Mecanismos de autenticação e controle de acesso para uma arquitetura de Internet do Futuro

Pedro Henrique Aparecido Damaso de Melo

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

Uberlândia
2017

Pedro Henrique Aparecido Damaso de Melo

Mecanismos de autenticação e controle de acesso para uma arquitetura de Internet do Futuro

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

Área de concentração: Ciência da Computação

Orientador: Flávio de Oliveira Silva
Coorientador: Pedro Frosi Rosa

Uberlândia
2017

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

M528m
2017

Melo, Pedro Henrique Aparecido Damaso de, 1991-
Mecanismos de autenticação e controle de acesso para uma arquitetura de Internet do Futuro / Pedro Henrique Aparecido Damaso de Melo. - 2017.
102 f. : il.

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

1. Computação - Teses. 2. Computadores - Controle de acesso - Teses. 3. Programação para internet - Teses. 4. Internet - Sistemas de segurança - Teses. I. Silva, Flávio de Oliveira. II. Rosa, Pedro Frosi. III. Universidade Federal de Uberlândia. Programa de Pós-Graduação em Ciência da Computação. IV. 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 **"Mecanismos de autenticação e controle de acesso para uma arquitetura de Internet do Futuro"** por **Pedro Henrique Aparecido Damaso de Melo** como parte dos requisitos exigidos para a obtenção do título de **Mestre em Ciência da Computação**.

Uberlândia, 20 de fevereiro de 2017

Orientador: _____
Prof. Dr. Flávio de Oliveira Silva
Universidade Federal de Uberlândia

Coorientador: _____
Prof. Dr. Pedro Frosi Rosa
Universidade Federal de Uberlândia

Banca Examinadora:

_____
Prof. Dr. Bruno Bogaz Zarpelão
Universidade Estadual de Londrina

_____
Prof. Dr. Rodrigo Sanches Miani
Universidade Federal de Uberlândia

Dedico este trabalho aos meus pais Pedro e Lusia,
pelo apoio incondicional e constante incentivo ao longo da minha vida.

Agradecimentos

Agradeço em primeiro lugar a Deus, que iluminou o meu caminho durante esta caminhada e por ter me dado força e coragem para desbravar esse período de muito estudo e aprendizado, conseguindo assim concretizar mais um sonho.

Aos meus pais, um grande e sincero obrigado, sempre me apoiaram com muito carinho. Eu devo tudo que sou a vocês, e se sinto orgulho de mim e do lugar onde cheguei, é porque sei que vocês vieram segurando a minha mão.

A Ana Caroline, que sempre esteve ao meu lado, me fazendo acreditar que posso mais que imagino. Devido a seu companheirismo, amizade, paciência, compreensão, apoio e alegria, este trabalho pôde ser concretizado.

Aos orientadores Flávio de Oliveira Silva e Pedro Frosi Rosa, pelo convívio, apoio, compreensão, amizade e palavras regadas de sabedoria. Sempre me apoiando e incentivando durante todo o processo.

Aos professores da Faculdade de Computação, pelo apoio e ensinamentos, que foram muito além dos conteúdos programáticos, com os quais foi possível adquirir aprendizados importantes para a vida toda.

Aos amigos, que me acompanharam durante essa jornada e sempre me deram apoio, seja com palavras, tecnicamente ou por uma troca de experiência. Foi bom poder contar com vocês!

Por fim agradeço a CAPES pelo apoio através do Programa de Apoio ao Ensino e à Pesquisa Científica e Tecnológica em Defesa Nacional (Pró-Defesa).

"Que os vossos esforços desafiem as impossibilidades, lembrai-vos de que as grandes coisas do homem foram conquistadas do que parecia impossível."

(Charles Chaplin)

Resumo

Mesmo com evoluções, a Internet atual não consegue tratar adequadamente requisitos como *multihoming*, *Quality of Service* (QoS), mobilidade, *multicast* e segurança. Vários grupos de pesquisa ao redor mundo estão envolvidos em criar, de forma experimental e incremental, a próxima geração da arquitetura da Internet.

Atualmente, o conhecimento e a informação são fatores importantes para qualquer pessoa, organização ou nação. Pensando nisso, a segurança é um pré-requisito para todo e qualquer sistema de computação, mas quando a Internet foi projetada, a segurança não era uma necessidade da época, provocando um problema crônico nas últimas décadas.

Sempre que surgem novas vulnerabilidades em um sistema computacional, um novo mecanismo é criado para combater essa ameaça, sendo assim, o mecanismo é adicionado ao projeto da Internet como uma sobreposição, em vez da arquitetura fornecer a segurança de forma intrínseca.

No que tange à essas arquiteturas, o Brasil possui algumas iniciativas e uma delas é a Entity Title Architecture (ETArch). Ela possui uma visão conceitual muito próxima da abstração proposta pelas Redes Definidas por Software e portanto, desde o seu primeiro protótipo utiliza o protocolo OpenFlow para materializar essa visão. Desde a sua criação, pesquisadores de várias universidades vêm trabalhando para incorporar à ETArch, de forma incremental, soluções que visam atender os requisitos de Internet do Futuro.

Apesar da segurança ser um requisito fundamental para implementações em arquiteturas de Internet do Futuro, na ETArch tal requisito ainda não foi projetado. Deste modo, as principais contribuições deste trabalho são elaborar e implementar dois mecanismos de segurança: um para autenticação e outro para o controle de acesso.

A implementação dos mecanismos demonstraram-se viáveis com um acréscimo médio relativamente pequeno em termos de tempo, se considerar os benefícios adquiridos pelos mecanismos de autenticação e controle de acesso incorporados à ETArch.

Palavras-chave: Internet do Futuro. Autenticação. Controle de Acesso.

Abstract

Even with evolutions, the current Internet can not properly handle requirements such as multihoming, Quality of Service, mobility, multicasting and security. Several research groups around the world are involved in experimentally and incrementally creating the next generation of Internet architecture.

Currently, knowledge and information are the factors of extreme importance for any person, company or nation. Therefore, the information security is a prerequisite for any information system. However, when the Internet was designed and security was not a necessity at the moment, this became a chronic problem in the last decades.

Whenever new vulnerabilities emerge on the network, a new mechanism is created to combat this threat, so the mechanism is added to the design of the Internet as an overlay, rather than the architecture providing security intrinsically. In this way, including security aspects is a fundamental requirement for the Future Internet architecture.

With regard to these architectures, Brazil has some initiatives and one of them in an ETArch. It has a conceptual view very close to the definition of Software Defined Networks and therefore since its first prototype uses the OpenFlow protocol to materialize this vision. From its creation, researchers from several universities are working to incorporate in the ETArch, in an incremental way, solutions that meet the requirements of the Future Internet.

The mechanisms implementation proved viable with a reasonable average increase in time, considering the resources acquired by the mechanisms of authentication and access control incorporated into ETArch.

Keywords: Future Internet. Authentication. Access control.

Lista de ilustrações

Lista de tabelas

Lista de siglas

AP Access Point

API Application Programming Interface

ACL Access Control List

AICT Advanced International Conference on Telecommunications

DTS Domain Title Service

DTSA Domain Title Service Agent

ETArch Entity Title Architecture

ETCP Entity Title Control Protocol

EDOBRA ExtenDing Ofelia in BRAzil

FIRE Future Internet Research and Experimentation

FIBRE Future Internet Brazilian Environment for Experimentation

GENI Global Environment for Network Innovations

ICN Information-Centric Networking

IRATI Investigating RINA as an Alternative to TCP/IP

ISP Internet Service Provider

JCP Java Community Process

JSR Java Specification Request

MDTSA Master DTSA

MIH Media Independent Handover

NE Network Element

ONF Open Network Foundation

PoA Point of Attachment

PKI Public Key Infrastructure

QoE Quality of Experience

QoS Quality of Service

RINA Recursive InterNetwork Architecture

RA Resource Adapter

SDN Software Defined Networking

SP Service Provider

SBB Service Building Block

Sumário

Capítulo 1

Introdução

Os avanços tecnológicos tanto em hardware como em software, as novas tecnologias de acesso em banda larga e as redes de telecomunicações móveis propiciaram o surgimento de novos serviços e aplicações que seriam difíceis de se imaginar nos anos setenta.

A necessidade de mudança na arquitetura da Internet é discutida pelos cientistas desde o início da década de noventa (CLARK et al., 1991). A arquitetura inicial já passou por várias evoluções como por exemplo, endereçamento (EGEVANG; FRANCIS, 1994) (SRISURESH; EGEVANG, 2001) (GROUP; HINDEN, 1993) e roteamento (REKHTER; LI; HARES, 2006).

Mesmo com evoluções, a Internet atual não consegue tratar adequadamente requisitos como *multihoming*, Quality of Service (QoS), mobilidade, *multicast* e segurança (HANDLEY, 2006). Várias iniciativas de pesquisa (PAN; PAUL; JAIN, 2011) (GAVRAS et al., 2007) ao redor mundo estão empenhadas em fornecer uma solução para as novas exigências relativas à arquitetura de Internet. Duas abordagens (REXFORD; DOVROLIS, 2010) são possíveis: *clean slate* na qual os pesquisadores ignoram a arquitetura legada, ficando livres para propor uma nova arquitetura (FELDMANN, 2007) (ROBERTS, 2009) e evolucionária, onde a nova arquitetura deve partir da atual.

Segurança não era uma necessidade na arquitetura original da Internet e se tornou um problema grave nas últimas décadas. Sempre que surge uma nova vulnerabilidade nos dispositivos, um novo mecanismo é criado para combater essa ameaça, isto é, o mecanismo é adicionado ao projeto da Internet como uma sobreposição, em vez da arquitetura fornecer a segurança de forma intrínseca. Com base nesses requisitos que estão surgindo, vários grupos de pesquisa ao redor mundo estão envolvidos em criar a próxima geração da arquitetura da Internet (PAN; PAUL; JAIN, 2011).

Para auxiliar esses grupos de pesquisa e criar experimentos com essas novas arquiteturas de rede, várias infra estruturas estão sendo implantadas ao redor do mundo, como OFELIA (OFELIA, 2014) e várias outras dentro da iniciativa Future Internet Research and Experimentation (FIRE) (GAVRAS et al., 2007) na Europa, Global Environment for Network Innovations (GENI) (ELLIOTT, 2010) nos Estados Unidos e Future Internet

Brazilian Environment for Experimentation (FIBRE) (MACHADO et al., 2014) no Brasil, em um esforço conjunto com a Europa. Estas infra estruturas permitem a implantação e escala de experimentos que são necessários para enfrentar a proporção da Internet atual.

Em geral, os grupos de pesquisa utilizam uma abordagem experimental com a criação de protótipos dessas arquiteturas para experimentação. Como a gama de requisitos é ampla, os protótipos são aprimorados ao longo do ciclo de pesquisa. Um exemplo é a arquitetura Recursive InterNetwork Architecture (RINA), cujo protótipo para uso em cenários de produção foi criado no âmbito do projeto Investigating RINA as an Alternative to TCP/IP (IRATI) (VRIJDERS et al., 2014). Posteriormente o protótipo foi evoluído no âmbito do projeto PRISTINE (GRASA et al., 2016a). Nesse último foram acrescentadas capacidades de autenticação e controle de acesso na arquitetura RINA.

No que tange as arquiteturas de Internet do Futuro, o Brasil possui algumas iniciativas, uma delas é a ETArch (SILVA, 2013), cujo protótipo possui uma visão conceitual muito próxima da abstração proposta pelas Redes Definidas por Software e portanto, desde o seu primeiro protótipo utiliza o protocolo OpenFlow (MCKEOWN et al., 2008) para materializar essa visão, ou seja, possui um plano de controle independente do hardware/-software do equipamento de rede, manipulando o plano de dados. Na ETArch um dos protocolos utilizados no plano de controle é o Entity Title Control Protocol (ETCP) cuja responsabilidade é gerenciar ciclo de vida das entidades e dos canais de comunicação.

A partir da sua criação, pesquisadores de várias universidades vêm trabalhando para incorporar à ETArch de forma incremental, soluções e mecanismos que visam atender os requisitos atuais de uma nova arquitetura para a Internet como mobilidade (GUIMARAES et al., 2014), *multicast* (GONçALVES et al., 2014), QoS (CASTILLO et al., 2014) e roteamento (NETO et al., 2015).

Apesar de a segurança ser um requisito fundamental para implementações de arquiteturas de Internet do Futuro (HANDLEY, 2006), na arquitetura ETArch tal requisito ainda não foi abordado. Deste modo, as principais contribuições deste trabalho são elaborar e implementar dois mecanismos: um para autenticação que possui a responsabilidade de verificar a autenticidade dos usuários, sistemas ou processos e outro para o controle de acesso que é responsável por assegurar que apenas entidades autorizadas utilizem recursos protegidos.

A arquitetura ETArch revisa o conceito de endereçamento de *hosts* de origem e destino. Nela, as entidades que desejem se comunicar precisam se registrar e fazer parte de um canal de comunicação, denominado *Workspace*. O mecanismo de autenticação será empregado no momento em que uma entidade deseja realizar o registro e o controle de acesso quando uma entidade deseja fazer parte de um *Workspace*.

1.1 Motivação

Atualmente, o conhecimento e a informação são fatores de extrema importância para qualquer pessoa, empresa ou nação, com isso em mente a segurança da informação é um pré-requisito para todo e qualquer sistema de informação. A Internet não foi projetada pensando em segurança. Com o seu enorme sucesso nas últimas décadas, o tráfego continua crescendo, apesar de seu crescimento ter trazido grandes soluções para diversas áreas, ele também é responsável por vários problemas (JAIN, 2006).

Não há como ignorar as ameaças de segurança, que hoje estão cada vez maiores, a exemplo dos acessos não autorizados, espionagem, vulnerabilidades, entre outros (WHITMAN, 2003). As formas de ataque estão gradativamente mais sofisticadas e se adaptam às evoluções dos sistemas de defesa. Um dos principais motivos que colaboram para os problemas de segurança atuais é a ausência de segurança no projeto da arquitetura de rede (MOREIRA et al., 2009). Devido a tantos problemas associados à arquitetura atual, surge o estímulo para contribuir com uma rede de nova geração em prol da segurança.

Várias alterações (EGEVANG; FRANCIS, 1994) (SRISURESH; EGEVANG, 2001) (GROUP; HINDEN, 1993) (REKHTER; LI; HARES, 2006) foram feitas ao longo dos anos na arquitetura Internet, que se quer eram imaginadas na época de seu projeto, com o objetivo de suprir as necessidades que surgiram ao longo do tempo (HANDLEY, 2006). Mesmo com todas essas alterações, ainda existem requisitos que não foram atendidos, como por exemplo segurança (DING; YAN; DENG, 2016), mobilidade (GLADISCH; DAHER; TAVANGARIAN, 2014), *multicast*, entre outros.

Diante desse cenário surgiram várias propostas para arquiteturas de Internet do Futuro (DAY; MATTA; MATTAR, 2008) (SESKAR et al., 2011) (HAN et al., 2012), sendo segurança uma de suas maiores preocupações. A ETArch como uma nova arquitetura para a Internet do Futuro também foi concebida para suportar a segurança de forma nativa. Apesar disso, o desenvolvimento dos vários componentes da arquitetura ocorrem de forma incremental e até o momento aspectos de segurança ainda não foram desenvolvidos e acoplados à arquitetura.

Portanto, este trabalho objetiva projetar e implementar mecanismos de autenticação e controle de acesso para a arquitetura ETArch, sendo que, a autenticação busca definir diretrizes para que uma entidade se identifique à rede e o controle de acesso garanta que somente entidades autorizadas possam acessar determinado *Workspace*.

1.2 Objetivos e Desafios da Pesquisa

O objetivo geral deste trabalho é propor dois mecanismos de segurança para a arquitetura ETArch, sendo um para autenticação e outro para controle de acesso. Por exemplo, o mecanismo de autenticação terá que identificar qualquer entidade (pessoas, computado-

res, serviços, coisas, etc) que deseje se registrar na arquitetura e o controle de acesso terá que limitar o acesso das entidades aos *Workspaces*. Para se alcançar o objetivo citado, os seguintes objetivos específicos devem ser alcançados:

- ❏ Projetar dois mecanismos de segurança para a arquitetura ETArch, autenticação e controle de acesso;
- ❏ Estender o protocolo ETCP para tratar as novas primitivas de segurança;
- ❏ Realizar experimentos com os mecanismos de autenticação e controle de acesso propostos;
- ❏ Analisar os resultados obtidos com os novos mecanismos em cenários experimentais, e realizar uma avaliação comparativa da ETArch com e sem os mecanismos de segurança.

Os desafios do presente trabalho estão relacionados a elaborar e implementar mecanismos de autenticação e controle de acesso para uma nova arquitetura de rede de forma intrínseca, considerando a existência de peculiaridades não tratadas na arquitetura atual. Por conseguinte, é necessário repensar a segurança por meio de um modelo adequado para esta arquitetura.

1.3 Hipótese

Acredita-se que a autenticação e o controle de acesso possam ser inseridos inerente e transparente na arquitetura, para atender os requisitos de segurança das aplicações de forma efetiva. Desse modo, é possível a arquitetura garantir que a informação seja acessível apenas àqueles que possuam autorização.

1.4 Contribuições

O conhecimento adquirido sobre a arquitetura ETArch permitiu a contribuição com o grupo envolvido na implantação da ETArch em uma rede de uma operadora de telecomunicações (THEODORO et al., 2015).

Com a finalidade de levantar o cenário atual dos mecanismos de autenticação e controle de acesso existentes nas arquiteturas de Internet do Futuro, foi feita uma revisão global sobre as arquiteturas e seus mecanismos de segurança. O conhecimento adquirido nessa revisão foi de grande importância para projetar os mecanismos de segurança na arquitetura ETArch. Tal estudo pode ser utilizado como base para outros trabalhos relacionados a segurança para arquiteturas de Internet do Futuro.

Além disso, as demais contribuições do trabalho estão relacionadas à ETArch que até a conclusão deste projeto não dispunha de nenhum mecanismo de segurança. Os mecanismos desenvolvidos estão disponibilizados para todos os pesquisadores envolvidos com a ETArch e toda comunidade científica. Desta maneira, o presente trabalho amplia o nível de segurança da arquitetura, possibilitando a sua evolução.

1.5 Organização do Trabalho

Esta dissertação está organizada em seis capítulos. O Capítulo 2 apresenta os aspectos conceituais sobre autenticação e controle de acesso, explanando sua função nas redes. Ademais, são introduzidas as Redes Definidas por Software, que serão utilizadas neste trabalho. Por fim apresenta-se o estado da arte de pesquisas em Internet do Futuro, com foco especial nos mecanismos de autenticação e controle de acesso.

O Capítulo 3 apresenta os aspectos conceituais e principais implementações da arquitetura ETArch. Inicia-se com um detalhamento dos componentes principais e posteriormente é mostrado como cada componente foi desenvolvido.

O Capítulo 4 apresenta a proposta principal deste trabalho, ou seja, os mecanismos de autenticação e controle de acesso. Nesse capítulo, os aspectos conceituais e arquiteturais da ETArch são explorados minuciosamente, e é apresentada a situação atual de segurança da arquitetura.

O Capítulo 5 apresenta os experimentos realizados tanto na autenticação como no controle de acesso utilizando uma topologia, bem como a apresentação dos resultados e a avaliação dos mecanismos.

O Capítulo 6 faz uma conclusão, apresentando as contribuições do trabalho, os objetivos alcançados e os trabalhos futuros.

Capítulo 2

Fundamentação Teórica

Este capítulo se destina a apresentar a fundamentação teórica deste trabalho, versando sobre mecanismos de segurança em ambientes de rede, Redes Definidas por Software e arquiteturas para Internet do Futuro.

A autenticação é a base da segurança em sistemas distribuídos e, portanto, é essencial que esse mecanismo exista e funcione corretamente. Por esta razão, este trabalho tratou a autenticação como prioridade. O objetivo da autenticação é bastante claro, i.e., ela garante que o 'autenticado é quem diz ser'. As entidades autenticadas (pessoas, computadores, serviços, coisas etc) acreditam que estão se comunicando em um ambiente apropriado (BURROWS; ABADI; NEEDHAM, 1989).

O controle de acesso depende e coexiste com outros mecanismos de segurança, como por exemplo a autenticação. Controle de acesso tem o objetivo de garantir acessos às entidades autenticadas somente naqueles casos nas quais elas detenham autorização. Deste modo, é comum se ler que controle de acesso está preocupado em limitar a atividade de usuários legítimos (SANDHU; SAMARATI, 1994).

Com a implantação de uma nova arquitetura de Internet é necessário analisar se os quesitos atendidos na arquitetura atual são satisfeitos e se as necessidades relativas à segurança que advirem ao longo do tempo, não incorporadas no projeto da arquitetura atual, são resolvidas. Além disso, ela precisa ser aberta e extensível para acomodar aspectos de segurança que podem surgir futuramente.

Os aspectos conceituais sobre os mecanismos de autenticação e controle de acesso também serão apresentados. Por conseguinte, as novas propostas de modelos e arquiteturas para Internet do Futuro serão descritas, com foco nos trabalhos sobre autenticação e controle de acesso. Por fim será feita uma análise comparativa dos mecanismos de autenticação e controle de acesso encontrados em cada arquitetura.

2.1 Autenticação

Com a ubiquidade dos sistemas computacionais, surgem diversos problemas de segurança, que começam com roubo de senhas, interrupção de serviços e até roubo de identidade, no qual um atacante se faz passar por usuário legítimo para adquirir suas prerrogativas de acessos.

Dessa forma, surgiu a necessidade do mecanismo de Autenticação, ou seja, um mecanismo capaz de verificar a autenticidade da identidade tanto de usuários quanto de sistemas ou processos (BURROWS; ABADI; NEEDHAM, 1989). Autenticar é garantir que uma entidade é quem diz ser.

Os mecanismos de autenticação podem ser classificados em três tipos:

- Conhecimento: a autenticação é baseada em algo que a entidade conheça exclusivamente;
- Propriedade: a autenticação é baseada em algo que a entidade possua exclusivamente; ou,
- Característica: a autenticação é baseada em característica física, humana ou comportamental exclusiva.

Senha é um dos mecanismos mais utilizados, constituindo-se em um exemplo de autenticação baseada em Conhecimento. Esse mecanismo possui algumas limitações, pois as senhas podem ser adivinhadas, roubadas ou esquecidas (LAMPORT, 1981).

Cartão Inteligente (*smartcard*) (HWANG; LI, 2000), uma Chave ou *Token* são exemplos de autenticação baseada em Propriedade. As desvantagens dessa autenticação está na possibilidade de perda, esquecimento ou roubo do dispositivo.

Biometria (WAYMAN et al., 2005) é um exemplo de autenticação baseada em característica física. Nesse caso temos a garantia que a pessoa esteja presente no ponto de autenticação.

A autenticação de um usuário, sistema ou processo desempenha um papel importante para o mecanismo de controle de acesso, uma vez que suas operações são realizadas com base em uma entidade autêntica.

2.2 Controle de Acesso

Um sistema de gerenciamento de informações possui requisitos de extrema importância, por exemplo, proteger os dados e recursos contra a divulgação não autorizada e modificações impróprias. Portanto, tais requisitos exigem que cada acesso a um sistema e seus recursos sejam controlados e que ocorram somente acessos autorizados. Sendo assim, surge a necessidade de um mecanismo de segurança para tratar tais ameaças (SANDHU; SAMARATI, 1994).

O controle de acesso é o mecanismo responsável por assegurar que apenas entidades autorizadas utilizem recursos protegidos. O processo de autorização decide se uma entidade tem permissão para acessar determinado dado, informação ou recurso. Se caso tal mecanismo não existisse, as entidades não teriam nenhum tipo de privacidade, sendo assim, seria possível acessar qualquer tipo de informação. (SANDHU; SAMARATI, 1994).

É importante observar que a autenticação é fundamental para garantir a autenticidade da entidade que se identifica, mas os acessos a locais, recursos, informações, entre outros, passa por um segundo passo que é o controle de acesso. Nem todas as pessoas autenticadas têm permissões para todos os acessos.

Mesmo com a definição de políticas de controle de acesso, implementar tal mecanismo está longe de ser um processo trivial. Uma das grandes dificuldades está na interpretação de políticas de segurança do mundo real, muitas vezes complexas e ambíguas, e na sua tradução em regras bem definidas, claras e não ambíguas que possam ser aplicadas em um sistema distribuído. Muitas situações do mundo real têm políticas complexas, onde as decisões de acesso dependem da aplicação de diferentes regras que vêm, por exemplo, de leis, práticas e regulamentos organizacionais.

As políticas de controle de acesso podem ser agrupadas em três classes principais: DAC (*Discretionary Access Control*); MAC (*Mandatory Access Control*); e, RBAC (*Role-Based Access Control*) (SAMARATI; VIMERCATI, 2000).

As políticas DAC realiza um controle de acesso baseado na identidade da entidade e nas regras de acesso, que juntas indicam se a entidade possui ou não permissão para acessar determinado recurso. MAC controla o acesso com base em regulamentos obrigatórios determinados por uma autoridade central. RBAC controla o acesso dependendo das funções que os usuários possuem dentro do sistema e das regras que indicam quais acessos são permitidos aos usuários em determinadas funções (SAMARATI; VIMERCATI, 2000).

Tanto a autenticação quanto o controle de acesso foram adicionados como uma sobreposição na arquitetura atual, em vez de ser uma parte intrínseca da arquitetura. Sendo assim, esse é um dos objetivos a serem alcançados por novas arquiteturas de Internet do Futuro.

2.3 Redes Definidas por Software

Há uma tendência das novas arquiteturas de redes se basearem no conceito de Redes Definidas por Software ou Software Defined Networking (SDN) (KREUTZ et al., 2015) (HAKIRI et al., 2014). Tal conceito é definido como um novo paradigma que dá esperança para mudanças em relação às limitações de infra-estruturas de redes atuais. Um dos principais objetivos de SDN é criar uma rede ágil e flexível que suporte mudanças rápidas com base nas necessidades e demandas tanto de empresas quanto de aplicações.

SDN refere-se a uma arquitetura de rede que possui o plano de controle separado do

plano de dados. Basicamente, a rede é controlada por uma plataforma de software central, separada do *hardware* que é responsável por encaminhar os pacotes de dados da rede. Um controlador pode ser responsável por vários *switches*, ou seja, eles compartilham o mesmo plano de controle. Com isso, é possível alterar as regras de definição de fluxos dos *switches* a partir do plano de controle.

Atualmente, existem requisitos de comunicação na Internet como *multihoming*, segurança, mobilidade, *multicast* que são questões em aberto e vários grupos de pesquisa estão envolvidos no projeto da próxima geração de Internet que objetiva satisfazer esses requisitos. Neste âmbito, o *OpenFlow* que trata-se de uma materialização do SDN, é uma tecnologia que possibilita apoiar a pesquisa experimental com foco na Internet do futuro, a arquitetura *MobilityFirst*, por exemplo, possui um protótipo de suas funcionalidades usando o protocolo *OpenFlow* (KRISHNAMOORTHY, 2013).

2.3.1 OpenFlow

O OpenFlow (MCKEOWN et al., 2008) atrai uma atenção expressiva tanto da academia quanto da indústria. Um grupo de operadores de rede, prestadores de serviços e fornecedores criaram a Open Network Foundation (ONF) (FOUNDATION, 2017), uma organização para promover SDN e padronizar o protocolo OpenFlow.

A abstração do OpenFlow é composta pelo protocolo, switch e controlador. O *switch* OpenFlow é responsável pela materialização do SDN, ou seja, é encarregado pela separação do plano de controle e plano de dados. Nesse *switch* um dado tem seu controle feito por um elemento externo, o Controlador OpenFlow. Por fim a comunicação entre *switch* e o controlador é realizada através do protocolo OpenFlow (MCKEOWN, 2009).

O protocolo Openflow define algumas mensagens para realizar a comunicação entre o Controlador e os *switches*, por exemplo, PACKET_IN, PACKET_OUT e FLOW_MOD. A mensagem do tipo PACKET_IN é enviada do *switch* para o Controlador quando um pacote recebido não fez *match* com nenhuma entrada na tabela de fluxos do *switch*. A mensagem PACKET_OUT é usada quando o Controlador envia um pacote através de uma ou mais portas do *switch*. Por fim a mensagem do tipo FLOW_MOD é enviada do Controlador para o *switch* com o objetivo de instruí-lo a adicionar um fluxo em particular na sua tabela de fluxos.

A Figura 1 representa os elementos do OpenFlow. O Switch possui tabelas de fluxos (*Flow Tables*) e estabelece uma comunicação segura com o Controlador. Assim que um frame é recebido pelo *switch* ele é analisado e caso não haja entrada equivalente na tabela de fluxos, o switch deve ser capaz de encapsular o pacote e encaminhá-lo para o controlador através do canal seguro. O controlador irá decidir o que fazer com esse pacote. Inclusive, decidirá se uma entrada na tabela de fluxos do *switch* será adicionada ou não.

O controlador é uma parte fundamental do OpenFlow, já que são responsáveis por definir a lógica de encaminhamento através das regras que irão configurar nos *switches*

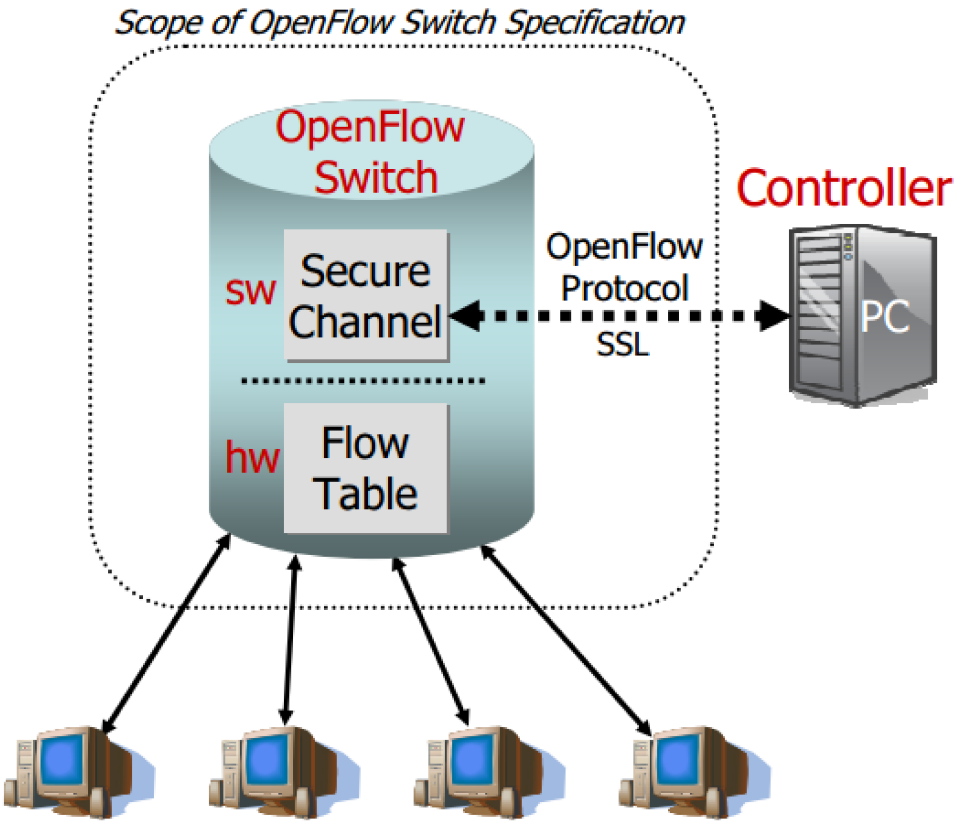

Figura 1 – Visão geral do OpenFlow

Fonte: (MCKEOWN et al., 2008)

OpenFlow. Atualmente, existem diversos controladores disponíveis para o uso, por exemplo, Floodlight (Big Switch Networks, 2014), Hyperflow (TOOTOONCHIAN; GANJALI, 2010), Maestro (NG; CAI; COX, 2010), entre outros. A Tabela 1 apresenta um resumo das características de alguns controladores.

2.4 Propostas para Internet do Futuro

Várias propostas de Internet do Futuro foram feitas nos últimos anos para suportar requisitos do futuro. Sabe-se que a arquitetura atual não consegue lidar com alguns desses requisitos, como mobilidade, segurança, eficiência energética, entre outros. Objetivando preparar a rede para tratar tais questões, pesquisadores vêm trabalhando em novos modelos e arquiteturas. A seguir serão apresentadas algumas dessas propostas com foco em segurança, abordando os mecanismos de autenticação e controle de acesso de cada proposta.

Tabela 1 – Controladores OpenFlow

Fonte: (NUNES et al., 2014)

Controlador	Implementação	Software Livre	Desenvolvedor
POX	Python	Sim	Nicira
NOX	Python/C++	Sim	Nicira
MUL	C	Sim	Kulcloud
Maestro	Java	Sim	Universidade Rice
Trema	Ruby/C	Sim	NEC
Beacon	Java	Sim	Stanford
Jaxon	Java	Sim	Desenvolvedores Independentes
Helios	C	Não	NEC
Floodlight	Java	Sim	BigSwitch
SNAC	C++	Não	Nicira
Ryu	Python	Sim	NTT, grupo OSRG
NodeFlow	JavaScript	Sim	Desenvolvedores Independentes
ovs-controller	C	Sim	Desenvolvedores Independentes
Flowvisor	C	Sim	Stanford/Nicira
RouteFlow	C++	Sim	CPqD

2.4.1 RINA

A *Recursive InterNetwork Architecture* (RINA) é uma arquitetura de rede de computadores que unifica computação distribuída e de telecomunicações. O princípio dessa arquitetura é que redes de computadores são apenas *Inter-Process Communication* (ICP). Ela é basicamente a implementação de uma teoria conhecida informalmente como "IPC model"(DAY; MATTA; MATTAR, 2008), sendo que os seus conceitos e resultados são genéricos para qualquer aplicação distribuída e não apenas para redes de computadores.

Diferentemente da arquitetura atual, que foi projetada em várias camadas sobrepostas, onde cada uma possui funções específicas, esta proposta defende o uso de uma única camada que se repete, o *Distributed IPC Facility* (DIF), que provê comunicação IPC, de maneira distribuída (WANG et al., 2014). Um aspecto significativo na arquitetura RINA é que ela fornece mobilidade, *multihoming* e *Quality of Service* (QoS) de maneira intrínseca.

IPC model captura os elementos comuns das aplicações distribuídas, chamados de *Distributed Application Facilities* (DAFs). Um DAF é composto por um ou mais *Distributed Application Processes* (DAPs), que colaboram para executar uma tarefa.

DAPs se comunicam usando um único protocolo chamado *Common Distributed Application Protocol* (CDAP), que permite que dois DAPs troquem dados estruturados na forma de objetos. O uso de um único protocolo fornece as aplicações uma maior simplicidade, ao contrário da arquitetura atual que contém vários protocolos, por exemplo, HTTP, FTP, SSH, entre outros.

Todas as informações visíveis externamente através do DAP são representadas na forma de objetos e são estruturadas no *Resource Information Base* (RIB), que fornece uma organização lógica dos objetos e um *naming schema*. O CDAP permite que os DAPs realizem seis operações *(create, delete, read, write, start and stop)* remotas em objetos.

Com o objetivo de trocar informações, os DAPs precisam de um mecanismo subjacente para fornecer comunicação entre eles. Este mecanismo é outro DAF cuja tarefa é fornecer e gerenciar o serviço de IPC dentro de um determinado escopo, sendo assim, esse DAF pode ser chamado de DIF. Um DIF permite a um DAP alocar fluxos para um ou mais DAPs, fornecendo apenas os nomes dos DAPs desejados e as características necessárias para esse fluxo, por exemplo: atraso, limites sobre a perda de dados, ordem da entrega de dados, confiabilidade, etc. Os DAPs não podem confiar no DIF que estão usando, portanto, podem decidir proteger os seus dados antes de escrevê-los no fluxo, por exemplo, usando criptografia através do módulo de proteção *Service Data Unit*(SDU).

Os DIFs também podem ser utilizados por outros DIFs subjacentes, criando assim a estrutura recursiva da arquitetura RINA, onde cada DIF fornece serviços IPC ao longo de um escopo limitado. O primeiro nível opera em cima de um meio físico e suas políticas são otimizadas para lidar com as particularidades do meio físico. O primeiro nível de DIFs fornece serviços IPC para o segundo nível e assim por diante. Os protocolos em cada camada são os mesmos. Eles só usam configurações ou políticas diferentes para cumprir os requisitos específicos de cada camada.

Os DAPs, que são membros de uma DIF, são chamados de *IPC Processes* (IPCP). Eles possuem a mesma estrutura de um DAP genérico, além de algumas tarefas específicas para fornecer e gerenciar um IPC. Estas tarefas podem ser dividias em três categorias: *data transfer, data transfer control and layer management*. Um ponto relevante é que a RINA utiliza uma abordagem evolutiva que propõe uma transição gradativa da arquitetura atual para a nova, utilizando a vantagem do DIF poder operar acima, abaixo, ou ao lado da Internet.

A segurança na arquitetura atual é projetada em cada protocolo, ao invés de ser no nível do sistema, fazendo com que a segurança se torne complexa. No modelo recursivo da arquitetura RINA, ela fornece um modelo de segurança claro, pois as relações de confiança entre as camadas (DAF ou DIFs) e os membros de uma única camada são bem identificadas. Nas subseções seguintes serão detalhados os mecanismos de segurança presentes na estrutura da arquitetura RINA responsáveis pela autenticação e controle de acesso.

2.4.1.1 Autenticação

A autenticação(GRASA et al., 2016a) na arquitetura RINA é realizada quando um membro deseja participar de um DIF. Várias políticas de autenticação podem ser inseridas no *Common Application Connection Establishment Phase* (CACEP) (GRASA et al.,

2012), variando de políticas que não utilizam nenhum tipo de autenticação até aquelas que usam técnicas criptográficas assimétricas. Essas políticas permitem que o CACEP se torne extensível, propiciando atender os requisitos de autenticação para qualquer configuração.

Em (VRIJDERS et al., 2014) são discutidas algumas implementações que pertencem ao projeto IRATI, sendo que o mecanismo de autenticação não é explorado de uma maneira que comprove o seu funcionamento. No projeto PRISTINE (GRASA et al., 2016a) foi feita uma demonstração do funcionamento do mecanismo de autenticação.

2.4.1.2 Controle de Acesso

O controle de acesso(GRASA et al., 2016a) nessa arquitetura acontece no momento em que um fluxo de comunicação é alocado. Pode acontecer com processos locais (internos a um DIF) sendo impedidos de alocação de fluxos de saída, assim como processos remotos podem ser impedidos de alocação de fluxos de entrada.

Nas implementações que surgiram durante o projeto IRATI (VRIJDERS et al., 2014), não foi criado nenhum protótipo e nem foi demonstrado o funcionamento do mecanismo de controle de acesso. Entretanto, no PRISTINE (GRASA et al., 2016a) tal mecanismo é implementado e demostrado.

2.4.2 MobilityFirst

A arquitetura *MobilityFirst* possui aspectos fundamentais como mobilidade e confiabilidade. Ela assume que todos os equipamentos conectados à Internet são móveis, sendo assim, trata-se de uma arquitetura centrada na mobilidade como regra e não como exceção (BRONZINO et al., 2013). Para garantir a robustez às comunicações, mesmo na ocorrência de desconexões, ela utiliza *Generalized Delay-Tolerant Networking* (GDTN)(SESKAR et al., 2011).

Sua estrutura é composta por três identificadores e dois serviços de mapeamento que exercem papéis de grande importância. Os identificadores são: *Network Address* (NA); *Globally Unique Identifier* (GUID); e *Human-Readable Name* (HRN). A função do HNR é garantir a mobilidade separando informações de localização da rede (NA) de sua identidade (GUID). Os serviços de mapeamento são: *Name Assignment Services* (NAS) e *Global Name Resolution Service* (GNRS).

O NAS vincula um HRN a um GUID e o GNRS (BRONZINO; RAYCHAUDHURI; SESKAR, 2015) mapeia o GUID em um NA. O GNRS tem função de vincular dinamicamente o nome e a localização. Quando o conteúdo está disponível em mais de um local, GNRS escolhe o local mais próximo do solicitante do conteúdo. Quando não é mais possível encontrar o caminho pelo NA, o GNRS tenta descobrir o novo NA associado ao GUID.

Por conseguinte é possível identificar alguns mecanismos de segurança na arquitetura, sendo um deles o uso de um GUID que é atribuído a cada conteúdo como um endereço (BRONZINO; RAYCHAUDHURI; SESKAR, 2015). O GUID é resultado do *hashing* do conteúdo e pode ser usado como uma chave pública para o mecanismo de criptografia. Outro é a frequente atualização de roteamento usando a função de GNRS (ZHANG et al., 2012). Por fim temos o uso de um protocolo integrado que permite auto-certificação de chaves públicas (RAYCHAUDHURI; NAGARAJA; VENKATARAMANI, 2012).

Após analisar alguns aspectos de segurança da arquitetura, é possível verificar quais ataques são prevenidos. Ela é robusta contra o ataque de *snooping* que gera invasões sem fins lucrativos, pois ele tem um mecanismo para cifrar os pacotes usando o nome ou a GUID do conteúdo como o identificador. Para o ataque de *data modification* que provoca a alteração dos dados antes que cheguem até o destinatário, ela é capaz de atenuá-lo atribuindo um GUID exclusivo para cada conteúdo. O GUID é um hash de um conteúdo, portanto, o receptor pode verificar a exatidão do conteúdo recebido.

MobilityFirst é capaz de atenuar os ataques conhecidos como *man-in-the-middle* (MITM), *reflection* e *masquerading*, pois usa um protocolo que é capaz de autenticar um usuário e possui um GUID exclusivo para cada conteúdo. É possível impedir o ataque de DoS utilizando a função do GNRS para ativar a atualização de roteamento e atenuar o ataque de repúdio por ter uma mensagem de não-repúdio criada pela PKI.

Apesar desses benefícios, a MobilityFirst não pode atenuar a *traffic analysis* e *replaying attacks*. No primeiro caso, a arquitetura não possui um mecanismo que possibilite a comunicação anônima. No segundo, a arquitetura não possui um mecanismo para ligar as mensagens às sessões. Nas subseções seguintes serão detalhados os mecanismos de segurança presentes na estrutura da arquitetura responsáveis pela autenticação e controle de acesso.

2.4.2.1 Autenticação

O *name service* traduz um nome legível para um GUID exclusivo (identificador global). Esse GUID é derivado simplesmente como um *hash* unidirecional de uma chave pública. Assim, um GUID pode ser autenticado usando um procedimento simples desafio-resposta, bilateral, que não exige autoridade de certificação externa. Os *principals* (MUKHERJEE et al., 2014) (VU et al., 2012) podem se autenticar entre si utilizando o GUID, sem a necessidade de certificação de terceiros. É possível observar que a arquitetura fornece o mecanismo de autenticação, entretanto não foi demonstrada nenhuma implementação na arquitetura utilizando tal mecanismo.

2.4.2.2 Controle de Acesso

O controle de acesso é integrado ao GNRS utilizando a *eXtensible Access Control Markup Language* (XACML), ou seja, essas políticas de controle de acesso são escritas

diretamente no GNRS. A XACML pode lidar com o controle de acesso básico utilizando uma *Identity-Based Access Control* (IBAC) e *Attribute-based Access Control* (ABAC) (LIU; TRAPPE; ZHANG, 2013) (VENKATARAMANI et al., 2013). Apesar de descrever o mecanismo de controle de acesso e apresentar como seria o seu funcionamento, até o momento não foi encontrado nenhum protótipo que comprove o seu funcionamento.

2.4.3 XIA

A *eXpressive Internet Architecture* (XIA) (HAN et al., 2012) fornece suporte nativo para diversos *Principals* (*host, service, content, admin domain, geocast)* e a capacidade de evoluir suas funcionalidades para atender novos requisitos e *Principals* ao longo do tempo. Ela fornece a segurança de forma intrínseca, ou seja, permite validar propriedades de segurança específicas sem depender de informações externas, por exemplo, um banco de dados ou configurações externas.

Principals são identificados por *unique eXpressive Identifiers* (XIDs), que são utilizados como origem e destino para o encaminhamento dos pacotes na rede. Estes XIDs são intrinsecamente seguros e devem ser exclusivos. Portanto, XIDs devem ser gerados de forma distribuída e resistente a colisão, sem a ajuda de uma autoridade central.

Identificando as entidades que fazem parte da comunicação, a complexidade e o *overhead* são reduzidos, pois não há necessidade de forçar toda a comunicação ao nível dos *hosts*, como na Internet hoje. Para referenciar os XIDs(ANAND et al., 2011) associados aos *Principals* são usados o *Content ID* (CID), *Host ID* (HID),*Service ID* (SID) e *Administrative Domains* (ADs). O XID é gerado por meio de criptografia de algo relacionado ao *Principal* associado, por exemplo, utilizando um hash da sua chave pública.

A arquitetura XIA possui como base três requisitos básicos (ANAND et al., 2011): usuários e aplicações devem ser capazes de expressar a sua intenção; tipos de *principals* devem ser capazes de evoluir; e os identificadores dos *principals* devem ser intrinsecamente seguros.

Os usuários e aplicações devem ser capazes de expressar as suas intenções, ou seja, a arquitetura tem de suportar uma ampla gama de intenções. Com esse requisito, obtém-se uma maior flexibilidade na rede, sendo capaz de evoluir, ou seja, deverá ser possível acrescentar ou modificar modelos que compõem o núcleo da arquitetura com uma complexidade baixa.

Os tipos de *principals* devem ser capazes de evoluir, ou seja, é possível introduzir continuamente novos tipos de *principals* ao longo do tempo e modificar as funcionalidades dos *principals* existentes. Por exemplo, um *host* ou *administrative domain* não são capazes de expressar o desejo de se comunicar com todos os *nodes* de uma determinada área geográfica, entretanto, pesquisas posteriores poderiam definir o "*GeoCast*"como um tipo de *principal* para atender essa necessidade. Além disso, a complexidade de adicionar suporte

para um novo tipo *principal* deve ser razoavelmente baixa. Sendo possível adicionar os tipos de *principals* de forma incremental, por exemplo, primeiro adicionando suporte fim-a-fim, em seguida, fornecer suporte parcial da rede e posteriormente global.

Identificadores devem ser intrinsecamente seguros, permitindo às entidades validarem se estão comunicando com o *principal* correto, ou seja, a semântica de segurança pode e variará conforme o tipo do *principal*. Por exemplo, o requisito básico de uma comunicação baseada em *host* é autenticar os respectivos *hosts*, enquanto que na recuperação do conteúdo, é garantir a integridade e validade dos dados obtidos.

A arquitetura possui alguns mecanismos de segurança, por exemplo, a sua forma de recuperar o conteúdo com os seus identificadores: CID, HID e SID. O *Lightweight Anonymity and Privacy* (LAP) (HSIAO et al., 2012), permite comunicação anônima para evitar rastreamento remoto. O STRIDE (HSIAO et al., 2013) aloca a largura de banda disponível em uma topologia baseada em árvore e por fim o *Accountable Key Infrastructure* (AKI) (KIM et al., 2013a), que fornece o processo de autenticação dos dados e faz uso da arquitetura SCION (*Scalability, Control, and Isolation On Next-Generation Networks*). Nas subseções seguintes serão descritos os mecanismos de segurança presentes na estrutura da arquitetura responsáveis pela autenticação e controle de acesso.

2.4.3.1 Autenticação

Essa arquitetura possui uma *Accountable Key Infrastructure* (KIM et al., 2013b), que fornece um processo de autenticação de dados confiável, e o *source authentication* (KIM et al., 2014) permite que roteadores autentiquem o remetente e o conteúdo do pacote. O XID de um *principal* por ser gerado através do *hash* de uma chave pública, sendo assim, qualquer cliente que acesse o serviço pode intrinsecamente verificar se ele está se comunicando com uma instância adequada do serviço. Apesar de apresentar o mecanismo de autenticação, não foi encontrado até o momento uma implementação que mostre resultados com o mecanismo elaborado.

2.4.3.2 Controle de Acesso

A arquitetura não mostra claramente como é feito o controle de acesso, apenas cita que existe um *provider* que é responsável por tal tarefa.

2.4.4 Named Data Networking

Assim como o sistema de telefonia não é a solução mais eficiente para a difusão de conteúdo feito por TVs e rádios, a arquitetura de rede atual também não é a solução mais viável para a sua principal aplicação. Entretanto, a *Named Data Networking* (NDN) utiliza alguns princípios da arquitetura atual (JACOBSON et al., 2009).

NDN utiliza seis princípios arquitetônicos, sendo que os três primeiros são derivados do sucesso da Internet de hoje e os três últimos das lições aprendidas ao longo dos anos. Ela reconhece a importância do protocolo TCP/IP mas relembra que a ideia original foi resolver a questão de comunicação ponto-a-ponto entre duas entidades, além disso, relembra que o IP é utilizado de maneira massiva para distribuição de conteúdo.

A arquitetura de *hourglass* é o que torna o projeto original da Internet poderoso. Concentra-se sobre uma camada de rede universal (IP) implementando o mínimo de funcionalidade possível para a interconexão global. Isso é chamado de *thin waist* e tem sido um fator fundamental de crescimento explosivo da Internet atual, permitindo que as tecnologias de camada superior e inferior inovem sem restrições. A NDN mantém a mesma arquitetura em forma de ampulheta.

Similar à arquitetura IP de hoje, o *thin waist* é a peça central da arquitetura NDN. No entanto, a *thin waist* da NDN utiliza *data names* em vez de endereços IP para entrega de dados. A arquitetura define dois tipos de pacotes: um para a requisição *(interest packet)* e o outro para a resposta *(data packet)*. O *interest packet* possui dois campos essenciais: *content name* que identifica o conteúdo e *once* derivado de *number once* que vincula cada sessão de comunicação. No *data packet*, ele carrega o nome e conteúdo dos dados, em conjunto com a assinatura digital e informações assinadas.

Atualmente, os datagramas IP somente nomeiam os *endpoints* da comunicação com o IP de destino e origem. A NDN sugere remover essa restrição, permitindo adotar uma estrutura hierárquica de nomes para a comunicação. Por exemplo, essa estrutura poderia ser usada para nomear determinados tipos de dados em uma conversa, dados do Youtube ao invés de forçar a criação de uma conversa entre o consumidor e youtube.com. No modelo TCP/IP a responsabilidade de segurança (ou falta dela) fica para os *endpoints*, na arquitetura NDN os dados são protegidos, obrigando quem produziu o dado a assinar digitalmente cada pacote de dados. Essa assinatura garante a integridade e permite determinar a origem do dado.

É possível identificar alguns mecanismos de segurança presentes na arquitetura como a assinatura digital dos pacotes de dados, o envio do *nonce* (Number Once) dentro dos *interest packets* e pode ser usado uma criptografia fim-a-fim para criptografar os dados. No *trust model*, o *namespace* corresponde a uma hierarquia de confiança, ou seja, */author*, */admin*, */article*. Sendo assim, a publicação de chaves com um determinado nome na hierarquia vai autorizá-los a assinar pacotes de dados específicos e limita o seu escopo. Nas subseções seguintes serão descritos os mecanismos de segurança presentes na estrutura da arquitetura responsáveis pela autenticação e controle de acesso.

2.4.4.1 Autenticação

A autenticação nessa arquitetura é construída de forma nativa para validar os dados trafegados. Assim que um pacote é disponibilizado para distribuição, ele é explicitamente

nomeado e assinado pelo seu criador. Roteadores e clientes que consomem o conteúdo podem validar as assinaturas para garantir a autenticidade dos dados. Apesar da utilização de uma assinatura digital nos pacotes assegurar a autenticação e fonte dos dados, ela também impõe uma sobrecarga computacional significativa sobre a rede. Além disso, o tempo de vida dos dados pode ultrapassar o tempo de vida dos certificados de chave pública, o que implica em graves problemas de segurança (YU et al., 2015).

2.4.4.2 Controle de acesso

O controle de acesso não é fornecido pela camada de rede. Mesmo sendo possível implementar uma ACL, a NDN escolhe delegar o controle de acesso para a camada de aplicação. Em (YU; AFANASYEV; ZHANG, 2015), a arquitetura utiliza o *Name-based Access Control* (NAC) que mostra um modelo de controle de acesso baseado no conteúdo onde o proprietário dos dados controla diretamente o acesso e a produção dos dados, ou seja, quem está autorizado a produzir dados e quem está autorizado a ler os dados.

2.4.5 NEBULA

A arquitetura NEBULA (ANDERSON et al., 2014) facilita a *data centers*, em um ambiente de *cloud*, se comunicarem de forma confiável. Ela é composta por três componentes: *NEBULA Core* (NCore); *NEBULA Data Plane* (NDP); e *NEBULA Virtual and Extensible Networking Techniques* (NEVENT). O NCore interliga os *data centers*, utilizando um mecanismo de roteamento confiável. NDP é o plano de dados que fornece controle de acesso flexível e mecanismos de segurança. NEVENT é o plano de controle, responsável por determinar o caminhos dos pacotes para alcançar o seu destino.

Essa arquitetura tem como objetivo principal de projeto conceber um serviço de *cloud* incorporado com segurança e confiabilidade, alta disponibilidade, integração com *data centers* e roteadores que possua viabilidade econômica e regulamentar. A seguir são descritos alguns princípios da arquitetura:

- ❏ Caminhos paralelos entre *data centers* e *core routers*;
- ❏ Garantir segurança no acesso e trânsito da informação;
- ❏ Um mecanismo de seleção de caminho baseado em políticas; e
- ❏ Autenticação executada durante o estabelecimento da conexão.

É possível identificar os seguintes mecanismos de segurança na arquitetura:

- ❏ *Proof of Consent* (PoC) mecanismo para autorizar um pacote em caminho;
- ❏ *Proof of Path* (PoP) a fim de garantir que o pacote flui apenas no caminho autorizado;

- *Token* para ligar uma sessão de comunicação autorizada;
- Existe um servidor de consentimento na NEBULA que pode atuar como um terceiro de confiança. Este servidor vai provar que a comunicação entre dois usuários realmente ocorreu porque os usuários que querem enviar um pacote vão contactá-lo para obter o PoC.

Nas subseções seguintes serão detalhados os mecanismos de segurança presentes na estrutura da arquitetura responsáveis pela autenticação e controle de acesso.

2.4.5.1 Autenticação

O mecanismo de autenticação não é detalhado na literatura, deixando claro que tal mecanismo não foi criado, entretanto ele pode ser realizado durante o estabelecimento da conexão.

2.4.5.2 Controle de acesso

O controle de acesso acontece no *NEBULA data plane* (NDP) onde estão localizados os mecanismos de *Proof of Consent* (PoC), que são responsáveis por autorizar um pacote e um caminho e o *Proof of Path* (PoP) que serve para garantir que o pacote flua apenas no caminho autorizado.

2.5 Análise

As arquiteturas que estão surgindo para Internet do Futuro possuem a preocupação de incorporar a segurança em seu projeto, pois muitos problemas que surgiram em relação à segurança na arquitetura atual foi gerado pela falta de segurança incorporada ao seu projeto.

Todas as arquiteturas apresentadas anteriormente possuem preocupação com os mecanismos de autenticação e autorização, algumas discutem esses mecanismos em alto nível, o que dificulta uma comparação mais detalhada entre as arquiteturas. Entretanto, o objetivo nesse momento é mostrar o estado atual dos mecanismos de autenticação e controle de acesso em cada arquitetura.

A Tabela 2 resume os mecanismos de autenticação das entidades e de controle de acesso das arquiteturas apresentadas na Seção 2.4. Nela são usadas duas classificações: Completo, representado por um *Bullet* (•), indicando que o mecanismo está disponível para utilização; e Incompleto, representado por um *Binary Times* (⊗), indicando que o mecanismo está disponível parcialmente na arquitetura ou a arquitetura fornece meios para facilitar sua implementação.

Tabela 2 – Comparativo dos Mecanismos de Segurança nas Arquiteturas

Mecanismos	Arquiteturas					
	RINA IRATI	RINA PRISTINE	MobilityFirst	XIA	NDN	Nebula
Autenticação	⊗	•	⊗	⊗	•	⊗
Controle de Acesso	⊗	•	⊗	⊗	⊗	⊗

É possível observar na Tabela 2 que tais mecanismos não estão disponíveis para uso nas arquiteturas MobilityFirst, XIA, Nebula. Existe uma discussão na literatura de como tais mecanismos podem funcionar nessas arquiteturas, mas, até onde pesquisamos, não há nada que comprove que algo foi feito nesse sentido.

A arquitetura NDN atualmente fornece somente o mecanismo de autenticação, entretanto, não possui implementado o mecanismo de controle de acesso, apenas apresenta como ele pode ser realizado na arquitetura, mas não existe nenhum trabalho demonstrando um protótipo efetivo ou apresentando o seu funcionamento.

A arquitetura RINA sofreu evoluções em dois diferentes projetos: FP7 IRATI (VRIJDERS et al., 2014) e FP7 PRISTINE (GRASA et al., 2016a). No primeiro projeto não houve a preocupação com mecanismos de segurança, entretanto, quando iniciou o segundo projeto um dos objetivos principais era a segurança e foi então que a arquitetura obteve tais mecanismos. Entretanto, a RINA não separa os planos de dados e de controle, mantendo algumas limitações atuais. Deve ser destacado também que o protótipo resultante do projeto PRISTINE foi liberado recentemente no ano de 2016 (GRASA et al., 2016a).

Capítulo 3

Entity Title Architecture

Elaborar e qualificar uma arquitetura alternativa, para substituir ou evoluir a estrutura da Internet atual é um dos principais objetivos das pesquisas em Internet do Futuro. A necessidade de mudança é discutida pelos cientistas desde o início da década de noventa (CLARK et al., 1991).

A arquitetura inicial da Internet já passou por várias evoluções como, por exemplo, o endereçamento (EGEVANG; FRANCIS, 1994) (GROUP; HINDEN, 1993) (SRISURESH; EGEVANG, 2001) e o roteamento (REKHTER; LI; HARES, 2006). Varias iniciativas de pesquisa (PAN; PAUL; JAIN, 2011) (GAVRAS et al., 2007) estão empenhadas a fim de fornecer uma solução para as novas exigências relativas à arquitetura Internet.

Uma inciativa brasileira nessa área é a ETArch (SILVA, 2013) que foi construída sobre uma infraestrutura baseada em SDN, na qual o Plano de Controle é separado do Plano de Dados. O seu processo de definição partiu de um modelo de referência (Bass, 2006), neste caso o *Entity Title Model* (PEREIRA et al., 2011) e padrões arquiteturais (BUSCHMANN et al., 1996) (SCHMIDT et al., 2000) adequados que levam em consideração os requisitos propostos pela arquitetura.

O *Entity Title Model* separa a identificação de uma entidade de sua localização (endereço), propondo uma forma de endereçamento horizontal, onde os diversos elementos na rede são identificados por um único nome (PEREIRA, 2012).

A ETArch é uma arquitetura que se propõe a atender requisitos da Internet do Futuro. Desde sua criação, pesquisadores de várias universidades vêm trabalhando para solucionar problemas presentes na arquitetura atual de Internet, como mobilidade (GUIMARAES et al., 2014), multicast (GONçALVES et al., 2014) e qualidade de serviço (QoS) (CASTILLO et al., 2014).

Neste capítulo, serão expostos os principais conceitos da ETArch, bem como as suas camadas e funções.

3.1 Aspectos Conceituais e Arquiteturais da ETArch

Esta seção expõe os principais conceitos propostos pelo *Entity Title Model* e seu uso pela ETArch. Além dos conceitos, são apresentados os principais componentes e suas respectivas funções.

3.1.1 Camadas ETArch

A ETArch adota um padrão arquitetural em camadas(*layers*). Apesar de ser uma abordagem *clean slate*, a arquitetura se propõe a suportar protocolos presentes na camada de Aplicação e continuar utilizando os protocolos da camada Física existente.

A arquitetura ETArch define três camadas denominadas *Application*, *Communication* e *Link*, representadas na Figura 2. A principal mudança em relação ao modelo de endereçamento da arquitetura Internet atual reside no fato de ser implementado o endereçamento horizontal.

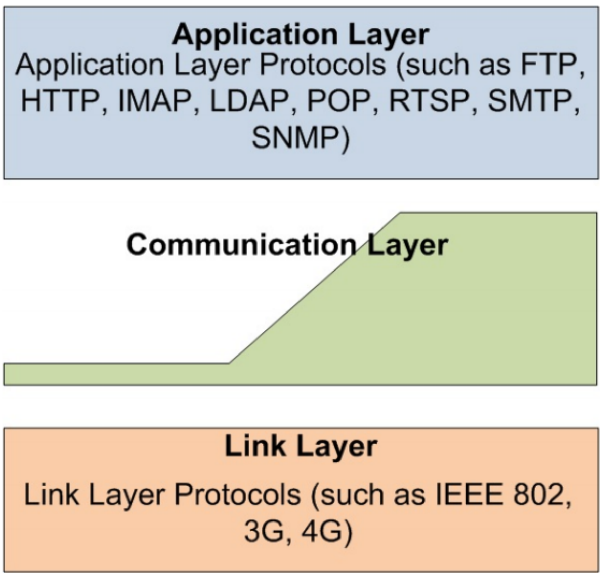

Figura 2 – Camadas definidas pela ETArch

Fonte: (SILVA et al., 2012)

Note-se que a camada *Communication* possui um formato atípico para uma camada, indicando a capacidade de atender requisitos de complexidade variável. Esta forma pretende explicitar a flexibilidade da arquitetura ETArch, permitindo atender os requisitos de aplicações. Por exemplo, uma aplicação pode requerer confidencialidade e confiabilidade.

3.1.2 Título

Título é um conceito fundamental para a ETArch e é definido como um identificador único, não ambíguo e independente da topologia da rede subjacente (PEREIRA, 2012).

A ETArch utiliza o conceito de Título para identificar as entidades de forma unívoca em seu espaço de nomes (*Namespace*), que também deve ser único. Alguns exemplos de entidades identificáveis em um *Namespace* e, sendo assim, passíveis de serem tituladas são: *host*, sensor, *phone*, conteúdo, entre outros.

Namespace é basicamente um espaço de Nomes, com semântica própria, e tem a responsabilidade de acrescentar um significado inicial à associação da Entidade com seu Título. Um Título também pode ser visto como uma credencial que pode ser usada para relacionar aspectos de segurança (REC, 1991).

É importante ressaltar que um Título acompanha o ciclo de vida da Entidade, independentemente de sua localização, diferentemente da forma hierárquica criada no mundo IP. Qualquer entidade que possua a intenção de usufruir da Internet e que pretenda se comunicar, deve possuir um nome válido. A exclusividade do Título, e a independência da localização, favorece a mobilidade de Entidades. Com isso, o Título possui a atribuição central de promover o endereçamento horizontal (PEREIRA; KOFUJI; ROSA, 2010) de entidades.

Na ETArch, o Título também é utilizado para nomear *Workspaces* (ver 3.1.6). Essa nomeação acontece de forma semelhante às arquiteturas do tipo Information-Centric Networking (ICN) (XYLOMENOS et al., 2014) em relação à nomeação de informações. O fato do *Workspace* possuir um Título permite nomear o que será solicitado em uma determinada comunicação.

3.1.3 Entidade

Uma Entidade ETArch é definida como um ente que possui capacidade de se comunicar em um ambiente distribuído. Sendo assim, exemplos de Entidades podem ser hosts, smartphones, Network Elements (NEs), usuários, aplicações, coisas, dentre outros.

Entidades podem manter relações com outras entidades e, por meio dessas relações, podem herdar propriedades, exceto Título (ver 3.1.2). Por exemplo, uma aplicação pode estar ligada a um *host* e um sensor pode estar vinculado a um smartphone.

Uma Entidade possui pelo menos um Título, que a identifica unicamente, e uma localização, denominado Point of Attachment (PoA). Exemplos de PoA podem ser *switches*, Access Points (APs), etc. Esta separação permite que a localização de uma entidade possa variar ao longo do tempo (mobilidade).

Além do Título e da Localização, uma entidade possui um conjunto de requisitos, tais como largura de banda, QoS, Quality of Experience (QoE), entre outros. Dessa forma, antes de iniciar uma comunicação, uma Entidade especifica seus requisitos.

Essa abordagem permite uma maior flexibilidade e capacidade evolutiva, pois ela possibilita lidar com uma grande diversidade de condições e fornece meios para que novos requisitos sejam definidos ao longo do tempo.

3.1.4 *Domain Title Service Agent*

Na ETArch, o Domain Title Service (DTS) é um sistema distribuído responsável pela materialização do Plano de Controle. Ele é responsável por controlar o ciclo de vida de Entidades, desde seus registros, durante os contextos de comunicação, até a finalização de suas atividades. Sendo assim, o DTS mantém a base de conhecimento sobre todos os aspectos de controle do ambiente distribuído, por exemplo, o rol de *Workspaces*, Títulos, Entidades, Network Elements (NEs) etc.

O DTS é composto por agentes denominados Domain Title Service Agents (DTSAs), como representado na Figura 3, formando assim um superconjunto de controladores SDN. Os Domain Title Service Agents (DTSAs) são entidades responsáveis por comunicações de controle e monitoração com um ou mais Network Elements (NEs) (*switches*) e, também, com outros Domain Title Service Agents (DTSAs).

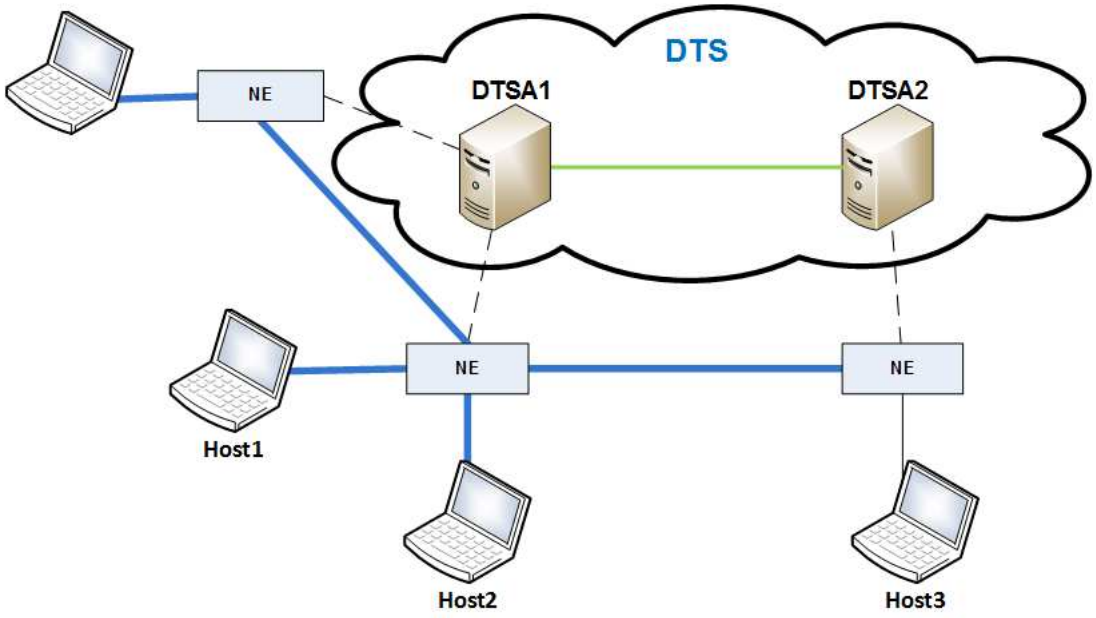

Figura 3 – Visão Geral do DTS

A interconexão dos DTSAs é realizada por meio de Network Elements (NEs), por exemplo switches, que podem ser exclusivos para o plano de controle ou compartilhado com o plano de dados. Na presença de uma falha em um NE, o DTS é capaz de realizar a reconfiguração do *workspace*, com o objetivo de manter a comunicação.

Na Figura 3, as linhas em azul são responsáveis por encaminhar dados entre os NEs e a nuvem composta pelos DTSA1 a DTSA2 representam o Plano de Controle. Note-se que um DTSA pode controlar 1 ou mais elementos de rede, sendo representado pelas linhas tracejadas.

3.1.5 Master-DTSA

Um conjunto de DTSAs é conhecido como um domínio, por exemplo, imagine duas cidades diferentes, onde cada uma possui o seu domínio, para que os DTSA existentes em domínios diferentes possam se comunicar, é necessário um agente chamado Master DTSA (MDTSA).

O MDTSA é a interface de contato entre dois ou mais domínios cujo *Workspaces* de Controle podem ser Públicos ou Privados. O MDTSA possui informações sobre as interconexões dos DTSAs do domínio.

As arquiteturas tradicionais possuem domínios para oferecer interconexão, os Service Providers (SPs), um exemplo disso são os Internet Service Providers (ISPs). No caso do MDTSA pode se comunicar com outro através de um *Workspace* de Controle Público.

Um MDTSA conhece as informações sobre Workspaces mantidos por DTSAs sob seu domínio. Sendo assim, quando uma entidade cria ou estende um Workspace, o DTSA, que controla localmente a referida entidade deverá informar ao MDTSA do domínio.

A Figura 4 apresenta a estrutura onde o MASTER_DTSA_1 possui sob seu domínio o DTSA1 e DTSA2 e o MASTER_DTSA_2 o DTSA3 e DTSA4.

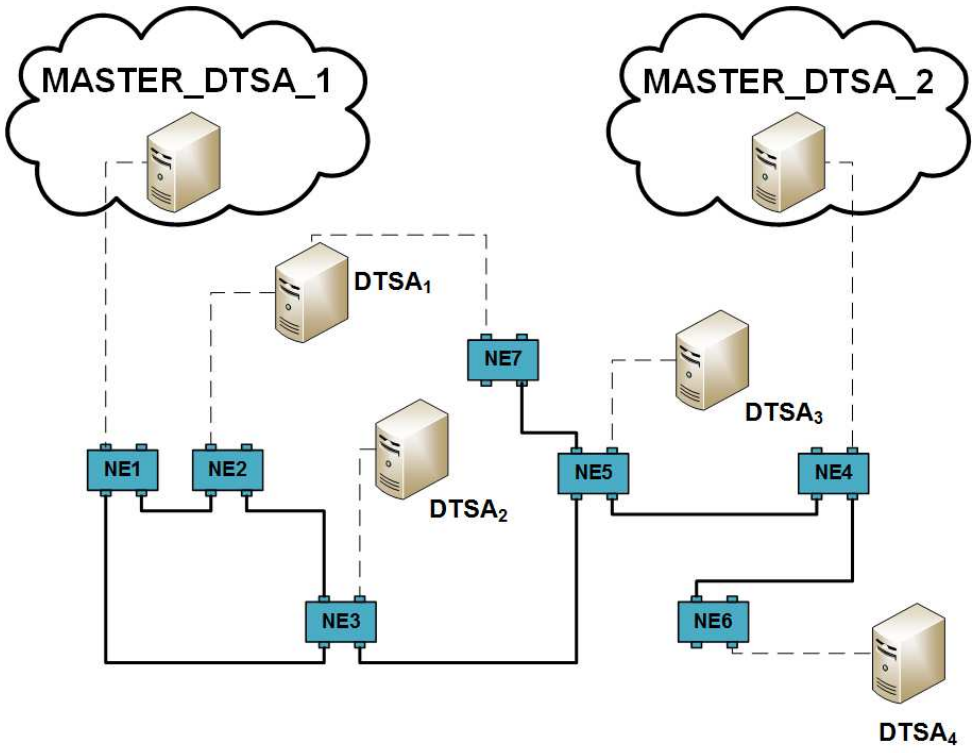

Figura 4 – Visão do Master-DTSA

3.1.6 *Workspace*

Workspace é um conceito fundamental na arquitetura ETArch e pode ser definido como um barramento lógico, que independe da topologia e da natureza da rede subjacente.

Workspace é o elemento fundamental nas comunicações na arquitetura ETArch, sendo condição *sine quo non* para as comunicações entre Entidades.

Um *Workspace* é identificado por seu Título e pode ser distribuído através de um ou mais elementos de rede (NE), cabeados ou sem fio, controlados por um ou mais DTSAs, e possuem as seguintes propriedades:

- ❏ Título: elemento chave, identifica univocamente um *Workspace*;
- ❏ Lista de NE: enumera os Títulos de elementos de rede que fazem parte do *Workspace*, bem como a especificação da conectividade;
- ❏ Lista de Capacidades: enumera as capacidades que podem ser oferecidas pelo *Workspace* às entidades, sendo exemplos parâmetros de QoS associados à comunicação, confidencialidade de dados, entre outras;
- ❏ Lista de Requisitos: enumera os requisitos que as entidades que desejam se ligar a determinado *Workspace* devem suportar, sendo semelhantes às Capacidades, porém neste caso representam requisitos que devem ser satisfeitos pelas entidades;
- ❏ Visibilidade: especifica se um *Workspace* é Público ou Privado;
- ❏ Nível: número natural que especifica o âmbito do domínio do *Workspace*, sendo que 'zero' permite visibilidade mundial.

Um *Workspace* pode ser criado a partir de requisições que Entidades enviam aos DTSAs. Nessa requisição, a entidade informa os requisitos que devem ser suportados pelo *Workspace*, bem como as capacidades que ele pode oferecer. A Figura 5 representa um *Workspace* e demais componentes da arquitetura que suportam seu funcionamento.

Uma vez criado, um *Workspace* é passível de ser compartilhado por Entidades, que queiram fazer parte desse domínio de comunicação. Entidades que atendam às propriedades do *Workspace*, podem se ligar (*attach*) a ele e nesse caso o DTSA é responsável por reconfigurar os Network Elements (NEs) para ligar a Entidade requisitante ao *Workspace*.

As capacidades de comunicação de um *Workspace* são configuradas no ambiente distribuído por meio de primitivas de Dados (Plano de Dados) e Controle (Plano de Controle). É importante frisar que o Título do *Workspace* é referenciado como endereço de destino das primitivas de comunicação durante as comunicações entre as Entidades, tanto dados quanto controle, e, então, ele é responsável por entregar as primitivas a todas as Entidades que estão conectadas a ele. A Figura 5 representa o *Workspace* e demais componentes da arquitetura.

Primitivas de Controle são os veículos para o transporte de informações de controle entre as Entidades que compõem o DTS, bem como, as Primitivas de Dados são os veículos para o transporte de dados entre as Entidades de Aplicação. Deste modo, há dois tipos de *Workspaces*, sendo *Workspace* de Controle e *Workspace* de Dados.

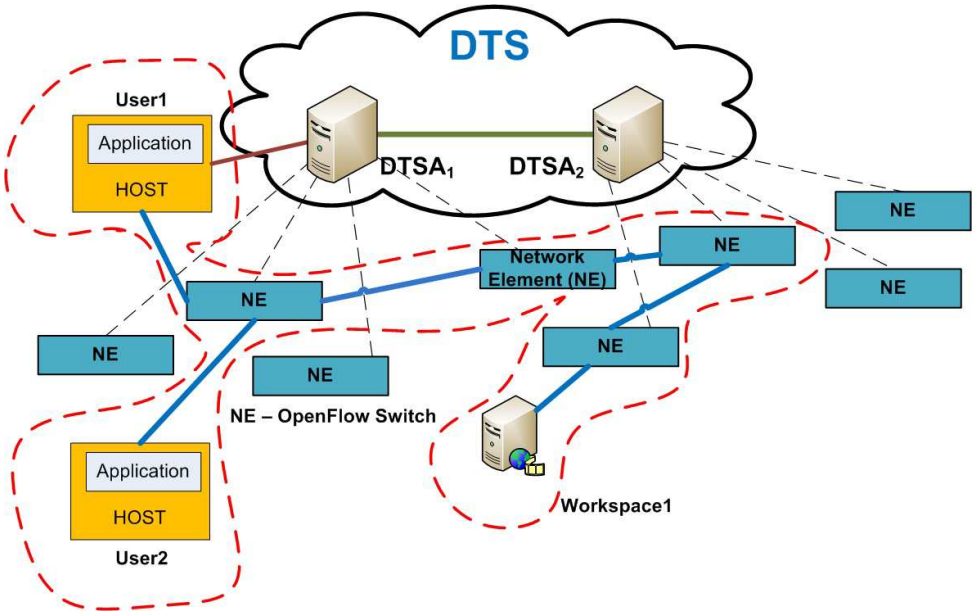

Figura 5 – Principais Componentes da Arquitetura ETArch

Fonte: (SILVA, 2013)

3.1.6.1 *Wokspace* de Controle

Wokspaces de Controle são criados para transportar as primitivas de controle responsáveis por manter o ambiente distribuído do DTS. Essencialmente, este tipo de *Workspace* mantém as comunicações entre DTSAs que compõem o DTS.

Wokspaces de Controle diferem do *Wokspace* Dados por três aspectos principais: i) como dito, eles transmitirem somente primitivas de controle, não servindo para transferência de dados genéricos das aplicações; ii) são criados durante a configuração do ambiente, a priori, e praticamente não sofrem alteração ao longo do tempo; e iii) podem se ligar a esse tipo de *Workspace* somente entidades dos tipos DTSA e MDTSA.

3.1.6.2 *Wokspace* de Dados

Wokspaces de Dados existem para suportar a troca de primitivas de dados entre Entidades de Aplicações. Esse tipo de *Workspace* é criado dinamicamente por Entidades que possuem privilégios para tal e que tenham algo que possa ser consumido em um ambiente distribuído. Eles são criados, mantidos e encerrados através de serviços do Plano de Controle (requeridos através de *Workspaces* de Controle), sendo que durante o seu ciclo de vida estão disponíveis para receber o *attach* de Entidades que queiram participar do domínio de comunicação.

3.1.7 Protocolos ETArch

Esta seção tem o objetivo de apresentar uma visão geral dos protocolos utilizados na arquitetura ETArch. A Figura 2 apresentou as camadas presentes na ETArch, a camada *Communication* possui dois conjuntos de serviços onde um está associado ao plano de Controle e o outro ao plano de Dados.

Os protocolos do plano de Controle são responsáveis pelo ciclo de vida de Entidades e *workspaces* de Dados e oferecem serviços como: registro de uma entidade no DTSA, criação de um *Workspace*, entrada e saída de entidades em um *workspace*, entre outros.

O *Entity Title Control Protocol* (ETCP) oferece serviços relacionados entre as entidades e os DTSAs. O *DTS Control Protocol* (DTSCP) é responsável por interações entre DTSAs e MDTSAs. Na Figura 6, esses protocolos são identificados respectivamente por P1 e P2.

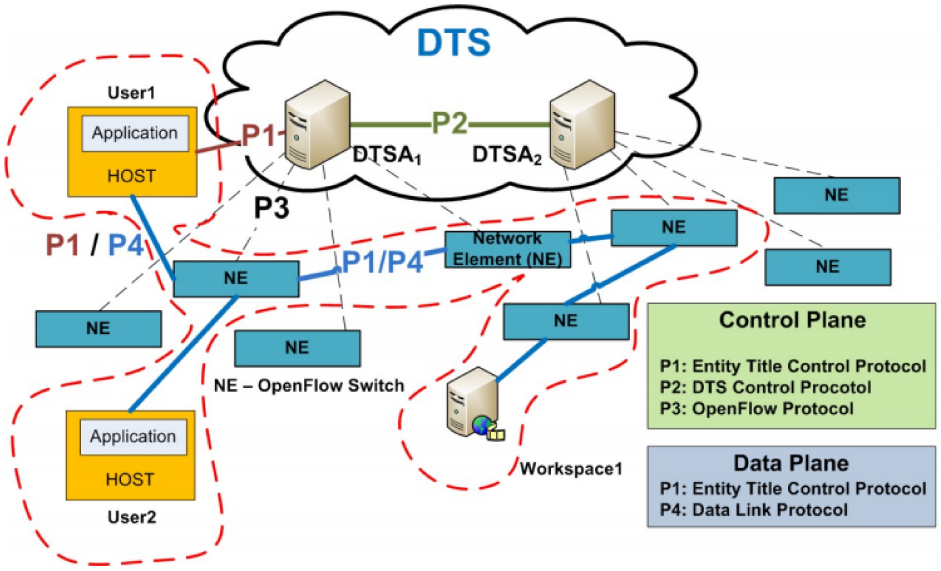

Figura 6 – Representação dos protocolos da arquitetura ETArch

Fonte: (SILVA, 2013)

Para uma Entidade poder interoperar no ambiente distribuído, ela precisa se registrar no DTS, que entre outras coisas autentica a Entidade solicitante. Após o registro, o ambiente do DTS sabe o Título da Entidade, que estará apta a requerer outros serviços. Assumindo-se que a referida Entidade queira se ligar (*Attach*) a um *Workspace* de Dados existente, a entidade entra na fase de operação do plano de Dados, essa fase as entidades estão aptas a enviar e receber primitivas de dados. Durante a fase de operação do plano de Dados, a entidade fará uso dos serviços do protocolo identificado como *Data Link Protocol* (P4).

3.1.7.1 Entity Title Control Protocol

O *Entity Title Control Protocol* (ETCP) é utilizado na camada *Communication* para regular a relação entre uma entidade e o DTSA. A Tabela 3 a seguir resume os nomes e funcionalidades das primitivas deste protocolo.

Tabela 3 – Primitivas do Entity Title Control Protocol (ETCP)

Fonte: (SILVA, 2013)

PRIMITIVA	FUNÇÃO
ENTITY-REGISTER	Registra uma entidade no DTS. A fim de ser registrada uma entidade deve apresentar o seu título, suas capacidades e requisitos de comunicação. Antes de iniciar qualquer comunicação a entidade deve realizar o seu registro. O Registro é realizado pelo DTSA que controla o NE onde a entidade está inicialmente conectada.
ENTITY-UNREGISTER	Remove uma entidade registrada do DTS e que não participa de nenhuma comunicação
WORKSPACE-ATTACH	Anexa uma entidade ao workspace. Para realizar este processo o DTSA realiza a configuração de todos os network elements (NE) a fim de extender o workspace até o NE através do qual a entidade está diretamente conectada.
WORKSPACE-CREATE	Cria um workspace. Inicialmente está presente apenas no DTSA local. Ao receber essa primitiva o DTSA pode executar uma divulgação (advertise), enviando suas informações para o Workspace Database de um terminado nível de rede.
WORKSPACE-DELETE	Remove um workspace, realizando a liberação de recursos necessários e atualização de todos os NEs envolvidos.
WORKSPACE-DETACH	Remove uma entidade de um workspace.
WORKSPACE-MODIFY	Modifica as capacidades de um determinado workspace.

Para que uma aplicação possa fazer uso deste protocolo é disponibilizado uma *Application Programming Interface* descrita na Tabela 4, que disponibiliza os serviços do ETCP para as entidades.

Tabela 4 – ETCP Application Programming Interface

Fonte: (SILVA, 2013)

FUNÇÃO	DESCRIÇÃO DA CHAMADA
register	Responsável pelo registro de uma entidade.
unregister	Responsável pela remoção do registro de uma entidade junto ao DTS.
createWorkspace	Utilizada por uma aplicação que é responsável pela criação de um Workspace.
attachWorkspace	Utilizada por uma aplicação para se anexar a um Workspace já existente.
detachWorkspace	Utilizada por uma aplicação a fim de se separar de um workspace.
deleteWorkspace	Utilizada pela aplicação que criou o workspace a fim de removê-lo ao DTS.
send	Utilizada para o envio de dados para um Workspace.
receive	Utilizada para o recebimento de dados de um Workspace.

3.1.7.2 DTS Control Protocol

O *DTS Control Protocol* (DTSCP) é responsável pelas relações de controle internas ao DTSA, ou seja, aquelas que envolvem DTSA-DTSA, DTSA-MDTSA, MDTSA-MDTSA. A Tabela 5 descreve as primitivas definidas para este protocolo. Assim como no protocolo ETCP, oferece-se uma API, cujos serviços seguem a mesma lógica de nomeação.

3.2 Aspectos de Implementação da ETArch

Esta seção tem como objetivo apresentar uma das principais implementações da ETArch, que se deu através do projeto ExtenDing Ofelia in BRAzil (EDOBRA). Nessa implementação foi utilizada a linguagem Java, o protocolo OpenFlow através do controlador Floodlight e o conceito de JAIN SLEE. A seguir serão apresentados os conceitos básicos do JAIN SLEE e como ele foi utilizado na arquitetura.

3.2.1 JAIN SLEE

O JAIN SLEE (FEMMINELLA et al., 2011) é uma Application Programming Interface (API) de linguagem de programação Java para desenvolver e implementar serviços de rede que exigem um ambiente de execução para aplicações de alta capacidade e escalabilidade. Sua especificação foi criada no Java Community Process (JCP) por vários indivíduos e empresas e foi padronizado nas Java Specification Requests (JSRs) JSR 22 e JSR 240.

Tabela 5 – Primitivas do DTS Control Protocol (DTSCP)

Fonte: (SILVA, 2013)

PRIMITIVA	**FUNÇÃO**
WORKSPACE-LOOKUP	Utilizada pelo DTSA junto ao seus Resolvers. Ocorre na presença de uma primitiva WORKSPACE-ATTACH, caso o DTSA onde a entidade está registrada não possua informações sobre este workspace. Essa operação inicia-se no nível onde o DTSA está localizado e pode ser encaminhada até o nível raiz.
DTSA-REGISTER	Responsável por registrar um DTSA. Ao registrar-se o DTSA deve incluir no DTSA Resolver Database uma entrada indicando quais são os seus resolvers. O registro é feito no Master DTSA do mesmo nível do DTSA que se registra.
WORKSPACE-ADVERTISE	Insere, remove ou atualiza o Workspace Database, indicando em qual DTSA um dado workspace está presente. Este serviço deve receber o nível que indica qual será a visibilidade do workspace. Essa operação inicia-se no Master DTSA do mesmo nível onde o DTSA está localizado e pode ser encaminhada até o nível raiz.
DTS-MESSAGE	Permite a comunicação entre diferentes DTSAs no âmbito do DTS. Caso o DTSA de origem conheça previamente o caminho até o DTSA de destino, este caminho está presente no cabeçalho da mensagem. Caso contrário, a mensagem será encaminhada para os seus Resolvers, até que um deles saiba calcular o caminho até o DTSA de destino. Caso um MTDSA do nível raiz não possa calcular o caminho até o destino então o envio desta mensagem falhará.

O Mobicents é a plataforma que possui a implementação *open source* do JAIN SLEE, que fornece um modelo de componentes e um ambiente de execução robusto para aplicações de Telecom, permitindo o desenvolvimento de aplicações de voz, vídeo e dados.

O JAIN SLEE apresenta dois conceitos principais: Resource Adapter (RA) e Service Building Block (SBB). O RA é responsável por abstrair os protocolos de rede, transformando suas primitivas em serviços. O SBB é responsável por capturar os serviços de cada RA e lançá-los para outros SBBs, para que recebam tratamento específico.

A Figura 7 apresenta o DTSA desenvolvido no projeto EDOBRA. Nele foi desenvolvido dois Resource Adapters (RAs), o primeiro foi o OpenFlow RA e o segundo foi o Media Independent Handover (MIH) RA. O OpenFlow RA tem objetivo de abstrair o protocolo OpenFlow através da lib do Floodlight e o MIH RA tem objetivo de realizar a comunicação

entre o DTSA e a rede sem fio.

Como pode ser observado na Figura 7, além dos dois RAs foram desenvolvidos quatro SBBs, representados na cor verde. O NEConnector é apresentado conceitualmente como uma camada genérica entre a rede e o DTSA, acima do NEConnector estão os SBBs responsáveis por capturar os eventos lançados pelo próprio NEConnector. Cada um desses SBBs são descritos a seguir:

- EntityManager: Responsável por capturar primitivas relacionadas ao registro e remoção de entidades. Depois que as primitivas são tratadas o EntityManager reponde para o NEConnector informando se a operação solicitada foi realizada com sucesso.

- WorkspaceManager: Responsável por capturar primitivas relacionadas a criação e manutenção de um workspace.

- MobilityManager: Responsável por capturar primitivas lançadas pelo pelo MIH RA. Tais primitivas são lançadas quando uma entidade faz *handover* para um novo AP, MIH RA receberá a primitiva e lançará ao NEConnector, que então lançará ao MobilityManager.

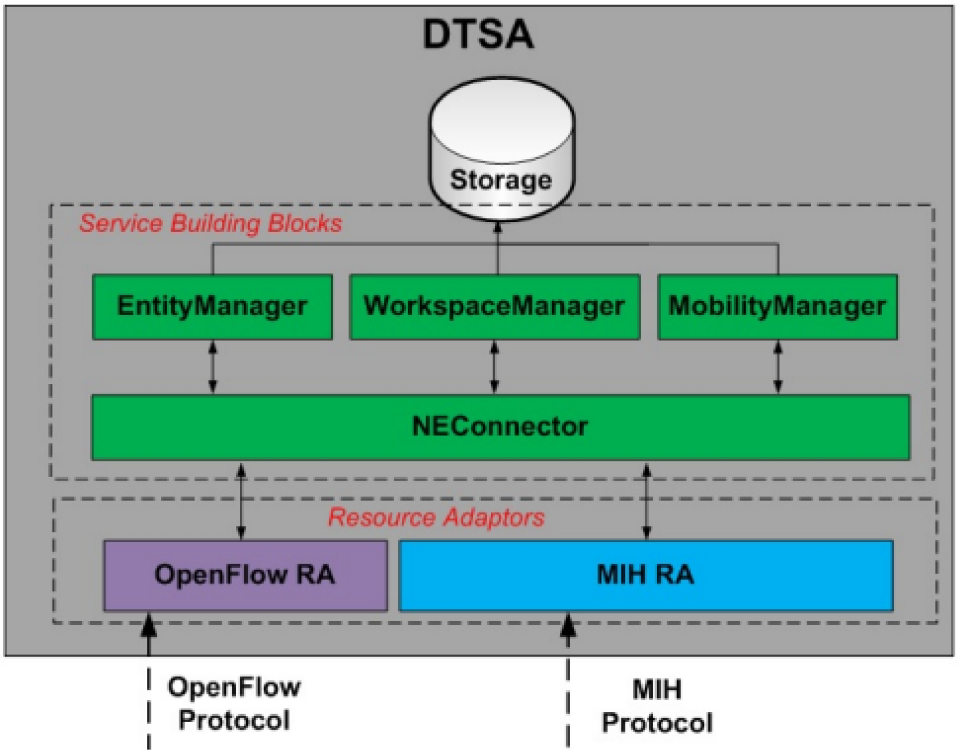

Figura 7 – DTSA do Projeto EDOBRA

Fonte: (SILVA, 2013)

Todos os Service Building Blocks (SBBs) são responsáveis por manipular informações que são armazenadas no *Storage* do DTSA, que é o banco de dados que armazena informações das entidades, workspaces, entre outras.

O JAIN SLEE foi utilizado no projeto da arquitetura com objetivo de que ela pudesse suportar novos serviços a partir da criação de novos SBBs.

Capítulo 4

Mecanismos de Autenticação e Controle de Acesso para a Arquitetura ETArch

4.1 Introdução

O surgimento de SDN na área de redes de computadores modifica a maneira como diversos aspectos de comunicação de dados podem ser tratados, a exemplo da possibilidade de alguns tópicos relacionados a segurança serem tratados no plano de controle. Deste modo, o presente trabalho busca explorar essas novas possibilidades, propondo mecanismos de autenticação e controle de acesso no plano de controle na arquitetura ETArch.

Atualmente a arquitetura ETArch não possui nenhum mecanismo de segurança implementado, isto é, toda comunicação que acontece na arquitetura até o momento é acessível a qualquer entidade conectada na rede. Por exemplo, uma aplicação de chat cria um *workspace* para que determinadas entidades possam se comunicar, porém não existe nenhum mecanismo que controle quem pode participar ou não desse canal de comunicação. Desse modo as entidades que participam desse canal de comunicação não possuem privacidade.

Na Figura 8 é possível observar a estrutura e componentes do DTSA. O DTSA dispõe de dois *Resource Adaptors* (RAs) para o tratamento dos protocolos do plano de controle: o OpenFlow RA e *Media Independent Handover* (MIH) RA. O OpenFlow RA tem o objetivo de abstrair o protocolo OpenFlow enquanto MIH RA tem o objetivo de abstrair a comunicação entre o DTSA e a rede sem fio para aspectos de controle e *handover* das entidades móveis. Cada RA transforma primitivas do protocolo em eventos que são tratados pelos serviços do DTSA.

Além dos RAs, a primeira implementação do DTSA possui três outros serviços representados na Figura 8 com cor laranja. O NEConnector, é uma camada genérica de comunicação entre a rede e o DTSA. Os protocolos que se comunicam com o NEConnec-

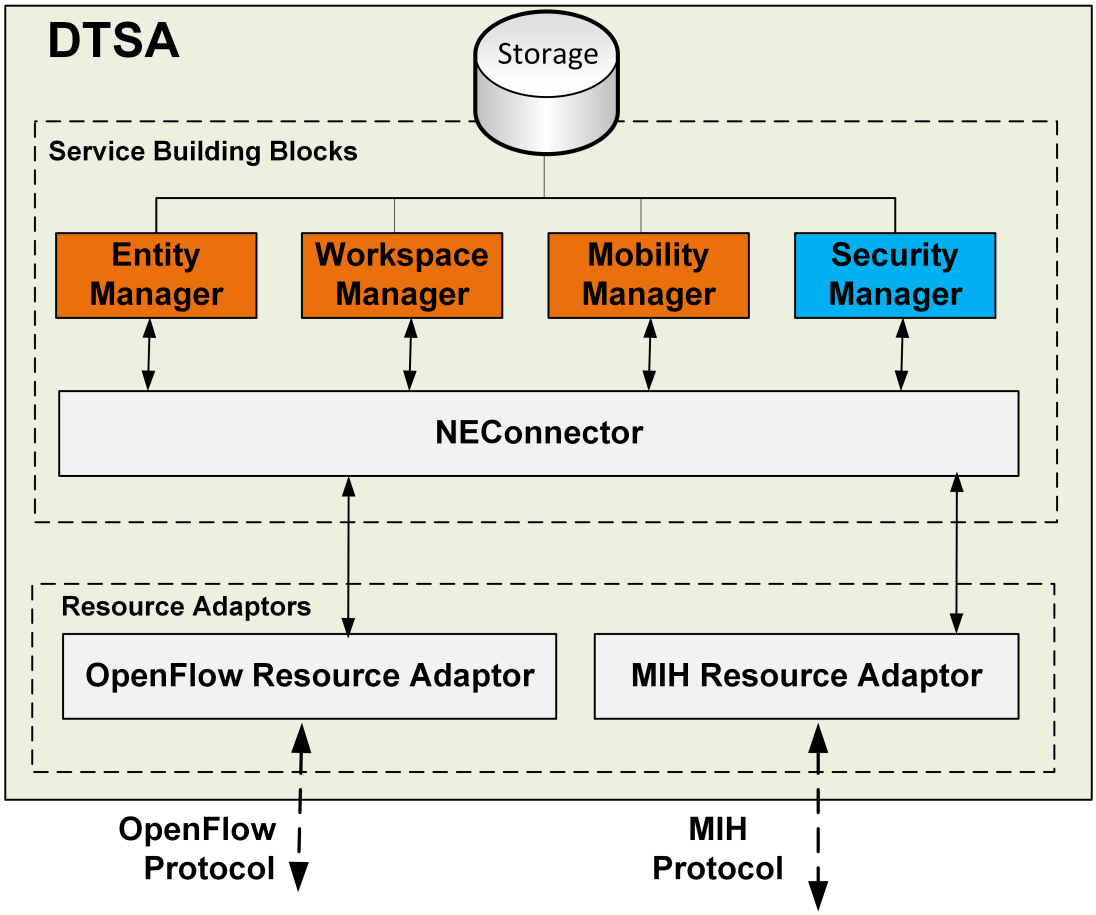

Figura 8 – Descrição dos componentes do DTSA

tor são o OpenFlow e MIH. Futuramente novos protocolos podem ser adicionados e os serviços criados acima do NEConnector podem ser reaproveitados.

Uma entidade, para se registrar, envia uma primitiva ENTITY_REGISTER, através do elemento de rede, nesse caso, um *switch* OpenFlow ao qual está ligada fisicamente. No elemento de rede, essa primitiva é encapsulada em um PACKET_IN e enviada ao controlador (DTSA). Ao chegar no DTSA, o RA OpenFlow transforma o PACKET_IN em um serviço e o dispara a uma camada superior, nesse caso, o NEConnector é responsável por capturar todas as mensagens do tipo PACKET_IN lançados pelo RA OpenFlow.

O NEConnector analisa todas as mensagens PACKET_IN recebidas, objetivando resgatar as primitivas ETArch nelas encapsuladas. Após realizar essa análise, o NEConnector lança essas primitivas para que outros serviços possam capturá-las. Primitivas relacionadas às entidades são capturadas pelo EntityManager. Tudo que está relacionado com um *Workspace* é capturado pelo WorkspaceManager. E todas as primitivas que envolvem mobilidade são capturadas pelo MobilityManager.

Na versão atual da ETArch, qualquer entidade pode se registrar e participar de um *workspace*, nesse caso, o controlador não é capaz de identificar quais entidades são confiáveis e quais são maliciosas.Dessa maneira, esse trabalho objetiva aumentar a segurança

da arquitetura ETArch através de um novo serviço que será implantado no DTSA para garantir a execução segura de aplicações, sem que uma aplicação ou entidade interfira em outras ou execute ações proibidas na rede.

A Figura 8 apresenta na cor azul um novo SBB, o SecurityManager, que é responsável por tratar a segurança de forma genérica no DTSA, onde ficam localizados os mecanismos de autenticação e autorização. Novos mecanismos podem ser adicionados futuramente, como por exemplo, detecção de intrusões. Esse novo serviço será responsável por capturar os serviços lançados pelo NEConnector, depois de passar por um processo de validação e avaliação serão encaminhados para os SBBs que são responsáveis por tais serviços.

4.1.1 Autenticação

No estado atual da arquitetura, qualquer entidade pode se registrar em um DTSA informando apenas o seu nome. Isso implica em diversos aspectos que prejudicam a segurança da ETArch.

Fica claro que nesse momento a arquitetura não possui a capacidade de garantir que uma entidade que se conecte a ela seja autêntica, por conseguinte, ela abre espaço para uma grande gama de vulnerabilidades. Com essas vulnerabilidades pode acontecer o roubo de identidade, isto é, uma entidade tenta se passar por outra, com o objetivo de obter vantagens indevidas.

Este trabalho busca resolver esse problema verificando a identidade da entidade no momento em que ela solicita o seu registro. Esta autenticação irá depender de informações de autenticação que são passadas pela entidade no momento da sua solicitação.

O mecanismo de autenticação proposto foi elaborado com base na recomendação X.811 (UNION, 1995a), porém aplicado em um novo cenário onde é utilizado SDN e detalhes da nova arquitetura.

Na ETArch, o conceito de entidade é genérico e pode ser entendido como tudo que possui capacidade de se comunicar, sendo assim, a autenticação precisa tratar as entidades de uma forma genérica, onde é possível adaptar-se a diversos cenários. Por exemplo, esse mecanismo precisa autenticar aplicações, elementos de rede, sensores, *smartphones*, usuários, entre outros.

Elaborar o mecanismo de forma genérica é um grande desafio, pois é preciso pensar não só nos tipos de entidades que existem hoje, mas também nas que possam surgir futuramente, ou seja, o mecanismo implementado deve possuir a capacidade de evoluir de maneira rápida e fácil para suportar novas entidades.

A Figura 9 apresenta um cenário em que diversas entidades estão enviando informações de autenticação para o DTSA. Essas informações variam de acordo com o tipo de entidade, ou seja, as informações que um sensor irá enviar para realizar a autenticação são diferentes de uma aplicação ou um *smartphone*.

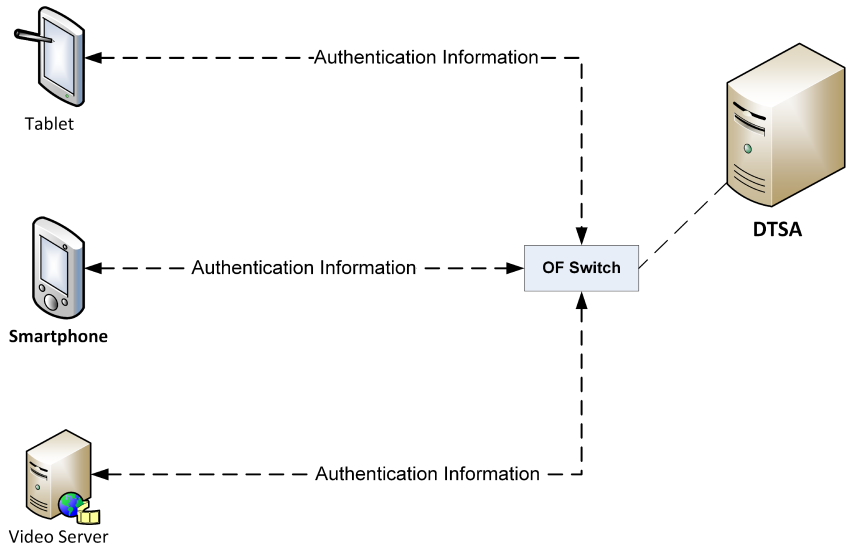

Figura 9 – Entidades enviando informações de autenticação para o DTSA

A Figura 10 apresenta uma visão do novo SBB criado. O Verificador possui funções necessárias para realizar trocas de informações de autenticação e possui a responsabilidade de verificar a identidade de cada entidade e realizar a autenticação.

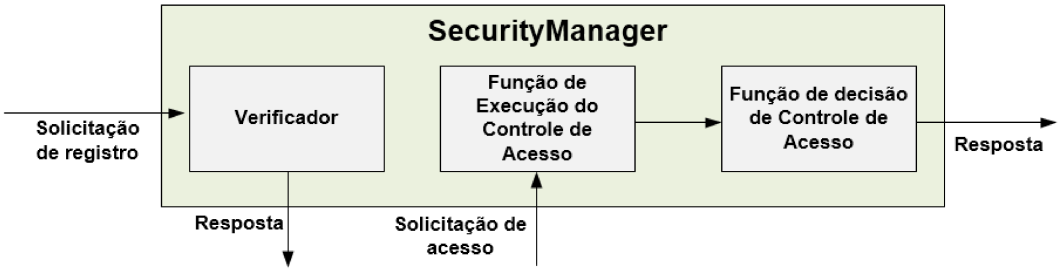

Figura 10 – Componentes existentes no SecurityManager.

Os componentes Função de Execução do Controle de Acesso (AEF) e Função de Decisão de Controle de Acesso(ADF) vistos na Figura 10 serão explorados na próxima seção, pois são responsáveis pelo controle de acesso.

4.1.2 Controle de Acesso

Atualmente, nenhuma verificação é feita na arquitetura ETArch no instante em que uma entidade solicita participar de workspace, ou seja, nenhuma entidade que participa de um canal de comunicação possui privacidade para transmitir ou receber informações.

Existe outro fator que prejudica a falta de controle de acesso hoje. No momento em que uma entidade cria um workspace, a única informação que ela passa é o nome que será dado ao workspace, logo, é impossível determinar quais entidades possuem permissão

para acessá-lo. Desta forma, o workspace criado fica disponível para qualquer entidade que deseja participar daquele domínio de comunicação.

O controle de acesso foi criado baseado na recomendação X.812 (UNION, 1995b), que é aplicado no plano de controle da arquitetura ETArch para verificar quais entidades possuem permissão para se conectar a determinado *Workspaces*.

Na Figura 10, é possível identificar dois componentes de grande importância para o mecanismo de controle de acesso. O primeiro é o AEF, uma função especializada que faz parte do caminho de acesso entre as entidades e um *Workspace* em cada solicitação de acesso.O segundo é o ADF, responsável pelas decisões de controle de acesso com base em políticas de segurança, informações de controle de acesso fornecidas por entidades e informações de contexto. Por exemplo, uma entidade solicita acesso a um *Workspace* chamado W1, o AEF recebe essa solicitação e encaminha para o ADF que é responsável por decidir se a entidade pode ou não acessar aquele *Workspace*, depois de tomar a decisão, o *SecurityManager* envia a sua decisão.

É possível ter uma visão detalhada do ADF na Figura 11, ele é responsável por receber as requisições de controle de acesso e tomar uma decisão baseada em políticas de controle de acesso. As informações que colaboram para a tomada de decisão são os *workspaces*, entidades, informações de contexto da requisição e decisões anteriores armazenadas no ADF.

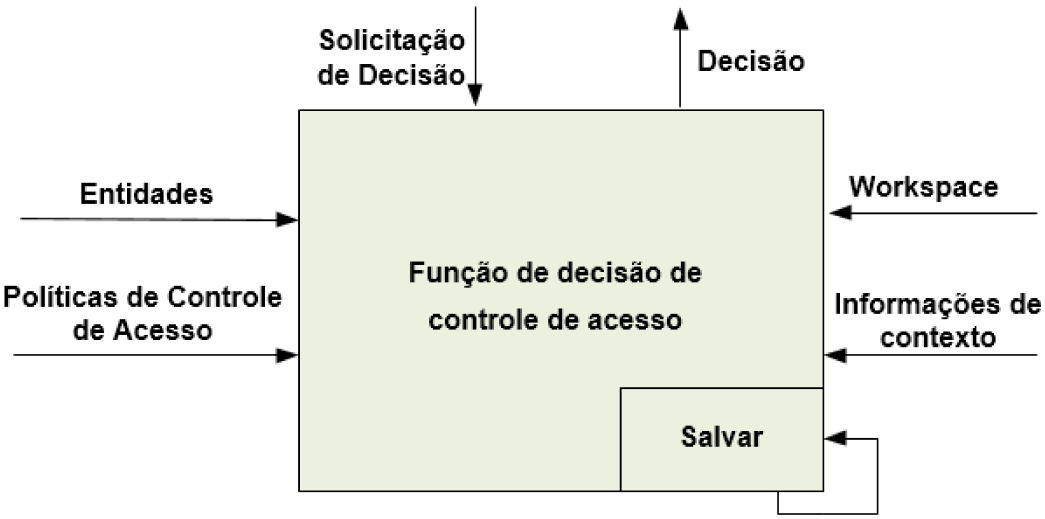

Figura 11 – Função responsável pelo controle de acesso

Para adicionar os mecanismos de autenticação e controle de acesso descritos é necessário complementar e alterar algumas funcionalidades nas primitivas do protocolo ETCP. Para o mecanismo de autenticação essa mudança ocorreu na primitiva ENTITY-REGISTER e para o controle de acesso a modificação aconteceu na primitiva WORKSPACE-CREATE e WORKSPACE-MODIFY.

Para que uma aplicação possa utilizar este protocolo, essas funções são disponibilizadas através de uma API *(Application Programming Interface)*. A Tabela 6 descreve as novas funções adicionadas em cada primitiva.

Tabela 6 – Funções adicionadas ao ETCP

Primitiva	**Função**	**Descrição**
ENTITY-REGISTER	accessInformation	Informação utilizada no momento do registro para realizar a autenticação.
	authenticationType	Tipo de autenticação que pode ser associada à entidade no momento do registro.
WORKSPACE-CREATE	accessControlList	Informação utilizada no momento em que o Workspace é criado para informar quais entidades podem acessá-lo.
WORKSPACE-MODIFY	modifyInformation	Informação utilizada para realizar a alteração no Workspace.
	modifyType	Tipo de alteração que será realizada no Workspace.

Embora o desempenho seja uma questão importante, assim como ocorreu com a Internet em seus primórdios, não faz parte do escopo deste trabalho dirimir questões relativas a essas características no plano de controle do DTS.

4.2 Implementação

Essa Seção tem como objetivo apresentar alguns aspectos da implementação descritos conceitualmente na Seção 4.1. Inicialmente serão retratados os aspectos relacionados a autenticação que estão diretamente envolvidos com a primitiva ENTITY-REGISTER. Em seguida, será apresentado o mecanismo de controle de acesso que trabalha com duas primitivas para seu funcionamento, o WORKSPACE-CREATE e WORKSPACE-ATTACH.

4.2.1 Autenticação

O mecanismo de autenticação já foi descrito na Seção 4.1.1, entretanto não foram mostrados as trocas de mensagens e como o DTSA lida com as informações fornecidas pelas entidades.

Como pode-se observar na Seção 4.1.1, o mecanismo foi criado de uma forma genérica, possuindo a capacidade de evoluir o seu método de autenticação com o passar do tempo. Como prova de conceito foi utilizada uma solução descrita na recomendação x.509 da ITU-T que define uma plataforma de autenticação baseada em certificados, os quais são expedidos por uma entidade certificadora (CA).

A autenticação nesse caso é realizada por meio de troca de informações criptografadas a partir de uma chave pública das duas partes envolvidas, isto é, se uma das entidades não for quem afirma ser, ela não conseguirá validar a entidade.

A criação de uma Public Key Infrastructure (PKI) foge do escopo desse trabalho. Para validar o mecanismo criado através de certificado foi criada uma chave privada e um certificado para o DTSA e todas entidades que desejam autenticar no DTSA precisam possuir um certificado válido. Todavia, o procedimento de criar as chaves e gerar certificados válidos não será descrito nesse trabalho.

A Figura 12 apresenta as mensagens que são trocadas entre a entidade e o DTSA no momento do seu registro. A seguir será descrita a função de cada mensagem:

1. Entidade solicita um ENTITY_REGISTER, enviando as informações necessárias para que o DTSA possa garantir que a entidade é autêntica. Nesse caso será enviado o título, o tipo de autenticação e a informação de autenticação;

2. ENTITY_REGISTER chega até o NE no qual a entidade está ligada fisicamente, logo após a sua chegada o NE encapsula o ENTITY_REGISTER em um PACKET_IN e lança para o DTSA;

3. Após fazer realizar os procedimentos necessário para validar a entidade no DTSA, a resposta é encapsulada em PACKET_OUT e a envia ao NE da entidade que solicitou o registro.

4. NE envia a resposta para entidade, informando se ela conseguiu efetuar o seu registro ou não.

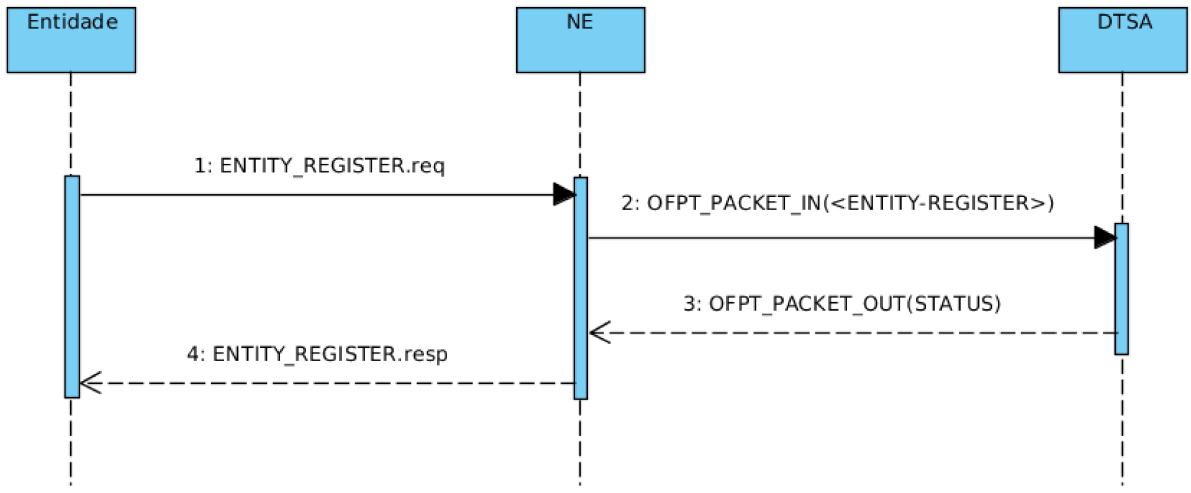

Figura 12 – Mensagens de registro de uma entidade

A Figura 13 apresenta as mensagens que são trocadas no DTSA no momento em que uma entidade solicita o seu registro. O NEConnector encaminha a solicitação para o

SecuriyManager, que irá verificar todas as informações que foram fornecidas pela entidade para realizar a autenticação. Após verificar as informações o SecuriyManager envia a resposta para o NEConnector, se as informações fornecidas forem válidas, o NEConnector possui permissão para encaminhar a solicitação para o EntityManager onde o registro da entidade é realizado.

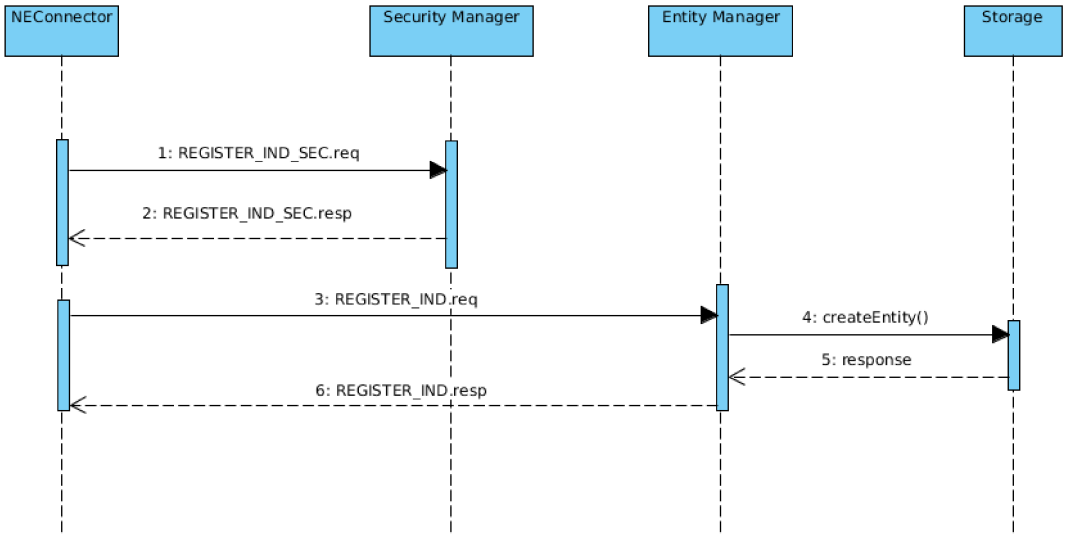

Figura 13 – Mensagens de registro de uma entidade no DTSA

4.2.2 Controle de Acesso

O mecanismo de controle de acesso já foi descrito na Seção 4.1.2, entretanto as mensagens que são trocadas para realizar todo o processo ainda não foram apresentadas. Essa Seção pretende demonstrar as mensagens envolvidas quando uma entidade solicita criar, alterar e participar de um workspace.

Para que uma entidade possa criar um workspace necessariamente ela já precisa estar registrada no DTSA. A Figura 14 apresenta as mensagens no momento em que uma entidade solicita criar um workspace. A seguir será descrita a função de cada mensagem:

1. Entidade solicita um WORKSPACE_CREATE, enviando as informações necessárias para que o DTSA possa criar o workspace solicitado. Nesse caso será enviado o seu título e uma Access Control List (ACL) contendo as entidades que possuem permissão para acessar o workspace que está sendo criado;

2. WORKSPACE_CREATE chega até o NE no qual a entidade está ligada fisicamente, logo após a sua chegada o NE encapsula o WORKSPACE_CREATE em um PACKET_IN e lança para o DTSA;

3. O DTSA envia um FLOW_MOD para que as regras sejam configuradas nos *switches* OpenFlow.

4. Após realizar os procedimentos necessários para criar o workspace, a resposta é encapsulada em PACKET_OUT e a envia ao NE da entidade que solicitou criar o workspace.

5. NE envia a resposta para entidade, informando se sua solicitação de criar o workspace foi atendida.

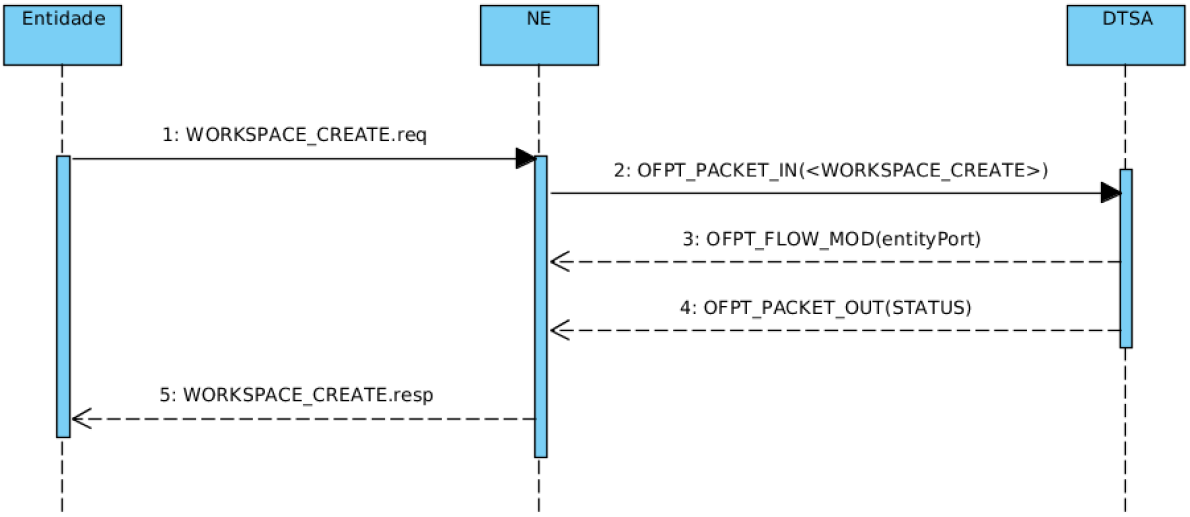

Figura 14 – Mensagens ao criar um workspace

A Figura 15 apresenta as mensagens que são trocadas dentro do DTSA no momento em que uma entidade solicita a criação de um workspace. O NEConnector encaminha a solicitação para o SecuriyManager, onde serão realizadas as verificações das informações fornecidas pela entidade, um exemplo de informação que uma entidade pode enviar é uma lista com os nomes das entidades que possuem permissão para acessar o workspace que está sendo criado. Após essa verificação uma resposta será enviada para o NEConnector, onde ele decide se irá encaminhar ou não a solicitação para o WorkspaceManager para realizar a crição do worksspace.

Semelhante o procedimento descrito na Figura 14 uma entidade já registrada pode solicitar participar de workspace existente usando a primitiva WORKSPACE_ATTACH, nenhuma informação foi adicionada a essa primitiva durante a implementação deste projeto. Apesar disso, a maneira como ela é tratada no DTSA foi totalmente modificada para verificar se a entidade possui permissão para acessar o workspace solicitado.

A Figura 16 apresenta a troca de mensagens realizadas no momento do WORKSPACE_ATTACH, os passos nesse momento não serão descritos, pois são semelhantes aos que acontecem no WORKSPACE_CREATE.

A única diferença entre as primitivas WORKSPACE_ATTACH e WORKSPACE_CREATE, é que no passo 1 da Figura 16 a primitiva exige apenas que a entidade passe o Título do workspace que ela deseja se conectar.

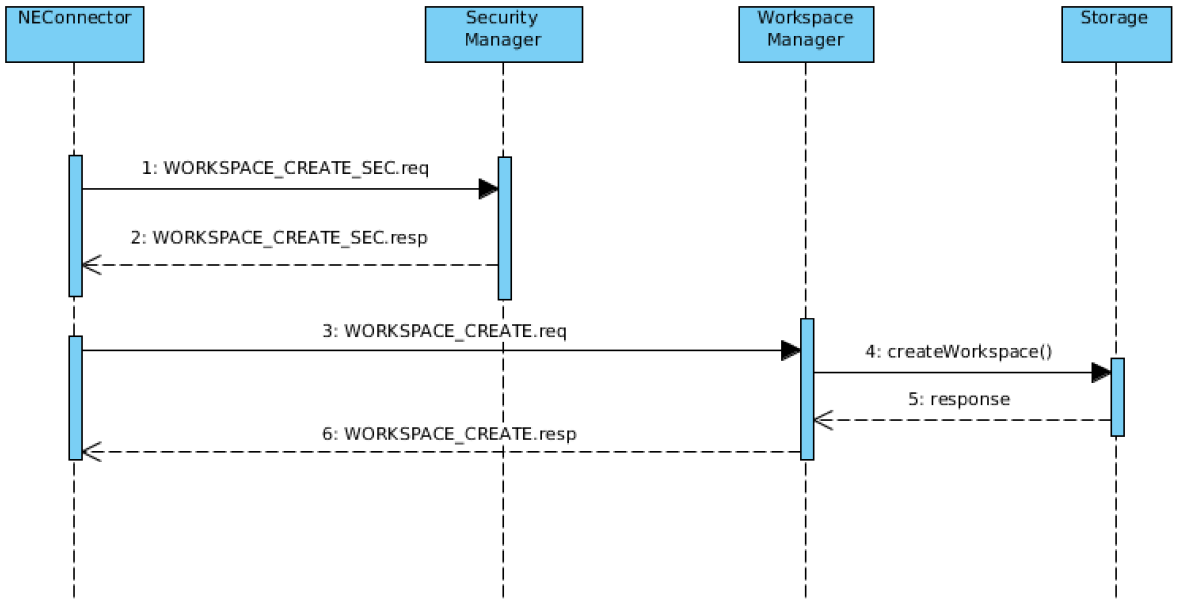

Figura 15 – Mensagens ao criar um workspace no DTSA

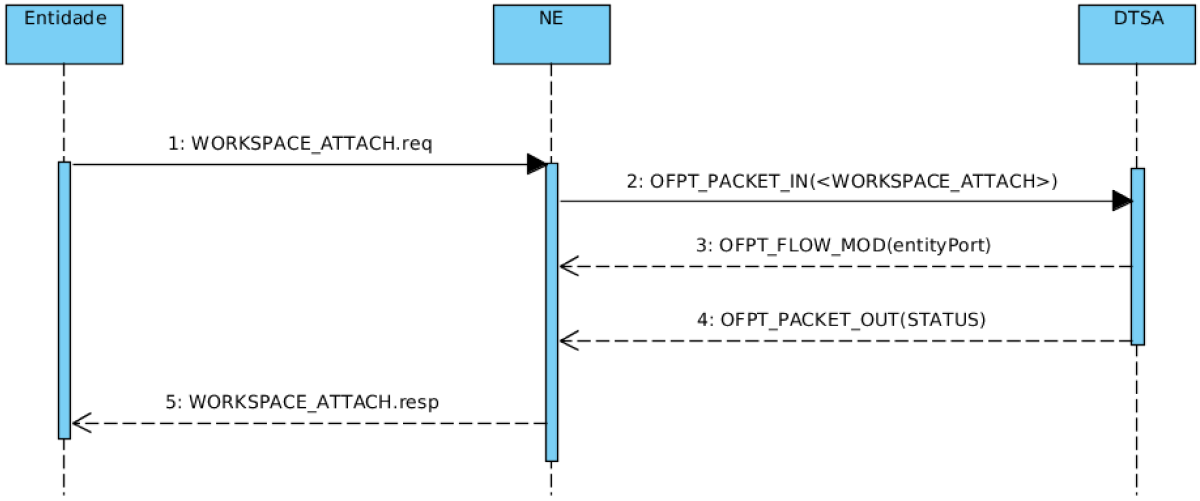

Figura 16 – Mensagens ao conectar em um workspace

A Figura 17 exibe as mensagens que são trocadas dentro do DTSA no momento em que uma entidade solicita participar de um workspace. O NEConnector encaminha a solicitação para o SecuriyManager, onde serão realizadas as verificações para garantir que somente as entidades que possuam as permissões necessárias possam acessar o canal de comunicação. Após realizar essa verificação o SecurityManager envia uma mensagem para o NEConnector informando se a entidade pode ou não participar do workspace solicitado, se a entidade possuir permissão o NEConnector encaminha a mensagem para o WorkspaceManager para que ela seja incluída no canal de comunicação, caso contrário uma mensagem é enviada à entidade informando que sua solicitação foi negada.

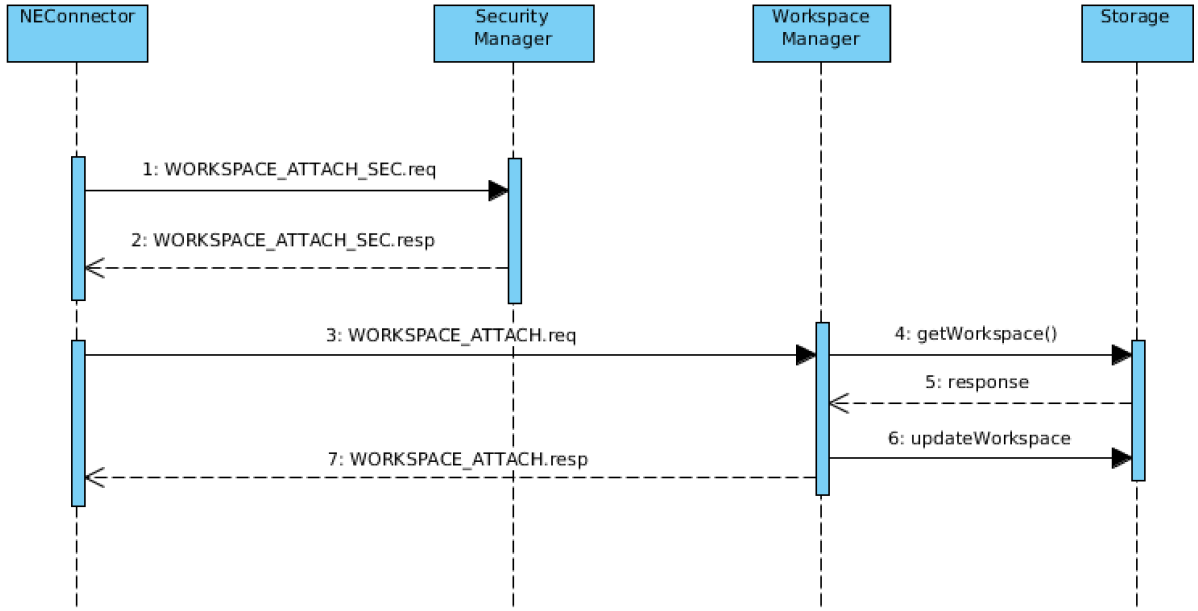

Figura 17 – Mensagens ao conectar em um workspace no DTSA

Capítulo 5

Experimentos e Análise dos Resultados

Como descrito na Seção 4.1, a ETArch não possuía um mecanismo de segurança definido. Por isto, este trabalho buscou elaborar e implementar dois mecanismos que contribuíssem para a segurança da arquitetura, o primeiro mecanismo possui a função de realizar a autenticação e o segundo o controle de acesso.

Os mecanismos de autenticação e controle de acesso são empregados em um novo contexto. A autenticação foi elaborada levando em consideração a diversidade de entidades possíveis. Esse mecanismo é aplicado sempre que uma entidade tenta se registrar. O controle de acesso é definido por uma entidade responsável que criou o *workspace* e são aplicados sempre que uma entidade deseja se juntar a determinado canal de comunicação.

Inicialmente, será descrito o método utilizado para validar a hipótese apresentada na Seção 1.3. Na Seção 5.2, serão descritos os experimentos realizados, demostrando o ambiente utilizado, topologias, tipos de máquinas etc. A seção 5.3 apresentará uma breve avaliação sobre os resultados obtidos nos experimentos, mostrando o custo-benefício, capacidades e limitações deste trabalho.

5.1 Método para a Avaliação

Objetivando validar de forma efetiva os mecanismos de autenticação e controle de acesso implementados neste trabalho, foram criados cenários de teste onde tais mecanismos são consumidos por uma aplicação de chat. Os experimentos descritos na Seção 5.2 envolvem três cenários de teste.

O primeiro demonstra o mecanismo de autenticação, no qual uma entidade consegue se registrar corretamente utilizando um certificado digital. A maneira como este certificado é criado ou gerido foi abstraído, pois não é o foco deste trabalho. Também foi demonstrado quando o pedido de registro da entidade é recusado. Neste caso essa falha aconteceu porque o certificado está com a validade expirada.

O segundo demonstra o mecanismo de controle de acesso, em que uma entidade cria um *workspace* e informa uma *access control list* (ACL) com os títulos de entidades que

podem acessá-lo. Em um segundo momento outra entidade tentar acessar o *workspace* criado e tem sua permissão concedida. A partir desse momento, esta entidade participa do canal de comunicação criado. Outra demonstração realizada nesse experimento é uma entidade solicitando acesso a um determinado *workspace*,no entanto, tem o seu pedido é negado.

Por último é apresentado um cenário com uma topologia contendo quinze *hosts* em que é medido o tempo de autenticação e controle de acesso para cada entidade que se conecta em um determinado *workspace*. Este mesmo cenário também foi realizado em um contexto onde tais mecanismos de segurança eram inexistentes. Com objetivo de demonstrar o custo benefício para arquitetura com a implementação deste trabalho.

Para melhor representar os experimentos, são apresentadas imagens das topologias utilizadas em cada experimento, além de outras imagens demonstrando o funcionamento dos mecanismos criados.

5.2 Experimentos

Os experimentos realizados pretendem demonstrar o funcionamento dos mecanismos de autenticação e controle de acesso. A ideia é validar o uso de tais mecanismos no plano de controle. Serão detalhados os cenários, topologias e ambiente computacional utilizados.

5.2.1 Mecanismo de autenticação

A autenticação é realizada no momento em que uma entidade solicita o seu registro enviando a primitiva ENTITY_REGISTER através do *switch OpenFlow* na qual a entidade está ligada fisicamente. No *switch OpenFlow*, essa primitiva é encapsulada em um PACKET_IN e enviada ao controlador (DTSA) do *switch OpenFlow*. Ao chegar no DTSA, o OpenFlow RA(através do Floodlight) transforma o PACKET_IN em um serviço e o lança para ser capturado em um nível superior.

O serviço que foi lançado é capturado pelo SBB NEConnector, que analisa todas as mensagens recebidas, com o objetivo de retirar a primitiva ETArch que está encapsulada no PACKET_IN capturado. Após realizar essa análise o NEConnector lança tal primitiva para que outros SBBs possam capturá-la. Quando a primitiva recebe o tratamento adequando a sua resposta é enviada através de um PACKET_OUT para o *switch OpenFlow*. Os detalhes da estrutura do DTSA podem ser observados na Figura 8.

Para exemplificar melhor a troca de pacotes é possível observar na Figura 18 o pacote com o protocolo "0x880"utilizado para representar os pacotes do ETArch. Neste momento esse pacote carrega a primitiva ENTITY_REGISTER na qual será enviada ao *switch OpenFlow* em que a entidade está conectada e lá ele será encapsulado em um PACKET_IN e será enviado ao DTSA.

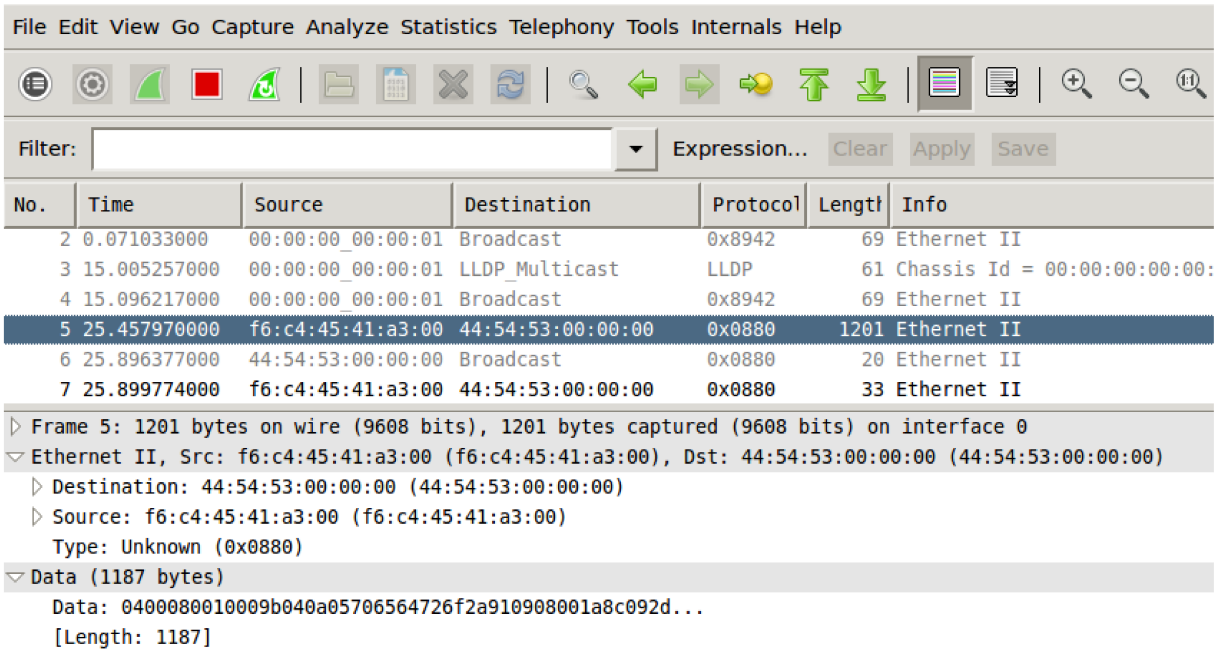

Figura 18 – Pacote sendo enviado ao DTSA pela entidade.

A Figura 19 mostra o momento em que o PACKET_IN que guarda a primitiva ETArch chega ao DTSA. Então o OpenFlow RA transforma o PACKET_IN em um serviço para que possa ser tratado pela arquitetura. A sequência de trocas de primitivas do protocolo ETCP é similar, alterando somente o conteúdo das primitivas.

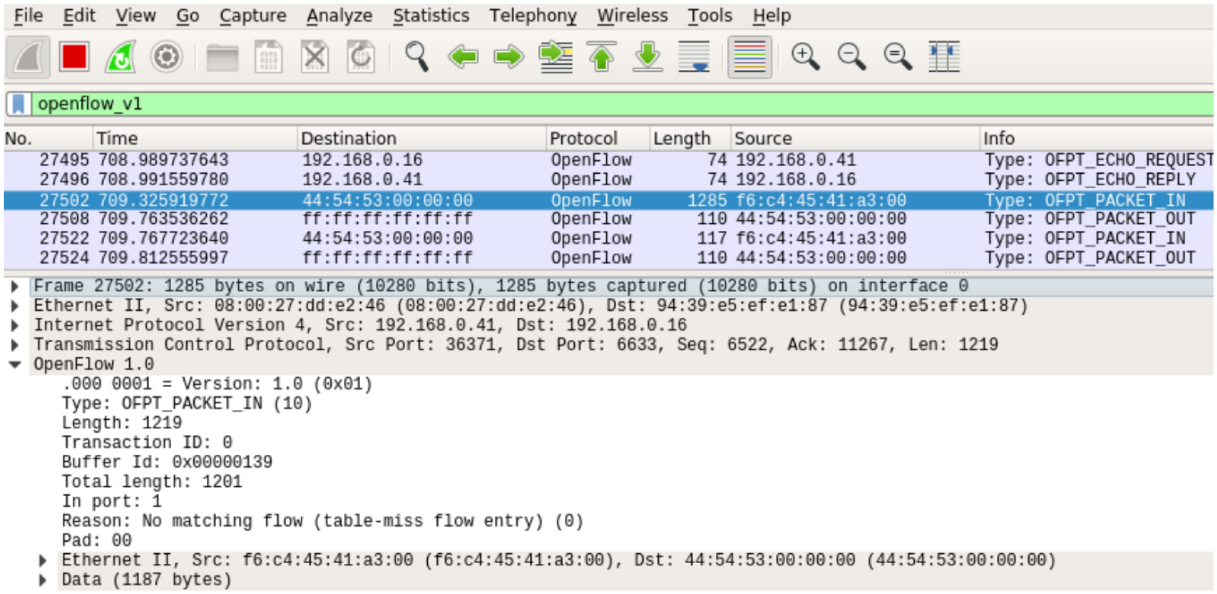

Figura 19 – Pacote sendo recebido no DTSA.

As capturas de pacotes realizadas na Figura 18 e Figura 19 se deram através do experimento representado na Figura 20 onde o Client1 envia a primitiva ENTITY_REGISTER

carregando a informação de autentificação.

Assim que o NEConnector captura tal primitiva e a lança, o SBB SecurityManager captura e verifica se as informações fornecidas pela entidade estão corretas, nesse caso a verificação se deu através de um certificado digital. Após validar o certificado o SecurityManager garante que a entidade é autêntica e permite que o seu registro seja criado, ao passo que ele lança um serviço que é capturado pelo SBB EntityManager. O SBB EntityManager realiza o registro da entidade e envia uma resposta encapsulada em um PACKET_OUT pelo OpenFlow RA é enviada ao *switch OpenFlow* onde a entidade está conectada.

No experimento realizado o Client1 consegue se conectar com sucesso, pois ele tinha posse de um certificado válido. Então o SecuriyManager consegue garantir que o Client1 é autentico. Já a requisição do Client3 foi negada pelo SecurityManager, pois sua solicitação foi realizada utilizando um certificado que continha uma validade expirada. Neste exemplo foi utilizado certificado digital para autenticar as entidades, contudo o mecanismo proposto é extensível, sendo possível incluir novos métodos de autenticação.

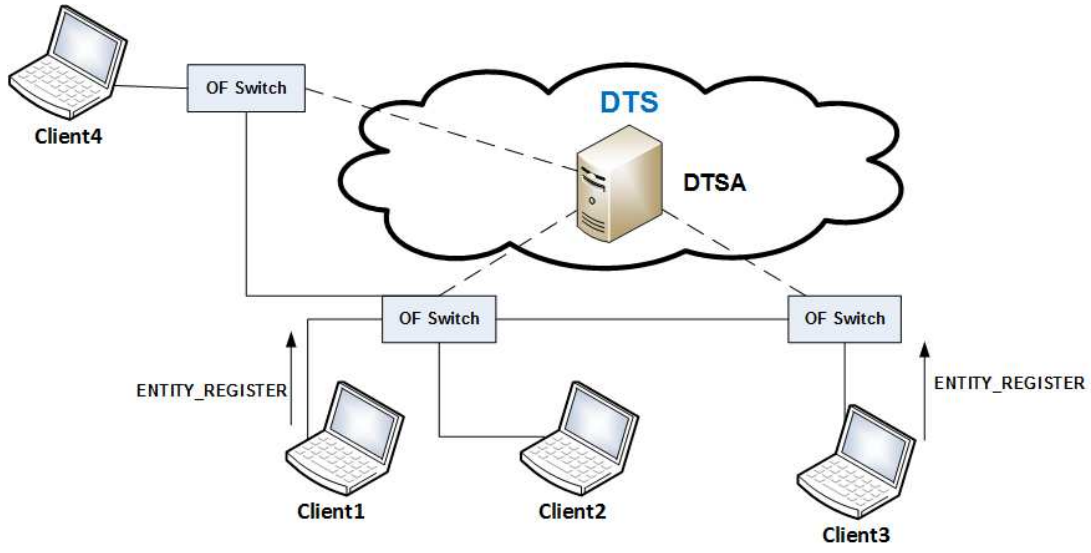

Figura 20 – Solicitação de registro do Client1 e Client2

5.2.2 Mecanismo de controle de acesso

O controle de acesso implementado na arquitetura ETArch acontece no momento em que uma entidade solicita participar de um determinado *workspace*, ou seja, assim que ela envia a primitiva WORKSPACE_ATTACH, o SecurityManager é responsável por verificar se ela possui permissão para participar do domínio de comunicação solicitado. A troca de mensagens acontece de maneira semelhante a descrita na Seção 5.2.1. No entanto as primitivas carregam conteúdos diferentes.

A Figura 21 mostra a topologia utilizada no experimento realizado com o mecanismo de controle de acesso. Todas as entidades apresentadas (Client1, Client2, Client3, Client4) estão devidamente registradas, ou seja, autenticadas na ETArch. A princípio, o Client1 envia a primitiva WORKSPACE_CREATE que possui a lista de entidades que possuem permissão para acessar o *workspace* que será criado. No caso desse experimento, a lista de permissão é composta por Client2 e Client3.

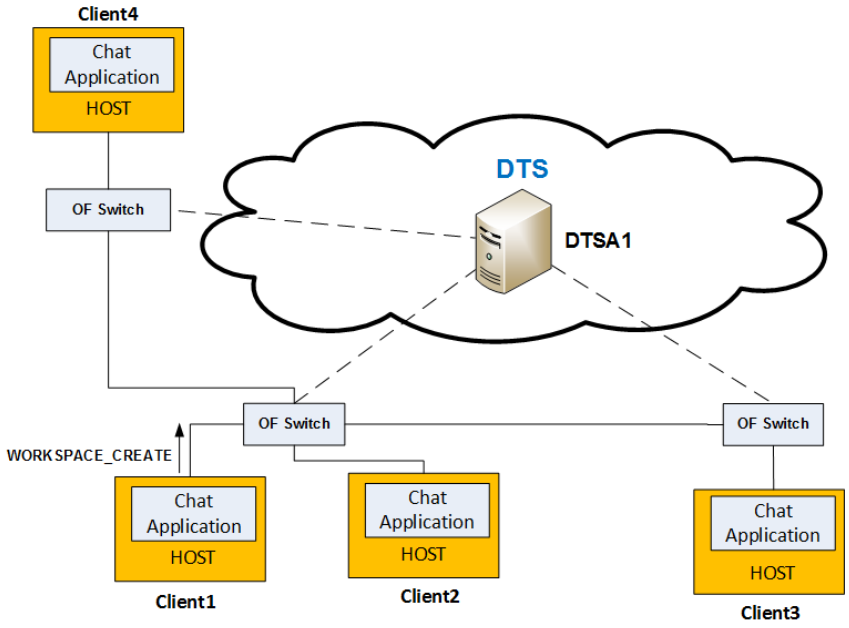

Figura 21 – Solicitação para criar um workspace do Client1

O SecurityManager fica encarregado de receber tal primitiva enviada pelo Client1 e cuidar da lista de entidades que possuem permissão para acessar o *workspace*. Depois de verificar essa solicitação, ele dispara um serviço que é capturado pelo SBB *Workspace Manager*, o qual é responsável por criar o *workspace*. Após a criação de uma mensagem de resposta, a mesma é enviada através de um PACKET_OUT para o *switch OpenFlow*.

Assim que o *workspace* é criado automaticamente, a entidade que solicitou o WORKSPACE_CREATE faz parte desse canal de comunicação. Neste experimento em especial, está sendo utilizada uma aplicação de chat, na qual todos que estão participando do *Workspace* conseguem receber mensagens que estão sendo trocadas.

É possível observar o *Workspace* criado através da linha tracejada em vermelho na Figura 22. Outro detalhe importante nesta imagem é a solicitação enviada pelo Client2 de WORKSPACE_ATTACH para participar do *Workspace* criado pelo Client1. Assim que o Security Manager recebe tal solicitação ele irá verificar se a entidade que está solicitando o acesso possui permissão, se sim ele irá disparar um serviço para o *Workspace Manager* onde a entidade solicitante será incluída no canal de comunicação, caso ele não tenha acesso, o pedido será negado.

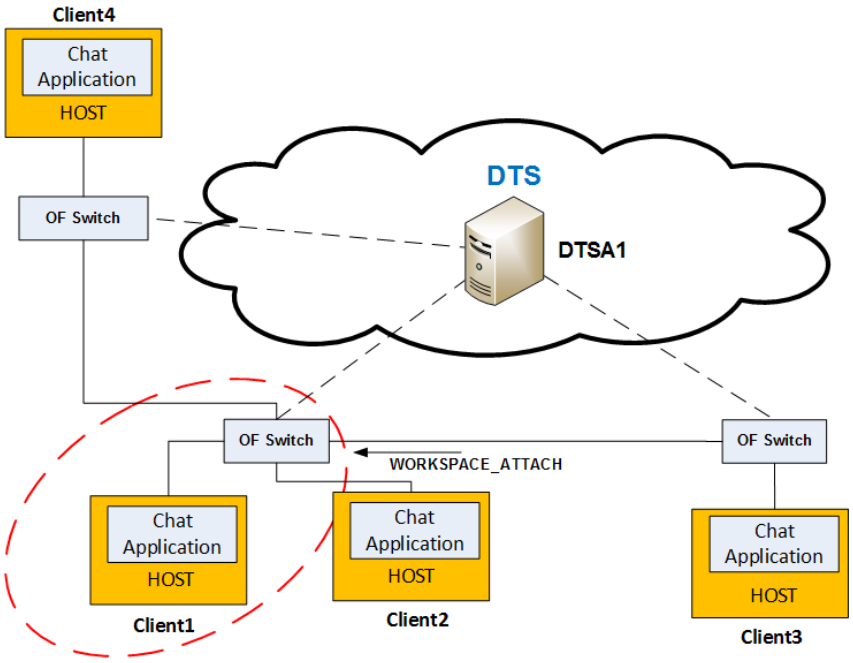

Figura 22 – Solicitação do Client2 para participar do workspace

Assim que o pedido chegou no *Security Manager* foi constatado que o Client2 possui acesso ao *Workspace* solicitado, logo, o seu pedido de acesso foi concedido com sucesso e a partir desse momento todas as mensagens que são trocadas na aplicação de chat são recebidas pelo Client1 e Client2. A Figura 23 mostra a nova estrutura do *Workspace*, e apresenta duas novas solicitações de WORKSPACE_ATTACH a do Client3 e Client4.

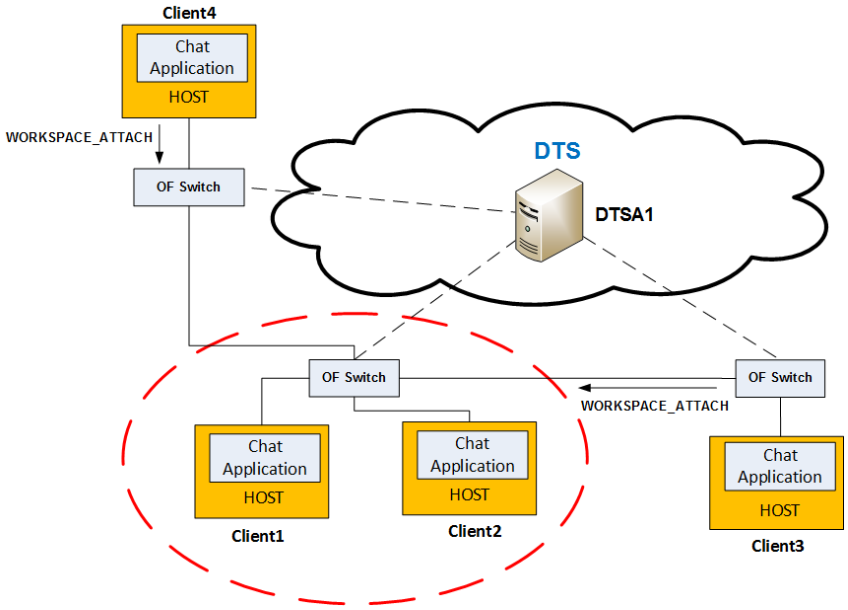

Figura 23 – Solicitação do Client3 e Client4 para participar do workspace

Com base nas solicitações o *Security Manager* verificou-se que o Client3 possui acesso ao *Workspace* solicitado, diferente do Client4 que teve o seu pedido recusado. A Figura

24 apresenta a nova estrutura do *Workspace* onde as três entidades podem se comunicar através da aplicação de chat sem ter sua conversar comprometida pelo Client4 que não possui permissão para acessar tal conteúdo.

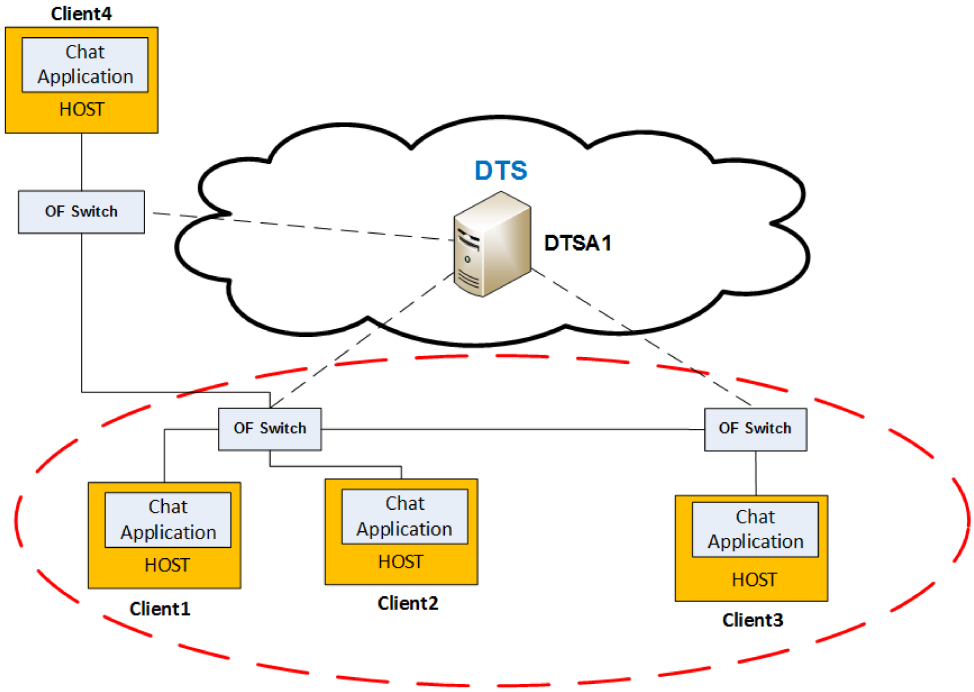

Figura 24 – Representação final do workspace criado pelo Client1

A Figura 25 apresenta o Client1, Client2 e Client3 utilizando a aplicação de chat, onde as mensagens são enviadas através do workspace W1 que foi criado pelo Client1. Todas as entidades que estão conectadas a esse canal de comunicação recebem as mensagens que são transmitidas.

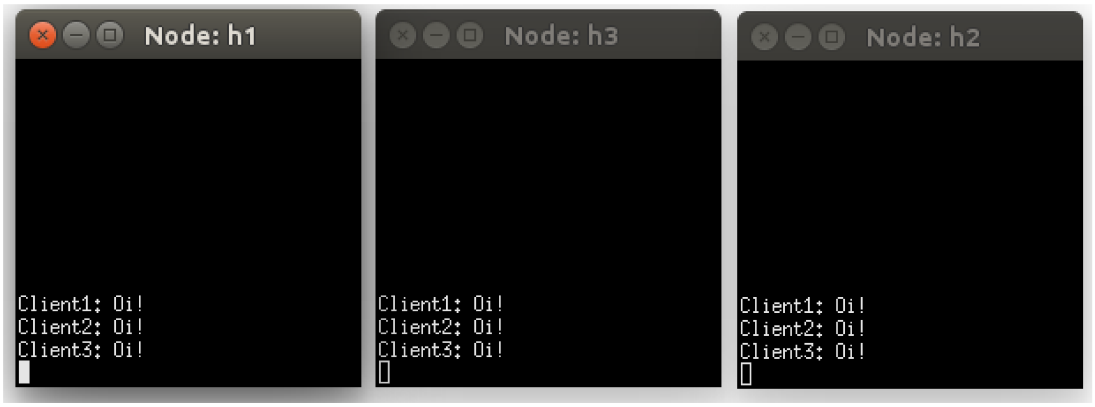

Figura 25 – Mensagens enviadas pelo Client1, Client2 e Client3.

5.2.3 ETArch com Security Manager x ETArch

Neste experimento, para o DTSA foi utilizado um computador com o sistema operacional Ubuntu 16.04.1 LTS, com o processador Intel(R) Core(TM) i5-2430M CPU @ 2.40GHz e 6 GB de memória RAM. Foi utilizado o Mininet para simular uma topologia de rede usando uma maquina virtual com sistema operacional Ubuntu 14.04 LTS com 512MB de memória RAM. Na Figura 26, é possível observar a topologia usada nos testes. Essa representação gráfica foi criada através do MiniEdit (FONTES et al., 2014). Na Figura 26, c0 representa o DTSA. Os *OpenFlow Switches* são representados pelo conjunto {s1, s2, s3, s4, s5, s6} e {h1, h2, h3, h4, h5, h6, h7, h8, h9, h10, h11, h12, h13, h14, h15} são os *hosts*.

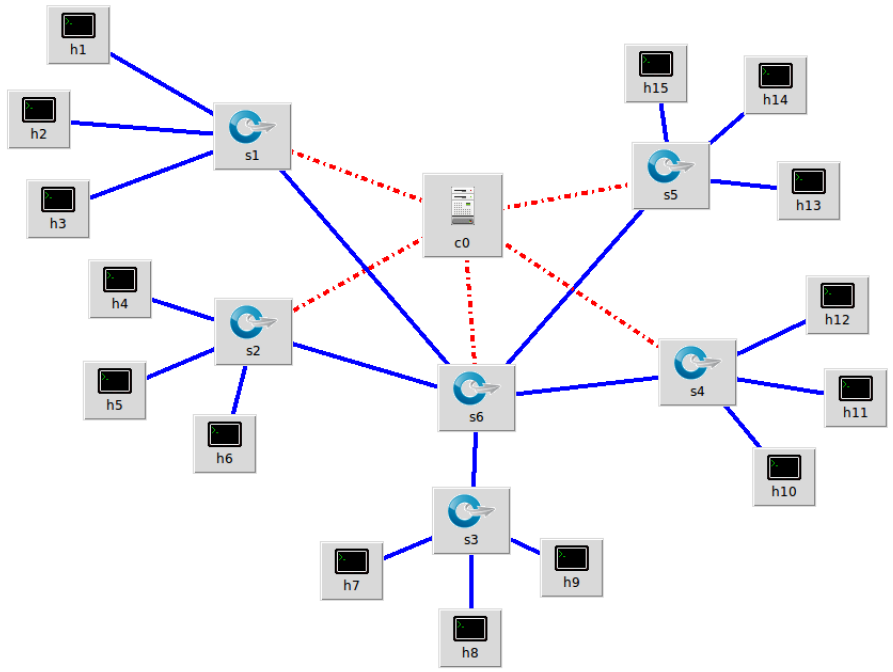

Figura 26 – Topologia

Para o teste, foi utilizada uma aplicação de chat, para a qual foi criado um *Workspace* chamado W1 e os *hosts* representados na topologia se conectaram a esse W1. Por conseguinte, foi medido o tempo que os *hosts* levaram para realizar o seu registro no ETArch (*Entity Register*) e o tempo para para se conectarem ao *Workspace* (*Workspace Attach*). Cada experimento, envolvendo toda a topologia, foi executado 10 vezes, com o objetivo de obter dados mais precisos.

Na ETArch a autenticação é desempenhada pelo *Entity Register* e o controle de acesso pelo *Workspace Attach*. Para fazer uma comparação foi medido o tempo do *Entity Register* antes da implementação proposta neste trabalho e depois da implementação. O mesmo foi feito para o *Workspace Attach*.

Antes da implementação proposta por este trabalho, a arquitetura não possuía nenhum mecanismo de segurança. Deste modo, toda a comunicação na arquitetura estava

comprometida. Com o presente trabalho, é possível realizar a autenticação e controle de acesso de cada entidade que deseja comunicar-se através da arquitetura ETArch.

A Figura 27 apresenta um comparativo do *Entity Register* sendo utilizado com e sem autenticação. No *Entity Register* o tempo médio com segurança foi de 23.05±1.54 ms enquanto o tempo sem segurança médio foi de 15.93±1.33 ms. Isso representa um acréscimo médio de 7.12ms no cenário com segurança.

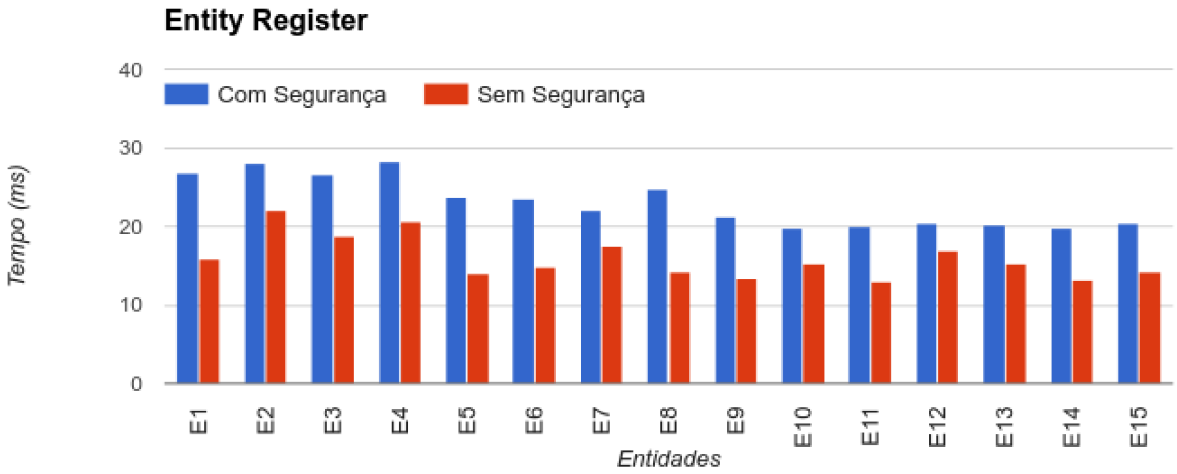

Figura 27 – Avaliação Comparativa para o Entity Register

Na Figura 28 demonstra um comparativo do *Workspace Attach* sendo aplicado em um cenário com e sem o controle de acesso. Para o *Workspace Attach* o tempo médio foi de 16.86±1.53ms e 12.26±1.96ms nos cenários com e sem segurança respectivamente, o que indica um acréscimo médio de 4.59 ms.

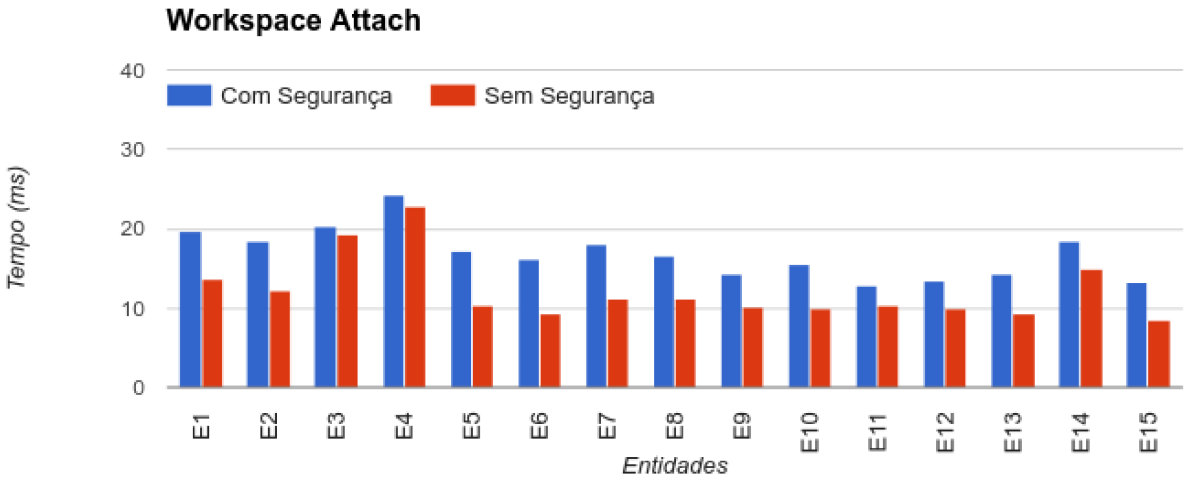

Figura 28 – Avaliação Comparativa para o Workspace Attach

5.3 Avaliação dos Resultados

Os resultados alcançados nos experimentos descritos na Seção 5.2 demonstraram que é possível que os mecanismos de autenticação e o controle de acesso sejam inseridos na arquitetura, atendendo a requisitos de segurança das aplicações.

No experimento descrito na Seção 5.2.1, foi criada uma prova de conceito para o mecanismo de autenticação, e os casos de teste demonstraram-se efetivos para o seu propósito. Outro fator importante demonstrado é como as aplicações podem fazer uso de tal mecanismo.

O experimento realizado na Seção 5.2.2 criou uma prova de conceito do mecanismo de controle de acesso. Neste cenário, o mecanismo implementado foi utilizado por uma aplicação de chat, onde foram demonstrados alguns casos onde entidades podem ou não participar de um canal de comunicação.

O experimento descrito na Seção 5.2.3 não teve o intuito de mostrar o desempenho dos mecanismos criados, mas sim de demonstrar que o acréscimo médio é comparativamente irrelevante em termos de tempo se considerarmos os benefícios dos mecanismos de autenticação e controle de acesso incorporados à ETArch.

Com o mecanismo de autenticação implementado é possível identificar a entidade como um sujeito legítimo. Deste modo, no momento em que uma dada entidade A comunicar-se com B, A terá a garantia que B é autêntico e vice-versa. Com esse benefício adquirido pela implementação, o tempo médio de acréscimo de 7.12 ms é um tempo relativamente insignificante.

Já com o mecanismo de controle de acesso a arquitetura pode permitir ou negar a participação de uma determinada entidade a um *workspace*. Desta forma a entidade responsável pelo *workspace* possui a garantia que estará transmitindo somente para entidades que possuem permissão. Esta garantia custou um acréscimo médio de 7.12 ms, um tempo pequeno para garantir que as informações sejam enviadas e cheguem somente a entidades que possuem permissão para acessar tal conteúdo.

A Tabela 7 mostra um resumo dos mecanismos de autenticação e controle de acesso em cada arquitetura. Onde o mecanismo está representado com um *Binary Times* (⊗) indica que ele está disponível parcialmente na arquitetura ou ela fornece meios para facilitar sua implementação. Já *Bullet* (•), indica que o mecanismo está disponível para utilização.

Tabela 7 – Comparativo dos Mecanismos de Segurança nas Arquiteturas

Mecanismos	Arquiteturas						
	RINA v1	RINA v2	MobilityFirst	XIA	NDN	Nebula	ETArch
Autenticação	⊗	•	⊗	⊗	•	⊗	•
Controle de Acesso	⊗	•	⊗	⊗	⊗	⊗	•

É possível observar na Tabela 7 que a implementação da arquitetura RINA v1 criada no

projeto IRATI, a arquitetura MobilityFirst, XIA, NEBULA não possuem os mecanismos de autenticação e controle de acesso implementados. A arquitetura NDN possui somente a autenticação.

As arquiteturas buscam evoluir de forma incremental como aconteceu com arquitetura RINA, que na sua v1 implementada no projeto IRATI ela não possuía os mecanismos implementados. Entretanto, a implementação RINA v2 criada no projeto PRISTINE possui tais mecanismos implementados.

Tal como aconteceu na arquitetura ETArch, a implementação dos mecanismos de autenticação e controle de acesso na arquitetura RINA também geraram um acréscimo de tempo nos mecanismos (GRASA et al., 2016b).

É possível observar que este trabalho elevou a arquitetura ETArch no quesito de segurança a outro nível, se comparado com as outras arquiteturas, onde tais mecanismos são inexistentes.

Capítulo 6

Conclusão

Este capítulo apresenta o término deste trabalho, retratando as principais conclusões obtidas ao propor novos mecanismos de autenticação e controle de acessos adequados para a arquitetura ETArch.

Realizou-se em primeiro lugar uma revisão de literatura, sobre os mecanismos de autenticação e controle de acesso empregados em propostas de novas arquiteturas para Internet do Futuro. Há várias propostas desses mecanismos aplicados em diferentes cenários e arquiteturas. Entretanto, poucos trabalhos demonstraram esses mecanismos de forma efetiva, como está sendo proposto neste trabalho.

Cada objetivo específicado foi tratado minuciosamente para alcançar o objetivo geral descrito na Seção 1.2, buscando sempre contribuições significativas para a arquitetura ETArch. Desta maneira, o trabalho guiou uma hipótese de que os mecanismos de autenticação e controle de acesso fossem oferecidos de forma intrínseca.

Os objetivos introduzidos na Seção 1.2 foram resolvidos no Capítulo 4 e demonstrados na Seção 5.2. Os mecanismos de autenticação e controle de acesso estão de acordo com as peculiaridades da arquitetura ETArch.

O Capítulo 5 apresentou demonstrações, de forma a atingir o objetivo de avaliar os mecanismos elaborados e implementados.A proposta do Capítulo 4 foi viabilizado através da implementação em um ambiente ETArch. O Capítulo demonstrou a utilização dos mecanismos propostos em topologias distintas e criou provas de conceito onde uma aplicação consome os mecanismos implementados.

A questão principal deste trabalho foi propor mecanismos de autenticação e controle de acesso para a arquitetura ETArch de forma inerente. Percebe-se que, como foi demonstrado nas seções 5.2.1 e 5.2.2 a proposição foi demonstrada, uma vez que nessas seções realizou-se uma prova de conceito do mecanismo proposto no Capítulo 4. Em vista disso, os problemas com autenticação e controle de acesso está resolvido na arquitetura ETArch.

Com os objetivos específicos alcançados e as demonstrações realizadas nos experimentos, conclui-se como cumprido o objetivo geral que é desenvolver mecanismos de autenticação e controle de acesso para a arquitetura ETArch de forma intrínseca. Visto que

os objetivos gerais e específicos foram atingidos. As próximas seções buscam destacar as principais contribuições deste trabalho e apresentar os trabalhos futuros para melhorar o cenário atual. Por fim as contribuições em produções bibliográficas serão listadas.

6.1 Principais Contribuições

O trabalho apresentou uma proposta de mecanismos de autenticação e controle de acesso para uma arquitetura de Internet do Futuro. A contribuição geral foi a definição dos mecanismos de autenticação e controle de acesso para ser aplicado na ETArch, visto que tais mecanismos não haviam sido discutidos em trabalhos anteriores. Os mecanismos foram propostos objetivando uma maior nível de segurança na arquitetura.

Os mecanismos foram definidos e implementados no plano de controle, utilizando primitivas já existentes no protocolo ETCP. No entanto foi realizado alterações nas primitivas ENTITY_REGISTER, WORKSPACE_CREATE e WORKSPACE_MODIFY.

As alterações realizadas no ENTITY_REGISTER permitem que o DTSA receba informações de acesso, o que possibilita o DTSA realizar a autenticação. No WORKSPACE_CREATE as modificações aconteceram para aceitar uma lista de controle de acesso, ou seja, a entidade que solicita criar um *workspace* informa quais entidades podem ter acesso a ele e o DTSA fica responsável por garantir que apenas as entidades informadas tenham acesso ao *workspace* solicitado. No WORKSPACE_MODIFY as modificações aconteceram para contribuir com a manutenção do *workspace*, por exemplo, no caso de uma entidade que criou o *workspace* queira bloquear o acesso de uma determinada entidade.

A utilização das primitivas ENTITY_REGISTER, WORKSPACE_CREATE e WORKSPACE_MODIFY foram demostradas e viabilizadas nos experimentos. Os mecanismos de autenticação e controle de acesso conseguiram atender de forma efetiva as necessidades de uma aplicação de chat propondo segurança de forma intrínseca. Fica claro que tais mecanismos podem ser utilizados em diversos cenários.

Foi realizado uma análise comparativa da utilização dos novos mecanismos em um dos cenários experimentais. Na avaliação foi utilizada a arquitetura ETArch com e sem os mecanismos de segurança, demonstra-se um resultado que já era esperado. Com os mecanismos desenvolvidos temos um acréscimo médio de tempo para o *Entity Register* e o *Workspace Attach*, apesar disso, há um ganho indiscutível em relação a segurança.

Outra contribuição importante deste trabalho são os mecanismos desenvolvidos incorporados ao projeto EDOBRA. Os próximos trabalhos podem aproveitar estes mecanismos para o seu desenvolvimento, bem como novas aplicações podem ser propostas utilizando tais mecanismos. A implementação foi feita utilizando a linguagem Java e implantada na plataforma Mobicents.

Por fim, pode-se concluir que é viável a utilização dos mecanismos desenvolvidos para

a ETArch. Um fato importante é que tais mecanismos são expostos de forma inerente à arquitetura, não sendo necessário ter ao longo da rede serviços de autenticação e controle de acesso para cada aplicação.

6.2 Trabalhos Futuros

Com os mecanismos de autenticação e controle de acesso definidos e implementados em um ambiente ETArch é possível que novos trabalhos possam ser conduzidos. A princípio, espera-se melhorias na proposta deste trabalho. O passo inicial seria trabalhar a inclusão de outros métodos de autenticação para suportar uma grande diversidade de entidades. Como prova de conceito, foram utilizados neste trabalho, certificados digitais próprios para prover tal funcionalidade.

Este trabalho propôs mecanismos para um único DTSA, em ambiente distribuído composto por mais de um DTSA. Os mecanismos referidos precisam ser tratados de forma global, significando que quando uma entidade se registrar em um determinado *workspace* a arquitetura toma conhecimento de tal, logo, quando ela registrar-se novamente em outro DTSA as informações de autenticação podem ser buscadas e validadas no registro atual.

Um *Intrusion Detection System* (IDS) pode ser explorado na ETArch através do *SecurityManager*, visto que este pode analisar os pedidos de autenticação, controle de acesso e criação de *workspaces*. Neste caso os trabalhos deverão propor um IDS que levam em consideração todas as primitivas, requisitos, capacidades dos *workspaces* e entidades.

Uma das características importantes da arquitetura ETArch é a capacidade de evoluir, ou seja, a arquitetura é capaz de tratar novos requisitos que possam surgir futuramente com facilidade. Para isso trabalhos de alta disponibilidade e desempenho devem ser explorados.

A questão de diagnósticos de problemas também deve ser considerada nos trabalhos futuros, isto é, uma ferramenta que permita identificar a origem de problemas de funcionamento da arquitetura. Dessa forma é possível reduzir atrasos e custos de manutenção.

Além dos trabalhos esperados em relação a ETArch citados, existe uma enorme necessidade de desenvolvimento de novas aplicações para testar de forma passiva os mecanismos implementados atualmente. No momento presente, todos os testes foram realizados através de uma aplicação de chat e *streaming* de vídeo.

6.3 Contribuições em Produção Bibliográfica

- Artigo aceito na *Advanced International Conference on Telecommunications (AICT)* 2015 - Entity Title Architecture Pilot: Deploying a Clean Slate SDN Based Network at a Telecom Operator

- Artigo aceito na *International Conference on Networks* (ICN) 2017 - Entity Title Architecture Pilot: Scaling Out the Deployment Clean Slate SDN Based Network at a Telecom Operator.

Referências

ANAND, A. et al. XIA: An Architecture for an Evolvable and Trustworthy Internet. In: **Proceedings of the 10th ACM Workshop on Hot Topics in Networks**. New York, NY, USA: ACM, 2011. (HotNets-X), p. 2:1–2:6. ISBN 978-1-4503-1059-8. Disponível em: <http://doi.acm.org/10.1145/2070562.2070564>.

ANDERSON, T. et al. A Brief Overview of the NEBULA Future Internet Architecture. **SIGCOMM Comput. Commun. Rev.**, v. 44, n. 3, p. 81–86, jul. 2014. ISSN 0146-4833. Disponível em: <http://doi.acm.org/10.1145/2656877.2656889>.

Bass. **Software Architecture in Practice**. 2 edition. ed. [S.l.]: Dorling Kindersley Pvt Ltd, 2006. ISBN 9788177589962.

Big Switch Networks. **Floodlight OpenFlow Controller**. 2014. Http://www.projectfloodlight.org/. Disponível em: <http://www.projectfloodlight.org/>.

BRONZINO, F. et al. Network Service Abstractions for a Mobility-centric Future Internet Architecture. In: **Proceedings of the Eighth ACM International Workshop on Mobility in the Evolving Internet Architecture**. New York, NY, USA: ACM, 2013. (MobiArch '13), p. 5–10. ISBN 978-1-4503-2366-6. Disponível em: <http://doi.acm.org/10.1145/2505906.2505908>.

BRONZINO, F.; RAYCHAUDHURI, D.; SESKAR, I. Experiences with testbed evaluation of the MobilityFirst Future Internet Architecture. In: **2015 European Conference on Networks and Communications (EuCNC)**. [S.l.: s.n.], 2015. p. 507–511.

BURROWS, M.; ABADI, M.; NEEDHAM, R. M. A logic of authentication. In: THE ROYAL SOCIETY. **Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences**. [S.l.], 1989. v. 426, n. 1871, p. 233–271.

BUSCHMANN, F. et al. **Pattern-Oriented Software Architecture Volume 1: A System of Patterns**. Volume 1 edition. Chichester ; New York: Wiley, 1996. ISBN 9780471958697.

CASTILLO, J. et al. Additions to the ETArch control plane to support multimedia QoS-guaranteed content transport over OpenFlow-enabled SDN future internet systems. In: **Globecom Workshops (GC Wkshps), 2014**. [S.l.: s.n.], 2014. p. 172–177.

CLARK, D. et al. **Towards the Future Internet Architecture**. IETF, 1991. (Request for Comments, 1287). Published: RFC 1287 (Informational). Disponível em: <http://www.ietf.org/rfc/rfc1287.txt>.

DAY, J.; MATTA, I.; MATTAR, K. Networking is IPC: A Guiding Principle to a Better Internet. In: **Proceedings of the 2008 ACM CoNEXT Conference**. New York, NY, USA: ACM, 2008. (CoNEXT '08), p. 67:1–67:6. ISBN 978-1-60558-210-8. Disponível em: <http://doi.acm.org/10.1145/1544012.1544079>.

DING, W.; YAN, Z.; DENG, R. H. A Survey on Future Internet Security Architectures. **IEEE Access**, v. 4, p. 4374–4393, 2016. ISSN 2169-3536.

EGEVANG, K.; FRANCIS, P. **The IP Network Address Translator (NAT)**. IETF, 1994. (Request for Comments, 1631). Published: RFC 1631 (Informational) Obsoleted by RFC 3022. Disponível em: <http://www.ietf.org/rfc/rfc1631.txt>.

ELLIOTT, C. GENI: Opening Up New Classes of Experiments in Global Networking. **IEEE internet computing**, v. 14, n. 1, p. 39–42, 2010. ISSN 1089-7801. Disponível em: <http://cat.inist.fr/?aModele=afficheN&cpsidt=22353104>.

FELDMANN, A. Internet Clean-slate Design: What and Why? **SIGCOMM Comput. Commun. Rev.**, v. 37, n. 3, p. 59–64, jul. 2007. ISSN 0146-4833. Disponível em: <http://doi.acm.org/10.1145/1273445.1273453>.

FEMMINELLA, M. et al. Implementation and performance analysis of advanced IT services based on open source JAIN SLEE. In: **2011 IEEE 36th Conference on Local Computer Networks**. [S.l.: s.n.], 2011. p. 746–753.

FONTES, R. R. et al. Authoring of openflow networks with visual network description (sdn version)(wip). In: SOCIETY FOR COMPUTER SIMULATION INTERNATIONAL. **Proceedings of the 2014 Summer Simulation Multiconference**. [S.l.], 2014. p. 22.

FOUNDATION, O. N. **ONF Overview - Open Networking Foundation**. 2017. Disponível em: <https://www.opennetworking.org/about/onf-overview>.

GAVRAS, A. et al. Future Internet Research and Experimentation: The FIRE Initiative. **SIGCOMM Comput. Commun. Rev.**, v. 37, n. 3, p. 89–92, jul. 2007. ISSN 0146-4833. Disponível em: <http://doi.acm.org/10.1145/1273445.1273460>.

GLADISCH, A.; DAHER, R.; TAVANGARIAN, D. Survey on Mobility and Multihoming in Future Internet. **Wireless Pers Commun**, v. 74, n. 1, p. 45–81, jan. 2014. ISSN 0929-6212, 1572-834X. Disponível em: <http://link.springer.com/article/10.1007/s11277-012-0898-6>.

GONçALVES, M. A. et al. Multicast Traffic Aggregation through Entity Title Model. In: . [s.n.], 2014. p. 175–180. ISBN 978-1-61208-360-5. Disponível em: <https://www.thinkmind.org/index.php?view=article&articleid=aict_2014_7_40_10177>.

GRASA, E. et al. From protecting protocols to layers: Designing, implementing and experimenting with security policies in rina. In: IEEE. **2016 IEEE International Conference on Communications (ICC)**. [S.l.], 2016. p. 1–7.

_____. From protecting protocols to layers: Designing, implementing and experimenting with security policies in RINA. In: **2016 IEEE International Conference on Communications (ICC)**. [S.l.: s.n.], 2016. p. 1–7.

_____. Developing a RINA Prototype over UDP/IP Using TINOS. In: **Proceedings of the 7th International Conference on Future Internet Technologies**. New York, NY, USA: ACM, 2012. (CFI '12), p. 31–36. ISBN 978-1-4503-1690-3. Disponível em: <http://doi.acm.org/10.1145/2377310.2377321>.

GROUP, I. E. S.; HINDEN, R. **Applicability Statement for the Implementation of Classless Inter-Domain Routing (CIDR)**. IETF, 1993. (Request for Comments, 1517). Published: RFC 1517 (Historic). Disponível em: <http://www.ietf.org/rfc/rfc1517.txt>.

GUIMARAES, C. et al. IEEE 802.21-enabled Entity Title Architecture for handover optimization. In: **2014 IEEE Wireless Communications and Networking Conference (WCNC)**. [S.l.: s.n.], 2014. p. 2671–2676.

HAKIRI, A. et al. Software-Defined Networking: Challenges and research opportunities for Future Internet. **Computer Networks**, v. 75, Part A, p. 453–471, dez. 2014. ISSN 1389-1286. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1389128614003703>.

HAN, D. et al. Xia: Efficient support for evolvable internetworking. In: **NSDI**. [S.l.: s.n.], 2012. v. 12, p. 23–23.

HANDLEY, M. Why the Internet Only Just Works. **BT Technology Journal**, v. 24, n. 3, p. 119–129, jul. 2006. ISSN 1358-3948. Disponível em: <http://dx.doi.org/10.1007/s10550-006-0084-z>.

HSIAO, H.-C. et al. STRIDE: Sanctuary Trail – Refuge from Internet DDoS Entrapment. In: **Proceedings of the 8th ACM SIGSAC Symposium on Information, Computer and Communications Security**. New York, NY, USA: ACM, 2013. (ASIA CCS '13), p. 415–426. ISBN 978-1-4503-1767-2. Disponível em: <http://doi.acm.org/10.1145/2484313.2484367>.

_____. LAP: Lightweight Anonymity and Privacy. In: **2012 IEEE Symposium on Security and Privacy (SP)**. [S.l.: s.n.], 2012. p. 506–520.

HWANG, M.-S.; LI, L.-H. A new remote user authentication scheme using smart cards. **IEEE Transactions on Consumer Electronics**, New York: Institute of Electrical and Electronics Engineers, v. 46, n. 1, p. 28–30, 2000.

JACOBSON, V. et al. Networking Named Content. In: **Proceedings of the 5th International Conference on Emerging Networking Experiments and Technologies**. New York, NY, USA: ACM, 2009. (CoNEXT '09), p. 1–12. ISBN 978-1-60558-636-6. Disponível em: <http://doi.acm.org/10.1145/1658939.1658941>.

JAIN, R. Internet 3.0: Ten Problems with Current Internet Architecture and Solutions for the Next Generation. In: **IEEE Military Communications Conference, 2006. MILCOM 2006**. [S.l.: s.n.], 2006. p. 1–9.

KIM, T. H.-J. et al. Lightweight source authentication and path validation. In: ACM. **ACM SIGCOMM Computer Communication Review**. [S.l.], 2014. v. 44, n. 4, p. 271–282.

_____. Accountable Key Infrastructure (AKI): A Proposal for a Public-key Validation Infrastructure. In: **Proceedings of the 22Nd International Conference on World Wide Web**. Republic and Canton of Geneva, Switzerland: International World Wide Web Conferences Steering Committee, 2013. (WWW '13), p. 679–690. ISBN 978-1-4503-2035-1. Disponível em: <http://dl.acm.org/citation.cfm?id=2488388.2488448>.

_____. Accountable key infrastructure (aki): a proposal for a public-key validation infrastructure. In: ACM. **Proceedings of the 22nd international conference on World Wide Web**. [S.l.], 2013. p. 679–690.

KREUTZ, D. et al. Software-Defined Networking: A Comprehensive Survey. **Proceedings of the IEEE**, v. 103, n. 1, p. 14–76, jan. 2015. ISSN 0018-9219.

KRISHNAMOORTHY, A. **Implementation and evaluation of the MobilityFirst protocol stack on software-defined network platforms**. Tese (Doutorado) — Rutgers, The State University of New Jersey, 2013.

LAMPORT, L. Password authentication with insecure communication. **Communications of the ACM**, ACM, v. 24, n. 11, p. 770–772, 1981.

LIU, X.; TRAPPE, W.; ZHANG, Y. Secure name resolution for identifier-to-locator mappings in the global internet. In: IEEE. **2013 22nd International Conference on Computer Communication and Networks (ICCCN)**. [S.l.], 2013. p. 1–7.

MACHADO, I. et al. Building an infrastructure for experimentation between brazil and europe to enhance research collaboration in future internet. 2014.

MCKEOWN, N. Software-defined networking. **INFOCOM keynote talk**, v. 17, n. 2, p. 30–32, 2009.

MCKEOWN, N. et al. OpenFlow: Enabling Innovation in Campus Networks. **SIGCOMM Comput. Commun. Rev.**, v. 38, n. 2, p. 69–74, mar. 2008. ISSN 0146-4833. Disponível em: <http://doi.acm.org/10.1145/1355734.1355746>.

MOREIRA, M. D. et al. Internet do futuro: Um novo horizonte. **Minicursos do Simpósio Brasileiro de Redes de Computadores-SBRC**, v. 2009, p. 1–59, 2009.

MUKHERJEE, S. et al. **Network-Assisted Multihoming in the Mobility-First Future Internet Architecture**. [S.l.], 2014.

NETO, N. Vieira de S. et al. Control Plane Routing Protocol for the Entity Title Architecture. In: . [s.n.], 2015. p. 185–190. ISBN 978-1-61208-398-8. Disponível em: <https://www.thinkmind.org/index.php?view=article&articleid=icn_2015_7_40_30210>.

NG, E.; CAI, Z.; COX, A. Maestro: A system for scalable openflow control. **Rice University, Houston, TX, USA, TSEN Maestro-Techn. Rep, TR10-08**, 2010.

NUNES, B. A. A. et al. A Survey of Software-Defined Networking: Past, Present, and Future of Programmable Networks. **IEEE Communications Surveys Tutorials**, v. 16, n. 3, p. 1617–1634, 2014. ISSN 1553-877X.

OFELIA. **OpenFlow in Europe - Linking Infrastructure and Applications**. 2014. Disponível em: <http://www.fp7-ofelia.eu/about-ofelia/>.

PAN, J.; PAUL, S.; JAIN, R. A survey of the research on future internet architectures. **IEEE Communications Magazine**, v. 49, n. 7, p. 26–36, jul. 2011. ISSN 0163-6804.

PEREIRA, J. H. d. S. **Modelo de título para a próxima geração de Internet.** Tese (Doutorado) — Universidade de São Paulo, 2012.

PEREIRA, J. H. d. S.; KOFUJI, S. T.; ROSA, P. F. Horizontal Addressing by Title in a Next Generation Internet. In: **2010 Sixth International Conference on Networking and Services (ICNS)**. [S.l.: s.n.], 2010. p. 7–11.

PEREIRA, J. H. de S. et al. Title model ontology for future internet networks. In: _____. **The Future Internet: Future Internet Assembly 2011: Achievements and Technological Promises**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011. p. 103–114. ISBN 978-3-642-20898-0. Disponível em: <http://dx.doi.org/10.1007/978-3-642-20898-0_8>.

RAYCHAUDHURI, D.; NAGARAJA, K.; VENKATARAMANI, A. Mobilityfirst: a robust and trustworthy mobility-centric architecture for the future internet. **ACM SIGMOBILE Mobile Computing and Communications Review**, ACM, v. 16, n. 3, p. 2–13, 2012.

REC, I. X. 800 security architecture for open systems interconnection for ccitt applications. **ITU-T (CCITT) Recommendation**, 1991.

REKHTER, Y.; LI, T.; HARES, S. **A Border Gateway Protocol 4 (BGP-4)**. [S.l.], 2006. Disponível em: <https://www.rfc-editor.org/info/rfc4271>.

REXFORD, J.; DOVROLIS, C. Future Internet Architecture: Clean-slate Versus Evolutionary Research. **Commun. ACM**, v. 53, n. 9, p. 36–40, set. 2010. ISSN 0001-0782. Disponível em: <http://doi.acm.org/10.1145/1810891.1810906>.

ROBERTS, J. The clean-slate approach to future Internet design: a survey of research initiatives. **annals of telecommunications - annales des télécommunications**, v. 64, n. 5-6, p. 271–276, maio 2009. ISSN 0003-4347, 1958-9395. Disponível em: <http://link.springer.com/article/10.1007/s12243-009-0109-y>.

SAMARATI, P.; VIMERCATI, S. C. d. Access Control: Policies, Models, and Mechanisms. In: **Foundations of Security Analysis and Design**. Springer, Berlin, Heidelberg, 2000. p. 137–196. DOI: 10.1007/3-540-45608-2_3. Disponível em: <http://link.springer.com/chapter/10.1007/3-540-45608-2_3>.

SANDHU, R. S.; SAMARATI, P. Access control: principle and practice. **IEEE communications magazine**, IEEE, v. 32, n. 9, p. 40–48, 1994.

SCHMIDT, D. et al. **Pattern-Oriented Software Architecture Volume 2: Patterns for Concurrent and Networked Objects**. Volume 2 edition. Chichester England ; New York: Wiley, 2000. ISBN 9780471606956.

SESKAR, I. et al. MobilityFirst Future Internet Architecture Project. In: **Proceedings of the 7th Asian Internet Engineering Conference**. New York, NY, USA: ACM, 2011. (AINTEC '11), p. 1–3. ISBN 978-1-4503-1062-8. Disponível em: <http://doi.acm.org/10.1145/2089016.2089017>.

SILVA, F. d. O. **Endereçamento por título: uma forma de encaminhamento multicast para a próxima geração de redes de computadores.** Tese (Doutorado) — Universidade de São Paulo, out. 2013.

SILVA, F. d. O. et al. On the analysis of multicast traffic over the Entity Title Architecture. In: **2012 18th IEEE International Conference on Networks (ICON)**. [S.l.: s.n.], 2012. p. 30–35.

SRISURESH, P.; EGEVANG, K. **Traditional IP Network Address Translator (Traditional NAT)**. IETF, 2001. (Request for Comments, 3022). Published: RFC 3022 (Informational). Disponível em: <http://www.ietf.org/rfc/rfc3022.txt>.

THEODORO, L. et al. Entity Title Architecture Pilot: Deploying a Clean Slate SDN Based Network at a Telecom Operator. In: . [s.n.], 2015. p. 144–149. ISBN 978-1-61208-411-4. Disponível em: <http://www.thinkmind.org/index.php?view=article&articleid=aict_2015_7_40_10152>.

TOOTOONCHIAN, A.; GANJALI, Y. Hyperflow: A distributed control plane for openflow. In: **Proceedings of the 2010 internet network management conference on Research on enterprise networking**. [S.l.: s.n.], 2010. p. 3–3.

UNION, I. T. X.811:Information technology - Open Systems Interconnection - Security frameworks for open systems: Authentication framework. In: . [s.n.], 1995. Disponível em: <https://www.itu.int/rec/T-REC-X.811>.

_____. X.812:Information technology - Open Systems Interconnection - Security frameworks for open systems: Access control framework. In: . [s.n.], 1995. Disponível em: <https://www.itu.int/rec/T-REC-X.812-199511-I/en>.

VENKATARAMANI, A. et al. Design requirements of a global name service for a mobility-centric, trustworthy internetwork. In: IEEE. **2013 Fifth International Conference on Communication Systems and Networks (COMSNETS)**. [S.l.], 2013. p. 1–9.

VRIJDERS, S. et al. Prototyping the recursive internet architecture: the IRATI project approach. **IEEE Network**, v. 28, n. 2, p. 20–25, mar. 2014. ISSN 0890-8044.

VU, T. et al. Dmap: A shared hosting scheme for dynamic identifier to locator mappings in the global internet. In: IEEE. **Distributed Computing Systems (ICDCS), 2012 IEEE 32nd International Conference on**. [S.l.], 2012. p. 698–707.

WANG, Y. et al. Introducing protorina: a prototype for programming recursive-networking policies. **ACM SIGCOMM Computer Communication Review**, ACM, v. 44, n. 3, p. 129–131, 2014.

WAYMAN, J. et al. **An introduction to biometric authentication systems**. [S.l.]: Springer, 2005.

WHITMAN, M. E. Enemy at the gate: threats to information security. **Communications of the ACM**, ACM, v. 46, n. 8, p. 91–95, 2003.

XYLOMENOS, G. et al. A Survey of Information-Centric Networking Research. **IEEE Communications Surveys Tutorials**, v. 16, n. 2, p. 1024–1049, 2014. ISSN 1553-877X.

YU, Y. et al. Schematizing Trust in Named Data Networking. In: **Proceedings of the 2Nd ACM Conference on Information-Centric Networking**. New York, NY, USA: ACM, 2015. (ACM-ICN '15), p. 177–186. ISBN 978-1-4503-3855-4. Disponível em: <http://doi.acm.org/10.1145/2810156.2810170>.

YU, Y.; AFANASYEV, A.; ZHANG, L. Name-based access control. **Named Data Networking Project, Technical Report NDN-0034**, 2015.

ZHANG, F. et al. Content delivery in the mobilityfirst future internet architecture. In: IEEE. **Sarnoff Symposium (SARNOFF), 2012 35th IEEE**. [S.l.], 2012. p. 1–5.

Apêndices

APÊNDICE **A**

Codificação das topologias utilizadas

As topologias utilizadas para os experimentos no Capítulo **??** *foram criadas através de scripts em linguagem Python. Esses scripts foram passados para o Mininet, sendo assim, o Mininet inicia-se considerando a topologia presente em cada script.*

A.1 Script Python do experimento I

```
#!/usr/bin/python

import re
from mininet.net import Mininet
from mininet.node import OVSSwitch, Controller, RemoteController
from mininet.topolib import TreeTopo
from mininet.topo import LinearTopo
from mininet.log import setLogLevel, info
from mininet.cli import CLI
from mininet.topo import Topo

def toID(mac):
    return '0000' + re.sub('[:]', '', mac)

def myTopo():

        info( '*** Adding controller\n' )
    #net = Mininet( controller=RemoteController)
    #net.addController('c0')

    net = Mininet( topo=None,
                   build=False,
                   ipBase='10.0.0.0/8')
```

```
c0=net.addController(name='c0',
                     controller=RemoteController,
            ip='192.168.0.152',
                     protocol='tcp',
            port=6633)


info( '*** Add switches\n')
    s1 = net.addSwitch('s1', dpid=toID('00:00:00:00:00:01'))
    s2 = net.addSwitch('s2', dpid=toID('00:00:00:00:00:02'))
    s3 = net.addSwitch('s3', dpid=toID('00:00:00:00:00:03'))

info( '*** Add hosts\n')
h1 = net.addHost('h1', ip='10.0.0.1')
h2 = net.addHost('h2', ip='10.0.0.2')
h3 = net.addHost('h3', ip='10.0.0.3')
h4 = net.addHost('h4', ip='10.0.0.4')

# Add links
net.addLink(s1, h1, 1, 1)
net.addLink(s1, h2, 2, 1)
net.addLink(s2, h4, 1, 1)
net.addLink(s3, h3, 1, 1)

net.addLink(s1, s2, 3, 2)
net.addLink(s1, s3, 4, 2)

info( '*** Starting network\n')

net.build()

info( '*** Starting controllers\n')

    for controller in net.controllers:
        controller.start()

    info( '*** Starting switches\n')
```

```
        net.get('s1').start([c0])
    net.get('s2').start([c0])
        net.get('s3').start([c0])

        CLI(net)
        net.stop

if __name__ == '__main__':
    setLogLevel('info')
    myTopo()
```

A.2 Script Python do experimento II

```
#!/usr/bin/python

import re
from mininet.net import Mininet
from mininet.node import OVSSwitch, Controller, RemoteController
from mininet.topolib import TreeTopo
from mininet.topo import LinearTopo
from mininet.log import setLogLevel, info
from mininet.cli import CLI
from mininet.topo import Topo

def toID(mac):
    return '0000' + re.sub('[:]', '', mac)

def myTopo():

        info( '*** Adding controller\n' )
    #net = Mininet( controller=RemoteController)
    #net.addController('c0')

    net = Mininet( topo=None,
                   build=False,
                   ipBase='10.0.0.0/8')

    c0=net.addController(name='c0',
                         controller=RemoteController,
              ip='192.168.0.152',
```

```
                protocol='tcp',
            port=6633)


info( '*** Add switches\n')
    s1 = net.addSwitch('s1', dpid=toID('00:00:00:00:00:01'))
    s2 = net.addSwitch('s2', dpid=toID('00:00:00:00:00:02'))
    s3 = net.addSwitch('s3', dpid=toID('00:00:00:00:00:03'))
    s4 = net.addSwitch('s4', dpid=toID('00:00:00:00:00:04'))
    s5 = net.addSwitch('s5', dpid=toID('00:00:00:00:00:05'))
    s6 = net.addSwitch('s6', dpid=toID('00:00:00:00:00:06'))

info( '*** Add hosts\n')
h1 = net.addHost('h1', ip='10.0.0.1')
h2 = net.addHost('h2', ip='10.0.0.2')
h3 = net.addHost('h3', ip='10.0.0.3')

h4 = net.addHost('h4', ip='10.0.0.4')
h5 = net.addHost('h5', ip='10.0.0.5')
h6 = net.addHost('h6', ip='10.0.0.6')

h7 = net.addHost('h7', ip='10.0.0.7')
h8 = net.addHost('h8', ip='10.0.0.8')
h9 = net.addHost('h9', ip='10.0.0.9')

h10 = net.addHost('h10', ip='10.0.0.10')
h11 = net.addHost('h11', ip='10.0.0.11')
h12 = net.addHost('h12', ip='10.0.0.12')

h13 = net.addHost('h13', ip='10.0.0.13')
h14 = net.addHost('h14', ip='10.0.0.14')
h15 = net.addHost('h15', ip='10.0.0.15')

# Add links
net.addLink(s1, h1, 1, 1)
net.addLink(s1, h2, 2, 1)
net.addLink(s1, h3, 3, 1)

net.addLink(s2, h4, 1, 1)
```

```
net.addLink(s2, h5, 2, 1)
net.addLink(s2, h6, 3, 1)

net.addLink(s3, h7, 1, 1)
net.addLink(s3, h8, 2, 1)
net.addLink(s3, h9, 3, 1)

net.addLink(s4, h10, 1, 1)
net.addLink(s4, h11, 2, 1)
net.addLink(s4, h12, 3, 1)

net.addLink(s5, h13, 1, 1)
net.addLink(s5, h14, 2, 1)
net.addLink(s5, h15, 3, 1)

net.addLink(s1, s2, 4, 4)
net.addLink(s6, s3, 1, 4)
net.addLink(s6, s4, 2, 4)
net.addLink(s6, s1, 3, 5)
net.addLink(s6, s5, 4, 4)

info( '*** Starting network\n')

net.build()

info( '*** Starting controllers\n')

for controller in net.controllers:
    controller.start()

info( '*** Starting switches\n')

net.get('s1').start([c0])
net.get('s2').start([c0])
net.get('s3').start([c0])
net.get('s4').start([c0])
net.get('s5').start([c0])
net.get('s6').start([c0])
```

```
    CLI(net)
    net.stop

if __name__ == '__main__':
    setLogLevel('info')
    myTopo()
```