PolicyFlow: Uma Arquitetura de Gerência Baseada em Políticas Semânticas com OpenFlow

Joseilson Albuquerque de França
Centro de Informática – Universidade Federal de Pernambuco (UFPE)
Caixa Postal 7581 – 50732-970 – Recife – PE – Brasil
jaf@cin.ufpe.br

Luciana Pereira Oliveira
Unidade Acadêmica de Informática – Instituto Federal da Paraíba (IFPB)
Av. 1º de Maio, 720, Jaguaribe - 58.015-430 - João Pessoa - PB - Brasil
luciana.oliveira@ifpb.edu.br

Djamel Fawzi Hadj Sadok
Centro de Informática – Universidade Federal de Pernambuco (UFPE)
Caixa Postal 7581 – 50732-970 – Recife – PE – Brasil
jamel@cin.ufpe.br

RESUMO: *O gerenciamento das redes está mais complexo a cada dia. Por outro lado, recursos da web semântica e o uso de ontologias por meio de linguagens de alto nível e amigável permitem um alto poder de definição e administração das redes. Com o intuito de unir a complexidade atual das redes com o poder e os mecanismos de web semântica propomos o framework ProNet. Neste artigo, apresentamos um estudo de caso do ProNet utilizando o OpenFlow que permite a configuração dinâmica de switches. Nossas avaliações demonstraram as vantagens de um ambiente que permite descrever ontologias e regras em linguagem de alto nível para definir o comportamento da rede, analisar e controlar todos os fluxos de dados que por ela passem.*

PALAVRAS CHAVE*: Redes Definidas por Software (SDN), Web Semâmtica, OpenFlow, Protégé e Gerenciamento de Redes*

ABSTRACT: *The network management has become more complex every day. On the other hand, the resources and the use of semantic web ontologies by means of high-level languages and friendly power to allow a high-definition and management of networks. In order to unite the complexity of current networks with the power and the mechanisms of the semantic web, we propose ProNet framework. In this paper, we present a case study that integrates it and OpenFlow that that allows the dynamic configuration of switches. Our evaluations have demonstrated the advantages of an environment for describing ontologies and rules in high-level language to define the behavior of the network analyze and control all data flow that pass through network.*

KEY-WORDS: *Software Defined Network (SDN), Web Semantic, OpenFlow, Protégé, Network Management*

1. Introdução

Formada por equipamentos dos mais diversos fabricantes e modelos, as redes de computadores se apresentam cada vez mais complexas e heterogêneas, fazendo da sua administração um tarefa árdua e de difícil resolução. Esta realidade exige dos administradores de redes um conhecimento cada vez maior das tecnologias e das particularidades de cada fabricante, e um frequente apoio de especialistas para resolução de questões que exigem um conhecimento de mais baixo nível destes equipamentos. Por outro lado, visando alcançar um modelo simples e intuitivo para representar os conhecimentos das mais diversas áreas, recursos da web semântica se apresentam como uma solução, fazendo-se uso da ontologia com uma linguagem de alto nível e com uma grande capacidade de definição de bases de conhecimento, inclusive das redes de computadores. Diante do exposto vislumbramos a possibilidade de gerenciamento das redes de computadores através de regras estabelecidas em web semântica, tornando tal tarefa mais acessível e de fácil compreensão.

No entanto, para que tais regras e definições sejam aplicadas em uma rede heterogênea conforme abordado, é necessário o desenvolvimento de uma estrutura de *software* em que seja possível a definição de regras nesta linguagem de alto nível (ontologia) e tradução desta em comandos, que utilizam linguagens de mais baixo nível, para os elementos de uma rede composta pelos mais diversos fabricantes de equipamentos. Tais equipamentos possuem comandos diferentes e incompatíveis entre fabricantes, o que dificulta ainda mais tal tarefa.

Para alcançarmos o objetivo de desenvolver uma solução com as características de flexibilidade de reprogramação de equipamentos de redes, suporte a ontologia e fácil integração de comandos de alto e baixo nível, teremos que alcançar a respostas para as seguintes questões:

- É possível fazer o gerenciamento usando ontologia com eficiência?
- Qual é o ganho nesta evolução para um sistema de gerenciamento de redes?

Em busca de um modelo que apresente tais características, podem-se encontrar ferramentas que permitem a edição de informações semânticas, processamento de informações, configuração dinâmica de elementos de redes e outros mecanismos a serem utilizadas para o gerenciamento de redes. Entre as alternativas, (OLIVEIRA *et al.*, 2008) apresenta um primeiro passo para integrar esses mecanismos em um ambiente para construção de simulações de gerenciamento de redes. Neste trabalho, apresenta-se um segundo passo para controlar a rede por meio da associação de informação semântica a elementos de redes configuráveis.

Pra isso, este documento está estruturado da seguinte forma: na seção 2 listamos alguns trabalhos que possuem relação com o presente trabalho, a seção 3 apresenta o *framework* ProNet que representa a proposta de (OLIVEIRA *et al.*, 2008); a seção 4 descreve o OpenFlow que oferece soluções para permitir que elementos de redes sejam dinamicamente configuráveis; a seção 5 apresenta a proposta deste trabalho; a seção 6 demonstra como foi realizada a codificação desta proposta de solução denominada de PolicyFlow; a seção 7 apresenta a avaliação desta proposta; e finalmente na seção 8 são feitas as considerações finais e propostas de trabalhos futuros.

2. Trabalhos relacionados

Vislumbrando a utilização da Web Semântica para o gerenciamento de redes de computadores existem alguns trabalhos como (XU; XIAO, 2006), que descrevem o uso de regras semânticas. Usando linguagens como OWL[1], OWL-S[2] e SWRL[3] através do sistema Protégé[4] e demonstrado um exemplo de ontologia que poderia ser implementada para o gerenciamento de redes IP. E tem como fundamental intuito o incentivo a utilização da ontologia para este fim. Os mesmos autores em (XU; XIAO, 2007) fazem uma evolução de sua proposta acima relatada e demonstra um modelo para este fim que utiliza as mesmas linguagens de definição de ontologia OWL, SWRL e OWL-S. E Sugerem um passo a passo

[1] http://www.w3.org/2004/OWL/ - acessado em 17/02/2012.

[2] http://www.w3.org/Submission/OWL-S/ - acessado em 05/03/2012.

[3] http://www.w3.org/Submission/SWRL/ - acessado em 21/02/2012.

[4] http://protege.stanford.edu/ - acessado em 21/02/2012.

para a construção de uma solução baseada em Web Semântica para o gerenciamento de uma rede de computadores. Contudo tais propostas são apenas conceituais, não sendo apresentada a codificação de suas propostas em uma ambiente real. Este conceito é detalhados por (KEENEY *et al.*, 2006) que apresenta um sistema denominado AKDF (*Autonomic Knowledge Delivery Framework*). Este utiliza os recursos da Web Semântica para facilitar o gerenciamento de redes, não exigindo do usuário um conhecimento profundo a respeito dos equipamentos da rede. Toda via, esta solução depende de *softwares* comerciais para seu funcionamento como a solução para CBN (*Content Based Network*) Elvin (SEGALL *et al.*, 2000) e a plataforma de gerenciamento de elementos de rede Q3A-DE[5]. Esses *softwares* são proprietários, limitando sua utilização, pois torna a solução dependente dos interesses particulares de seus desenvolvedores. Pertinente ao tema, contudo com uma abordagem mais direcionada, existem trabalhos como (MORAES *et al.*, 2008) que apresenta uma ontologia chamada MonONTO que tem a função de permitir ao usuário de uma rede de computadores solicitar certa qualidade de serviço (QoS) para atender uma aplicação (seja de voz, vídeo ou texto), a qual será analisada pelo sistema utilizando como base recursos de ontologia. Utiliza o Protégé como Editor para as linguagens OWL, SWRL e SPARQL[6] em sua implementação. E para inferência é utilizado o Jess[7]. Neste caso a ontologia não está associada a uma ferramenta de aplicação dos conceitos desenvolvidos em uma rede. Fornece apenas a ontologia para aplicações que necessitem de QoS. Em busca de soluções para economia de energia, ou seja, redes verdes, (ROSSELLO-BUSQUET *et al.*, 2011) tratam a respeito de um sistema chamado de HEMS (*Home Energy Management System*). Cuja função é controlar o consumo de energia de todos os aparelhos elétricos de uma residência através de regras semânticas utilizando as linguagens SWRL e OWL. Neste sistema o usuário cria estas regras de funcionamento de seus aparelhos usando SWRL, e o sistema, utilizando o Jess como máquina de inferência, que possui uma ontologia específica para cada tipo de aparelho, aplica tais regras com o intuito na redução do consumo de energia gerado por uma rede residencial. Com uma abordagem voltada para segurança de redes e serviços (FITZGERALD; FOLEY, 2010) explana a respeito da inclusão da ontologia como ferramenta para fazer o controle de acesso de segurança a serviços em uma rede de computadores. Como exemplo principal é descrito o controle proposto para o protocolo de mensagens instantâneas XMPP. Neste caso com o intuito de fazer controle de SPAM sobre mensagens instantâneas (SPIM). Com isto observamos a utilização da Web Semântica na análise e controle de serviços no nível de aplicação, e demonstra que a aplicação deste recurso pode abordar todos os níveis da comunicação de dados. Ainda em (DAVY *et al.*, 2008) sugere-se a utilização da ontologia como uma ferramenta mais eficaz no tratamento de conflitos entre políticas de rede. Usa como exemplo políticas de segurança de um *firewall* e o serviço VPNSec (*Virtual Private Network Security*). Que em uma determinada situação um conflito de políticas específicas de cada um destes serviços poderia gerar a quebra da política do outro. Então é explanado, com este exemplo, como a situação é tratada com o uso da ontologia. Assim é demonstrado como a ontologia tem potencial para resolver conflitos de forma mais rápida e eficiente. Podemos ainda observar o poder e a variabilidade de aplicações da Web Semântica em ambientes de redes de computadores em (VISWANATHAN *et al.*, 2011) que apresenta o SAF (*Semantic Analysis Framework*), um sistema de análise de tráfego baseado em regras semânticas. Tem como principal objetivo permitir que através de uma linguagem de alto nível (semântica) seja possível analisar dados que estão representados em formato de baixo nível. Com este tipo de sistema seria possível realizar análises da forma como o usuário entende a rede, não dependendo de um conhecimento mais aprofundado dos detalhes existentes em pacotes que fluem em uma rede de computadores.

Finalmente, é utilizado como base fundamental para início deste presente trabalho o ProNet (OLIVEIRA *et al.*, 2008), um *framework* que será detalhado na próxima seção.

---

5 http://www.uhcommunications.com/products/3g/3g.html - acessado em 28/02/2012.

6 http://www.w3.org/TR/rdf-sparql-query/ - acessado em 05/03/2012.

7 http://www.jessrules.com/ - acessado em 21/02/2012.

3. ProNet

O ProNet é um *framework* que permite o desenvolvimento de soluções para gerenciamento autonômico em uma rede de computadores. Este é composto de funcionalidades que utilizam plugins no Protégé, as linguagens OWL (*Web Ontology Language*) e SWRL (*Web Semantic Rule Language*), e a ferramenta de inferência Jess. O ProNet possui a ontologia OWL-Net como um dos hotspot (ponto flexível) do *framework*. Esta permite descrever a topologia de redes, o que inclui detalhamentos de hosts, links, portas e status desses. Outro *hotspot* é WrapperNet, que tem a função de enviar os comandos de acordo com os parâmetros atualizados da ontologia através de uma comunicação por *socket*. Tais comandos podem criar nós, ativar ou desativar suas interfaces de comunicação com a rede.

No *framework* ProNet, o gerenciamento é feito através de regras definidas através da linguagem SWRL, cuja estrutura é uma combinação de sublinguagens do OWL (OWL DL e OWL Lite). Para a criação destas regras todas as condições definidas na ontologia desenvolvida são analisadas e situações conflitantes e concorrentes são rejeitadas. Pois uma regra ativa não pode ser contrária à outra já definida. Com isto verificamos que uma ontologia bem definida resolve automaticamente grandes problemas que falhas humanas poderiam permitir, como apresentado no trabalho (DAVY *et al.*, 2008). Após a definição das regras uma ferramenta de inferência, o Jess, associado ao SWRL analisa estas regras criadas e realiza as modificações necessárias nos parâmetros da ontologia para que a mesma seja efetivada. A máquina de inferência Jess, no Protégé, trabalha em conjunto com o *plugin* da linguagem SWRL, ou seja, na mesma aba. E são *softwares* que tem a função de gerar conhecimento a partir de um conhecimento já desenvolvido. Quando na Web Semântica descrevemos um conhecimento utilizando uma linguagem apropriada, podemos através de máquinas de inferência gerar outros conhecimentos, relativos ao primeiro, a partir de regras estabelecidas. E para toda regra criada e ativada é possível realizar a inferência que resultará na modificação da ontologia.

Uma primeira solução desenvolvida com o ProNet define uma arquitetura integrada com um simulador de redes OMNet++[8] de acordo com (OLIVEIRA *et al.*, 2008). Esta solução foi avaliada em um cenário composto de nove hosts, sendo inseridas algumas regras utilizando o *plugin* do SWRL, as quais bloqueiam o uso da rede para alguns destes hosts. Para que estas regras modifiquem os parâmetros da ontologia criada, a ferramenta de inferência Jess é utilizada em conjunto com o SWRL. Então, vê-se nesta proposta inicial um ambiente para realização de testes do sistema ProNet em uma simulação, onde é possível testar as regras e o comportamento do sistema criado para posterior aplicação em uma situação real.

4. OpenFlow

Esta é uma arquitetura criada pela universidade de Stanford para permitir o uso das redes em produção dos Campi Universitários para análise de novos serviços e protocolos sem interferir no funcionamento da rede em atividade. Isso porque esta arquitetura tornou possível a separação do plano de controle e do plano de dados, que até então ambos estavam integrados nos equipamentos de rede por meio de códigos proprietários e fechados. O plano de controle é a camada de *software* programável por desenvolvedores e o plano de dados corresponde ao *hardware* que executa o comportamento definido pela camada de controle. Dessa forma, o OpenFlow foi desenvolvido para oferecer aos desenvolvedores um ambiente real para as avaliações de seus códigos que fazem uso da rede, que até então estavam restritos aos simuladores.

Descrito em (MCKEOWN *et al.*, 2008), o OpenFlow é constituído pelos elementos: *Controller* que está no plano de controle; equipamento gerenciável com tabelas de fluxos que estão no plano de dados e protocolos OpenFlow. Para realizar a tarefa de gerenciamento, a *controller* gerencia de maneira centralizada as tabelas de fluxos existentes no plano de dados dos switches, onde ficam os registros das portas de origem e destino dos fluxos para encaminhamento de dados já instalados. A comunicação entre a *Controller* e os switches é realizada através do OpenFlow *Protocol* (OFP) que é utilizado para reprogramar o controle do tráfego.

[8] http://www.omnetpp.org/ - acessado em 21/02/2012.

Atualmente vários fabricantes de dispositivos de rede (Cisco, Juniper, IBM, etc.) estão oferecendo em seus equipamentos a compatibilidade com o OFP, que é Open source, o que demonstra sua aceitação e potencial. Para que o OpenFlow seja atuante em uma rede, é necessária a presença da *Controller*, elemento externo ao sistema, e que hoje oferece várias soluções candidatas, tais como: Beacon[9], Floodlight[10], NOX (GUDE *et al.*, 2008) e Trema[11].

A *Controller Beacon* é baseada na linguagem multiplataforma Java e tem uma razoável maturidade e oferece como vantagem para os desenvolvedores a integração com a IDE Eclipse. A *Controller Floodlight* é uma bifurcação da *Controller Beacon*, contudo é suportada pela empresa Big Switch Networks que fornece uma *Controller* comercial baseada nesta plataforma, apesar do *Floodlight* ter o código aberto. Contudo, tal *Controller* foi disponibilizada recentemente (Dez 2011). A *Controller* NOX é a mais madura entre as *Controllers*, foi desenvolvida utilizando a linguagem C++ e permite a utilização da linguagem Python, o que a torna bem amigável para o desenvolvimento de aplicações para esta plataforma. E finalmente a *Controller* Trema tem como principal característica a possibilidade de utilização da linguagem Ruby, a plataforma foi desenvolvida em C e não tem maturidade.

5. Proposta

O PolicyFlow tem como objetivo permitir a operacionalidade do uso de semânticas para gerenciar as redes de computadores. Para isso, definiu-se a arquitetura da Figura 1 composta do Protégé que oferece a facilidade de extensão por meio de *plugins* e a interconexão entre NOX e Openflow que são os mecanismos mais referenciados nesta área de configuração dinâmica de elementos de redes. Utilizando essas ferramentas, esta arquitetura permite a edição de semânticas descritas com a linguagem OWL, a criação de regras usando SWRL e a descobertas de novas informações que são extraídas por meio de inferência de regras utilizando o Jess. Os resultados da inferência geram comandos que são passados para o NOX (*controller*) que as traduz em comandos a serem executados na rede OpenFlow e dessa forma reprogramar os equipamentos de redes de acordo com as regras definidas pelo administrador da rede.

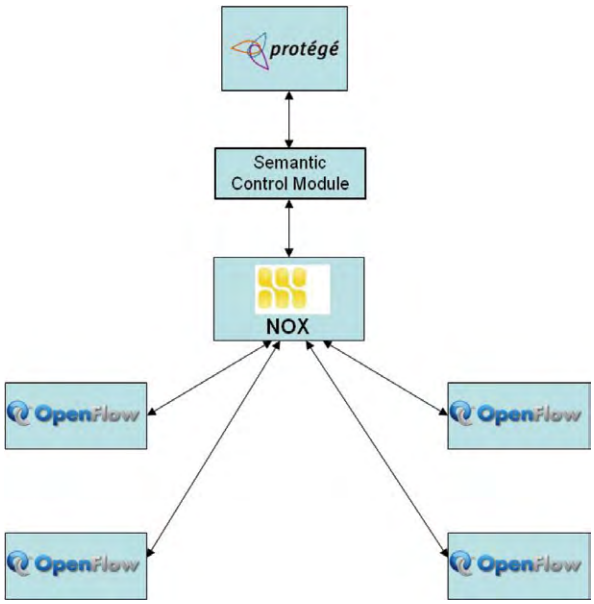

Figura 1. Conexões entre os sistemas.

Esta arquitetura foi integrada ao *framework* ProNet, pois já desenvolveu um *plugin* no Protégé para gerenciar redes de computadores. A intenção foi adicionar ao ProNet a possibilita de interação deste *framework* com uma rede real, avançando do ambiente de simulação para um ambiente real e fazendo uso de uma tecnologia inovadora e em ascensão, o OpenFlow. Com isto estaremos chegando cada vez mais próximo da implantação em uma rede operacional, fazendo os recursos da Web Semântica serem explorados em ambientes de gerenciamento de redes de computadores. Entre tais recursos está a definição da rede através de ontologia, que permite por um lado o entendimento de uma forma mais simples o funcionamento e relações entre os elementos que a compõe, e por outro, como é característica da ontologia, evita a criação de definições conflitantes ou concorrentes. Desta forma busca-se permitir um gerenciamento mais simples e de fácil entendimento, como também mais estável e livre de erros.

Para se alcançar esta integração, utilizou-se a *Controller* NOX, que dentre as opções rela-

9 https://openflow.stanford.edu/display/Beacon/Home - acessado em 29/02/2012.

10 http://floodlight.openflowhub.org/ - acessado em 29/02/2012.

11 http://trema.github.com/trema/ - acessado em 29/02/2012.

tadas na seção anterior tem como principal vantagem a sua maturidade e ampla utilização, além da possibilidade de uso da linguagem Python, que demonstra ter um grande potencial para o desenvolvimento de módulos para tal plataforma. Utilizamos o Python no desenvolvimento do módulo Semantic Control para o NOX, que faz esta integração. Após a modificação da ontologia pela ferramenta de inferência, conforme relatado anteriormente, é necessário, para que a ontologia interfira no funcionamento da rede, que o Wrapper-Net envie comandos para a *Controller*/NOX. Para isto o WrapperNet estabelece uma conexão socket com o módulo *Semantic Control* que recebe os comandos do ambiente semântico e os traduz em comandos do OpenFlow utilizando a estrutura fornecida pelo NOX. A arquitetura PolicyFlow foi desenvolvida para controlar os fluxos de dados de todos os switches que compõe a rede, permitindo ou não o encaminhamento de dados de acordo com as políticas definidas pelo ProNet. Na Figura 2 é demonstrado como está estruturado o ProNet, onde se visualiza que a linguagem para definição da ontologia (OL), de suas regras (OL-Rule) e o Wrapper-Net fazem parte do Editor de Ontologia. Tal estrutura formada dentro do Editor se comunica com a ferramenta de inferência, que por sua vez responde com modificações na ontologia. Entre o ambiente de ontologia e o ambiente de rede é estabelecida uma comunicação entre o Wrapper-Net e o Módulo *Semantic Control*, este último sendo parte da *Controller* do sistema OpenFlow, interage com a rede para a execução das regras definidas no ambiente de ontologia. Desta forma demonstramos como reproduzimos as regras definidas em SWRL no ProNet em ações no comportamento dos fluxos de uma rede controlada pelo uso do sistema OpenFlow.

6. Codificação da solução

Esta seção tem como objetivo mostrar os principais trechos de código do módulo *Semantic Control*, componente fundamental para a integração do ambiente semântico com o ambiente de Rede OpenFlow. Como descrito anteriormente, este tem a função de receber conexões por *socket* realizadas pelo Wrapper-Net, coletar os comandos advindos desta conexão, coletar as informações importantes destes comandos e executá-los na rede usando recursos do NOX/OpenFlow. O trecho de código abaixo contém o método *executePolicyCommand()* que tem a função de analisar o comando enviado pelo Wrapper-Net por meio da inserção de comandos em uma estrutura de fila. O comando é composto das seguintes informações: ação e equipamento.

Quando o módulo *Semantic Control* recebe o comando com a ação referente à desconexão, a tupla <desconexão, equipamento> é inserida na estrutura de lista de negação de acesso *denyList*. Caso seja recebido um comando com ação referente à permissão de acesso, a tupla <desconexão, equipamento> é removida da estrutura *denyList*. Utiliza-se os métodos *findCommand()* e *findNode()*, descritos abaixo, que auxiliam na detecção da ação a ser tomada.

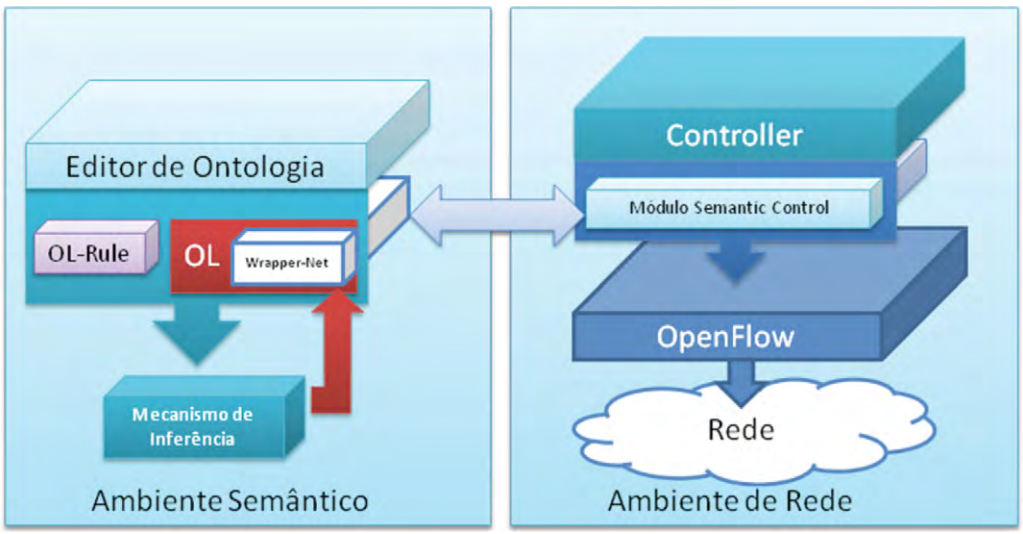

Figura 2. Framework ProNet com o módulo Semantic Control.

```
def executePolicyCommand():
    global denyList
    global nodeDict
    while data :
        fullText = data.pop(0)
        Command,Rest = findCommand(fullText)
        Node = findNode(Rest)
        if Node <> "cloud" :
            if Command == "disconnect" :
                if nodeDict[Node] not in
denyList:
                    denyList.
append(nodeDict[Node])
                    print denyList
            else:
                if Command == "connect" and
nodeDict[Node] in denyList:
                    denyList.
remove(nodeDict[Node])
```

No método *findCommand()*, abaixo, é realizada a identificação do tipo de comando recebido pelo módulo Semantic Control. Neste cenário existem dois tipos: conexão e desconexão.

```
def findCommand(fullText):
    if fullText <> "":
        i = 0
        command = ""
        while fullText[i] <> " ":
            command += fullText[i]
            i += 1
        rest = fullText[i:]
        return command, rest
    else:
        return None
```

Já no método *findNode()*, abaixo, é realizada a busca do equipamento associado ao comando que foi recebido no formato de String. Este método é utilizado pelo *executePolicyCommand()* para aplicar uma restrição ou permissão de acesso.

```
def findNode(restText):
    node = ""
    if restText <> "":
        i = 1
        while restText[i] <> ":":
            node += restText[i]
            i += 1
        return node
    else:
        return None
```

Finalmente, abaixo está exposto o código do Método *packet_in_callback()*. Este método tem, a princípio, no funcionamento de um switch a definição do comportamento dos elementos controlados pelo PolicyFlow. Tal comportamento é mandatório se não houver uma política definida no ambiente semântico do PolicyFlow que seja divergente. Neste caso as políticas têm precedência sobre o comportamento dos elementos componentes da rede. Quando o método *executePolicyCommand()* insere alguma tupla <ação, equipamento> na tabela denyList este método retira os fluxos referentes a este equipamento nas tabelas de fluxos dos elementos controlados pelo PolicyFlow, como também não permite a inserção de novos fluxos deste equipamento enquanto o mesmo estiver na *denyList*. Para isto chama os métodos *forward_l2_packet() e not_foward_l2_packet()*.

```
def packet_in_callback(dpid, inport, reason,
len, bufid, packet):
    global denyList
    if not packet.parsed:
        log.msg('Ignoring incomplete
packet',system='PyPolicyFlow')

    if not inst.st.has_key(dpid):
        log.msg('registering new switch %x' %
dpid,system='PyPolicyFlow')
        inst.st[dpid] = {}

    # don't forward lldp packets
    if packet.type == ethernet.LLDP_TYPE:
        return CONTINUE

    # learn MAC on incoming port
    do_l2_learning(dpid, inport, packet)

    flow = extract_flow(packet)
    executePolicyCommand()
    if mac_to_str(flow['dl_src']) in denyList :

not_forward_l2_packet(dpid,inport,packet)
    else :
        forward_l2_packet(dpid, inport, packet,
packet.arr, bufid)
```

Na página seguinte está o código do método *forward_l2_packet()* que instala os fluxos para os tráfegos permitidos nos elementos controlados pelo PolicyFlow.

```
def forward_l2_packet(dpid, inport, packet,
buf, bufid):
    dstaddr = packet.dst.tostring()
    if not ord(dstaddr[0]) & 1 and inst.
st[dpid].has_key(dstaddr):
        prt = inst.st[dpid][dstaddr]
        if  prt[0] == inport:
            log.err('**warning** learned port =
inport', system="PyPolicyFlow")
            inst.send_openflow(dpid, bufid, buf,
openflow.OFPP_FLOOD, inport)
        else:
            # We know the outport, set up a flow
            log.msg('installing flow for ' +
str(packet), system="PyPolicyFlow")
            flow = extract_flow(packet)
            flow[core.IN_PORT] = inport
            actions = [[openflow.OFPAT_OUTPUT,
[0, prt[0]]]]
            inst.install_datapath_flow(dpid,
flow, CACHE_TIMEOUT, openflow.OFP_FLOW_PERMANENT,
actions,bufid,openflow.OFP_DEFAULT_PRIORITY,
inport, buf)
    else:
        # haven't learned destination MAC.
Flood
        inst.send_openflow(dpid, bufid, buf,
openflow.OFPP_FLOOD, inport)
```

E com a função contrária ao código anterior o método *not_forward_l2_packet()* não permite novas conexões e retira o registro de tráfegos nas tabelas de fluxos dos equipamentos controlados pelo PolicyFlow.

```
def not_forward_l2_packet(dpid, inport, packet):
    #not permit installing flow
    log.msg('deleting flow for' + str(packet),
system="PyPolicyFlow")
    flow = extract_flow(packet)
    flow[core.IN_PORT] = inport
    inst.delete_datapath_flow(dpid,flow)
```

7. Avaliação

O trabalho de (LANTZ *et al.*, 2010) apresenta a emulação de rede com hosts e switches OpenFlow utilizando a ferramenta Mininet, que foi desenvolvida para utilização de experimentos de elementos OpenFlow. Esta ferramenta permite a criação de uma rede com todos os seus elementos configuráveis utilizando apenas um computador. Isso permite avaliar um comportamento idêntico a uma rede física, sendo possível que a codificação, utilizada na emulação, possa ser aplicada diretamente em um ambiente real.

Dessa forma, escolheu-se o uso do MiniNet e a mesma ontologia desenvolvida no trabalho (OLIVEIRA *et al.*, 2008). Entretanto foi definido um novo cenário composto de novas regras. Neste novo cenário, a ontologia possui a descrição de um ambiente de sala de aula com a finalidade do professor ser capaz de aplicar regras de bloqueio de uso da rede pelos equipamentos utilizados por seus alunos com o intuito de facilitar a busca da total atenção dos alunos ao conteúdo que está sendo apresentado. Para tal, o professor pode ativar regras semânticas em SWRL que atuem no comportamento da rede, e não permitir que as máquinas dos alunos consigam fazer uso da rede na sala de aula. A tabela 1 está apresentando duas regras desenvolvidas para este fim. Na Regra 1 os estudantes que utilizarem suas máquinas (computadores) para conectar à rede terá esta conexão negada (state DOWN). Ou seja, no momento em que houver tráfego proveniente das máquinas dos estudantes, este tráfego será bloqueado. Já na Regra 2 está definida a situação inversa, que deve ser utilizada quando da liberação dos computadores dos estudantes para acesso aos recursos da rede, tendo o status de sua conexão alternada para "UP".

Mudando o ambiente de rede (*Network environment*) do trabalho (OLIVEIRA *et al.*, 2008), agora estaremos utilizando o conjunto NOX/OpenFlow/Mininet no lugar do simulador Omnet++. Após a aplicação das regras, o Jess realiza o processamento e alterações na ontologia e os resultados desse processamento atuam no status dos links dos hosts emulados pelo Mininet.

Isso porque os resultados do processamento pelo Jess são transformados em comandos enviados pelo WrapperNet. Estes comandos exemplificados na tabela 2 são traduzidos em comandos compreensíveis pelo MiniNet por meio de uma conexão *socket* estabelecida. O *Semantic Control* realizou com precisão a tradução para os comandos do OpenFlow através da *Controller* NOX.

Estas operações foram realizadas de forma dinâmica, ou seja, com a rede emulada em pleno funcionamento, e produziu os efeitos esperados no comportamento do tráfego da rede. Tanto na ativação de regras quanto no seu cancelamento. Durante os testes verificamos a eficácia da implementação, que demonstrou reproduzir o comportamento esperado para cada regra criada.

Tabela 1. Regras SWRL.

Regra 1	Regra 2
Student(?human) ∧ networkConnectFrom(?human, ?machine) ∧ hasSimple(?machine, HTTPClient) ∧ Connection(?connection) ∧ hasSource(?connection, ?machine) ∧ hasDestination(?connection, cloud) → hasState(?connection, "DOWN")	*Student(?human) ∧ networkConnectFrom(?human, ?machine) ∧ hasSimple(?machine, HTTPClient) ∧ Connection(?connection) ∧ hasSource(?connection, ?machine) ∧ hasDestination(?connection, cloud) → hasState(?connection, "UP")*

Tabela 2. Exemplos de comandos do ProNet.

Comandos de desconexão	Comandos de conexão
disconnect cloud:out0:3,client1:in:-1	*connect cloud:out0:5,client3:in:0*
disconnect client1:out:-1,cloud:in0:3	*connect client3:out:-1,cloud:in0:5*

8. Conclusão e trabalhos futuros

Com a obtenção dos resultados do presente trabalho, verificou-se que esta proposta denominada de PolicyFlow tem potencial para sua implantação em soluções que requeiram a manipulação, extração e associação de informações. O cenário de avaliação demonstrou que o PolicyFlow possibilita o gerenciamento de redes utilizando regras definidas com as facilidades da Web Semântica, e oferece uma interface de definição de regras mais amigável como vantagens para a administração de redes. Um dos trabalhos futuros será a especificação da integração e isolamento de informações referentes a equipamentos de vários fabricantes. Também se pretende definir um ambiente mais complexo onde se utilizarão outros parâmetros dos fluxos de dados providos pelo OpenFlow para um controle mais específico do tráfego. Além disso, a intenção será aplicar o PolicyFlow em uma rede real e adicionar mais recursos a essa arquitetura visando sua atuação de forma autonômica.

9. Referências

DAVY, S.; JENNINGS, B.; STRASSNER, J. Using an Information Model and Associated Ontology for Selection of Policies for Conflict Analysis. **2008 IEEE Workshop on Policies for Distributed Systems and Networks**, n. 1, p. 82-85, jun 2008.

FITZGERALD, W. M.; FOLEY, S. **Management of heterogeneous security access control configuration using an ontology engineering approach**. Proceedings of the 3rd ACM workshop on Assurable and usable security configuration. **Anais**... New York, New York, USA: ACM. Disponível em: <http://portal.acm.org/citation.cfm?doid=1866898.1866903>. Acesso em: 22 dez. 2011. , 2010

GUDE, N.; KOPONEN, T.; PETTIT, J. *et al.* NOX: towards an operating system for networks. **ACM SIGCOMM Computer Communication Review**, v. 38, n. 3, p. 105–110, 2008.

KEENEY, J.; LEWIS, D.; O'SULLIVAN, D. *et al.* **Runtime semantic interoperability for gathering ontology-based network context**. Network Operations and Management Symposium, 2006. NOMS 2006. 10th IEEE/IFIP. **Anais**... [S.l.]: IEEE. Disponível em: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1687538>. Acesso em: 2 jan. 2012. , 2006

LANTZ, B.; HELLER, B.; MCKEOWN, N. **A network in a laptop: Rapid prototyping for software-defined networks**. Proceedings of the Ninth ACM SIGCOMM Workshop on Hot Topics in Networks. **Anais**... [S.l.]: ACM. Disponível em: <http://dl.acm.org/citation.cfm?id=1868466>. Acesso em: 3 fev. 2012. , 2010

MCKEOWN, N.; ANDERSON, T.; BALAKRISHNAN, H. *et al.* OpenFlow: enabling innovation in campus networks. **ACM SIGCOMM Computer Communication Review**, v. 38, n. 2, p. 69–74, 2008.

MORAES, P. S.; SAMPAIO, L. N.; MONTEIRO, J. A. S.; PORTNOI, M. **Mononto: A domain ontology for network monitoring and recommendation for advanced internet applications users**. Network Operations and Management Symposium Workshops, 2008. NOMS Workshops 2008. IEEE. **Anais**... [S.l.]: IEEE. Disponível em: <http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4509937>. Acesso em: 20 dez. 2011. , 2008.

OLIVEIRA, L.; ABREU, R.; ALBUQUERQUE, H.; SADOK, D. **ProNet: An Ontology Simulation Framework for Autonomic Networks**. III Latin American Autonomic Computing Symposium. **Anais**... Gramado: [s.n.]. Disponível em: <http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:ProNet:+An+Ontology+Simulation+Framework+for+Autonomic+Networks#0>. Acesso em: 30 dez. 2011. , 2008

ROSSELLO-BUSQUET, A; BREWKA, L. J.; SOLER, J.; DITTMANN, L. OWL Ontologies and SWRL Rules Applied to Energy Management. **2011 UkSim 13th International Conference on Computer Modelling and Simulation**, p. 446-450, mar 2011.

SEGALL, B.; ARNOLD, D.; BOOT, J.; HENDERSON, M.; PHELPS, T. **Content based routing with elvin4**. Proceedings of AUUG2K. **Anais**... [S.l: s.n.]. Disponível em: <http://books.google.com/books?hl=en&lr=&id=LrYoRLIEraEC&oi=fnd&pg=PA55&dq=Content+Based+Routing+with+Elvin4&ots=RYZsgngwHa&sig=4IosITw_k4VcX3C5W37JB33yCZY>. Acesso em: 28 fev. 2012. , 2000

VISWANATHAN, A.; HUSSAIN, A.; MIRKOVIC, J. A semantic framework for data analysis in networked systems. **on Networked Systems**, p. 127–140, 2011.

XU, H.; XIAO, D. **A common ontology-based intelligent configuration management model for ip network devices**. Innovative Computing, Information and Control, 2006. ICICIC'06. First International Conference on. **Anais**... [S.l.]: IEEE. Disponível em: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1691820>. Acesso em: 29 dez. 2011. , 2006

XU, H.; XIAO, D. Applying Semantic Web Services to Automate Network Management. **2007 2nd IEEE Conference on Industrial Electronics and Applications**, p. 461-466, maio 2007.