UNIVERSIDADE FEDERAL DE UBERLÂNDIA
FACULDADE DE COMPUTAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

ETARCH: PROJETO E DESENVOLVIMENTO DE UMA ARQUITETURA PARA O MODELO DE TÍTULO COM FOCO NA AGREGAÇÃO DE TRÁFEGO MULTICAST

MAURÍCIO AMARAL GONÇALVES

Uberlândia - Minas Gerais
2014

UNIVERSIDADE FEDERAL DE UBERLÂNDIA
FACULDADE DE COMPUTAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

MAURÍCIO AMARAL GONÇALVES

ETARCH: PROJETO E DESENVOLVIMENTO DE UMA ARQUITETURA PARA O MODELO DE TÍTULO COM FOCO NA AGREGAÇÃO DE TRÁFEGO MULTICAST

Dissertação de Mestrado apresentada à Faculdade de Computação da Universidade Federal de Uberlândia, Minas Gerais, como parte dos requisitos exigidos para obtenção do título de Mestre em Ciência da Computação.

Área de concentração: Sistemas de Computação.

Orientador:
Prof. Dr. Pedro Frosi Rosa

Uberlândia, Minas Gerais
2014

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

G635e
2014

Gonçalves, Maurício Amaral, 1988-
Etarch: projeto e desenvolvimento de uma arquitetura para o modelo de título com foco na agregação de tráfego Multicast / Maurício Amaral Gonçalves. - 2014.
144 f. : il.

Orientador: Pedro Frosi Rosa.
Dissertação (mestrado) - Universidade Federal de Uberlândia, Programa de Pós-Graduação em Ciência da Computação.
Inclui bibliografia.

1. Computação - Teses. 2. Internet - Teses. 3. Telecomunicações - Teses. I. Rosa, Pedro Frosi. II. Universidade Federal de Uberlândia. Programa de Pós-Graduação em Ciência da Computação. III. Título.

CDU: 681.3

UNIVERSIDADE FEDERAL DE UBERLÂNDIA
FACULDADE DE COMPUTAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Os abaixo assinados, por meio deste, certificam que leram e recomendam para a Faculdade de Computação a aceitação da dissertação intitulada "**ETArch: Projeto e Desenvolvimento de uma arquitetura para o Modelo de Título com foco na Agregação de Tráfego Multicast**" por **Maurício Amaral Gonçalves** como parte dos requisitos exigidos para a obtenção do título de **Mestre em Ciência da Computação**.

Uberlândia, 26 de Setembro de 2014

Orientador: _____

Prof. Dr. Pedro Frosi Rosa
Universidade Federal de Uberlândia

Banca Examinadora:

_____

Prof. Dr. Flávio de Oliveira Silva
Universidade Federal de Uberlândia

_____

Prof. Dr. João Henrique de Souza Pereira
Universidade Federal de Uberlândia

_____

Prof. Dr. Augusto José Venâncio Neto
Universidade Federal do Rio Grande do Norte

UNIVERSIDADE FEDERAL DE UBERLÂNDIA
FACULDADE DE COMPUTAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Data: 26 de Setembro de 2014

Autor:	**Maurício Amaral Gonçalves**
Título:	**ETArch: Projeto e Desenvolvimento de uma arquitetura para o Modelo de Título com foco na Agregação de Tráfego Multicast**
Faculdade:	**Faculdade de Computação**
Grau:	**Mestrado**

Fica garantido à Universidade Federal de Uberlândia o direito de circulação e impressão de cópias deste documento para propósitos exclusivamente acadêmicos, desde que o autor seja devidamente informado.

_____

Autor

Dedicatória

Dedico este trabalho aos meus ídolos, meu pai, Maurício da Penha Gonçalves, e minha mãe, Zilda Alves do Amaral, e ao meu mestre Pedro Frosi Rosa, meu grande incentivador.

Agradecimentos

Agradeço a Deus pela vida que me foi concedida, pelas conquistas proporcionadas, pelos dons adquiridos, pela família amorosa, e pelos melhores amigos que existem.

Agradeço à minha família: meu pai Maurício, fonte de inspiração e orgulho, minha mãe Zilda, pelo amor incondicional e apoio, minha irmã Angélica, minha grande amiga, e à minha querida companheira Valquíria, por me dar todo o carinho, amor e atenção que um homem pode querer.

Agradeço ao meu querido mestre Pedro Frosi Rosa, pela orientação em meu trabalho, e pela valorosa amizade conquistada. Agradeço pela inspiração, bons exemplos, e pela oportunidade que me foi dada.

Agradeço ao meu caro amigo Flávio Silva, pela coorientação, paciência, respeito e ajuda no desenvolvimento desse trabalho. Espero que esse seja apenas o início de uma frutífera parceria e amizade.

Agradeço ao João Henrique, pelo grande incentivo, pela compreensão como líder, e pelo belo trabalho desenvolvido, que sem dúvidas foi a maior fonte de inspiração para este trabalho.

Agradeço aos meus queridos amigos Natal Neto e Caio César, companheiros de trabalho, pesquisa e de vida. Obrigado pela inestimável ajuda e apoio. Este trabalho não seria possível sem vocês.

Agradeço aos meus irmãos de espírito e companheiros de vida: Alessandro Davi, Nathalia Azer, Alípio Neto, Caio Naves, Antônio Araújo, Daniel Rocha, Bruno Raphael, Pedro Henriques, Pablo Araújo e Lucas Alves, pela eterna amizade, companheirismo e confiança em mim depositados. Sinto-me abençoado por tê-los ao meu lado e desejo que a nossa amizade perdure pelos anos que virão.

Agradeço aos colegas de trabalho da Algar Telecom, a melhor empresa pra se trabalhar: ao Marco Antônio, Umberto Silva, Jhony Teixeira, Elias Alves, Emílio Dias, Nycholas Oliveira, Fabíola Fernandes, e Cesar William, pela amizade, compreensão, incentivo e confiança.

Agradeço aos meus companheiros de banda, e amigos de Heavy Metal: Juarez Tavora, José Maria Souto, William Cunha, professor Alex e Wagner Lamounier, pelo apoio incondicional, confiança e valorosa amizade. Que os dias de ausência sejam recompensados.

Agradeço aos meus queridos amigos da minha turma "adotiva", a 71^a turma da Faculdade de Engenharia Elétrica da UFU: Breno Barini, Luiz Eduardo Rios, Will Roger, Everton Carlos Dias, Américo Corrêa, Luiz Peixoto, Carlos Resende e Matheus Garcia, pelo apoio, amizade, companheirismo e incentivo.

Agradeço também à todos os colegas pesquisadores do grupo MEHAR. Aos meus professores da UFU, colegas de graduação e pós-graduação, e assistentes administrativos da FACOM.

"Juntos nós resistimos, divididos nós caímos...", por isso agradeço à todas as pessoas que contribuíram de alguma maneira para a minha conquista, pois sem elas nada disso seria possível.

"L'homme est condamné à être libre." *Jean-Paul Sartre - L'Être et le Néant*

Resumo

O projeto original da Internet foi iniciado há mais de quarenta anos, em um contexto totalmente diferente do atual. Nesse tempo, a rede ganhou novos propósitos e passou a ser utilizada em áreas e atividades que seriam impensáveis durante a sua concepção. As novas aplicações às quais a rede foi submetida trouxeram consigo diversos novos requisitos, que em sua maioria não foram adequadamente atendidos devido a limitações na arquitetura. Embora a especificação original da Internet tenha um importante papel na sua popularização, hoje ela atua como principal limitador de sua evolução, o que fundamenta a visão de que a arquitetura deva ser revista em uma abordagem *clean slate*. Essa estratégia incentiva a inovação nas propostas para as redes futuras, por não submetê-las às limitações da arquitetura atual, e por libertar os pesquisadores do problema de suporte à rede legada. Neste contexto, o *Modelo de Título* representa uma forma revolucionária de entender semanticamente os novos requisitos da Internet, observando também as entidades da comunicação e suas capacidades, de maneira a definir e implementar as melhores estratégias para o tratamento da comunicação. A materialização desse modelo é realizada pela *Entity Title Architecture*, uma nova e flexível arquitetura que propõe uma releitura de importantes aspectos das redes de computadores, sobretudo das estratégias de endereçamento e roteamento. Este trabalho propõe uma implementação dessa arquitetura através de um protótipo baseado na especificação OpenFlow, e de uma aplicação prática com o requisito de comunicação *multicast*. A abordagem proposta é capaz de fornecer o serviço de *multicast* de forma eficiente, e com uma solução adequada na camada de rede, o que é suportado naturalmente pela arquitetura. Neste trabalho são apresentados também os resultados de alguns experimentos comparativos, com uma aplicação de vídeo, primeiro implementado utilizando a arquitetura TCP/IP com os serviços *unicast* e *multicast*, e depois, utilizando a arquitetura *Entity Title Architecture* com foco na agregação de tráfego através de *multicast*. Os resultados demonstraram que o consumo de banda nos testes com a abordagem proposta permanece constante, enquanto na abordagem TCP/IP com serviços *unicast*, ela cresce de forma linear, proporcional ao número de clientes conectados. Já na abordagem TCP/IP com serviços *multicast*, o padrão de consumo de largura de banda é similar, no entanto, a abordagem *Entity Title Architecture* apresenta ganhos por diminuir o *overhead* desnecessário na comunicação, e dessa maneira, por utilizar uma largura de banda menor; por fornecer melhores estratégias para o plano de controle, através da separação do plano de dados; por melhorar a capacidade de endereçamento do grupo *multicast*, baseando-se na utilização de uma nova designação única, não ambígua e independente de topologia; e por fim, por apresentar uma proposta real de implantação na rede, devido ao crescente suporte ao protocolo OpenFlow, promovido pelos principais fabricantes de equipamentos.

Palavras chave: redes de telecomunicações; Internet; multicast; Internet do futuro; clean slate; modelo de entidades

Abstract

The original design of the Internet was started over forty years ago, in a totally different context of today's. At that time, the network gained new purposes and began to be used in areas and activities that would have been unthinkable during its design. New applications based on networks usage brought a new set of requirements, most of whom were not adequately met due to limitations in architecture. Although the original specification of the Internet has an important role in its popularization, today it represents the main limiter of its evolution, which fosters the thought that the architecture should be reviewed in a *clean slate* approach. This strategy encourages innovation in the proposals for future networks, by not submitting them to the limitations of the current architecture, and free researchers from the problem of supporting legacy networks. In this context, the *Entity Title Model* represents a revolutionary way to semantically understand the new Internet requirements, also managing the communication entities and their capabilities, in order to define and implement the best strategies for the treatment of communication. The materialization of this model is performed by *Entity Title Architecture*, a new flexible architecture that proposes a rereading of important aspects of computer networks, particularly in strategies for addressing and routing. This work proposes an implementation of this architecture through a prototype based on the OpenFlow specification, and a practical application with the *multicast* communication requirement. The proposed approach is able to provide the *multicast* service efficiently, and with an appropriate solution at the network layer, which is naturally supported by the architecture. Are also presented in this paper the results of some comparative experiments with a video application, first implemented using the TCP/IP architecture with *unicast* and *multicast* services, and then, using the *Entity Title Architecture*, focusing on traffic aggregation through *multicast*. The results showed that the bandwidth tests with the proposed approach remains constant, while in TCP/IP approach with *unicast* services, it grows linearly, proportional to the number of connected client. On TCP/IP approach with *multicast* services, the pattern of bandwidth consumption is similar, however, the approach *Entity Title Architecture* has won by: decreasing the unnecessary *overhead* in communication, and thus using less bandwidth; providing better strategies for the control plane, by separating the data plane; and improving the *multicast* addressability, based on the use of a unique designation, unambiguous and independent of topology; and finally, by presenting a proposal for deployment in real network, because of the Openflow broad support by leading equipment suppliers.

Keywords: telecommunications networks; Internet; multicast; future Internet; clean slate; entity title model.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas e Siglas

AAC: Advanced Audio Coding
AICT: Advanced International Conference on Telecommunications
ALM: Application Layer Multicast
AS: Autonomous Systems
ARP: Address Resolution Protocol
ARPANET: Advanced Research Projects Agency Network
API: Application Programming Interface
AVI: Audio Video Interleave
AWT: Abstract Window Toolkit
BGP: Border Gateway Protocol
CDP: Cisco Discovery Protocol
CIDR: Classless Inter-Domain Routing
CLI: Command Line Interface
CPqD: Centro de Pesquisa e Desenvolvimento em Telecomunicações
CSAP: Communication Service Access Provider
DARPA: Defense Advanced Research Projects Agency
DTMC: Discrete-Time Markov Chain
DNS: Domain Name System
DTS: Domain Title Service
DTSA: Domain Title Service Agent
DTSCP: DTS Control Protocol
DVRMP: Distance Vector Multicast Routing Protocol
EDOBRA: Extending and Deploying OFELIA in Brazil
EGP: Exterior Gateway Protocol
ESM: End System Multicast
ETArch: Entity Title Architecture
ETCP: Entity Title Control Protocol

FI: Future Internet
FIA[1]: Future Internet Architectures
FIA[2]: European Future Internet Assembly
FIB: Forwarding Information Table
FIBRE: Future Internet Testbeds Experimentation Between Brazil and Europe
FIND: Future Internet Design
FINLAN: Fast Integration of Network Layers
FIRE: Future Internet Research and Experimentation
FORCES: Forwarding and Control Element Separation
FP6: Sixth Framework Programme
FP7: Seventh Framework Programme
FP8: Eighth Framework Programme
FSM: Finite State Machine
GENI: Global Environment for Network Innovations
GoF: Gang of Four
GUID: Globally Unique Identifier
IaaS: Infrastructure as a Service
IAB: Internet Architecture Board
IANA: Internet Assigned Numbers Authority
ICANN: Internet Corporation for Assigned Names and Numbers
ICMP: Internet Control Message Protocol
ICMPv6: Internet Control Message Protocol version 6
ICN: Information-Centric Networking
ICON: IEEE International Conference on Networks
ICT: Information and Communication Technologies
IETF: Internet Engineering Task Force
IRTF: Internet Research Task Force
IGMP: Internet Group Management Protocol
IGP: Interior Gateway Protocol
IoC: Inversion of Control
IP: Internet Protocol
IPnG: IP - Next Generation
IPTV: Television over IP
IPv4: Internet Protocol version 4
IPv5: Internet Protocol version 5 - Internet Stream Protocol
IPv6: Internet Protocol version 6
ISP: Internet Service Provider
ISO: International Organization for Standardization
JNI: Java Native Interface

JPEG: Joint Photographic Experts Group
LAN: Local Area Network
LLDP: Link Layer Discovery Protocol
LNCS: Lecture Notes in Computer Science
MAC: Medium Access Control
MAN: Metropolitan Area Network
MDTSA: Master Domain Title Service Agent
MEHAR: Mondial Entities Horizontally Addressed by Requirements
MIB: Management Information Base
MJPEG: Motion JPEG
MOSPF: Multicast Open Shortest Path First
MPEG-2: Moving Picture Experts Group - Version 2
MPLS: Multi-Protocol Label Switching
NAT: Network Address Translation
NE: Network Element
NEC: Network Element Connector
NFV: Network Functions Virtualization
NLR: National LambdaRail
NSF: National Science Foundation
OFELIA: OpenFlow in Europe: Linking Infrastructure and Applications
OFP: OpenFlow Protocol
OSI: Open Systems Interconnection
OSPF: Open Shortest Path First
OWL: Web Ontology Language
P2P: Peer to Peer
PaaS: Platform as a Service
PAN: Personal Area Network
PDU: Protocol Data Unit
PIM: Protocol Independent Multicast
PIM-DM: Protocol Independent Multicast - Dense-Mode
PIM-SM: Protocol Independent Multicast - Sparse-Mode
PoA: Point of Attachment
PoS: Point of Service
QoE: Quality of Experience
QoS: Quality of Service
RAM: Requirement Analysis Module
RCP: Routing Control Platform
REST: Representational State Transfer
RFC: Request for Comments

RIB: Routing Information Base
RIP: Routing Information Protocol
RIR: Regional Internet Registries
RMM: Requirement Manager Module
RNP: Rede Nacional de Pesquisa
RTP: Real-time Transport Protocol
SaaS: Software as a Service
SAP: Service Access Point
SBRC: Simpósio Brasileiro de Redes de Computadores
SCTP: Stream Control Transmission Protocol
SDN: Software Defined Networking
SIP: Simple IP
SIPP: Simple IP Plus
SNMP: Simple Network Management Protocol
SOAP: Simple Object Access Protocol
SoC: Separation of Concerns
SRI: Stanford Research Institute
SSH: Secure Shell
ST: Internet Stream Protocol
TCP: Transport Control Protocol
TMR: Triple Modular Redundancy
TSAP: Transport Service Access Provider
UDP: User Datagram Protocol
VM: Virtual Machine
VoD: Video on Demand
VoIP: Voice over IP
WAN: Wide Area Network
WLAN: Wireless Local Area Network
WPEIF: Workshop de Pesquisa Experimental da Internet do Futuro

Capítulo 1

Introdução

Os princípios básicos da Internet foram concebidos na década de sessenta [Baran 1964], e o seu desenvolvimento baseou-se numa pesquisa experimental utilizando a ARPANET, uma rede militar desenvolvida pela *Defense Advanced Research Projects Agency* (DARPA) durante a guerra fria. O modelo de referência da arquitetura Internet baseou-se no Modelo OSI (*Open Systems Interconnection*), que foi criado e aprimorado cerca de uma década depois [Padlipsky 1982], e tratou o problema da comunicação de dados por meio de camadas ortogonais, concebidas com princípios bem definidos e importância equivalente. Os principais protocolos componentes dessa arquitetura foram propostos ainda nos anos setenta [Cerf and Kahn 1974], obedecendo aos objetivos definidos pelo modelo, e com enfoque nos requisitos básicos da época.

Nesse período, a camada física passou por uma evolução extraordinária, impulsionada por processadores cada vez mais rápidos, conforme preconizado pela Lei de Moore [Moore 2006]. Este fator foi determinante para a evolução das redes, pois possibilitou ampliação na largura de banda e na capacidade de vazão dos enlaces a baixo custo. A malha de elementos de rede da Internet se estendeu por todo o mundo e o número de *hosts* interconectados aumentou rapidamente. Foram criadas novas tecnologias de transmissão de dados, tanto *wired* quanto *wireless*, bem como foram introduzidos muitos protocolos da camada física.

Por sua vez, a camada de Aplicação, impulsionada pelos avanços tecnológicos do meio físico e a crescente abrangência da Internet na sociedade moderna, também experimentou grandes mudanças. O contexto de uso da rede, bem como o padrão de tráfego de dados, foi completamente alterado, devido à concepção de novas aplicações para os mais diversos fins, onde destacam-se as de transmissões *multimídia*, tais como VoIP, IPTV, VoD, *Live*

Streaming e Vídeo conferência; de compartilhamento de arquivos, através de *Torrent*, *File Sharing* e *P2P*; e de processamento nas nuvens, com estratégias como SaaS, PaaS, IaaS dentre outras.

Toda as evoluções nas camadas física e de aplicação impuseram sobre a Internet mudanças drásticas nos requisitos definidos em sua concepção. Há quatro décadas, eles basicamente se referiam a aspectos de confiabilidade e de baixo *overhead* na troca de mensagens, enquanto hoje contemplam necessidades bem mais complexas, tais como: QoS, QoE, mobilidade, segurança, *multicast* e *real-time*. Esses requisitos não foram devidamente endereçados pelas camadas de rede e transporte, em parte pela inexpressividade semântica das interfaces definidas pelo modelo Internet [Pereira 2012], o que tornou-o insensível às reais necessidades do usuário; pela maneira pouco flexível como esses protocolos foram projetados, o que dificulta a experimentação de novas abordagens; e finalmente, por interesses comerciais de fornecedores de equipamentos e serviços, o que acabou sobrepondo-se aos objetivos científicos, contribuindo para a desaceleração da evolução da Internet [Comer 2000].

Em geral, os novos requisitos foram trabalhados na camada de aplicação, e não nas inferiores, o que causou uma disfunção nos princípios filosóficos definidos pelo Modelo OSI. Um exemplo disso é o protocolo RTP [Group et al. 1996], que adiciona controles de fluxo intrínsecos à camada de transporte, ao protocolo UDP [Postel 1980c], com uma implementação no nível de aplicação. No outro extremo, verificou-se um movimento de especialização do *hardware* nas primitivas dos protocolos de rede e transporte, o que contribuiu para o engessamento das redes atuais, e agravamento do problema de suporte às redes legadas.

Por todas essas razões, a Internet enfrenta grandes desafios para sua evolução [Zahariadis et al. 2011], algo vislumbrado desde a década de noventa [Clark et al. 1991], quando a Internet começou a enfrentar sérios problemas relativos ao esgotamento da capacidade de endereçamento [Egevang and Francis 1994, Group and Hinden 1993], ou mesmo antes, quando percebida a necessidade de um identificador com maior expressividade semântica [Postel 1983, Mockapetris 1983a, Mockapetris 1983b]. Outro ponto motivador é a grande complexidade dos planos de dados [Rekhter and Li 1995], arquitetura, e plano de controle [Aguiar 2008], que impulsionados pelo crescimento acelerado da Internet, tornaram a operação das redes uma atividade onerosa e de alto risco.

Nesse contexto, pesquisadores do mundo todo têm trabalhado em duas abordagens [Rexford and Dovrolis 2010]: a "revolucionária" ou *clean slate* [Roberts 2009], que não se limita à especificação da arquitetura atual e possibilita a concepção de novas ideias; e a "evolucionária", que visa evoluir a arquitetura atual, mantendo os seus princípios básicos.

É consenso que qualquer que seja o novo modelo de Internet aceito, ele deva ser flexível, de forma a suportar as mudanças de requisitos das próximas gerações de aplicações. Uma abordagem que vem tomando força nos últimos anos é a da SDN (*Software Defined*

Network) [Goth 2011, Greene 2009], que cria uma camada de abstração para o controle dos elementos de rede, promovendo uma separação entre os planos de dados e de controle. Essa ortogonalidade do plano de controle permite que as redes possam ser programadas de acordo com um comportamento específico, o que abre uma ampla margem de possibilidades para as redes atuais e futuras. Essa camada programável está para as redes, assim como o sistema operacional está para os computadores.

O *Modelo de Título* [Pereira et al. 2011] propõe uma abordagem *clean slate* para a Internet do futuro, que visa solucionar os problemas de endereçamento e agregação de tráfego via *multicast* através de *títulos*, que são identificadores de *entidades*, únicos, não ambíguos e independentes da topologia. Entende-se por *entidade* tudo o que se comunica e que pode ser endereçado em uma rede, isto é, desde aplicações até elementos de rede, sensores, etc. No *Modelo de Título* é definida uma nova camada de Comunicação, que na arquitetura proposta define dois domínios principais da comunicação: Net-Ontology [de Oliveira Silva et al. 2012a] e DL-Ontology.

A sub-camada Net-Ontology é responsável por interpretar semanticamente as necessidades das *entidades* usuárias da camada de Comunicação (Aplicações), e por viabilizá-las eficientemente. Esse suporte aos requisitos da aplicação ocorre através de módulos responsáveis por tratar as primitivas da rede, que implementam algoritmos específicos para o atendimento de cada requisito. A sub-camada DL-Ontology, por sua vez, é responsável pela transmissão das primitivas de comunicação através do conceito de *workspace*, que é essencialmente um barramento lógico, que se constitui um grupo *multicast*, com requisitos bem definidos, e naturalmente suportado pela arquitetura. A definição dos requisitos da comunicação se dá por meio de uma descrição em alto nível, através de uma ontologia baseada nas redes atuais, descrita em OWL [Lacy 2005]. Essa comunicação com alta expressividade semântica, diminui o acoplamento entre as camadas; elimina o *overhead* de funções sobrepostas; torna a definição de novos serviços mais flexível; e garante que a rede "entenda" corretamente as necessidades das *entidades*.

O plano de controle é tratado por um sistema distribuído, que é um super conjunto do controlador utilizado na abordagem SDN, chamado DTS (*Domain Title Service*). Este sistema é responsável por gerir o ciclo de vida de *entidades* e *workspaces*, "entendendo" semanticamente os seus *requisitos* e *capacidades*; por armazenar informações e estatísticas da rede; e por estabelecer a comunicação entre *entidades* através de configurações nos elementos de rede.

1.1 Hipótese

Acredita-se possível a utilização do endereçamento horizontal proposto no *Modelo de Título* para a agregação de tráfego via *multicast* de forma real. Assim, acredita-se também que tal modelo seja implementável, e que a aproximação semântica proposta seja capaz de

proporcionar maior flexibilidade para as redes de computadores, garantindo um melhor entendimento de requisitos de aplicações atuais e futuras.

1.2 Justificativa e Motivação

Ao observar que a arquitetura atual é um fator limitante para a evolução da Internet, percebe-se que a abordagem *clean slate* é a única capaz de tornar a nova Internet flexível aos requisitos atuais e futuros, bem como endereçar todos os problemas identificados nas últimas décadas.

Para a resolução de comunicação *multicast*, foco do trabalho relatado desta dissertação, é necessária uma reformulação no endereçamento, que é em geral a maior restrição desse requisito (*multicast*). Entende-se como necessária uma separação entre identificação e endereçamento, por compreenderem aspectos de naturezas completamente distintas.

A distância semântica, entre a aplicação e a rede, representa um dos principais motivos para a inabilidade da rede em atender aos novos requisitos. Basicamente, a pilha de protocolos não é capaz de se adaptar aos requisitos da aplicação, o que gera um "*gap*", entre necessidade e funcionalidade, e obriga a camada superior a definir explicitamente as instâncias das camadas inferiores que mais se aproximam de sua necessidade.

1.3 Objetivo

O objetivo geral deste trabalho é implementar e avaliar os componentes da *Entity Title Architecture* (ETArch), para proporcionar uma solução adequada ao requisito de comunicação *multicast*, baseada no conceito de *workspace*, que separa os aspectos de endereçamento e identificação, oferecendo suporte natural de comunicação a grupos de *entidades*.

Para tal, os seguintes objetivos específicos devem ser alcançados:

- Validar o *Modelo de Título* e detalhar os seus componentes, protocolos, algoritmos e estratégias de implementação, com enfoque no requisito de comunicação *multicast*;
- Implementar a especificação da DL-Ontology através de uma API (*Application Layer Interface*) de baixo nível, com acesso direto à placa de rede (em linguagem C), e outra API de alto nível, para a simplificação do desenvolvimento de aplicações;
- Implementar uma versão inicial da especificação do DTS com suporte ao protocolo *OpenFlow*, e com pelo menos um agente controlador da rede ativo;
- Implementar bibliotecas do plano de controle e os protocolos definidos pela arquitetura: *Entity Title Control Protocol* (ETCP) e *Domain Title Service Control Protocol* (DTSCP);

- Descrever cenários que se beneficiem da proposta de *multicast* apresentada por esse trabalho; e,
- Implementar uma aplicação que atenda ao cenário de *streaming* de vídeo sobre multicast, de maneira a demonstrar a aplicabilidade da proposta.

1.4 Resultados Esperados

Espera-se que esse trabalho forneça base experimental para uma maior adoção da *Entity Title Architecture* em trabalhos relacionados às redes futuras, justificando-se a partir da comprovação da relevância dos resultados apresentados. Assim, espera-se que este trabalho contribua para a definição do modelo de referência da Internet do futuro, sendo um ponto chave para a redução da complexidade dos protocolos, obtido através da aproximação semântica entre as camadas de rede.

1.5 Escopo e Limitações do Trabalho

O escopo deste trabalho é o de validar o *Modelo de Título* e apresentá-lo através de uma implementação inicial dos componentes da *ETArch* na forma de uma prova de conceito. Também faz parte do escopo explorar o requisito de *multicast*, identificando problemas e soluções para as questões que inviabilizam a sua adoção no modelo atual; resolvê-las com *Modelo de Título* através da separação entre localização e identificação; e por fim, implementar uma aplicação funcional que se beneficie do *multicast* apresentado, demonstrando assim a aplicabilidade da abordagem e colhendo resultados para análise comparativa.

Não é objetivo desse trabalho o desenvolvimento final de nenhum dos módulos componentes da ETArch, bem como não são garantidas escalabilidade e alta disponibilidade requeridas para aplicações comerciais.

Também não é escopo dessa dissertação apresentar os avanços atuais em linhas paralelas do grupo de pesquisa, de maneira a não se afastar dos objetivos definidos para o trabalho. No entanto tais avanços podem ser conferidos em recentes publicações, como: [Oliveira Silva et al. 2013, Silva et al. 2014a, Silva et al. 2014b]

Não é escopo desse projeto a interpretação semântica das necessidades das *entidades*, e nem o aprovisionamento automático das capacidades da rede que as atendam, sendo assumido que o grupo de *multicast* é homogêneo, e não há necessidades fundamentais para o estabelecimento da comunicação dentro do grupo.

Por fim, é objetivo do projeto validar a *ETArch*, posicionando-a como uma arquitetura à ser utilizada na Internet do Futuro, analisando sobretudo aspectos de implantação, como a coexistência com outras abordagens e redes legada, e suas limitações e aplicações nas

redes atuais e futuras.

1.6 Organização

Essa dissertação está organizada da seguinte forma:

- Capítulo 1 - *Introdução*

  Introduz o assunto e apresenta hipótese, justificativa, objetivo, motivação, resultados esperados, escopo, limitações e organização desse trabalho.

- Capítulo 2 - *Estado da Arte*

  Apresenta um breve histórico sobre a Internet; discorre sobre a sua evolução; mostra conceitos empregados nas redes atuais; apresenta os trabalhos das últimas décadas voltados para resolução de problemas da arquitetura atual; e descreve os trabalhos relacionados atuais para a Internet do futuro.

- Capítulo 3 - *Detalhamento da Proposta*

  Apresenta a especificação *Entity Title Architecture* (ETArch) definindo: conceitos, camadas, protocolos, sistemas, algoritmos, técnicas e abordagens utilizadas, bem como aspectos das redes e das *entidades*.

- Capítulo 4 - *Implementação*

  Descreve uma implementação inicial para a arquitetura ETArch, detalhando todas as aplicações dos componentes implementados, bem como algoritmos, técnicas e tecnologias empregadas. Também apresenta a implementação de uma aplicação de *streaming* de vídeo, que utiliza a implementação anterior de forma à colher os benefícios da agregação de tráfego via *multicast*.

- Capítulo 5 - *Resultados Obtidos*

  Apresenta as contribuições dessa dissertação, aplicações em cenários reais, e resultados de testes comparativos preliminares.

- Capítulo 6 - *Conclusão e Trabalhos Futuros*

  Conclui o trabalho, ressalta as suas contribuições e apresenta as perspectivas futuras.

Capítulo 2

Aspectos Conceituais e Contextualização da Pesquisa

Acredita-se que para a concepção de uma proposta viável para as redes futuras, faz-se necessário um profundo conhecimento da história da Internet até os dias atuais, de maneira a obter uma visão clara do *estado da arte* das redes de computadores. O posicionamento da abordagem proposta ante a trabalhos correlatos, mostra-se benéfico ao amadurecimento da pesquisa, uma vez que permite avaliar os ganhos e detectar problemas, gerando assim oportunidades de melhoria.

É importante lembrar que mesmo com o crescente movimento de novas propostas na área de FI (*Future Internet*), ainda não há uma perspectiva clara do rumo que a Arquitetura da Nova Internet tomará nos próximos anos. Tal movimento ganhou força na década de 1990, com a iminência do esgotamento da capacidade de endereçamento do IPv4 [Postel 1981b], e com a criação de novos tipos de conteúdo para Internet, o que demandou também novos requisitos de comunicação, tais como *multicast*, segurança, QoS (*Quality of Service*), QoE (*Quality of Experience*), mobilidade e *multihoming*; no entanto, diversas questões permanecem abertas, e a arquitetura de rede continua praticamente inalterada.

Abordagens recentes, como a ETArch (proposta neste trabalho), SDN e NFV (*Network Functions Virtualization*) [Cui et al. 2012], têm aquecido a comunidade científica com importantes trabalhos propondo uma reformulação da arquitetura Internet. Entretanto, essas propostas encontram-se em estágios iniciais, o que inviabiliza as suas adoções à curto prazo.

Nas próximas seções será apresentado um levantamento histórico sobre as redes de

computadores, sobressaltando os principais avanços nos últimos cinquenta anos, e as mudanças de contexto às quais a Internet foi submetida. Além disso, são explorados aspectos conceituais, envolvendo a formalização de termos, e a explanação de *aspectos-chave* para a comunicação com o requisito de *multicast*. Por fim, são apresentados diversos trabalhos correlatos no Brasil e no mundo, e como eles se relacionam com a abordagem apresentada neste trabalho.

2.1 Breve Histórico

A DARPA (*Defense Advanced Research Projects Agency*) é uma agência fundada em 1958, por pesquisadores militares americanos, com o objetivo de garantir a superioridade tecnológica dos Estados Unidos. Em 1969 a agência criou a ARPANET, uma rede precursora da Internet que forneceu uma maneira mais segura e resiliente para troca de dados. Logo no início da década 1970, a rede começou a ser compartilhada entre as universidades americanas, e em 1972 a rede já começava a se expandir para fora do país. Em 1974, na RFC 675 (*Specification of Internet Transmission Control Program*) [Cerf et al. 1974], Vinton Cerf, Yogen Dalal, e Carl Sunshine utilizaram pela primeira vez o termo Internet, como uma abreviação para "*Internetworking*". Em 1981 foram apresentados os principais protocolos, componentes da arquitetura que nascia, publicados nas RFC 791, 792, 793 (IP, ICMP e TCP) [Postel 1981b, Postel 1981a, Postel 1981c]. Em 1982 a arquitetura TCP/IP foi formalizada, e foi introduzido o conceito de Internet como uma rede mundial.

Na década de 1980, a Internet experimentou um rápido crescimento, o que motivou a criação do IAB (*Internet Architecture Board*) em 1983, que é responsável por manter a arquitetura da Internet, e gerenciar, dentre outros grupos, o IETF (*Internet Engineering Task Force*), responsável pelo estabelecimento dos padrões da Internet, e o IRTF (*Internet Research Task Force*), cuja função é promover a pesquisa para a evolução da Internet. Nessa mesma época foi criado o IANA (*Internet Assigned Numbers Authority*), que controla a numeração dos protocolos, coordena o DNS Root e o endereçamento IP, e provê a designação de números para o RIR (*Regional Internet Registries).*

Em 1987, foram publicadas as RFC 1034 e 1035, que descrevem o sistema e o protocolo para a resolução de nomes de domínios, o DNS (*Domain Name System*) [Mockapetris 1987a, Mockapetris 1987b]. A necessidade de uma identificação com expressividade semântica, única e não ambígua, já havia sido observada desde a ARPANET, quando a SRI (*Stanford Research Institute*) começou a trabalhar em um mecanismo de mapeamento de nomes baseado em configurações, que utilizava um arquivo chamado "hosts.txt", o que ainda existe nos atuais sistemas UNIX com o nome de "hosts". Com o crescimento da rede, manter esse arquivos atualizados se tornou uma tarefa inviável, o que levou Paul Mockapetris a criar em 1983 um esboço do DNS, publicado nas RFC 882 e 883 [Mockapetris 1983a, Mockapetris 1983a], e consolidado anos mais tarde.

Em 1988 foi especificado o RIP (*Routing Information Protocol*) na RFC 1058 [Hedrick 1988], com um algoritmo baseado em vetores de distância para a identificação das melhores rotas entre dois nós, através de uma comparação matemática. Em 1989 surgiu o OSPF (*Open Shortest Path First*), definido na RFC 1131 [Moy 1989], que foi uma evolução natural do RIP, através do emprego de técnicas de computação de grafos. Ao invés de manter uma tabela de rotas, no OSPF, cada nó mantém dados sobre todos os links da rede, compondo assim uma abstração de um grafo, o que torna possível o cálculo do menor caminho de forma independente, através de um algoritmo baseado no *Dijkstra*. Tanto o RIP quanto o OSPF são considerados protocolos IGP (*Interior Gateway Protocol*), por trabalharem no mesmo AS (Autonomous System), um grupo de redes IP que é gerenciada por um ou mais operadores de rede, com uma política única de roteamento. Para a interconexão de ASs foi concebido o EGP (*Exterior Gateway Protocol*), descrito inicialmente em 1982 na RFC 827 [Rosen 1982], e que já trazia uma abordagem baseada em vetor de distâncias, assim como o seu sucessor, o BGP (*Border Gateway Protocol*), proposto inicialmente em 1989 na RFC 1105 [Lougheed and Rekhter 1989], e amplamente utilizado ainda nos dias de hoje. O BGP apresentou ganhos em relação ao EGP por possibilitar um maior controle das políticas de roteamento por AS, além de corrigir problemas de *loops* e apresentar melhoria de desempenho no algoritmo.

Na década de 1990, o IETF começou à trabalhar com projetos de protocolos para a substituição do IPv4, dentre eles o SIP (*Simple IP*), e a sua extensão, o SIPP (*Simple IP Plus*) [Hinden 1994], que propôs uma simplificação na primitiva de rede, ampliação do tamanho do campo de endereçamento, de 32 para 64 bits, e novas abordagens de roteamento, como por exemplo a utilização de *cluster addresses* (mapeamento de regiões ao invés de nós isolados), e de *scopes* (endereçamento *multicast*). Em 1995, foi proposto o IPv6 na RFC 1883 [Deering and Hinden 1995], que se consolidou como proposta oficial de IPnG (*IP - Next Generation*) do IETF, e cujas ideias foram influenciadas pela abordagem do SIPP. O IPv5 foi designado para o protocolo ST (*Internet Stream Protocol*), que foi inicialmente proposto em 1979 na IEN 119 [Forgie 1979], e posteriormente nas RFCs 1190 de 1990 e 1819 de 1995 [Delgrossi and Berger 1995, Topolcic 1990]. O ST foi um protocolo experimental, voltado para transmissões de áudio e vídeo, que nunca foi introduzido ao público como uma versão oficial do IP, mas que acabou influenciando o modelo atual, como referência para o protocolo MPLS (*Multi-Protocol Label Switching*).

O IPv6 propôs diversas evoluções em relação ao IPv4, tais como: ampliação da capacidade de endereçamento, com espaço de endereços de até 128 bits; aprimoramentos no cabeçalho, através da inclusão de novos campos opcionais e formatação mais flexível; e suporte para alocação de recursos. No entanto, a adesão ao IPv6 se mostrou lenta, dada a dificuldade de adaptação à rede legada. Paralelamente à pesquisa de IPnG foi especificado em 1993, na RFC 1519, o CIDR (*Classless Inter-Domain Routing*) [Fuller et al. 1993], que forneceu uma nova forma de endereçamento de redes e nós, com a eliminação

das classes de endereçamento definidas no modelo anterior. Esse método trouxe ganhos tanto no roteamento, através da agregação de prefixos, quanto no endereçamento, com um melhor aproveitamento da gama de endereços disponíveis. Em 1996, na RFC 1918, foi especificado o NAT (*Network Address Translator*) [Rekhter et al. 1996], que define um mecanismo para mapeamento de endereços "não-válidos" de uma rede interna em endereços "válidos" de uma rede externa, por meio de uma tabela *hash* e reescrita das primitivas de rede. O CIDR e o NAT juntos prolongaram a vida do IPv4, por resolverem parcialmente o problema de limitação no endereçamento IP, e dessa maneira, contribuíram para o atraso do movimento de migração para o IPv6.

Em 1996 foi especificado na RFC 1889 o protocolo RTP (*Real Time Protocol*), que visa o suporte para transmissões de dados em tempo real. Esse protocolo foi implementado na camada de aplicação, sobre UDP, e adiciona controles de fluxo e de temporização ao protocolo de transporte. RTP é voltado ao atendimento do requisito de *tempo real*, e por isso é amplamente utilizado em aplicações de *streaming* de áudio e vídeo. Evidencia-se que o uso de um protocolo de transporte na camada de aplicação traz um *overhead* desnecessário, comprometendo o desempenho da aplicação, desperdiçando recursos, além da violação de aspectos filosóficos e arquiteturais. Entretanto, o modelo atual é pouco flexível, o que dificulta a extensão desses protocolos. Um bom exemplo desse cenário é o protocolo SCTP (*Stream Control Transmission Protocol*), especificado na RFC 2960 de 2000 [Stewart et al. 2000]. Trata-se de um novo protocolo de transporte voltado para tratamento de fluxos de mídia, e que mantém as funcionalidades do UDP (transmissão orientada a mensagem), e TCP (controle de sequência e congestionamento), mas que não é amplamente utilizado ainda hoje, devido à necessidade de suporte pelas redes e aplicações legadas.

2.2 Evolução da Internet

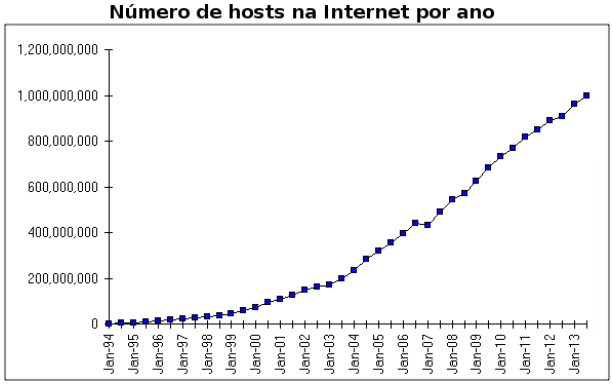
Nos 40 anos subsequentes ao nascimento da Internet, houve pouca evolução nos aspectos arquiteturais das redes, principalmente nas camadas de rede e transporte. Em contrapartida houve um considerável aumento na complexidade da rede, que se mostrou incapaz de atender aos novos requisitos e se adaptar às novas proporções da rede.

A Figura 2.1a apresenta a taxa de crescimento do número de *hosts* na rede nos últimos dez anos. Nela pode ser observado que a Internet passou a interconectar mais de um bilhão de *hosts* só na última década, e esse número tende a aumentar. Parte desse aumento se deve à criação de novos tipos de dispositivos, redução dos custos, melhorias nas redes, investimentos do governo e das empresas de telecomunicações, e pelo comportamento viral da Internet.

Os padrões de utilização das redes foram completamente alterados, o que impulsionou uma crescente demanda por novas funcionalidades de rede, necessárias para o atendimento

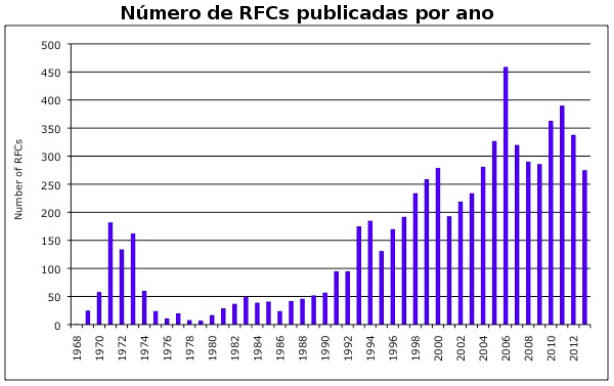
de novos requisitos, tais como mobilidade, segurança, QoS e comunicação *multicast*, que não foram adequadamente resolvidos por limitações na arquitetura atual. Estes requisitos muitas vezes requerem soluções complexas, com a combinação de diferentes funcionalidades, o que se agrava em cenários de múltiplos requisitos, especialmente quando estes apresentam características muito divergentes.

Devido à adoção em larga escala da arquitetura TCP/IP, para a qual existe uma base de software sem precedentes, alterar as camadas de rede e transporte se mostrou uma tarefa quase impossível. Embora alguns avanços tenham sido realizados como mostra a Figura 2.2, eles representam uma pequena parcela das especificações publicadas nos últimos quarenta anos. A maioria das RFCs apresentadas na Figura 2.1b se concentraram na camada de aplicação, o que nos trouxe ao cenário da Internet atual, composta por um emaranhado de protocolos. Muitos desses protocolos, como é o caso do protocolo RTP, deveriam estar posicionados na camada de transporte, no entanto, por limitações arquiteturais, eles foram implementados nas camadas superiores, o que gera um acréscimo de *overhead* na comunicação e perda de desempenho.



Fonte: https://www.isc.org/services/survey

(a) Crescimento do número de hosts.

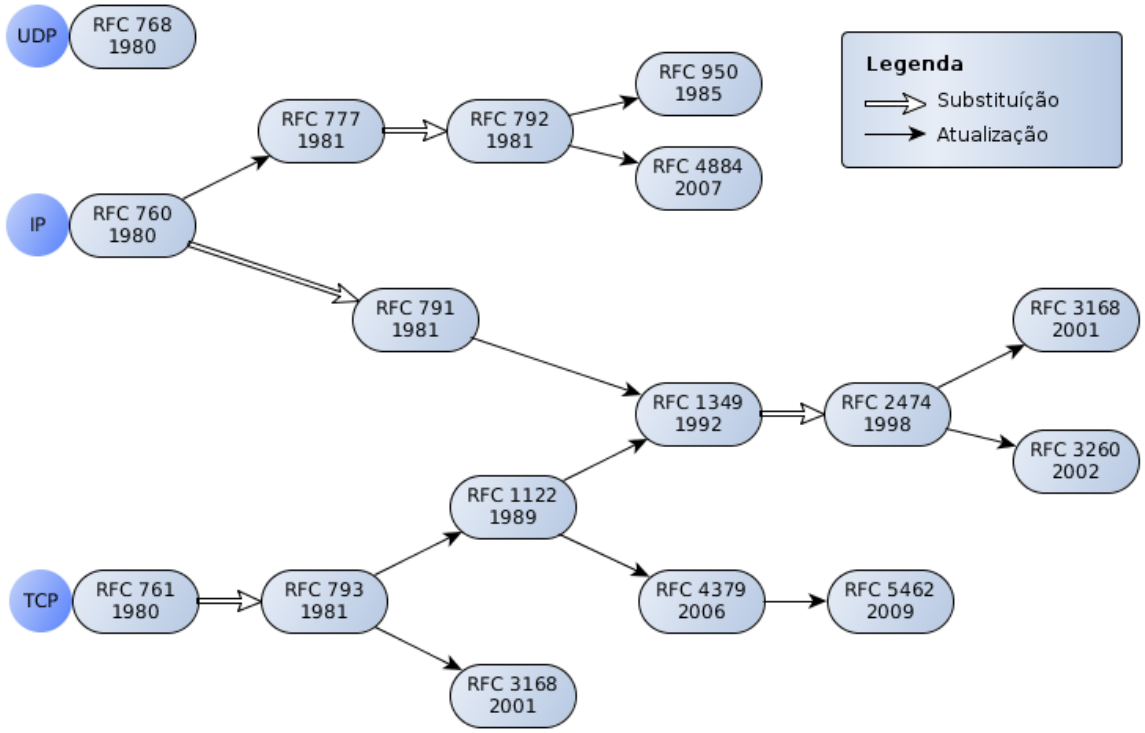


Fonte: http://www.rfc-editor.org/num_rfc_year.html

(b) Aumento do número de RFCs publicadas.

Figura 2.1: Relação entre crescimento da rede e demanda de alterações na arquitetura.

A evolução dos protocolos centrais da arquitetura TCP/IP é evidenciada pela Figura 2.2, que apresenta uma visão resumida das RFC 760 (IP), 761 (TCP) e 768 (UDP) [Postel 1980a, Postel 1980b, Postel 1980c]. Pode ser observado que a especificação do protocolo UDP não sofreu alterações desde a sua concepção em 1980, mesmo sendo amplamente utilizada nos dias de hoje. As maiores alterações das especificações dos protocolos IP e TCP ocorreram em 1981 e foram consolidadas nas RFCs 791 e 793 [Postel 1981b, Postel 1981c]. Depois disso houve pequenas alterações relacionadas à definição e utilização de campos do cabeçalho, mas nada substancial para resolução das demandas crescentes por novas funcionalidades.



Inspirada em [Pereira 2012]

Figura 2.2: A evolução das RFC 760, 761 e 768.

2.3 Multicast na Arquitetura TCP/IP

Com a popularização das mídias digitais e ampliação da capacidade de vazão dos enlaces, a comunicação *multicast* se tornou um dos principais requisitos a serem atendidos nas redes atuais e futuras. Os primeiros relatos de estudos no atendimento deste requisito datam de meados dos anos oitenta e foram formalizados inicialmente pela RFC 966 [Deering and Cheriton 1985], que conceituou o termo *multicast* como a transmissão de um único datagrama para um grupo de zero ou mais *hosts*, identificados por um endereço único. Nesse trabalho, *Deering* propôs uma extensão do protocolo IP para a provisão do serviço de *multicast*, baseado no modelo de datagrama e mecanismos de transmissão do *unicast*.

Para o gerenciamento dos grupos *multicast* foi definido o protocolo IGMP (*Internet Group Management Protocol*), descrito no *Apêndice I* da RFC 966 [Deering and Cheriton 1985]. O IGMP apresenta diversas similaridades com o ICMP (*Internet Control Message Protocol*), inclusive na formatação de mensagens, que são encapsuladas em pacotes IP. Essa semelhança foi proposital, uma vez que os autores vislumbravam a possibilidade de unificação desses protocolos nas implementações futuras. A evolução da proposta ocorreu através das RFC 988 [Deering 1986], RFC 1054 [Deering 1988] e RFC 1112 [Simpson 1996].

Similarmente ao serviço de IP *Unicast*, o IP *Multicast* não é orientado a conexão, e

define uma estratégia de envio de mensagens num esquema de melhor esforço (*best effort*). A diferenciação se dá através da classe de endereçamento do IP de destino, que no caso do *multicast* se restringe à classe D, que compreende o grupo de endereços entre 224.0.0.0 e 239.255.255.255. Nesta estratégia, *hosts* interessados em algum conteúdo devem se associar à um grupo *multicast*, notificando os elementos de rede através do protocolo IGMP.

No protocolo IPv6, a fatia reservada para o serviço de *multicast* é a compreendida pelo prefixo ff::0/8. O conceito de endereços de *broadcast* foi substituído pelo de endereços *multicast* [Hinden and Deering 1998]. Além disso, as interfaces de rede tornaram-se capazes de se atachar à diferentes grupos *multicast*. A arquitetura proposta também fornece alocação dinâmica de endereços IP [Thaler et al. 2000], que podem ser definidos em diferentes *scopes* [Hinden and Deering 2006].

Os três principais protocolos de roteamento de tráfego *multicast* são o DVRMP (*Distance Vector Multicast Routing Protocol*) [Waitzman et al. 1988], MOSPF (*Multicast Open Shortest Path First*) [Moy 1994], e o PIM (*Protocol Independent Multicast*) [Estrin et al. 1997, Adams et al. 2005]. O DVRMP utiliza uma estratégia de cálculo a partir de um vetor de distâncias, assim como o *Routing Information Protocol* (RIP) [Hedrick 1988]. Ele é o mais simples dentre os protocolos, mas apresenta limitações quando aplicado a redes de larga escala. Da mesma maneira, o MOSPF foi baseado no OSPF, que analisa a rede como um grafo, e realiza cálculos utilizando o estado do enlace (*link-state*) por meio do algoritmo baseado em *Dijkstra*. Quanto menos densa a rede, melhor é o desempenho do algoritmo empregado pelo OSPF, por isso este protocolo é tradicionalmente evitado em redes com essas características de enlace. O PIM utiliza uma estratégia de roteamento que não depende de um protocolo para descoberta (*discovery*) de topologia, por isso ele é considerado como um mecanismo *protocol-independent*. Ao invés disso, o PIM utiliza outros protocolos de roteamento como o próprio OSPF e RIP. Há duas versões de PIM: uma voltada para ambientes esparsos, o PIM-SM; e uma voltada para ambientes densos, o PIM-DM. As diferenças entre ambas as versões estão nas estratégias de roteamento, que são otimizadas para ambientes de alta ou baixa densidade.

O *multicast* baseado em IPv4 apresenta limitações, tanto em aspectos técnicos quanto de negócios [Diot et al. 2000], tais como: número limitado de endereços *multicast*; a incapacidade da gestão dos grupos de forma dinâmica; restrições de segurança; arquitetura complexa e dificuldades na implantação.

O *multicast* baseado em IPv6 apresenta desafios no requisito de segurança [Davies et al. 2007], com vulnerabilidades que podem ser exploradas por ataques. Além disso, cenários com requisitos de mobilidade, onde os usuários compartilham canais com largura de banda limitada, apresentam uma série de desafios [Romdhani et al. 2004], dada a combinação desses dois requisitos.

Devido a essas limitações, a implantação do IP *Multicast* ocorre lentamente [Yiu and

Chan 2008], o que promoveu a adoção de *Application Layer Multicast* (ALM) [Hosseini et al. 2007], também conhecido como *End System Multicast* (ESM), na qual a maioria dos problemas de *multicast* sobre IP são abordados na camada de aplicação, o que facilita sua adoção por não implicar mudanças na arquitetura de rede.

A capacidade de implantar facilmente o protocolo ALM é uma grande vantagem em relação ao IP *Multicast*. Por outro lado, o IP *Multicast* oferece uma melhor utilização de largura de banda utilizada na comunicação, que é parcialmente desperdiçada no ALM, devido à sua estratégia de *multicast*, que se baseia na replicação do pacote sobre as árvores de distribuição [Hosseini et al. 2007]. Além disso, mesmo usando ALM, questões como a mobilidade apresentam vários desafios relacionados às limitações impostas pela arquitetura TCP/IP.

2.4 Termos e conceitos da Internet

Este trabalho almeja a implementação de uma nova arquitetura para a Internet, capaz de suportar os requisitos atuais e ser flexível aos futuros, corrigindo aspectos intrínsecos à atual, sobretudo nas estratégias de endereçamento e roteamento, bem como validá-la através de um novo conceito de comunicação *multicast*, que é atendido de forma natural pela arquitetura. Para alcançar tal objetivo, é necessário repensar estratégias criadas e adotadas há anos, o que demanda uma quebra dos paradigmas atuais. Dada a natureza profunda dessas mudanças, entende-se que é necessário o entendimento adequado dos termos e conceitos ligados à Internet, tais como: arquitetura, modelo, comunicação, endereçamento, roteamento, semântica, ontologia etc.

2.4.1 Arquitetura

O termo *arquitetura* vem do grego *arkhé* e *tékhton*, que significa respectivamente *principal* e *construção*. Embora já tenha sido empregado nos mais diversos ramos, como: matemática, política, filosofia e história, o significado de *arquitetura*, enquanto atividade, remete ao ato de projetar algo, de maneira a posteriormente torná-lo real, o que deu origem ao verbo *arquitetar*. Ao resultado da ação de *arquitetar* também dá-se o nome de *arquitetura*, como um sinônimo de projeto. A expressão "*Arquitetura TCP/IP*" utilizada neste trabalho se refere ao projeto original da Internet que guiou a sua construção.

2.4.2 Modelo

O termo *modelo* vem do latim *modus*, que significa *modo*, *maneira*. Assim, o *modelo* pode ser encarado como uma definição de como algo deve ser. O termo *modelo* também é empregado como sinônimo de *exemplo*, que em geral é uma representação mais simples de algo mais complexo, ou mais abrangente. A expressão "*Modelo OSI*" se refere à uma

especificação ISO (*International Organization for Standardization*), que define um padrão para interconexão de sistemas abertos, do qual se pode instanciar a arquitetura atual da Internet.

2.4.3 Comunicação

O termo *comunicação* vem do latim *communicatìo* e *ónis*, que significa a ação de se comunicar. Na origem, comunicação se referia àquilo que era comum que era compartilhado. Toda comunicação envolve um canal, interlocutores e uma mensagem. Tais requisitos se aplicam a quaisquer comunicações, mesmo as humanas. Nestas, os interlocutores são dotados de mecanismos de transmissão, como fala, escrita e gestos; e de recepção, como audição, visão e tato. Os canais variam de acordo com o mecanismo utilizado, por exemplo, na comunicação através da fala, ondas sonoras são transmitidas pelo meio físico por meio de um canal com largura de banda de 20kHz, e são receptadas pelos interlocutores através da audição. A comunicação nas redes de computadores ocorre de maneira similar, uma vez que as tecnologias atuais de transmissão de dados envolvem a utilização de diferentes meios físicos, e estratégias de envio e recepção.

2.4.4 Endereçamento

O termo *endereçamento* refere-se ao ato de endereçar, que vem do latim medieval *directiare* unido ao sufixo *en*, e que significa a ação de tornar direto. O endereço define uma *chave* utilizada para alcançar algo, o que é um aspecto essencial da comunicação. O conceito de endereçamento se confunde com os conceitos de *identificação* e *localização*, que embora de naturezas distintas, são normalmente representados como endereços. Isso ocorre pois em geral os emissores destinam suas mensagens diretamente aos receptores, sem se importar com a localização dos alvos. Dessa maneira, a identificação torna-se o endereço, mesmo que não represente a localização que é de fato o meio de alcançar o destinatário.

2.4.5 Roteamento

O termo *roteamento* deriva da palavra *rota*, que vem do latin *rupta*, que significa *via*. *Roteamento* é o nome que se dá à estratégia para definição de uma rota, que pela definição acima, trata-se de um caminho para o alcance de um destino determinado. Em redes de computadores existem tradicionalmente dois tipos de *roteamento*, o estático e o dinâmico. No *roteamento estático*, existe uma tabela de rotas construída manualmente em cada elemento de rede, com endereços de *hosts* e de redes vinculados à interfaces do equipamento. Essas tabelas não se adaptam às alterações na rede, e não "escalam" em redes de grandes proporções, o que limita consideravelmente o seu uso. No *roteamento*

dinâmico a tabela de rotas é criada por protocolos de roteamento, tais como o BGP e o OSPF. Esses protocolos são programados para distribuir informações de rede entre os roteadores, de forma a adaptar as rotas refletindo a topologia atual.

2.4.6 Semântica

O termo *semântica* vem do grego *semasia*, que significa *significado*. A *semântica* é o estudo dos significados, e é aplicado em diversos ramos das ciências humanas como um mecanismo de relação entre significante e significado. O termo *semântica* quando aplicado à seres, normalmente é considerado como sinônimo de *sentido*. Na *Web Semântica* por exemplo, faz-se uso de meta-dados para dar *sentido* à conteúdos na Web.

2.4.7 Ontologia

O termo *ontologia* vem do grego *ontos* e *logoi* que significa *ciência do ser*. A *ontologia* foi concebida pelos gregos e explorada em diversos estudos filosóficos subsequentes, sendo comumente definida como parte da *metafísica*. Esta ciência trata do *ser enquanto ser*, isto é, do ser concebido como tendo características comuns à todos os seres. Usualmente, refere-se por *ontologia* o estudo que visa a definição de um ser, o que é muito comum na *ciência da computação*. Na *Web Semântica* por exemplo, a *ontologia* representa um mecanismo útil para um maior aprofundamento na descrição e interpretação do conteúdo. Nesses casos, faz-se uso de OWL (*Web Ontology Language*), uma linguagem utilizada na definição formal de "algo", bem como suas características e relações.

2.5 Endereçamento nas redes de computadores

O *endereçamento* é um aspecto primordial em qualquer tipo de comunicação, por se tratar do mecanismo que viabiliza a identificação e localização, e por conseguinte, a entrega das mensagens aos destinatários. Toda informação é direcionada à um público, o que dá sentido para a comunicação, e se baseia na definição de um endereço.

Desde os projetos das primeiras redes de computadores, o *endereçamento* foi tratado como um importante aspecto. A camada de rede, cuja a principal instância na arquitetura atual é o protocolo IP, é uma camada basicamente voltada ao endereçamento de *hosts*, o que revela a importância do endereço nessa abordagem.

Com o passar dos anos, a rede cresceu de maneira exponencial, e uma das soluções para esse problema foi a divisão de porções da rede, com espaços de endereçamento distintos. A Figura 2.3 apresenta os principais tipos redes encontrados, que também são descritos abaixo.

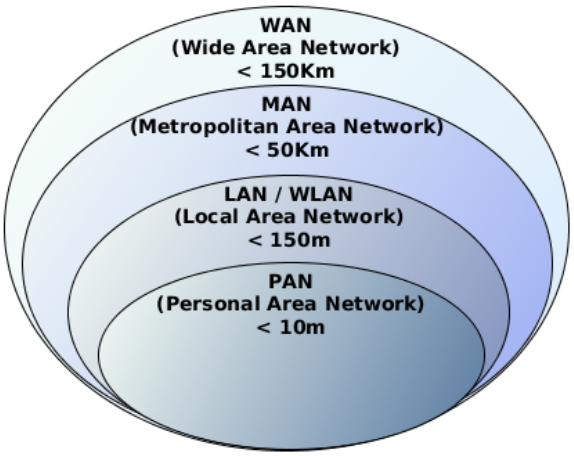


Figura 2.3: Tipos de redes utilizadas.

PAN *Personal Area Network* – são redes pessoais, voltadas para comunicação entre dispositivos, como por exemplo: Bluetooth e UWB.

LAN *Local Area Network* – representam as redes locais, formada por nós conhecidos e de pequenas proporções.

WLAN *Wireless Local Area Network* – são LANs onde os hosts estão interligados através de tecnologias sem fio.

MAN *Metropolitan Area Network* – são redes em escalas metropolitanas, normalmente formadas a partir da interconexão entre LANs em uma região definida.

WAN *Wide Area Network* – são redes mais complexas, com nós espalhados geograficamente, e de grandes proporções.

Não obstante a definição de porções de redes, o número de dados que trafegam entre os *hosts* é enorme, e desde os primórdios das redes, em áreas com poucos *hosts* interconectados, o processamento desses dados demandava alto consumo de CPU. Por isso foi criada a camada MAC (*Medium Access Control*), que baseada em um endereço atribuído à interface da placa de rede, filtra os pacotes endereçados ao *host* antes do processamento pelo sistema operacional. A esse endereço dá-se o nome de *MAC Address*, e em teoria devem ser únicos para cada interface de rede. Com essa estratégia, surgiu uma duplicação no endereço das primitivas de rede, que começaram a ter endereços na camada de rede (IP) e de enlace (MAC). Dessa maneira, todo nó da rede deve manter sincronizados os endereços de MAC e IP, para garantir que a rede funcione adequadamente.

Para resolver o problema de descoberta do endereço MAC, em 1982 foi proposto o ARP (*Address Resolution Protocol*) [Plummer 1982], que é um protocolo para o mapeamento entre os dois endereços. Todos os nós da rede possuem uma tabela de mapeamento chamada *Tabela ARP*, que é atualizada periodicamente através de *broadcasts* de requisições ARP conforme definido no protocolo.

Uma solução possível para o problema de tráfego desnecessário nas interfaces de rede dos *hosts* seria ter nós de rede que chaveassem as mensagens diretamente entre os *hosts*, baseando-se apenas no endereço definido na camada rede. Dessa maneira, não seria necessário o filtro da camada MAC, pois todas as mensagens que chegassem na placa de rede seriam destinadas ao *host*, e assim deveriam ser encaminhadas para processamento. No entanto, tal mudança demandaria alterações nas placas de rede e nos nós, o que contribuiu para a utilização do endereço MAC ainda nos dias de hoje.

Para atender aos requisitos de comunicação nas redes de computadores, a arquitetura TCP/IP definiu quatro tipos de endereçamentos conforme apresentados na Figura 2.4: *Unicast*, a mensagem é entregue a apenas uma entidade; *Anycast*, a mensagem é entregue a uma entidade dentro de um grupo de possibilidades; *Multicast*, a mensagem é entregue para um grupo de entidades; e, *Broadcast*, a mensagem é entregue a todas as entidades do domínio.

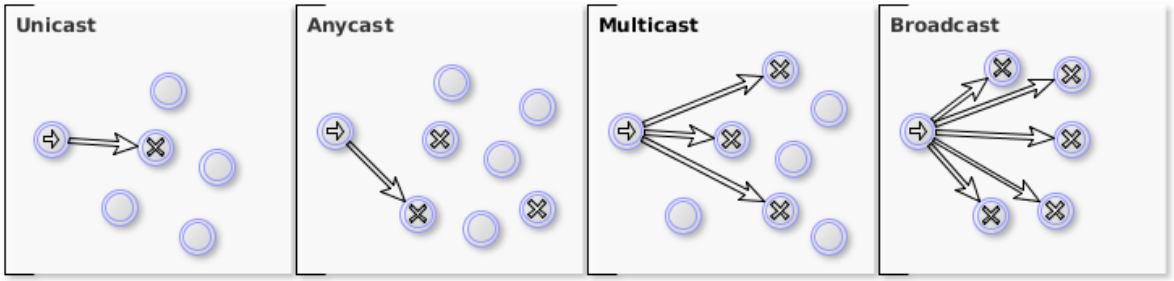


Figura 2.4: Estratégias de endereçamento.

O endereço IP é um espaço de 32 bits tradicionalmente observados como 4 octetos. Os primeiros bits identificam a classe e são reservados para endereçamento da rede, e os demais para endereçamento dos *hosts*. O espaço de 2^{32} endereços possíveis foi inicialmente dividido em cinco classes de endereçamento, chamadas A, B, C, D e E . As classes A, B e C definem subespaços para endereçamento de redes e *hosts*; a classe D foi definida para o endereçamento de grupos *multicast*; e a classe E ficou reservada para uso futuro. A Tabela 2.1 apresenta as classes de endereçamento, com o as faixas de endereços que elas cobrem, o padrão inicial de bits utilizado na designação da classe, a máscara de rede padrão, a quantidade de redes possíveis, e a quantidade de IPs disponíveis para endereçamento de *hosts*.

A classes apresentavam tamanhos fixos de sub-redes, que eram definidas de acordo com a classes de endereçamento. Essa característica começou a se tornar um problema com a iminência do esgotamento de endereços IPs disponíveis, o que motivou o surgimento do CIDR, que flexibilizou a definição de porções da rede através de máscaras de comprimento variado.

Classe	Faixa de endereços	Padrão Inicial	Máscara Padrão	Número	
				Redes	Ips/Rede
A	1.0.0.0 a 127.0.0.0	0	255.0.0.0	128	16.777.214
B	128.0.0.0 a 191.255.0.0	10	255.255.0.0	16.384	65.534
C	192.0.0.0 a 223.255.255.0	110	255.255.255.0	2.097.152	254
D	224.0.0.0 a 239.255.255.255	1110	-	-	-
E	240.0.0.0 a 255.255.255.254	1111	-	-	-

Tabela 2.1: Classes de endereçamento IP.

2.6 Internet do Futuro

Pesquisadores do mundo todo têm trabalhado em duas abordagens [Rexford and Dovrolis 2010]. A primeira, conhecida como "revolucionária" ou *clean-slate*, visa substituir completamente a arquitetura atual, o que apresenta grandes vantagens no desenvolvimento de uma nova solução, por não se limitar à especificação da arquitetura vigente. A grande desvantagem dessa abordagem está na dificuldade de implantação, que ocorre pela necessidade de suporte, ou coexistência, com a rede legada. A segunda abordagem, conhecida como "evolucionária", visa o aprimoramento da arquitetura atual através de novas ideias. A grande vantagem está na fácil implantação, que se deve à fidelidade das ideias propostas ao legado, o que por outro lado limita o campo de ação. Uma das principais abordagens "evolucionárias" para a Internet é o IPv6 [Deering and Hinden 1995], cuja proposta realizada em 1995, ainda hoje não foi completamente implantada [Wood 2011], principalmente por problemas de suporte da rede. Ao se considerar todo o desenvolvimento havido desde os anos 2.000 em torno da arquitetura Internet, não é difícil perceber que uma abordagem "evolucionária" de *facto* já está em curso desde então.

2.6.1 Movimentos ao redor do mundo

Nos Estados Unidos há diversas iniciativas que propõem uma nova arquitetura para Internet através de uma visão *clean slate*. Entre 2000 e 2003, o projeto NewArch [Clark et al. 2004] começou a discutir os objetivos e direcionar a pesquisa relacionada à Internet do futuro. Entre 2005 e 2010, um outro projeto, o FIND (*Future Internet Design*) [Fisher 2007], reforçou a visão da nova arquitetura para Internet definida através de uma abordagem *clean slate*, fornecendo amplo apoio para mais de cinquenta projetos na área. Nesse mesmo período, entre 2006 e 2012, a ideia da reformulação completa da arquitetura atual tomou força com o projeto da Universidade de Stanford chamado *Clean Slate* [STANFORD 2006].

GENI (*Global Environment for Network Innovations*) é uma iniciativa do NSF (*National Science Foundation*), assim como o FIND, que consiste na criação de um laboratório virtual para a viabilização de pesquisas e experimentos de inovação para transformação da rede. O projeto contempla uma rede distribuída pelos Estados Unidos, com infra-

estrutura de rede, processamento e armazenamento em cada nó, e é compartilhado por diversos pesquisadores em todo país. Com essa rede, tornam-se possíveis experimentos das pesquisas relacionadas às redes futuras em escala.

Outra importante iniciativa da NSF é o FIA (Future Internet Architectures) [Foundation 2011], que é um programa criado para estimular a pesquisa e inovação nos trabalhos relacionados à arquitetura da Internet do futuro. Esse programa contém quatro projetos que atualmente estão lidando diretamente com os aspectos da rede, tais como redes *content-centric*, mobilidade, computação na nuvem e segurança. Um desses projetos é o *MobilityFirst* [Seskar et al. 2011], cuja arquitetura foca no aspecto de mobilidade. Essa abordagem propõe uma nova pilha de protocolos e considera um novo esquema de endereçamento baseado em identificadores globais únicos chamados GUID (*Globally Unique Identifier*), o que é similar ao conceito de *Título* apresentado neste trabalho. Dessa maneira, o GUID também viabiliza o atendimento dos requisitos mobilidade e *multicast*, no entanto na abordagem *MobilityFirst*, o plano de controle ocorre necessariamente em uma estratégia *in-band*, enquanto na apresentada neste trabalho há uma ortogonalidade no plano de controle, que pode ser tratado também por uma estratégia *out-of-band*.

Na Europa, o FP7 (*Seventh Framework Programme*), um programa de financiamento para a investigação e o desenvolvimento tecnológico dos países pertencentes à União Europeia, foi programado para durar 7 anos (entre 2007 e 2013), e foi substituído em 2014 pelo *Horizon 2020*.

O FP7 é constituído por quatro programas específicos (cooperação, ideias, pessoas e capacidades), mais um quinto programa voltado para a energia nuclear. Cada programa define temas à serem abordados e cada tema traz um conjunto de desafios, e por fim, cada desafio define um conjunto de objetivos. Foram abertas diversas chamadas de propostas para o financiamento de projetos que envolvessem os objetivos definidos e ao todo foram programados investimentos de cerca de 50,5 bilhões de euros, um aumento de 63% em relação ao programa anterior, o FP6.

Um dos temas principais do programa de *cooperação* é o *Information and Communication Technologies* (ICT), cujo Desafio 1 (*Pervasive and Trusted Network and Service Infrastructures*) [Commission 2012a], traz dois objetivos ligados às novas arquiteturas de redes. São eles o objetivo 1.1 (*Future Networks*) e o 1.6 (*Future Internet Research and Experimentation*). Assim, diversos projetos com foco na Internet do futuro foram financiados pelo FP7, como o 4WARD, CHANGE, MEDIEVAL, PURSUIT, SAIL, SENSEI, TRILOGY e UNIVERSELF [Commission 2012b]. Estes projetos trabalham com diferentes aspectos das redes futuras e muitos deles apresentam abordagens *clean slate*.

O *European Future Internet Assembly*, também conhecido como FIA [Eurescom 2012], é uma união de cerca de 150 projetos relacionados ao desafio 1 do tema de ICT do FP7, cujo objetivo é incentivar a colaboração entre os grupos de pesquisa, fomentando um movimento em prol da evolução da Internet. Para isso, o FIA define grupos de trabalho

com focos específicos, além de organizar eventos e publicações ligadas ao tema.

O 4WARD *NetInf* [D'Ambrosio et al. 2010] apresenta um paradigma de rede centrado em informações conhecido como ICN (*Information-Centric Networking*), com base em um sistema distribuído que controla a comunicação e fornece serviços úteis, tais como *caching*, armazenamento e transporte. Ele utiliza um esquema de nomes independente da topologia, baseado em uma chave chamada de *Identificador*, que está relacionada com o *Título* apresentado neste trabalho. Esses identificadores são usados para registrar e resolver os *Information Objects*, que são primitivas trocadas durante a comunicação na abordagem *NetInf*.

2.6.2 Ambientes de teste no Brasil

No Brasil, o projeto FIBRE (*Future Internet Testbeds Experimentation Between Brazil and Europe*) [Sallent et al. 2012], reúne diversas instituições de ensino de todo o país, a RNP (*Rede Nacional de Pesquisa*) e o CPqD (*Centro de Pesquisa e Desenvolvimento em Telecomunicações*).

O projeto é o resultado de uma proposta conjunta entre Brasil e União Europeia, cujo principal objetivo é o de criar uma infraestrutura no Brasil para a pequisa e experimentação de novas arquiteturas de rede, ligada à infraestrutura europeia. Esse projeto cria uma rede transcontinental, o que melhora os testes em escala e estreita parcerias entre pesquisadores da América do Sul e Europa. O projeto está alinhado ao objetivo 1.6 do FP7, o FIRE, e por isso foi financiado por esse programa.

O EDOBRA (*Extending and Deploying OFELIA in Brazil*) [ATNOG 2012], é fruto de um pacote de trabalho em uma chamada aberta em 2012 pelo FP7/OFELIA, que inclui Universidade Federal de Uberlândia e a Universidade de São Paulo, no Brasil; e a Universidade de Aveiro, em Portugal. Financiado pelo FP7, o EDOBRA promoveu uma extensão no Brasil do OFELIA (*OpenFlow in Europe: Linking Infrastructure and Applications*), que é uma *testbed* europeia financiada pelo FP7. Nesse projeto foram realizados testes com a arquitetura desenvolvida neste trabalho, a ETArch, e os resultados preliminares foram apresentados nas convenções europeias do FP7, congressos e workshops.

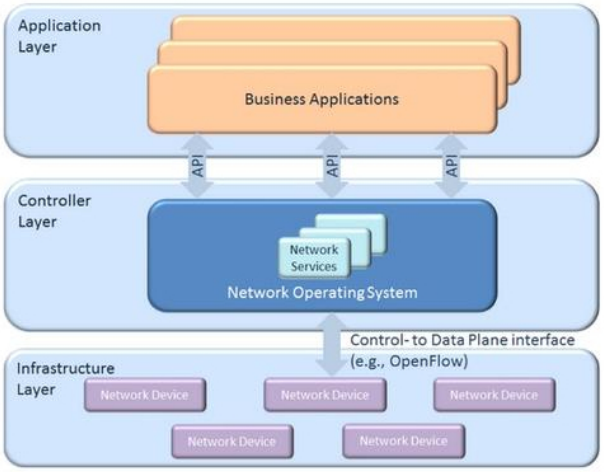
2.6.3 Redes Definidas por Software

SDN é uma abordagem para redes de computadores, que cria uma interface lógica aberta e bem definida para o controle da rede, como é apresentado na Figura 2.5. Essa camada abstrai a rede e fornece serviços que permitem a sua reconfiguração em tempo de execução. Por esse motivo, o SDN é considerado uma ferramenta de inovação para pesquisas sobre as redes futuras, por separar o plano de controle do plano de dados, através de uma solução de software.

Esta arquitetura desacopla o controle de rede e funções de encaminhamento que permitem o controle da rede para se tornar diretamente programável e da infra-estrutura subjacente a ser captada para aplicações e serviços de rede.

Através da abordagem SDN tona-se possível a separação entre os planos de dados e controle, que podem ser programados de maneira independente da infraestrutura subjacente. Isso simplifica drasticamente o processo de administração das redes, por definir uma camada que abstrai os diferentes tipos de equipamentos, e flexibiliza a implantação de comportamentos de maneira pró-ativa ou reativa, através de um *software* centralizado responsável pela tramitação da comunicação.

Diversas aplicações têm se beneficiado dessa flexibilidade provida pela SDN. De fato, a implementação deste trabalho se baseia em OpenFlow, que materializa os conceitos de SDN. Também na abordagem de [Rothenberg et al. 2012], o BGP ganha um importante reforço de um sistema baseado em RCP (*Routing Control Platform*), controlado via protocolo OpenFlow.



Fonte: https://www.opennetworking.org/

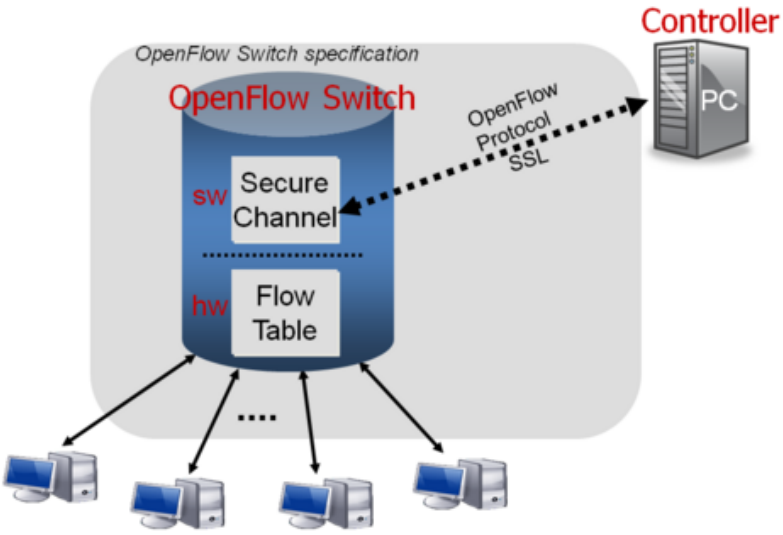
Figura 2.5: Visão SDN.

2.6.4 OpenFlow

O OpenFlow é a materialização da abordagem SDN, e implementa a separação dos planos de controle e dados através do conceito de fluxo e do protocolo aberto homônimo. A orquestração no nível de controle é realizada pelo Controlador OpenFlow, um *software* extensível que dá suporte ao protocolo OpenFlow, e que permite a criação de comportamentos de maneira flexível como proposto pela SDN. Dessa maneira, torna-se possível a implementação de algoritmos sensíveis ao contexto de utilização, estado da rede e às requisições de sistemas externos de maneira pouco intrusiva. O Controlador OpenFlow realiza configurações de regras nos elementos de rede de forma a guiar o tratamento das primitivas dos fluxos de dados no plano de encaminhamento. Essas regras definem pa-

drões à serem utilizados como filtros para as primitivas de redes, e ações à serem tomadas, como por exemplo: encaminhar a primitiva para uma determinada porta e/ou alterar um campos do cabeçalho. Elas são armazenadas em tabelas de fluxos de maneira similar às tabelas de roteamento nos *switches* convencionais.

O OpenFlow versão 1.0 define a utilização de apenas uma tabela de fluxo no *switch* conforme apresentado na Figura 2.6, enquanto nas versões posteriores foram adicionadas múltiplas tabelas que são analisadas sequencialmente. O objetivo dessa estratégia é diminuir a latência da rede através de uma otimização no processo de identificação de fluxos. As versões 1.1 e 1.2 apresentaram também mudanças nas ações possíveis para o tratamento dos fluxos. A versão 1.3 apresentou importantes evoluções, sobretudo na capacidade de criação de regras para identificação de fluxos de maneira flexível, e não mais limitada aos protocolos da Arquitetura TCP/IP. Essa nova estratégia foi chamada de *OpenFlow Extended Match* (OXM) [Fernandes and Rothenberg 2014], e fornece um importante apoio para abordagens disruptivas, por facilitar o suporte à novos protocolos pelos *switches* e *controladores* OpenFlow.



Fonte: http://yuba.stanford.edu

Figura 2.6: OpenFlow Switch 1.0

Atualmente verifica-se um grande crescimento no número de elementos de redes com suporte ao OpenFlow. Grandes fabricantes de equipamentos, como Cisco, NEC, Juniper e HP, têm investido em pesquisas sobre SDN e oferecem equipamentos com suporte ao protocolo OpenFlow. Entretanto, o fato de SDN ter trazido à luz a possibilidade de inferir no comportamento programado da rede, não resolve a tarefa de dar suporte aos requisitos atuais. Nesse contexto, pesquisadores ao redor do mundo, como [Foster et al. 2013] e [Kim and Feamster 2013], estão engajados na criação de software capazes de abstrair as

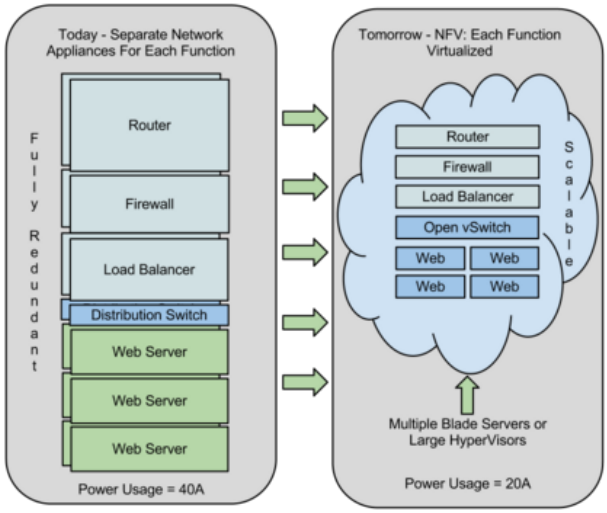
funcionalidades da rede.

2.6.5 Virtualização de Funções de Rede

No universo das funcionalidades de uma rede, NFV (*Network Functions Virtualization*), que complementa a SDN, reforça o conceito de manter o controle da rede em uma camada de *software* independente. O NFV se alinha com os avanços da pesquisa de *cloud computing*, por utilizar técnicas bem difundidas de virtualização e sistemas distribuídos. Essa abordagem visa preencher a lacuna deixada pelos elementos com funções bem definidas na arquitetura atual, como *firewalls* e *load-balancers*, na abordagem SDN. Dessa maneira, torna-se possível alocar "máquinas virtuais" para a realização de funções de rede de maneira flexível, por se basear em *software*, e escalável, por ser virtualizado.

A abordagem NFV torna possível a utilização de servidores comuns para a execução de funções da rede que tradicionalmente seriam tratadas por *appliances* como apresentado na Figura 2.7, o que otimiza a utilização de *hardware* e economiza recursos. Essa abordagem também se beneficia de todas as vantagens da virtualização de servidores, como elasticidade e resiliência, e de *cloud computing*, como processamento compartilhado e disponibilizado como serviço.

As funções de redes definidas pela abordagem NFV podem ser mais facilmente alteradas, assim como a pilha de funções de rede utilizadas no tratamento das primitivas, pois as ligações entre as máquinas virtuais é lógica, diferentemente da abordagem utilizada nas redes atuais. A questão do desempenho no acesso à placada de rede pelas máquinas virtuais vem sendo aprimoradas pelos principais fabricantes de componentes, como a Intel com o *Data Plane Development Kit* (DPDK), que possibilita um alto desempenho no processamento de primitivas de rede através do acesso direto ao *hardware*.



Fonte: Steve Noble - http://www.sonn.com/

Figura 2.7: Comparação entre a abordagem tradicional e a NFV.

Capítulo 3

ETArch: Visão Geral de Arquitetura para Internet do Futuro

Este capítulo objetiva apresentar a Arquitetura ETArch (*Entity Title Architecture*), que introduz uma aproximação semântica entre a camada de aplicação e as camadas inferiores. Tal aproximação contribui para a redução do *overhead* de comunicação e diminuição da redundância das funções da rede, especialmente no aspecto de *endereçamento*. A maior expressividade semântica nas camadas de rede flexibiliza a definição de requisitos para o estabelecimento da comunicação, e facilita a oferta de novos serviços às camadas superiores.

A ETArch foi concebida em uma abordagem *clean slate* e utilizou o *Modelo de Título* [Pereira 2012] como referência, materializando os conceitos de aproximação semântica e da comunicação sem a pilha de protocolos TCP/IP. Ela foi definida em [Oliveira Silva 2013], onde foram especificados os principais componentes dessa arquitetura. Este trabalho apresenta um refinamento de trabalhos anteriores, bem como uma implementação dos conceitos envolvidos, visando as comunicações *multicast* nas redes atuais e futuras.

Através de uma estratégia de definição de serviços com "contratos fracos", baseados em trocas de documentos de descrição semântica, o *Modelo de Título* viabiliza a construção de uma arquitetura para funcionalidades da rede similar a um *framework*, na qual podem ser criados e alterados algoritmos utilizados no tratamento de primitivas, de maneira a prover comportamentos específicos para cada requisito comunicacional definido. Nesse sentido, as abordagens de NFV e ETArch convergem para a flexibilização das funcionalidades de rede. A abordagem proposta também se relaciona com a de SDN, pela separação do plano de controle, que é definido através de um sistema distribuído controlador da rede.

As inovações nos aspectos de *identificação* e *localização* viabilizam a agregação de tráfego através do serviço *multicast*, baseando-se no conceito de *workspace*, foco deste trabalho. Embora tal requisito já tenha sido abordado em diversos trabalhos, com soluções tanto na camada de rede quanto na de aplicação, nenhuma solução se mostrou eficiente de fato, por se limitar à arquitetura atual. Essas limitações são impostas não somente às soluções de *multicast*, mas também às de diversos outros requisitos, como o de segurança e QoS. O problema está ligado à baixa flexibilidade da arquitetura TCP/IP e ao endereçamento hierárquico, que define uma única chave utilizada em ambos os aspectos de *identificação* e *localização*.

3.1 Modelo de Título

O *Modelo de Título* é um modelo de referência para as redes futuras que define, dentre outras característica, a utilização de uma nova estratégia de endereçamento, chamado *endereçamento horizontal*, cujo principal objetivo é resolver o problema de ambiguidade do endereçamento na arquitetura atual, que representa tanto a *identificação* quanto a *localização*. Nessa nova estratégia, *Títulos* servem para identificação unívoca de *entidades* independentemente de topologia.

Removida a ambiguidade no endereçamento, boa parte dos problemas da arquitetura atual são resolvidos, por estarem intimamente ligados às limitações desse aspecto. Dentre esses requisitos destacam-se os de *mobilidade* e *multicast*, que se baseiam prioritariamente nas estratégias de roteamento, que por sua vez, são dependentes dos mecanismos de localização. Uma vez que a identificação não tem vínculo com a localização da *entidade*, elimina-se a necessidade de troca de identificação durante a migração para outros pontos de acesso e torna-se possível identificar um grupo sem se limitar à localização física de seus componentes.

Assim como o *Modelo OSI*, o *Modelo de Título* é um modelo em camadas voltado para a interconexão de redes. A grande diferença está na substituição das camadas de rede e transporte, por uma única camada de Comunicação, responsável pela entrega das primitivas de rede e pelo aprovisionamento de funcionalidades para tratamento dos requisitos de comunicação das *entidades*. A interface dessa camada utiliza serviços de baixo acoplamento, baseado na troca de documentos semânticos, o que flexibiliza a "contratação de serviços" das camadas superiores. Com a "eliminação" de camadas, e maior expressividade nas interfaces, considera-se que esse modelo provê uma *aproximação semântica* entre a aplicação e a rede.

Este modelo introduz também o conceito de *Workspace* que é um barramento lógico independente das topologias e interconexões das redes subjacentes e que funciona como um "duto" para o envio de primitivas às *Entidades* participantes do contexto de comunicação. Todas primitivas são transmitidas através do *Workspace*, que também é identificado por

um título. Dessa maneira, o *Modelo de Título* provê suporte natural ao requisito de *multicast*, uma vez que toda a comunicação é destinada a um grupo de *entidades*.

3.2 ETArch: *Entity Title Architecture*

Entity Title Architecture é a materialização do *Modelo de Título* e apresenta soluções para os conceitos de *endereçamento horizontal*, *aproximação semântica* e *barramento lógico* (*Workspace*). Trata-se de uma arquitetura *clean slate*, voltada para a Internet do futuro, cujo principal objetivo é flexibilizar a rede, de maneira a facilitar o atendimento dos requisitos atuais e futuros, por meio da utilização de linguagens com expressividade semântica.

A ETArch introduz o conceito de separação dos planos de dados e controle, por meio de um sistema distribuído responsável pelo controle da rede. Esse sistema chama-se DTS e é composto por agentes espalhados geograficamente sobre um esquema hierárquico. O DTS é o meio pelo qual o conceito de *endereçamento horizontal* se torna possível, pois como o Título nesse novo modelo só representa a identificação da *entidade*, a localização fica à cargo desse sistema, que é responsável também por armazenar informações sobre as *entidades*, bem como aprovisionar os *barramentos lógicos* necessários para o estabelecimento da comunicação.

O conceito de *barramento lógico* é materializado pelo *Workspace*, que é um *provider* para as comunicações de um grupo de *entidades*, e é identificado por meio de um título e um conjunto de requisitos à serem atendidos. Tradicionalmente, em termos físicos, o *workspace* é encarado como uma *árvore geradora mínima* entre as *entidades* participantes de um grupo naturalmente *multicast*. Toda comunicação é destinada a um *workspace*, que é de fato o primeiro nível de endereçamento nessa arquitetura. Dessa maneira, pode-se dizer que a ETArch provê suporte natural à comunicação entre um grupo de *entidades*, o que chamamos de *multicast natural*.

Para a ETArch não faz mais sentido falar em quatro tipos de endereços (*Unicast*, *Anycast*, *Multicast* e *Broadcast*) para uma entidade, existindo apenas o endereçamento *Multicast*, sendo os demais casos particulares deste.

A arquitetura também define os protocolos responsáveis pela implementação da camada de Comunicação. Tratam-se dos protocolos *DL-Ontology* e *Net-Ontology*, responsáveis respectivamente pelo encaminhamento e tratamento das primitivas da arquitetura. Esses protocolos são dispostos de maneira não necessariamente paralela, respeitam os princípios filosóficos definidos pelo *Modelo de Título*, e contemplam aspectos intrínsecos da implementação e de implantação nas redes atuais. Também são apresentados dois protocolos para o plano de controle, o ETCP e DTSCP, que são responsáveis respectivamente pelo plano de controle entre as *entidades* e o DTS, e pelo plano de controle interno ao DTS.

3.3 Definição de Termos e Conceitos ETArch

Nesta seção serão apresentados os principais Termos e Conceitos introduzidos pelo *Modelo de Título*, e respectivas materializações na Arquitetura *Entity Title Architecture*, bem como os componentes filosóficos e arquiteturais essenciais ao entendimento desse trabalho.

Entidade

Elemento principal capaz de se comunicar no ambiente distribuído criado pela arquitetura e cujas necessidades de comunicação podem ser semanticamente interpretadas e suportadas pela camada de Comunicação. Uma *Entidade* pode ser identificada por um conjunto de *títulos*, numa relação unívoca, e pode se mover durante suas comunicações ao longo do *Workspace*. Exemplos de *entidades* no mundo real são: aplicação, conteúdo, servidor, usuário e sensores.

Título

Designação única e não ambígua utilizada para garantir a identificação inequívoca de elementos da arquitetura ETArch. Um *Título* designa apenas um elemento da arquitetura, sendo que um elemento pode ser identificado por mais de um *Título*. O *Título* desempenha um papel-chave no mapeamento do endereçamento horizontal de elementos da arquitetura para fins de localização.

Requisito

É uma necessidade associada a elementos da arquitetura, que representa o que é necessário para o tratamento da comunicação, e que deve ser garantida pelo *Workspace*. Um elemento (por exemplo uma Entidade) pode definir zero, um ou mais requisitos para seu funcionamento, tais como QoS, QoE, *multicast* e segurança.

Capacidade

É um recurso oferecido (suportado) por elementos provedores da arquitetura, que deverão ser capazes de atender aos *requisitos* estabelecidos, por exemplo, para uma determinada comunicação.

Endereçamento Horizontal

É um tipo de endereçamento que independe da localização física ou lógica de *Entidades* e outros elementos da arquitetura, permitindo que haja mobilidade sem a necessidade de troca de endereço.

Domain Title Service (DTS)

É um sistema distribuído responsável pelo gerenciamento do ciclo de vida de todos os elementos da arquitetura ETArch, tais como *Entidades* e *Workspaces*. Como consequência, o aprovisionamento de capacidades em equipamentos físicos (por exemplo

switches), relativas a requisitos de comunicação de *Entidades*, é responsabilidade do DTS. Este sistema "entende" os requisitos e aprovisiona as *capacidades* das *entidades*, gerencia funcionalidades e configurações nos equipamentos para o tratamento adequado da comunicação. Ele é composto por *Domain Title Service Agents* (DTSA), que estão distribuídos ao longo da rede como agentes dedicados ou em equipamentos específicos, tais como *switches* e roteadores.

Workspace

É um barramento lógico *full-duplex* independente do esquema de interconexão e topologias da rede subjacente, que provê as capacidades necessárias às comunicações de duas ou mais *entidades* e pode ser identificado por um ou mais *Títulos* numa relação unívoca. O *Workspace* é criado por uma *Entidade* que queira disponibilizar um espaço de comunicação (barramento lógico) com um propósito específico. Durante a sua criação, a *Entidade* informa o conjunto de requisitos que devem ser suportados pelo *Workspace* a fim de atender os requisitos de todas as *entidades* comunicantes. Uma nova *Entidade* pode se ligar também a um *Workspace* já existente e, nesse caso, o barramento lógico é estendido de forma a conectar a nova *entidade* participante. Da mesma forma, uma *entidade* pode se mover pela rede mantendo-se associada ao *workspace* do qual ela já faz parte.

DL-Ontology

Provê um mecanismo de transmissão de primitivas na camada de Comunicação, desempenhando o papel de uma sub-camada de enlace (DL – *Data Link*). Esta subcamada utiliza uma sintaxe abstrata baseada em Ontologia (OWL), o que flexibiliza a definição dos formatos das primitivas, eliminando assim o uso de formatos pré-definidos reinantes nas tecnologias atuais, por exemplo *Ethernet*. Isso permite que o formato da primitiva de rede se adapte de acordo com os requisitos comunicacionais definidos pela entidade, o que facilita a implantação de novas funcionalidades para o atendimento de novos requisitos. A transmissão de dados é realizada a partir da análise dessas primitivas pelos elementos de rede, que colhem informações para tratá-las e encaminhá-las segundo as políticas definidas pelo DTS. Elas referenciam *workspaces* a partir da declaração de títulos, o que torna-se possível devido ao estabelecimento prévio do *barramento lógico* pelo DTS, e pelas novas estratégias de *identificação*, *localização* e *roteamento* apresentadas pela Arquitetura ETArch.

Net-Ontology

É responsável pela interpretação semântica das necessidades das *Entidades*, e pela implementação das funcionalidades que atendam à essas necessidades. Trata-se de uma sub-camada, que utiliza uma sintaxe abstrata baseada em Ontologia (OWL), com um mecanismo de interpretação semântica e de modularização de funcionalida-

des, que combina *Requisitos* e *Capacidades*, de forma a estabelecer a comunicação entre um grupo de *Entidades*.

Entity Title Control Protocol (ETCP)
É o protocolo que define a interface entre *Entidades* e o plano de controle gerenciado pelo DTS. Oferece serviços para gestão dos ciclos de vida de *entidades* e *workspaces*. Exemplo: entity-register, workspace-attach etc.

Domain Title Service Control Protocol (DTSCP)
É o protocolo que disciplina as comunicações internas ao plano de controle do DTS nas interações entre seus membros (DTSAs).

3.4 DTS : *Domain Title Service*

O DTS é responsável pela orquestração do plano de controle que mantém os registros de *entidades* e gerencia os ciclos de vida de *workspaces* para suporte às comunicações entre *entidades*. Ele é responsável pela identificação inequívoca de *títulos*, e pelo gerenciamento de requisitos comunicacionais definidos pelas *entidades*, assim como pelo aprovisionamento dos mecanismos utilizados no tratamento das primitivas de rede. Por fim, o DTS é responsável pelo estabelecimento da comunicação, o que se dá através da criação do *workspaces*.

Esse sistema é composto por um conjunto de agentes chamados DTSAs, que são responsáveis pelo controle de porções da rede, como apresentado na Figura 3.1. O *Serviço de Domínio de Título*, como o DTS pode ser traduzido, tem um funcionamento hierárquico, parecido com o do DNS, baseado em tabelas de expiração periódica e elementos distribuídos na rede em níveis hierárquicos, o que contribui para escalabilidade e resiliência da arquitetura ETArch. A interconexão entre os agentes pode ser realizada em uma estratégia *in-band*, utilizando a própria infraestrutura de rede como meio de transmissão; ou *out-of-band*, com uma infraestrutura dedicada ao plano de controle. Da mesma maneira, o plano de controle entre equipamentos de rede e os agentes do DTS pode ser realizado de dois modos, tanto *in-band* quanto *out-of-band*, de acordo com níveis de confiança ou número de recursos disponíveis para a configuração da rede. A disposição hierárquica dos DTSAs viabilizam a divisão da rede em fatias gerenciáveis e simplifica o controle desses subconjuntos, por dividi-los em diferentes níveis.

A arquitetura ETArch independe da topologia da infraestrutura de rede física subjacente, deste modo, a estratégia de interconexão de redes é planejada similarmente às redes atuais, compondo-se de LANs, WLANs, MANs, WANs e WWANs. O DTS provê um mecanismo de controle lógico hierárquico cujas atividades são desempenhadas por agentes com responsabilidades bem definidas. Dessa maneira, propõe-se um controle da rede em

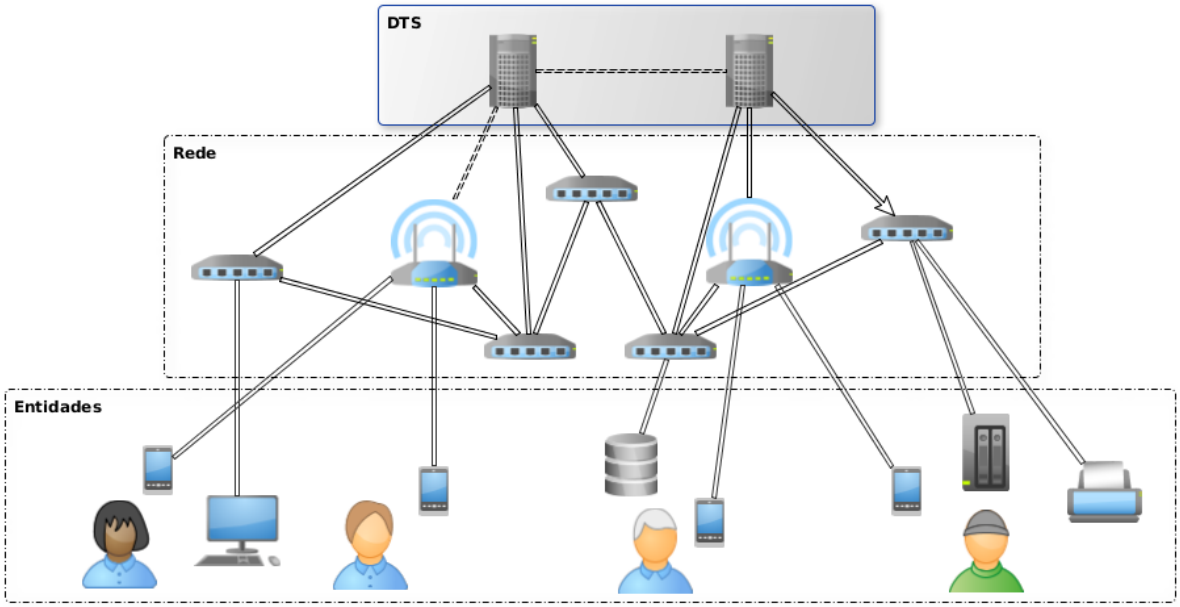


Figura 3.1: Visão Conceitual do DTS.

níveis (*tier*), no qual cada nível "esconde" detalhes de configuração/implementação de outros níveis, a fim de facilitar o controle de grandes redes.

Os níveis de hierarquia (*tiers*) do DTS são interconectados por agentes especiais denominados MDTSAs (*Master Domain Title Service Agents*). Estes agentes são responsáveis por porções da infra estrutura de rede, análogos a um domínio controlado por um ISP (*Internet Service Provider*). Os agentes do nível mais baixo da hierarquia são responsáveis pelo gerenciamento de *entidades* e dos elementos de redes (equipamentos) em sua localidade, enquanto agentes de níveis superiores na hierarquia tendem a orquestrar a rede estritamente do ponto de vista lógico, deixando que os detalhes de configuração mais específicos sejam aprovisionados pelos agentes dos níveis inferiores. O nível mais alto da hierarquia de DTSAs é chamado de *root level* e é responsável por uma visão macro da rede global, sendo responsável pelo gerenciamento de *workspaces* visíveis à todas as *entidades* do mundo.

A estrutura aplicada ao DTS segue um esquema de árvore (invertida), justificando o termo *root* para a designação dos agentes do nível zero. Por isso, algorítimos clássicos de busca e balanceamento podem ser aplicados a esse sistema, o que também contribui para a estratégia de distribuição do plano de controle. A Figura 3.2 apresenta uma visão simplificada do DTS implantado em escala global.

O DTS tem um papel fundamental na mudança do paradigma *cliente/servidor*, muito aplicado desde os primórdios da Internet. Nesse método, a aplicação é iniciada do lado do *servidor*, que permanece em estado de espera enquanto aguarda as solicitações de conexões dos *clientes*. Essa situação gera o problema da confusão da sincronização entre estações [Comer 2000], no qual o *servidor* sempre deve iniciar antes dos *clientes*. No *Modelo de Título*, todas as *entidades* nascem aptas à serem tanto *clientes* quanto *servidores*, o que

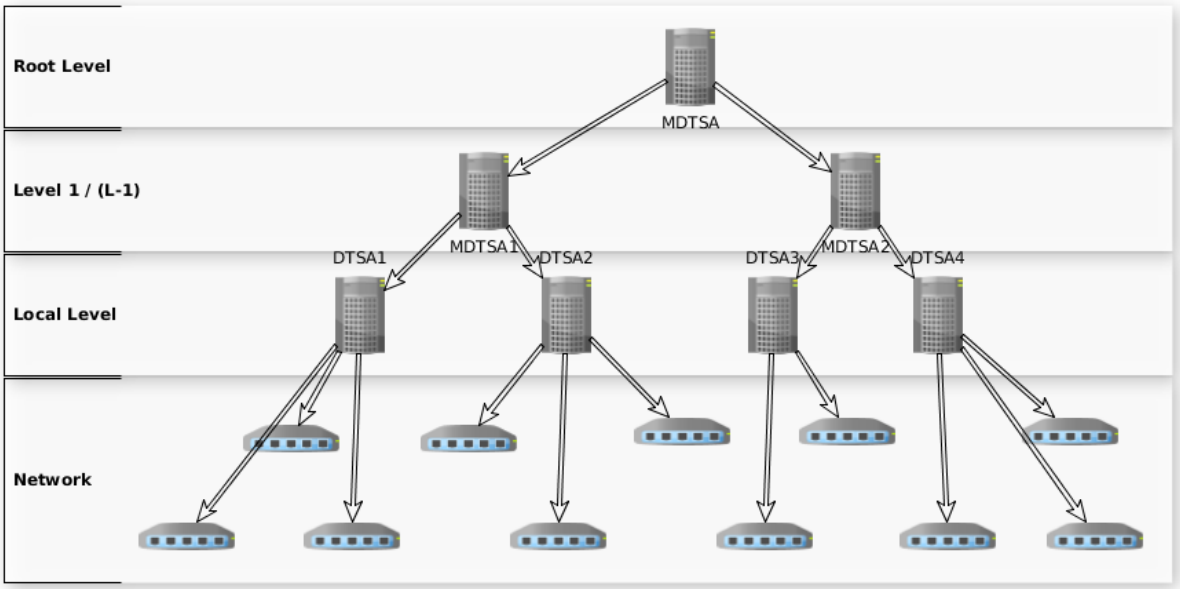


Figura 3.2: Hierarquia de DTSAs em três níveis.

introduz o paradigma *entidade/entidade*. Nesse novo método, quaisquer das *entidades* podem assumir o papel de *servidora* das informações, obedecendo a regras definidas no estabelecimento da comunicação. Isso facilita o atendimento dos requisitos de *resiliência* por meio de um *load-balancing natural*, e viabiliza a comunicação *multicast real*, por meio do suporte natural à troca de mensagens em grupo.

Os aspectos de segurança na identificação das *entidades* podem fazer uso de estratégias de senhas, certificados digitais, cadastro de origens conhecidas e análise de reputação, o que também é escopo do DTS. Tal requisito é de suma importância na estratégia de *endereçamento horizontal* proposta, por lidar com um escopo de endereçamento aberto. O resultado disso é a necessidade de métodos mais eficientes para atestação da identidade da *entidade*, de maneira a identificá-la inequivocamente, e facilitar a sua disponibilidade para o estabelecimento da comunicação (ligação ao *workspace*). Enquanto as estratégias que envolvem senhas, certificados digitais e cadastro de origens conhecidas são amplamente utilizados na camada de aplicação hoje em dia, aspectos como análise de reputação ainda estão sendo pesquisados em linhas paralelas.

O DTSA é composto por um conjunto de módulos, cujos os objetivos estão ligados ao armazenamento e processamento de dados, e de disponibilização de serviços para rede. Abaixo uma breve descrição desses componentes.

- **Entity Manager** – Responsável pela manutenção dos estados da *entidade*, bem como pela sua relação com um ou mais *títulos*.
- **Workspace Manager** – Responsável pelo gerenciamento do ciclo de vida dos *workspaces*, controlando estados, requisitos, *entidades* participantes e configuração da rede.

- **Topology Manager** – Responsável pelo conhecimento da topologia de rede, o que é indispensável para o aprovisionamento de *workspaces*. Esse módulo tem interfaces distintas para busca de dados de topologia, dentre elas o SNMP (*Simple Network Management Protocol*) e o próprio OpenFlow. As informações de topologia normalmente são mantidas através do protocolo aberto LLDP (*Link Layer Discovery Protocol*) ou através de protocolos particulares, como é o caso do CDP (*Cisco Discovery Protocol*).
- **ETCP Module** – Módulo que implementa o autômato do protocolo ETCP, responsável pelo plano de controle entre o DTS e e *entidades*.
- **DTSCP Module** – Módulo que implementa o autômato do protocolo DTSCP, responsável pelo plano de controle entre o DTSAs.
- **NEC -*Network Element Connector*** – Responsável pela comunicação com os elementos de rede e efetiva configuração do *workspace* na rede. Esse componente abstrai os diversos tipos de equipamentos existentes e fornece amplo suporte ao protocolo *OpenFlow*, que visa padronizar essa interface de configuração através de um formato flexível baseado no conceito de fluxos, base para a implementação do conceito de *workspace*.
- **Authentication Provider** – Responsável pela autenticação de *entidades* por meio de identidades, usando estratégias que envolvem o uso de senhas, certificados digitais, cadastro de origens conhecidas e análise de reputação.
- **Semantic Reasoner** – Motor de inferências para a interpretação semântica dos requisitos das *entidades*. Interpreta OWL, uma linguagem de representação semântica que é utilizada para definir os requisitos da aplicação durante o estabelecimento da comunicação.
- **Routing Module** – Responsável pelo cálculo da melhor rota para a ligação de *entidades* a um *workspace*. Esse módulo se baseia no cálculo de árvores geradoras mínimas através do uso de métricas relacionadas a custo, utilização, velocidade, confiabilidade e impacto ambiental.
- **Database Module** – Responsável pelo armazenamento dos dados referentes a *entidades*, *workspaces* e *topologias*.
- **Core Module** – Responsável pelos aspectos gerais do DTS, bem como pela junção e utilização dos demais módulos.

3.5 Workspaces

Workspace é um dos conceitos centrais na arquitetura ETArch, tornando transparente os aspectos físicos da comunicação (meios físicos, topologias, interconexão de redes etc)

por uma representação de alto nível de abstração. O *Workspace* representa essencialmente um conteúdo, uma conversa, ou um propósito que se torna disponível por meio de um barramento lógico. Entidades, que queiram compartilhar a *coisa* oferecida pelo *Workspace*, podem fazê-lo ligando-se ao barramento. Embora as capacidades do *workspace* definam aspectos a ser configurados na infraestrutura de interconexão física, a ETArch torna esses aspectos da configuração transparentes às aplicações (*entidades*). Dessa maneira, o *Workspace* é encarado como um "fim", e não como um "meio", o que é primordial para o entendimento deste conceito e é uma mudança fundamental em relação as arquiteturas tradicionais.

De maneira geral, os usuários não estão interessados nos servidores, mas sim no objeto (*coisa*) disponibilizado. Quando o usuário busca um determinado conteúdo, ele não se atém aos detalhes relacionados à localização física e infraestrutura de rede, ele apenas manifesta o interesse em um conteúdo. Nesse contexto, são muito comuns abordagens *content-centric* aplicadas aos sistemas atuais, principalmente em grandes provedores de conteúdo como *Facebook* e *Google*. Nessas aplicações são implantadas soluções em nível de aplicação, que direcionam a busca de um conteúdo para um determinado servidor. De maneira semelhante, na Arquitetura ETArch, o DTS é responsável pelo estabelecimento do *workspace* para a disponibilização do conteúdo de maneira transparente para a *entidade*.

Toda comunicação se dá por um canal, que é uma partição (física ou lógica) do meio, utilizada para ligar os interlocutores. Na arquitetura atual, o canal é definido em uma abordagem *fim-a-fim*, na qual ele representa uma ligação de "um-para-um" entre duas *entidades*. O conceito de canal introduzido pelo *workspace* é radicalmente alterado, e se assemelha à ideia de um barramento lógico, no qual todas as *entidades* interessadas em se comunicar se conectam. Esse tipo de canal passou a ser chamado de *multi-end*, e é o mecanismo pelo qual a arquitetura proposta fornece suporte natural ao requisito de *multicast*.

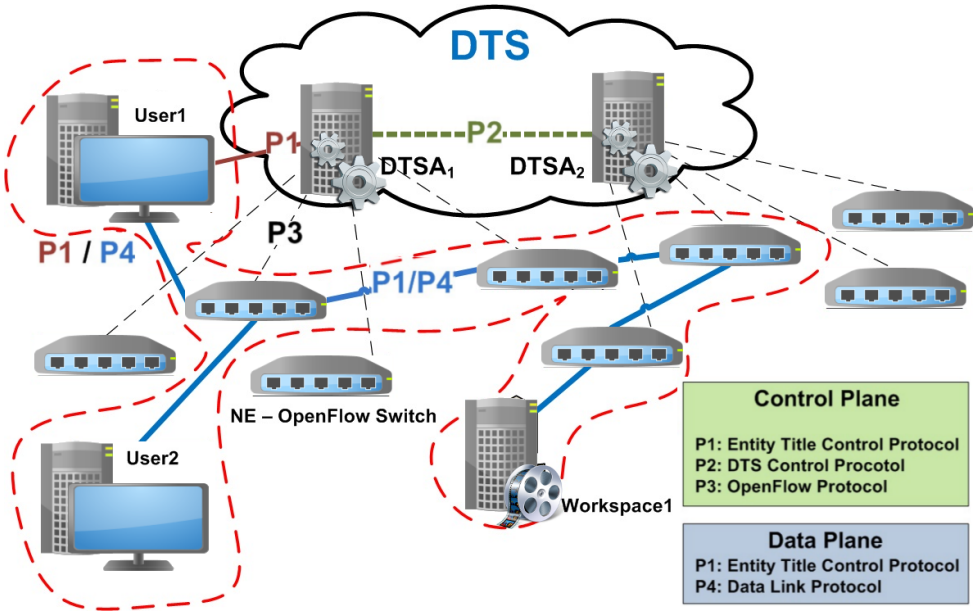
O gerenciamento de *workspaces* e as necessárias configurações de *enlaces lógicos* são responsabilidades do DTS, como já apresentado na secção anterior. Dessa maneira, a alteração da configuração da rede, necessária para cenários de mobilidade e inclusão de novos membros, é transparente para as *entidades* participantes, o que contribui para a visão da rede sobre um ponto de vista *content-centric*.

Em casos onde o plano de controle ocorre numa estratégia *in-band*, é utilizado um tipo específico de *workspace* chamado de *workspace de controle*. Nele, são transmitidos apenas dados relacionados ao controle da rede e podem ser estabelecidos entre elementos de redes e DTSAs ou na interconexão de DTSAs. Esse tipo de *workspace* não é acessível às *entidades* normais e utiliza técnicas específicas de segurança, como criptografia das mensagens e troca de certificados.

O *workspace* é identificado por um tópico, representado por meio de um *título*, assim como ocorre na identificação das *entidades*. Ele representa uma visão lógica que é trans-

formada em uma configuração que abrange elementos de rede, possivelmente passando por túneis, roteamentos convencionais, e diferentes tipos de redes, tanto *wired* quanto *wireless*. Além disso, o *workspace* define níveis de visibilidade, correspondente aos níveis utilizados na distribuição do DTS. Dessa maneira, torna-se possível limitar o acesso de um determinado *workspace* à porções da rede. É possível definir também se o *workspace* é público ou privado para acesso das *entidades*. Um *workspace* público pode ser acessado por qualquer *entidade* interessada, já em um *workspace* privado, a *entidade* precisa ser autorizada pelo criador do *workspace* (ou outras *entidades* administradoras), à fazer parte do grupo.

A Figura 3.3 apresenta uma visão sobre como o *workspace* funciona. Nela é possível identificar duas porções da rede, controladas por dois DTSAs distintos. É calculado e configurado um *workspace*, representado pela linha pontilhada, e é utilizado na interconexão do provedor de conteúdo e dos usuários User1 e User2.



Inspirada em [Oliveira Silva 2013]

Figura 3.3: Visão do workspace sobre a rede.

3.6 Camadas e Protocolos de Comunicação

Boa parte do sucesso da arquitetura Internet se deve à adoção do modelo de resolução de problemas em camadas (Layer), também adotado pelo Modelo de Referência OSI, que é muito eficiente para garantir o desacoplamento de problemas ortogonais, a partir de interfaces e serviços bem definidos. Deve-se lembrar que as redes de computadores atuais são essencialmente baseadas em protocolos, cuja interconexão é realizada por processos pares [Day 2008].

O padrão de camadas é o ponto chave para a "interconexão de sistemas abertos", pois permite a aplicação do conceito de *Separation of Concerns* (SoC) de Dijkstra [Dijkstra 1982]. O SoC é um eficiente recurso para modularização, pois facilita a evolução de cada módulo (ou camada) isoladamente. Essa modularização é viabilizada pela utilização de princípios filosóficos bem definidos para as camadas, o que contribui para a alta coesão dos protocolos que as compõem.

O *Modelo de Título* também é definido em camadas e, embora seja uma abordagem *clean slate*, apenas reexamina as camadas de rede e transporte, definidas pelo modelo de referência OSI [Tanenbaum 2002], através de uma nova camada de Comunicação, que é de fato o foco deste trabalho. Isso simplifica o processo de implantação, uma vez que preserva as camadas física e de aplicação, que foram alvos das maiores evoluções com o crescimento da Internet.

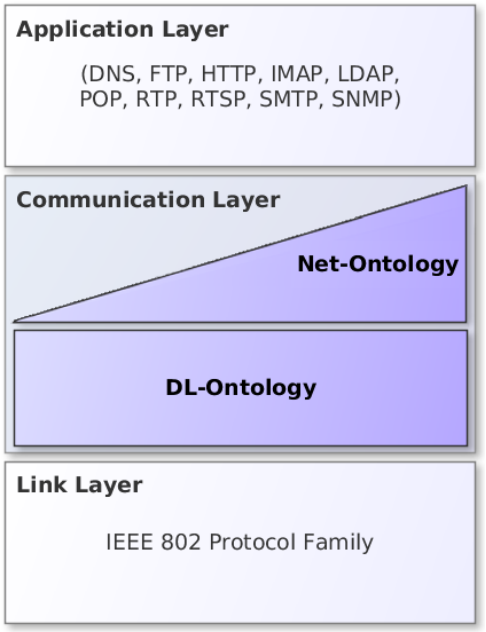


Figura 3.4: Pilha de Protocolos na ETArch.

A Figura 3.4 apresenta as camadas do *Modelo de Título* (*Application*, *Communication* e *Link*), ressaltando as sub-camadas utilizadas na composição da camada de Comunicação, a *DL-Ontology* e a *Net-Ontology*. O formato não usual da subcamada *Net-Ontology* pretende demonstrar a flexibilidade da arquitetura em se adaptar aos requisitos de comunicação das *entidades* de aplicação (*Application Layer*). A ideia é utilizar a *Net-Ontology* de acordo com os requisitos da aplicação. Aplicações com mais requisitos utilizam também mais módulos e funções de rede, enquanto aplicações sem requisitos, ou com o requisito de *baixo overhead*, estabelecem a comunicação diretamente pela subcamada de encaminhamento (*DL-Ontology*). Por exemplo, se dois *hosts* estabelecem um *workspace* para comunicação em uma rede local, não há necessidade da função de roteamento, como acontece atualmente com a arquitetura Internet.

A interface da camada de Comunicação, assim como no Modelo OSI, é baseada no conceito de SAP (*Service Access Point*), o que contribui para o desacoplamento por meio de serviços de interface padronizados. A CSAP (*Communication Service Access Point*) define um contrato de serviços similar à Interface *Socket*, que é amplamente utilizada pela camada de aplicação, diferentemente da TSAP (*Transport Service Access Point*), cujos serviços são encapsulados por esta *interface*. Isso significa que a camada de Comunicação além de prover maior flexibilidade na definição de requisitos comunicacionais por meio de declarações ontológicas, define um SAP mais natural para a utilização das camadas superiores. Dessa maneira, espera-se minimizar o impacto da implantação da arquitetura proposta na camada de aplicação, mantendo uma interoperabilidade com os sistemas legados.

3.7 Protocolos do Plano de Controle

A orquestração da rede, como apresentado na Secção 3.4, é tarefa do DTS, que utiliza os protocolos ETCP e DTSCP respectivamente para a comunicação com *entidades* e DTSAs. Através destes protocolos são realizadas tarefas de controle referentes ao gerenciamento do ciclo de vida das *entidades* e *workspaces*, tais como o *registro de entidades* e a *busca de workspaces*.

Com o controle da rede centralizada no DTS, ocorre uma separação entre os planos de controle e de dados, intimamente ligados à arquitetura ETArch. O desacoplamento destes planos melhora a operação da rede, por flexibilizar a definição de novas políticas e estratégias de configuração da infraestrutura, e por simplificar o encaminhamento das primitivas de rede, através da eliminação da necessidade de cálculo de rotas *on-the-fly*, ou seja, em tempo de execução.

No plano de controle proposto, o *Workspace* (barramento lógico) é estabelecido antes da comunicação em si, similarmente ao controle nas redes de telecom. Um exemplo desse comportamento no mundo real é observado ao iniciar uma viagem. Pode-se realizar um plano de viagem antes de iniciá-la, ou guiar-se quanto as direções corretas durante o percurso. Acredita-se que a primeira situação seja melhor, por facilitar a locomoção, simplificar as vias e evitar erros.

Nas subsecções seguintes serão apresentados os dois protocolos utilizados no plano de controle, bem como os serviços disponibilizados.

3.7.1 ETCP - Entity Title Protocol

O *Entity Title Control Protocol* (ETCP) é o protocolo utilizado na comunicação entre as *entidades* e DTS que as controla. Esse protocolo fornece basicamente serviços para o gerenciamento de *entidades* e *workspaces*, conforme apresentado na tabela 3.1.

Serviço	Descrição
ENTITY-REGISTER	Registra uma *entidade* no DTS através de um título que é analisado no processo de autenticação. Armazena requisitos e capacidades da *entidade*, bem como a sua atual localização e detalhes de acesso.
ENTITY-UNREGISTER	Remove uma *entidade* de todos os *workspaces* dos quais ela faz parte, e remove o seu registro da *Entity Database*.
WORKSPACE-CREATE	Cria um *workspace* na *Workspace Database*, registra o seu *título* através de um processo de autenticação, e registra as capacidades comunicacionais oferecidas às *entidades* que queiram fazer parte do *workspace*.
WORKSPACE-ATTACH	Adiciona uma *entidade* a um *workspace* já existente. Neste estágio, o DTSA precisa modificar o *barramento lógico* para contemplar a nova *entidade*, o que é realizado por meio de um novo cálculo de árvore geradora mínima, e (re)configuração da rede.
WORKSPACE-DETACH	Remove uma *entidade* de um *workspace*, e dispara um processo para a modificação do *barramento lógico* semelhante ao do serviço WORKSPACE-ATTACH, mas com uma *entidade* à menos.
WORKSPACE-DELETE	Remove um *workspace* da *Workspace Database*, e libera os recursos de rede alocados no estabelecimento da comunicação.
WORKSPACE-MODIFY	Modifica as características de um *workspace* já criado, como níveis de acesso ou capacidades comunicacionais.

Tabela 3.1: Serviços do Entity Title Control Protocol (ETCP).

3.7.2 DTSCP - Domain Title Service Protocol

O *Domain Title Service Control Protocol* (DTSCP) é o protocolo utilizado na comunicação entre DTSAs, para consumo interno do DTS. Esse protocolo fornece basicamente serviços para o gerenciamento e registro de DTSAs em Master DTSAs e busca de *workspaces* conforme apresentado na tabela 3.2.

Serviço	**Descrição**
WORKSPACE-LOOKUP	Utilizado para a busca de *workspaces* entre DTSAs e MDTSAs. Recurso pelo qual *workspaces* fora do domínio utilizado podem ser estendidos para acesso.
DTSA-REGISTER	Registra um DTSA em seu respectivo MDTSA, de nível imediatamente superior.
WORKSPACE-ADVERTISE	Insere, remove ou atualiza a *Workspace Database*, indicando que um *workspace* deve ser estendido por determinados agentes, de acordo com os níveis de visibilidade acordados, com o intuito de estabelecer a comunicação entre *entidades* distribuídas em mais de um domínio.

Tabela 3.2: Serviços do Domain Title Service Control Protocol (DTSCP).

Capítulo 4

Aspectos de Projeto e Implementação da Arquitetura ETArch

Este capítulo tem por objetivo apresentar detalhes sobre o projeto e desenvolvimento da Arquitetura ETArch, bem como apresentar bibliotecas e aplicações de apoio utilizadas pelo projeto. Dadas as limitações no teste e implantação de uma abordagem *clean slate*, deve-se ressaltar que o resultado obtido ainda não pode ser utilizado em operação. Entretanto, o desenvolvimento foi realizado em uma estratégia de "protótipo de prova de conceito" (*proof of concept prototype*) [Miller 1990], que é amplamente utilizada na literatura correlata, e fornece um mecanismo simplificado e eficiente para os testes de ideias.

Esta implementação da ETArch visa prioritariamente testar o conceito de *workspace* e como ele pode ser aplicado na obtenção da *agregação de tráfego via multicast*. Os testes conduzidos foram voltados aos cenários que definem a comunicação *multicast* como um requisito central e para isso foram desenvolvidas duas aplicações: uma de *chatting*; e outra de *streaming* de vídeo. A implementação dessas aplicações baseou-se nos conceitos de endereçamento propostos por esta arquitetura, que viabilizam a comunicação *multicast* de uma maneira natural, e sobretudo real, sem redundância desnecessária de dados na rede.

No processo de desenvolvimento, identificou-se a possibilidade de utilização de uma implementação SDN na comunicação com os elementos de rede, pela convergência das ideias propostas. Essa junção gera uma boa perspectiva quanto à interoperabilidade de propostas para as redes futuras, pois contribui para uma padronização factível com foco na separação do plano de controle. Nesta implementação, a Arquitetura ETArch fornece

amplo suporte ao protocolo OpenFlow, que já é suportado pelos principais fabricantes de equipamentos de redes. Outra grande vantagem reside na possibilidade de utilização de *frameworks* e *testbeds* que já dão suporte ao protocolo OpenFlow, o que tem sido muito explorado pela comunidade científica. Nesse contexto, tornou-se possível a experimentação da arquitetura proposta através do *software* de simulação de redes *mininet* [Lantz et al. 2010], e a implantação na *testbed* europeia OFELIA [Suñé et al. 2014], que já foi estendida até o Brasil, pela Universidade Federal de Uberlândia, via projeto EDOBRA[1].

Nas próximas seções serão descritos os detalhes de projeto dos componentes da arquitetura ETArch apresentados no Capítulo 3, bem como aplicações e bibliotecas utilizadas no desenvolvimento e testes.

4.1 Sub-camada Net-Ontology

A modelagem da sub-camada *Net-Ontology* tem quatro principais módulos, projetados com o intuito de fornecer serviços de alta expressividade semântica para as camadas superiores, e um mecanismo para análise de requisitos e orquestração das funções de rede. A Figura 4.1 apresenta esses módulos e como eles se relacionam. Nas próximas subseções são detalhados o funcionamento e a implementação de cada módulo.

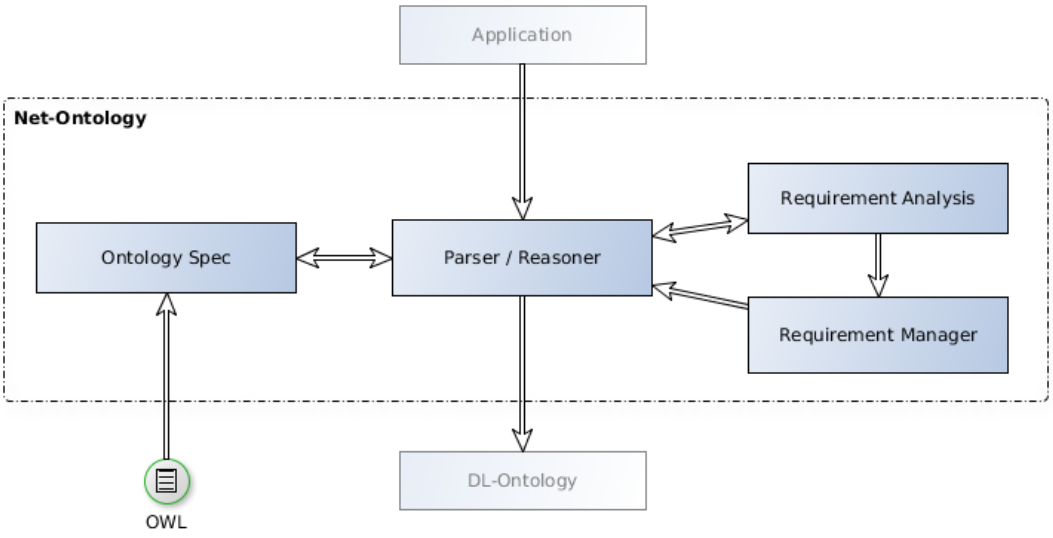


Figura 4.1: Estrutura *Net-Ontology*.

4.1.1 Ontology Spec

O módulo *Ontology Spec* é responsável pela modelagem da *ontologia* que representa a rede na Arquitetura ETArch. Esse conceito já foi apresentada na Seção 2.4.7 e representa

[1]http://www.fp7-ofelia.eu/ofelia-facility-and-islands

uma descrição formal da rede através de uma linguagem de alto nível, que é apresentada no Anexo A.

A modelagem das estruturas e entidades de rede foi realizada com o intermédio da ferramenta Protégé [Horridge et al. 2012], versão 4.0.1. Esse *software* é bastante útil na descrição formal de "seres", e por esse motivo, foi utilizado na criação da *ontologia* de rede utilizada. Ele suporta a extração de dados modelados em diferentes formatos, dentre eles o OWL, que foi utilizado neste módulo.

Com a *ontologia* descrita em OWL foi criada uma API em linguagem JAVA através da OWL API [Horridge et al. 2007], versão 3.0.0.1451. Essa API descreve com fidelidade o modelo de dados desenhado e foi utilizada na concepção de diversas outras componentes da arquitetura, principalmente na implementação do DTSA.

4.1.2 Parser/Reasoner

O módulo de *Parser/Reasoner* é responsável pela interpretação da descrição ontológica, realizada em OWL e modelada no módulo *Ontology Spec*. Trata-se de um *motor de inferências*, cujo papel é decisivo na flexibilização da rede, por proporcionar um método mais eficiente para o "entendimento" das necessidades da aplicação. Embora esse módulo ainda precise ser aprimorado, nesta etapa foi utilizada uma estratégia *descritiva* e sem inferências, baseada unicamente na análise léxica das definições de requisitos por meio de *expressões regulares*.

É importante lembrar que tal implementação não fornece alta capacidade de *interpretação semântica* na interface da camada de Comunicação como proposto na arquitetura. Todavia, este módulo encontra-se em estágio de evolução por meio de uma implementação utilizando a API JENA [Wei et al. 2010]. Além disso, o requisito de comunicação *multicast*, que é o foco deste trabalho, não requer serviços adicionais para a efetividade da comunicação, pois se beneficia naturalmente das inovações nos aspectos de endereçamento apresentadas na Arquitetura ETArch.

4.1.3 RAM - *Requirement Analysis Module*

O módulo RAM de *Análise de Requisitos* é responsável por manipular os requisitos da aplicação, que são obtidos a partir da interpretação da descrição OWL realizada pelo módulo *Parser/Reasoner*, e por combiná-los através da *Lógica de Leśniewski's*, proposta em [Leśniewski 1930]. Esse mecanismo é sensível a mudanças de contexto que ocorrem durante a comunicação e, a partir de uma análise de expressões lógicas, encontra quais recursos tecnológicos relacionados às funcionalidades requiridas devem ser acionados.

Como exemplo, vamos supor que um serviço $S1$, num momento t_1, requeira o estabelecimento da comunicação com o serviço $S2$, definindo um requisito específico. O RAM verifica se este requisito pode ser atendido através dos recursos tecnológicos $R1$ e $R2$. Em

outro momento, t_2, $S1$ deseja melhorar a segurança da comunicação através do requisito de *confidenciabilidade*. Para isso, torna-se necessária a utilização do recurso tecnológico $R3$. Estes cenários são interpretados e representados pelos seguintes axiomas:

$$S1S2t_1 \rightarrow R1 \wedge R2 \tag{4.1}$$

$$S1S2t_2 \rightarrow (R1 \wedge R2) \wedge R3 \tag{4.2}$$

Observe-se, então, que se pode perceber uma reconfiguração importante da arquitetura, em termos dos módulos e funcionalidades requeridos em um determinado contexto, através de alterações na especificação ôntica.

4.1.4 RMM - *Requirement Manager Module*

O módulo RMM de *Gerenciamento de Requisitos* é responsável por interpretar as regras de requisitos definidos pelo módulo de análise e por acionar os algoritmos relacionados às funcionalidades requeridas, de forma a aplicar as alterações na rede necessárias para transmissão de dados. Esses algoritmos operam de acordo com os papéis praticados pelos elementos de rede, aplicando comportamentos específicos ao contexto de utilização relacionados à estratégia implantada. Por exemplo, no atendimento do requisito de segurança, uma abordagem possível consiste em criptografar os dados na origem e descriptografá-los no destino. Dessa maneira, os elementos centrais da rede não realizam nenhum tipo de tratamento nas primitivas encaminhadas, apenas os de borda. Em outro exemplo, no atendimento do requisito de QoS, uma possível abordagem reside na utilização de algoritmos de priorização e de garantia de banda nos elementos centrais.

Utilizando o exemplo citado na seção anterior, no momento t_1, o RMM recebe os requisitos $R1$ e $R2$ do RAM. Ele então utiliza o módulo *Ontology Spec* para adicionar fragmentos de OWL na primitiva, declarando que P_1 tem os requisitos R_1 e R_2, e depois aciona os algoritmos implantados A_1 e A_2 para o processamento inicial.

4.2 Sub-camada DL-Ontology

A arquitetura da sub-camada *DL-Ontology* se baseia em dois módulos principais, responsáveis essencialmente pelo transporte das primitivas de rede. Esta sub-camada utiliza *título* como identificador de fluxo e é responsável pelo encaminhamento dos dados destinados a um *workspace*, e consequentemente para todas as *entidades* participantes. Quando verificada a existência de fragmentos ontológicos inseridos pela instância da sub-camada *Net-Ontology*, esta é acionada antes do efetivo encaminhamento, para garantir que os requisitos comunicacionais sejam cumpridos. A Figura 4.2 apresenta esses módulos, e como

eles se relacionam. Nas próximas subsecções, será detalhado o funcionamento de ambos os módulos.

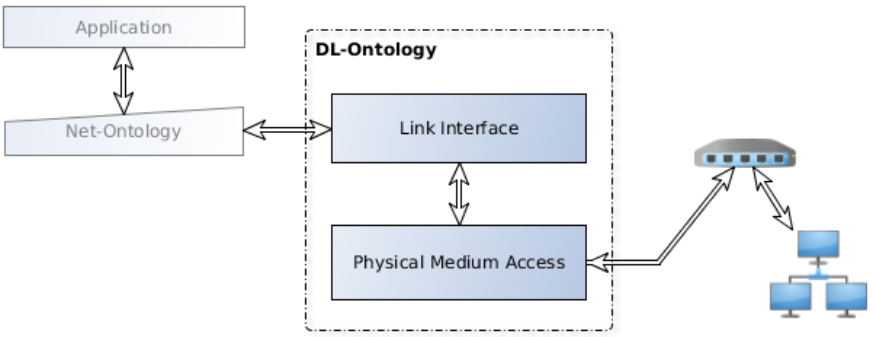


Figura 4.2: Estrutura *DL-Ontology*.

4.2.1 Link Interface

O módulo de *interface* de enlace é responsável pelo aprovisionamento das funções para o envio e recepção de primitivas. Ele fornece uma interface que visa o atendimento dos serviços definidos pela sub-camada *DL-Ontology* e que são acionados tanto pela camada de aplicação, quanto pela sub-camada *Net-Ontology*.

No atual estágio, foi desenvolvida uma biblioteca em linguagem JAVA, que utiliza o módulo de acesso ao meio físico através de uma abstração de alto nível. Essa interface se baseia nos recursos da API JNI (*Java Native Interface*), o que torna transparente a implementação de baixo nível dos métodos e simplifica o desenvolvimento de aplicações complexas. Para fins de interoperabilidade, essa API define uma classe chamada *FinSocket* que é bastante similar à classe *Socket*. O contrato com a camada de aplicação foi parcialmente mantido, no entanto, alguns novos conceitos foram introduzidos para o suporte à *DL-Ontology*.

4.2.2 Physical Medium Access

O módulo de acesso ao meio físico é responsável pelo acesso direto à interface de rede, o que se realiza por meio de módulos do sistema operacional ou *drivers* da placa de rede. Trata-se de um módulo que abstrai a relação entre o meio físico e a camada de Comunicação a partir de serviços de baixo nível.

Esse módulo foi desenvolvido em linguagem C, através da API de *Raw Sockets* da interface *Socket* para permitir a eliminação da pilha de protocolos da arquitetura TCP/IP, de forma a permitir acesso direto às interfaces de rede, como pode ser visto na Figura 4.3. Como a sub-camada MAC também foi eliminada na Arquitetura ETArch, tornou-se necessário impedir o funcionamento dos filtros MAC nas interfaces de rede, como apresentado na Seção 2.5. Isso é realizado a partir da configuração da interface de rede para operação

em *modo promíscuo*, o que requer execução nos sistemas Linux em modo *super-usuário*.

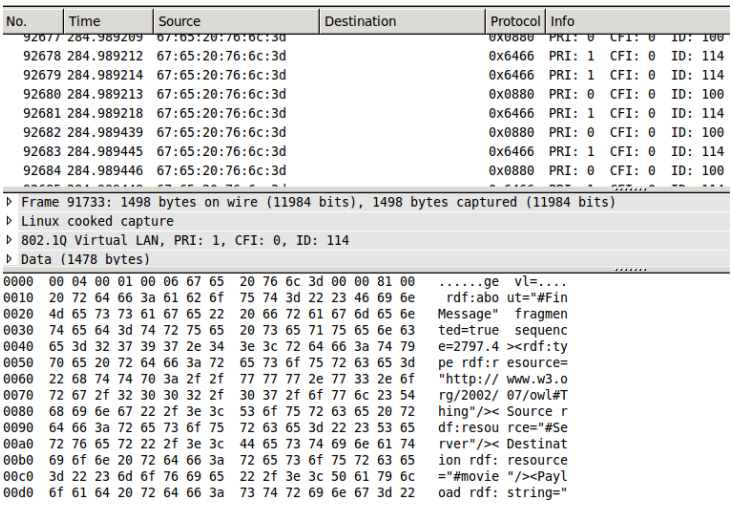


Figura 4.3: Captura das primitivas de rede na Arquitetura ETArch.

Embora tal implementação não seja viável em aplicações de larga escala, ela se mostrou muito eficiente no alcance do objetivo de experimentação da arquitetura, não impactando nos resultados obtidos. A versão atual desse desenvolvimento pode ser conferido no Apêndice C.

4.3 Biblioteca FINLAN

O termo FINLAN é um acrônimo para *Fast Integration of Network Layers* e implementa o plano de dados do *Modelo de Título*. Esse conceito nasceu no ano de 2006, com o início do grupo MEHAR. Por esse motivo a biblioteca utilizada no desenvolvimento de aplicações na Arquitetura ETArch foi chamada de *Biblioteca FINLAN* e reúne a implementação dos protocolos *DL-Ontology* e *Net-Ontology*, que são utilizados no plano de dados; e dos protocolos ETCP e DTSCP, utilizados no plano de controle.

A implementação dos protocolos do plano de controle foi realizada através da linguagem de programação JAVA. Nenhuma biblioteca externa foi utilizada nesse estágio de implementação, contudo acredita-se que em uma etapa posterior, soluções de melhor desempenho possam ser exploradas.

Nesta biblioteca são definidos os serviços, as primitivas e os autômatos relacionados aos protocolos ETCP e DTSCP, descritos na Seção 3.7. As primitivas destes protocolos foram definidas através de mecanismos padrões de serialização de dados JAVA e o desenvolvimento se baseou no padrão *Observer* definido pelo GoF (*Gang of Four*) [Gamma et al. 1994] em seus *Design Patterns*. Esse padrão facilita a utilização desta biblioteca, o que é útil na implementação tanto do DTSA, quanto de aplicações usuárias.

4.4 DTSA - *Domain Title Service Agent*

Como apresentado na Seção 3.4, o DTS é um sistema distribuído composto por agentes divididos em níveis, cuja principal função é gerenciar o ciclo de vida das *entidades* e dos *workspaces*. Baseando-se na descrição da Seção 2.6.3, percebe-se que há uma certa similaridade entre a tarefa de gerenciamento de fluxos do *Controlador OpenFlow* e a de gerenciamento de *workspaces* do *Agente DTS*. Tal similaridade se manifesta pela natureza congruente das abordagem ETArch e SDN, que defendem pontos em comum como o da separação do plano de controle e o de dados. Também no início deste capítulo foi defendida a utilização do protocolo OpenFlow, com o intuito de facilitar a implantação e a realização de testes da abordagem proposta. Dessa maneira, a implementação do DTSA utilizada neste trabalho se baseia na extensão do controlador *Floodlight* [Big Switch 2012], que fornece suporte ao protocolo OpenFlow versão 1.0 através de uma implementação em linguagem JAVA.

A extensão do controlador *Floodlight* se deu por meio de uma reimplementação da Classe *Controller*, denominada por *OpenflowDtsController*. Esse controlador customizado utiliza módulos nativos do *Floodlight*, que são necessários para o funcionamento básico do controlador, e um módulo customizado chamado *DTSModule*, que é responsável pela instanciação do módulo principal do DTSA, o *Core Module*. Além disso, o *DTSModule* é responsável por "estimular" o DTSA, que opera em modo passivo, com informações sobre topologia e mensagens recebidas da rede. Também foi criado o *OpenFlow Connector*, que é acoplado ao barramento *Network Equipment Connector* (NEC), responsável pela comunicação com a rede, e que é utilizado sobretudo na configuração de *workspaces*.

O DTSA adota uma estratégia baseada na utilização de módulos, que é apresentada em detalhes na Figura 4.4. Esses módulos foram implementados segundo os padrões de *design* definidos pelo GoF, a partir de mecanismos nativos de modularização JAVA. Os detalhes sobre a implementação desses módulos, bem como a relação deles com o controlador *Floodlight*, são apresentados nas subsecções seguintes.

4.4.1 Core Module

O módulo de *Core* é o núcleo do sistema e é responsável pela instanciação e acionamento dos demais módulos. Ele utiliza parâmetros de inicialização que guiam o seu curso de operação, tais como nível de ação, título e modo de operação (ativo ou reativo). No modo ativo, o DTSA realiza buscas de topologia, registra os elementos de rede e cria um *workspace de controle*. No modo reativo, o DTSA recebe essas informações a partir de um agente externo. O nível de ação se refere à posição hierárquica (*tier*) que o DTSA opera, sendo o nível local o mais baixo, e que lida diretamente com os elementos de rede. Nesse contexto, o DTSA utilizado nesse trabalho foi instanciado no nível local e em modo

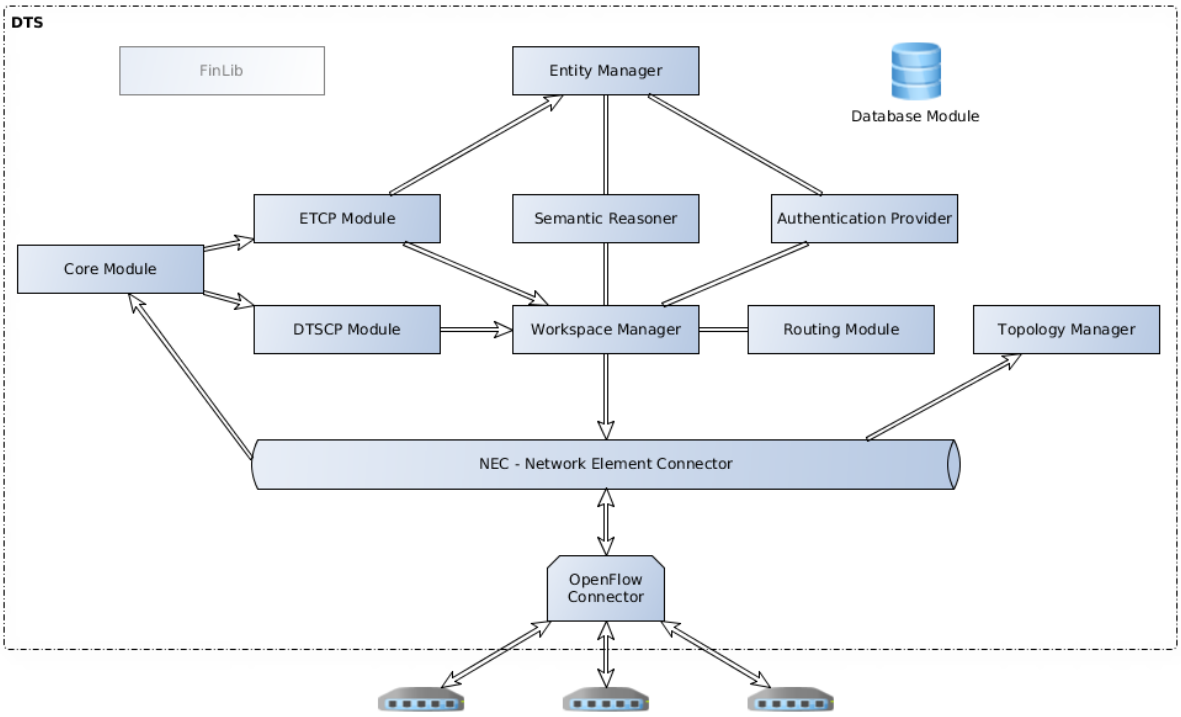


Figura 4.4: Estrutura DTS.

reativo, sendo estimulado por uma extensão do controlador *OpenFlow Floodlight*.

Este módulo também define o modelo de dados utilizado, que é derivado do modelo de Ontologia trabalhado na implementação do protocolo *Net-Ontology*, e é utilizado por todos os demais módulos do sistema. Além disso, ele também define a lógica de funcionamento do DTSA e como os módulos são utilizados.

São definidos pontos de alteração do comportamento por meio do padrão *Observer*, similares aos *joint points* do paradigma orientado à aspectos [Kiczales et al. 1997], o que permite a alteração do comportamento do DTSA de maneira pouco intrusiva. Atualmente a maioria dos módulos são acoplados através do padrão *Observer*, dentre eles o *Workspace Manager* e *Topology Manager*. Outro padrão utilizado no projeto do DTSA é o IoC (*Inversion of Control*) [Johnson and Foote 1988], aplicado no conceito de *Dependency Injection*. É dessa maneira que o mecanismo de *logging* e o módulo *Database Module* são acoplados à aplicação.

4.4.2 ETCP/DTSCP Module

Os módulos ETCP e DTSCP são responsáveis pela interpretação das mensagens e pela execução dos autômatos dos protocolos homônimos pelo DTSA, que são apresentados na Figura 4.5. Todos os serviços são mapeados através da *Biblioteca FINLAN*, que define também as primitivas de rede. Outros módulos do sistema são acionados de acordo com os serviços solicitados. Por exemplo, numa mensagem de ENTITY_REGISTER, o *ETCP Module* aciona o módulo *Entity Manager*, que realiza a autenticação a partir do módulo

Authentication Provider, e caso seja bem sucedido, o registro da *entidade* é concluído com uma mensagem de sucesso.

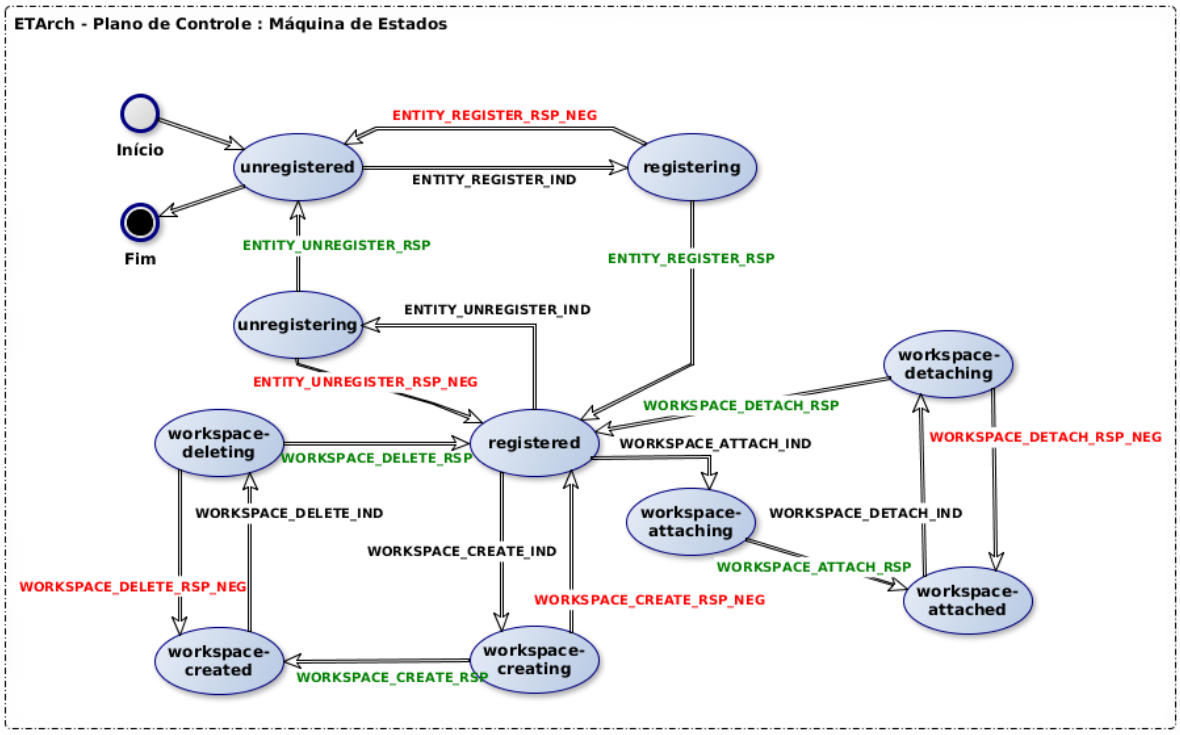


Figura 4.5: Autômato Finito que descreve o Plano de Controle na Arquitetura ETArch.

Na estratégia de implementação do DTSA como um super conjunto de um controlador OpenFlow, foi definida a utilização do serviço OFPT_PACKET_IN para o encapsulamento das primitivas de controle. Um fator que corrobora com essa abordagem está no comportamento padrão dos *switches* OpenFlow, que enviam as mensagens recebidas via OFPT_PACKET_IN para o controlador quando não há fluxos com regras que combinem.

Outra possibilidade reside na configuração de uma regra que case com todas as mensagens de controle e as direcione ao controlador. Por fim, é possível a criação de um *workspace de controle*, conforme proposto na Seção 3.5, que é estabelecido via configuração inicial dos elementos de rede, e que destinam as mensagens recebidas ao DTSA responsável pelo controle do domínio de rede. Na implementação atual é utilizada a primeira abordagem apresentada, que explora o comportamento padrão dos *switches OpenFlow*. Da mesma maneira, as mensagens enviadas às *entidades* utilizam o serviço OFPT_PACKET_OUT, que é utilizado também para o retorno de respostas em casos de requisições de serviços síncronos.

4.4.3 Database Module

O módulo *Database Module* é responsável pela definição e implementação das estratégias de persistência, que envolvem mecanismos de armazenamento e de controle de

transações. Esse módulo é acoplado no sistema via *Dependency Injection* e fornece serviços para armazenamento e consulta de dados através da implementação da interface *GenericStorage*.

Abordagens de alto desempenho no armazenamento e acesso a dados se fazem necessárias, por influenciarem diretamente na latência de rede durante o estabelecimento da conexão. Além disso, a base deve ser capaz de comportar dados de toda a rede, bem como dados referentes às *entidades* e *workspace*. Por esse motivo, uma abordagem possível consiste na utilização de bancos de dados não-relacionais (NoSQL) [Huang and Luo 2014], que são comumente utilizados em aplicações que armazenam grandes quantidades de dados em estruturas simples, como o *Google* e o *Facebook*. Entretanto, não é descartada a utilização de um banco de dados convencional, com um modelo de dados relacional, como *Oracle* e *PostgreSQL*. Essa abordagem traz o benefício de utilizar padrões de mercado consolidados e com desempenho aceitável, apesar das dificuldades de escalamento.

Nesse ponto do projeto, as estratégias utilizadas na implementação desse módulo são simples e baseiam-se somente na utilização de recursos nativos da linguagem JAVA, como armazenamento em memória e sincronização dos métodos e atributos para acesso paralelo.

4.4.4 Topology Manager

O módulo *Topology Manager* é responsável por prover informações relacionadas a infraestrutura de rede, que são armazenadas através do *Database Module*, e são acessadas pelos demais módulos, como por exemplo, pelo *Workspace Manager*.

Normalmente essas informações são obtidas a partir de protocolos de *discovery*, como o protocolo aberto LLDP (*Link Layer Discovery Protocol*), ou através de protocolos particulares, como o CDP (*Cisco Discovery Protocol*). Esses protocolos são suportados pelos elementos de rede, que armazenam as informações de topologia em tabelas internas. Essas tabelas são consultadas a partir de diferentes estratégias e a mais comum consiste na utilização do protocolo SNMP, através de MIBs (*Management Information Base*) especializadas em informações de topologia.

Também é possível recuperar as informações de topologia a partir do acesso direto ao *prompt* de comando dos equipamentos, realizado a partir dos protocolos SSH (*Secure Shell*) ou Telnet. Nesse caso são utilizados "robôs" de software especializados no *software* proprietário dos equipamentos. Alguns equipamentos também disponibilizam uma interface gráfica WEB, ou mesmo serviços (REST ou SOAP), que fornecem abstrações de alto nível para recuperação de dados de topologia. Por fim, alguns fornecedores disponibilizam protocolos proprietários para a comunicação com os equipamentos de rede, normalmente acessados a partir de programas específicos de gerenciamento.

Na implementação deste trabalho, o módulo de topologia baseou-se no serviço *TopologyService* do *Floodlight*. Essa integração se deu a partir da instanciação do módulo

TopologyImpl no controlador estendido OpenFlow. Esse módulo envia mensagens LLDP diretamente para os elementos de rede, que respondem com informações próprias e dos vizinhos. Esses dados são persistidos através do módulo *StorageSource* do *Floodlight*.

O *DTS Module*, que é um módulo utilizado no controlador estendido OpenFlow, define um *listener* de eventos de alteração da base de topologia, através da classe *IStorageSourceListener*. São interceptadas as alterações das estruturas *controller_switch*, *controller_port*, *controller_host*, *controller_link* e *controller_hostattachmentpoint*. Essas informações são interpretadas e enviadas ao DTS através do *Topology Module*, que opera em modo passivo. Por fim, o módulo de topologia manipula os dados recebidos, que posteriormente são persistidos através do *Database Module*.

4.4.5 Authentication Provider

O módulo *Authentication Provider* é responsável pela verificação da identidade das *entidades* que pleiteiam um registro no DTSA. O mesmo conceito de validação se estende ao registro de *workspaces*, cujos títulos guiam as *entidades* no estabelecimento da comunicação. Este módulo é utilizado pelos módulos *Entity Manager* e *Workspace Manager*, sempre que realizado um novo pedido de registro de *título*.

O módulo de autenticação é de suma importância no aspecto de segurança da Arquitetura ETArch. Ele é responsável por prover o registro inequívoco das *entidades*, garantindo a integridade das aplicações e evitando fraudes, o que não é um problema recente na Internet. Um dos principais métodos de fraude utilizados hoje em dia é a utilização de nomes de domínios dúbios, que enganam os usuários mais desatentos, e colhem informações privadas que são utilizadas em outros golpes. A solução aplicada à WEB consiste na utilização de certificados digitais, dados criptografados e registro de domínios, que possivelmente serão mantidos nas redes futuras.

Uma abordagem que vem sendo trabalhada em uma linha de pesquisa paralela no grupo MEHAR é a de autenticação por reputação. Essa técnica converge com os objetivos da Arquitetura ETArch, por fornecer um mecanismo seguro e transparente para o registro de títulos. A ideia se baseia na reputação dos provedores de conteúdo, calculado a partir de métricas de acesso e análise de variáveis culturais dos usuários. Um conceito semelhante já é utilizado em aplicações WEB de busca, como o Google, que apresentam primeiramente os resultados com melhores "reputações", mesmo que sem o uso explícito desse termo.

Nesse trabalho, os aspectos de segurança e autenticação de títulos foram postergados para trabalhos futuros, por estarem fora do foco tratado, que consiste unicamente na utilização da Arquitetura ETArch na agregação de tráfego via *multicast*. Por isso, o módulo *Authentication Manager*, no estágio atual, apenas impede o registro de *entidades* e *workspaces* com títulos já utilizados. Embora tal implementação não seja viável em larga escala, ela não compromete o resultado almejado por esse trabalho.

4.4.6 Semantic Reasoner

O módulo *Semantic Reasoner* é utilizado para a interpretação e combinação dos requisitos da *entidade*, bem como nos registros de suas capacidades. A sua implementação se baseia na utilização da *Biblioteca FINLAN*, que reúne módulos responsáveis pela materialização do modelo ontológico da rede e pela interpretação semântica de mensagens dentre outras funcionalidades, e que também é utilizada na implementação da instancia da subcamada *Net-Ontology*. Esse módulo fornece expressividade semântica na definição de *entidades* e de *workspaces*, que expressam suas *capacidades* e *requisitos* em linguagens de alto nível semântico.

Esse módulo utiliza os módulos *Reasoner*, RAM e RMM, cujos detalhes de implementação já foram descritos respectivamente nas Secções 4.1.2, 4.1.3 e 4.1.4. O resultado dessa análise é utilizada pelo módulo *Workspace Manager*, que realiza alterações nos elementos de rede para atender os requisitos definidos, e também pelo módulo *Entity Manager*, que persiste as informações sobre as capacidades das *entidades*.

4.4.7 Entity Manager

O módulo *Entity Manager* é responsável por gerenciar o ciclo de vida das *entidades*. Para tanto, ele mapeia os serviços relacionados ao plano de controle das *entidades* fornecidos pelo módulo *ETCP Module*, em ações definidas num esquema de máquina de estados. O tratamento desses serviços ocorre conforme descrito abaixo:

ENTITY_REGISTER

Inicialmente é realizada uma verificação sobre o estado da *entidade* através do *título* informado. Caso a *entidade* já esteja registrada, ou tenha sido suspensa por alguma irregularidade (listada na *black-list*), o registro é negado. Caso contrário, o processo de autenticação do *título* da *entidade* é realizado por meio do módulo *Authentication Provider*, que opera da maneira descrita na Seção 4.4.5. Depois de validada a titularidade, é analisada a descrição em OWL das *capacidades/requisitos* da *entidade* a partir do módulo *Semantic Reasoner*. Por fim, os dados referentes à *entidade* são persistidos através do módulo *Database Module* e é retornada uma DTS_MESSAGE de sucesso.

ENTITY_UNREGISTER

Assim como no processo de registro, primeiramente é verificado o estado da *entidade* que precisa estar em estado ativo para ser apta à remoção. Depois disso, ocorre o processo já utilizado de autenticação do *título* da *entidade* através do módulo *Authentication Provider*. Comprovada a autenticidade da *entidade*, verifica-se se ela está ligada a algum *workspace* e posteriormente é realizada a remoção da entidade desse *workspace*. Essa remoção pode gerar diversas pequenas alterações na rede e

tal processo é disparado de forma paralela. Só então o ação de remoção da *entidade* é concluída e a base de dados é atualizada através do módulo *Database Manager*.

Nos dois processos apresentados acima, caso alguma das validações falhe ou caso algum erro ocorra, é retornada uma mensagem de erro à *entidade* solicitante, através do módulo *NEC*. Trata-se de uma DTS_MESSAGE, com status de falha e um código de erro. A mensagem é destinada ao elemento de rede no qual a *entidade* está conectada, que por sua vez encaminha a mensagem à própria *entidade*.

4.4.8 Workspace Manager

O módulo *Workspace Manager* gerencia o ciclo de vida de *workspaces*, através de uma máquina de estados que implementa a lógica de tratamento desse conceito. Esse módulo é acessado a partir dos módulos *ETCP Module* e *DTSCP Module*, que fornecem serviços para a manipulação de *workspaces* no plano de controle. Esses serviços foram tratados conforme descrito abaixo:

WORKSPACE_CREATE

Inicialmente é verificado o estado da *entidade* através do módulo *Entity Manager*. Caso a *entidade* esteja registrada e ativa, é realizado um teste de autenticidade a partir do módulo *Authentication Provider*, por questões de segurança. Sendo comprovada a identidade da *entidade*, é realizado um processo de autorização do título do *workspace*, que também utiliza o módulo *Authentication Provider*. Esse processo de autorização pode utilizar um esquema de registro, semelhante ao utilizado hoje em dia, mas sem as limitações do endereçamento IP. Depois de autorizado o *título* do *workspace*, o módulo *Semantic Reasoner* é utilizado no processamentos dos requisitos da comunicação e, por fim, o *workspace* é criado (salvo) através do *Database Module*. É importante ressaltar que nenhuma alteração é realizada na rede nesse estágio, o que significa que o processo de criação do *workspace* representa mais um processo de registro que de configuração de fato.

WORKSPACE_ATTACH

Nesse processo é realizada uma validação no estado e autenticação da *entidade*. Em caso de sucesso, são recuperados os *requisitos* da *entidade* que são *comparados* com as *capacidades* do *workspace* através dos módulos *Database Module* e *Semantic Reasoner*. Se for comprovada a capacidade da *entidade* de participar do *workspace*, é iniciado o processo de reconfiguração da rede. Primeiramente é calculada a melhor rota para a ligação das *entidades* que compõem o *workspace* a partir do *Routing Module*. Esse caminho então é submetido à rede, a partir do módulo *NEC*, que o traduz em configurações nos equipamentos de rede. Por fim, a base é atualizada a partir do módulo *Database Module*.

WORKSPACE_DETACH

Primeiramente é validado o estado e a autenticidade da *entidade*. Além dessas validações, também é verificado se a *entidade* realmente faz parte do *workspace* do qual ela almeja sair. Depois disso, é recalculado a melhor rota para a ligação das *entidades* remanescentes que compõem o *workspace* a partir do *Routing Module*. Assim como no processo de inscrição, o novo caminho é submetido à rede a partir do módulo *NEC*, que o traduz em configurações nos equipamentos de rede. Depois disso a base é atualizada a partir do módulo *Database Module*.

WORKSPACE_REMOVE

Novamente é validado o estado e a autenticidade da entidade. Depois é verificada se a *entidade* está registrada como administradora do *workspace*, que é um pré-requisito para a sua desativação. Em caso afirmativo, a rota de encaminhamento do *workspace* é recuperada a partir do *Database Module* e é utilizada para remover a configuração da rede, através do módulo *NEC*. Por fim, a base é novamente atualizada a partir do módulo *Database Module*.

WORKSPACE_MODIFY

Também é validado o estado e a autenticidade da *entidade*. Posteriormente é verificado se a *entidade* está registrada como administradora do *workspace*, assim como no processo de remoção. Em caso positivo, as alterações solicitadas nos atributos do *workspace* são processadas. Em casos de mudança de visibilidade, as regras passam a valer para as próximas solicitações de inscrição. Em casos de mudança dos níveis de ação, caso o nível aplicado seja inferior ao nível do DTSA, todas as *entidades* vinculadas a partir de outros DTSAs do mesmo nível são removidas do *workspace*. Em casos de mudanças na titularidade, é realizado um novo processo de autorização no título. No caso de mudança da *entidade* adinistradora, a alteração é realizada desde que a nova *entidade* administradora esteja inscrita no grupo. E por fim, na mudança de requisitos, é feita uma reavaliação dos requisitos pelo módulo *Semantic Reasoner*, que os campara com as *capacidades* das *entidades* inscritas no *workspace*. Caso as *entidades* não satisfaçam aos novos *requisitos* elas são removidas do *workspace*. Neste ponto, caso *entidades* tenham sido removidas do *workpaces*, ou novos *requisitos* tenham sido definidos, o módulo *Routing Module* recalcula o caminho ótimo para as *entidades* inscritas que é utilizado na configuração da rede através do módulo *NEC*. Todas as alterações são então persistidas via *Database Module*.

WORKSPACE_LOOKUP

É realizada uma busca no banco de dados com os critérios de buscas definidos na mensagem de *lookup*. Os *workspaces* podem ser filtrados por quaisquer dos seus atributos e os valores podem ser definidos através do uso de *wild-cards* [.*?], o que flexibiliza a busca. Os resultados da busca são retornados através de uma DTS_

MESSAGE ao emissor, seja ele um DTSA ou uma *entidade*. Apenas *workspaces* públicos são retornados na busca quando o solicitante é uma *entidade*. Caso seja um DTSA, nenhum resultado é omitido.

WORKSPACE_ADVERTISE

Esse serviço é utilizado na extensão do *workspace* para outros domínios, bem como a alteração e revogação dessa extensão. Nesse caso, a mensagem recebida contém informações sobre o tipo de ação e os dados para a sua execução. No caso de extensão são definidos os dois pontos extremos da rede no qual o *workspace* deve ser estabelecido. Os *requisitos* são analisados a partir do módulo *Semantic Reasoner*. O melhor caminho possível é calculado a partir do módulo *Routing Module* e a configuração é realizada na rede a partir do módulo *NEC*. Os dados são então persistidos na base de dados a partir do módulo *Database Module*. No procedimento para a remoção dos dados referentes ao registro do *workspace*, as informações são recuperadas da base a partir do módulo *Database Module*, e são utilizadas mara a remoção da configuração da rede a partir do módulo *NEC*. O procedimento de modificação ocorre de maneira similar ao da tratativa do serviço WORKSPACE_MODIFY.

Em todos os processos apresentados acima, caso alguma das validações falhe ou caso algum erro ocorra é retornada uma mensagem de erro à *entidade* solicitante. Isso é realizado através de uma DTS_MESSAGE que é enviada através do módulo *NEC*. A mensagem é destinada à *entidade* requisitante, através do elemento de rede no qual a *entidade* está conectada.

4.4.9 Routing Module

O módulo de roteamento é responsável pela escolha do melhor caminho que envolva todas as *entidades* participantes de um *workspace*. Esse caminho é tradicionalmente encarado como uma árvore geradora mínima, no qual os vértices são os elementos de rede utilizados como acesso pelas *entidades*.

As melhores estratégias para o cálculo desse caminho envolvem a utilização de um dos algoritmos clássicos da área. São eles o *Prim* de Robert Clay Prim e o *Kruskal* de Joseph Kruskal [Cormen et al. 2009]. O problema dessas abordagens está na estratégia *gulosa* que eles adotam, o que os tornam algoritmos não resolvíveis em tempo polinomial (classe NP). No entanto, abordagem de divisão de porções da rede e utilização de diferentes níveis simplifica esse processo e viabiliza a aplicação desses algoritmos. A estratégia consiste em limitar a visibilidade da rede em cada nível de ação. Dessa maneira, a rede sempre aparenta ser mais simples do que de fato é. Ao invés de um grafo único para a representação da rede, são utilizados pequenos grafos por níveis, que apresentam apenas a

visão simplificada da rede visível no nível. A simplificação consiste em mapear domínios, e tuplas de *switches* e portas, utilizadas na interconexão desses domínios.

Em cenários reais, as principais ramificações da rede estão no nível local. No core da rede há poucas variações e reduzidas possibilidades de links. Dessa maneira, a estratégia utilizada na implementação desse módulo, utiliza uma estratégia de dividir o problema de escolha da melhor rota em pequenos problemas, onde cada nível é responsável por uma decisão parcial.

Os algoritmos de busca de árvores geradoras mínimas utilizam "pesos" nas arestas e é a partir desses "pesos" que se considera um caminho melhor que o outro. De maneira semelhante nas redes de computadores, há diversas métricas envolvidas no cálculo do "peso" do enlace, sejam elas qualitativas ou quantitativas. Além disso, outras técnicas são úteis nesse cálculo, tais como análise de histórico, consulta de reservas, predição de utilização, valor monetário, análise de reputação e impacto ambiental. Dessa maneira, o módulo de roteamento não deve apenas aplicar um algoritmo para cálculo do melhor caminho, mas também deve atuar como um analisador de "custos" do enlace, conforme as técnicas descritas acima.

Na versão atual foi utilizado um algoritmo baseado no de Kruskal e as métricas utilizadas contemplaram apenas o número de *hops*, que é uma técnica muito utilizada em estratégias de roteamento atuais, como no protocolo BGP. Além disso, os testes limitaram-se ao nível de ação local, o que simplifica o cenário de experimentação. No entanto, acredita-se que os resultados apresentados sejam válidos, por atenderem aos objetivos definidos. O aprimoramento do módulo de roteamento, bem como a sua implementação definitiva encontra-se em desenvolvimento em uma linha de pesquisa paralela.

4.4.10 NEC -*Network Element Connector*

O módulo NEC funciona como uma camada de abstração de hardware para a comunicação com a rede, no qual são utilizados conectores com diferentes protocolos e estratégias. Esses conectores implementam os serviços da interface *ElementConnector* e os mapeiam na linguagem dos elementos de rede aos quais se destinam.

As redes atuais são marcadas por uma miscelânea de equipamentos, de diferentes fornecedores e modelos. Esses equipamentos normalmente não são padronizadas, o que dificulta muito a operação da rede. Embora tentativas de padronização já tenham sido realizadas, como por exemplo na definição do protocolo SNMP e na publicação de MIBs com uma ontologia de rede genérica, tal objetivo nunca foi alcançado efetivamente, por falta de interesse dos fornecedores.

A abordagem SDN, por outro lado, simplifica a operação das redes a partir de uma separação entre o plano de controle e o plano de dados. Na materialização desse conceito, realizada pelo OpenFlow, a padronização do controle da rede foi aprimorada ainda mais,

a partir da utilização do protocolo homônimo. Nesse modelo, os equipamentos passam a obrigatoriamente suportar esse protocolo, que atua de maneira simples, uma vez que o plano de controle não ocorre mais no âmbito dos elementos de rede. O protocolo *OpenFlow* define uma camada de rede programável, assim como ocorre nos sistemas operacionais, e a partir desse mecanismo o comportamento da rede é programado externamente através do elemento controlador.

Embora exista uma tendência crescente de suporte ao protocolo OpenFlow, encabeçado por alguns dos maiores fabricantes de equipamentos de rede, tais como Cisco, Juniper, NEC e HP, a rede ainda é composta por inúmeros equipamentos que dão suporte apenas às abordagens convencionais, e o processo de migração poderá ser lento e gradual. Trata-se do problema de suporte ao legado, sendo que o módulo NEC foi planejado exatamente para cenários mistos, o que casa com os objetivos de interoperabilidade com a rede legada. Este é um dos grandes diferenciais dessa abordagem, por representar uma possibilidade real de implantação de uma proposta "revolucionária" na rede atual.

Uma das possibilidades de acesso aos equipamentos atuais está na utilização da interface em *linha de comando*, presente desde os primórdios da rede. A estratégia consiste no desenvolvimento de "robôs" de *software* especializados no sistema operacional do equipamento sendo acessado, que traduzem as instruções do DTSA em comandos de configuração no *prompt* de comando. O acesso ocorre através dos bem conhecidos protocolos SSH ou Telnet, e utiliza uma estratégia de *expect* para a leitura das respostas.

Também é possível a utilização de *web-services*, disponibilizados por *software* proprietários ou pelos próprios equipamentos. A ideia consiste no desenvolvimento de clientes REST ou SOAP, para a tradução das instruções do DTSA em chamadas de *web-services*. Também é possível implementar conectores específicos para um determinado equipamento, o que é indicado caso a única interface possível seja por meio de um protocolo proprietário. Por fim, caso o único acesso possível seja a partir de um *software* proprietário ou sistema WEB, são necessários conectores especializados mais complexos, capazes de "*hackear*" essa aplicações, e assim, viabilizar a configuração dos equipamentos que elas fazem interface.

Na implementação atual, o único conector implementado foi o *OpenFlow Connector*, que é responsável pela tradução das instruções de configuração dos *workspace* em regras de fluxos nos switches OpenFlow. Esse conector cria regras na tabela de fluxos dos elementos de rede a partir do serviço OFPT_FLOW_MOD. As regras definidas seguem o formato *dl_vlan=$workspace_id,actions=output:$port1_id,output:$port2_id*, onde o $workspace_id é utilizado como id da vlan (processo acordado na primitiva de rede *DL-Ontology*), e as variáveis $port1_id e $port2_id representam as portas por ondem os dados devem ser encaminhados. O conector OpenFlow foi implementado como uma extensão do módulo *StaticFlowEntryPusher* definido no controlador estendido *Floodlight*. Ele se conecta ao barramento NEC a partir do padrão *Observer*, e se comunica com os *switches* a partir

das conexões já estabelecidas com o controlador. A maneira como os fluxos são criados e alterados, de acordo com as rotas calculadas no estabelecimento dos *Workflows*, pode ser observada no Apêndice B.

4.5 Aplicações da Prova de Conceito

Após a implementação dos componentes da arquitetura ETArch, foi identificada a necessidade de outras aplicações que seriam utilizadas nos experimentos. A primeira delas foi o *DTSA Viewer*, que foi implementado para facilitar a visualização de *logs*, de estado e da topologia de rede. Posteriormente foram implementadas aplicações de propósitos específicos, uma de troca de mensagens (*FinChat*), e outra de *streaming* de vídeo (*FinMovie*). O objetivo dessas aplicações foi o de validar a implementação da arquitetura ETArch utilizada, e testar a proposta de agregação de tráfego via *multicast* que é o objetivo desse trabalho.

4.5.1 DTSA Viewer

O *DTSA Viewer* é uma aplicação utilizada para a visualização do DTSA desenvolvida em JAVA. Ela utiliza a biblioteca de componentes gráficos SWING e a API de grafos JGraph para a renderização da topologia. A aplicação consiste na apresentação de quatro telas, como pode ser observado na Figura 4.6. Essas telas têm propósitos bem definidos, que são descritos abaixo:

Topology – apresenta uma visão da topologia de rede controlada pelo DTSA, composta por *entidades*, *switches*, *hosts* e *links*, sendo que essa tela também define dois modos de operação com o mouse – o "mover", que move todos os elementos de posição, e o "editar", que move apenas o elemento selecionado;

Console – apresenta todos as mensagens de *log* do sistema, que estão ligadas à execução de todos os módulos do DTSA, desde o *discovery* de topologia até a configuração do *workspace*;

Entities – apresenta uma lista de todas as *entidades* atualmente registradas; e,

Workspaces – apresenta uma lista com todos os *workspaces* atualmente registrados.

4.5.2 FinChat

O *FinChat* é a primeira aplicação de teste da arquitetura ETArch desenvolvida. Ela foi implementada em linguagem JAVA, somente com recursos da biblioteca de componentes gráficos nativa, a SWING.

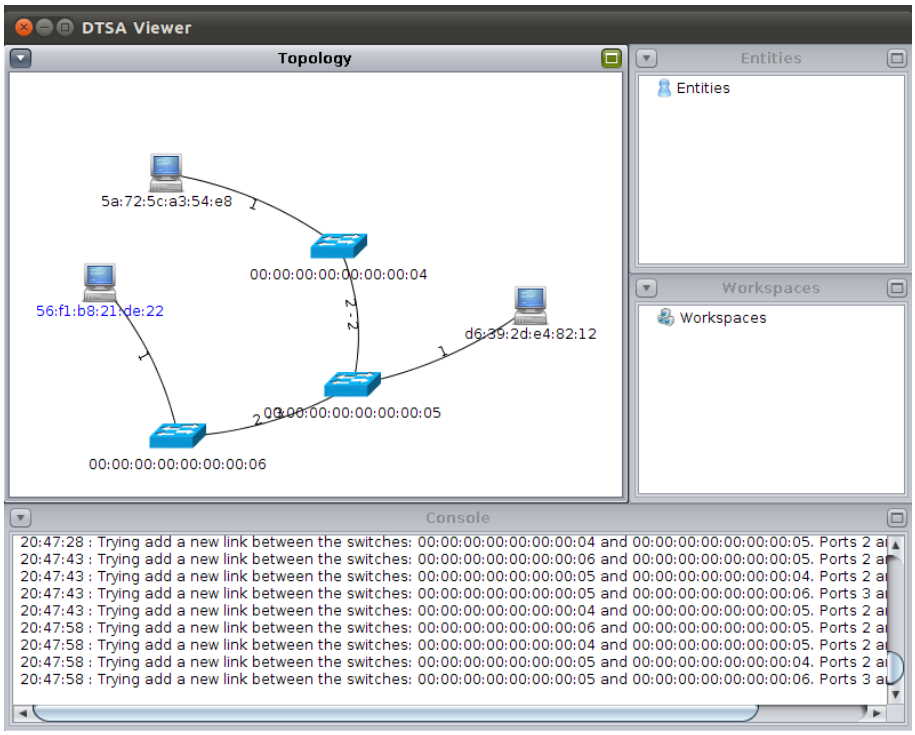


Figura 4.6: DTSA Viewer : Sistema de visualização do DTSA.

Trata-se de uma aplicação na qual é possível criar *workspaces* e utilizá-los como "salas de bate-papo". Ao se registrar no *workspace* a *entidade* entra na sala e a partir de então, todas as mensagens enviadas para a sala são direcionadas às demais *entidades* participantes sem replicação de dados.

Como pode ser observado na Figura 4.7, a aplicação consiste em uma tela para impressão das mensagens recebidas, que podem conter "emoticons"; um campo para escrita da mensagem; um *label* com o título da *entidade* registrada; uma lista de *workspaces* selecionáveis, onde o selecionado é utilizado como destino para as mensagens enviadas; e por fim, botões de ação com as funcionalidades de inscrição e abandono de *workspaces*.

4.5.3 FinMovie

A aplicação *FinMovie* é uma ferramenta utilizada para reproduzir *streaming* de vídeo sobre a arquitetura ETArch. Essa aplicação na verdade é composta por outras duas aplicações: uma *publisher*; e outra *subscriber*.

Trata-se de uma aplicação desenvolvida em JAVA com recursos gráficos nativos da linguagem (AWT) conforme apresentado na Figura 4.8. O formato de vídeo utilizado é o MJPEG, que é composto por um vetor de imagens concatenadas, codificadas no formato JPEG, sem áudio.

O funcionamento da aplicação ocorre da seguinte maneira: a aplicação *publisher* (uma entidade) se registra no DTS e solicita a criação de um *workspace* para uma instância de transmissão. A aplicação *subscriber-1* (outra entidade) também se registra no DTS

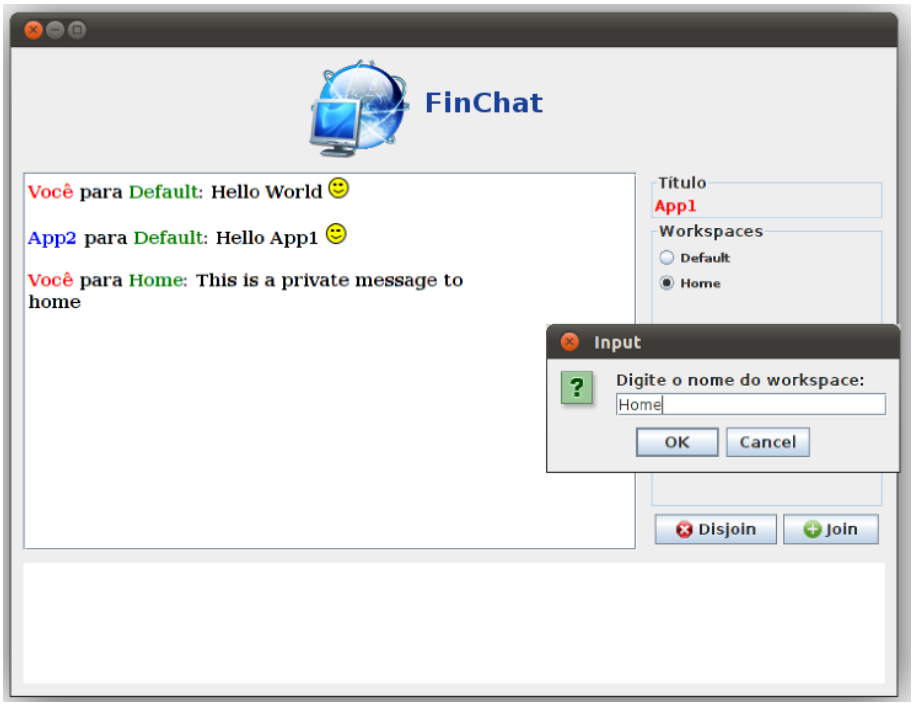


Figura 4.7: FinChat : Ferramenta de chat baseada no ETArch.

e se liga (*attach*) ao *workspace* criado pela *entidade publisher*. Nesse momento o DTSA configura o *workspace* nos elementos de rede e os *frames* enviados pelo publisher chegam ao *subscriber-1* onde são renderizados. Um novo cliente, a aplicação *subscriber-2* (uma outra entidade), também se liga ao *workspace*. O DTSA reconfigura o *workspace*, fazendo com que o *subscriber-2* passe a receber o *streaming* de vídeo.

Figura 4.8: FinMovie : Ferramenta de vídeo *streaming* baseada no ETArch.

Foram desenvolvidas outras duas versões dessa aplicação, que foram utilizadas nos experimentos, uma utilizando o serviço *unicast* da arquitetura atual, com um *multicast* no nível de aplicação, e outra com o serviço de *multicast* da arquitetura atual. As três aplicações são muito parecidas e se diferem basicamente nos tipos de *sockets* instanciados.

Dessa maneira, acredita-se que os dados alcançados na experimentação representem de maneira fiel as características de cada abordagem.

Estas duas versões adicionais de aplicações, que foram criadas para uso nos experimentos, objetivam a testes comparativos de medição tráfegos na arquitetura ETArch e também na arquitetura Internet. Os resultados obtidos podem ser vistos no Capítulo 5.

Capítulo 5

Experimentos e Resultados Comparativos

Este capítulo apresenta os experimentos realizados neste trabalho, detalhando cenários, testes e resultados. O objetivo primordial destes experimentos é demonstrar a viabilidade dos conceitos de *Workspace* e agregação de tráfego via *Multicast*. Além disso, apresenta números comparativos de aplicações (*entidades*) fazendo uso da arquitetura Internet e da arquitetura ETArch.

5.1 Justificativa do Experimento

Hoje em dia, aplicações como o *Netflix* e *Youtube* são responsáveis por grande parte do tráfego da rede, e embora utilizem estratégias baseadas no modelo *content-centric*, não colhem os benefícios da comunicação *multicast* como é proposto nesse trabalho. A ETArch é uma arquitetura flexível que utiliza novas estratégias de endereçamento e roteamento capazes de prover uma experiência *multicast* real. Isso significa que a rede torna-se responsável pela replicação dos dados transmitidos, que é realizada de maneira transparente nos elementos de rede, sem desperdício de banda. O caminho do fluxo de transmissão (*Workspace*) é estabelecido no início da comunicação e pode ser alterado de acordo com mudanças no contexto de comunicação, tais como indisponibilidade de elementos de rede, crescimento dos níveis de utilização dos enlaces e alteração das localizações das entidades. A rota utilizada é representada por uma "árvore geradora mínima", o que elimina a utilização de caminhos redundantes desnecessariamente, e assim, reduz o tráfego de dados, por meio de uma *agregação de tráfego via multicast*.

Acredita-se que essa *agregação de tráfego via multicast* possa ser utilizada pelas aplicações já citadas, diminuindo o tráfego de dados redundantes, reduzindo custos operacionais, e melhorando a qualidade do serviço. Por isso, o experimento foi planejado para utilizar uma aplicação de vídeo *streaming*. Acredita-se que os resultados obtidos forneçam uma base para justificar a utilização da arquitetura ETArch nas aplicações de vídeo atuais, e assim, disseminar a abordagem proposta de maneira a contribuir para a Internet do futuro.

5.2 Descrição do Experimento

No experimento foi utilizada a aplicação *FinMovie*, descrita na Seção 4.5.3. Foram utilizadas seis instâncias executáveis dessa aplicação, divididas em três pares *publisher/-subscriber*, que representam as três abordagens citadas abaixo.

5.2.1 IP Unicast

A abordagem *IP Unicast* se baseia na utilização de uma estratégia *client/server* convencional. Essa abordagem utiliza uma estratégia de replicação dos dados no *server*, que inicia um novo processo à cada requisição, para o envio do *streaming* de vídeo a um único *cliente*.

O fluxo de operação da aplicação *publisher* se inicia com a instanciação de um *DatagramSocket*, que é uma implementação de *Socket UDP/Unicast* em JAVA. Depois disso, a aplicação realiza um processo de *bind* em uma porta específica, e permanece em estado de *listening*, aguardando a conexão da aplicação *subscriber*.

Do outro lado da rede, a aplicação *subscriber* é iniciada. É criada uma tela que será utilizada para apresentar os resultados e uma *thread* que se conecta à aplicação *subscriber*, utilizando o seu endereço IP e porta UDP. Depois de estabelecer a conexão, a *thread* inicia um processo de leitura dos frames JPEGs, que são transmitidas através do protocolo RTP, e renderizadas na tela através da classe *Toolkit*, presente na biblioteca gráfica JAVA AWT.

Na aplicação *publisher*, quando é recebida uma solicitação de conexão da aplicação *subscriber*, é iniciada uma nova *thread* para o atendimento da requisição e a aplicação permanece em estado *listening*. Essa *thread* lê o arquivo de vídeo a partir da classe *MjpegInputStream*, desenvolvida segunda a especificação MJPEG, e começa a enviar os *frames* JPEG periodicamente para a aplicação *subscriber*.

5.2.2 IP Multicast

A abordagem *IP Multicast* se baseia nos serviços de *multicast* nativos da arquitetura TCP/IP. Ela utiliza o protocolo IGMP, que é responsável pela sinalização, e que é in-

terpretado pelo roteador. A partir dessa negociação, os roteadores com suporte ao *IP Multicast*, realizam a replicação dos dados de maneira "transparente".

Nessa estratégia, há poucas diferenças de implementação se comparada à abordagem *IP Unicast*. Isso se deve à especificação do serviço de *multicast* da arquitetura TCP/IP, que se baseou nos padrões já definidos do serviço *unicast*. A diferença na aplicação *subscriber* é que não é realizado o processo de *bind* em uma porta específica. Dessa maneira, a aplicação não fica em estado *listening* e nem utiliza múltiplas *threads* no atendimento das requisições. Na verdade nenhuma requisição é realizada na aplicação *publisher*, que não estabelece uma relação *fim-a-fim* com a aplicação *subscriber*.

O processo de execução da aplicação *publisher* se inicia com a criação de um *DatagramSocket*. Posteriormente é instanciada uma *thread* (única para todo o processo), que realiza a leitura do arquivo de vídeo MJPEG a partir da classe *MjpegInputStream*, e envia os *frames* JPEG periodicamente para um endereço IP classe D definido para a transmissão.

A aplicação *subscriber* dessa abordagem também tem uma implementação bem similar à utilizada na abordagem *IP Unicast*. A diferença está na utilização da classe *MulticastSocket* ao invés da *DatagramSocket*, que é uma implementação nativa JAVA de *Socket UDP/Multicast*. A aplicação *subscriber* inicia os mesmos mecanismos de leitura e renderização dos *frames* já apresentados, mas antes disso é realizada a conexão com o grupo *multicast* através do método *joinGroup*, que recebe como parâmetro o mesmo endereço classe D utilizado na aplicação *publisher*.

5.2.3 ETArch Workspace

A abordagem *ETArch Workspace* se baseia nos conceitos apresentados nesse trabalho, através dos quais é aprovisionado um mecanismo de *multicast* real, suportado naturalmente pela arquitetura.

Embora as mudanças propostas sejam profundas em natureza, espera-se que os impactos dessa mudança na camada de aplicação sejam mínimos, o que de fato ocorre na implementação das aplicações *publisher* e *subscriber* dessa abordagem. O resultado final é que as aplicações utilizadas na abordagem *ETArch Workspace* apresentam uma implementação bem próxima das aplicações utilizadas atualmente nas abordagens *IP Multicast* e (*IP Unicast*). Isso ocorre pois a natureza dos serviços é mantida, o que facilita no atendimento dos "contratos" já estabelecidos com a camada de aplicação.

O curso de execução da aplicação *publisher* na abordagem *ETArch Workspace* se inicia com a criação de um *socket* para a comunicação, assim como nas demais abordagens. Trata-se do *FinSocket*, cujos conceitos já foram apresentados na Seção 4.3. A aplicação realiza o processo de registro do seu *título* e do *workspace*, que será utilizado para a transmissão dos dados. Por fim, a aplicação se inscreve no *workspace* e usa a classe

MjpegInputStream para ler o arquivo de vídeo MJPEG a ser transmitido. Similarmente a outras aplicações, os *frames* são enviados periodicamente através do *socket* aberto e são direcionados ao *workspace* que foi registrado nas etapas anteriores.

A aplicação *subscriber* dessa abordagem também apresenta diversas similaridades às demais. São utilizados os mesmos mecanismos de leitura e renderização de *frames* já apresentados. A diferença importante está na utilização da classe *FinSocket*, que também foi utilizada na aplicação *publisher*. A aplicação *subscriber* realiza o registro do seu *título* e se inscreve no *workspace* criado pela aplicação *publisher*. A partir desse momento, os *frames* começam à chegar na aplicação e são lidos a partir do *socket*, exatamente como nas outras abordagens.

5.3 Cenário do Experimento

O experimento foi realizado em uma máquina virtual com o sistema operacional Ubuntu versão 11.04, com um Kernel Linux 2.6.38, arquitetura 32 bits. Essa máquina virtual foi executada a partir do software VirtualBox, em uma máquina hospedeira com Ubuntu versão 13.04, Kernel Linux 3.8.8, arquitetura 64 bits, com 8 Gb de memória RAM, e um processador Intel Core i5 que opera a 2.6 GHz.

A rede utilizada no experimento foi simulada através do software Mininet, versão 1.0.0 [Lantz et al. 2010]. Esse sistema cria uma topologia de rede que pode ser utilizada em testes, a partir de uma técnica de virtualização baseada em processos, e de interfaces de rede virtuais, mapeadas através do conceito de *network namespaces*, presente no Kernel Linux a partir da versão 2.6.26. O Mininet é implementado na linguagem de programação Python e fornece amplo suporte ao protocolo OpenFlow, além de possibilitar a utilização de *OpenFlow Software-Switches* na composição da topologia. A versão do protocolo OpenFlow utilizada é a 1.0, que embora apresente limitações, não interferiu na realização do experimento.

A aplicação utilizada no experimento, como descrito anteriormente, foi a *FinMovie* descrita na Seção 4.5.3 e utilizada em três abordagens distintas: *IP Unicast*, *IP Multicast* e *ETArch Workspace*. Todas as três abordagens apresentadas foram submetidas aos mesmos casos de teste, em condições e cenários equivalentes. Foi utilizado um vídeo MJPEG de 43Mb, gerado a partir do processamento de um arquivo de vídeo pela ferramenta *ffmpeg*. Foi utilizado um "vídeo-clipe" (Be Yourself – Audioslave), codificado originalmente com os codecs MPEG2 e AAC, e com o container AVI. As transmissões foram analisadas a partir do software *wireshark* por dez minutos em cada bateria de testes. Os dados foram obtidos a partir da funcionalidade *Statistics/IO Graphs* desse *software* e foram plotados através do LibreOffice. Foram realizados testes com um, dois, quatro, oito, dezesseis, trinta e dois, sessenta e quatro e oitenta clientes simultâneos respectivamente.

A topologia de rede foi programada a partir de um *script* em linguagem Python

(Apêndice A). Esse *script* descreve a topologia da Figura 5.1 e é passado como parâmetro de inicialização do Mininet. A rede utilizada é composta por seis *switches* e nove *hosts* que hospedam a aplicação *FinMovie*. O *host* h2 que representa o (*Video Server*), é utilizado na instanciação do *publisher*, enquanto os demais *hosts* hospedam instancias de *subscribers* que representam os clientes do *streaming* de vídeo. Para iniciar as instancias de *subscribers* nos hosts foi criado um *script* que inicia um número configurável de instancias simultaneamente.

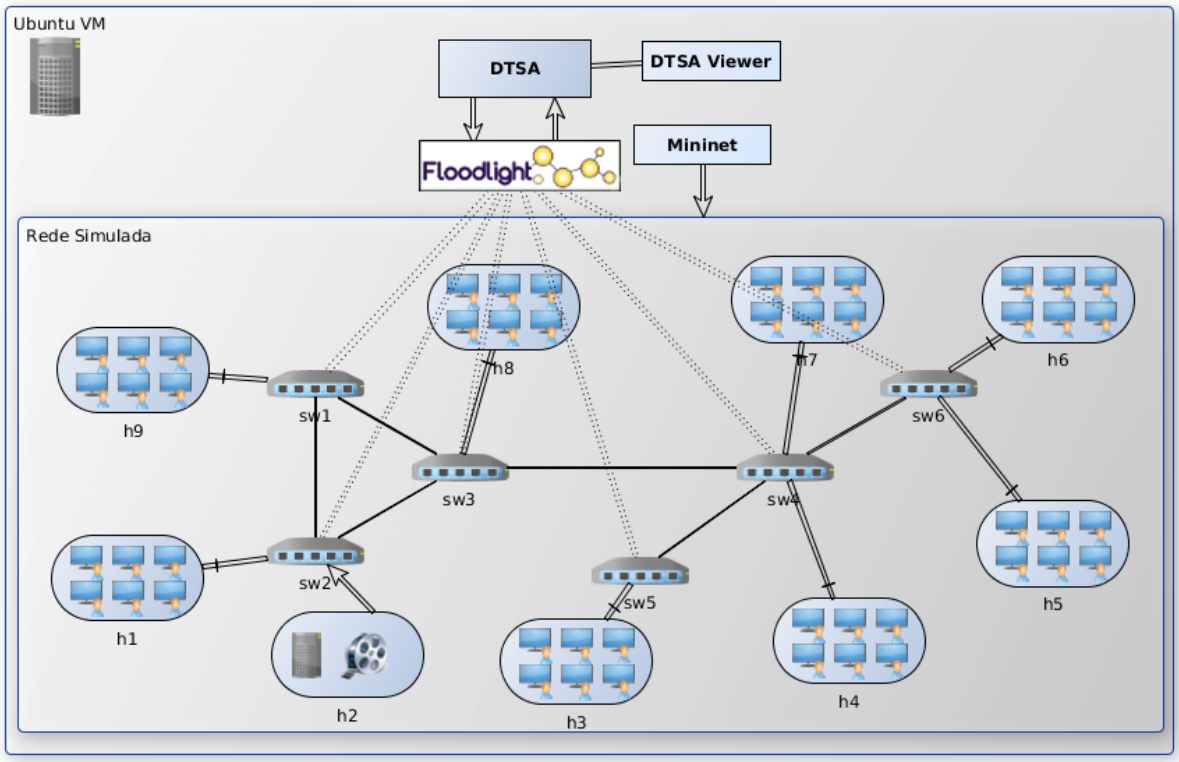


Figura 5.1: Cenário utilizado no experimento.

No cenário de experimento com a abordagem *ETArch Workspace*, o Mininet foi configurado (via parâmetro de inicialização) para utilizar um controlador externo. O controlador utilizado é o DTSA, a partir do controlador estendido *Floodlight* descrito na Seção 4.4. No cenário de experimento com o *IP Unicast*, o Mininet foi configurado para utilizar o controlador OpenFlow padrão. Depois de iniciada a topologia, foram criadas regras de fluxos em todos os *switches* manualmente, através da ferramenta *ovs-ofctl*. Por fim, no cenário de experimento com o *IP Multicast*, o Mininet foi utilizado apenas para a virtualização da topologia de rede, dessa maneira não foi utilizado nenhum controlador, ou mesmo *switch* OpenFlow. Foram criadas rotas estáticas em cada *host*, para o encaminhamento da gama de endereços IP classe D para o *Gateway* da rede. Dessa maneira, foi possível negociar a *inscrição* dos *subscribers* no grupo *multicast*, através do envio de primitivas IGMP ao roteador com suporte ao *IP Multicast*.

5.4 Análise dos Resultados

Os resultados alcançados nos experimentos demonstram a viabilidade dos conceitos de *Workspace* e agregação de tráfego via *Multicast* propostos nesse trabalho. Foi constatado também que o conceito *ETArch Workspace* apresentou os melhores resultados em todos os testes, como pode ser observado nas figuras subsequentes.

A Figura 5.2 apresenta uma comparação entre todas as abordagens do experimento. Como esperado, a abordagem *IP Unicast* apresenta o maior consumo de banda, por apresentar uma estratégia de replicação dos dados no provedor de conteúdo em nível de aplicação. A largura de banda nessa abordagem apresentou um crescimento linear, que aumenta de acordo com o número de clientes conectados. É importante lembrar, que a maioria das aplicações hoje em dia, utiliza uma solução muito próxima a esta abordagem, o que representa uma grande oportunidade de melhoria a partir das outras abordagens propostas.

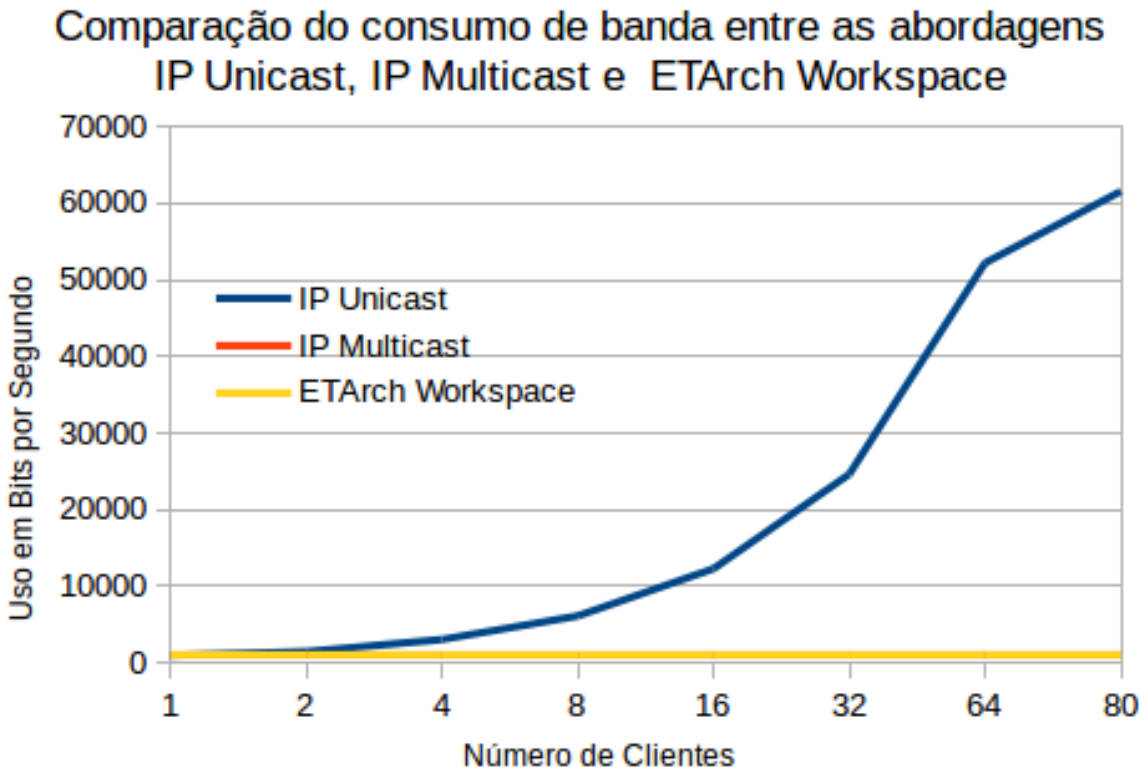


Figura 5.2: Comparação entre todas as abordagens do experimento.

Já a abordagem *ETArch Workspace*, diferentemente da abordagem *IP Unicast*, apresentou um consumo de banda constante, independentemente do número de clientes conectados. Isso ocorre pois a abordagem desse trabalho propõe uma estratégia de *multicast* provido naturalmente para todas as comunicações pela rede. Quando a aplicação *subscriber* se inscreve no *workspace*, o DTSA recalcula as rotas de fluxo utilizadas na interconexão

das *entidades*, e reconfigura os elementos de rede de maneira à estabelecer o *enlace lógico*. A partir daí, todos os dados transmitidos ao *workspace*, são replicados à todas as *entidades* participantes.

A abordagem *IP Multicast* apresentou um comportamento parecido com o da abordagem *ETArch Workspace*, como pode ser visto em mais detalhes na Figura 5.3. Isso ocorre porque essencialmente ambas as abordagem apresentem estratégias de comunicação *multicast*, com conceitos similares, como por exemplo a relação entre o conceito de grupo *multicast* e *workspace*, e entre os protocolos IGMP e ETCP.

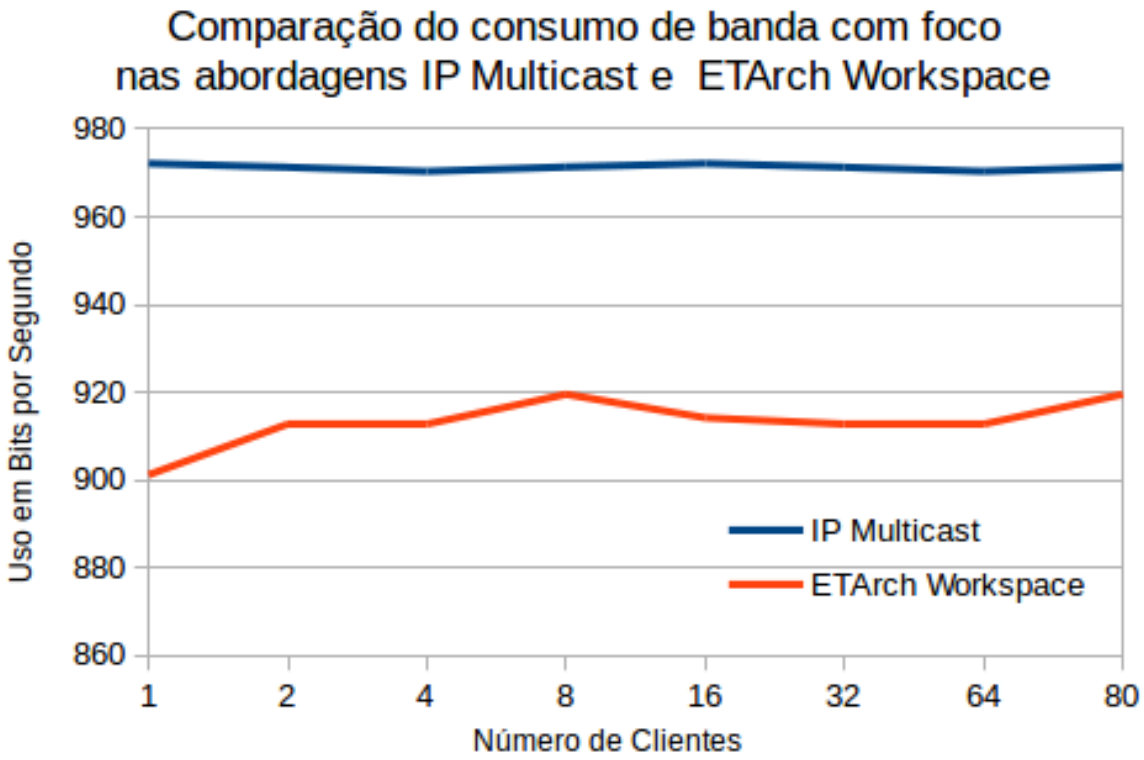


Figura 5.3: Comparação entre a abordagem proposta e a de *multicast* convencional.

Em ambas as abordagens, os dados são replicados nos elementos de rede, no entanto, o *IP Multicast* apresenta sérias limitações relacionadas à arquitetura TCP/IP, nos seguintes aspectos:

Capacidade de Endereçamento – por utilizar uma estratégia de *endereçamento* naturalmente limitado, o que é agravado pela restrição da gama de endereços possíveis à classe D; e pela impossibilidade de utilizar esse endereço para a entrega dos dados, por não representar uma localização física, o que conflitua diretamente com as estratégias de roteamento atuais;

Suporte da Rede – por requerer um amplo suporte dos elementos de rede e não apenas do "core" da rede. Sem isso, a replicação dos dados fica restrita a porções da rede, o que inviabiliza a ampla utilização desse serviço;

Sinalização de Controle – que se baseia no protocolo IGMP, que apresenta sérias limitações na implantação em redes de grandes proporções [Boudani and Cousin 2003] e pelo alto *overhead* imposto; e,

Granularidade do Endereço IP Multicast – pois nos dias atuais várias aplicações podem requerer comunicação em um *host* e o endereço IP *Multicast* endereça nós da rede, sendo que, a tendência é que o endereçamento vá além de aplicações e atinja conteúdos, coisas etc.

Os resultados apontam também que a abordagem *ETArch Workspace* utiliza ligeiramente menos banda que a *IP Multicast*, o que se deve à estratégia de *sinalização* simplificada e pela eliminação do *overhead* desnecessário nas primitivas de rede.

Por fim, a Figura 5.4 apresenta os resultados já expostos, com mais enfoque no comportamento da abordagem *ETArch Workspace*. Pode ser percebido que essa abordagem disponibiliza um serviço de *multicast real*, uma vez a variação tanto em relação ao tempo, quanto em ao número de usuários é mínima.

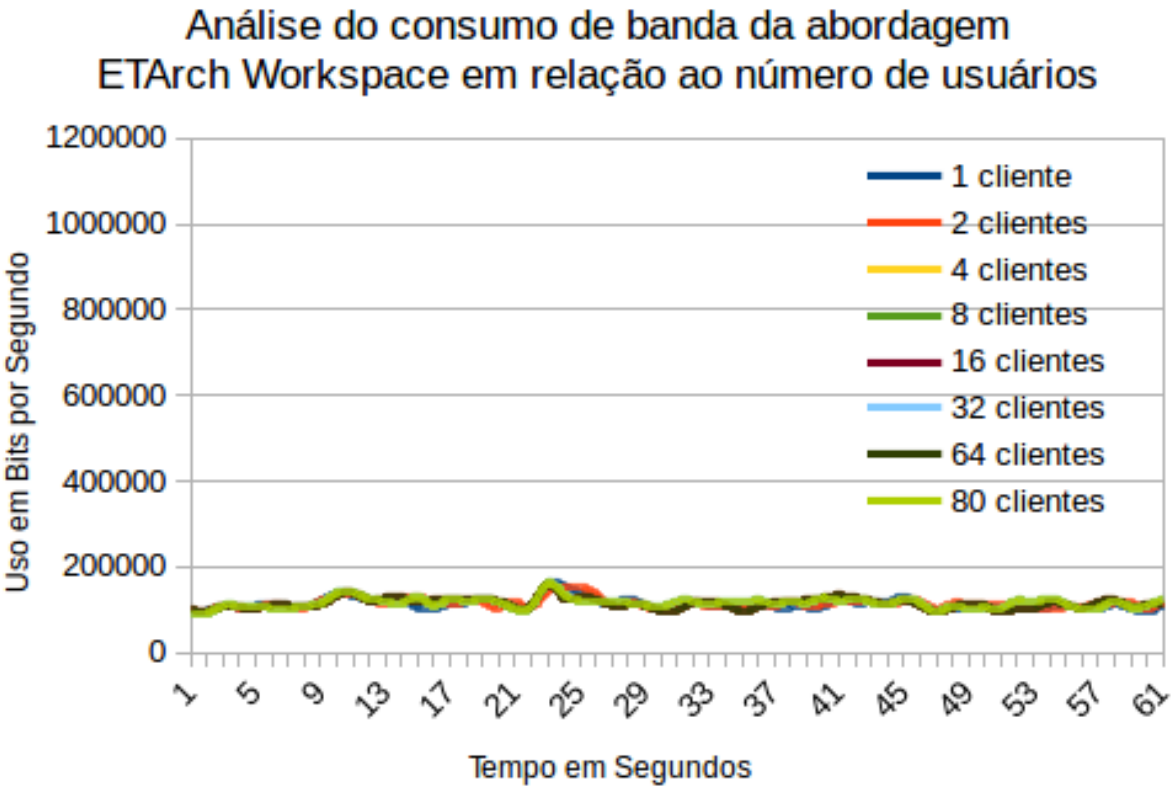


Figura 5.4: Relação entre número de usuários e consumo de banda na abordagem proposta.

A comunicação *Multicast* oferecida pelo conceito de *Workspace* permite que várias instâncias de aplicações, com necessidades específicas de tráfego e com ciclos de vida independentes, possam residir no mesmo *host* naturalmente, o que não é o caso das tecnologias baseadas no protocolo IP.

Capítulo 6

Conclusão e Perspectivas Futuras

Em função da evolução havida no âmbito das aplicações distribuídas, o TCP/IP apresenta sérias limitações arquiteturais, sobretudo nos aspectos de endereçamento e de suporte aos requisitos dessas aplicações. Como resultado, houve um intenso movimento de evolução na camada de aplicação, com o intuito de remediar esses problemas. No entanto, verifica-se que tais soluções, além de parciais ou paliativas, não apresentam bom desempenho, por lidarem com o *overhead* imposto pela arquitetura Internet. Este cenário é marcado por um desvirtuamento dos princípios filosóficos das camadas definidas pelo Modelo OSI, pois obriga a camada de aplicação a tratar aspectos cuja natureza é voltada para as questões de transporte, sessão ou endereçamento.

Esse trabalho defende o uso de uma linguagem com expressividade semântica, apta a descrever os requisitos e capacidades das *entidades* em alto nível. Nessa abordagem, a rede é responsável por interpretar esses requisitos, por garantir que eles sejam atendidos por meio de alocação de recursos de rede e por fomentar a utilização de algoritmos especializados no tratamento das primitivas. Essa tarefa é desempenhada na arquitetura ETArch, que centraliza o plano de controle no DTS, o orquestrador da rede e que também é responsável pelo gerenciamento do ciclo de vida das *entidades* e dos *workspaces*. Esse conceito é bem similar ao SDN, por se basear essencialmente em uma separação entre o plano de controle e o de dados.

Na estratégia de *endereçamento* da arquitetura Internet, uma mesma chave é utilizada tanto para *identificação* quanto *localização* dos *hosts*, o que gera uma dubiedade na utilização do endereço IP. Além disso, há uma ambiguidade no aspecto de endereçamento, que é tratado por pelo menos duas camadas: a de enlace, que utiliza o endereço MAC; e a de rede, que utiliza o endereço IP. Assim, verifica-se a necessidade de uma nova estra-

tégia de endereçamento, baseada em uma identificação única inequívoca, não ambígua e independente da topologia, o que é proporcionado pela arquitetura proposta através do conceito de *título*.

Observe que não basta introduzir uma linguagem com expressividade semântica nas camadas superiores. Esta linguagem precisa ser compreendida pelas camadas subjacentes, de tal modo que os requisitos das aplicações possam ser entendidos pelas camadas inferiores, incluindo as camadas física e de enlace.

A Arquitetura ETArch propõe também uma revisão no aspecto de *localização*, que está intimamente ligado às estratégias de *roteamento*. Entende-se que a dubiedade apresentada entre *identificação* e *localização* no endereço IP, seja o grande limitador no atendimento de requisitos de *multicast* e *mobilidade*, que se baseiam em uma definição de *localização* diferente da convencional, composta por múltiplos endereços, no caso do *multicast*, ou por um endereço que pode ser alterado durante a comunicação, no caso da *mobilidade*. Na abordagem proposta nesse trabalho, o DTS é o responsável por controlar a *localização* das *entidades*, que podem ser alteradas em qualquer momento. Essas informações são utilizadas na composição de *workspaces*, que são os mecanismos pelos quais a comunicação é estabelecida.

6.1 Resultados Experimentais

A avaliação experimental demonstrou que a abordagem proposta nesse trabalho apresenta um uso constante da largura de banda, independente do número de *clientes* conectados, o que caracteriza a comunicação *multicast* real. O impacto real desse resultado está na possibilidade de utilização desse conceito na agregação de tráfego da rede, a partir de uma estratégia de *multicast*, o que torna o processo todo mais barato, através da redução de custos com servidores e elementos de rede.

Além disso, melhora a percepção dos usuários, uma vez que a rede passa a garantir os requisitos definidos pela aplicação. Outro ponto importante é a minimização dos impactos ambientais praticados pela rede, com destaque à redução da utilização de energia elétrica e água na infraestrutura de rede. Esse processo democratiza também a utilização da rede, uma vez que todas as *entidades* tornam-se aptas à fornecerem conteúdos numa proporção global, sem a necessidade de uma grande infraestrutura para isso.

Isto ocorre pois além da redução significativa do *overhead* de comunicação (melhor *Code Rate*), os elementos de rede passam a reduzir o processamento necessário ao encaminhamento de primitivas, uma vez que não há mais roteamento *in band*. Na arquitetura ETArch toda a estratégia de estabelecimento do *workspace* é feita pelo plano de controle.

A arquitetura ETArch foi projetada como um quebra-cabeças de blocos funcionais (*building blocks*) onde somente os blocos necessários a um *workspace* são invocados naquele contexto de comunicação. Esta economia é totalmente possível, pois com a construção de

elementos de redes por meio de hardware reconfigurável, somente consomem energia os blocos funcionais em operação.

Os resultados apontam também que a abordagem proposta representa uma forma mais eficiente de comunicação *multicast* que as soluções atuais, tais como *IP Multicast* (na camada de rede) e ALM (na camada de aplicação), por não se limitar às estratégias da arquitetura TCP/IP. O ETArch provê uma comunicação *multicast* real, por meio de mudanças drásticas nos esquemas de endereçamento e encaminhamento. Não há repetição de primitivas na comunicação dentro do *workspace*, que apresenta um suporte natural para a comunicação entre múltiplas *entidades*, diferentemente da abordagem utilizada no ALM, que apesar de reduzir a replicação, não a elimina de fato. A estratégia do ALM consiste em utilizar árvores de distribuição de dados através de um contexto parcial dos elementos de rede. Além disso, embora essa abordagem forneça uma solução aceitável para a comunicação *multicast*, ela ainda é submetida às limitações arquiteturas do TCP/IP no que se refere ao requisito de *mobilidade*.

O experimento com a abordagem *IP Multicast* apresentou um resultado bem próximo do experimento com a abordagem proposta, o que indica o sucesso na estratégia de comunicação *multicast*, por demonstrar que o tráfego de dados não se altera de acordo com o número de clientes conectados. No entanto, a abordagem *ETArch Workspace*, utiliza menos banda de dados, o que se deve à estratégia de *sinalização* mais enxuta, e eliminação de *overhead* desnecessário nas primitivas de rede. Além disso o *IP Multicast* apresenta sérias limitações nos aspectos de: endereçamento, por causa do número limitado de *endereços multicast*, que são restritos à *classe D*; suporte da rede, uma vez que torna-se necessário o amplo suporte dos elementos de rede para a utilização desse serviço; e sinalização de controle, que impõe uma sobrecarga impraticável em escala global.

Embora tenha apresentado o pior desempenho nos experimentos, a abordagem *IP Unicast* é a mais utilizada na maioria das aplicações de vídeo *streaming* e compartilhamento de arquivos atuais, que são as principais beneficiadas pela comunicação *multicast*. Isso se deve à baixa flexibilidade da arquitetura TCP/IP, que impede a adoção de novas estratégias. Diante a impossibilidade de uma abordagem melhor para a comunicação em grupo, essas aplicações lançam mão de desempenho, e fornecem uma solução "força-bruta" para o problema, o que resulta em uma super-utilização da rede, e desperdício de dinheiro com investimentos em infraestrutura. Movida por esse cenário, a comunidade científica tem trabalhado em estratégias voltadas para essa comunicação, através do conceito *content-centric*. Este conceito utiliza esquemas de replicação de conteúdo e *load-balancing*, que provêm uma minimização dos impactos do uso inadequado de serviços *unicast* para comunicação em grupo.

Portanto, é interessante frisar que, apesar do protocolo IP apresentar uma classe de endereçamento *Multicast*, a realidade demonstra que não há aplicações na Internet que façam uso desta classe. A arquitetura ETArch oferece um mecanismo de comunicação

multicast que de fato é factível e utilizável pela camada de aplicação.

6.2 Objetivos Alcançados

As contribuições deste trabalho, como planejado no Capítulo 1 são listadas abaixo.

- Foi realizada a validação do *Modelo de Título* e uma implementação da Arquitetura ETArch. Nesse processo foram redefinidos diversos conceitos, especialmente na estratégia de *endereçamento* e *encaminhamento*, com enfoque no requisito de *multicast*.
- Foi implementado o protocolo *DL-Ontology*, que é o responsável pela comunicação nessa abordagem, através de uma linguagem de "baixo" nível, com acesso direto à placa de rede. Assim, foi desenvolvida a *libFinlan.so*, voltada para sistemas Linux 32 bits.
- Foi implementada uma versão inicial do DTS, a partir de uma extensão de um controlador OpenFlow *open-source*. Essa versão já apresenta uma estrutura que poderá ser ampliada em trabalhos futuros.
- Foi implementada uma biblioteca com todos os protocolos de controle da arquitetura ETArch, chamada de *Biblioteca FINLAN*, ou apenas *FinLib*. Essa biblioteca também apresenta uma abstração de *socket*, o que facilita a sua utilização, e mantém o contrato estabelecido entre a camada de transporte e aplicação.
- Foram descritos os principais cenários beneficiados com a agregação de tráfego via *multicast*, bem como principais aplicações da WEB destes cenários, quais as estratégias utilizadas, e como a abordagem proposta poderia ser utilizada.
- Foram desenvolvidas duas aplicações, uma de *chatting* e outra de *streaming* de vídeo, específicas para os experimentos. Essas aplicações apresentam suporte às três abordagens utilizadas, *IP Unicast*, *IP Multicast* e *ETArch Workspace*.

No curso dessa pesquisa foram realizadas diversas publicações, tanto nacionais quanto internacionais. Abaixo a lista de publicações.

- *Multicast Traffic Aggregation through Entity Title Model* - AICT 2014 – [Amaral Gonçalves et al. 2014].
- *Cross Layers Semantic Experimentation for Future Internet* – SBRC/WPEIF 2013 – [Dias et al. 2012].
- *Implementing the Domain Title Service atop OpenFlow* – SBRC/WPEIF 2013 – [Oliveira Silva et al. 2012].

- *On The Analysis of Multicast Traffic Over The Entity Title Architecture* – ICON 2012 – [de Oliveira Silva et al. 2012c].
- *Semantically Enriched Services to Understand the Need of Entities* – LNCS e FIA 2012 – [de Oliveira Silva et al. 2012a, de Oliveira Silva et al. 2012b].

6.3 Perspectivas Futuras

Espera-se que a Arquitetura ETArch evolua até um estágio que viabilize a sua implantação nas redes atuais. Dessa maneira, os trabalhos futuros contemplarão prioritariamente a evolução de aspectos-chave da arquitetura, tais como *segurança*, *mobilidade*, *roteamento*, sustentabilidade, QoS/QoE etc. Estes trabalhos atualmente encontram-se em curso e serão apresentados em dissertações e teses futuras.

Com o estreitamento das relações entre o grupo MEHAR e os pesquisadores europeus, acredita-se que será possível a realização de um trabalho conjunto, em prol de uma abordagem única para as redes futuras. Além disso, a possibilidade de utilização das *testbeds* europeias representa uma grande oportunidade de experimentação da arquitetura ETArch em larga escala, o que já começou a ser explorado em linhas de pesquisa paralela.

Por fim, acredita-se numa convergência entre o *Modelo de Título* e as abordagens SDN e NFV, que representam as principais propostas para as redes futuras. Há diversas similaridades entre as propostas que podem se complementar, de maneira a conceber uma nova e forte arquitetura para a Internet.

Referências

[Adams et al. 2005] Adams, A., Nicholas, J., and Siadak, W. (2005). Protocol Independent Multicast - Dense Mode (PIM-DM): Protocol Specification (Revised). RFC 3973 (Experimental). Internet Engineering Task Force. Number 3973 in Request for Comments.

[Aguiar 2008] Aguiar, R. L. (2008). Some comments on hourglasses. *SIGCOMM Comput. Commun. Rev.*, 38(5):69–72.

[Amaral Gonçalves et al. 2014] Amaral Gonçalves, M., de Oliveira Silva, F., de Souza Pereira, J. H., and Frosi Rosa, P. (2014). Multicast traffic aggregation through entity title model. In *AICT 2014, The Tenth Advanced International Conference on Telecommunications*, pages 175–180.

[ATNOG 2012] ATNOG (2012). EDOBRA (extending and deploying OFELIA in BRAzil).

[Baran 1964] Baran, P. (1964). On distributed communications networks. *IEEE Transactions on Communications Systems*, 12(1):1–9.

[Big Switch 2012] Big Switch (2012). Floodlight OpenFlow controller. http://floodlight.openflowhub.org/.

[Boudani and Cousin 2003] Boudani, A. and Cousin, B. (2003). SEM: a new small group multicast routing protocol. In *10th International Conference on Telecommunications, 2003. ICT 2003*, volume 1, pages 450–455.

[Cerf et al. 1974] Cerf, V., Dalal, Y., and Sunshine, C. (1974). Specification of Internet Transmission Control Program. RFC 675. Internet Engineering Task Force. Number 675 in Request for Comments.

[Cerf and Kahn 1974] Cerf, V. and Kahn, R. (1974). A protocol for packet network intercommunication. *Communications, IEEE Transactions on*, 22(5):637–648.

[Clark et al. 2004] Clark, D., Braden, R., Sollins, K., Wroclawski, J., and Katabi, D. (2004). New ARCH: Future generation internet architecture. Technical report, MIT Computer Science & AI Lab.

[Clark et al. 1991] Clark, D., Chapin, L., Cerf, V., Braden, R., and Hobby, R. (1991). Towards the Future Internet Architecture. RFC 1287 (Informational). Internet Engineering Task Force. Number 1287 in Request for Comments.

[Comer 2000] Comer, D. E. (2000). *Internetworking with TCP/IP: Principles, Protocols and Architecture v.1: Principles, Protocols, and Architecture: Principles, Protocols and Architecture Vol 1.* Addison Wesley, Upper Saddle River, N.J, 4 edition edition.

[Commission 2012a] Commission, E. (2012a). ICT challenge 1: Pervasive and trusted network and service infrastructures.

[Commission 2012b] Commission, E. (2012b). The network of the future - projects. http://cordis.europa.eu/fp7/ict/future-networks/projects_en.html.

[Cormen et al. 2009] Cormen, T. H., Stein, C., Rivest, R. L., and Leiserson, C. E. (2009). *Introduction to algorithms.* Mit Press.

[Cui et al. 2012] Cui, C., Deng, H., Telekom, D., Michel, U., Damker, H., Italia, T., Guardini, I., Demaria, E., Minerva, R., and Manzalini, A. (2012). Network functions virtualisation. *SDN and OpenFlow World Congress.*

[Davies et al. 2007] Davies, E., Krishnan, S., and Savola, P. (2007). IPv6 Transition/Co-existence Security Considerations. RFC 4942 (Informational). Internet Engineering Task Force. Number 4942 in Request for Comments.

[Day 2008] Day, J. (2008). *Patterns in Network Architecture: A Return to Fundamentals (paperback): A Return to Fundamentals.* Prentice Hall, 1 edition.

[de Oliveira Silva et al. 2012a] de Oliveira Silva, F., Dias, A., Ferreira, C., De Souza Santos, E., Pereira, F., de Andrade, I., de Souza Pereira, J., Camargos, L., Theodoro, L., Gonçalves, M., Pasquini, R., Venâncio Neto, A., Rosa, P., and Kofuji, S. (2012a). Semantically enriched services to understand the need of entities. In Álvarez, F., Cleary, F., Daras, P., Domingue, J., Galis, A., Garcia, A., Gavras, A., Karnourskos, S., Krco, S., Li, M.-S., Lotz, V., Müller, H., Salvadori, E., Sassen, A.-M., Schaffers, H., Stiller, B., Tselentis, G., Turkama, P., and Zahariadis, T., editors, *The Future Internet*, volume 7281 of *Lecture Notes in Computer Science*, page 142–153. Springer Berlin / Heidelberg.

[de Oliveira Silva et al. 2012b] de Oliveira Silva, F., Dias, A., Ferreira, C., Santos, E., Pereira, F., Andrade, I., Pereira, J., Camargos, L., Theodoro, L., Gonçalves, M., Pasquini, R., Venâncio Neto, A., Rosa, P., and Kofuji, S. (2012b). Semantically enriched services to understand the need of entities. In Álvarez, F., Cleary, F., Daras, P., Domingue, J., Galis, A., Garcia, A., Gavras, A., Karnourskos, S., Krco, S., Li, M.-S., Lotz, V., Müller, H., Salvadori, E., Sassen, A.-M., Schaffers, H., Stiller, B., Tselentis, G., Turkama, P., and Zahariadis, T., editors, *The Future Internet*, volume 7281 of *Lecture Notes in Computer Science*, pages 142–153. Springer Berlin Heidelberg.

[de Oliveira Silva et al. 2012c] de Oliveira Silva, F., Goncalves, M., de Souza Pereira, J., Pasquini, R., Rosa, P., and Kofuji, S. (2012c). On the analysis of multicast traffic over the entity title architecture. In *2012 18th IEEE International Conference on Networks (ICON)*, page 30–35.

[Deering 1986] Deering, S. (1986). Host extensions for IP multicasting. RFC 988. Internet Engineering Task Force. Number 988 in Request for Comments. Obsoleted by RFCs 1054, 1112.

[Deering 1988] Deering, S. (1988). Host extensions for IP multicasting. RFC 1054. Internet Engineering Task Force. Number 1054 in Request for Comments. Obsoleted by RFC 1112.

[Deering and Cheriton 1985] Deering, S. and Cheriton, D. (1985). Host groups: A multicast extension to the Internet Protocol. RFC 966. Internet Engineering Task Force. Number 966 in Request for Comments. Obsoleted by RFC 988.

[Deering and Hinden 1995] Deering, S. and Hinden, R. (1995). Internet Protocol, Version 6 (IPv6) Specification. RFC 1883 (Proposed Standard). Internet Engineering Task Force. Number 1883 in Request for Comments. Obsoleted by RFC 2460.

[Delgrossi and Berger 1995] Delgrossi, L. and Berger, L. (1995). Internet Stream Protocol Version 2 (ST2) Protocol Specification - Version ST2+. RFC 1819 (Experimental). Internet Engineering Task Force. Number 1819 in Request for Comments.

[Dias et al. 2012] Dias, A., Santos, E., Pereira, F., Oliveira Silva, F., Pereira, J., Camargo, L., Theodoro, L., Gonçalves, M., Pasquini, R., Rosa, P., and Kofuji, S. (2012). Cross layers semantic experimentation for future internet. In *XXX Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos. Anais do III Workshop de Pesquisa Experimental na Internet do Futuro (WPEIF)*, pages 16–19, Ouro Preto, Brasil. Sociedade Brasileira de Computação.

[Dijkstra 1982] Dijkstra, P. D. E. W. (1982). On the role of scientific thought. In *Selected Writings on Computing: A personal Perspective*, Texts and Monographs in Computer Science, pages 60–66. Springer New York.

[Diot et al. 2000] Diot, C., Levine, B., Lyles, B., Kassem, H., and Balensiefen, D. (2000). Deployment issues for the IP multicast service and architecture. *IEEE Network*, 14(1):78 –88.

[D'Ambrosio et al. 2010] D'Ambrosio, M., Marchisio, M., Vercellone, V., Ahlgren, B., and Dannewitz, C. (2010). *4WARD. Second NetInf architecture description.* 4WARD.

[Egevang and Francis 1994] Egevang, K. and Francis, P. (1994). The IP Network Address Translator (NAT). RFC 1631 (Informational). Internet Engineering Task Force. Number 1631 in Request for Comments. Obsoleted by RFC 3022.

[Estrin et al. 1997] Estrin, D., Farinacci, D., Helmy, A., Thaler, D., Deering, S., Handley, M., Jacobson, V., Liu, C., Sharma, P., and Wei, L. (1997). Protocol Independent Multicast-Sparse Mode (PIM-SM): Protocol Specification. RFC 2117 (Experimental). Internet Engineering Task Force. Number 2117 in Request for Comments. Obsoleted by RFC 2362.

[Eurescom 2012] Eurescom (2012). Future internet assembly - european future internet portal.

[Fernandes and Rothenberg 2014] Fernandes, E. L. and Rothenberg, C. E. (2014). Openflow 1.3 software switch. *Anais do XXXII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos.*

[Fisher 2007] Fisher, D. (2007). US national science foundation and the future internet design. *ACM SIGCOMM Computer Communication Review*, 37(3):85–87.

[Forgie 1979] Forgie, J. W. (1979). ST–A Proposed Internet Stream Protocol. Bolt, Beranek and Newman, Inc. (BBN), Massachusetts Institute of Technology (MIT). Lincoln Laboratory. Number 119 in Internet Experiment Note.

[Foster et al. 2013] Foster, N., Guha, A., Reitblatt, M., Story, A., Freedman, M. J., Katta, N. P., Monsanto, C., Reich, J., Rexford, J., Schlesinger, C., Walker, D., and Harrison, M. R. (2013). Languages for Software-Defined Networks. *IEEE Communications Magazine*, page 128.

[Foundation 2011] Foundation, N. S. (2011). NSF future internet architecture project. http://www.nets-fia.net/.

[Fuller et al. 1993] Fuller, V., Li, T., Yu, J., and Varadhan, K. (1993). Classless Inter-Domain Routing (CIDR): an Address Assignment and Aggregation Strategy. RFC 1519 (Proposed Standard). Internet Engineering Task Force. Number 1519 in Request for Comments. Obsoleted by RFC 4632.

[Gamma et al. 1994] Gamma, E., Helm, R., Johnson, R., and Vlissides, J. (1994). *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley Professional, Reading, Mass, 1 edition edition.

[Goth 2011] Goth, G. (2011). Software-defined networking could shake up more than packets. *IEEE Internet Computing*, 15(4):6–9.

[Greene 2009] Greene, K. (2009). TR10: software-defined networking. *MIT Technology Review*, 112(2).

[Group et al. 1996] Group, A.-V. T. W., Schulzrinne, H., Casner, S., Frederick, R., and Jacobson, V. (1996). RTP: A Transport Protocol for Real-Time Applications. RFC 1889 (Proposed Standard). Internet Engineering Task Force. Number 1889 in Request for Comments. Obsoleted by RFC 3550.

[Group and Hinden 1993] Group, I. E. S. and Hinden, R. (1993). Applicability Statement for the Implementation of Classless Inter-Domain Routing (CIDR). RFC 1517 (Historic). Internet Engineering Task Force. Number 1517 in Request for Comments.

[Hedrick 1988] Hedrick, C. (1988). Routing Information Protocol. RFC 1058 (Historic). Internet Engineering Task Force. Number 1058 in Request for Comments. Updated by RFCs 1388, 1723.

[Hinden 1994] Hinden, R. (1994). Simple Internet Protocol Plus White Paper. RFC 1710 (Informational). Internet Engineering Task Force. Number 1710 in Request for Comments.

[Hinden and Deering 1998] Hinden, R. and Deering, S. (1998). IPv6 Multicast Address Assignments. RFC 2375 (Informational). Internet Engineering Task Force. Number 2375 in Request for Comments.

[Hinden and Deering 2006] Hinden, R. and Deering, S. (2006). IP Version 6 Addressing Architecture. RFC 4291 (Draft Standard). Internet Engineering Task Force. Number 4291 in Request for Comments. Updated by RFCs 5952, 6052, 7136.

[Horridge et al. 2007] Horridge, M., Bechhofer, S., and Noppens, O. (2007). Igniting the OWL 1.1 touch paper: The OWL API. In *OWLED*, volume 258, pages 6–7. Citeseer.

[Horridge et al. 2012] Horridge, M., Musen, M., Nyulas, C., Tu, S., and Tudorache, T. (2012). protégé.

[Hosseini et al. 2007] Hosseini, M., Ahmed, D. T., Shirmohammadi, S., and Georganas, N. D. (2007). A survey of application-layer multicast protocols. *Commun. Surveys Tuts.*, 9(3):58–74.

[Huang and Luo 2014] Huang, Y. and Luo, T.-j. (2014). *Pervasive computing and the networked world: Joint International Conference, ICPCA/SWS 2013, Vina del Mar, Chile, December 5-7, 2013. Revised Selected Papers*. Number 8351 in Lecture notes in computer science. Springer, 1st edition edition.

[Johnson and Foote 1988] Johnson, R. E. and Foote, B. (1988). Designing reusable classes. *Journal of object-oriented programming*, 1(2):22–35.

[Kiczales et al. 1997] Kiczales, G., Lamping, J., Mendhekar, A., Maeda, C., Lopes, C., Loingtier, J.-m., and Irwin, J. (1997). Aspect-oriented programming. In *ECOOP*. SpringerVerlag.

[Kim and Feamster 2013] Kim, H. and Feamster, N. (2013). Improving Network Management with Software Defined Networking. *IEEE Communications Magazine*, page 114.

[Lacy 2005] Lacy, L. W. (2005). *OWL: representing information using the web ontology language*. Trafford, Victoria, BC, Canada.

[Lantz et al. 2010] Lantz, B., Heller, B., and McKeown, N. (2010). A network in a laptop: Rapid prototyping for software-defined networks. In *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*, Hotnets-IX, page 19:1–19:6, New York, NY, USA. ACM.

[Leśniewski 1930] Leśniewski, S. (1930). Comptes rendus des séances de la Société des Sciences et des Lettres de Varsovie. *Class III*, pages 111–132.

[Lougheed and Rekhter 1989] Lougheed, K. and Rekhter, Y. (1989). Border Gateway Protocol (BGP). RFC 1105 (Experimental). Internet Engineering Task Force. Number 1105 in Request for Comments. Obsoleted by RFC 1163.

[Miller 1990] Miller, M. A. (1990). *PCIM '90 Proceedings of the 18th International Intelligent Motion Conference, Oct. 1990*. Intertec Communications, Inc., 18th edition edition.

[Mockapetris 1983a] Mockapetris, P. (1983a). Domain names: Concepts and facilities. RFC 882. Internet Engineering Task Force. Number 882 in Request for Comments. Obsoleted by RFCs 1034, 1035, updated by RFC 973.

[Mockapetris 1983b] Mockapetris, P. (1983b). Domain names: Implementation specification. RFC 883. Internet Engineering Task Force. Number 883 in Request for Comments. Obsoleted by RFCs 1034, 1035, updated by RFC 973.

[Mockapetris 1987a] Mockapetris, P. (1987a). Domain names - concepts and facilities. RFC 1034 (INTERNET STANDARD). Internet Engineering Task Force. Number 1034 in Request for Comments. Updated by RFCs 1101, 1183, 1348, 1876, 1982, 2065, 2181, 2308, 2535, 4033, 4034, 4035, 4343, 4035, 4592, 5936.

[Mockapetris 1987b] Mockapetris, P. (1987b). Domain names - implementation and specification. RFC 1035 (INTERNET STANDARD). Internet Engineering Task Force. Number 1035 in Request for Comments. Updated by RFCs 1101, 1183, 1348, 1876, 1982, 1995, 1996, 2065, 2136, 2181, 2137, 2308, 2535, 2673, 2845, 3425, 3658, 4033, 4034, 4035, 4343, 5936, 5966, 6604.

[Moore 2006] Moore, G. E. (2006). Cramming more components onto integrated circuits, reprinted from electronics, volume 38, number 8, april 19, 1965, pp. 114 ff. *Solid-State Circuits Society Newsletter, IEEE*, 11(5):33–35.

[Moy 1989] Moy, J. (1989). OSPF specification. RFC 1131 (Proposed Standard). Internet Engineering Task Force. Number 1131 in Request for Comments. Obsoleted by RFC 1247.

[Moy 1994] Moy, J. (1994). MOSPF: Analysis and Experience. RFC 1585 (Informational). Internet Engineering Task Force. Number 1585 in Request for Comments.

[Oliveira Silva 2013] Oliveira Silva, F. (2013). *Endereçamento por Título: Uma Forma de Encaminhamento Multicast para a Próxima Geração de Redes de Computadores*. PhD thesis, Universidade de São Paulo, Escola Politécnica, São Paulo, Brasil.

[Oliveira Silva et al. 2013] Oliveira Silva, F., Corujo, D., Guimarães, C., Souza Pereira, J., Rosa, P., Kofuji, S., and Aguiar, R. (2013). Enabling network mobility by using IEEE 802.21 integrated with the entity title architecture. In *XXI Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos. Anais do IV Workshop de Pesquisa Experimental na Internet do Futuro (WPEIF)*, pages 29–34, Brasília. Sociedade Brasileira de Computação.

[Oliveira Silva et al. 2012] Oliveira Silva, F., Gonçalves, M., Pereira, J., Camargo, L., Pasquini, R., Rosa, P., and Kofuji, S. (2012). Implementing the domain title service atop openflow. In *XXX Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos. Anais do III Workshop de Pesquisa Experimental na Internet do Futuro (WPEIF)*, pages 38–41, Ouro Preto, Brasil. Sociedade Brasileira de Computação.

[Padlipsky 1982] Padlipsky, M. (1982). Perspective on the ARPANET reference model. RFC 871. Internet Engineering Task Force. Number 871 in Request for Comments.

[Pereira et al. 2011] Pereira, J. d. S., de Oliveira Silva, F., Filho, E., Kofuji, S., and Rosa, P. (2011). Title model ontology for future internet networks. In Domingue, J., Galis, A., Gavras, A., Zahariadis, T., Lambert, D., Cleary, F., Daras, P., Krco, S., Mÿller, H., Li, M.-S., Schaffers, H., Lotz, V., Alvarez, F., Stiller, B., Karnouskos, S., Avessta, S., and Nilsson, M., editors, *The Future Internet*, volume 6656 of *Lecture Notes in Computer Science*, pages 103–114. Springer Berlin / Heidelberg. 10.1007/978-3-642-20898-0_8.

[Pereira 2012] Pereira, J. H. d. S. (2012). *Modelo de Título para a Próxima Geração de Internet*. PhD thesis, Universidade de São Paulo, São Paulo.

[Plummer 1982] Plummer, D. (1982). Ethernet Address Resolution Protocol: Or Converting Network Protocol Addresses to 48.bit Ethernet Address for Transmission on Ethernet Hardware. RFC 826 (INTERNET STANDARD). Internet Engineering Task Force. Number 826 in Request for Comments. Updated by RFCs 5227, 5494.

[Postel 1980a] Postel, J. (1980a). DoD standard Internet Protocol. RFC 760. Internet Engineering Task Force. Number 760 in Request for Comments. Obsoleted by RFC 791, updated by RFC 777.

[Postel 1980b] Postel, J. (1980b). DoD standard Transmission Control Protocol. RFC 761. Internet Engineering Task Force. Number 761 in Request for Comments. Obsoleted by RFC 793.

[Postel 1980c] Postel, J. (1980c). User Datagram Protocol. RFC 768 (INTERNET STANDARD). Internet Engineering Task Force. Number 768 in Request for Comments.

[Postel 1981a] Postel, J. (1981a). Internet Control Message Protocol. RFC 792 (INTERNET STANDARD). Internet Engineering Task Force. Number 792 in Request for Comments. Updated by RFCs 950, 4884, 6633, 6918.

[Postel 1981b] Postel, J. (1981b). Internet Protocol. RFC 791 (INTERNET STANDARD). Internet Engineering Task Force. Number 791 in Request for Comments. Updated by RFCs 1349, 2474, 6864.

[Postel 1981c] Postel, J. (1981c). Transmission Control Protocol. RFC 793 (INTERNET STANDARD). Internet Engineering Task Force. Number 793 in Request for Comments. Updated by RFCs 1122, 3168, 6093, 6528.

[Postel 1983] Postel, J. (1983). Domain names plan and schedule. RFC 881. Internet Engineering Task Force. Number 881 in Request for Comments. Updated by RFC 897.

[Rekhter and Li 1995] Rekhter, Y. and Li, T. (1995). A Border Gateway Protocol 4 (BGP-4). RFC 1771 (Draft Standard). Internet Engineering Task Force. Number 1771 in Request for Comments. Obsoleted by RFC 4271.

[Rekhter et al. 1996] Rekhter, Y., Moskowitz, B., Karrenberg, D., de Groot, G. J., and Lear, E. (1996). Address Allocation for Private Internets. RFC 1918 (Best Current Practice). Internet Engineering Task Force. Number 1918 in Request for Comments. Updated by RFC 6761.

[Rexford and Dovrolis 2010] Rexford, J. and Dovrolis, C. (2010). Future internet architecture: clean-slate versus evolutionary research. *Communications of the ACM*, 53(9):36–40.

[Roberts 2009] Roberts, J. (2009). The clean-slate approach to future internet design: a survey of research initiatives. *annals of telecommunications - annales des télécommunications*, 64(5-6):271–276.

[Romdhani et al. 2004] Romdhani, I., Kellil, M., Lach, H.-Y., Bouabdallah, A., and Bettahar, H. (2004). IP mobile multicast: Challenges and solutions. *IEEE Communications Surveys Tutorials*, 6(1):18–41.

[Rosen 1982] Rosen, E. (1982). Exterior Gateway Protocol (EGP). RFC 827. Internet Engineering Task Force. Number 827 in Request for Comments. Updated by RFC 904.

[Rothenberg et al. 2012] Rothenberg, C., Nascimento, M., Salvador, M. R., Corrêa, C., Lucena, S., Vidal, A., and Verdi, F. (2012). Revisiting IP Routing Control Platforms with OpenFlow-based Software-Defined Networks. *XXX SBRC - III Workshop de Pesquisa Experimental da Internet do Futuro-WPEIF*, page 6.

[Sallent et al. 2012] Sallent, S., Abelém, A., Machado, I., Bergesio, L., Fdida, S., Rezende, J., Azodolmolky, S., Salvador, M., Ciuffo, L., and Tassiulas, L. (2012). Fibre project: Brazil and europe unite forces and testbeds for the internet of the future. In *Testbeds and Research Infrastructure. Development of Networks and Communities*, pages 372–372. Springer.

[Seskar et al. 2011] Seskar, I., Nagaraja, K., Nelson, S., and Raychaudhuri, D. (2011). MobilityFirst future internet architecture project. In *Proceedings of the 7th Asian Internet Engineering Conference*, AINTEC '11, page 1–3, New York, NY, USA. ACM.

[Silva et al. 2014a] Silva, F., Castillo-Lema, J., Neto, A., Silva, F., Rosa, P., Corujo, D., and Guimaraes, C. (2014a). Quality-oriented mobility control architecture for etarch handover optimization. In *SBRC 2014 - WPEIF ()*.

[Silva et al. 2014b] Silva, F., Ferreira, C. C., Neto, N., Mota, A., Theodoro, L. C., Pereira, J., Neto, A., and Rosa, P. (2014b). Enabling a carrier grade sdn by using a top-down approach. In *SBRC 2014 - WPEIF ()*.

[Simpson 1996] Simpson, W. (1996). PPP Link Quality Monitoring. RFC 1989 (Draft Standard). Internet Engineering Task Force. Number 1989 in Request for Comments.

[STANFORD 2006] STANFORD (2006). Clean slate design for the internet.

[Stewart et al. 2000] Stewart, R., Xie, Q., Morneault, K., Sharp, C., Schwarzbauer, H., Taylor, T., Rytina, I., Kalla, M., Zhang, L., and Paxson, V. (2000). Stream Control Transmission Protocol. RFC 2960 (Proposed Standard). Internet Engineering Task Force. Number 2960 in Request for Comments. Obsoleted by RFC 4960, updated by RFC 3309.

[Suñé et al. 2014] Suñé, M., Bergesio, L., Woesner, H., Rothe, T., Köpsel, A., Colle, D., Puype, B., Simeonidou, D., Nejabati, R., Channegowda, M., Kind, M., Dietz, T., Autenrieth, A., Kotronis, V., Salvadori, E., Salsano, S., Körner, M., and Sharma, S. (2014). Design and implementation of the {OFELIA} {FP7} facility: The european openflow testbed. *Computer Networks*, 61(0):132 – 150. Special issue on Future Internet Testbeds – Part I.

[Tanenbaum 2002] Tanenbaum, A. (2002). *Computer Networks*. Prentice Hall Professional Technical Reference, 4th edition.

[Thaler et al. 2000] Thaler, D., Handley, M., and Estrin, D. (2000). The Internet Multicast Address Allocation Architecture. RFC 2908 (Historic). Internet Engineering Task Force. Number 2908 in Request for Comments. Obsoleted by RFC 6308.

[Topolcic 1990] Topolcic, C. (1990). Experimental Internet Stream Protocol: Version 2 (ST-II). RFC 1190 (Experimental). Internet Engineering Task Force. Number 1190 in Request for Comments. Obsoleted by RFC 1819.

[Waitzman et al. 1988] Waitzman, D., Partridge, C., and Deering, S. (1988). Distance Vector Multicast Routing Protocol. RFC 1075 (Experimental). Internet Engineering Task Force. Number 1075 in Request for Comments.

[Wei et al. 2010] Wei, Z., Liguo, D., and Junjie, C. (2010). Reasoning and realization based on ontology model and jena. In *2010 IEEE Fifth International Conference on Bio-Inspired Computing: Theories and Applications (BIC-TA)*, pages 1057–1060.

[Wood 2011] Wood, G. (2011). IPv6: making room for the world on the future internet. *IEEE Internet Computing*, 15(4):88–89.

[Yiu and Chan 2008] Yiu, W.-P. K. and Chan, S.-H. G. (2008). Offering data confidentiality for multimedia overlay multicast: Design and analysis. *ACM Trans. Multimedia Comput. Commun. Appl.*, 5(2):13:1–13:23.

[Zahariadis et al. 2011] Zahariadis, T., Papadimitriou, D., Tschofenig, H., Haller, S., Daras, P., Stamoulis, G. D., and Hauswirth, M. (2011). Towards a future internet architecture. In Domingue, J., Galis, A., Gavras, A., Zahariadis, T., and Lambert, D., editors, *The Future Internet. Future Internet Assembly 2011: Achievements and Technological Promises*, volume 6656 of *LNCS*, pages 7–18. Springer-Verlag, Berlin, Heidelberg.

Apêndice A

Script Python com a Topologia do Mininet utilizada nos Experimentos

```
"""Custom topology used in Finlan Experiments

Adding the 'topos' dict with a key/value pair to generate our
    newly defined topology enables one to pass in '--topo=
   mytopo' from the command line. """

from mininet.topo import Topo, Node

class MyTopo( Topo ):
  "ETArch Experiment Topology."

  def __init__( self, enable_all = True ):
    "Create custom topo."

    # Add default members to class.
    super( MyTopo, self ).__init__()

    # Set Node IDs for hosts and switches
    sw1 = 1
    sw2 = 2
    sw3 = 3
```

```
20      sw4 = 4
21      sw5 = 5
22      sw6 = 6
23      h1 = 7
24      h2 = 8
25      h3 = 9
26      h4 = 10
27      h5 = 11
28      h6 = 12
29      h7 = 13
30      h8 = 14
31      h9 = 15
32
33      # Add nodes
34      self.add_node( sw1, Node( is_switch=True ) )
35      self.add_node( sw2, Node( is_switch=True ) )
36      self.add_node( sw3, Node( is_switch=True ) )
37      self.add_node( sw4, Node( is_switch=True ) )
38      self.add_node( sw5, Node( is_switch=True ) )
39      self.add_node( sw6, Node( is_switch=True ) )
40      self.add_node( h1, Node( is_switch=False ) )
41      self.add_node( h2, Node( is_switch=False ) )
42      self.add_node( h3, Node( is_switch=False ) )
43      self.add_node( h4, Node( is_switch=False ) )
44      self.add_node( h5, Node( is_switch=False ) )
45      self.add_node( h6, Node( is_switch=False ) )
46      self.add_node( h7, Node( is_switch=False ) )
47      self.add_node( h8, Node( is_switch=False ) )
48      self.add_node( h9, Node( is_switch=False ) )
49
50      # Add edges
51      self.add_edge( sw1, sw2 );
52      self.add_edge( sw2, sw3 );
53      self.add_edge( sw3, sw1 );
54      self.add_edge( sw3, sw4 );
55      self.add_edge( sw4, sw5 );
56      self.add_edge( sw4, sw6 );
57
58      self.add_edge( h1, sw2 )
```

```
59      self.add_edge( h2, sw2 )
60      self.add_edge( h3, sw5 )
61      self.add_edge( h4, sw4 )
62      self.add_edge( h5, sw6 )
63      self.add_edge( h6, sw6 )
64      self.add_edge( h7, sw4 )
65      self.add_edge( h8, sw3 )
66      self.add_edge( h9, sw1 )
67
68
69      # Consider all switches and hosts 'on'
70      self.enable_all()
71
72  topos = { 'finlan': ( lambda: MyTopo() ) }
```

Apêndice

B

Inicialização do Mininet e Fluxos Utilizados no Workspace

Inicialização da rede simulada pelo Mininet.

```
root@NetExpVM:~# mn --custom /home/openflow/
    customComplexTopos.py --topo finlan
custom in sys.argv
*** Adding controller
*** Creating network
*** Adding hosts:
h1 h2 h3
*** Adding switches:
s4 s5 s6
*** Adding links:
(h1, s4) (h2, s5) (h3, s6) (s4, s5) (s5, s6)
*** Configuring hosts
h1 h2 h3
*** Starting controller
*** Starting 3 switches
s4 s5 s6
*** Starting CLI:
mininet>
```

Fluxos depois de iniciar uma aplicação 'publisher' no host1 e uma 'subscriber' no host3.

```
mininet> dpctl dump-flows
*** s4
   ----------------------------------------------------------------------------

stats_reply (xid=0xfaa33da8): flags=none type=1(flow)
  cookie=0, duration_sec=8s, duration_nsec=971000000s,
     table_id=0, priority=0, n_packets=1766, n_bytes=2506564,
      idle_timeout=0,hard_timeout=30,dl_vlan=0x0064,actions=
     output:2,output:1
*** s5
   ----------------------------------------------------------------------------

stats_reply (xid=0xe4e84504): flags=none type=1(flow)
  cookie=0, duration_sec=9s, duration_nsec=70000000s,
     table_id=0, priority=0, n_packets=1822, n_bytes=2581487,
      idle_timeout=0,hard_timeout=30,dl_vlan=0x0064,actions=
     output:2,output:3
*** s6
   ----------------------------------------------------------------------------

stats_reply (xid=0xf1b2f907): flags=none type=1(flow)
  cookie=0, duration_sec=9s, duration_nsec=167000000s,
     table_id=0, priority=0, n_packets=962, n_bytes=1331035,
     idle_timeout=0,hard_timeout=30,dl_vlan=0x0064,actions=
     output:1,output:2
mininet>
```

Fluxos depois de iniciar uma aplicação 'subscriber' no host2.

```
mininet> dpctl dump-flows
*** s4
   ----------------------------------------------------------------------------

stats_reply (xid=0xa25eac26): flags=none type=1(flow)
  cookie=0, duration_sec=5s, duration_nsec=889000000s,
     table_id=0, priority=0, n_packets=12360, n_bytes
     =17264947, idle_timeout=0,hard_timeout=30,dl_vlan=0x0064
     ,actions=output:2,output:1
```

```
*** s5
    ----------------------------------------------------------------------

stats_reply (xid=0x6c688d89): flags=none type=1(flow)
  cookie=0, duration_sec=5s, duration_nsec=980000000s,
     table_id=0, priority=0, n_packets=456, n_bytes=648009,
     idle_timeout=0,hard_timeout=30,dl_vlan=0x0064,actions=
     output:2,output:3,output:1
*** s6
    ----------------------------------------------------------------------

stats_reply (xid=0x91421000): flags=none type=1(flow)
  cookie=0, duration_sec=6s, duration_nsec=38000000s,
     table_id=0, priority=0, n_packets=11545, n_bytes
     =16077879, idle_timeout=0,hard_timeout=30,dl_vlan=0x0064
     ,actions=output:1,output:2
```

Apêndice

C

FinLib : DL-Ontology - Physical Medium Access

```
1  #include <jni.h>
2  #include "br_ufu_facom_network_dlop_FinSocket.h"
3
4  #include <net/if.h>
5  #include <netinet/ether.h>
6  #include <sys/ioctl.h>
7  #include <stdio.h>
8  #include <string.h>
9  #include <malloc.h>
10 #include <sys/socket.h>
11 #include <net/ethernet.h>
12 #include <linux/if_packet.h>
13 #include <math.h>
14
15 #include <errno.h>
16
17 /*
18  * Class:     FinSocket
19  * Method:    finOpen
20  * Signature: ()I
21  */
```

```
22 JNIEXPORT jint JNICALL
      Java_br_ufu_facom_network_dlop_FinSocket_finOpen
23   (JNIEnv * env , jobject obj){
24   int s;
25
26   s = socket(AF_PACKET, SOCK_RAW, htons(ETH_P_ALL));
27   if (s == -1)
28     printf("socket error\n");
29
30   return s;
31 }
32
33 /*
34  * Class:     FinSocket
35  * Method:    finClose
36  * Signature: (I)Z
37  */
38 JNIEXPORT jboolean JNICALL
      Java_br_ufu_facom_network_dlop_FinSocket_finClose
39   (JNIEnv *env, jobject obj, jint sock){
40
41   return close(sock);
42 }
43
44 /*
45  * Class:     FinSocket
46  * Method:    finWrite
47  * Signature: (Ljava/lang/String;Ljava/lang/String;I)Z
48  */
49 JNIEXPORT jboolean JNICALL
      Java_br_ufu_facom_network_dlop_FinSocket_finWrite
50   (JNIEnv * env, jobject obj, jint ifIndex, jint soc,
       jbyteArray data, jint offset, jint len){
51     int result;
52     jbyte *buf;
53
54     /*target address*/
55     struct sockaddr_ll socket_address;
56
```

```
57      /*we don't use a protocol above ethernet layer. Just use
           anything here*/
58      socket_address.sll_protocol = htons(ETH_P_ALL);
59
60      socket_address.sll_ifindex  = ifIndex;
61
62      /*address size*/
63      socket_address.sll_halen = ETH_ALEN;
64
65      buf = (*env)->GetByteArrayElements(env, data, NULL);
66
67      if(finSelect(soc,0,5,0)<0)
68          return 1;
69
70      result = sendto(soc, &buf[offset], len, 0,(struct
           sockaddr*)&socket_address, sizeof(socket_address));
71
72      if(result < 0){
73          printf("Error sending the packet: %d - Size %d - %s\n
               ", errno, len,strerror( errno ) );
74      }
75
76      (*env)->ReleaseByteArrayElements(env, data, buf,
           JNI_ABORT);
77
78
79      return result > 0;
80  }
81
82  JNIEXPORT jint JNICALL
       Java_br_ufu_facom_network_dlop_FinSocket_finRead
83    (JNIEnv * env, jobject obj, jint soc, jbyteArray data, jint
         offset, jint len){
84
85      jbyte *buf;
86      int result;
87
88      buf = (*env)->GetByteArrayElements(env, data, NULL);
89
```

```
90      result = recv(soc, buf+offset, len, 0);
91
92      (*env)->ReleaseByteArrayElements(env, data, buf, 0);
93
94      return result;
95  }
96
97  JNIEXPORT jboolean JNICALL
       Java_br_ufu_facom_network_dlop_FinSocket_setPromiscousMode
98    (JNIEnv * env, jobject obj, jstring ifName,  jint soc){
99      struct ifreq ifr;
100
101     // this procedure sets the interface in promiscous mode
102     const char *ifNameChars = (*env)->GetStringUTFChars(env,
          ifName, 0);
103     strcpy(ifr.ifr_name, ifNameChars);
104     (*env)->ReleaseStringUTFChars(env, ifName, ifNameChars);
105
106     if(ioctl(soc, SIOCGIFINDEX, &ifr) < 0) return 0;
107     if(ioctl(soc, SIOCGIFFLAGS, &ifr) < 0) return 0;
108     ifr.ifr_flags |= IFF_PROMISC;
109     if(ioctl(soc, SIOCSIFFLAGS, &ifr) < 0) return 0;
110
111     return 1;
112 }
113
114 int finSelect(int socket, int read, int seconds, int
       microseconds){
115   int result;
116   struct timeval timeout;
117   fd_set *rset = NULL, *wset = NULL, errset, fdset;
118
119   FD_ZERO(&fdset);
120   FD_ZERO(&errset);
121   FD_SET(socket, &fdset);
122   FD_SET(socket, &errset);
123
124   timeout.tv_sec  = seconds;
125   timeout.tv_usec = microseconds;
```

```
126
127   if(read){
128     printf("Read! =0\n");
129     rset = &fdset;
130   }else
131     wset = &fdset;
132
133   result = select(socket + 1, rset, wset, &errset, &timeout);
134
135   if(result >= 0) {
136     if(FD_ISSET(socket, &errset)){
137       result = -1;
138     }else if(FD_ISSET(socket, &fdset))
139       result = 0;
140     else {
141       result = -1;
142     }
143   }
144   return result;
145 }
146
147 JNIEXPORT jobject JNICALL
     Java_br_ufu_facom_network_dlop_FinSocket_getNetIfs
148 (JNIEnv * env, jobject obj){
149     jclass mapClass = (*env)->FindClass(env, "java/util/
        HashMap");
150     jclass intClass = (*env)->FindClass(env, "java/lang/
        Integer");
151     jclass stringClass = (*env)->FindClass(env, "java/lang/
        String");
152
153     if(mapClass == NULL || intClass == NULL || stringClass ==
         NULL){
154         return NULL;
155     }
156
157     jmethodID initMap = (*env)->GetMethodID(env, mapClass, "<
        init>", "()V");
```

```
158     jmethodID initInt = (*env)->GetMethodID(env, intClass, "<
           init>", "(I)V");
159 
160     jobject hashMap = (*env)->NewObject(env, mapClass,
           initMap);
161 
162     jmethodID put = (*env)->GetMethodID(env, mapClass, "put",
           "(Ljava/lang/Object;Ljava/lang/Object;)Ljava/lang/
           Object;");
163 
164     //Searching for network interfaces
165     struct if_nameindex *pif;
166     struct if_nameindex *head;
167     head = pif = if_nameindex();
168 
169     while (pif->if_index) {
170 
171         jobject index = (*env)->NewObject(env, intClass,
               initInt, pif->if_index);
172         jstring name = (*env)->NewStringUTF(env, pif->if_name
               );
173 
174         (*env)->CallObjectMethod(env, hashMap, put, index,
               name);
175 
176         pif++;
177     }
178 
179     //Clean
180     if_freenameindex(head);
181     (*env)->DeleteLocalRef(env, mapClass);
182     (*env)->DeleteLocalRef(env, intClass);
183 
184     return hashMap;
185 }
```

Anexo

A

Descrição Ontológica do Modelo de Título - Código OWL

Publicado originalmente em [Pereira 2012].

```
<?xml version="1.0"?>

<!DOCTYPE rdf:RDF [
    <!ENTITY owl "http://www.w3.org/2002/07/owl#" >
    <!ENTITY dc "http://purl.org/dc/elements/1.1/" >
    <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
    <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >
    <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
    <!ENTITY Ontology1280888215552
    "http://www.semanticweb.org/ontologies/2010/7/Ontology1280888215552.
       owl#" >
]>

<rdf:RDF xmlns="http://www.semanticweb.org/ontologies/2010/7/
   Ontology1280888215552.owl#"
    xml:base="http://www.semanticweb.org/ontologies/2010/7/
       Ontology1280888215552.owl"
    xmlns:dc="http://purl.org/dc/elements/1.1/"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
```

```
18      xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
19      xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
20      xmlns:Ontology1280888215552="http://www.semanticweb.org/ontologies
            /2010/7/Ontology1280888215552.owl#">
21
22      <owl:Ontology
23          rdf:about="http://www.semanticweb.org/ontologies/2010/7/
                Ontology1280888215552.owl">
24          <owl:versionInfo>Versao 0.1</owl:versionInfo>
25          <dc:language>Portugues Brasileiro</dc:language>
26          <rdfs:comment>Esta ontologia representa o Modelo de Titulo de
                Entidade, para a proxima geracao de Internet. Esta ontologia e
                utilizada para a comunicacao entre as camadas deste modelo,
                entre as entidades e entre as entidades e o DTS (DomainS Title
                Service).</rdfs:comment>
27          <dc:title>Modelo de Titulo de Entidade</dc:title>
28          <dc:contributor>Grupos FINLAN e MEHAR.</dc:contributor>
29          <dc:date>02-Novembro-2010</dc:date>
30          <dc:creator> Msc. Joao Henrique de Souza Pereira e Ph.D. Sergio
                Takeo Kofuji</dc:creator>
31      </owl:Ontology>
32
33  <!--
34  ///////////////////////////////////////////////////////////////////////
35  //
36  // Annotation properties
37  //
38  ///////////////////////////////////////////////////////////////////////
39  -->
40  <owl:AnnotationProperty rdf:about="&owl;versionInfo"/>
41  <owl:AnnotationProperty rdf:about="&dc;creator"/>
42  <owl:AnnotationProperty rdf:about="&dc;contributor"/>
43  <owl:AnnotationProperty rdf:about="&dc;language"/>
44  <owl:AnnotationProperty rdf:about="&dc;date"/>
45  <owl:AnnotationProperty rdf:about="&rdfs;comment"/>
46  <owl:AnnotationProperty rdf:about="&dc;title"/>
47
48  <!--
49  ///////////////////////////////////////////////////////////////////////
```

```
50 //
51 // Object Properties
52 //
53 /////////////////////////////////////////////////////////////////////////
54 -->
55 <!--http://www.semanticweb.org/ontologies/2010/7/Ontology1280888215552.
      owl#tem_Necessidade-->
56 <owl:ObjectProperty rdf:about="&TitleModel;tem_Necessidade">
57     <rdfs:domain rdf:resource="&TitleModel;Entidade"/>
58     <rdfs:range rdf:resource="&TitleModel;Necessidade"/>
59     <rdfs:subPropertyOf rdf:resource="&owl;topObjectProperty"/>
60 </owl:ObjectProperty>
61 <!--http://www.semanticweb.org/ontologies/2010/7/Ontology1280888215552.
      owl#tem_Titulo-->
62 <owl:ObjectProperty rdf:about="&TitleModel;tem_Titulo">
63     <rdfs:domain rdf:resource="&TitleModel;Entidade"/>
64     <rdfs:range rdf:resource="&TitleModel;Titulo"/>
65     <rdfs:subPropertyOf rdf:resource="&owl;topObjectProperty"/>
66 </owl:ObjectProperty>
67 <!--http://www.w3.org/2002/07/owl#topObjectProperty-->
68 <owl:ObjectProperty rdf:about="&owl;topObjectProperty"/>
69
70 <!--
71 /////////////////////////////////////////////////////////////////////////
72 //
73 // Data properties
74 //
75 /////////////////////////////////////////////////////////////////////////
76 -->
77 <!--http://www.w3.org/2002/07/owl#topDataProperty-->
78 <owl:DatatypeProperty rdf:about="&owl;topDataProperty"/>
79
80 <!--
81 /////////////////////////////////////////////////////////////////////////
82 //
83 // Classes
84 //
85 /////////////////////////////////////////////////////////////////////////
86 -->
```

```
87 <!--http://www.semanticweb.org/ontologies/2010/7/Ontology1280888215552.
      owl#AAA-->
88 <owl:Class rdf:about="&TitleModel;AAA">
89     <rdfs:subClassOf rdf:resource="&TitleModel;Seguran&#231;a"/>
90 </owl:Class>
91 <!--http://www.semanticweb.org/ontologies/2010/7/Ontology1280888215552.
      owl#Acesso_Contabiliza&#231;&#227;o-->
92 <owl:Class rdf:about="&TitleModel;Acesso_Contabiliza&#231;&#227;o">
93     <rdfs:subClassOf rdf:resource="&TitleModel;AAA"/>
94 </owl:Class>
95 <!--http://www.semanticweb.org/ontologies/2010/7/Ontology1280888215552.
      owl#Aplicacao-->
96 <owl:Class rdf:about="&TitleModel;Aplicacao">
97     <rdfs:subClassOf rdf:resource="&TitleModel;Entidade"/>
98 </owl:Class>
99 <!--http://www.semanticweb.org/ontologies/2010/7/Ontology1280888215552.
      owl#Assinatura_Digital-->
100 <owl:Class rdf:about="&TitleModel;Assinatura_Digital">
101     <rdfs:subClassOf rdf:resource="&TitleModel;Titulo"/>
102 </owl:Class>
103 <!--http://www.semanticweb.org/ontologies/2010/7/Ontology1280888215552.
      owl#Audio-->
104 <owl:Class rdf:about="&TitleModel;Audio">
105     <rdfs:subClassOf rdf:resource="&TitleModel;Tipo"/>
106 </owl:Class>
107 <!--http://www.semanticweb.org/ontologies/2010/7/Ontology1280888215552.
      owl#Autentica&#231;&#227;o-->
108 <owl:Class rdf:about="&TitleModel;Autentica&#231;&#227;o">
109     <rdfs:subClassOf rdf:resource="&TitleModel;AAA"/>
110 </owl:Class>
111 <!--http://www.semanticweb.org/ontologies/2010/7/Ontology1280888215552.
      owl#Autoriza&#231;&#227;o-->
112 <owl:Class rdf:about="&TitleModel;Autoriza&#231;&#227;o">
113     <rdfs:subClassOf rdf:resource="&TitleModel;AAA"/>
114 </owl:Class>
115 <!--http://www.semanticweb.org/ontologies/2010/7/Ontology1280888215552.
      owl#Baixa_Latencia-->
116 <owl:Class rdf:about="&TitleModel;Baixa_Latencia">
117     <rdfs:subClassOf rdf:resource="&TitleModel;Necessidade"/>
```

```
118 </owl:Class>
119 <!--http://www.semanticweb.org/ontologies/2010/7/Ontology1280888215552.
       owl#Baixo_Jitter-->
120 <owl:Class rdf:about="&TitleModel;Baixo_Jitter">
121     <rdfs:subClassOf rdf:resource="&TitleModel;Necessidade"/>
122 </owl:Class>
123 <!--http://www.semanticweb.org/ontologies/2010/7/Ontology1280888215552.
       owl#Banda_Throughput-->
124 <owl:Class rdf:about="&TitleModel;Banda_Throughput">
125     <rdfs:subClassOf rdf:resource="&TitleModel;Necessidade"/>
126 </owl:Class>
127 <!--http://www.semanticweb.org/ontologies/2010/7/Ontology1280888215552.
       owl#Bidirecional-->
128 <owl:Class rdf:about="&TitleModel;Bidirecional">
129     <rdfs:subClassOf rdf:resource="&TitleModel;Direcao"/>
130 </owl:Class>
131 <!--http://www.semanticweb.org/ontologies/2010/7/Ontology1280888215552.
       owl#Broadcast-Multiponto-->
132 <owl:Class rdf:about="&TitleModel;Broadcast-Multiponto">
133     <rdfs:subClassOf rdf:resource="&TitleModel;Comunicacao"/>
134 </owl:Class>
135 <!--http://www.semanticweb.org/ontologies/2010/7/Ontology1280888215552.
       owl#Camada-->
136 <owl:Class rdf:about="&TitleModel;Camada">
137     <rdfs:subClassOf rdf:resource="&owl;Thing"/>
138 </owl:Class>
139 <!--http://www.semanticweb.org/ontologies/2010/7/Ontology1280888215552.
       owl#Camada_Fisica-->
140 <owl:Class rdf:about="&TitleModel;Camada_Fisica">
141     <rdfs:subClassOf rdf:resource="&TitleModel;Camada"/>
142 </owl:Class>
143 <!--http://www.semanticweb.org/ontologies/2010/7/Ontology1280888215552.
       owl#Camada_de_Enlace-->
144 <owl:Class rdf:about="&TitleModel;Camada_de_Enlace">
145     <rdfs:subClassOf rdf:resource="&TitleModel;Camada"/>
146 </owl:Class>
147 <!--http://www.semanticweb.org/ontologies/2010/7/Ontology1280888215552.
       owl#Camada_de_Entidade-->
148 <owl:Class rdf:about="&TitleModel;Camada_de_Entidade">
```

```
149     <rdfs:subClassOf rdf:resource="&TitleModel;Camada"/>
150 </owl:Class>
151 <!--http://www.semanticweb.org/ontologies/2010/7/Ontology1280888215552.
       owl#Camada_de_Servico-->
152 <owl:Class rdf:about="&TitleModel;Camada_de_Servico">
153     <rdfs:subClassOf rdf:resource="&TitleModel;Camada"/>
154 </owl:Class>
155 <!--http://www.semanticweb.org/ontologies/2010/7/Ontology1280888215552.
       owl#Clareza-->
156 <owl:Class rdf:about="&TitleModel;Clareza">
157     <rdfs:subClassOf rdf:resource="&TitleModel;Necessidade"/>
158 </owl:Class>
159 <!--http://www.semanticweb.org/ontologies/2010/7/Ontology1280888215552.
       owl#Comunicacao-->
160 <owl:Class rdf:about="&TitleModel;Comunicacao">
161     <rdfs:subClassOf rdf:resource="&TitleModel;Necessidade"/>
162 </owl:Class>
163 <!--http://www.semanticweb.org/ontologies/2010/7/Ontology1280888215552.
       owl#Conex&#227;o-->
164 <owl:Class rdf:about="&TitleModel;Conex&#227;o">
165     <rdfs:subClassOf rdf:resource="&TitleModel;Necessidade"/>
166 </owl:Class>
167 <!--http://www.semanticweb.org/ontologies/2010/7/Ontology1280888215552.
       owl#Confiabilidade-->
168 <owl:Class rdf:about="&TitleModel;Confiabilidade">
169     <rdfs:subClassOf rdf:resource="&TitleModel;Seguran&#231;a"/>
170 </owl:Class>
171 <!--http://www.semanticweb.org/ontologies/2010/7/Ontology1280888215552.
       owl#Confirmada-->
172 <owl:Class rdf:about="&TitleModel;Confirmada">
173     <rdfs:subClassOf rdf:resource="&TitleModel;Garantia_de_Entrega"/>
174 </owl:Class>
175 <!--http://www.semanticweb.org/ontologies/2010/7/Ontology1280888215552.
       owl#Conteudo-->
176 <owl:Class rdf:about="&TitleModel;Conteudo">
177     <rdfs:subClassOf rdf:resource="&TitleModel;Entidade"/>
178 </owl:Class>
179 <!--http://www.semanticweb.org/ontologies/2010/7/Ontology1280888215552.
       owl#Controle_Tamanho_Pacote-->
```

```
180 <owl:Class rdf:about="&TitleModel;Controle_Tamanho_Pacote">
181     <rdfs:subClassOf rdf:resource="&TitleModel;Necessidade"/>
182 </owl:Class>
183 <!--http://www.semanticweb.org/ontologies/2010/7/Ontology1280888215552.
    owl#Criptografia-->
184 <owl:Class rdf:about="&TitleModel;Criptografia">
185     <rdfs:subClassOf rdf:resource="&TitleModel;Seguran&#231;a"/>
186 </owl:Class>
187 <!--http://www.semanticweb.org/ontologies/2010/7/Ontology1280888215552.
    owl#DNA-->
188 <owl:Class rdf:about="&TitleModel;DNA">
189     <rdfs:subClassOf rdf:resource="&TitleModel;Titulo"/>
190 </owl:Class>
191 <!--http://www.semanticweb.org/ontologies/2010/7/Ontology1280888215552.
    owl#DTS-->
192 <owl:Class rdf:about="&TitleModel;DTS"/>
193 <!--http://www.semanticweb.org/ontologies/2010/7/Ontology1280888215552.
    owl#Dados-->
194 <owl:Class rdf:about="&TitleModel;Dados">
195     <rdfs:subClassOf rdf:resource="&TitleModel;Tipo"/>
196 </owl:Class>
197 <!--http://www.semanticweb.org/ontologies/2010/7/Ontology1280888215552.
    owl#Direcao-->
198 <owl:Class rdf:about="&TitleModel;Direcao">
199     <rdfs:subClassOf rdf:resource="&TitleModel;Comunicacao"/>
200 </owl:Class>
201 <!--http://www.semanticweb.org/ontologies/2010/7/Ontology1280888215552.
    owl#Duplex-->
202 <owl:Class rdf:about="&TitleModel;Duplex">
203     <rdfs:subClassOf rdf:resource="&TitleModel;Bidirecional"/>
204 </owl:Class>
205 <!--http://www.semanticweb.org/ontologies/2010/7/Ontology1280888215552.
    owl#Email-->
206 <owl:Class rdf:about="&TitleModel;Email">
207     <rdfs:subClassOf rdf:resource="&TitleModel;Titulo"/>
208 </owl:Class>
209 <!--http://www.semanticweb.org/ontologies/2010/7/Ontology1280888215552.
    owl#Enderecamento-->
210 <owl:Class rdf:about="&TitleModel;Enderecamento">
```

```
211     <rdfs:subClassOf rdf:resource="&TitleModel;Necessidade"/>
212 </owl:Class>
213 <!--http://www.semanticweb.org/ontologies/2010/7/Ontology1280888215552.
       owl#Enderecamento_Estacao-->
214 <owl:Class rdf:about="&TitleModel;Enderecamento_Estacao">
215     <rdfs:subClassOf rdf:resource="&TitleModel;Enderecamento"/>
216 </owl:Class>
217 <!--http://www.semanticweb.org/ontologies/2010/7/Ontology1280888215552.
       owl#Enderecamento_Processo-->
218 <owl:Class rdf:about="&TitleModel;Enderecamento_Processo">
219     <rdfs:subClassOf rdf:resource="&TitleModel;Enderecamento"/>
220 </owl:Class>
221 <!--http://www.semanticweb.org/ontologies/2010/7/Ontology1280888215552.
       owl#Entidade-->
222 <owl:Class rdf:about="&TitleModel;Entidade">
223     <rdfs:subClassOf rdf:resource="&owl;Thing"/>
224 </owl:Class>
225 <!--http://www.semanticweb.org/ontologies/2010/7/Ontology1280888215552.
       owl#Entrega_Condicionada-Pipe_Line-->
226 <owl:Class rdf:about="&TitleModel;Entrega_Condicionada-Pipe_Line">
227     <rdfs:subClassOf rdf:resource="&TitleModel;Necessidade"/>
228 </owl:Class>
229 <!--http://www.semanticweb.org/ontologies/2010/7/Ontology1280888215552.
       owl#Entrega_de_Dados_Ordenados-->
230 <owl:Class rdf:about="&TitleModel;Entrega_de_Dados_Ordenados">
231     <rdfs:subClassOf rdf:resource="&TitleModel;Necessidade"/>
232 </owl:Class>
233 <!--http://www.semanticweb.org/ontologies/2010/7/Ontology1280888215552.
       owl#Estacao-->
234 <owl:Class rdf:about="&TitleModel;Estacao">
235     <rdfs:subClassOf rdf:resource="&TitleModel;Entidade"/>
236 </owl:Class>
237 <!--http://www.semanticweb.org/ontologies/2010/7/Ontology1280888215552.
       owl#Garantia_de_Entrega-->
238 <owl:Class rdf:about="&TitleModel;Garantia_de_Entrega">
239     <rdfs:subClassOf rdf:resource="&TitleModel;Necessidade"/>
240 </owl:Class>
241 <!--http://www.semanticweb.org/ontologies/2010/7/Ontology1280888215552.
       owl#Gerenciamento-->
```

```
242 <owl:Class rdf:about="&TitleModel;Gerenciamento">
243     <rdfs:subClassOf rdf:resource="&TitleModel;Necessidade"/>
244 </owl:Class>
245 <!--http://www.semanticweb.org/ontologies/2010/7/Ontology1280888215552.
    owl#Grava&#231;&#227;o-->
246 <owl:Class rdf:about="&TitleModel;Grava&#231;&#227;o">
247     <rdfs:subClassOf rdf:resource="&TitleModel;Necessidade"/>
248 </owl:Class>
249 <!--http://www.semanticweb.org/ontologies/2010/7/Ontology1280888215552.
    owl#Half_Duplex-->
250 <owl:Class rdf:about="&TitleModel;Half_Duplex">
251     <rdfs:subClassOf rdf:resource="&TitleModel;Bidirecional"/>
252 </owl:Class>
253 <!--http://www.semanticweb.org/ontologies/2010/7/Ontology1280888215552.
    owl#Hash-->
254 <owl:Class rdf:about="&TitleModel;Hash">
255     <rdfs:subClassOf rdf:resource="&TitleModel;Titulo"/>
256 </owl:Class>
257 <!--http://www.semanticweb.org/ontologies/2010/7/Ontology1280888215552.
    owl#Hierarquico-->
258 <owl:Class rdf:about="&TitleModel;Hierarquico">
259     <rdfs:subClassOf rdf:resource="&TitleModel;Enderecamento"/>
260 </owl:Class>
261 <!--http://www.semanticweb.org/ontologies/2010/7/Ontology1280888215552.
    owl#Horizontal-->
262 <owl:Class rdf:about="&TitleModel;Horizontal">
263     <rdfs:subClassOf rdf:resource="&TitleModel;Enderecamento"/>
264 </owl:Class>
265 <!--http://www.semanticweb.org/ontologies/2010/7/Ontology1280888215552.
    owl#Imagens-->
266 <owl:Class rdf:about="&TitleModel;Imagens">
267     <rdfs:subClassOf rdf:resource="&TitleModel;Tipo"/>
268 </owl:Class>
269 <!--http://www.semanticweb.org/ontologies/2010/7/Ontology1280888215552.
    owl#Jitter-->
270 <owl:Class rdf:about="&TitleModel;Jitter">
271     <rdfs:subClassOf rdf:resource="&TitleModel;QoS"/>
272 </owl:Class>
273 <!--http://www.semanticweb.org/ontologies/2010/7/Ontology1280888215552.
```

```
    owl#Latencia-->
<owl:Class rdf:about="&TitleModel;Latencia">
    <rdfs:subClassOf rdf:resource="&TitleModel;QoS"/>
</owl:Class>
<!--http://www.semanticweb.org/ontologies/2010/7/Ontology1280888215552.
    owl#Mensagem_Instantanea-->
<owl:Class rdf:about="&TitleModel;Mensagem_Instantanea">
    <rdfs:subClassOf rdf:resource="&TitleModel;Tempo_Real"/>
</owl:Class>
<!--http://www.semanticweb.org/ontologies/2010/7/Ontology1280888215552.
    owl#Mobilidade-->
<owl:Class rdf:about="&TitleModel;Mobilidade">
    <rdfs:subClassOf rdf:resource="&TitleModel;Necessidade"/>
</owl:Class>
<!--http://www.semanticweb.org/ontologies/2010/7/Ontology1280888215552.
    owl#Monitora&#231;&#227;o_da_Entrega_de_Dados-->
<owl:Class rdf:about="&TitleModel;Monitora&#231;&#227;
    o_da_Entrega_de_Dados">
    <rdfs:subClassOf rdf:resource="&TitleModel;Necessidade"/>
</owl:Class>
<!--http://www.semanticweb.org/ontologies/2010/7/Ontology1280888215552.
    owl#Multicast-Ponto_Multiponto-->
<owl:Class rdf:about="&TitleModel;Multicast-Ponto_Multiponto">
    <rdfs:subClassOf rdf:resource="&TitleModel;Comunicacao"/>
</owl:Class>
<!--http://www.semanticweb.org/ontologies/2010/7/Ontology1280888215552.
    owl#Nao_Confirmada-->
<owl:Class rdf:about="&TitleModel;Nao_Confirmada">
    <rdfs:subClassOf rdf:resource="&TitleModel;Garantia_de_Entrega"/>
</owl:Class>
<!--http://www.semanticweb.org/ontologies/2010/7/Ontology1280888215552.
    owl#Nao_Orientado-->
<owl:Class rdf:about="&TitleModel;Nao_Orientado">
    <rdfs:subClassOf rdf:resource="&TitleModel;Conex&#227;o"/>
</owl:Class>
<!--http://www.semanticweb.org/ontologies/2010/7/Ontology1280888215552.
    owl#Necessidade-->
<owl:Class rdf:about="&TitleModel;Necessidade">
    <rdfs:subClassOf rdf:resource="&owl;Thing"/>
```

```
304 </owl:Class>
305 <!--http://www.semanticweb.org/ontologies/2010/7/Ontology1280888215552.
       owl#Nuvem_Computacional-->
306 <owl:Class rdf:about="&TitleModel;Nuvem_Computacional">
307     <rdfs:subClassOf rdf:resource="&TitleModel;Entidade"/>
308 </owl:Class>
309 <!--http://www.semanticweb.org/ontologies/2010/7/Ontology1280888215552.
       owl#Ordenacao-->
310 <owl:Class rdf:about="&TitleModel;Ordenacao">
311     <rdfs:subClassOf rdf:resource="&TitleModel;Necessidade"/>
312 </owl:Class>
313 <!--http://www.semanticweb.org/ontologies/2010/7/Ontology1280888215552.
       owl#Orientado-->
314 <owl:Class rdf:about="&TitleModel;Orientado">
315     <rdfs:subClassOf rdf:resource="&TitleModel;Conex&#227;o"/>
316 </owl:Class>
317 <!--http://www.semanticweb.org/ontologies/2010/7/Ontology1280888215552.
       owl#Privacidade-->
318 <owl:Class rdf:about="&TitleModel;Privacidade">
319     <rdfs:subClassOf rdf:resource="&TitleModel;Seguran&#231;a"/>
320 </owl:Class>
321 <!--http://www.semanticweb.org/ontologies/2010/7/Ontology1280888215552.
       owl#Prote&#231;&#227;o-->
322 <owl:Class rdf:about="&TitleModel;Prote&#231;&#227;o">
323     <rdfs:subClassOf rdf:resource="&TitleModel;Necessidade"/>
324 </owl:Class>
325 <!--http://www.semanticweb.org/ontologies/2010/7/Ontology1280888215552.
       owl#QoS-->
326 <owl:Class rdf:about="&TitleModel;QoS">
327     <rdfs:subClassOf rdf:resource="&TitleModel;Necessidade"/>
328 </owl:Class>
329 <!--http://www.semanticweb.org/ontologies/2010/7/Ontology1280888215552.
       owl#Rede_de_Sensor-->
330 <owl:Class rdf:about="&TitleModel;Rede_de_Sensor">
331     <rdfs:subClassOf rdf:resource="&TitleModel;Entidade"/>
332 </owl:Class>
333 <!--http://www.semanticweb.org/ontologies/2010/7/Ontology1280888215552.
       owl#Seguran&#231;a-->
334 <owl:Class rdf:about="&TitleModel;Seguran&#231;a">
```

```
335     <rdfs:subClassOf rdf:resource="&TitleModel;Necessidade"/>
336 </owl:Class>
337 <!--http://www.semanticweb.org/ontologies/2010/7/Ontology1280888215552.
       owl#Stream_de_Audio-->
338 <owl:Class rdf:about="&TitleModel;Stream_de_Audio">
339     <rdfs:subClassOf rdf:resource="&TitleModel;Tempo_Real"/>
340 </owl:Class>
341 <!--http://www.semanticweb.org/ontologies/2010/7/Ontology1280888215552.
       owl#Stream_de_Video-->
342 <owl:Class rdf:about="&TitleModel;Stream_de_Video">
343     <rdfs:subClassOf rdf:resource="&TitleModel;Tempo_Real"/>
344 </owl:Class>
345 <!--http://www.semanticweb.org/ontologies/2010/7/Ontology1280888215552.
       owl#Tempo_Real-->
346 <owl:Class rdf:about="&TitleModel;Tempo_Real">
347     <rdfs:subClassOf rdf:resource="&TitleModel;Necessidade"/>
348 </owl:Class>
349 <!--http://www.semanticweb.org/ontologies/2010/7/Ontology1280888215552.
       owl#Texto-->
350 <owl:Class rdf:about="&TitleModel;Texto">
351     <rdfs:subClassOf rdf:resource="&TitleModel;Tipo"/>
352 </owl:Class>
353 <!--http://www.semanticweb.org/ontologies/2010/7/Ontology1280888215552.
       owl#Tipo-->
354 <owl:Class rdf:about="&TitleModel;Tipo">
355     <rdfs:subClassOf rdf:resource="&TitleModel;Necessidade"/>
356 </owl:Class>
357 <!--http://www.semanticweb.org/ontologies/2010/7/Ontology1280888215552.
       owl#Titulo-->
358 <owl:Class rdf:about="&TitleModel;Titulo">
359     <rdfs:subClassOf rdf:resource="&owl;Thing"/>
360 </owl:Class>
361 <!--http://www.semanticweb.org/ontologies/2010/7/Ontology1280888215552.
       owl#Transfer&#234;ncia_de_Dados-->
362 <owl:Class rdf:about="&TitleModel;Transfer&#234;ncia_de_Dados">
363     <rdfs:subClassOf rdf:resource="&TitleModel;Necessidade"/>
364 </owl:Class>
365 <!--http://www.semanticweb.org/ontologies/2010/7/Ontology1280888215552.
       owl#Unicast-Ponto_a_Ponto-->
```

```
366 <owl:Class rdf:about="&TitleModel;Unicast-Ponto_a_Ponto">
367     <rdfs:subClassOf rdf:resource="&TitleModel;Comunicacao"/>
368 </owl:Class>
369 <!--http://www.semanticweb.org/ontologies/2010/7/Ontology1280888215552.
    owl#Unidirecional_Simplex-->
370 <owl:Class rdf:about="&TitleModel;Unidirecional_Simplex">
371     <rdfs:subClassOf rdf:resource="&TitleModel;Direcao"/>
372 </owl:Class>
373 <!--http://www.semanticweb.org/ontologies/2010/7/Ontology1280888215552.
    owl#Usuario-->
374 <owl:Class rdf:about="&TitleModel;Usuario">
375     <rdfs:subClassOf rdf:resource="&TitleModel;Entidade"/>
376 </owl:Class>
377 <!--http://www.semanticweb.org/ontologies/2010/7/Ontology1280888215552.
    owl#Velocidade-->
378 <owl:Class rdf:about="&TitleModel;Velocidade">
379     <rdfs:subClassOf rdf:resource="&TitleModel;Necessidade"/>
380 </owl:Class>
381 <!--http://www.semanticweb.org/ontologies/2010/7/Ontology1280888215552.
    owl#V&#237;deo-->
382 <owl:Class rdf:about="&TitleModel;V&#237;deo">
383     <rdfs:subClassOf rdf:resource="&TitleModel;Tipo"/>
384 </owl:Class>
385 <!--http://www.w3.org/2002/07/owl#Thing-->
386 <owl:Class rdf:about="&owl;Thing"/>
387
388 <!--
389 ////////////////////////////////////////////////////////////////////////
390 //
391 // Individuals
392 //
393 ////////////////////////////////////////////////////////////////////////
394 -->
395 <!--http://www.semanticweb.org/ontologies/2010/7/Ontology1280888215552.
    owl#Programacao_Distribuida-->
396 <owl:Thing rdf:about="&TitleModel;Programacao_Distribuida">
397     <rdf:type rdf:resource="&TitleModel;Aplicacao"/>
398     <rdf:type rdf:resource="&owl;NamedIndividual"/>
399     <rdfs:comment>Comunicacao de necessidades, entre a camada de entidade
```

```
        e a camada de
400         servico, para ambientes de programacao distribuida.
401     </rdfs:comment>
402 </owl:Thing>
403 </rdf:RDF>
404 <!-- Generated by the OWL API (version 3.0.0.1451) http://owlapi.
        sourceforge.net-->
```