4.2.3). A quarta etapa é quantitativa: prova conceitualmente os objetivos arquitetônicos do *FIXP* e mede a eficiência da comunicação no plano de dados (Subseção 4.2.4).

Tabela 5 – Tabela comparativa que exibe os mecanismos utilizados por cada meta-arquitetura que suporta diversas FIAs

INT - Interoperability. **SDN** - Software Defined Networking. **VIR** - Network Virtualization.
OVE - Network Overlay. **TUN** - Tunneling. **ACT** - Active Networks.
PST - Implementation of the protocol stack. **P4** - P4 language
PRO - Additional processing into the switches. **STA** -Additional deployment of protocol stacks into the switches.
LEV - Level of difficulty in the deployment of new FIAs.
IAs - Internet Architectures integrated into the network.
x - It does not have the functionality. red - Represents negative side effects.

Related works.	Mechanisms used in each related work.								Side effects of the mechanisms used.			
	INT	SDN	VIR	OVE	TUN	ACT	PST	P4	PRO	STA	LEV	IAs
FIFu	v	v					v		Yes.	Yes.	High.	(i) PURSUIT. (ii) IP. (iii) NDN.
Linux XIA							v		Yes	Yes	High.	(i) Serval. (ii) zFilter. (iii) IP.
FIXP		v						v	No.	No.	Low.	(i) IP. (ii) ETArch. (iii) NovaGenesis.

4.2.1 Avaliação das características do FIXP em relação aos trabalhos relacionados

As tabelas 5 e 6 são importantes para posicionar o *FIXP* no estado da arte. A Tabela 5 apresenta uma correlação direta entre as escolhas de projeto (mecanismos utilizados) de alguns dos trabalhos relacionados (apresentados na Seção 2.2) e os efeitos colaterais (positivos e negativos) produzidos na comunicação dos seus respectivos planos de dados. A Tabela 6 mostra as diferentes características das *FIAs* que cada uma dessas meta-arquiteturas suporta. Tanto a Tabela 5 quanto a Tabela 6 exibem apenas os trabalhos relacionados que possuem o mesmo objetivo arquitetônico do *FIXP*, ou seja, aquelas meta-arquiteturas apresentadas na Seção 2.2 que suportam a comunicação de *FIAs* heterogêneas.

FIFu e Linux *XIA* são meta-arquiteturas que têm como mecanismos principais, respectivamente, a interoperabilidade e o desenvolvimento de pilhas de protocolos. Cada um desses mecanismos produz efeitos negativos na comunicação (ver Subseção 2.2.6). Os principais são: nível de dificuldade de implementação de novas *FIAs* é alto; ineficiência de implementação: em alguns casos, proporciona perdas de funcionalidade das *FIAs* implantadas; sobrecarregamento dos elementos de rede em termos de processamento, me-

Tabela 6 – Tabela comparativa das características das FIAs implantadas em cada meta-arquitetura.

Meta network architecture	Integrated architectures	Features
FIFu	IP	**Design orientation:** Host-oriented. **Communication Paradigm:** Push. **Network Adresses:** Hierarquical identifier; Exclusive; Fixed length. **Network Protocols:** IP. **Routing features:** Supports host-based unicast communication. **Switching element:** IP-Dedicated Switch. **Security:** The identifier is self-certified.
	NDN	**Design orientation:** ICN. **Communication Paradigm:** Pull(Request/Response). **Network Adresses:** Hierarchical (similar to URL); Exclusive, unique and immutable; Variable length. Adressing is an association of Identifiers and Locators. **Network Protocols:** NDN's protocols (interest and data primitives). **Routing features:** Symmetric, stateful and multicast routing. **Switching element:** . NDN-Dedicated Switch. **Security:** Data is self-authenticating (it contains the producer's signature).
	PURSUIT	**Design orientation:** ICN. **Communication Paradigm:** Push(Publish/Subscribe). **Network Adresses:** Flat label; Exclusive; Variable length. **Network Protocols:** PURSUITś protocol stack that allows publication and subscription of resources. **Routing features:** Supports multicast, anycast, and multi-homing communication. **Switching element:** Click Router framework (KOHLER et al., 2000). **Security:** packet level authentication.
Linux XIA	IP	**Design orientation:** Host-oriented. **Communication Paradigm:** Push. **Network Adresses:** Hierarquical identifier; Exclusive; Fixed length. **Network Protocols:** IP. **Routing features:** Supports host-based unicast communication. **Switching element:** IP-Dedicated Switch. **Security:** The identifier is self-certified.
	Serval	**Design orientation:** SCN. **Communication Paradigm:** Pull(Request/Response) (optional); Push (optional); Controllers act proactively (optional) and reactively (optional). **Network Adresses:** Flat (optional). Hierarchical (optional). Variable length(optional). Fixed length (optional). It has three identifiers: service (unique per instance), flow (unique per host), locators (depends on the implementation). Use of prefixes (optional). **Network Protocols:** Serval's protocols. **Routing features:** Service Name-Based Routing (ServiceIds). **Switching element:** . Serval-dedicated service switches. **Security:** Service identifiers are self-certifying (optional). Flow identifiers are random and supplemented by nonces (optional).
	zFilter	**Design orientation:** Data-oriented. Supports data caching in the network. Recursive architecture (recursive headers). **Communication Paradigm:** Push. zFilter extends the publish/subscribe protocol. **Network Adresses:** Variable length. Unicast/multicast trees of physical or virtual links registered in probabilistic structures called Bloom filters. **Network Protocols:** zFilter protocols extend the functionality of the Publish/Subscribe paradigm. The zFilter primitive has recursive headers. **Routing features:** Probabilistic source routing based on bloom filters (default/optional). Stateful multicasting routing based on virtual links (optional). **Switching element:** zFilter-dedicated service switches. **Security:** Identificadores são ocultos dentro de filtros Bloom.
FIXP	ETArch	**Design orientation:** SDN-OpenFlow; Worspace-oriented communication. **Communication Paradigm:** Proactive (ETArch Controller Forwarding Rules Policy); Push/Pull. **Network Adresses:** Flat; Fixed lenght; Separation of Identifiers/Locators. **Network Protocols:** ETArchś protocols (ETCP e DTSCP). **Routing features:** Workspace-based multicast routing. **Switching element:** . switch Openflow. **Security:** Identifiers are self-certifying.
	IP	**Design orientation:** Host-oriented. **Communication Paradigm:** Push. **Network Adresses:** Hierarquical identifier; Exclusive; Fixed length. **Network Protocols:** IP. **Routing features:** Supports host-based unicast communication. **Switching element:** IP-Dedicated Switch. **Security:** The identifier is self-certified.
	NovaGenesis	**Design orientation:** SCN and Service-Oriented Architecture (SOA), Principles of ICN and Active Networks. **Communication Paradigm:** Push(Publish/Subscribe). **Network Adresses:** Flat (optional); Hierarchical (Optional); Variable length; Name Buildings (NB); Network addressing can have multiple interpretations; Natural Language Naming (Optional); Self-Verifying Name (SVN) via hash. **Network Protocols:** NovaGenesis Protocol. **Routing features:** NB-based routing; Stateful routing (in fragmentation scenarios). **Switching element:** It is a software (service) called PGCS. **Security:** Self-Verifying Name via hash.
	XIA(Prototype)	**Design orientation:** Principal-oriented (meta architecture (MACHADO; DOUCETTE; BYERS, 2015)). **Communication Paradigm:** In this paper, Pull(Request/Response). *Each principal has its communication paradigm. **Network Adresses:** Flat (optional); Hierarchical (optional); Variable length. **Network Protocols:** XIA protocol: eXpressive Internet Protocol (XIP) **Routing features:** Principals-based routing. **Switching element:** XIA-dedicated Switch. **Security:** Intrinsic, the identifier is self-certified.

mória e armazenamento; a segmentação de rede em regiões do processo de interoperação inviabiliza a implantação dessas meta-arquiteturas em escala global.

O *FIXP* é uma meta-arquitetura, baseada em *SDN-P4*, que possui uma visão de rede mais simples. Na visão do *FIXP*, a rede é única e pode ser compartilhada por *FIAs* heterogêneas. Isso não quer dizer que o *FIXP* não seja capaz de intercomunicar *FIAs* no cenário mais complexo, onde as entidades comunicantes pertencem a domínios administrativos que possuem diferentes substratos físicos de comunicação. A Figura 17 apresenta um desses cenários: os domínios *ETArch* 2 e *FIXP* 1 comunicam-se entre si e possuem substratos físicos distintos: *switches ETArch* e *FIXP*, respectivamente. O *FIXP* consegue fazer a comunicação entre esses domínios independentemente da infraestrutura física de cada um deles. No entanto, *FIXP* vislumbra uma rede única, onde todas as *FIAs* adequem-se à sua arquitetura de maneira gradual. Dessa forma, o *FIXP* não é apenas um ponto de troca lógico entre *FIAs*, mas um framework que padroniza a comunicação e os substratos físicos de rede, facilitando a integração de arquiteturas de Internet em escala mundial. Isso possibilita que as operadoras integrem várias *FIAs* com um único núcleo padrão de rede, formando topologias físicas de *switches FIXP-P4*.

Outro ponto importante é a separação dos planos de dados e controle. Essa separação reduz a sobrecarga dos *switches*, que possuem apenas a função de reconhecimento de protocolos e repasse; não possuem as funcionalidades de controle, nem as pilhas de protocolos de cada *FIA*, nem mecanismos de interoperabilidade que podem onerar o funcionamento desses *switches* em termos de armazenamento, memória e processamento.

No caso da implementação e implantação de novas *FIAs*, o FIXP não possui os mesmos problemas que estão presentes em meta-arquiteturas que utilizam mecanismos de interoperação e/ou desenvolvimento de pilhas de protocolos, que conforme descritos na Seção 2.2.6, são basicamente dois: nível de dificuldade de implementação de novas *FIAs* é alto e ineficiente; em mecanismos de interoperação, a implementação da meta-arquitetura em ambientes de produção é inviável em termos de gerenciamento e custo, porque os agentes de telecomunicações, necessariamente, teriam que replicar as infraestruturas físicas de cada FIA que participa da comunicação. A subseção 4.2.3 avalia o nível de dificuldade de implementar novas *FIAs* na infraestrutura FIXP, classificando-o como baixo em relação aos trabalhos relacionados. Além disso, demonstra a eficiência da implantação de serviços de rede das FIAs no FIXP, sem perda de funcionalidades. Isso resolve o primeiro problema. Essa mesma subseção (4.2.3), avalia como o FIXP pode implementar e implantar diferentes arquiteturas de Internet sem precisar replicar suas infraestruturas físicas. Isso resolve o segundo problema.

Outro ponto a ser mencionado nessa discussão é que todas as meta-arquiteturas discutidas na Seção 2.2, inclusive *FIXP*, possuem problemas de hardware. É necessário que os *switches* forneçam capacidades que se adaptem aos requisitos das *FIAs*. Um exemplo é que o *switch P4* padrão não possui escalonador de primitivas prioritárias, o que limita

o *FIXP* em comunicações que necessitam de qualidade de serviço. Essa funcionalidade pode ser estendida, mas esse procedimento causa dependência da arquitetura do *switch P4* onde a extensão for implementada. Como dito, a evolução da linguagem padrão *P4* proporciona, inexoravelmente, a evolução das funcionalidades do protocolo *FIXP*. Outra forma de mitigar essa limitação é o *protocolo FIXP* reconhecer o *switch FIXP* da infraestrutura e oferecer recursos de acordo com os *switches* instalados. Trabalhos futuros vão decidir a direção do *FIXP* na resolução desse problema.

A Tabela 6 mostra as características detalhadas das arquiteturas integradas no *FIFu*, *FIXP* e Linux *XIA*.

FIFu (GUIMARãES et al., 2019), em um de seus casos de uso, faz comunicação entre três *FIAs* distintas da seguinte forma: navegadores WEB *NDN* e PURSUIT (clientes) requisitam recursos de um servidor WEB IP. Para isso, segmenta a rede em regiões e realiza a comunicação via gateway, que faz o processo de interoperação.

Na visão do *FIXP*, esse tipo de comunicação não é a melhor opção, pois não é do interesse de uma empresa de software fornecer um aplicativo onde servidores e clientes possuem *FIAs* distintas. Os motivos são dois: complexidade no desenvolvimento porque utilizam frameworks de comunicação distintas; perda de qualidade na comunicação. O *FIXP* oferece duas abordagens. A primeira, julga-se a mais corriqueira, funciona da seguinte forma: aplicações ETArch comunicam-se com aplicações ETArch, aplicações NovaGenesis comunicam-se com aplicações NovaGeneis, e assim por diante. Essa forma de comunicação, na visão do FIXP, é a mais adequada, pois suaviza a complexidade em relação à engenharia de software e ao mesmo tempo mantém a qualidade da comunicação, já que o controle do tráfego de mensagens é feito pela FIA correspondente, sem adicionamento de processamentos adicionais no plano de dados. A segunda abordagem refere-se a visão que o FIXP possui de uma aplicação multiarquitetura: a aplicação utiliza um serviço X que realiza comunicação via *ETArch*, e ao mesmo tempo, possui outro serviço Y, que realiza comunicação via *NovaGenesis*. A aplicação é multiarquetura porque possui serviços distintos que utilizam *FIAs* distintas. Essa aplicação, similarmente à FIFu, continua utilizando mais de um *framework* de Internet do Futuro, mas em contrapartida, mantém a qualidade do serviço, pois está sendo gerenciado pela FIA correspondente, desde o envio da mensagem até a recepção final. Na visão do FIXP, os casos de uso deste trabalho que refletem essa comunicação multiarquitetura são importantes, pois simulam a execução de diversos microsserviços que atuam na mesma aplicação e/ou máquina virtual, um fato bem frequente nas redes atuais. A diferença é que esses microsserviços implementam *frameworks* distintos de arquiteturas de Internet.

Por outro lado, Linux *XIA* tende a enxergar a rede igual ao *FIXP*, com uma diferença. Em seus experimentos, Linux *XIA* implementa duas pilhas de protocolos novas no *kernel* do Linux. Uma das pilhas é referente à arquitetura *Serval*, a outra é referente à arquitetura *zFilter*. A Tabela 6 apresenta as características dessas pilhas. O processo de

implementação se dá quando os autores traduzem as funcionalidades de *Serval* e *zFilter* para princípios de uma arquitetura já existente: *Linux* XIA. A rede final desse processo é única como a do *FIXP*, porém esse mecanismo tem problemas já discutidos nessa seção e em outras (ver Subseção 2.2.6), como por exemplo, possível perda de funcionalidades das *FIAs* implantadas.

4.2.2 Avaliação da arquitetura FIXP

O FIXP propõe uma abordagem em que uma rede única é capaz de absorver as funcionalidades de todas as FIAs. Sendo assim, essa rede possibilita a inclusão de novos serviços ou protocolos de controle, facilitando a evolução das capacidades de comunicação frente a novos desafios. Esse é o objetivo arquitetônico do FIXP e tem influência direta em suas escolhas de projeto.

O primeiro conceito arquitetônico do *FIXP* (primeira escolha do projeto) é a orientação de design baseada em *SDN*. O paradigma *SDN* separa os planos de dados e controle. Isso é importante para o projeto, porque para cada *FIA* nova que participa da rede, existe um controlador *SDN* que possui todas as suas funcionalidades de controle. Dessa forma, a rede possibilita que cada *FIA* evolua conforme as necessidades das suas aplicações. *FIXP* não possui limitações que dificultam a inclusão e evolução de protocolos porque possui um plano de controle extensível, configurável e programável. O controlador é um componente funcional, que juntamente com o *FIXP*, gerencia o comportamento do fluxo do plano de dados. Além disso, o plano de controle programável possibilita que as serviços de rede das FIAs sejam implantadas eficientemente, sem perda de funcionalidades.

O segundo conceito arquitetônico do *FIXP* é o fornecimento de um plano de dados programável, e, por conseguinte, a segunda escolha do projeto é a utilização da linguagem P4 para implementar o paradigma *SDN*. O programa P4 é um componente essencial da arquitetura, pois possibilita as configurações dos blocos funcionais programáveis que gerenciam o comportamento do switch. O plano de dados é programável e extensível; permite que as *FIAs* configurem, através do *FIXP*, o pipeline de dados de todos os seus protocolos. Além disso, a programabilidade do plano de dados é um recurso que flexibiliza o reconhecimento de novos protocolos e a evolução dos formatos das primitivas já existentes. A flexibilidade dos planos de dados e controle permite que o *FIXP* integre em sua rede *FIAs* heterogênas, com diferentes características.

A Tabela 6 apresenta as características das 03 *FIAs* distintas que participam da rede *FIXP*. Cada uma dessas *FIAs* possui diferentes orientações de design, paradigmas de comunicação, esquemas de endereçamento, protocolos de rede, algoritmos de roteamento. dispositivos de comutação, entre outros.

Os *papers*, já publicados, que apresentam alguns aspectos da integração dessas *FIAs* no *FIXP*, realizam experimentos diferentes na rede. Cada *FIA* tem seu próprio domínio e seus próprios aplicativos. Cada um desses aplicativos comunicam-se entre si através de

um ponto de troca *FIXP*. ETArch utiliza um aplicativo próprio para realizar uma comunicação *multicast* entre três entidades através de um canal lógico denominado Workspace (GAVAZZA et al., 2020) . O protocolo ETCP gerencia essa comunicação. IP utiliza um aplicativo próprio para realizar uma comunicação *end-to-end* entre duas entidades através dos protocolos UDP/IP (GAVAZZA et al., 2020) . As entregas das mensagens são realizadas por um algoritmo que simula o protocolo *OSPF* (MOY, 1998). NovaGenesis (NG) executa o cenário de um aplicativo de rede de distribuição de conteúdo através do paradigma *Publish/Subscribe* (SILVA et al., 2022) . Esse aplicativo requisita recursos através de nomes de conteúdo e armazena esses conteúdos temporariamente no núcleo da rede, utilizando princípios de ICN.

Esses casos de uso, já publicados, são suficientes para demonstrar que os princípios arquitetônicos do *FIXP* são relevantes para as redes atuais; além disso, cumpre os objetivos do projeto: coexistir arquiteturas de Internet heterogêneas de forma transparente e possibilitar a evolução, a inovação e a inclusão de novas *FIAs* ou serviços de rede sem comprometer a complexidade da infraestrutura e a eficiência da comunicação. Percebe-se também que os princípios arquitetônicos do *FIXP* são relevantes em relação aos trabalhos relacionados, pois, no cumprimento de seus objetivos arquitetônicos, não adiciona processamentos adicionais que possam, eventualmente, sobrecarregar os dispositivos de comutação em termos de armazenamento, processamento e memória, como é o caso das soluções relacionadas (ver subseção 2.2.6).

4.2.3 Avaliação da implementação e implantação do FIXP nas redes atuais

A viabilidade do *FIXP* como proposta de uma rede única ou um ponto de troca lógico capaz de promover um ambiente de comunicação onde coexistem diversas *FIAs* passa pela estratégia de implementação e implantação dos seus recursos arquitetônicos nas redes atuais. O *FIXP* deve ter capacidade de evolução (avaliada na Subseção 4.2.2); o nível de dificuldade de inclusão de novas *FIAs* na rede *FIXP* não deve ser alto; e, deve haver um planejamento de implantação onde não seja necessário replicar os recursos físicos de cada *FIA* que participa da comunicação, viabilizando, dessa forma, o projeto, do ponto de vista financeiro e de gerenciamento de rede.

Essa seção avalia a eficiência da implementação dos princípios arquitetônicos e a implantação desses princípios nas redes atuais. Afinal de contas, se operadores de rede não conseguem instanciar uma meta-arquitetura, ela não tem utilidade prática.

A Seção 3.6 descreve a estratégia de implementação e implantação do *FIXP* em escala global. Operadoras de telecomunicações (*ISPs*, *IXPs*, *PoPs*, etc.) podem possuir um núcleo de rede *FIXP*, onde a infraestrutura física (plano de dados) possui uma topologia com switches *FIXP* e cada *FIA* integrada possui um controlador com seus serviços de

gerenciamento de rede. Esses serviços gerenciam o plano de dados através da interface *Southbound* do *FIXP*. Se a primitiva de determinada *FIA* chega em um dispositivo de borda através de um link dedicado ao *FIXP* (ver IXP na Figura 17), essa primitiva de ingresso alcança o núcleo *FIXP*, que encaminha as primitivas para seu destino. Se a primitiva chega na borda através de um link compartilhado com os núcleos IP e *FIXP* (ver ISP Regional 01 na Figura 17), o *middlebox* vai analisar o pacote e enviá-lo para o núcleo correspondente. Se a primitiva de ingresso alcança o núcleo *FIXP*, ele envia-a para o destino correspondente.

Essas operadoras (*ISPs*, *IXPs*, *PoPs*, etc.), que possuem essa configuração (ver Figura 17), permitem a coexistência de *FIAs* heterogêneas. A conexão entre os núcleos de rede *FIXP* com outros núcleos da rede (*FIXP* ou outros) promove uma rede em escala global. Na figura, o plano de dados do IXP comunica-se com o plano de dados IP do ISP Regional 02, porque a rede *FIXP* do IXP tem um controlador IP, e por conta disso, absorve as funcionalidades dessa arquitetura. Cada um dos domínios de *FIAs* que participam da comunicação pode possuir sua própria rede. Como exemplo, os domínios ETArch da figura possuem infraestrutura de rede com switches SDN-OpenFlow-ETArch e controladores ETArch. Esses domínios também já poderiam estar adaptados ao *FIXP*. Como exemplo, o domínio *FIXP* 1 da figura possui switches *FIXP* e controladores de diversas *FIAs*. O *FIXP* comunica aplicativos ETArch com aplicativos ETArch, aplicativos NovaGenesis com aplicativos NovaGenesis, aplicativos Multiarquitetura com aplicativos Multiarquitetura, e assim por diante; independentemente da infraestrutura física de seus domínios. Isso possibilita que cada uma das *FIAs* adeque-se ao *FIXP*, gradualmente. Enquanto isso, *FIXP* consegue suportar as comunicações normalmente.

Do ponto de vista holístico, o FIXP, como ponto de troca lógico que interconecta domínios distintos que possuem diferentes infraestruturas físicas, não é a solução ideal desse ambiente de comunicação, pois os agentes de telecomunicações teriam que replicar a infraestrutura que suporta as FIAs desses domínios, causando complexidade de gerenciamento e altos custos com operação e equipamento. Porém, o FIXP possibilita a solução ideal, que é a integração total do FIXP nas redes atuais, onde cada agente de telecomunicações tem um núcleo que possui infraestruturas físicas padronizadas compostas por switches FIXP e um conjunto de controladores que suportam as FIAs desejadas. Essa solução traz várias vantagens: otimização de gerenciamento e custos, padronização da infraestrutura física independentemente dos requisitos da aplicação, padronização da comunicação entre entidades de controle e os elementos de rede independentemente dos fabricantes, possibilidade de escolha da arquitetura de Internet que melhor atende aos requisitos das aplicações.

Outro ponto a ser considerado é que a integração de uma nova *FIA* no *FIXP* tem um nível de dificuldade baixo em relação aos trabalhos correlatos. A Tabela 2 mostra a tendência das *FIAs* em implantar suas arquiteturas em P4. Nesses casos, em que as

FIAs são SDN-P4; o nível de dificuldade de implantá-las no FIXP é baixo, pois elas já possuem ou estão desenvolvendo um controlador P4. Sendo assim, apenas uma ação é necessária: adaptar o controlador à interface *Southbound* do *FIXP*. Em outros casos, em que a FIA é SDN, mas não utiliza a linguagem P4, o nível de dificuldade também é baixo. O processo de implantação é análogo ao anterior, porém uma ação adicional é necessária: especificar o plano de dados da FIA correspondente através de um programa P4, que reconhece seus protocolos, primitivas, serviços e mecanismos operacionais. No caso em que a *FIA* não possui um controlador P4, nem tem o SDN como paradigma de comunicação; os desenvolvedores devem implementar um controlador com as funcionalidades de controle da *FIA* que está sendo integrada. As funcionalidades já existem, basta transferi-las para um controlador SDN. Análogo à primeira situação, o segundo passo é integrar a interface *Southbound* do *FIXP* a esse controlador. Além disso, na última etapa dessa implementação, o autor tem que apresentar um programa P4 que configura o plano de dados do *FIXP* e reconhece os novos serviços, protocolos, primitivas e mecanismos operacionais da nova *FIA* integrada. Como o último caso tem um passo a mais, considera-se o nível de dificuldade de implantação como sendo médio em relação ao caso anterior, mas ainda baixo em relação aos trabalhos relacionados. É importante salientar que os níveis de dificuldade de implantação tomam como referência os trabalhos relacionados ao FIXP e que possuem o mesmo objetivo arquitetônico. O autor das implantações de uma FIA no FIXP não tem que interpretar, nem conhecer a fundo várias *FIAs* para realizar a implantação da sua arquitetura, como é o caso de implantações de *FIAs* em meta-arquiteturas que possuem mecanismos de interoperação e/ou desenvolvimento de pilhas de protocolos (ver Subseção 2.2.6). Por isso, em relação a esses trabalhos, o nível de dificuldade de implantação de novas FIAs no FIXP é baixo.

ETArch (GUIMARãES et al., 2014; SILVA et al., 2012) é um exemplo de arquitetura com nível de dificuldade de integração baixo, pois seu paradigma de comunicação original é SDN-OpenFlow. Os dois únicos passos da integração (GAVAZZA et al., 2020) foram adaptar o controlador ETArch à interface *Southbound* do protocolo *FIXP* e descrever a especificação do plano de dados que a arquitetura necessita para realizar a sua comunicação. NovaGenesis (ALBERTI et al., 2019) é um exemplo de arquitetura com nível médio de integração. É uma arquitetura baseada em serviços, cujo paradigma de comunicação é *Publish-Subscribe*. Para integrá-la ao *FIXP* (SILVA et al., 2022), os autores transferiram as funcionalidades do componente *PGCS* para um controlador, e logo após, adaptou-o à interface *Southbound* do protocolo FIXP. Logo após, os autores apresentaram um programa P4 capaz de reconhecer os novos protocolos, primitivas, serviços e mecanismos da NovaGenesis na rede *FIXP*. Nos dois casos, não foi necessário interpretar uma arquitetura em outra, nem configurar descrições de contexto, nem implementar novamente o código de uma FIA em outra linguagem de programação, nem instalar pilhas de protocolos nos gateways e/ou dispositivos de comutação; como acontece nas meta-arquiteturas

que possuem mecanismos de interoperação e desenvolvimento de pilhas de protocolos (ver Subseção 2.2.6 e Seção 4.2.1).

Outra dificuldade encontrada nos trabalhos relacionados é a ineficiência dos mecanismos de interoperação e de desenvolvimento manual de protocolos na implementação das FIAs quando as arquiteturas envolvidas possuem diferentes capacidades de serviços. Ambos os processos podem causar perda de funcionalidades na implantação dessas *FIAs*, e, por conseguinte, perda de qualidade da comunicação. O *FIXP* não tem essa limitação, pois cada controlador *SDN-P4* de cada *FIA* pode implementar todos os seus serviços de rede.

Aliás, os controladores das *FIAs* são componentes essenciais para a rede *FIXP*. O *FIXP* imagina, no futuro, um repositório de controladores *online* que possui a implementação de diversas *FIAs*. Quando um operador de rede tomar a decisão de implantar determinada *FIA* em suas dependências, basta entrar no repositório e fazer download do controlador. Outra alternativa seria oferecer a rede FIXP como serviço de nuvem; de forma análoga aos trabalhos (NASCIMENTO et al., 2011; STRINGER et al., 2014), que oferecem rede IP como serviço (*IP network as a Service (IP-NaaS)*).

As discussões teóricas e os exemplos de integração de *FIAs* no *FIXP* demonstram que o nível de dificuldade de implantação de novas *FIAs* no *FIXP* é baixa em relação aos trabalhos correlatos. Por outro lado, a implantação do *FIXP* em operadoras ou *IXPs* exige apenas switches *FIXP* e controladores das *FIAs* que a meta-arquitetura deve suportar. Como o *FIXP* suporta coexistência de *FIAs* heterogêneas independentemente da infraestrutura física dos seus domínios, ele possibilita a implantação gradual dessas *FIAs* na rede *FIXP*. O intuito é que os pontos lógicos FIXP locais, presentes nos agentes de telecomunicações, conectem-se entre si, gradualmente, até alcançar escala global. Isso possibilita a padronização de todas as redes de comunicação do planeta em uma só, com planos de dados e controle programáveis. A implantação de uma rede *FIXP* em escala global é viável em relação aos benefícios que esta rede proporciona: um deles é a coexistência entre *FIAs* distintas, onde cada serviço de rede pode utilizar uma *FIA* diferente, aquela que melhor atenda seus requisitos, aprimorando, dessa forma, a qualidade da comunicação e a experiência do usuário dos aplicativos atuais e da próxima geração.

4.2.4 Avaliação de performance

Esta subseção apresenta a avaliação quantitativa. Basicamente, ela tem como objetivo principal demonstrar conceitualmente a arquitetura FIXP, pois essa demonstração faz parte do objetivo geral do trabalho. Mas, além dos cumprimentos obrigatórios do trabalho, essa avaliação também foca em resultados de performance tanto em termos de sobrecarregamento da infraestrutura *FIXP* no momento da execução dos casos de uso quanto em termos da comunicação realizada pelas aplicações que participam dessa rede. O intuito é averiguar o comportamento dos *switches FIXP* sob estresse. Teoricamente,

esse comportamento seria o mesmo se os *switches* dos experimentos fossem equipamentos físicos, já que esses *switches* teriam os mesmos procedimentos operacionais implementados no *switch BMv2* (software) utilizado na execução dos casos de uso propostos nesta seção. O que mudaria de um switch para outro são as taxas de linha, ou seja; a inundação de tráfego necessária para provocar sobrecarga e estresse na infraestrutura *FIXP* seria muito maior. Isso deve-se ao fato de que o processamento de um *switch* de software, como o BMv2, é muito menor do que o de dispositivos físicos de comutação baseados na arquitetura *PISA* (KAUR; KUMAR; AGGARWAL, 2021).

Essa análise quantitativa demanda a implementação da arquitetura *FIXP* descrita no Capítulo 3; o planejamento dos experimentos através da descrição de casos de uso selecionados (Subseção 4.2.4.1); a coleta de dados brutos referentes à comunicação e à infraestrutura FIXP que garanta a geração das métricas propostas nesse trabalho (Subseção 4.2.4.2); a categorização dos dados coletados em métricas previamente selecionadas (Subseção 4.2.4.3); a aplicação de técnicas estatísticas que geram intervalos de confiança de médias aritméticas de grandezas escalares de performance de rede (Subseção 4.2.4.4); a elaboração de um cenário de avaliação que suporta os experimentos (Subseção 4.2.4.5); a utilização de ferramentas ou habilitadores tecnológicos que materializam o cenário de avaliação, os principais procedimentos responsáveis pela coleta e processamento das métricas (*key performance indicator (KPIs)*), como também pela execução dos casos de uso selecionados (Subseção 4.2.4.6); a medição dos experimentos para cada caso de uso selecionado, sendo que essas medições são colhidas de forma separada para aplicativos de Chat (Subseção 4.2.4.7), aplicativos de *streaming* de vídeo (Subseção 4.2.4.8) e aplicativos multiarquitetura (Subseção 4.2.4.9).

Essa separação da medição por aplicativo é facilmente justificada, já que cada aplicação de rede possui seus próprios requisitos para um boa experiência do usuário, e, dessa forma, as métricas utilizadas para cada aplicação são diferentes. Outra coisa importante é que, dessa forma, o *FIXP* pode ser avaliado em relação a dois tráfegos de naturezas distintas: a aplicação de Chat, baseada em Internet Relay Chat (OIKARINEN, 1993), trafega textos pequenos que visam a conversação entre pessoas; a aplicação de *streaming* de Vídeo *On Demand* trafega pedaços de vídeo de forma contínua; a aplicação multiarquitetura trafega esses dois tráfegos simultaneamente, sendo que em uns casos de uso, o Chat é um serviço da ETArch e o *streaming* de vídeo é um serviço do IP; em outros, acontece justamente o contrário. Os aplicativos multiarquitetura simulam a execução de microsserviços em uma mesma aplicação e/ou máquina virtual, fato muito presente nos dias atuais.

A base de referência dos experimentos são as métricas de qualidade de serviço (*QoS*) que cada aplicativo possui para que a comunicação proporcione uma boa qualidade de experiência do usuário (*QoE*). O *paper* (CHEN; FARLEY; YE, 2004) apresenta essas referências para as duas aplicações que fazem parte dos nossos experimentos, quais sejam: *Interactive Video On Demand* e *Chat* baseado em *Internet Relay Chat* (OIKARINEN,

Tabela 7 – Lista de casos de uso proposta nesta tese.

NMSA - Number of messages sent by the application. **NMPS** - Number of messages processed by the switch. **NE** - Number of entities participating in the communication.

Name	App./Archit.	Communication	NMSA	NMPS	Message size	NE
Case 01	Chat/IP	Synchronous/bidirectional	2000 messages	2000 messages	542 bytes	02
Case 02	Chat/IP	Synchronous/bidirectional	16000 messages	16000 messages	1000 bytes	02
Case 03	Chat/IP	Synchronous/bidirectional	30000 messages	30000 messages	1500 bytes	02
Case 04	Chat/IP	Asynchronous/bidirectional	2000 messages	2000 messages	542 bytes	02
Case 05	Chat/IP	Asynchronous/bidirectional	16000 messages	16000 messages	1000 bytes	02
Case 06	Chat/IP	Asynchronous/bidirectional	30000 messages	30000 messages	1500 bytes	02
Case 07	Chat/ETArch	Synchronous/bidirectional	2000 messages	4000 messages	542 bytes	02
Case 08	Chat/ETArch	Synchronous/bidirectional	16000 messages	32000 messages	1000 bytes	02
Case 09	Chat/ETArch	Synchronous/bidirectional	30000 messages	60000 messages	1500 bytes	02
Case 10	Chat/ETArch	Asynchronous/bidirectional	2000 messages	4000 messages	542 bytes	02
Case 11	Chat/ETArch	Asynchronous/bidirectional	16000 messages	32000 messages	1000 bytes	02
Case 12	Chat/ETArch	Asynchronous/bidirectional	30000 messages	60000 messages	1500 bytes	02
Case 13	Chat/ETArch	Asynchronous/bidirectional	5000 messages	25000 messages	1500 bytes	05
Case 14	Video/IP	Asynchronous/Unidirectional	∼ 21784 messages	∼ 21784 messages	542 bytes	02
Case 15	Video/IP	Asynchronous/Unidirectional	∼ 15959 messages	∼ 15959 messages	1000 bytes	02
Case 16	Video/IP	Asynchronous/Unidirectional	∼ 11.268 messages	∼ 11.268 messages	1500 bytes	02
Case 17	Video/ETArch	Asynchronous/Unidirectional	∼ 19520 messages	∼ 39040 messages	542 bytes	02
Case 18	Video/ETArch	Asynchronous/Unidirectional	∼ 15791 messages	∼ 31582 messages	1000 bytes	02
Case 19	Video/ETArch	Asynchronous/Unidirectional	∼ 11226 messages	∼ 22452 messages	1500 bytes	02
Case 20	Video/ETArch	Asynchronous/Unidirectional	∼ 10038 messages	∼ 50190 messages	1500 bytes	05
Case 21	Chat/ETArch	Asynchronous/bidirectional	∼ 1000 messages	∼ 2000 messages	542 bytes	02
	Video/IP	Asynchronous/Unidirectional	∼ 11017 messages	∼ 11017 messages	1500 bytes	02
Case 22	Chat/IP	Asynchronous/bidirectional	∼ 1000 messages	∼ 1000 messages	542 bytes	02
	Video/ETArch	Asynchronous/Unidirectional	∼ 10292 messages	∼ 20584 messages	1500 bytes	02

1993).

4.2.4.1 Casos de uso

Levando em conta a previsão de que 66% da população mundial usa a Internet e que grande parte do fluxo são provenientes de "vídeo na Internet" e "Web, email e dados" (CISCO, 2020); os casos de uso selecionados incidem na utilização de dois aplicativos: um que gera *streaming* de vídeo; outro que possibilita canais de chat entre os participantes da comunicação. A primeira aplicação envia pedaços (*chunks*) de vídeos de forma contínua; o segundo tem a característica de enviar pequenos textos, conforme a necessidade dos participantes.

A Tabela 7 apresenta todos os casos de uso propostos nesse *paper*. Cada caso de uso possui características distintas: aplicativo e arquitetura de Internet que são testados; tipo de comunicação das aplicações; número de mensagens que a aplicação envia para a rede

FIXP; número de mensagens que a rede *FIXP* processa; tamanho da mensagem enviada; e, número de entidades que participam da comunicação.

O Caso de uso 06, por exemplo, testa um aplicativo de Chat, baseado no protocolo Internet Relay Chat (OIKARINEN, 1993), que utiliza a arquitetura de Internet IP. A comunicação é assíncrona/bidirecional. Isso indica que as Entidades 1 e 2 enviam um montante de 30000 mensagens simultaneamente para o canal de chat: 15000 cada uma. Essas mensagens trafegam pela infraestrutura física do *FIXP*. Dessa forma, a rede *FIXP* processa essas 30000 mensagens. O tamanho de cada primitiva é de 1500 bytes e apenas 02 entidades participam da comunicação.

Dois adendos são importantes: nem sempre a quantidade de mensagens que a aplicação envia é igual a quantidade de mensagens que a rede *FIXP* processa. No caso 20, a aplicação envia 10038 mensagens, no entanto, a rede *FIXP* processa o quíntuplo dessas mensagens, cujo montante é 50190. Isso acontece porque a ETArch é *multicast*, e nesse caso, replica as mensagens no núcleo da rede para 05 entidades que participam do grupo de comunicação.

4.2.4.2 Sobre a coleta de dados

O propósito da coleta é registrar uma massiva quantidade de dados brutos com o intuito de categorizá-la, posteriormente, em classes ou categorias de informações que representam a performance das aplicações e o sobrecarregamento do plano de dados no momento de execução desses aplicativos.

Levando em conta os objetivos da coleta, decide-se por dois grupos de dados. O primeiro grupo refere-se a dados de performance das aplicações. Para isso, é necessária a coleta de dados que têm relação com a comunicação, ou seja, com o envio e recebimento de mensagens. Essa coleta ocorre no momento em que as aplicações estão sendo executadas nos sistemas finais. O segundo grupo refere-se a dados que medem a utilização do processador e da memória *RAM* dos dispositivos de comutação que participam da rede. Essa coleta ocorre no momento em que a infraestrutura física do FIXP recebe o tráfego dos casos de uso propostos.

Em relação à coleta do primeiro grupo; cada instância da aplicação possui algoritmos que coletam dados da comunicação, quais sejam: identificador exclusivo da mensagem; data/hora de envio ou recebimento da mensagem em três formatos diferentes (*Datetime*, *String* e *Timestamp*), com precisão de 06 casas decimais (*microssegundos (ms)*); tamanho da primitiva; *FIA* da primitiva; identificadores do caso de uso e do experimento que estão sendo realizados; identificador da entidade que está enviando ou recebendo a primitiva.

Em relação à coleta do segundo grupo; cada instância de switch *FIXP* contém um *daemon* que mede a utilização do processador e da memória RAM do dispositivo através de interfaces do próprio sistema operacional. Os dados coletados, nessa fase, são: percentual de utilização do processador (%) e quantidade de memória RAM utilizada

(*kibibyte (KiB)*). As informações que são registradas são apenas aquelas que são coletadas no momento em que as aplicações estão sendo executadas.

A classificação ou categorização desses dados em tabelas é importante para uma posterior análise qualitativa desses valores; portanto, é imprescindível que o conjunto desses dados sejam suficientes para gerar as métricas propostas nesse trabalho, quais sejam: *RTEU*, *Delay*, *Jitter*, *Required Bandwidth*, *Loss Rate*, *Error Rate*, *Memory*, *Processing*, *Data Sending Rate* e *Transfer*. Cada uma dessas métricas é descrita, detalhadamente, na subseção correspondente.

A geração dessas métricas originam-se do processamento dos dados coletados. A data/hora de envio e recebimento de primitivas é fonte essencial para a geração da maioria das métricas descritas; por isso, é essencial que as máquinas virtuais estejam sincronizadas corretamente. Este trabalho utiliza um protocolo que sincroniza os relógios das máquinas virtuais (*Network Time Protocol (NTP)*) (NTP.BR, 2025). A padronização de tempo utilizada nessas máquinas virtuais é o tempo universal coordenado (*Coordinated Universal Time (UTC)* + 0) (TIME, 2025).

A Subseção 4.2.4.3 descreve e as tabelas 10, 11 e 12 apresentam as métricas propostas nesta tese, geradas a partir do processamento e categorização dos dados coletados, demonstrando, dessa forma, que a coleta realizada tem caráter completo, na medida em que cumpre os objetivos estabelecidos.

Tabela 8 – Lista de métricas propostas para a avaliação qualitativa.

Métrica / Categoria	**Descrição**
RTEU	Tempo decorrido entre envio da solicitação e o recebimento da primeira resposta do usuário.
Delay	Tempo decorrido entre envio e recebimento da informação.
Jitter	Variação do atraso (delay) na entrega de primitivas sucessivas de dados.
Required Bandwidth(kib/s).	Taxa de transferência de dados necessária (*required data transfer rate*) para cada aplicação específica. Essa métrica inclui dados e sobrecarga (overhead).
Loss Rate(%).	Taxa de perda de primitivas entre remetente e receptor.
Error Rate(%).	Taxa de erro de transmissão entre remetente e receptor.
*Memory(*KiB*).*	Quantidade de memória (mínima - máxima) que cada caso de uso utiliza.
Processing(%CPU).	Percentual de processamento máximo da CPU que cada caso de uso utiliza.
Data Sending Rate (primitivas/s)	Taxa de envio de dados da aplicação para a rede FIXP.
*Transfer (*MiB*).*	Quantidade de dados que a aplicação transfere para a rede *FIXP* no decorrer da sua execução.

4.2.4.3 Sobre as métricas

O trabalho propõe 10 métricas para avaliar cada caso de uso proposto. Algumas dessas métricas avaliam a pontualidade (*timeliness*), a precisão (*preciseness*) e a acurá-

cia(*accuracy*) (CHEN; FARLEY; YE, 2004) dos tráfegos de vídeo e texto que os aplicativos IP e ETArch geram; outras, avaliam a sobrecarga da infraestrutura física do *FIXP*.

A Tabela 8 apresenta a descrição de todas as métricas selecionadas.

O intervalo das métricas *RTEU*, *Delay*, *Required Bandwidth* e *Data Sending Rate* (ver tabelas 10, 11 e 12) possui 95% de confiança, com 09 graus de liberdade. Isso quer dizer que 10 experimentos, por caso de uso, geram as médias de cada uma dessas métricas; e esses intervalos tem 95% de chance de contê-las.

4.2.4.4 Sobre as técnicas estatísticas

As técnicas estatísticas aplicadas neste trabalho diz respeito à teoria elementar da amostragem, à teoria estatística da estimação e à teoria das pequenas amostras (SPIEGEL; SCHILLER; SRINIVASAN, 2016).

O planejamento da coleta de dados surge da teoria elementar da amostragem, que é o estudo das relações existentes entre a população (estatística) e as amostras obtidas dela (SPIEGEL; SCHILLER; SRINIVASAN, 2016). A primeira constatação é que a população que fundamenta a avaliação quantitativa deste trabalho é infinita, pois representa um conjunto ilimitado de eventos ou experimentos que podem ocorrer em diversos instantes diferentes e que tem a única função de coletar os dados propostos. Por conta disso, os parâmetros populacionais, como médias ou variâncias, são desconhecidos. No entanto, o trabalho propõe a geração de diversas médias que estão relacionadas a diferentes grandezas: *RTEU*, *Delay*, *Jitter*, *Required Bandwidth*, *Loss Rate*, *Error Rate* e *Data Sending Rate*. Se as médias populacionais são desconhecidas, de onde advém as médias geradas? Elas derivam dos experimentos ou eventos citados, ou seja, da própria amostra.

Por esse motivo, a segunda técnica utilizada diz respeito à Teoria Estatística da Estimação, que na falta dos parâmetros populacionais, consegue deduzi-los, eficientemente, da distribuição amostral correspondente (SPIEGEL; SCHILLER; SRINIVASAN, 2016). Nessa fase, decide-se pela estimativa por intervalo, que indica precisão e exatidão. Cada caso de uso (ver Tabela 7) é testado por 10 experimentos, e cada experimento gera 7 médias aritméticas (uma para cada métrica que gera intervalo), obtendo, no final, 70 médias aritméticas (10 amostras para cada métrica que gera intervalo) para cada caso de uso proposto. Ao todo, no trabalho, são realizados 220 experimentos para os 22 casos de uso da tese. Isso corresponde a análise de mais de 5.000.000 (cinco milhões) de registros de dados.

A terceira técnica estatística é referente à Teoria Das Pequenas Amostras. Ela mostrou-se necessária, porque o número de amostras, quando menor que 30 ($N < 30$), é considerado baixo, e portanto, as distribuições amostrais não se aproximam da função normal ou Gaussiana (SPIEGEL; SCHILLER; SRINIVASAN, 2016); na verdade, quanto menor a quantidade de amostras, pior é essa aproximação (SPIEGEL; SCHILLER; SRINIVASAN, 2016).

Dessa forma, uma opção que melhor se adequa às necessidades deste trabalho é a *distribuição t de student* (SPIEGEL; SCHILLER; SRINIVASAN, 2016), cujos resultados obtidos das distribuições amostrais são válidos tanto para pequenas quanto para grandes amostras. Se as amostras ultrapassam o número 30 (N > 30), a curva da *distribuição t de student* (SPIEGEL; SCHILLER; SRINIVASAN, 2016) se aproxima da curva da distribuição normal.

O nível de confiança das médias das métricas calculadas neste trabalho é de 95%, com 09 graus de liberdade. O intervalo de 95% de confiança significa que se 100 intervalos fossem calculados, baseados em amostras de mesmo tamanho, 95 desses intervalos iriam conter o parâmetro populacional sob estudo. Apenas 5 não conteriam esse valor. Isso quer dizer que o intervalo calculado tem 95% de chance de conter o valor real da população, que no nosso caso, é o valor da média de uma determinada métrica.

Segue abaixo a inequação do intervalo de confiança para as médias das métricas que possuem intervalo de confiança (ver tabelas 10, 11 e 12).

$$\bar{X} - Z_{c(0,975)} \cdot \frac{s}{\sqrt{N}} < \mu < \bar{X} + Z_{c(0,975)} \cdot \frac{s}{\sqrt{N}} \quad \text{, onde}$$

$\bar{X}$ é a média amostral

$Z_{c(0,975)}$é o valor do coeficiente crítico com intervalo de confiança de 95%

s é o desvio padrão amostral

N é o tamanho da amostra

μ é a média real da população e seu valor pertence ao intervalo de confiança calculado (1)

A anotação do coeficiente crítico na equação 1 citada acima tem o padrão da tabela *t de student* apresentada em (SPIEGEL; SCHILLER; SRINIVASAN, 2016).

4.2.4.5 Cenário de Avaliação

O cenário de avaliação da Figura 21 possui cinco entidades comunicantes, dois controladores (um IP e um ETArch) e 05 switches *FIXP*. A figura não descreve a arquitetura de Internet (IP, ETArch, outras) das aplicações que cada entidade representa porque essa informação depende do caso de uso que está sendo executado.

As entidades representam Chats/IP se o caso de uso refere-se ao aplicativo Chat/IP, representam Vídeos/ETArch se o caso de uso refere-se ao aplicativo Vídeo/ETArch, e assim por diante. No entanto, a localização das entidades sempre são as mesmas, independentemente do caso de uso executado.

Os experimentos não utilizam, necessariamente, todas as entidades disponíveis. Somente os casos de uso 13 e 20 utilizam as cinco entidades. O restante utiliza apenas as entidades 1 e 2. A Tabela 7 mostra, detalhadamente, cada caso de uso proposto nesta tese.

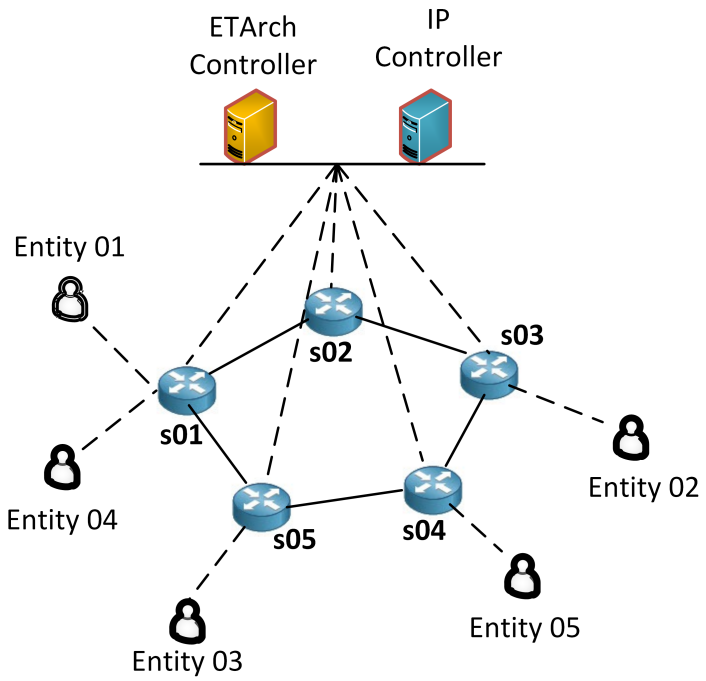

Figura 21 – Cenário de avaliação.

4.2.4.6 Ferramentas tecnológicas utilizadas

As demandas da avaliação quantitativa são muitas e abrangem, principalmente, a implementação dos princípios arquitetônicos da solução e a preparação do ambiente onde a rede *FIXP* possa demonstrar-se como prova de conceito através de casos de uso selecionados.

Vários procedimentos são realizados para que essa demanda se materialize, e cada um dos processos estabelece-se através de ferramentas ou habilitadores tecnológicos.

A Tabela 9 mapeia as ferramentas utilizadas nos principais procedimentos executados para a elaboração desse ambiente que torna possível as execuções experimentais realizadas na rede *FIXP*. Além disso, apresenta os recursos computacionais do *host*, máquina física que comporta a infraestrutura FIXP; como também apresenta os recursos computacionais das máquinas virtuais ou entidades que compõem essa rede.

4.2.4.7 Performance do aplicativo "Chat" das arquiteturas ETArch e IP

Esta subseção discute as medições realizadas na execução dos casos de uso 1 ao 13 (ver Tabela 7), selecionados para cumprir as metas de análise quantitativa da comunicação dos aplicativos "*Chat ETArch*"e "*Chat IP*". Esses aplicativos simulam o funcionamento do protocolo Internet Relay Chat (OIKARINEN, 1993), cuja principal característica é gerar tráfego de pequenos textos em canais de chat que possuem dois ou mais participantes. Dessa forma, a subseção analisa a primeira natureza de tráfego na rede *FIXP*: texto; e, o primeiro aplicativo: chat, que possui seus próprios requisitos para uma boa experiência

Tabela 9 – Ferramentas tecnológicas utilizadas em cada demanda da análise qualitativa.

Máquina *host*.	
Ferramentas / Recursos	**Descrição do procedimento**
Windows 11 Home Single Language (Versão 23H2)	Este é o sistema operacional utilizado na máquina *host*, que comporta a infraestrutura *FIXP*.
Processador: 11h Gen Intel (R) Core (TM) i7-1165G7 @ 2.80GHz **Memória RAM:** 16GB	O *host* é a máquina física que comporta a infraestrutura *FIXP*. O equipamento dispõe de um único processador físico, composto por 04 núcleos de processamento, sendo que cada núcleo consegue processar 2 *threads* simultâneas com *Hyper-Threading*, o que melhora o desempenho de tarefas paralelas. Além disso, possui memórias Cache L1, L2 e L3 de 320 KB, 5 MB e 12 MB, respectivamente.
Entidades ou máquinas virtuais da infraestrutura *FIXP*.	
Ferramentas utilizadas	**Descrição do procedimento**
Processador: 11h Gen Intel (R) Core (TM) i7-1165G7 @ 2.80GHz **Memória RAM:** 1.4GB	Todas as máquinas virtuais possuem os mesmos recursos computacionais. Apenas 1 núcleo do processador está disponível para a demanda de processamento das máquinas virtuais.
Oracle VirtualBox 6.0	Cada entidade comunicante (*switch*, controlador, aplicação, outros) da rede *FIXP* é uma máquina virtual. Por isso é necessário um software de virtualização que gerencie máquinas virtuais e forneça a capacidade de configurar diferentes modos de rede para conectar essas máquinas entre si e com a rede externa.
Debian 9 4.9.0-3-amd64 x86_64 GNU/Linux Ubuntu 16.04.4 4.4.0-116-generic x86_64 GNU/Linux	Cada máquina virtual possui um sistema operacional que permite a instalação dos blocos funcionais da arquitetura *FIXP*, das bibliotecas que representam os protocolos de comunicação, das bibliotecas P4, entre outros. Ubuntu é utilizado nas máquinas virtuais que representam os *switches FIXP*. Todas as outras entidades executam o Debian.
Python 3.5.2	Os blocos funcionais da arquitetura *FIXP*, os protocolos de comunicação, os coletores de dados, entre outros; são desenvolvidos na linguagem Python.
Scapy 3.0	É uma biblioteca do Python que permite a manipulação de pacotes. Essa biblioteca foi utilizada na construção dos algoritmos de coleta de dados, atuando como um *sniffer* de rede. Outra função importante do *Scapy*, neste projeto, é que ele abstrai os *NICs*, possibilitando que os repasses de primitivas sejam realizados através dos nomes dessas interfaces. Isso tem um efeito importante na comunicação interna do *FIXP*, tornando-o independente de quaisquer protocolos de enlace e rede no que tange à comunicação interna dos seus blocos funcionais.
Linguagem $P4_{(16)}$, versão 1.10.0-2319548	Os *switches FIXP* estendem bibliotecas da linguagem $P4_{(16)}$, que oferece a arquitetura "*BMv2 Simple Switch target*" e o compilador dos programas $P4_{(16)}$. O compilador gera a *APIRuntimeP4* e o pipeline de dados do *switch* em tempo de execução. O protocolo *FIXP* estende algumas características da *APIRuntimeP4*, como por exemplo, a capacidade de gerenciar os fluxos da rede e configurar grupos de comunicação *multicast*.

do usuário.

A Tabela 10 apresenta as medições mencionadas. Na primeira linha, existe um conjunto de valores de referência que permite ao usuário ter uma boa experiência na utilização do *Internet Relay Chat* (CHEN; FARLEY; YE, 2004). Essas referências de *QoE* norteiam a discussão da performance dos aplicativos testados.

As médias das métricas *RTEU* e *Delay* estão abaixo dos valores de referência, portanto são bons indicadores de *QoE*.

Tabela 10 – Medições realizadas nas aplicações "*Chat IP*" e "*Chat ETArch*", por caso de uso, na rede FIXP.

QoE Reference - References are required for the chat to provide quality service for a good user experience. (CHEN; FARLEY; YE, 2004)
A.C.N - Analogue of Use Case "N".

Description	**Archit.**	**Response Time Expected by Users(s)**	**Delay(ms)**	**Required Bandwidth (Kib/s)**	**Loss Rate (%)**
QoE Reference	–	$\leq$ 1 second	$<$ 200	$<$ 1	0
Case 1	IP	0,110429 +- 0,089795	4,732 +- 0,426	1476,459126 +- 121,707943	0 +- 0
Case 2	IP	0,065621 +- 0,000285	2,835 +- 0,119	4160,050998 +- 279,790563	0 +- 0
Case 3	IP	0,146975 +- 0,02420	3,504 +- 0,177	18253,175537 +- 2362,786208	0 +- 0
Case 4	IP	0,065949 +- 0,000715	3,847 +- 0,311	2512,945015 +- 117,170182	0,38 +- 0,19
Case 5	IP	0,068426 +- 0,007537	4,225 +- 0,420	15314,047605 +- 1174,127796	0,09 +- 0,06
Case 6	IP	0,064612 +- 0,000541	4,320 +- 0,190	27007,864125 +- 1222,574220	0,017 +- 0,002
Case 7 (A.C.1)	ETArch	0,454821 +- 0,000160	110,504 +- 0,016	38,332808 +- 0,005943	0 +- 0
Case 8 (A.C.2)	ETArch	0,454849 +- 0,000205	110,528 +- 0,011	70,734489 +- 0,037048	0 +- 0
Case 9 (A.C.3)	ETArch	0,454791 +- 0,000497	110,559 +- 0,007	106,057149 +- 0,007503	0 +- 0
Case 10 (A.C.4)	ETArch	0,406451 +- 0,000715	61,169 +- 0,428	151,945415 +- 2,763047	1,16 +- 0,47
Case 11 (A.C.5)	ETArch	0,347252 +- 0,000833	3,880 +-0,084	6709,624267 +- 333,252338	0,51 +- 0,11
Case 12 (A.C.6)	ETArch	0,348721 +- 0,028959	3,210 +- 0,033	10370,736721 +- 782,219881	0,11 +- 0,02
Case 13	ETArch	0,369185 +- 0,017581	118,312 +- 62,786	3310,782583 +- 952,941751	3,05 +- 2,16

Description	**Archit.**	**Error Rate (%)**	**Memory (KiB)**	**Processing (%CPU)**	**Data Sending Rate (primitive/s)**	**Transfer (MiB)**
QoE Reference	–	0	–	–	–	–
Case 1	IP	0 +- 0	77892 - 87608	9,97	145,220636 +- 6,957112	1,03
Case 2	IP	0 +- 0	83956 - 94976	10,03	107,083065 +- 6,55839	15,25
Case 3	IP	0 +- 0	84572 - 97820	9,70	96,281818 +- 0,897336	42,91
Case 4	IP	0 +- 0	81580 - 87876	14,07	315,013501 +- 8,528826	1,03
Case 5	IP	0,000630 +- 0,001431	83260 - 94396	15,53	238,439312 +- 8,960428	15,25
Case 6	IP	0 +- 0	69328 - 101456	15,70	198,310349 +- 1,947803	42,91
Case 7 (A.C.1)	ETArch	0 +- 0	81476 - 91912	10,93	209,255949 +- 3,662403	1,03 / 2,06
Case 8 (A.C.2)	ETArch	0 +- 0	83584 -97700	15,97	161,750077 +- 5,865295	15,25 / 30,50
Case 9 (A.C.3)	ETArch	0 +- 0	89356 - 107940	10,43	126,66652 +- 2,557889	42,91 / 85,82
Case 10 (A.C.4)	ETArch	0 +- 0	66796 - 112528	15,60	598,664600 +- 51,076501	1,03 / 2,06
Case 11 (A.C.5)	ETArch	0 +- 0	77708 - 88216	27,33	360,770285 +- 15,464625	15,25 / 30,50
Case 12 (A.C.6)	ETArch	0 +- 0	68452 +- 95904	22,57	267,722725 +- 9,566481	42,91 / 85,82
Case 13	ETArch	0 +- 0	88680 +- 105744	31,38	203,533368 +- 25,018579	2,58 / 12,90

O valor de referência do *RTEU* é menor ou igual a 1s, os maiores valores coletados são aproximadamente 0.45s e pertencem aos casos 7, 8 e 9 da ETArch. Os valores coletados de *RTEU* dos casos de uso da ETArch são praticamente iguais, giram em torno de [0,34s , 0,46s]. O mesmo acontece em relação aos casos de uso do IP, que giram em torno de [0,06s , 0,15s]. Isso indica que a diferença de tamanho de pacote e a espécie de comunicação (síncrona/assíncrona) não interferem no desempenho dessa métrica. No entanto, os valores coletados nos casos de uso da ETArch são maiores que os valores coletados nos casos de uso do IP: cerca de 3,8 vezes. Isso se dá porque a ETArch possui um controlador proativo e realiza um *handshake* antes da comunicação (ver Figura 18), o que atrasa a primeira resposta ao usuário. Já a arquitetura UDP/IP tem o paradigma de comunicação *push*: a mensagem é simplesmente enviada sem que aja uma negociação do contexto de comunicação ou uma solicitação prévia do recurso (ver Figura 19).

O valor de referência do *Delay* é menor que 200ms, o maior valor coletado é 118,31ms do caso 13 da ETArch. Para casos análogos que possuem arquiteturas de Internet distintas, como por exemplo, casos 1 e 7, o *Delay* da ETArch geralmente é maior. Isso deve-se ao fato de que para casos análogos, a rede *FIXP* processa mais primitivas ETArch do que primitivas IP(ver coluna de Transferência). A ETArch é naturalmente *multicast*, e mesmo para casos de comunicação entre duas entidades, o *switch FIXP* replica a primitiva para todos os participantes do grupo, podendo causar um atraso maior no processamento do pacote. No entanto, em casos excepcionais, como por exemplo, casos 5 e 11; e, casos 6 e 12, mesmo que o *switch* replique as primitivas da ETArch, os Casos 11 e 12 (da Etarch) possuem um *Delay* reduzido, equiparando-se aos casos do IP. Os casos mencionados possuem conexões assíncronas, perda de pacotes e maior sobrecarregamento dos *switches*. Por conta do sobrecarregamento, o esperado é que as taxas de *Delay* dos casos 10, 11 e 12 (assíncronos) sejam maiores que os casos 7, 8 e 9 (síncronos) da ETArch, mas não é o que acontece. Por certo, a taxa de perda de primitivas pode ter influenciado esses valores, pois pacotes que se perdem não tem a métrica computada; e a taxa de perda dos casos 10, 11 e 12 giram em torno de [0,11% , 1,16%], que é considerado alto para um aplicativo de chat.

O valor de referência de *Required Bandwidth* é menor que 1*kibibit (Kib)*/s, ou seja, o chat consiste na transmissão de caracteres simples, portanto, o requisito de largura de banda é bem pequeno, normalmente, inferior a 1*Kib*/s (CHEN; FARLEY; YE, 2004). Todos os casos de uso utilizam uma banda maior do que essa. O caso 3, por exemplo, solicita uma banda de 18253,17 *Kib*/s, ou seja, cerca de 18253 vezes maior. Isso deve-se ao fato de que humanos, não máquinas, utilizam aplicativos baseados em *Internet Relay Chat*. Então o usuário do aplicativo envia uma mensagem pequena, espera uma resposta, logo após envia uma outra mensagem, e assim por diante. Os experimentos desse *paper* submetem os casos de uso da Tabela 10 a grandes estresses através de geradores automáticos de mensagens, o que justifica o alto consumo de banda. No caso 12, por exemplo, cada entidade que participa da comunicação manda 15000 mensagens de 1500 bytes simultaneamente, o que resulta em um montante de 30000 mensagens e aproximadamente 43.945 *kib* enviados à rede *FIXP* de forma assíncrona, em um curto intervalo de tempo. Feito essa análise, percebe-se na tabela que quanto maior o tamanho do pacote, maior a banda requisitada. Como exemplo; verifica-se os casos 1, 2 e 3. Outro ponto importante é que os casos de uso assíncronos solicitam mais banda larga do que os casos síncronos quando os pares são análogos. Como exemplo, verifica-se os pares de casos 1 e 4, 2 e 5, 3 e 6. Outra questão é que em casos análogos de ETArch e IP, a ETArch precisa de menos banda do que a arquitetura IP para transferir o mesma quantidade de mensagens. Como exemplo, veja casos 1 e 7, 2 e 8, e outros. Nos casos 1 e 7, o IP solicita 38 vezes mais banda que a ETArch.

Quanto ao *Loss Rate*, alguns casos de uso se mostram ineficientes na rede *FIXP*,

estão acima do valor de referência esperado, que é zero para esse tipo de aplicativo. As perdas acontecem nos casos 4, 5, 6, 10, 11, 12 e 13; todos eles possuem conexão assíncrona. Nesses casos, as entidades que participam da comunicação enviam mensagens simultaneamente. No caso 12, por exemplo, cada entidade envia 15000 mensagens de 1500 bytes. Essa transmissão contínua, certamente, sobrecarrega os algoritmos de arbitragem e/ou as filas das interfaces de Entrada ou Saída do *switch FIXP*, provocando descartes de primitivas. Em casos análogos, a ETArch perde mais primitivas do que o IP, pois o *switch FIXP* replica mensagens ETArch, adicionando sobrecarga no processamento da primitiva. Como exemplo, verifica-se os casos 4 e 10, 5 e 11, 6 e 12. Outro ponto importante é que quanto menor o tamanho da mensagem, maior é a taxa de perda. Isso acontece porque o tamanho do pacote influencia a taxa de envio de primitivas das aplicações para a rede. Quanto menor o tamanho do pacote, mais primitivas por segundo o *switch FIXP* tem que processar, o que aumenta o sobrecarregamento e a inevitável perda de dados. Os casos 4, 5 e 6; e os casos 10, 11 e 12 corroboram essa afirmação. As taxas de perda coletadas são pequenas, qualquer topologia de rede perde dados quando sobrecarregada. A solução desse problema passa pela inclusão de serviços de confiabilidade na comunicação IP e ETArch.

Em relação à *Error Rate*, o valor de referência para esse aplicativo também é zero. Só o caso 5 possui um pequeno percentual de erro que gira em torno de 5 casas decimais. Como no caso de *Loss Rate*, as aplicações ETArch e IP podem resolver essa situação facilmente incrementando serviço de confiabilidade à conexão. O erro apresentado, certamente, trata-se de um *outlier*.

Em relação às métricas que analisam a sobrecarga dos *switches FIXP*, o caso 13 é um dos que consomem mais memória e processamento. A rede *FIXP* recebe aproximadamente 2,58 *Mebibyte (MiB)* e processa 12,90 *MiB* de dados; e, a aplicação envia cerca de 203,53 primitivas/s. A diferença no número de dados processados pela rede FIXP e enviados pela aplicação deve-se ao fato de que o *switch FIXP* replica as primitivas ETArch para 05 entidades que participam da comunicação. Esse caso de uso consome até 104 *MiB* de memória e 31,38% do processamento de um núcleo de CPU virtual de 2,80 GHz. Outro ponto importante, é que os casos de uso 3, 6, 9 e 12; apesar de enviarem à rede o montante de 42,91 *MiB* com taxa de até 267,72 *primitivas/s* (medições maiores que no caso 13 anterior), ainda assim conseguem manter a utilização de memória RAM e processamento em um patamar aceitável (consumo menor ou equivalente ao caso 13). Esses valores de consumo da infraestrutura dão uma estimativa para que operadores de rede possam prever gargalos antes que aconteçam. Nos experimentos realizados, nenhum dos casos de uso sobrecarrega a topologia *FIXP* em relação ao processamento e à utilização de memória.

A taxa de envio de dados (*Data Sending Rate*) é influenciado pelo tamanho da primitiva. Quanto menor o tamanho das mensagens, maior é a taxa de envio de dados para a

rede *FIXP*. Ver casos 1, 2 e 3; 4, 5 e 6. Outro fator importante é que para casos análogos, os casos de uso assíncronos possuem taxa de envio maior que os casos síncronos. Ver, por exemplo, casos 1 e 4, 2 e 5, 3 e 6. Esse resultado é esperado, já que todos os participantes dos casos de uso que possuem conexões assíncronas enviam suas mensagens, simultaneamente, para a rede *FIXP*. Essa métrica é um bom indicativo de sobrecarregamento, pois quanto maior a taxa de envio, maior é a taxa de perda de primitivas. Ver casos 4, 5 e 6; e casos 10, 11, 12. Essa métrica também revela que o *switch FIXP* processa melhor primitivas maiores com menores taxas de envio do que o inverso. Ver casos 4, 5 e 6; e, casos 10, 11 e 12. A melhora da transmissão, nesses cassos de uso, ocorre basicamente por três motivos: diminuição da latência e da perda de pacotes; e, aumento considerável da taxa real de transferência.

Em relação à métrica *Transfer*, a aplicação de casos de uso análogos transferem a mesma quantidade de dados: ver casos 1 e 7; e, casos 2 e 8. A coluna exibe a replicação das primitivas da ETArch no núcleo da rede. Um exemplo é o caso 7. A aplicação transfere 1,03 *MiB* de dados para a rede *FIXP*, porém essa rede replica essa informação para um grupo *multicast* de 2 entidades, e a partir daí, a rede *FIXP* tem que lidar com 2,06 *MiB* de dados.

Na prática, quando os aplicativos são executados, o chat de todos os casos de uso propostos funciona bem. Quando uma entidade envia a mensagem, todas as outras que estão na comunicação recebem-na quase que instantaneamente. O caso de uso 13 é especial, pois é o único que possui mais de 2 participantes no canal de comunicação.

4.2.4.8 Performance do aplicativo de "*streaming* de vídeo on Demand" das arquiteturas ETArch e IP

Esta subseção discute as medições realizadas na execução dos casos de uso 14 ao 20 (ver Tabela 7), selecionados para cumprir as metas de análise quantitativa da comunicação dos aplicativos "*Vídeo de Streaming on Demand da ETArch*" e "*Vídeo de* streaming *on* Demand *do IP*". Esses aplicativos estendem as bibliotecas do *FFmpeg* (versão 3.2.18) (FFMPEG, 2024) e têm como principal característica a geração de tráfego de vídeo *on demand*, onde pedaços (*chunks*) desse vídeo trafegam pela rede e são reproduzidos individualmente no cliente de forma independente do vídeo maior.

A Tabela 11 apresenta as medições citadas. Na primeira linha, existe um conjunto de valores que permite ao usuário ter uma boa experiência na utilização de aplicações que trafegam vídeos no padrão *Moving Pictures Expert Group 4 (MPEG-4)*. Essas referências (CHEN; FARLEY; YE, 2004) norteiam a discussão da performance dos aplicativos testados. Caso nessa seção existam fatos semelhantes aos da Subseção 4.2.4.7, esses fatos são novamente relatados sem a justificativa da sua causa, que já se encontra na subseção citada.

O valor de referência do *RTEU* é de alguns segundos (*a few seconds*), portanto consi-

Tabela 11 – Medições realizadas nas aplicações "*Streaming de Vídeo On Demand IP*" e "*Streaming de Vídeo On Demand ETArch*", por caso de uso, na rede FIXP.

QoE Reference - References are required for the Interactive Video on Demand to provide quality service for a good user experience. (CHEN; FARLEY; YE, 2004)
A.C.N - Analogue of Use Case "N". **RTEU** - Response Time Expected by User **Video format**: *MPEG-4*

Description	Archit.	RTEU(s)	Delay(ms)	Jitter(ms)	Required Bandwidth (Kib/s)	Loss Rate (%)
QoE Reference	–	A few seconds	< 150	< 150	Local Network: 1536	< 0.001
Case 14	IP	0,065651 +- 0,000385	3,479 +- 0,120	3,210 +-0,060	1991,706316 +- 73,825258	0,06 +- 0,03
Case 15	IP	0,065284 +- 0,000715	2,983 +-0,257	4,315 +- 0,021	3909,374934 +- 173,976582	0,09 +- 0,05
Case 16	IP	0,069020 +- 0,007638	2,947 +- 0,108	6,105 +- 0,008	5078,739515 +- 208,710232	0,07 +- 0,04
Case 17 (A.C.14)	ETArch	0,344846 +-0,000201	800,913 +- 21,781	4,206 +- 0,082	11,182337 +- 3,092676	14,03 +- 0,70
Case 18 (A.C.15)	ETArch	0,347233 +- 0,000715	284,034 +- 60,010	4,540 +- 0,071	314,386295 +- 55,364309	2,38 +- 0,40
Case 19 (A.C.16)	ETArch	0,347232 +-0,000455	82,385 +- 9,727	6,151 +-0,060	1420,965445 +- 82,350500	0,28 +- 0,16
Case 20	ETArch	0,353823 +-0,000418	499,314 +- 78,113	1,769 +- 0,019	751,407275 +- 197,386324	0,45 +- 0,18

Description	Archit.	Error Rate(%)	Memory (KiB)	Processing (%CPU)	Data Sending Rate (primitive/s)	Transfer (MiB)
QoE Reference	–	< 0.001	–	–	–	–
Case 14	IP	0 +- 0	87848 - 101764	13,00	311,938891 - 5,968512	11,26
Case 15	IP	0 +- 0	89832 - 105188	14,60	231,969783 +- 1,175982	15,22
Case 16	IP	0 +- 0	119420 - 140344	10,27	163,923869 +- 0,240824	16,12
Case 17 (A.C.14)	ETArch	0 +- 0	87332 - 162268	15,70	279,563792 +- 5,624201	10,09 / 20,18
Case 18 (A.C.15)	ETArch	0 +- 0	77728 - 98920	12,57	228,083547 +- 3,154424	15,06 / 30,12
Case 19 (A.C.16)	ETArch	0 +- 0	90016 - 103972	10,53	163,195112 +- 1,647651	16,06 / 32,12
Case 20	ETArch	0 +- 0	121904 - 147696	18,68	143,742932 +- 1,399224	14,36 / 71,80

dera que o usuário pode esperar alguns segundos desde a primeira requisição para que o vídeo comece a ser executado. Os casos de uso 14 à 20 trafegam vídeos com tamanhos de pacotes e conexões diferentes (ver Tabela 7), porém esses critérios não influem no valor dessa métrica. O *RTEU* dos casos 14, 15 e 16 do IP é de cerca de [0.06s, 0.07s] e os casos 17, 18 e 19 da ETArch é de cerca de [0.34s, 0.36s]. Ambos os aplicativos desempenham, conforme indicador de referência, uma boa qualidade de comunicação.

O valor de referência do *Delay* é menor que 150 ms. Os casos 14, 15 e 16 do IP cumprem essa meta. Os casos de uso da ETArch ficam acima do valor de referência; a não ser o caso 19, que possui uma latência de aproximadamente 82,38 *ms*. Outro ponto é que o tamanho das primitivas influem na métrica: quanto maior o tamanho do pacote, menor o *Delay*. O caso 20 foge a essa regra porque é o único que envia mensagens para mais de duas entidades. Dos três casos de uso do IP - 14, 15 e 16 - o caso 16 é o que trafega primitivas de maior tamanho (1500 bytes), e, portanto, possui a menor latência e a menor taxa de envio de dados da aplicação para a rede *FIXP*. Os casos 17, 18 e 19 da ETArch apresentam esse mesmo padrão. O motivo para as altas latências dos casos da ETArch, principalmente nos casos 17 e 18 (que estão fora do intervalo de referência), são dois: os switchs FIXP replicam primitivas ETArch no núcleo da rede, portanto, o número de primitivas ETArch que trafegam no núcleo FIXP, nesses dois casos de uso, é maior que os pares de casos de uso análogos do IP (ver coluna *Transfer*); e, o switch FIXP lida melhor em processar primitivas ETArch maiores (de 1000 e 1500 bytes) com

menores taxas de recebimento do que o inverso.. Os casos 17, 18 e 19 corroboram essa afirmação: quanto maior a primitiva ETArch e menor a taxa de envio de primitivas pelas aplicações, melhor é a comunicação, pois aumenta a taxa real de transferência e diminui a taxa de perda da conexão.

O *Jitter* é uma métrica muito importante para a qualidade do vídeo. A referência para uma boa transmissão é de 150 ms. Todos os casos de uso conseguem manter essa métrica abaixo do valor de referência, indicando uma boa qualidade de transmissão. Os casos de uso do IP transmitem o vídeo com *Jitter* de aproximadamente [3,2ms , 6,2ms]. O *Jitter* dos casos da ETArch é de cerca de [1,7ms , 6.2ms].

De acordo com os valores de referência, para aplicações típicas de vídeo como é o caso de transmissões *MPEG-4*, a largura de banda para uma rede local do usuário é de 1536 *Kib*/s (CHEN; FARLEY; YE, 2004). As aplicações IP estão solicitando mais banda da rede do que o indicador de referência sugere, ou seja, possui uma taxa média real de transferência maior do que o esperado. Teoricamente, esse fato não atrapalha a qualidade da transmissão, pois quanto maior a taxa média da transmissão, melhor a qualidade de experiência do usuário. Os casos de uso 14, 15 e 16 mostram os valores de banda larga consumidos pela aplicação da arquitetura IP, que são 1991,70 *Kib*/s; 3909,37 *Kib*/s e 5078,73 *Kib*/s, respectivamente . Em relação à ETArch, a banda larga consumida pelo caso de uso 19 está bem próxima do esperado: 1420,96 *Kib*/s. Não à toa, o caso 19, na prática, é a melhor transmissão da arquitetura ETArch, tendo a menor taxa de perda entre os casos de uso da arquitetura. No entanto, os casos 17 e 18 são problemáticos. O caso 17 está abaixo do indicador de referência, sinalizando a utilização de 11 *Kib*/s; e o caso 18, também abaixo, utiliza 314,86 *Kib*/s. O motivo da taxa média real de transferência do caso 17 ser baixa é a alta latência (*Delay*) do tráfego. Essa alta latência (*Delay*) deve-se ao fato de que a transmissão de vídeo em primitivas de 542 bytes pelos aplicativos ETArch é ineficiente, pois aumenta a taxa de envio de dados da aplicação para a rede *FIXP*, ocasionando sobrecarregamento dos equipamentos de comutação. Esse fato une-se a outro: a replicação do tráfego de vídeo no núcleo da rede. Isso reflete nos valores das métricas, pois, por um lado, aumenta o *Delay* e o *Loss Rate*, por outro, diminui a taxa de transferência real da aplicação, causando, dessa forma, uma má experiência do usuário na utilização do aplicativo. Quanto ao caso 18, a aplicação envia mensagens de 1000 bytes, melhora um pouco o consumo de banda larga para 314,38 *Kib*/s, mas também não é suficiente para uma boa transmissão que garanta uma boa experiência do usuário. A justificativa da taxa média real de transmissão estar abaixo da meta esperada é o mesmo do caso 17, já relatado acima. Quanto à medição dessa métrica em todos os casos de uso, um padrão é encontrado: quanto maior o tamanho das primitivas, mais banda larga as aplicações consomem e, por conseguinte, maior a taxa real de transferência das aplicações (remetente para receptor); porque menor é a latência. Isso mostra dois fatos importantes: a taxa real de transferência média das aplicações aumenta, e por conseguinte, aumenta

a qualidade da comunicação; e, o pipeline do switch *FIXP* apresenta maior eficiência operacional ao processar primitivas maiores com taxas de recebimento menores do que o inverso, no mesmo intervalo de tempo.

Quanto ao *Loss Rate*, a referência para *streaming* de vídeo é menor que 0,001%. Todos os casos de uso possuem taxas de perdas maiores que esse indicador, sinalizando problemas na comunicação. No entanto, os casos 14, 15, 16 e 19 possuem as taxas de perda mais baixas, entre [0.06%, 0.28%]. Novamente, o caso 17 apresenta resultados que estão fora do limite ideal, com perda de 14,03% dos dados enviados; mostrando que a ETArch, na rede *FIXP*, não trafega vídeos com boa qualidade quando o tamanho das primitivas é de 500 bytes. O caso 18 da ETArch trafega primitivas de 1000 bytes, a taxa de perda cai, mas ainda é alta, de 2,38%. O caso 19 fornece a melhor transmissão de vídeo da ETArch no *FIXP*, trafegando primitivas de 1500 bytes, com 0,28% de perdas. Essa taxa ainda é alta quando comparada com o caso de uso análogo do IP, que tem taxa de aproximadamente 0,07%, cerca de 4 vezes menor.

Em relação ao *Error Rate*, o percentual de referência é menor que 0,001. Em relação aos casos de uso propostos, nenhuma primitiva apresentou erro de transmissão.

Em relação às métricas que analisam a sobrecarga dos switches *FIXP*, o caso 17 é o que mais consome memória. A quantidade de informações enviada é aproximadamente 10,09 *MiB* com taxa de envio de 279,56 primitivas/s. O switch *FIXP* replica as primitivas ETArch, processando 20,18 *MiB* de dados. Esse caso consome até 158,46 *MiB* de memória. O caso 20 é o que mais consome processamento de uma CPU virtual de 2.80 GHz, um consumo de até 18,68%. Isso deve-se ao fato de que o switch *FIXP* processa cerca de 71,80 *MiB* de dados, maior quantidade de todos os casos de uso, a uma taxa de 143,74 primitivas/s. Esperava-se que o caso 20 também consumisse maior quantidade de memória, mas como as coletas dessa métrica referem-se ao mínimo e máximo de memória utilizada em cada instante, o caso 17 teve alguns *outliers* que ultrapassaram os valores máximos de consumo do caso 20, que processa mais informações, porém com taxas menores de envio de primitivas (ver *Data Sending Rate*). Os *outliers* de medições de sobrecarregamento são importantes, na medida em que operadores de rede tomam ciência desses picos de utilização de recursos computacionais que podem sobrecarregar os equipamentos em determinados instantes. No mais, em relação aos experimentos realizados, nenhum dos casos de uso sobrecarregam a topologia FIXP em relação ao processamento e à utilização de memória.

As métricas transferência (*Transfer*) e *Data Sending Rate* seguem o mesmo padrão dos casos de uso da Subseção 4.2.4.7.

Na prática, as melhores transmissões de vídeo acontecem nos casos 15 e 16 do IP; e, 19 e 20 da ETArch. Isso mostra que a rede *FIXP* processa com mais eficiência primitivas maiores em menores quantidades do que o inverso, no mesmo intervalo de tempo. No caso da ETArch, os casos 19 e 20 enviam primitivas de 1500 bytes; no caso do IP, os casos

15 e 16 enviam primitivas de 1000 e 1500 bytes, respectivamente. O caso 20 é especial (ver Tabela 11), pois é o único que possui mais de 2 entidades na comunicação. Nesse caso de uso, o DTSA da ETArch (entidade 1) envia *streaming* de vídeo on *demand* para 4 consumidores ETArch (entidades 2, 3, 4 e 5), conforme cenário de avaliação mostrado na Figura 21.

4.2.4.9 Performance de serviços de "Chat" e "*streaming* de vídeo on demand" de um aplicativo multiarquitetura das arquiteturas ETArch e IP

Esta subseção, diferentemente das duas anteriores, não discute as medições realizadas na execução dos casos de uso 21 ao 22 (ver Tabela 7). No entanto, os dados da medição do aplicativo multiarquitetura são exibidos na Tabela 12 para averiguação. A razão para não discutir esses dados é que o aplicativo multiarquitetura gera naturezas de tráfego já discutidas anteriormente: texto e vídeo *on demand*. Uma nova discussão seria redundante. A única diferença é que a aplicação multiarquitetura envia esses tráfegos, simultaneamente, na rede FIXP.

Tabela 12 – Medições realizadas nos serviços de *"streaming de vídeo on demand"* e "*Chat*" executados por uma aplicação multiarquitetura que utiliza frameworks da arquitetura ETArch e IP.

QoE Reference - References are required for the Interactive Video on Demand to provide quality service for a good user experience. (CHEN; FARLEY; YE, 2004)
A.C.N - Analogue of Use Case "N". **RTEU** - Response Time Expected by User **Video format**: *MPEG-4*

Description	Archit.	RTEU(s)	Delay(ms)	Jitter(ms)	Required Bandwidth (Kib/s)	Loss Rate (%)
QoE Reference	Chat/–	$\leq$ 1 second	$<$ 200	–	$<$ 1	0
	Video/–	A few seconds	$<$ 150	$<$ 150	Local Network: 1536	$<$ 0.001
Case 21	Chat/ETArch	0,348241 +- 0,001600	75,476 +- 14,886	–	756,870592 +- 21,201313	0,49 +- 0,39
	Video/IP	0,066876 +-0,000380	4,374 +- 0,279	6,247 +- 0,048	3073,265153 +- 85,856613	0,004 +- 0,003
Case 22	Chat/IP	0,067934 +- 0,000715	5,199 +- 0,198	–	1255,044533 +- 25,330492	0,01 +- 0,02
	Video/ETArch	0,348388 +- 0,003288	132,268 +- 9,808	6,304 +- 0,001	755,653082 +- 23,469392	0,60 +- 0,11
Description	**Archit.**	**Error Rate(%)**	**Memory (KiB)**	**Processing (%CPU)**	**Data Sending Rate (primitive/s)**	**Transfer (MiB)**
QoE Reference	Chat/–	0	–	–	–	–
	Vídeo/–	0	–	–	–	–
Case 21	Chat/ETArch	0	75488 - 106444	11,13	273,803838 +- 7,897630	0,51 / 1,03
	Vídeo/IP	0	75488 - 106444	11,13	160,098544 +- 1,245694	15,76
Case 22	Chat/IP	0	82356 - 97716	11,57	126,018358 +- 2,606260	0,51
	Vídeo/ETArch	0	82356 - 97716	11,57	159,713190 +- 0,894141	15,71 / 31,42

Esta subseção demonstra a execução dos casos de uso 21 e 22 presentes na Tabela 12. Cada um desses casos de uso representa uma aplicação multiarquitetura que executa dois serviços que utilizam frameworks de arquiteturas de Internet distintas. O caso 21 possui um serviço de chat da arquitetura ETArch e um serviço de *streaming* de vídeo on *demand* da arquitetura IP. O caso 22 é o inverso, o chat é da arquitetura IP e o *streaming* de vídeo é da ETArch.

O intuito dos casos 21 e 22 são dois: provar, conceitualmente, a comunicação de uma aplicação multiarquitetura na rede FIXP, simulando a execução de microsserviços de diferentes *FIAs* na mesma aplicação ou máquina virtual, um fato bem frequente nas redes atuais; e, apresentar a visão que o FIXP possui de um aplicativo multiarquitetura.

Essa aplicação não tem, por exemplo, a visão do trabalho (GUIMARãES et al., 2019): um servidor WEB que fornece recursos em IP e um *browser* PURSUIT que requisita recursos desse servidor. A comunicação de uma aplicação como essa é possível através do processo de interoperação, o que adiciona sobrecarga nos dispositivos de rede. Outros trabalhos como (MACHADO; DOUCETTE; BYERS, 2015) têm a seguinte visão: uma aplicação multiarquitetura possui todas as pilhas de protocolos nas entidades comunicantes e nos elementos de comutação para se comunicar. Esse tipo de visão sobrecarrega os elementos de comutação, pois se existem 1000 arquiteturas de Internet distintas, 1000 pilhas de protocolos têm que ser instaladas nos switches e nos sistemas finais para que as aplicações dessas FIAs possam se comunicar. Outros problemas referentes a cada uma dessas visões estão descritos na Subseção 2.2.6.

A visão do FIXP é mais simples e eficiente. A solução está no sistema final. A aplicação multiarquitetura do FIXP utiliza frameworks da ETArch e do IP, portanto, possui serviços desenvolvidos em arquiteturas de Internet distintas. A rede FIXP possibilita a comunicação entre serviços que possuem a mesma arquitetura: serviços IP comunicam-se com serviços IP, serviços ETArch comunicam-se com serviços ETArch, e assim por diante. Dessa forma, aplicações multiarquitetura comunicam-se entre si de forma transparente e sem adições de operações que causam sobrecarga no núcleo ou na borda da rede.

4.3 Resposta das questões de pesquisa

Esta seção responde as questões de pesquisa propostas na seção 1.6, revelando, dessa forma, como cada desafio é superado. Algumas respostas mostram que o *FIXP*, na sua versão inicial, possui algumas limitações que não dizem respeito aos seus princípios arquitetônicos, mas à implantação desses princípios nas redes atuais, indicando, dessa forma, temas que devem ser debatidos em trabalhos futuros.

***Questão 1**: Se cada* FIA *possui seus próprios switches, como adequá-los a outra rede de comunicação?*

Esta resposta possui duas partes e as duas abordagens comentam fatos descritos na Seção 3.6, na Subseção 4.2.3 e apresentados nas figuras 1 e 17.

A primeira abordagem supõe que os domínios administrativos de cada *FIA* possui seus próprios aplicativos, *switches*, controladores; isto é, cada *FIA* possui sua própria infraestrutura de comunicação. Desse modo, a priori, a *FIA* que vai se integrar ao *FIXP* não precisa, imediatamente, substituir sua infraestrutura nativa pela infraestrutura do *FIXP*. Nesse caso, o *FIXP* funcionaria como um ponto de troca, possibilitando o envio e

recebimento de mensagens das aplicações que participam da rede e que estão em domínios administrativos distintos. As figuras 1 e 17 mostram essa situação. Na Figura 1, domínios ETArch comunicam-se entre si, domínios NovaGenesis comunicam-se entre si; e, assim por diante. Na Figura 17, a comunicação acontece do mesmo modo que na Figura 1 e quem possibilita a comunicação entre as aplicações é o IXP, que possui um núcleo *FIXP* com três controladores distintos, sendo que cada um desses controladores suporta uma das arquiteturas integradas: ETArch, NovaGenesis e IP. O *FIXP*, como ponto de troca, possibilita a integração das *FIAs* de maneira gradual, sem que elas precisem alterar em nada as suas infraestruturas. Na Figura 1, domínios ETArch, NovaGenesis e IP possuem suas próprias redes.

Quanto a segunda abordagem, supõem-se que essas *FIAs* modifiquem, no decorrer do tempo, suas infraestruturas para *FIXP*. Essa suposição vem do argumento de que operadoras não desejam replicar infraestruturas de cada *FIA* em seus domínios por questões financeiras e de gestão de rede. A rede *FIXP* é tida como uma metarede, pois possui a capacidade de absorver as funcionalidades de uma *FIA* nos planos de dados e controle. Na prática, a rede *FIXP*, através de um programa P4, permite o reconhecimento de quaisquer protocolos e serviços que os *switches* nativos de cada *FIA* reconhece; e, as funcionalidades de controle da arquitetura podem ser transportadas para um controlador *SDN-FIXP*. Na Figura 17, o domínio *FIXP* 1 suporta três arquiteturas de Internet distintas: IP, ETArch e NovaGenesis. Trata-se de um domínio administrativo que já está totalmente integrado ao *FIXP*, com planos de dados e controle *FIXP*. Isso quer dizer que o *switch* FIXP, nesse domínio, consegue reconhecer todas as primitivas dessas três arquiteturas; e, que o plano de controle possui pelo menos três controladores, onde cada um deles é responsável por uma das arquiteturas integradas. A ideia é que os domínios, no decorrer do tempo, formem uma rede única padrão que possibilite a coexistência entre aplicações que utilizem diferentes arquiteturas de Internet.

***Questão 2**: Se cada* FIA *possui diferentes funcionalidades, paradigmas de comunicação e orientações de design; como adaptá-las à infraestrutura de comunicação?*

Orientações de design, paradigmas de comunicação e funcionalidades resumem-se em protocolos, primitivas e procedimentos operacionais de controle nativos de uma determinada arquitetura de Internet. Esses procedimentos de controle específicos de cada *FIA* são colocados em uma peça de software externa logicamente centralizada, que é o controlador SND-*FIXP*. Em outras palavras, cada arquitetura integrada ao *FIXP* tem de fornecer um controlador que possui todos os seus serviços de controle. Nessa peça de software, são implementados os protocolos, os serviços, os procedimentos operacionais e as aplicações que gerenciam a rede. O protocolo *FIXP* fornece uma *Southbound API* que abstrai a complexidade da rede subjacente (do plano de dados) e possibilita a comunicação das aplicações de gerenciamento desse controlador com os *switches FIXP*. As Seções 3.7.2, 3.7.3 e 3.7.4 mostram, respectivamente, os procedimentos operacionais para inte-

grar ETArch, IP e NovaGenesis na rede *FIXP*; cada uma com suas próprias orientações de design, paradigmas de comunicação e várias outras funcionalidades de controle.

***Questão 3**: Como reconhecer a semântica e a sintaxe de diferentes protocolos de rede que possuem diferentes esquemas de endereçamento?*

Quem reconhece as peculiaridades sintáticas dos cabeçalhos que compõem as primitivas das arquiteturas de Internet são os controladores que cada *FIA* disponibiliza e os *switches FIXP*; porém, apenas o controlador é capaz de reconhecer o significado semântico de cada campo que compõe os cabeçalhos. A rede *FIXP* permite, através de um programa *P4*, que o operador de rede (que pode ser o controlador) especifique o pipeline de dados do *switch FIXP*, gerenciando, dessa forma, seu comportamento lógico. Assim, cada *FIA* configura os cabeçalhos de suas próprias primitivas, proporcionando o reconhecimento sintático de todos os seus protocolos. Nessa fase, a *FIA* também cria tabelas de encaminhamento, ações, chaves de correspondência (matching key), registradores que armazenam estados de comunicação, entre outros. Esses registradores podem auxiliar procedimentos operacionais de uma *FIA*, como por exemplo, roteamento com estado (*stateful routing*). Após definição da especificação do plano de dados, se o *switch FIXP* não identifica o endereço de destino de determinada primitiva nas tabelas de encaminhamento, ele envia-a para o controlador. Cada controlador possui os serviços de controle e gerenciamento de uma determinada *FIA*, e por conta disso, reconhece a sintaxe e a semântica dos cabeçalhos das suas primitivas (tais como os campos de endereçamento) e toma as devidas providências. Geralmente, ele executa um algoritmo de roteamento que otimiza o caminho entre dois ou mais pontos, e a partir daí, configura o plano de dados para que as próximas primitivas que possuam o mesmo esquema de endereçamento sejam enviadas para o receptor correspondente. No caso da ETArch (ver Seções 2.1.6 e Subseções 3.7.1 e 3.7.2), o *switch FIXP* é capaz de reconhecer sintaticamente os cabeçalhos da primitiva, mas quando não identifica, nas tabelas de encaminhamento, o endereço de destino MAC dentro do cabeçalho Ethernet, envia a primitiva para o controlador. Ele conhece a sintaxe (extrai os cabeçalhos) e a semântica (o significado de cada campo) dos cabeçalhos da primitiva. O endereço MAC de destino do quadro Ethernet, nessa arquitetura, significa o identificador de um *workspace*, canal lógico *multicast* da ETArch. Dependendo da primitiva de controle recebida, o controlador pode criar um novo *workspace*, vincular uma entidade comunicante ao *workspace* ou estender o *workspace* para que as mensagens enviadas cheguem a outros receptores que demandam o recurso.

***Questão 4**: Como suportar a evolução dos protocolos, funcionalidades e princípios arquiteturais de uma* FIA *que já se encontra integrada na rede?*

A evolução de protocolos, das funcionalidades de controle ou dos princípios arquiteturais passam por dois pontos: modificação da estrutura das primitivas e/ou evolução dos serviços de controle. Quanto à evolução dos serviços de controle, o autor da implementação pode atualizar as aplicações de controle da rede de determinada *FIA*, já que

o plano de controle do *FIXP* é configurável, extensível e programável. Se for necessário, dependendo da versão do formato da primitiva que chega ao controlador, ele pode executar serviços de controle distintos, um para cada versão. Em relação à evolução do formato das primitivas, o plano de dados do *FIXP* também é extensível, configurável e programável. Dessa forma, a rede *FIXP* possibilita que o operador da rede, que pode ser o controlador da *FIA*, atualize o programa *P4* que define o pipeline de dados da rede física, permitindo, dessa forma, o reconhecimento da evolução dos protocolos existentes ou de novos protocolos que podem ser implementados futuramente.

***Questão 5**: Como instalar diversos protocolos e funcionalidades de* FIAs *distintas sem sobrecarregar os dispositivos de comutação do plano de dados?*

A integração de uma nova *FIA* no *FIXP* não adiciona processos adicionais no plano de dados, conforme acontece nos trabalhos correlatos (ver Subseção 2.2.6, 4.2.1 e 4.2.2). Alguns dos trabalhos acrescentam processo de interoperação, outros acrescentam pilhas de protocolos para cada *FIA* integrada à solução (ver Tabela 2). O *FIXP*, resolve esta questão, através do paradigma *SDN-P4*. A separação dos planos de dados e controle possibilita que cada *FIA* integrada ao *FIXP* transfira suas funcionalidades de controle e gerenciamento para uma peça de software externa logicamente centralizada. Dessa forma, cada *FIA* integrada fornece um controlador *SDN-FIXP* que gerencia a rede de acordo com sua orientação de design, paradigma de comunicação, processo de resolução de nomes, esquemas de endereçamento, entre outros. Por outro lado, um programa *P4* especifica um *pipeline* de dados, no *switch FIXP*, para cada *FIA*. Dessa forma, o *switch* FIXP é simples, desprovido de funcionalidades de controle. Ele possui, basicamente, funções de repasse e de configuração de grupos multicast que possibilita à FIA replicar suas primitivas no núcleo da rede.

***Questão 6**: Como mitigar a dificuldade de integração de novas* FIAs *em relação aos trabalhos correlatos?*

A explicação dessa questão de pesquisa está detalhada na Subseção 4.2.3. As Tabelas 2 e 5 mostram que o nível de dificuldade de implementar uma nova *FIA* em soluções que possuem o mesmo objetivo arquitetônico do *FIXP* é alto e a explicação desse fato está na Subseção 2.2.6.

De forma sintética, para implantar uma nova *FIA* no *FIXP*, basta o fornecimento de um controlador vinculado à interface *SouthBound* do *FIXP* e uma especificação que descreve a configuração do pipeline de dados que reconhece, no plano de dados, os serviços e protocolos da arquitetura. Essa especificação define o comportamento dos *switches* e é realizada em uma linguagem conhecida (*P4*). Se a *FIA* a ser implantada é *SDN-P4*, só uma ação precisa ser tomada: vincular o controlador *SDN-P4* à camada *Southbound* do *FIXP*. Se a *FIA* a ser implantada é *SDN*, mas não utiliza a linguagem *P4*, duas ações devem ser tomadas: adequar o controlador ao protocolo *FIXP* (ação análoga à anterior) e criar um documento *P4* que programa o pipeline de dados da arquitetura. Essas duas

situações de implantação têm um nível de dificuldade baixo em relação aos trabalhos relacionados. ETArch é um exemplo de arquitetura *SDN-OpenFlow*, e portanto, de fácil implementação. Por outro lado, se a arquitetura não tem um controlador *SDN*, uma ação adicional deve ser tomada: transferir as funcionalidades de controle dessa arquitetura para um controlador FIXP. Mesmo com esse passo adicional, considera-se ainda baixo, em relação aos trabalhos relacionados, o nível de dificuldade de implantação dessas arquiteturas. NovaGenesis é um exemplo de implantação que precisou desenvolver um novo controlador *SDN-FIXP*. O nível baixo de dificuldade de implantar o NovaGenesis deve-se ao fato de que as suas funcionalidades de controle foram apenas transferidas para o controlador; em nenhum momento, os códigos foram interpretados ou reescritos em outras linguagens de programação ou outros princípios arquitetônicos.

Nos trabalhos correlatos, em geral, o autor da implementação tem que conhecer, detalhadamente, especificações de todas as arquiteturas envolvidas. No caso das soluções de interoperação, o autor tem que conhecer a especificação das arquiteturas que estão envolvidas na interoperação e detalhes dos componentes que realizam as descrições contextuais que os gateways utilizam para transformar uma primitiva em outra. Em soluções de desenvolvimento de pilhas de protocolo, o autor da implementação tem que conhecer, detalhadamente, a *FIA* integrada e os recursos da meta-arquitetura onde ocorre a implementação. Nesse último caso, o autor da implantação não transfere as funcionalidades para um controlador, como é o caso do *FIXP*; ele implementa novamente todas as funcionalidades de controle, utilizando para isso, os princípios arquitetônicos da meta-arquitetura; desenvolvendo, dessa forma, uma nova pilha de protocolos.

***Questão 7**: Como fazer a implantação de* FIA *de maneira eficiente, ou seja, sem perda de funcionalidades?*

A inclusão de novas *FIAs* no *FIXP* não causa perda de funcionalidades da arquitetura integrada em termos de implementação, porque todos os serviços de controle da *FIA* são transferidos para o controlador, como também, todos as primitivas, serviços e protocolos são reconhecidos pelo plano de dados. Desse ponto de vista, não há perda de funcionalidades de controle. No entanto, a rede *FIXP*, nessa primeira versão, utiliza o *switch FIXP*, que estende o *target Simple Switch BMv2*. Esse *switch* possui algumas limitações. Como exemplo, ele não possui escalonador de primitivas prioritárias que assegura qualidade de serviço por tipo de comunicação. Supondo que a *FIA* possua serviços de controle de gerenciamento de qualidade da comunicação e que esses serviços sejam utilizados por determinada aplicação; o protocolo *FIXP* não oferece interfaces que gerenciam *QoS* devido às limitações do *switch* utilizado. Essa limitação está presente em todos os trabalhos correlatos que têm os mesmos objetivos arquitetônicos do *FIXP*. A linguagem *P4* permite a extensão de funcionalidades, mas a especificação do plano de dados fica dependente do *switch* onde a extensão é realizada. A evolução dos serviços que o protocolo *FIXP* oferece está diretamente relacionada à evolução dos *switches P4*. Alguns trabalhos já caminham

na direção de fornecer *QoS* a partir do plano de dados programável da rede *P4* (FERNANDES; CAMARGOS, 2020; CHEN et al., 2019; PAOLUCCI et al., 2022). Uma forma de mitigar esta situação é fazer uma extensão das funcionalidades do BMv2 para que ele ofereça filas de prioridade que visam qualidade de serviço por tipo de comunicação; e, estender as interfaces do protocolo *FIXP* para que o controlador consiga prover qualidade de serviço para suas aplicações. É importante mencionar que essa limitação de qualidade de serviço não está relacionada a uma limitação da arquitetura *FIXP*, e sim, à capacidade atual do dispositivo de comutação que a rede *FIXP* utiliza. Em relação às arquiteturas ETArch, NovaGenesis e IP implantadas na rede *FIXP*, não há perda de funcionalidades no que diz respeito às aplicações experimentadas, pois elas não utilizam controle de *QoS* em suas versões originais.

***Questão 8**: Como viabilizar a implantação da meta-arquitetura proposta no ambiente corporativo, em escala mundial?*

A implantação se dá de forma gradual. Em um primeiro momento, o cenário é complexo, pois pode existir vários domínios administrativos de *FIAs* distintas, com seus próprios *switches*, aplicações, orientações de design, paradigmas de comunicação, entre outros. Nesse cenário, o *FIXP* é um ponto de troca lógico, que consegue fazer a intercomunicação desses domínios administrativos de tal forma que aplicações da *FIA* X, que estão em diferentes domínios administrativos, comunicam-se entre si; aplicações da *FIA* Y, que estão em diferentes domínios administrativos, comunicam-se entre si; e assim por diante. A evolução desse cenário é que cada um dos domínios administrativos implantem a infraestrutura *FIXP*, gradualmente, até que alcance todos os agentes de telecomunicações. Com a maioria ou a totalidade dos domínios contendo um núcleo *FIXP*, a conectividade entre esses domínios forma uma rede única que padroniza a comunicação de serviços que utilizam diferentes arquiteturas de Internet, alcançando, dessa forma, coexistência de *FIAs* em escala global.

4.4 Considerações finais

Esta seção realiza as considerações finais do capítulo. As reflexões desta seção extrapolam o objetivo geral do trabalho, fazendo um aprofundamento de como a arquitetura *FIXP* pode ser útil na incorporação das arquiteturas de Internet do Futuro nas redes corporativas (Subseção 4.4.1) e quais as contribuições que o *FIXP* fornece para as redes da próxima geração (Subseção 4.4.2).

4.4.1 Uma breve discussão que envolve as *FIAs*, redes corporativas e o *FIXP*.

As arquiteturas de Internet do Futuro, apesar das demandas existentes nas redes atuais, não se efetivam nas redes corporativas, fundamentalmente, por três razões. A primeira é a falta de implementação dessas arquiteturas em redes de alto desempenho para testar suas características em casos de uso específicos. A segunda refere-se ao fato de que as *FIAs* são desenvolvidas de forma disjunta e possuem diferenças substanciais. Em nível de hardware, cada arquitetura possui sua própria infraestrutura física de comunicação. Em nível de software, as *FIAs* diferem-se entre si quanto à orientação de design, paradigmas de comunicação, esquemas de endereçamento, entre outros. Além dessas duas situações, encontra-se um terceiro obstáculo: não existe um protótipo em funcionamento nas redes de telecomunicações que consiga realizar a coexistência de todas essas arquiteturas de Internet em um mesmo ambiente de comunicação. A importância da coexistência vem do fato de que o investimento de implantar "n" arquiteturas de Internet com "n" infraestruturas físicas diferentes é inviável do ponto de vista financeiro e de gerenciamento da rede.

Nesse cenário, o *FIXP* estende o paradigma *SDN-P4*, e coloca-se como candidato para solucionar os problemas descritos.

Em relação ao primeiro problema, que é o problema da implantação de *FIAs* em redes de alto desempenho, já existem arquiteturas padrões que tornam os chips de função programável com desempenhos tão eficientes quanto os chips de função fixa (MCKEOWN; REXFORD, 2016). Teoricamente, as funcionalidades do *switch FIXP* podem ser implantadas em quaisquer *switches* de função programável que implementa a arquitetura *PISA*. Dessa forma, *FIXP* é um candidato de infraestrutura de rede de alta performance capaz de testar a maturidade de todas as tecnologias de arquiteturas de Internet existentes e futuras.

Em relação ao segundo problema, o *FIXP* possui duas soluções. O *switch FIXP-P4* possui grande flexibilidade em relação à programabilidade do plano de dados e, na visão deste trabalho, é o dispositivo que vai padronizar a infraestrutura física de quaisquer serviços da Internet, inclusive os serviços de *FIAs* existentes e futuras. Isso resolve o problema das diferenças de dispositivos ou infraestruturas físicas. Por outro lado, o *FIXP* possibilita que as funcionalidades de rede de cada FIA seja transferida para um controlador *SDN-FIXP*, disponibilizando uma interface que assegura o gerenciamento da rede pelo operador. Isso soluciona o problema das diferenças arquiteturais.

O terceiro problema: implantação de várias FIAS em um ambiente de comunicação; o *FIXP* fornece uma rede única, com planos de dados e controle altamente programáveis, que possibilita a coexistência dos serviços de diferentes arquiteturas de Internet. Além disso, o *FIXP* padroniza a comunicação de aplicações que utilizam arquiteturas de Internet distintas, e como dito, se implantado com *switches* de alta performance, pode ser o

protótipo que agentes de telecomunicações precisam para construir um ambiente de rede capaz de testar diferentes alternativas *clean-slate* de comunicação, aproximando, dessa forma, capacidades de rede às demandas do usuário.

4.4.2 Uma breve discussão sobre o FIXP e as infraestruturas de redes do futuro

A Subseção 2.1.9 analisa quais são as tendências tecnológicas das redes do futuro quando se verifica as necessidades das redes atuais, o avanço tecnológico das redes móveis, a entrega de redes como serviço e os processos de gerenciamento cada vez mais complexos. A presente seção trata da relação que existe entre o *FIXP* e essas redes do futuro, descrevendo alguns pontos que o *FIXP* oferece e que são essenciais para as redes da próxima geração.

O *FIXP* oferece contribuições significativas para as redes do futuro, pois possibilita a oferta de serviços de rede de acordo com as necessidades do usuário. O provisionamento desses serviços advém dos controladores implantados na infraestrutura da solução. Redes móveis atuais, como o 5G, oferecem serviços de rede lógicos que herdam as limitações da arquitetura TCP/IP. Se por um lado, a virtualização de rede das arquiteturas atuais (como 5G) flexibiliza a inclusão de serviços de rede, toda a infraestrutura de comunicação ainda funciona em conjunto com a arquitetura legada. O *FIXP* proporciona a entrega de serviços, possibilitando que a aplicação escolha inclusive a arquitetura de Internet, não herdando, dessa forma, limitações de capacidades que custam caro para as operadoras, tais como multicasting de serviços de *streaming* de vídeo. O custo caro também advém da necessidade da implantação de novas infraestruturas físicas para cada requisito novo implantado. Por exemplo, tecnologias bastante utilizadas no mundo corporativo, tais como *Multi Protocol Label Switching (MPLS)*, *Virtual Private LAN Service (VPLS)*, *Virtual Extensible LAN (VXLAN)*, *Virtual Local Area Network (VLAN)* e *Virtual private network (VPN)* dependem de infraestruturas físicas diferentes que suportam seus serviços. O *FIXP*, além de oferecer uma rede que tem a capacidade de oferecer diversas arquiteturas de Internet; todas as funcionalidades dos serviços de rede dessas novas arquiteturas são executadas dentro de uma infraestrutura de rede única que as suporta. Isso é possível porque o FIXP possui planos de dados e controle programáveis, extensíveis e configuráveis. Se a infraestrutura FIXP precisa de novos serviços, ela não troca a infraestrutura física (a não ser por escalabilidade), ela adiciona o serviço (software) na infraestrutura. A entrega dessa infraestrutura programável pela solução *FIXP* é o primeiro grande benefício que a solução oferece para as redes do futuro. O segundo é que o protocolo *FIXP* abstrai a complexidade dessa rede subjacente e padroniza a comunicação dessa infraestrutura, independentemente dos fabricantes dos dispositivos de hardware que a topologia comporta, otimizando, assim, o gerenciamento da rede.

Espera-se que o oferecimento de uma infraestrutura única, a padronização de comunicação dessa infraestrutura e o provisionamento de diversas arquiteturas de Internet possam entregar uma rede como serviço, no futuro, mais completa do que as redes atuais oferecem.

Dois outros assuntos foram bastante discutidos na Subseção 2.1.9. O primeiro é a aproximação da camada de aplicação dos serviços de rede que a operadora oferece. O segundo é o gerenciamento das redes futuras.

Quanto ao primeiro tema, a aproximação de serviços de rede da aplicação do usuário é uma consequência direta da infraestrutura FIXP. Diferentemente do projeto Camara(ver Subseção 2.1.9.4), que aproxima os usuários dos serviços de rede através da camada de aplicação por conta das limitações da Internet, as FIAs que participam da rede FIXP oferecem serviços de rede diretamente para a aplicação através dos seus protocolos (no caso da ETArch, protocolos que atuam no nível de rede). Esse fato é singelo, porém importante. No caso da ETArch, FIA integrada ao FIXP, a complexidade do serviço oferecido está no nivel de rede e não na camada de aplicação. Ou seja, a ETArch é capaz de oferecer um canal de comunicação multicast com qualidade de serviço de forma transparente, pois a gerência da comunicação é realizada pela própria rede. Já no projeto Camara, ele aproxima serviços de rede das aplicações, mas quem gerencia o serviço é a aplicação. Ou seja, a complexidade do gerenciamento fica a cargo da aplicação, portanto, o serviço não é oferecido de forma transparente. Essa característica do *FIXP*, de incluir novos serviços na camada de rede, pode melhorar a concepção atual de fornecimento de serviços de rede por meio de APIs, fazendo com que o oferecimento, por exemplo, de *QoE*, para uma aplicação, seja gerenciado pela própria rede de telecomunicações, facilitando, dessa forma, o desenvolvimento de aplicativos atuais e futuros.

Em relação ao segundo tema, gerenciamento, já existem muitos trabalhos na literatura que tratam o gerenciamento e a auto-organização da rede em ambientes SDN (WANG; YAN, 2016; CANINI et al., 2017; RAMIREZ-PEREZ; RAMOS, 2016). Controladores SDN logicamente centralizados tem que implementar mecanismos de computação autônoma e autogerenciamento na infraestrutura FIXP. Esses controladores têm que ser padronizados e distribuídos na infraestrutura FIXP, de tal forma que a privacidade dos dados de gerenciamento das operadoras sejam preservados.

Uma das soluções futuras do protocolo FIXP e que está diretamente envolvida com as redes do futuro é o oferecimento de interfaces que ofereçam métricas relacionadas à qualidade de serviço, tais como taxa de transferência, latência, perda de pacotes e outros. Para isso, os switches FIXP têm que oferecer esses serviços de qualidade para que o protocolo FIXP estenda essas funcionalidades, abstraia a complexidade da camada subjacente e ofereça-as para os controladores implantados.

O *FIXP* está em fase inicial de implementação e os assuntos de gerenciamento, autonomia e qualidade de serviço ainda não foram abordados. É válido dizer que o protocolo

FIXP tem a única função de padronizar a comunicação entre a infraestrutura física FIXP e seus controladores, enquanto a infraestrutura FIXP tem a função mais abrangente de padronizar um ambiente de comunicação onde coexistam diferentes FIAs. Qualquer outro recurso que deva ser implantado na rede FIXP e que não tem relação com os objetivos do FIXP, vai ser integrado ao FIXP através de um controlador SDN-FIXP. como são os dois casos que vimos anteriormente: gerenciamento e fornecimento de serviços de APIs às aplicações pelas operadoras.

Capítulo 5

Conclusão

Este capítulo apresenta, de forma pontual, as principais realizações desta tese. A Seção 5.1 descreve brevemente as realizações concretas deste trabalho e a relação de cada uma delas com o objetivo geral (ver a Seção 1.3). A seção 5.2 relata os artigos publicados e submetidos à revisão, produzidos ao longo desta pesquisa. Finalmente, a Seção 5.3 discute os próximos passos para a evolução da arquitetura e da infraestrutura FIXP, em termos de princípios conceituais e de procedimentos operacionais.

5.1 Considerações finais

Esta tese estende as discussões teóricas sobre coexistência de FIAs em redes de telecomunicações, trazendo diversos trabalhos que estão envolvidos diretamente ou indiretamente com o assunto; mapeando os mecanismos que cada trabalho utiliza e os efeitos colaterais que essas decisões de projeto causam na comunicação (ver Tabela 2). A investigação da fundamentação teórica e dos trabalhos relacionados identifica alguns problemas que justificam o objetivo geral deste trabalho, quais sejam: falta de efetivação das *FIAs* nas redes corporativas, apesar da demanda; e, efeitos colaterais negativos na comunicação provocados por mecanismos/procedimentos operacionais das soluções que têm o mesmo objetivo arquitetônico do *FIXP* (ver Subseção 2.2.6). Alguns desses efeitos colaterais são: processos que podem sobrecarregar os elementos de rede e dificultar a implementação dos princípios arquitetônicos dessas soluções (meta-arquiteturas) nas redes atuais.

A compilação dos problemas existentes no estado da arte é a base do objetivo geral do trabalho, que é propor, desenvolver, analisar e avaliar uma arquitetura, cujos princípios arquitetônicos não possuem o engessamento das redes atuais e possibilitam a incorporação de serviços de rede/de transporte/de enlace que são *clean-slate* (REXFORD; DOVROLIS, 2010), evolucionários (REXFORD; DOVROLIS, 2010) ou legados, permitindo, dessa forma, a coexistência de arquiteturas de Internet distintas. O intuito desse modelo conceitual é melhorar a comunicação dos aplicativos recentes e futuros, já que os mesmos podem escolher, nessa rede, as capacidades de arquiteturas de Internet que melhor atendam aos

seus requisitos; como também permitir que *FIAs* possam ser incorporadas, gradualmente, nas redes corporativas, em escala global. O objetivo geral do trabalho (ver Seção 1.3) também descreve a necessidade de solucionar os problemas de comunicação mencionados, presentes nos trabalhos correlatos. Para isso, alguns requisitos são importantes: o processo de incorporação de novas *FIAs* no *FIXP* não deve ter um nível alto de dificuldade em relação aos trabalhos correlatos; deve haver flexibilidade para incorporação de novos serviços; não deve haver adicionamento de processos nos dispositivos de comutação que podem onerar negativamente a comunicação entre as aplicações; a implementação e implantação dos princípios arquitetônicos nas redes atuais deve ser eficiente, afinal de contas, se operadores de rede não conseguem instanciar uma meta-arquitetura, ela não tem utilidade prática; e, o projeto tem que levar em conta a implementação/implantação gradual das *FIAs* no *FIXP* e do *FIXP* nas redes atuais até que a comunicação alcance escala global.

O capítulo 3 define os princípios arquitetônicos de forma conceitual e lógica. Em relação aos conceitos, são descritos os objetivos de cada bloco funcional da arquitetura; quanto à lógica, são descritos a sequência de execução de cada bloco funcional e a comunicação interna entre eles. Também são descritos os procedimentos para implementar uma nova *FIA* no *FIXP* e a implantação da infraestrutura *FIXP* nas redes atuais até o atingimento de uma comunicação global. Em seguida, é apresentado o protocolo *FIXP*, suas primitivas e interfaces, que permitem a comunicação do controlador com os *switches* que compõem o plano de dados. Além disso, o capítulo mostra como se dá a integração das arquiteturas NovaGenesis, ETArch e IP na rede *FIXP* e a troca de mensagens existentes entre cada uma dessas arquiteturas e a rede. Essas descrições detalhadas da arquitetura, posteriormente, permitem posicionar o *FIXP* no estado da arte como um ponto de troca lógico que ISPs, IXPs, POPs e outras entidades de telecomunicações podem implantar gradualmente, transformando-o em uma rede única de comunicação, na qual podem coexistir diversas *FIAs* simultaneamente. Diferentemente de trabalhos relacionados (GUIMARãES et al., 2019; MACHADO; DOUCETTE; BYERS, 2015), a rede *FIXP* não sobrecarrega seus dispositivos de comutação (processamento, memória e armazenamento) com processos de interoperação e/ou a implantação de um número excessivo de serviços. Os princípios arquitetônicos do FIXP descritos no capítulo garantem que qualquer serviço de controle ou gerenciamento seja implantado em uma peça de software externa, o controlador-FIXP. O capítulo 3 é responsável por atender a grande parte do objetivo geral do trabalho, pois propõe princípios arquitetônicos capazes de sustentar *FIAs* distintas. Além disso, o modelo de arquitetura apresentado tem a finalidade de enfrentar os problemas existentes nos trabalhos correlatos, tais como ineficiência (em alguns casos) de implementação e implantação de novas FIAs nas meta-arquiteturas correspondentes e sobrecarregamento dos dispositivos de comutação devido às decisões de projeto das soluções, que incorporam procedimentos adicionais no núcleo da rede; tais como interope-

ração e pilhas de protocolos. A avaliação que verifica se esses objetivos foram cumpridos é realizada no capítulo 4.

O capítulo 4 apresenta as avaliações qualitativas e quantitativas da rede *FIXP*. As avaliações qualitativas (ver Subseções 4.2.1, 4.2.2 e 4.2.3) demonstram a relevância dos princípios arquitetônicos do *FIXP* em relação à pilha TCP/IP, às redes futuras e aos trabalhos relacionados; e, a eficiência de implementação e implantação desses princípios nas redes atuais. O *FIXP* mostrou-se flexível para a inserção de novos serviços na rede e para a implantação de novas *FIAs*, com nível de dificuldade baixo em relação aos trabalhos correlatos. Além disso, a implantação do *FIXP* em escala global requer apenas um núcleo de *switches FIXP* em cada agente de serviço de telecomunicações. Cada um desses núcleos é capaz de suportar as funcionalidades de diversas *FIAs*, respeitando, claro, os recursos computacionais da rede. Cada *FIA* é suportada pelo seu próprio controlador *SDN-FIXP*, que possui todas as suas funcionalidades de controle. O plano de dados mostrou-se programável, configurável e extensível, possibilitando dessa forma, o reconhecimento de quaisquer protocolos, primitivas ou serviços; e, a especificação de funcionamento do pipeline de dados do switch FIXP. A ideia geral de implantação é que as redes locais ou pontos lógicos locais do *FIXP* conectem-se entre si, formando uma malha ou rede única que padroniza a comunicação, suportando, dessa forma, *FIAs* distintas e as suas aplicações em escala global. Quanto às avaliações quantitativas (ver Subseção 4.2.4), foram realizados 22 casos de uso. Para cada caso de uso, fez-se 10 experimentos, totalizando 220 experimentos e mais de 5.000.000 de registros analisados. O objetivo é demonstrar conceitualmente os princípios arquitetônicos do *FIXP* e avaliar quantitativamente, por meio de cerca de 10 métricas, o *FIXP* em dois aspectos: sobrecarregamento da infraestrutura *FIXP* e desempenho das aplicações que participam da comunicação, considerando a qualidade de experiência do usuário (*QoE*). A análise e avaliação das medições propostas também permitiram formular conclusões sobre o comportamento dos *switches FIXP*, em estado de estresse, no processamento de tráfego de naturezas distintas (vídeo e texto) gerado pelas aplicações envolvidas na comunicação. Parte-se do pressuposto de que um *switch FIXP* (hardware) teria o mesmo comportamento do switch FIXP (software) utilizado nos experimentos, pois ambos possuiriam a mesma lógica de processamento de primitivas. Uma das diferenças entre esses dois componentes seria a taxa de linha do dispositivo de hardware, que é mais eficiente. A maioria dos casos de uso apresenta uma boa experiência do usuário, como, por exemplo, os casos 1 a 13 e 14, 16, 19, 20. Em outros, como o Caso 17, o *streaming* de vídeo possui uma taxa de perda considerável, e portanto, nesse caso, o serviço de *streaming* não performa qualidade na transmissão do vídeo, e por conseguinte, não proporciona uma boa experiência do usuário. Outro resultado importante é que nenhum caso de uso sobrecarrega os recursos computacionais da máquina virtual que representa os dispositivos de comutação. Os máximos de memória e de processamento utilizados, considerando os *outliers*, são 158,46 *MiB* e 31,38% de um processador virtual

de 2,80 GHz, respectivamente. Este resultado demonstra que, apesar de alguns casos de uso sobrecarregarem o switch FIXP devido às suas limitações tecnológicas, a máquina virtual que representa o switch físico da topologia não apresentou sobrecarregamento em relação à utilização de memória e de processador, o que revela um dos objetivos arquitetônicos do FIXP: não adicionar procedimentos que onerem a rede subjacente. Outro ponto importante é que o FIXP também não onera os dispositivos de comutação quanto ao armazenamento: para suportar 3 FIAs distintas, o FIXP produz um documento P4 em formato *json* com tamanho de apenas 65 *KiB*. Outras discussões qualitativas a respeito das métricas coletadas são realizadas, detalhadamente, na Subseção 4.2.4. Uma ressalva é que os experimentos são realizados no switch BmV2 (LANGUAGE, 2024), um *switch* de software, que possui limitações que não existem em hardwares de comutação. Como exemplo, a capacidade de processamento do Bmv2 é muito menor do que a dos *switches* de hardware reais(KAUR; KUMAR; AGGARWAL, 2021). Esse capítulo conclui o objetivo geral da tese, pois cumpre o desenvolvimento e a análise qualitativa dos princípios arquitetônicos discutidos no capítulo 3, bem como realiza a prova conceitual e a medição do sobrecarregamento da infraestrutura *FIXP*. Além disso, esse capítulo, após as avaliações propostas, segue respondendo, na Seção 4.3, aos desafios de pesquisa da tese (ver Seção 1.6). No final do capítulo, ainda são apresentadas duas reflexões que extrapolam os objetivos da tese: a contribuição dos princípios arquitetônicos do FIXP para as redes corporativas (ver Subseção 4.4.1) e para as tendências das redes da próxima geração (ver Subseção 4.4.2).

Quanto aos objetivos específicos (ver Seção 1.4), a Seção 1.8.2 estabelece o mapeamento entre as seções e subseções desta tese e o cumprimento de cada um dos objetivos descritos.

Em relação à hipótese do trabalho, verifica-se nos capítulos 3 e 4. Primeiramente, no Capítulo 3, é proposta uma arquitetura *SDN* que possui planos de dados e de controle programáveis (premissa ou antecedente da hipótese). No capítulo 4, demonstra-se, a partir de avaliações qualitativas e quantitativas, que a arquitetura FIXP proposta viabiliza a coexistência de múltiplas *FIAs* sem exceder os limites operacionais de processamento e de memória dos dispositivos de comutação atuais, mantendo uma complexidade de implementação inferior à de soluções que utilizam mecanismos de interoperabilidade e/ou pilhas de protocolos (consequente da hipótese). Através das avaliações quantitativas e qualitativas apresentadas nos capítulos citados; e das provas de conceito elaboradas através da execução de casos de uso selecionados, fica demonstrada a veracidade do consequente.

5.2 Contribuições em produções bibliográficas

Ao longo deste trabalho, o *FIXP* evoluiu gradualmente tanto no modelo arquitetônico quanto na infraestrutura. Como exemplo de evolução conceitual, cita-se o seguinte: a

primeira versão do *FIXP* é um ponto de troca capaz de interligar domínios distintos de *FIAs*. Nos dias atuais, é um ponto de troca lógico que pode ser implantado em qualquer agente de telecomunicações, de modo que a conectividade entre essas redes locais ou pontos de troca locais produz uma rede única que padroniza a comunicação em escala mundial. As redes ou pontos de troca locais permitem a incorporação gradual das arquiteturas de Internet (que possuem seus próprios domínios administrativos), até que a comunicação atinja uma padronização tanto em hardware quanto em software.

A evolução do *FIXP* está registrada nas seguintes publicações bibliográficas.

1. Gavazza, J.A.T. et al. (2020). Future Internet Exchange Point (FIXP): Enabling Future Internet Architectures Interconnection. In: Barolli, L., Amato, F., Moscato, F., Enokido, T., Takizawa, M. (eds) Advanced Information Networking and Applications. AINA 2020. Advances in Intelligent Systems and Computing, vol 1151. Springer, Cham. https://doi.org/10.1007/978-3-030-44041-1_62 - Publicado: 28.03.2020 - Qualis: A3

2. SILVA, Thiago Bueno da; GAVAZZA, José A. T.; VERDI, Fábio L.; SURUAGY, José A.; COELHO, Juliano; SILVA, Flávio; ALBERTI, Antônio M.. Plano de Controle da Arquitetura NovaGenesis para um Ponto de Interconexão de Tráfego Multi-Arquitetura. In: WORKSHOP DE PESQUISA EXPERIMENTAL DA INTERNET DO FUTURO (WPEIF), 12. , 2021, Uberlândia. Anais [...]. Porto Alegre: Sociedade Brasileira de Computação, 2021 . p. 31-36. ISSN 2595-2692. DOI: https://doi.org/10.5753/wpeif.2021.17197. - Publicado: 16.08.2021 - Qualis: B4

3. T. B. da Silva, F. L. Verdi, J. C. G. de Melo, J. A. Suruagy, F. Silva and A. M. Alberti, "Toward Next-Generation and Service-Defined Networks: A NovaGenesis Control Agent for Future Internet Exchange Point,"in IEEE Network, vol. 36, no. 3, pp. 74-81, May/June 2022, doi: 10.1109/MNET.008.2100555. keywords: Network slicing;Next generation networking;Process control;Programming;Market research;Hardware;Internet;Virtualization;Autonomous systems, - Publicado: 13.07.2022 - Qualis: A1

Outro *paper* sobre o FIXP, que apresenta vários resultados expostos nesta tese que ainda não haviam sido publicados em trabalhos anteriores, cujo título é "*Toward Next-Generation Networks: The FIXP Paradigm for Integrating Future Internet Architectures*", foi submetido recentemente. O veículo escolhido para uma eventual publicação é o "*Journal of Network and Computer Applications*", ISSN 1084-8045, está classificado no Qualis como A1.

Além desses artigos, há a publicação de um *paper* que propõe o protocolo autônomo *Conform*, que efetua inicialização automática (*bootstrapping*) do plano de controle de redes

SDN. Esse serviço não depende de outros protocolos tais como o *Link Layer Discovery Protocol (LLDP)* e possui duas realizações principais: estabelece automaticamente canais de comunicação *in-band* entre controlador e *switches SDN*; e, oferece a visão da topologia de rede para o controlador.

Esse *paper* é incluído em uma lista separada, pois trata-se de um tema indiretamente relacionado ao *FIXP*. As funcionalidades do protocolo *Conform* podem ser integradas, com algumas modificações, aos serviços oferecidos pelo protocolo *FIXP*, possibilitando, assim, o estabelecimento autônomo dos canais de controle e a descoberta automática da topologia subjacente.

4. M. Silva Freitas, R. Oliveira, D. Molinos, J. Melo, P. Frosi Rosa and F. de Oliveira Silva, "ConForm: In-band Control Plane Formation Protocol to SDN-Based Networks,"2020 International Conference on Information Networking (ICOIN), Barcelona, Spain, 2020, pp. 574-579, doi: 10.1109/ICOIN48656.2020.9016580. keywords: Protocols;Optical switches;Network topology;Topology;Security;TCPIP;Bootstrap;In-Band Control Plane;Clean Slate Architectures;Protocol Design;Self-establishment. - Publicado: 02.03.2020 - Qualis: A3

5.3 Trabalhos Futuros

Este trabalho apresenta a especificação e a implementação inicial da arquitetura *FIXP*; e por conta disso, há uma série de questões não tratadas. O gerenciamento da rede *SDN* possui alta complexidade, porque além dos elementos que já existem nas redes tradicionais, há a incorporação de outros componentes, como, por exemplo, controladores *SDN*, protocolos *SDN*, aplicações de gerenciamento , entre outros.

Apesar das vantagens nítidas desse paradigma, alguns problemas continuam em aberto e precisam ser analisados e as soluções incorporadas, posteriormente, na rede *FIXP*. Alguns exemplos podem ser citados: ausência de procedimentos de autocura e possível inconsistência entre os planos de dados e controle. Quanto ao primeiro problema, na arquitetura tradicional, os protocolos de rede podem alterar rotas quando ocorre uma falha na infraestrutura física da rede (switches e enlaces). Em *SDN*, não há protocolos específicos que garantam alta disponibilidade dos links ou a detecção e a correção de falhas na rede (CHANDRASEKARAN; TSCHAEN; BENSON, 2016). Quanto ao segundo problema, pode-se citar que a separação dos planos de dados e controle pode causar inconsistências de dados no controlador, podendo originar, por exemplo, *loops* de roteamento ((ABDELSALAM, 2018)).

Existe um conjunto de problemas vinculado ao SDN, diretamente ligado ao *FIXP* e, como tal, futuros trabalhos poderiam realizar um levantamento dessas questões. Grande parte desses problemas, como alguns citados, tem relação com redes autônomas. Nessa

linha, algumas ações poderiam ser adotadas. As questões tratadas abaixo (itens 1 a 5) são trabalhos que necessitam de um planejamento de longo prazo, pois envolvem o desenvolvimento de arquiteturas *SON* complexas, que preveem vários contextos de automação em redes *SDN*. Essas arquiteturas são alheias à arquitetura FIXP, não estão envolvidas diretamente com a arquitetura FIXP. A ideia é que essas infraestruturas de solução de rede, autônomas e independentes, sejam absorvidas pela infraestrutura FIXP (como se fosse um FIA), transformando-a em um sistema autônomo e autogerenciável.

1. Desenvolver serviços que promovam autocura (self-healing) na rede *FIXP*, por meio de técnicas de monitoramento em tempo real, de análise preditiva, de recuperação autônoma e de aprendizado contínuo.

2. Desenvolver serviços que promovam autoconfiguração (*self-configuration*), de tal modo que a rede *FIXP* possa, por exemplo, atualizar automaticamente a especificação de funcionamento do pipeline de dados dos *switches* no plano de dados; quando acontecer, por exemplo, a evolução de algum protocolo ou serviço.

3. Desenvolver/planejar mecanismos de autoproteção (*self-protection*) que detectem e reajam a ataques no plano de dados ou no plano de controle do *FIXP*.

4. Desenvolver serviços de auto-otimização (*self-optimizing*) para que a rede possa se configurar visando a otimização da comunicação em diferentes contextos.

5. Desenvolver serviços de *bootstrapping*, que inicializem a rede quando os *switches FIXP* ficarem ativos. Esses serviços têm de implementar a descoberta dos *switches FIXP* da rede subjacente e passar a visão global atualizada da topologia e do estado da infraestrutura física da rede, periodicamente, para os controladores que participam da comunicação.

Outras possibilidades de trabalhos futuros estão diretamente vinculadas à arquitetura ou à infraestrutura do FIXP. Esses trabalhos podem ser classificados em duas categorias: (i) aqueles que já estão em desenvolvimento ou que vão desenvolver uma funcionalidade que o protocolo FIXP já suporta, porém que não foi demonstrada por meio de experimentos; (ii) aqueles cujas funcionalidades ainda precisam ser desenvolvidas.

Em relação à primeira categoria (i), esses trabalhos exigem um planejamento de curto prazo, pois já estão em um estágio avançado de implementação; ou, se necessário, apenas o desenvolvimento e as análises experimentais de um serviço que o protocolo FIXP já suporta.

Seguem, abaixo, algumas sugestões de trabalhos futuros que pertencem à categoria (i).

6. Criar um ambiente de experimentação baseado em targets (hardware) de comutação *SDN-P4*, como, por exemplo, o *target TOFINO* da Intel (INTEL, 2021).

7. Desenvolver parcialmente uma virtualização de redes, criando instâncias lógicas de software que representam funcionalidades virtuais de comutação em um switch físico FIXP (atualmente representado por uma máquina virtual), de modo que essas instâncias lógicas possam materializar-se em redes virtuais com características de isolamento de serviços de rede (plano de controle) e de pipeline de dados (plano de dados). Dessa forma, cada FIA teria sua própria rede virtual. Esse é um passo inicial para a virtualização de rede da infraestrutura FIXP. Esse trabalho poderia verificar os benefícios dessa virtualização para eventuais atualizações de equipamentos de rede (por exemplo, do pipeline de dados) em produção e os avanços que essa virtualização traria para os consumidores de recursos da infraestrutura FIXP.

8. Implantar o *FIXP* em domínios administrativos distintos, emulando conectividade entre os pontos de troca lógicos locais até que se alcance a comunicação em escala global. Para que esse experimento seja possível, a FIA implantada deve ser distribuída, ou seja, conter interfaces *East/WestBound* que permitam a comunicação entre seus controladores em diferentes domínios administrativos.

9. Realizar a implantação da XIA no FIXP. Esse projeto consolidaria ainda mais o FIXP como uma meta-arquitetura madura, capaz de absorver, inclusive, outras meta-arquiteturas, como, por exemplo, a XIA. Este projeto é colocado como sendo de planejamento de curto prazo por vários motivos: a arquitetura FIXP já está materializada e avaliada por este trabalho; e, a arquitetura XIA já está parcialmente implantada pelo FIXP (SILVA, 2020). Os switches FIXP já reconhecem os protocolos e serviços do XIA, porém ainda resta o desenvolvimento do plano de controle para uma próxima publicação.

Em relação à segunda categoria (ii), esses trabalhos exigem um planejamento de longo prazo, basicamente por dois motivos: as funcionalidades ainda não existem na infraestrutura FIXP; e essas funcionalidades demandam mais tempo de estudo, análise, desenvolvimento e avaliação, por se tratar de soluções sofisticadas que o FIXP ainda não suporta.

Seguem, abaixo, algumas sugestões de trabalhos futuros que pertencem à categoria (ii).

10. O próximo passo para os itens 6 e 7 é complementar a virtualização das redes da infraestrutura FIXP. Se este recurso já estiver implantado em um *switch* de hardware como o *TOFINO* (item 6) (INTEL, 2021) e se as instâncias lógicas de *switches FIXP* já estão implantadas dentro do hardware (item 7), o próximo passo é acrescentar à solução a utilização de tecnologias tais como *NFV*, *VLAN* e *VXLAN*, completando dessa forma a virtualização de rede da infraestrutura FIXP, que através do desenvolvimento do item corrente, passaria a oferecer redes lógicas fim a fim. Para complementar, o protocolo FIXP também pode estender os recursos de *QoS*

do TOFINO (INTEL, 2021) e oferecê-los aos controladores. Além disso, o FIXP pode controlar esses recursos dinamicamente, em tempo de execução, conforme as solicitações dos controladores que integram a rede. Como essas FIAs aproximam seus recursos de rede das aplicações do usuário, em teoria, essas aplicações poderiam controlar a qualidade de suas próprias comunicações.

11. Desenvolver serviços no protocolo *FIXP* que possibilitam a formação de um *cluster* de camadas de abstração distribuídas no domínio, permitindo, assim, a escalabilidade horizontal dessas camadas em situações de sobrecarregamento, causadas, por exemplo, por um grande número de *switches* e controladores *FIXP* integrados à rede. Para que esse experimento seja possível, a FIA implantada deve ser distribuída, ou seja, conter interfaces *East/WestBound* que permitam a comunicação entre seus controladores em diferentes camadas de abstração (domínios internos de um agente de telecomunicações).

12. Realizar a implantação de outras arquiteturas de Internet do Futuro, promovendo, cada vez mais, a consolidação do *FIXP* como meta-arquitetura e incentivando outros desenvolvedores a testar suas *FIAs* em uma rede que possibilita a coexistência simultânea de vários serviços *clean-slate*.

Referências

3GPP, G. *5G System Overview.* 2022. Disponível em: <https://www.3gpp.org/technologies/5g-system-overview>. Acesso em: 30 jul. 2025.

3GPP, G. *3GPP CAPIF Framework - RNAA*. 2023. Disponível em: <https://www.3gpp.org/technologies/rnaa>. Acesso em: 30 jul. 2025.

ABDELSALAM, M. A. **Network Application Design Challenges and Solutions in SDN**. Tese (Doutorado) — Carleton University, 2018. Disponível em: <https://doi.org/10.1145/2890955.2890965>.

AFOLABI, I. et al. Network slicing and softwarization: A survey on principles, enabling technologies, and solutions. **IEEE Communications Surveys & Tutorials**, v. 20, n. 3, p. 2429–2453, 2018. Disponível em: <https://doi.org/10.1109/COMST.2018.2815638>.

AGENCY, D. D. A. R. P. **Internet Protocol - Specification**. USA: RFC Editor, 1981. Disponível em: <https://datatracker.ietf.org/doc/html/rfc791>.

AHMADI, S. **5G NR: Architecture, technology, implementation, and operation of 3GPP new radio standards**. [S.l.]: Academic Press, 2019.

ALBERTI, A. M. et al. Introducing novagenesis as a novel distributed system-based convergent information architecture. In: **Nature-Inspired Networking**. [S.l.]: CRC Press, 2018. p. 89–144.

_____. Advancing novagenesis architecture towards future internet of things. **IEEE Internet of Things Journal**, v. 6, n. 1, p. 215–229, 2019. Disponível em: <https://doi.org/10.1109/JIOT.2017.2723953>.

ANAND, A. et al. Xia: an architecture for an evolvable and trustworthy internet. In: **Proceedings of the 10th ACM Workshop on Hot Topics in Networks**. New York, NY, USA: Association for Computing Machinery, 2011. (HotNets-X). ISBN 9781450310598. Disponível em: <https://doi.org/10.1145/2070562.2070564>.

BAKTIR, A. C.; OZGOVDE, A.; ERSOY, C. Implementing service-centric model with p4: A fully-programmable approach. In: **NOMS 2018 - 2018 IEEE/IFIP Network Operations and Management Symposium**. [s.n.], 2018. p. 1–6. Disponível em: <https://doi.org/10.1109/NOMS.2018.8406282>.

BARAKABITZE, A. A. et al. 5g network slicing using sdn and nfv: A survey of taxonomy, architectures and future challenges. **Computer Networks**, v. 167, p. 106984, 2020. ISSN 1389-1286. Disponível em: <https://doi.org/10.1016/j.comnet.2019.106984>.

BARAN, P. On distributed communications networks. **IEEE Transactions on Communications Systems**, v. 12, n. 1, p. 1–9, 1964. Disponível em: <https://doi.org/10.1109/TCOM.1964.1088883>.

BIERMAN, A.; BJöRKLUND, M.; WATSEN, K. **RESTCONF Protocol**. RFC Editor, 2017. RFC 8040. (Request for Comments, 8040). Disponível em: <https://doi.org/10.17487/RFC8040>.

BLEULER, S. et al. Pisa — a platform and programming language independent interface for search algorithms. In: FONSECA, C. M. et al. (Ed.). **Evolutionary Multi-Criterion Optimization**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003. p. 494–508. ISBN 978-3-540-36970-7. Disponível em: <https://doi.org/10.1007/3-540-36970-8_35>.

BODDAPATI, G. et al. Assessing the security of a clean-slate internet architecture. In: **2012 20th IEEE International Conference on Network Protocols (ICNP)**. [s.n.], 2012. p. 1–6. Disponível em: <https://doi.org/10.1109/ICNP.2012.6459947>.

BOUBENDIR, A.; BERTIN, E.; SIMONI, N. Naas architecture through sdn-enabled nfv: Network openness towards web communication service providers. In: **NOMS 2016 - 2016 IEEE/IFIP Network Operations and Management Symposium**. [s.n.], 2016. p. 722–726. Disponível em: <https://doi.org/10.1109/NOMS.2016.7502885>.

BRADNER, S. O.; MANKIN, A. J. **The Recommendation for the IP Next Generation Protocol**. RFC Editor, 1995. RFC 1752. (Request for Comments, 1752). Disponível em: <https://doi.org/10.17487/RFC1752>.

CANINI, M. et al. A self-organizing distributed and in-band sdn control plane. In: **2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)**. [s.n.], 2017. p. 2656–2657. Disponível em: <https://doi.org/10.1109/ICDCS.2017.328>.

CASCONE, C. **Using P4 and Programmable Switches to Implement a 4G/5G UPF in Aether**. 2021. Disponível em: <https://opennetworking.org/news-and-events/blog/using-p4-and-programmable-switches-to-implement-a-4g-5g-upf-in-aether/>. Acesso em: 27 apr. 2025.

CASE, D. J. D. et al. **Protocol Operations for version 2 of the Simple Network Management Protocol (SNMPv2)**. RFC Editor, 1993. RFC 1448. (Request for Comments, 1448). Disponível em: <https://www.rfc-editor.org/info/rfc1448>.

CHANDRASEKARAN, B.; TSCHAEN, B.; BENSON, T. Isolating and tolerating sdn application failures with legosdn. In: **Proceedings of the Symposium on SDN Research**. New York, NY, USA: Association for Computing Machinery, 2016. (SOSR '16). ISBN 9781450342117. Disponível em: <https://doi.org/10.1145/2890955.2890965>.

CHEN, Y.; FARLEY, T.; YE, N. Qos requirements of network applications on the internet. **Information Knowledge Systems Management**, v. 4, n. 1, p. 55–76, 2004. Disponível em: <https://doi.org/10.3233/IKS-2004-00061>.

CHEN, Y.-W. et al. P4-enabled bandwidth management. In: **2019 20th Asia-Pacific Network Operations and Management Symposium (APNOMS)**. [s.n.], 2019. p. 1–5. Disponível em: <https://doi.org/10.23919/APNOMS.2019.8892909>.

CISCO. **Cisco Visual Networking Index (VNI) Complete Forecast Update, 2017–2022**. 2018. Disponível em: <https://www.cisco.com/c/dam/m/en_us/solutions/service-provider/vni-forecast-highlights/pdf/Global_Device_Growth_Traffic_Profiles.pdf>. Acesso em: 08 dez. 2024.

_____. **CISCO. Cisco Annual Internet Report (2018–2023) White Paper**. 2020. Disponível em: <https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html>. Acesso em: 08 dez. 2024.

COMMISSION, C. E. **Sixth framework programme of the European Community for research, technological development and demonstration activities, contributing to the creation of the European Research Area and to innovation (2002 to 2006)**. 2002. Disponível em: <https://cordis.europa.eu/programme/id/FP6>. Acesso em: 08 dez. 2024.

_____. **NSF 13-538: Future Internet Architectures – Next Phase (FIA-NP)**. 2007. Disponível em: <https://cordis.europa.eu/programme/id/FP7>. Acesso em: 08 dez. 2024.

_____. **NSF 13-538: Future Internet Architectures – Next Phase (FIA-NP)**. 2014. Disponível em: <https://cordis.europa.eu/programme/id/H2020>. Acesso em: 08 dez. 2024.

COX, J. H. et al. Advancing software-defined networks: A survey. **IEEE Access**, v. 5, p. 25487–25526, 2017. Disponível em: <https://doi.org/10.1109/ACCESS.2017.2762291>.

CROWCROFT, J. et al. Plutarch: an argument for network pluralism. In: **Proceedings of the ACM SIGCOMM Workshop on Future Directions in Network Architecture**. New York, NY, USA: Association for Computing Machinery, 2003. (FDNA ́03), p. 258–266. ISBN 1581137486. Disponível em: <https://doi.org/10.1145/944759.944763>.

CZYZ, J. et al. Measuring ipv6 adoption. In: **Proceedings of the 2014 ACM Conference on SIGCOMM**. New York, NY, USA: Association for Computing Machinery, 2014. (SIGCOMM '14), p. 87–98. ISBN 9781450328364. Disponível em: <https://doi.org/10.1145/2619239.2626295>.

DE, P. H. A. D. **Mecanismos de autenticação e controle de acesso para uma arquitetura de Internet do Futuro**. 2017. Disponível em: <https://repositorio.ufu.br/handle/123456789/18381>. Acesso em: 27 apr. 2025.

DEERING, S. E.; CHERITON, D. R. Multicast routing in datagram internetworks and extended lans. **ACM Trans. Comput. Syst.**, Association for Computing Machinery, New York, NY, USA, v. 8, n. 2, p. 85–110, maio 1990. ISSN 0734-2071. Disponível em: <https://doi.org/10.1145/78952.78953>.

DING, J. **Advances in network management**. Auerbach Publications, 2016. Disponível em: <https://doi.org/10.1201/9781420064551>.

DUAN, Q.; YAN, Y.; VASILAKOS, A. V. A survey on service-oriented network virtualization toward convergence of networking and cloud computing. **IEEE Transactions on Network and Service Management**, v. 9, n. 4, p. 373–392, 2012. Disponível em: <https://doi.org/10.1109/TNSM.2012.113012.120310>.

ELECTRICAL, I. of; ENGINEERS, I. E. **IEEE at a Glance**. 2025. Disponível em: <https://www.ieee.org/about/at-a-glance.html>. Acesso em: 27 apr. 2025.

ENNS, R. **NETCONF Configuration Protocol**. RFC Editor, 2006. RFC 4741. (Request for Comments, 4741). Disponível em: <https://doi.org/10.17487/RFC4741>.

ENNS, R. et al. **Network Configuration Protocol (NETCONF)**. RFC Editor, 2011. RFC 6241. (Request for Comments, 6241). Disponível em: <https://doi.org/10.17487/RFC6241>.

FEDOR, M. et al. **Simple Network Management Protocol (SNMP)**. RFC Editor, 1990. RFC 1157. (Request for Comments, 1157). Disponível em: <https://www.rfc-editor.org/info/rfc1157>.

FENG, W.; TAN, X.; JIN, Y. Implementing icn over p4 in http scenario. In: **2019 2nd International Conference on Hot Information-Centric Networking (HotICN)**. [s.n.], 2019. p. 37–43. Disponível em: <https://doi.org/10.1109/HotICN48464.2019.9063219>.

FERNANDES, L. B.; CAMARGOS, L. Bandwidth throttling in a p4 switch. In: **2020 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)**. [s.n.], 2020. p. 91–94. Disponível em: <https://doi.org/10.1109/NFV-SDN50289.2020.9289836>.

FFMPEG, F. **A complete, cross-platform solution to record, convert and stream audio and video.** 2024. Disponível em: <https://ffmpeg.org/>. Acesso em: 18 may. 2025.

FISHER, D. A look behind the future internet architectures efforts. **SIGCOMM Comput. Commun. Rev.**, Association for Computing Machinery, New York, NY, USA, v. 44, n. 3, p. 45–49, jul. 2014. ISSN 0146-4833. Disponível em: <https://doi.org/10.1145/2656877.2656884>.

FONSECA, P. C.; MOTA, E. S. A survey on fault management in software-defined networks. **IEEE Communications Surveys & Tutorials**, v. 19, n. 4, p. 2284–2321, 2017. Disponível em: <https://doi.org/10.1109/COMST.2017.2719862>.

FORCE, I. I. E. T. **Introduction to the IETF**. 2025. Disponível em: <https://www.ietf.org/about/introduction/>. Acesso em: 27 apr. 2025.

FOTIOU, N. et al. Developing information networking further: From psirp to pursuit. In: TOMKOS, I. et al. (Ed.). **Broadband Communications, Networks, and Systems**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012. p. 1–13. ISBN 978-3-642-30376-0. Disponível em: <https://doi.org/10.1007/978-3-642-30376-0_1>.

FOUNDATION, N. U. N. S. **NSF 10-528: Future Internet Architectures (FIA)**. 2010. Disponível em: <https://new.nsf.gov/funding/opportunities/fia-future-internet-architectures/503476/nsf10-528/solicitation>. Acesso em: 08 dez. 2024.

_____. **NSF 13-538: Future Internet Architectures – Next Phase (FIA-NP)**. 2013. Disponível em: <https://new.nsf.gov/funding/opportunities/fia-np-future-internet-architectures-next-phase/504882/nsf13-538/solicitation>. Acesso em: 08 dez. 2024.

FOUNDATION, O. O. N. **OpenFlow Table Type Patterns - Version No. 1.0**. 2014. Disponível em: <https://opennetworking.org/wp-content/uploads/2013/04/OpenFlow%20Table%20Type%20Patterns%20v1.0.pdf>. Acesso em: 27 apr. 2025.

_____. **OpenFlow Switch Specification - Version 1.5.1**. 2015. Disponível em: <https://opennetworking.org/wp-content/uploads/2014/10/openflow-switch-v1.5.1.pdf>. Acesso em: 27 apr. 2025.

_____. **Collaboratively transforming network infrastructure**. 2025. Disponível em: <https://opennetworking.org/>. Acesso em: 27 apr. 2025.

_____. **ONOS - Open Network Operating System**. 2025. Disponível em: <https://opennetworking.org/onos/>. Acesso em: 27 apr. 2025.

FOUNDATION, T. L. **CAMARA - THE TELCO GLOBAL API ALLIANCE**. 2023. Disponível em: <https://camaraproject.org/>. Acesso em: 30 jul. 2025.

GAVAZZA, J. A. T. et al. Future internet exchange point (fixp): Enabling future internet architectures interconnection. In: BAROLLI, L. et al. (Ed.). **Advanced Information Networking and Applications**. Cham: Springer International Publishing, 2020. p. 703–714. ISBN 978-3-030-44041-1.

GIMENEZ, S.; GRASA, E.; BUNCH, S. A proof of concept implementation of a rina interior router using p4-enabled software targets. In: **2020 23rd Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN)**. [s.n.], 2020. p. 57–62. Disponível em: <https://doi.org/10.1109/ICIN48450.2020.9059486>.

GONÇALVES, M. A. et al. **Multicast traffic aggregation through entity title model**. [S.l.]: Citeseer, 2014.

GUIMARãES, C. et al. Ieee 802.21-enabled entity title architecture for handover optimization. In: **2014 IEEE Wireless Communications and Networking Conference (WCNC)**. [s.n.], 2014. p. 2671–2676. Disponível em: <https://doi.org/10.1109/WCNC.2014.6952830>.

_____. Exploring interoperability assessment for future internet architectures roll out. **Journal of Network and Computer Applications**, v. 136, p. 38–56, 2019. ISSN 1084-8045. Disponível em: <https://doi.org/10.1016/j.jnca.2019.04.008>.

GUO, X. et al. An efficient ndn routing mechanism design in p4 environment. In: **2021 2nd Information Communication Technologies Conference (ICTC)**. [s.n.], 2021. p. 28–33. Disponível em: <https://doi.org/10.1109/ICTC51749.2021.9441639>.

GUPTA, A. et al. Sdx: a software defined internet exchange. In: **Proceedings of the 2014 ACM Conference on SIGCOMM**. New York, NY, USA: Association for Computing Machinery, 2014. (SIGCOMM Í4), p. 551–562. ISBN 9781450328364. Disponível em: <https://doi.org/10.1145/2619239.2626300>.

HAN, D. et al. {XIA}: Efficient support for evolvable internetworking. In: **9th USENIX Symposium on Networked Systems Design and Implementation (NSDI 12)**. San Jose, CA: USENIX Association, 2012. p. 309–322. ISBN 978-931971-92-8. Disponível em: <https://www.usenix.org/conference/nsdi12/technical-sessions/presentation/han_dongsu_xia>.

HE, C.-H. et al. A zero flow entry expiration timeout p4 switch. In: **Proceedings of the Symposium on SDN Research**. New York, NY, USA: Association for Computing Machinery, 2018. (SOSR '18). ISBN 9781450356640. Disponível em: <https://doi.org/10.1145/3185467.3190785>.

HU, F.; HAO, Q.; BAO, K. A survey on software-defined network and openflow: From concept to implementation. **IEEE Communications Surveys & Tutorials**, v. 16, n. 4, p. 2181–2206, 2014. Disponível em: <https://doi.org/10.1109/COMST.2014.2326417>.

INTEL, C. A. **P416 Programming for Intel Tofino using Intel P4 Studio**. 2021. Disponível em: <https://opennetworking.org/wp-content/uploads/2021/05/2021-P4-WS-Vladimir-Gurevich-Slides.pdf>. Acesso em: 17 ago. 2025.

ISHAKIAN, V. et al. On supporting mobility and multihoming in recursive internet architectures. **Computer Communications**, v. 35, n. 13, p. 1561–1573, 2012. ISSN 0140-3664. Disponível em: <https://doi.org/10.1016/j.comcom.2012.04.027>.

ITU-T, I.-T. **X700: Management framework for Open Systems Interconnection (OSI) for CCITT applications**. 1992. Disponível em: <https://www.itu.int/rec/T-REC-X.700-199209-I/en>. Acesso em: 30 jul. 2025.

JOKELA, P. et al. Lipsin: line speed publish/subscribe inter-networking. In: **Proceedings of the ACM SIGCOMM 2009 Conference on Data Communication**. New York, NY, USA: Association for Computing Machinery, 2009. (SIGCOMM '09), p. 195–206. ISBN 9781605585949. Disponível em: <https://doi.org/10.1145/1592568.1592592>.

JOUET, S.; CZIVA, R.; PEZAROS, D. P. Arbitrary packet matching in openflow. In: **2015 IEEE 16th International Conference on High Performance Switching and Routing (HPSR)**. [s.n.], 2015. p. 1–6. Disponível em: <https://doi.org/10.1109/HPSR.2015.7483106>.

KARRAKCHOU, O.; SAMAAN, N.; KARMOUCH, A. Endn: An enhanced ndn architecture with a p4-programmabie data plane. In: **Proceedings of the 7th ACM Conference on Information-Centric Networking**. New York, NY, USA: Association for Computing Machinery, 2020. (ICN 2́0), p. 1–11. ISBN 9781450380409. Disponível em: <https://doi.org/10.1145/3405656.3418720>.

KAUR, S.; KUMAR, K.; AGGARWAL, N. A review on p4-programmable data planes: Architecture, research efforts, and future directions. **Computer Communications**, v. 170, p. 109–129, 2021. ISSN 0140-3664. Disponível em: <https://doi.org/10.1016/j.comcom.2021.01.027>.

KOHLER, E. et al. The click modular router. **ACM Trans. Comput. Syst.**, Association for Computing Machinery, New York, NY, USA, v. 18, n. 3, p. 263–297, ago. 2000. ISSN 0734-2071. Disponível em: <https://doi.org/10.1145/354871.354874>.

KREUTZ, D. et al. Software-defined networking: A comprehensive survey. **Proceedings of the IEEE**, v. 103, n. 1, p. 14–76, 2015. Disponível em: <https://doi.org/10.1109/JPROC.2014.2371999>.

KUMAR, S. D.; SK, N. M. A novel ternary content-addressable memory (tcam) design using reversible logic. In: **2015 28th International Conference on VLSI Design**. [s.n.], 2015. p. 316–320. Disponível em: <https://doi.org/10.1109/VLSID.2015.99>.

KUROSE, K. R. Computer networking: A top-down approach by james. **Kurose, Keith W. Ross**, p. 601, 2017.

KWON, J. et al. Scionlab: A next-generation internet testbed. In: **2020 IEEE 28th International Conference on Network Protocols (ICNP)**. [s.n.], 2020. p. 1–12. Disponível em: <https://doi.org/10.1109/ICNP49622.2020.9259355>.

LANGUAGE, C. P. **P4$_1$6 Language Specification**. 2023. Disponível em: <https://p4.org/wp-content/uploads/sites/53/2024/10/P4-16-spec-v1.2.5.pdf>. Acesso em: 20 dec. 2024.

LANGUAGE, P. **The BMv2 Simple Switch target**. 2024. Disponível em: <https://github.com/p4lang/behavioral-model/blob/main/docs/simple_switch.md>. Acesso em: 27 apr. 2025.

LARA, A.; KOLASANI, A.; RAMAMURTHY, B. Network innovation using openflow: A survey. **IEEE Communications Surveys & Tutorials**, v. 16, n. 1, p. 493–512, 2014. Disponível em: <https://doi.org/10.1109/SURV.2013.081313.00105>.

LEINER, B. M. et al. A brief history of the internet. **SIGCOMM Comput. Commun. Rev.**, Association for Computing Machinery, New York, NY, USA, v. 39, n. 5, p. 22–31, out. 2009. ISSN 0146-4833. Disponível em: <https://doi.org/10.1145/1629607.1629613>.

LEMA, J. C. et al. Evolving future internet clean-slate entity title architecture with quality-oriented control plane extensions. In: CITESEER. **The Tenth Advanced International Conference on Telecommunications (AICT), ThinkMind, Ed. IARIA**. [S.l.], 2014. p. 161–167.

MACHADO, M.; DOUCETTE, C.; BYERS, J. W. Linux xia: an interoperable meta network architecture to crowdsource the future internet. In: **2015 ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS)**. [s.n.], 2015. p. 147–158. Disponível em: <https://doi.org/10.1109/ANCS.2015.7110128>.

MARCONI, M. d. A.; LAKATOS, E. M. Técnicas de pesquisa: planejamento e execução de pesquisa; amostragens e técnicas de pesquisa; elaboração, análise e interpretação de dados. In: **Técnicas de pesquisa: planejamento e execução de pesquisa; amostragens e técnicas de pesquisa; elaboração, análise e interpretação de dados**. [S.l.: s.n.], 2012. p. 277–277.

MCCLOGHRIE, K.; ROSE, M. **Management Information Base for Network Management of TCP/IP-based internets: MIB-II**. RFC Editor, 1991. RFC 1213. (Request for Comments, 1213). Disponível em: <https://doi.org/10.17487/RFC1213>.

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>.

MCKEOWN, N.; REXFORD, J. **Clarifying the differences between P4 and OpenFlow**. 2016. Disponível em: <https://opennetworking.org/news-and-events/blog/clarifying-the-differences-between-p4-and-openflow/>. Acesso em: 27 apr. 2025.

MOLERO, E. C.; VISSICCHIO, S.; VANBEVER, L. Hardware-accelerated network control planes. In: **Proceedings of the 17th ACM Workshop on Hot Topics in Networks**. New York, NY, USA: Association for Computing Machinery, 2018. (HotNets '18), p. 120–126. ISBN 9781450361200. Disponível em: <https://doi.org/10.1145/3286062.3286080>.

MOREIRA, R. et al. Nasor: A network slicing approach for multiple autonomous systems. **Computer Communications**, v. 179, p. 131–144, 2021. ISSN 0140-3664. Disponível em: <https://doi.org/10.1016/j.comcom.2021.07.028>.

MOY, J. **OSPF Version 2**. RFC Editor, 1998. RFC 2328. (Request for Comments, 2328). Disponível em: <https://doi.org/10.17487/RFC2328>.

NASCIMENTO, M. R. et al. Virtual routers as a service: the routeflow approach leveraging software-defined networks. In: **Proceedings of the 6th International Conference on Future Internet Technologies**. New York, NY, USA: Association for Computing Machinery, 2011. (CFI Í1), p. 34–37. ISBN 9781450308212. Disponível em: <https://doi.org/10.1145/2002396.2002405>.

NETO, N. V. d. S. et al. Autocura para redes definidas por software. Universidade Federal de Uberlândia, 2021.

NETO, N. V. de S. et al. Control plane routing protocol for the entity title architecture: Design and specification. **ICN 2015**, p. 197, 2015.

NGMN, N. **NGMN - ABOUT US**. 2025. Disponível em: <https://www.ngmn.org/>. Acesso em: 30 jul. 2025.

NORDSTRÖM, E. et al. Serval: An End-Host stack for Service-Centric networking. In: **9th USENIX Symposium on Networked Systems Design and Implementation (NSDI 12)**. San Jose, CA: USENIX Association, 2012. p. 85–98. ISBN 978-931971-92-8. Disponível em: <https://www.usenix.org/conference/nsdi12/technical-sessions/presentation/nordstrom>.

NTP.BR, N. **ntp.br**. 2025. Disponível em: <https://ntp.br/>. Acesso em: 08 aug. 2025.

OIKARINEN, J. **Internet Relay Chat Protocol**. RFC Editor, 1993. RFC 1459. (Request for Comments, 1459). Disponível em: <https://doi.org/10.17487/RFC1459>.

ORDONEZ-LUCENA, J.; DSOUZA, F. Pathways towards network-as-a-service: the camara project. In: **Proceedings of the ACM SIGCOMM Workshop on Network-Application Integration**. New York, NY, USA: Association for Computing Machinery, 2022. (NAI '22), p. 53–59. ISBN 9781450393959. Disponível em: <https://doi.org/10.1145/3538401.3546825>.

PAOLUCCI, F. et al. Latency control in service chaining using p4-based data plane programmability. **Computer Networks**, v. 216, p. 109227, 2022. ISSN 1389-1286. Disponível em: <https://doi.org/10.1016/j.comnet.2022.109227>.

PAPASTERGIOU, G. et al. De-ossifying the internet transport layer: A survey and future perspectives. **IEEE Communications Surveys & Tutorials**, v. 19, n. 1, p. 619–639, 2017. Disponível em: <https://doi.org/10.1109/COMST.2016.2626780>.

PEREIRA, J. H. de S.; KOFUJI, S. T.; ROSA, P. F. Horizontal addressing by title in a next generation internet. In: **2010 Sixth International Conference on Networking and Services**. [s.n.], 2010. p. 7–11. Disponível em: <https://doi.org/10.1109/ICNS.2010.9>.

PROJECT, C. P. C. **camaraproject/Governance**. 2025. Disponível em: <https://github.com/camaraproject/Governance/blob/main/PARTICIPANTS.MD>. Acesso em: 30 jul. 2025.

RAMIREZ-PEREZ, C.; RAMOS, V. Sdn meets sdr in self-organizing networks: fitting the pieces of network management. **IEEE Communications Magazine**, v. 54, n. 1, p. 48–57, 2016. Disponível em: <https://doi.org/10.1109/MCOM.2016.7378425>.

RAYCHAUDHURI, D.; NAGARAJA, K.; VENKATARAMANI, A. Mobilityfirst: a robust and trustworthy mobility-centric architecture for the future internet. **SIGMOBILE Mob. Comput. Commun. Rev.**, Association for Computing Machinery, New York, NY, USA, v. 16, n. 3, p. 2–13, dez. 2012. ISSN 1559-1662. Disponível em: <https://doi.org/10.1145/2412096.2412098>.

REXFORD, J.; DOVROLIS, C. Future internet architecture: clean-slate versus evolutionary research. **Commun. ACM**, Association for Computing Machinery, New York, NY, USA, v. 53, n. 9, p. 36–40, set. 2010. ISSN 0001-0782. Disponível em: <https://doi.org/10.1145/1810891.1810906>.

ROBERTS, J. The clean-slate approach to future internet design: a survey of research initiatives. **annals of telecommunications-annales des télécommunications**, Springer, v. 64, p. 271–276, 2009. Disponível em: <https://doi.org/10.1007/s12243-009-0109-y>.

ROSE, M. **Management Information Base for network management of TCP/IP-based internets: MIB-II**. RFC Editor, 1990. RFC 1158. (Request for Comments, 1158). Disponível em: <https://doi.org/10.17487/rfc1158>.

RUOHONEN, J. **Evaluating the Network Management Capabilities of YANG and NETCONF**. Tese (Doutorado) — Tampere University, 2016.

SAXENA, D. et al. Named data networking: A survey. **Computer Science Review**, v. 19, p. 15–55, 2016. ISSN 1574-0137. Disponível em: <https://doi.org/10.1016/j.cosrev.2016.01.001>.

SCHULZRINNE, H. et al. **RTP: A Transport Protocol for Real-Time Applications**. RFC Editor, 2003. RFC 3550. (Request for Comments, 3550). Disponível em: <https://doi.org/10.17487/RFC3550>.

SCIENCE, M. Laboratory for C.; INSTITUTE, I. I. C. S.; DIVISION, U. I. S. I. C. N. **NewArch Project: Future-Generation Internet Architecture**. 2000. Disponível em: <https://www.isi.edu/newarch/>. Acesso em: 08 dez. 2024.

SIGNORELLO, S. et al. Ndn.p4: Programming information-centric data-planes. In: **2016 IEEE NetSoft Conference and Workshops (NetSoft)**. [s.n.], 2016. p. 384–389. Disponível em: <https://doi.org/10.1109/NETSOFT.2016.7502472>.

SILVA, F. et al. Entity title architecture extensions towards advanced quality-oriented mobility control capabilities. In: **2014 IEEE Symposium on Computers and Communications (ISCC)**. [s.n.], 2014. p. 1–6. Disponível em: <https://doi.org/10.1109/ISCC.2014.6912459>.

SILVA, F. d. O.; KOFUJI, S. T.; ROSA, P. F. **Endereçamento por título: uma forma de encaminhamento multicast para a próxima geração de redes de computadores**. Dissertação (Mestrado) — Universidade de São Paulo, 2013.

SILVA, F. de O. et al. On the analysis of multicast traffic over the entity title architecture. In: **2012 18th IEEE International Conference on Networks (ICON)**. [s.n.], 2012. p. 30–35. Disponível em: <https://doi.org/10.1109/ICON.2012.6506529>.

SILVA, J. N. d. **Implementação de plano de dados programável para a eXpressive Internet Architecture usando a Linguagem P4**. Tese (Doutorado) — Universidade Federal de Pernambuco, UFPE, 2020. Disponível em: <https://repositorio.ufpe.br/handle/123456789/38544>.

SILVA, T. B. **Novagenesiscontrolagent for future Internet eXchange point**. Dissertação (Mestrado) — Instituto Nacional de Telecomunicações, 2021.

SILVA, T. B. da et al. Toward next-generation and service-defined networks: A novagenesis control agent for future internet exchange point. **IEEE Network**, v. 36, n. 3, p. 74–81, 2022. Disponível em: <https://doi.org/10.1109/MNET.008.2100555>.

SOFTWARE, O. O. S. **ONF OpenFlow Conformant: Certified Product List**. 2019. Disponível em: <https://opennetworking.org/product-registry/>. Acesso em: 27 apr. 2025.

SPIEGEL, M. R.; SCHILLER, J. J.; SRINIVASAN, R. A. **Probabilidade e Estatística: Coleção Schaum**. [S.l.]: Bookman Editora, 2016.

STRINGER, J. et al. Cardigan: Sdn distributed routing fabric going live at an internet exchange. In: **2014 IEEE Symposium on Computers and Communications (ISCC)**. [s.n.], 2014. p. 1–7. Disponível em: <https://doi.org/10.1109/ISCC.2014.6912501>.

SUJATA, J. et al. Impact of over the top (ott) services on telecom service providers. **Indian Journal of Science and Technology**, Indian Society for Education and Environment, v. 8, n. S4, p. 145–160, 2015. ISSN 0974-5645. Disponível em: <10.17485/ijst/2015/v8iS4/62238>.

TELECOMUNICAçõES, I. Instituto Nacional de. **NovaGenesis: Base Objects**. 2021. Disponível em: <https://inatel.br/novagenesis/specifications>. Acesso em: 27 apr. 2025.

_____. **Novagenesis control agent for future Internet eXchange point**. 2021. Disponível em: <https://bdtd.ibict.br/vufind/Record/INAT_c72db5692ab43581fd135c83eadf131f#details>. Acesso em: 25 may. 2025.

TENNENHOUSE, D. et al. A survey of active network research. **IEEE Communications Magazine**, v. 35, n. 1, p. 80–86, 1997. Disponível em: <https://doi.org/10.1109/35.568214>.

TIME, U. C. U. **UTC agora**. 2025. Disponível em: <https://time.is/pt_br/UTC>. Acesso em: 14 may. 2025.

TROSSEN, D.; PARISIS, G. Designing and realizing an information-centric internet. **IEEE Communications Magazine**, v. 50, n. 7, p. 60–67, 2012. Disponível em: <https://doi.org/10.1109/MCOM.2012.6231280>.

TROUVA, E. et al. Transport over heterogeneous networks using the rina architecture. In: MASIP-BRUIN, X. et al. (Ed.). **Wired/Wireless Internet Communications**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011. p. 297–308. ISBN 978-3-642-21560-5.

_____. Layer discovery in rina networks. In: **2012 IEEE 17th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD)**. [s.n.], 2012. p. 368–372. Disponível em: <https://doi.org/10.1109/CAMAD.2012.6335369>.

TURNER, J.; TAYLOR, D. Diversifying the internet. In: **GLOBECOM ́05. IEEE Global Telecommunications Conference, 2005.** [s.n.], 2005. v. 2, p. 6 pp.–760. Disponível em: <https://doi.org/10.1109/GLOCOM.2005.1577741>.

UFRJ, U. **Arquitetura de um SDN.** 2015. Disponível em: <https://www.gta.ufrj.br/ensino/eel879/trabalhos_vf_2015_2/SDN/architecture.html>. Acesso em: 25 may. 2025.

UNIVERSITY, L. B. **RINA: an architecture for policy-based dynamic service management**. 2013. Disponível em: <https://hdl.handle.net/2144/11422>. Acesso em: 27 apr. 2025.

VENKATARAMANI, A. et al. Mobilityfirst: a mobility-centric and trustworthy internet architecture. **SIGCOMM Comput. Commun. Rev.**, Association for Computing Machinery, New York, NY, USA, v. 44, n. 3, p. 74–80, jul. 2014. ISSN 0146-4833. Disponível em: <https://doi.org/10.1145/2656877.2656888>.

VRIJDERS, S. et al. Experimental evaluation of a recursive internetwork architecture prototype. In: **2014 IEEE Global Communications Conference**. [s.n.], 2014. p. 2017–2022. Disponível em: <https://doi.org/10.1109/GLOCOM.2014.7037104>.

WANG, C.; YAN, S. Scaling sdn network with self-adjusting architecture. In: **2016 IEEE International Conference on Electronic Information and Communication Technology (ICEICT)**. [s.n.], 2016. p. 116–120. Disponível em: <https://doi.org/10.1109/ICEICT.2016.7879664>.

WETHERALL, D. J. **Service introduction in an active network**. Tese (Doutorado) — Massachusetts Institute of Technology, 1998.

WORKING, G. P. A. **P4RuntimeSpecification**. 2024. Disponível em: <https://p4lang.github.io/p4runtime/spec/v1.5.0/P4Runtime-Spec.pdf>. Acesso em: 22 mar. 2025.

XIE, J. et al. Control plane of software defined networks: A survey. **Computer Communications**, v. 67, p. 1–10, 2015. ISSN 0140-3664. Disponível em: <https://doi.org/10.1016/j.comcom.2015.06.004>.

XYLOMENOS, G. et al. Caching and mobility support in a publish-subscribe internet architecture. **IEEE Communications Magazine**, v. 50, n. 7, p. 52–58, 2012. Disponível em: <https://doi.org/10.1109/MCOM.2012.6231279>.

YI, C. et al. A case for stateful forwarding plane. **Computer Communications**, v. 36, n. 7, p. 779–791, 2013. ISSN 0140-3664. Disponível em: <https://doi.org/10.1016/j.comcom.2013.01.005>.

ZANDER, J.; FORCHHEIMER, R. Softnet–an approach to high level packet communication. In: **2nd ARRL Amateur Radio Computer Networking Conference, San Francisco, CA, USA, March 19, 1983**. [S.l.: s.n.], 1983. p. 1–2.

ZANNA, P.; RADCLIFFE, P.; CHAVEZ, K. G. A method for comparing openflow and p4. In: **2019 29th International Telecommunication Networks and Applications Conference (ITNAC)**. [s.n.], 2019. p. 1–3. Disponível em: <https://doi.org/10.1109/ITNAC46935.2019.9077951>.

ZHANG, L. et al. Named data networking. **SIGCOMM Comput. Commun. Rev.**, Association for Computing Machinery, New York, NY, USA, v. 44, n. 3, p. 66–73, jul. 2014. ISSN 0146-4833. Disponível em: <https://doi.org/10.1145/2656877.2656887>.

ZHANG, X. et al. Scion: Scalability, control, and isolation on next-generation networks. In: **2011 IEEE Symposium on Security and Privacy**. [s.n.], 2011. p. 212–227. Disponível em: <https://doi.org/10.1109/SP.2011.45>.

Apêndices

APÊNDICE A

Apêndice A

A.1 Sobre a programabilidade do plano de dados do *switch* FIXP

Este apêndice tem o objetivo de apresentar parcialmente o documento P4 que especifica o plano de dados das arquiteturas implantadas na rede *FIXP*. O código do documento tem aproximadamente 550 linhas, pois integra e define o plano de dados de 03 arquiteturas de Internet diferentes: ETArch, IP e NovaGenesis. Além disso, esse documento também é responsável pelo comportamento do switch em relação às primitivas do protocolo FIXP. Por conta do tamanho do documento, decidiu-se apresentar neste apêndice apenas os principais trechos de código que são responsáveis pela programação do pipeline de dados da arquitetura ETArch.

A.1.1 Considerações iniciais

O fabricante de targets *P4* tem que fornecer pelo menos dois elementos que são essenciais para o seu funcionamento: uma definição de arquitetura programável (modelo da arquitetura) e um compilador (LANGUAGE, 2023). Um target é projetado a partir de uma determinada arquitetura. O modelo de arquitetura pode ser pensada como se fosse um contrato entre programa e target (LANGUAGE, 2023).

O *switch FIXP* é uma extensão do *BMv2 Simple Switch target* (LANGUAGE, 2024), que utiliza como referência a arquitetura *"v1model.p4"*.

As subseções a seguir descrevem as notações e implementações das interfaces dos blocos funcionais programáveis, como também outras declarações do *"v1model.p4"* que devem ser utilizadas para programar o plano de dados, mostrando dessa forma, como o pipeline de dados pode ser manipulado por um programa *P4*; e, por conseguinte, pelos controladores que participam da rede *FIXP*. Em termos gerais, o *switch FIXP* possui 03 blocos ou estágios programáveis: *Parser*, *Match-Action* e *Deparser*; e, a linguagem $P4_{16}$ possui 05 categorias principais de elementos: tipos de dados, expressões, elementos de

controle de fluxo, *Parsers* e *Externs* (LANGUAGE, 2023). Dentro dessas categorias, cita-se as estruturas de dados *header*, *struct*; e, as instruções *control*, *state*, *transition*, *table* e *action* (LANGUAGE, 2023). Alguns desses elementos estão contidos nos algoritmos que são apresentados nesta seção.

A Subseção A.1.2 apresenta como se faz as declarações de algumas das principais estruturas de dados da linguagem P4, que são diretamente responsáveis pela descrição dos formatos das primitivas. A Subseções A.1.3, A.1.4 e A.1.5 apresentam os estágios de processamento do pipeline de dados: *Parser*, *Match-Action* e *Deparser*, respectivamente. Esses algoritmos descritivos fazem parte do documento *P4* que configura o pipeline de dados dos switches *FIXP* utilizados nos casos de uso desta tese.

A.1.2 Declaração de dados

Basicamente, um programa *P4* começa com a definição dos cabeçalhos que devem ser reconhecidos pelo *switch FIXP*.

Os cabeçalhos dos protocolos de controle e as primitivas de dados da ETArch são encapsulados pelo protocolo *Ethernet* através de um *Raw Socket*. Por conta disso, o *switch FIXP* tem que extrair primeiramente o protocolo Ethernet, e em seguida, extrair os cabeçalhos da ETArch para posteriormente analisá-los; caso seja necessário. Como o endereço de destino da ETArch (*workspace*) está registrado no endereço de destino (*destination MAC*) do quadro Ethernet, o *switch FIXP* não precisa analisar os cabeçalhos da ETArch no momento da comutação.

Segue abaixo a definição dos cabeçalhos *Ethernet* e ETArch que o *switch* precisa reconhecer. Em um momento posterior, o Parser extrai as informações dos protocolos e as coloca nessas estruturas de dados que estão sendo declaradas logo no começo do documento.

```
1  /
2  typedef bit<9>  egressSpec_t;
3  typedef bit<48> macAddr_t;
4  typedef bit<32> ip4Addr_t;
5  typedef bit<32> xptoAddr_t;
6  typedef bit<16> mcastGroup_t;
7
8  header ethernet_t
9  {                    //nao e considerado o preambulo
10     macAddr_t dst;   //Endereco mac de destino
11     macAddr_t src;   //endereco mac de origem
12     bit<16>  type;  //tipo do protocolo encapsulado, ou seja, ethertype
13                     // nao e considerado o CRC
14 }
15
16 header etarch_t
```

```
{
    bit<368>  cabEtarch;  //pega o minimo do payload do protocolo Ethernet
}

struct headers
{
    ethernet_t         ethernet;
    ipv4_t             ipv4;
    raw_switch_t       raw_switch;
    etarch_switch_t    etarch_switch;
    etarch_switch_2_t  etarch_switch_2;
    etarch_t           etarch;
    ng_t               ng;
}
```

Algoritmo A.1 – Declaração de dados do Programa P4 dos swithes FIXP

A estrutura de dados *header* especifica o formato das primitivas que vão ser, posteriormente, extraídas para os campos definidos. *Struct* é uma estrutura de dados que lista os *headers* (definidos anteriormente) que são utilizados pelo estágio de processamento *Parser*.

A.1.3 Estágio de processamento Parser

Uma vez definidas os *headers*, precisa-se definir a lógica da máquina finita do estágio de processamento Parser. Cada estado da máquina finita extrai uma camada da primitiva de acordo com as necessidades de cada FIA.

```
/
state start
{
  transition parse_ethernet;
}

state parse_ethernet
{
  packet.extract(hdr.ethernet);

  transition select(hdr.ethernet.type, hdr.ethernet.dst, hdr.ethernet.src)
  {

    (0x800,_,_)                  : parse_ipv4;

    (0x0900,0x464958500000,_)    : accept;

    (0x0900,_,0x464958500000)    : accept;

```

```
20      (0x0900,_,_)                    : parse_fixp_switch;
21
22      (0x0880,0x445453000000,_)       : parse_etarch;
23
24      (0x0880,0xffffffffffff,_)       : parse_etarch_switch;
25
26      (0x0880,_,_)                    : parse_etarch;
27
28    }
29
30  }
31
32  state parse_etarch
33  {
34    packet.extract(hdr.etarch);
35    transition accept;
36  }
```

Algoritmo A.2 – Estágio de processamento Parser dos switches FIXP

O algoritmo, primeiramente, na linha 9, extrai da primitiva ETArch recebida (se esse for o caso) o cabeçalho Ethernet, e posteriormente, na linha 11, analisa os dados de três campos: *hdr.ethernet.type* (tipo), *hdr.ethernet.dst* (MAC de destino), *hdr.ethernet.src* (MAC de origem). Se a primitiva que chega ao *switch* possui o campo *type* 0x0880, quer dizer que essa primitiva pertence à arquitetura ETArch. Se essa mesma primitiva possui o campo dst (MAC de destino) igual a 0x445453000000 ('DTS' em decimal), quer dizer que se trata de uma primitiva de controle, portanto, a máquina finita modifica seu estado para "parse_etarch", como pode ser visto na linha 22. No estado "parse_etarch"(linha 32), há a extração dos cabeçalhos da primitiva ETArch, que pega os 46 bytes/368 bits do *payload* do protocolo *Ethernet*, conforme foi definido pelo *header etarch* (ver Subseção A.1.2). Logo em seguida, a transição é aceita (linha 35); ou seja, o processamento da máquina finita chega ao fim. Nesse ponto, houve as extrações dos cabeçalhos *Ethernet* e ETArch e os dados extraídos são colocados nas estruturas de dados anteriormente definidas. Todas essas informações, como também alguns metadados, são passados para o estágio posterior de processamento do pipeline, que é o *Match-Action*.

A.1.4 Estágio de processamento Match-Action

Geralmente, as unidades de *Match-Action* tais como ações e tabelas são declaradas dentro de um bloco de controle (LANGUAGE, 2023). Dentro desse bloco, podem ser instanciadas várias tabelas, como também podem ser invocadas ações de forma explícita (LANGUAGE, 2023).

Uma declaração de tabela introduz a instância dessa tabela no *switch FIXP*. Trata-se do que se chama na literatura de tabela de encaminhamento. Sinteticamente, do

ponto de vista de estrutura de dados, é um *map* cujo conteúdo é manipulado de forma assíncrona (*read/write*) pelo módulo *FRHS* do *FIXP* (ver Seção 3.3). Pode-se dizer, de maneira breve, que a tabela é definida em termos de um conjunto de propriedades chave-valor (*key-value*) (LANGUAGE, 2023). Suas principais propriedades são ações e chaves (LANGUAGE, 2023). As chaves indexam as entradas da tabela e determina o valor que é utilizado para invocar uma ação específica, caso aconteça uma correspondência na procura da chave. As ações, como o próprio nome sugere, é uma espécie de função, que é executada para determinada fim; como por exemplo, indicar o valor da porta de egresso de determinada primitiva.

Primeiramente, todas as ações e tabelas do *Match-Action*, no caso do documento *P4* da rede *FIXP*, são declarados dentro de um *control block*, conforme abaixo.

```
1  /
2  control FIXP_Switch_Ingress(inout headers hdr,
3                              inout metadata meta,
4                              inout standard_metadata_t standard_metadata)
5  {
6
7
8      action ToControllerEtharc() {}
9
10     action etarch_SetSpec_Group(mcastGroup_t mcastGroup) {}
11
12     table etarch_forward {}
13
14     apply { }
15 }
```

Algoritmo A.3 – Estágio de processamento Match-Action. Bloco de controle para primitivas de ingresso

As implementações das ações (*actions*) (linhas 8 e 10), tabela(*table*) (linha 12) e *apply* (linha 14), cujas assinaturas estão presentes no algoritmo acima, são apresentadas ainda nessa seção. Por enquanto, é preciso saber que as definições das ações e da tabela acima são suficiente para que o *switch FIXP* processe apenas algumas das primitivas da ETArch, pois apenas uma parte do algoritmo está sendo apresentado. Sendo assim, a tabela *table etarch_forward* representa a tabela de encaminhamento da arquitetura ETArch que é instanciada pelo *switch FIXP*. A ação *action etarch_SetSpec_Group(mcastGroup_t mcastGroup)* é executada quando o *switch FIXP* replica as primitivas ETArch para várias portas de egresso; como por exemplo, no caso em que um servidor ETArch está enviando streaming de vídeo para vários clientes simultaneamente. Para que essa ação seja invocada, deve existir uma correspondência de chaves entre a primitiva correspondente e a tabela de encaminhamento do *switch*. A ação *action ToControllerEtharc()* é executada quando essa correspondência não acontece, caso em que o *switch FIXP* encaminha a

primitiva para a porta de egresso do controlador correspondente. O sub-bloco apply é responsável por fazer a invocação explícita das tabelas (pesquisas de índices) ou das ações definidas no bloco.

```
1 /
2 action ToControllerEtharc()
3 {
4   if(hdr.etarch.isValid()) {
5     standard_metadata.egress_spec = 1;
6   }
7 }
```

Algoritmo A.4 – Estágio de processamento Match-Action. Ação ToControllerEtarch para primitivas de ingresso

A ação acima atribui ao metadado standard_metadata.egress_spec o valor 1 (um), que é a porta de controle padrão dos *switches FIXP*. Isso significa que o *switch FIXP* envia a primitiva correspondente da ETArch para o controlador adequado (camada de abstração da arquitetura FIXP) cada vez que essa ação é executada.

```
1 /
2 action etarch_SetSpec_Group(mcastGroup_t mcastGroup)
3 {
4   standard_metadata.mcast_grp = mcastGroup;
5 }
```

Algoritmo A.5 – Estágio de processamento Match-Action. Ação etarch_SetSpec_Group para primitivas de ingresso

A ação acima atribui ao metadado *standard_metadata.mcast_grp* o valor do parâmetro *mcastGroup*, que é o valor que a tabela *etarch_forward* passa para a ação quando acontece um correspondência entre a primitiva que está sendo processada e os índices dessa tabela. Esse parâmetro representa o identificador de grupo *multicast*. Supondo que o valor passado para a ação é 1 (um), e esse grupo *multicast* está vinculado às portas 3, 4 e 5; isso quer dizer que o *switch FIXP* vai replicar a primitiva original e enviá-las para as três portas correspondentes.

```
1 /
2 table etarch_forward
3 {
4   key = {
5     hdr.ethernet.dst: exact;
6   }
7
8   actions = {
9     etarch_SetSpec_Group;
10    ToControllerEtharc;
```

```
11   }
12
13   default_action = ToControllerEtharc();
14 }
```

Algoritmo A.6 – Estágio de processamento Match-Action. Tabela de encaminhamento etarch_forward para primitivas de ingresso

A tabela *etarch_forward*, definida acima (linha 2), é indexada através do campo *hdr.ethernet.dst* (MAC de destino) das primitivas ETArch (linha 5) e o parâmetro de pesquisa estabelecido é *exact*. Esse status de pesquisa indica que só há correspondência se o campo do cabeçalho que está sendo procurado (MAC de destino) tem exatamente o mesmo valor do índice da tabela. A ação *ToControllerEtharc* está configurada como *default* (linha 13) e é executada cada vez que não houver uma correspondência exata na pesquisa efetuada. Como já mencionado nesta mesma subseção, essa ação (*ToControllerEtharc*) envia a primitiva ETArch para o controlador. A primeira ação configurada na tabela é *etarch_SetSpec_Group* (linha 9). Isso quer dizer que essa ação (*etarch_SetSpec_Group*) é executada sempre que houver correspondência entre o índice da tabela de encaminhamento e o MAC de destino da primitiva que está sendo processada. Como já dito nesta mesma subseção, quando o switch FIXP reconhece o identificador do *workspace* (MAC de destino), essa ação replica a primitiva ETArch para todas as portas que estão vinculadas a esse canal de comunicação. Antes da ação ser executada, a tabela *etarch_SetSpec_Group* passa o valor vinculado ao *worskpace* para a ação *etarch_SetSpec_Group*. O parâmetro é o próprio identificador do grupo multicast(ver algoritmo A.5).

```
1  /
2  apply
3  {
4    if(hdr.ipv4.isValid())
5    {
6      ipv4_forward.apply();
7    } else if(hdr.raw_switch.isValid())
8    {
9      ToMakePacketOut();
10   } else if( (hdr.ethernet.isValid()) && (hdr.ethernet.type == 0x0880))
11   {
12     etarch_forward.apply();
13   } else if( (hdr.ethernet.isValid()) && (hdr.ethernet.type == 0x1235))
14   {
15     etarch_forward_ng.apply();
16   } else if(hdr.ng.isValid())
17   {
18     ng_forward.apply();
19   }
20   else{
```

```
21      dropPrimitive();
22    }
23  }
```

Algoritmo A.7 – Estágio de processamento Match-Action. Bloco apply para primitivas de ingresso

Uma das formas de uma tabela ou ação serem explicitamente invocadas é através do sub-bloco *apply* (linha 2) (LANGUAGE, 2023), que nesse caso, encontra-se dentro do bloco de controle *FIXP_Switch_Ingress* (ver algoritmo A.3). A parte desse código que interessa aos propósitos da seção está na linha 10, que analisa a primitiva ETArch. Trata-se de uma instrução de controle de fluxo (condicional) que possui dois argumentos lógicos. O primeiro verifica se a primitiva Ethernet é válida (*hdr.ethernet.isValid()*). Supondo que a resposta desse método é verdadeira (*true*), isso quer dizer que o *switch FIXP* extraiu anteriormente os dados do cabeçalho Ethernet e os colocou na estrutura de dados correspondente. O segundo argumento analisa se a primitiva que está sendo processada tem tipo *0x0880*. Se esse argumento é verdadeiro (*true*), quer dizer que se trata de uma primitiva ETArch. Dessa forma, quando os dois argumentos são verdadeiros, o sub-bloco *apply* invoca a tabela *etarch_forward* através do comando *etarch_forward.apply()* (linha 12). A invocação dessa tabela representa que a pesquisa do campo MAC de destino (*workspace*) da primitiva ETArch que está sendo processada é executada na tabela de encaminhamento *etarch_forward*. A partir daí, todos os processos discutidos nessa seção ao explicar a tabela *etarch_forward* são realizados. Dessa forma, a primitiva então será encaminhada para o controlador ou será replicada para todas os consumidores da aplicação.

```
1  /
2  control FIXP_Switch_Egress(inout headers hdr,
3                  inout metadata meta,
4                  inout standard_metadata_t standard_metadata)
5  {
6    apply {
7      if( (hdr.ethernet.isValid()) && (hdr.ethernet.type == 0x0880))  {
8        if(standard_metadata.egress_port == standard_metadata.ingress_port) {
9          standard_metadata.egress_spec = P4_DROP_PORT;
10       }
11     }
12   }
13 }
```

Algoritmo A.8 – Estágio de processamento Match-Action. Bloco control para primitivas de egresso

O bloco de controle acima faz parte também do estágio de processamento *Match-Action*. A função desse bloco está relacionada à arquitetura ETArch. A primeira instrução condicional (linha 7) verifica se o pacote é ETArch. Se é ETArch, tem probabilidade do

switch FIXP replicar essas primitivas no dispositivo. Então, para as primitivas de egresso, a segunda instrução condicional checa se a porta de egresso e ingresso de uma determinada primitiva são iguais (linha 8), caso em que o *switch FIXP* está enviando a primitiva para o próprio remetente. Nessa situação, o *switch FIXP* descarta a primitiva através de uma porta especial de descarte (*drop port*) (linha 9), representada no algoritmo pela variável *P4_DROP_PORT*.

O último estágio do *switch FIXP* é o *Deparser*, que recebe informações do Parser (dados dos cabeçalhos e metadados) e do *Match-Action* (metadados, como por exemplo, a porta de egresso da primitiva).

A.1.5 Estágio de processamento Deparser

Deparser monta novamente a primitiva, é o inverso do Parser (LANGUAGE, 2023). É implementado em um bloco de controle que tem ao menos um parâmetro, que é o packet_out (LANGUAGE, 2023).

```
1 /
2 control FIXP_Switch_Deparser(packet_out packet, in headers hdr)
3 {
4     apply
5     {
6         packet.emit(hdr.ethernet);
7         packet.emit(hdr.ipv4);
8         packet.emit(hdr.etarch_switch_2);
9         packet.emit(hdr.etarch);
10        packet.emit(hdr.ng);
11    }
12 }
```

Algoritmo A.9 – Estágio de processamento Deparser. Lógica de agrupamento para as primitivas de egresso

O código acima monta a primitiva, em sequência, da seguinte forma: (i) escreve o cabeçalho *Ethernet* (linha 6), (ii) escreve o cabeçalho IPv4 (linha 7), (iii) escreve o cabeçalho *etarch_switch_2* (linha 8), (iv) escreve o cabeçalho ETArch (linha 9) e (v) escreve o cabeçalho NovaGenesis (linha 10). Porém, essa seção está estudando o caso de uso ETArch. Na situação das primitivas ETArch estudadas nesta seção, apenas os cabeçalhos Ethernet (linha 6) e ETArch (linha 9) são encapsulados ou escritos na primitiva final. Isso acontece porque antes de escrever os protocolos na primitiva, o método *emit()* checa implicitamente se o protocolo é válido, ou seja, se esse protocolo foi uma das camadas extraídas no estágio de processamento Parser. Se o resultado é verdadeiro, o protocolo corresponde é escrito na primitiva de egresso. No caso de uso da ETArch, que é objeto de estudo desse apêndice, só os cabeçalhos Ethernet e ETArch foram previamente extraídos.

Assim sendo, só esses dois cabeçalhos compõem a primitiva de egresso enviada pelo switch FIXP ao seu destino final.

A.1.6 Considerações finais

Este apêndice explana parte do documento *P4* que especifica o plano de dados do *switch FIXP*. Nesse aspecto, a rede *FIXP* permite que cada uma dessas FIAs personalize cada estágio de processamento do pipeline de dados, possibilitando o reconhecimento de seus protocolos, a extração dos cabeçalhos, a inserção dos dados coletados em estruturas de dados para fins de análise posterior, a criação de tabelas de encaminhamento, índices, ações e objetos *stateful* que, de modo associativo, promovem a lógica de processamento da primitiva. Além dessas operações programáveis, o *switch FIXP* possui blocos de funções fixas, que realizam, por exemplo, enfileiramentos através de algoritmos de arbitragem que gerenciam as demandas nas portas de entrada e saída do dispositivo.

Como visto no decorrer do apêndice, o documento *P4* apresenta uma linguagem descritiva simples, que define um pipeline de dados com operações existentes em qualquer especificação de switch, seja ele programável ou tradicional. Sendo assim, os blocos do *pipeline* seguem uma ordem de execução sequencial com a possibilidade de uso de instruções de controle de fluxo tais como *if*, *switch* ou *select* (LANGUAGE, 2023). A linguagem P4 não permite instruções de controle de fluxo repetitivas simples ou aninhadas, que poderiam causar eventual aumento de complexidade nos algoritmos do dispositivo. Pode-se dizer que as funções mais complexas do código apresentado são aquelas que realizam operações de busca indexada no momento de invocação de uma tabela (*apply()*), extrações dos dados (*extract()*), escritas de cabeçalhos na montagem da primitiva (*emit()*) e as comparações de validade de *headers* (*valid()*). São funções simples que consomem recursos mínimos de processamento, típicos de dispositivos de comutação. Por exemplo, a busca indexada na tabela de comutação possui complexidade O(1), constante, pois o tempo de execução desse algoritmo é sempre o mesmo, independentemente do tamanho da entrada de dados. Isso acontece porque dispositivos tais como o *Tofino* (INTEL, 2021) possui memória *TCAM*, que realiza a busca com apenas 1 ciclo de *clock* (KUMAR; SK, 2015). Isso indica, que apesar da simplicidade do código, o cálculo da complexidade de um pipeline de dados é complexo, pois até o dispositivo de comutação é um fator que interfere na eficiência do processamento. Esse cálculo não faz parte do escopo deste trabalho, porém, fica demonstrado que o documento *P4* não adiciona sobrecarregamento adicional aos *switches FIXP*, quando comparados às funcionalidades dos outros *switches* de mercado; que possuem o mesmo padrão de processamento. Aliás, os experimentos realizados no Capítulo 4 não indicam sobrecarregamento nos *switches* em relação à armazenamento, processamento e utilização de memória RAM.