UNIVERSIDADE FEDERAL DE GOIÁS
INSTITUTO DE INFORMÁTICA

VINÍCIUS GONÇALVES BRAGA

Avaliação das funcionalidades do Broker e realização de testes fim-a-fim em recursos OpenFlow e sem fio com o OMF 6

Goiânia
2016

Sumário

Lista de Figuras **2**

Lista de Tabelas **3**

1 Introdução **4**

2 Ambiente de testes **5**
- 2.1 Principais componentes 5
- 2.2 Configuração do ambiente 6
- 2.3 Dificuldades e interação com o NITlab 9

3 Lições Aprendidas **10**
- 3.1 Experimentos fim-a-fim 10
  - 3.1.1 Experimento com nós sem fio 10
  - 3.1.2 Experimento com tecnologia OpenFlow 11
- 3.2 Investigação do Broker 11
  - 3.2.1 Arquitetura do Broker 11
  - 3.2.2 Experimentos realizados com o Broker 14
- 3.3 Considerações gerais 16

4 Conclusão e Avaliação **18**

Referências Bibliográficas **19**

A Controle de Recursos Básicos de uma Testbed Usando OMF 6 **20**

B Controle de recursos OpenFlow usando OMF 6 **22**

C Avaliação das funcionalidades do Broker e realização de testes fim-a-fim em recursos OpenFlow e sem fio com o OMF 6 **24**

Lista de Figuras

2.1 Modelo de implantação da *testbed*. 6
2.2 Modelo de implantação da *testbed*. 7

3.1 Cenário do experimento com nós sem fio. 10
3.2 Arquitetura do Broker. 12
3.3 Experimento com o *script omf6*. 15
3.4 Criação de um *sliver* no Broker, com concessão de acesso ao ICARUS 5. 16

A.1 Componentes que integram a arquitetura do OMF 5.4. 21

Lista de Tabelas

2.1 Configuração dos computadores utilizados na *testbed*. 7
2.2 Lista de programas instalados nas máquinas virtuais do Computador 1. 8
2.3 Lista de programas instalados no Computador 2 e nas máquinas virtuais hospedadas nesse computador. 8
2.4 Lista de programas instalados no Computador 3. 9

CAPÍTULO 1

Introdução

Durante a execução dos microprojetos “Controle de recursos OpenFlow usando o OMF 6”[1] e “Controle de recursos básicos de uma *testbed* usando OMF 6”’[2], realizamos a implantação de uma *testbed* funcional do OMF 6 e a avaliação do uso dos Resource Controllers (RCs) básicos e dos RCs relacionados a OpenFlow. A avaliação dos RCs foi feita isoladamente, comprovando suas funcionalidades e a capacidade do OMF 6 de controlar diferentes tipos de recursos. Contudo, não realizamos testes fim-a-fim para verificar a capacidade do OMF 6 em experimentos mais avançados, controlando recursos OpenFlow e sem fio.

Um outro aspecto não investigado a fundo durante a primeira etapa dos microprojetos foi o Broker. Ele é um componente que age como um Aggregate Manager (AM) e, dentre outras funcionalidades, permite cadastrar e requisitar informações sobre recursos, criar contas de usuários e, ainda, reservar recursos. O Broker oferece uma camada de comunicação com interfaces REST, FRCP e SFA. Durante a primeira etapa, utilizamos a interface FRCP para cadastrar e requisitar informações de recursos e a interface REST para visualizar os recursos cadastrados. Isso nos permitiu a execução de testes com o *script omf6*, o qual é responsável por interagir com os Chassi Managers (CMs) e por carregar e salvar imagens nos nós ICARUS. Contudo, não realizamos testes como criação de contas e reserva de recursos, que podem ser feitas utilizando as interfaces SFA e REST.

As pendências citadas foram objeto de investigação nesta segunda etapa dos microprojetos. Durante essa fase de extensão, nós realizamos experimentos mais complexos com o OMF 6 e verificamos sua viabilidade para controlar diferentes tipos de recursos de maneira simultânea. Investigamos também a arquitetura do Broker e validamos suas funcionalidades por meio de testes, os quais exercitaram todas as interfaces de comunicação do arcabouço.

Neste relatório, apresentamos, no Capítulo 2, o ambiente utilizado para implantação da *testbed*, mostrando os componentes de *software* instalados para a avaliação do OMF 6 e do Broker. Em seguida, no Capítulo 3, descrevemos os testes fim-a-fim realizados e suas limitações, apresentamos a arquitetura do Broker e mostramos os testes realizados para avaliar as funcionalidades implementadas pela equipe do NITlab [6]. Por fim, no Capítulo 4, apresentamos a conclusão e a avaliação que fazemos do OMF 6 e do Broker.

[1] Proposta apresentada no Apêndice A
[2] Proposta apresentada no Apêndice B

CAPÍTULO 2

Ambiente de testes

Neste capítulo, apresentamos os detalhes da configuração da *testbed* criada para a avaliação do OMF 6. Inicialmente, descrevemos alguns componentes importantes da *testbed* (Seção 2.1) e, em seguida, mostramos como o ambiente foi organizado (Seção 2.2). Por fim, apresentamos as dificuldades encontradas durante a criação da *testbed* e como a interação com a equipe do NITlab nos ajudou a superar essas dificuldades (Seção 2.3).

2.1 Principais componentes

Para se entender os detalhes da configuração do sistema, é importante conhecer o que cada componente da *testbed* faz. A seguir, descrevemos a função dos principais componentes.

- **FlowVisor** - Funciona como um *proxy* entre o contralador OpenFlow e o *switch* OpenFlow e é capaz de virtualizar o recurso do *switch* por meio de *slices*, permitindo que diferentes experimentadores compartilhem o mesmo *switch*. Nesse caso, apesar do equipamento ser compartilhado, cada experimentador visualiza seu próprio *switch*, como se estivessem acessando o equipamento diretamente.
- **Open vSwitch** - Programa que emula em *software* um *switch* OpenFlow.
- **POX** - Controlador OpenFlow.
- **Xen e KVM** - Ambos são hipervisores e permitem a virtualização do *hardware* para a hospedagem de diferentes máquinas virtuais e, possivelmente, com diferentes sistemas operacionais.
- **RabbitMQ** - Servidor AMQP. AMQP é um protocolo para *middlewares* orientados a mensagens e seu uso é recomendado para comunicação dos módulos do OMF 6.
- **OpenFire** - Servidor XMPP. O XMPP também um protocolo para *middlewares* orientados a mensagens e também pode ser utilizado no OMF 6. Contudo, o AMQP é mais estável e se mostrou mais eficiente nos testes realizados pelos desenvolvedores do OMF [4].
- **OMF Experiment Description Language (OEDL)** - Linguagem utilizada para descrever experimentos do OMF.
- **Experiment Controller (EC)** - Interpreta e executa um experimento escrito em OEDL, comunicando-se com os recursos por meio de um servidor *Publish/Subscribe*.
- **Resource Controller (RC)** - Oferece funções para controlar, configurar e requisitar informações de um recurso. Recebe instruções e envia informações através do servidor *Publish/Subscribe*. A API do OMF oferece uma abstração para controlar qualquer tipo de recurso, seja ele um recurso de *hardware* ou de *software*, bastando para isso, que um Resource Controller apropriado seja implementado.
- **Broker** - Módulo responsável pela descoberta, reserva e provisionamento de recursos.
- **omni** - Cliente SFA de linha de comando.
- **gcf-ch** - Implementação de referência da *clearinghouse* do GENI.

2.2 Configuração do ambiente

Para a realização da avaliação do OMF 6, implantamos uma *testbed* no LABORA utilizando 3 computadores, um *switch* e dois nós ICARUS. A Figura 2.1 apresenta a organização física do ambiente, mostrando as principais conexões entre os diferentes equipamentos. Configuramos 3 VLANs no *switch* físico, sendo a VLAN Internet alocada para conexão à rede do LABORA e à *Internet*; a VLAN Control utilizada para o controle dos recursos da *testbed*; e a VLAN CM utilizada para o comunicação com os Chassis Manager (CMs) dos nós ICARUS.

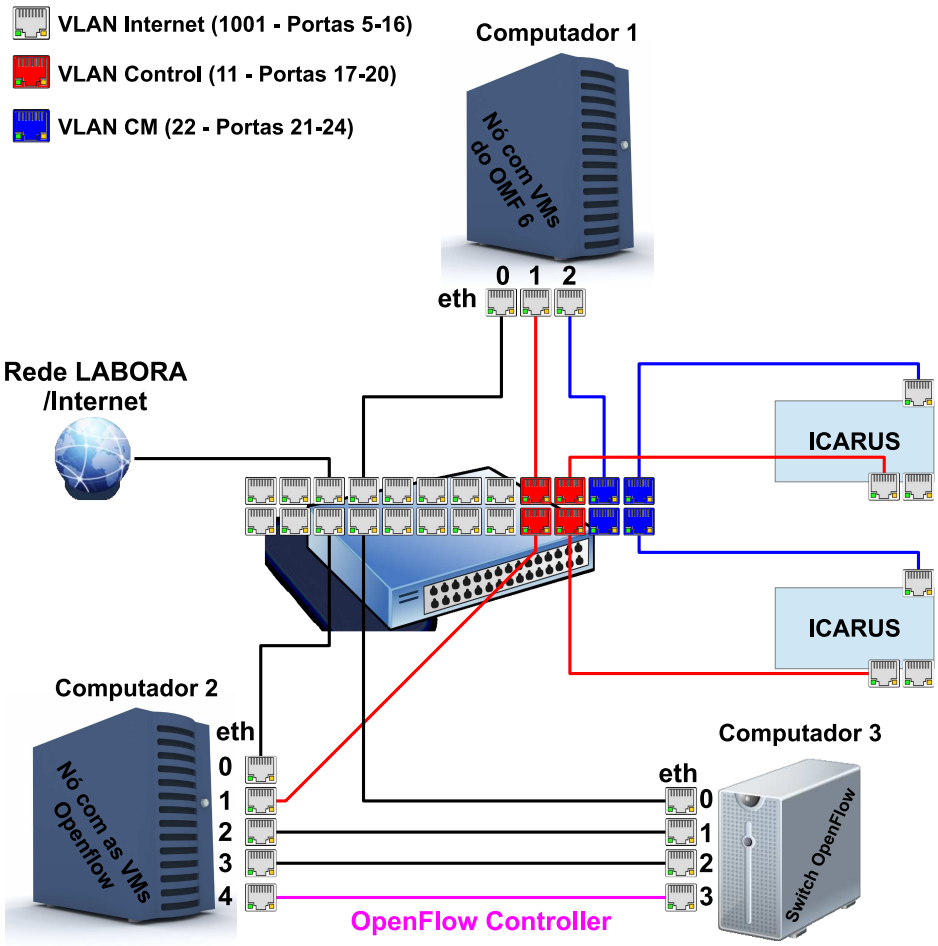

Figura 2.1: *Modelo de implantação da testbed.*

O Computador 1 (C1) é uma máquina com processador i7, com 8 GB de memória RAM e 3 placas de rede. No C1, assim como nos demais computadores, foi instalado o sistema operacional Ubuntu 14.04 LTS. Além disso, instalamos, no C1, o hipervisor Xen 4.4 para criação de duas máquinas virtuais com módulos do OMF 6 instalados. O Computador 2 (C2) possui configurações similares ao C1, porém foi equipado com 5 placas de rede. Instalamos o hipervisor KVM 2.0.0 no C2 a fim de criar máquinas virtuais para avaliação dos RCs relacionados ao controle de hipervisores e também para possibilitar os testes com a tecnologia OpenFlow. O Computador 3 (C3) é uma máquina com processador i5, 4 GB de memória RAM e 4 placas de rede. No C3, instalamos o Open vSwitch 2.0.2 e o FlowVisor 1.4, no intuito de avaliar os RCs para OpenFlow. A Tabela 2.1 mostra um resumo da configuração de cada um dos computadores.

Máquina	Processador	Memória	SO	Hipervisor	Placas de Rede
C1	Intel(R) Core(TM) i7-3770 CPU @ 3.40GHz	8 GB	Ubuntu 14.04	Xen 4.4	3 placas
C2	Intel(R) Core(TM) i7-3770 CPU @ 3.40GHz	8 GB	Ubuntu 14.04	KVM 2.0.0	5 placas
C3	Intel(R) Core(TM) i5-3470 CPU @ 3.20GHz	4 GB	Ubuntu 14.04	—	4 placas

Tabela 2.1: *Configuração dos computadores utilizados na testbed.*

A Figura 2.2 apresenta um modelo mais completo do ambiente, mostrando todas as conexões, pontes e máquinas virtuais criadas para possibilitar a execução dos testes. As linhas tracejadas, ligando algumas máquinas à rede do LABORA, representam conexões utilizadas apenas para acesso via *ssh* e para conexão à Internet. Essas conexões são importantes durante a fase de configuração e instalação dos pacotes nas máquinas e também para acesso aos *logs* dos RCs. Contudo, não são necessárias para a execução dos testes, uma vez que o experimentador precisa ter acesso apenas ao Experiment Controller e ao Broker. Os detalhes de instalação de cada um dos computadores serão explicados a seguir.

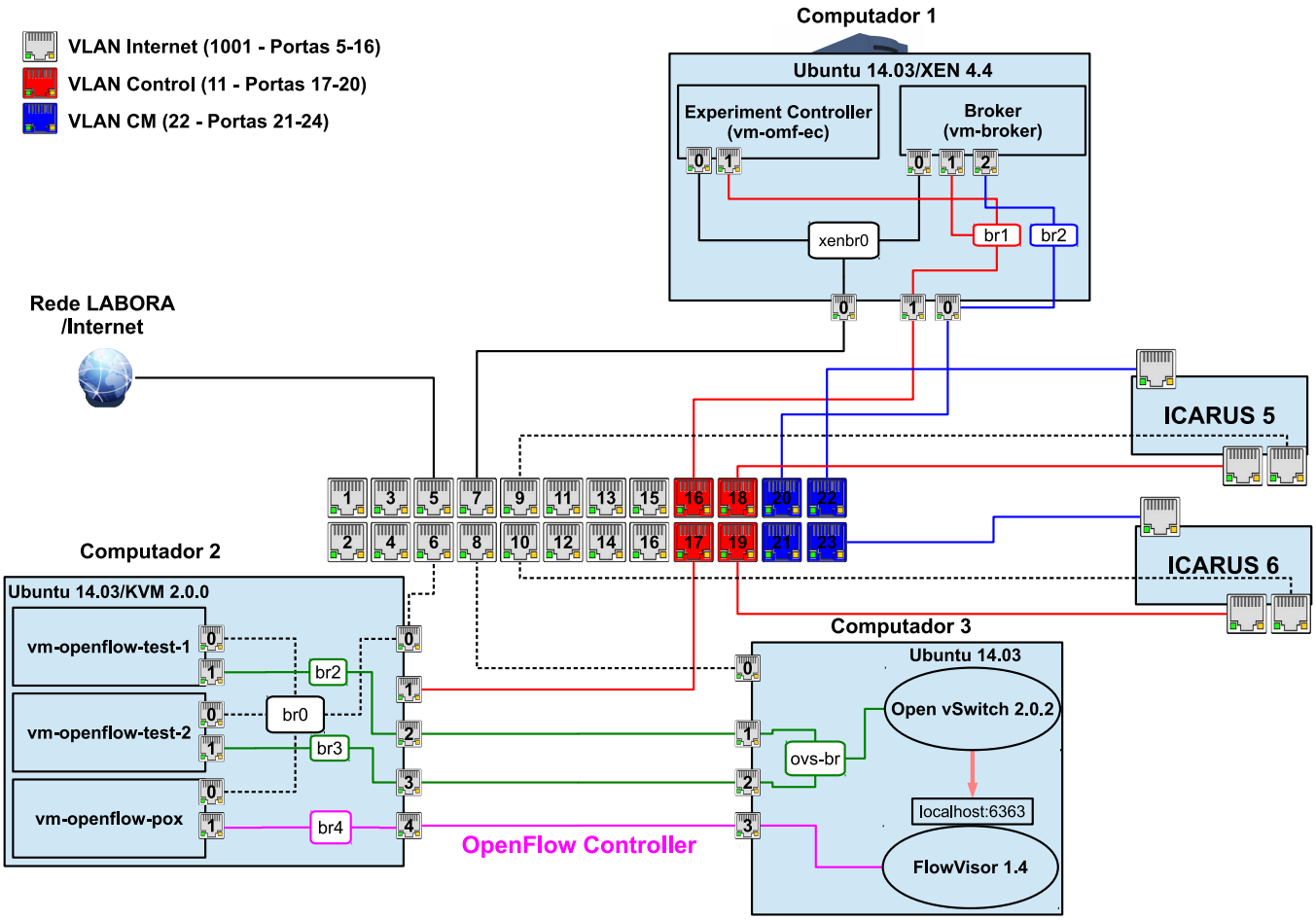

Figura 2.2: *Modelo de implantação da testbed.*

No C1, foram criadas duas máquinas virtuais: (1) *vm-omf-ec*, na qual instalamos o EC, o *gcf-ch* e o *omni*; e (2) *vm-broker*, onde instalamos o Broker, um servidor AMQP (RabbitMQ), um servidor XMPP

(OpenFire), e os RCs para interagir com o Broker. O EC se comunica com os servidores XMPP e AMQP através da rede da VLAN Control, como mostrado na Figura 2.2. Embora seja recomendado utilizar o protocolo AMQP no OMF 6, a implementação do Broker funciona apenas com XMPP até o momento e, por isso, houve a necessidade da instalação de um servidor XMPP. O *gcf-ch* e o *omni* foram instalados para possibilitar a comunicação com a interface SFA do Broker, permitindo a criação de contas e a reserva de recursos. A Tabela 2.2 mostra a lista dos programas instalados nas máquinas virtuais e uma descrição de sua funcionalidade.

Programa	**Descrição**	**Máquina**
Experiment Controller	Descrito na Seção 2.1	*vm-omf-ec*
gcf-ch	Descrito na Seção 2.1	*vm-omf-ec*
omni	Descrito na Seção 2.1	*vm-omf-ec*
RabbitMQ	Descrito na Seção 2.1	*vm-broker*
Openfire	Descrito na Seção 2.1	*vm-broker*
Nitos Testbed Resource Controllers (NTRC)	RCs de interação com o Broker para criação e carregamento de imagens e gerenciamento dos CMs	*vm-broker*

Tabela 2.2: *Lista de programas instalados nas máquinas virtuais do Computador 1.*

Na máquina C2, criamos 3 máquinas virtuais: (1) *vm-openflow-pox*, na qual foi instalado o controlador de *switch* OpenFlow POX; (2) *vm-openflow-test-1* e (3) *vm-openflow-test-2*, as quais foram conectadas às interfaces eth1 e eth2 da máquina C3. Essas interfaces foram configuradas como portas de um *switch* OpenFlow (o Open vSwitch), como mostrado na Figura 2.2. Instalamos também, na máquina física C2, o conjunto dos RCs padrões do OMF 6, o qual possui um RC para controlar o KVM. Nas máquinas virtuais (2) e (3), foi instalado o *iperf*, com o objetivo de testar a rede criada através do Open vSwitch. Um resumo dos programas instalados em C2 e nas máquinas virtuais hospedadas nesse computador é mostrado na Tabela 2.3.

Programa	**Descrição**	**Máquina(s)**
POX	Controlador OpenFlow. Configurado para se comunicar com o FlowVisor	*vm-pox*
Conjunto de RCs padrões	Contém os RCs básicos do OMF 6, dentre eles, o virtual_machine, responsável por controlar hipervisores. Por padrão, controla o KVM	C2
iperf	Ferramenta para teste de desempenho de redes	*vm-openflow-test-1* *vm-openflow-test-2*

Tabela 2.3: *Lista de programas instalados no Computador 2 e nas máquinas virtuais hospedadas nesse computador.*

Na máquina C3, instalamos o Open vSwitch 2.0.2 e deixamos, como já dito, as interfaces eth1 e eth2 alocadas para serem utilizadas como portas desse *switch*. Instalamos também o FlowVisor 1.4, deixando a interface eth3 alocada para a comunicação do FlowVisor com o POX, como ilustrado na Figura 2.2. Configuramos também o FlowVisor como sendo o controlador do Open vSwitch. Com essa

configuração, o POX controla o Open vSwitch através do FlowVisor. Para controle da parte OpenFlow, via EC, instalamos os RCs que controlam o Open vSwitch e o FlowVisor [1]. A Tabela 2.4, apresenta um resumo dos programas instalados em C3.

Programa	Descrição	Máquina
Open vSwitch	Descrito na Seção 2.1	C3
FlowVisor	Descrito na Seção 2.1	C3
RCs OpenFlow	virtual_openflow_switch - RC que controla o Open vSwitch openflow_slice - RC que controla o FlowVisor	C3

Tabela 2.4: *Lista de programas instalados no Computador 3.*

Nos ICARUS, foi necessário atualizar o *firmware*, uma vez que a versão anterior não funciona com o RC que controla os CMs. A nova versão do *firmware* foi fornecida pelo NITlab.

2.3 Dificuldades e interação com o NITlab

Durante a configuração do ambiente, as maiores dificuldades encontradas foram devido a falta de documentação, ou documentação desatualizada, dos diferentes programas que fazem parte da *testbed*. A documentação presente no site *mytestbed.com* [3] é muitas vezes desorganizada, insuficiente e não contempla informações sobre alguns módulos já desenvolvidos pela equipe do NITlab. O cliente SFA *omni* e o *gcf-ch* também possuem uma documentação desorganizada, a qual não oferece instruções adequadas para a configuração dos módulos.

O NITlab oferece tutoriais para instalação do Broker e dos RCs relacionados no GitHub, contudo os tutoriais apresentam alguns problemas. A instalação e configuração desses componentes, do *omni* e do *gcf-ch* dependeu de uma interação diária com os membros da equipe do NITlab, os quais foram bastante solícitos e ajudaram em todo o processo.

Lições Aprendidas

Na segunda fase dos microprojetos, conseguimos verificar a viabilidade do OMF 6 para executar experimentos mais complexos, envolvendo diferentes RCs e diferentes etapas em um mesmo experimento. A execução desses experimentos está descrita na Seção 3.1. Nesta etapa, investigamos também a arquitetura do Broker, exercitamos todas as suas interfaces de comunicação e utilizamos suas funcionalidades de criação de contas e reserva de recursos, através da interface SFA. Essa investigação é descrita na Seção 3.2.

3.1 Experimentos fim-a-fim

Nesta seção, descrevemos o conhecimento adquirido em relação as funcionalidades, limitações e problemas encontrados no OMF 6 durante a execução de cada um dos experimentos.

3.1.1 Experimento com nós sem fio

Neste experimento, utilizamos como referência o tutorial ”Hello World“ - Wireless [5] presente no portal do OMF 6. Para sua realização, utilizamos os dois nós ICARUS da *testbed*. No ICARUS 5, instalamos uma aplicação geradora de tráfego UDP, a ODG2. No ICARUS 6, instalamos uma aplicação para receber o tráfego gerado, a ODR2. Dessa forma, o ICARUS 5 é responsável por gerar o tráfego e o ICARUS 6 é responsável por recebê-lo. Tanto o ODG2 quanto o ODR2 são instrumentadas com OML e foram configurados para reportar as estatísticas sobre o tráfego enviado e recebido para o servidor OML instalado no Computador 2.

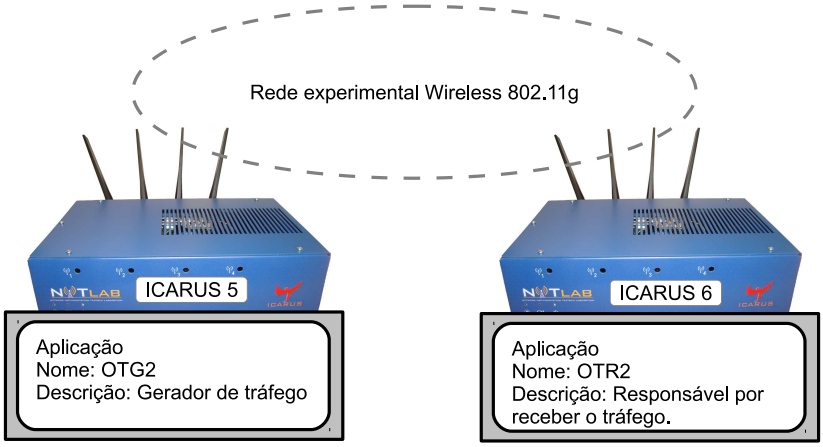

Figura 3.1: *Cenário do experimento com nós sem fio.*

Este experimento, como ilustrado na Figura 3.1, cria uma rede *adhoc* entre os dois nós ICARUS, através de suas interfaces sem fio. Após isso, ele inicia o ODG2 no ICARUS 5 e o ODR2 no ICARUS 6,

fazendo eles se comunicarem através da rede criada. Os resultados do tráfego são coletados e enviados ao servidor OML.

3.1.2 Experimento com tecnologia OpenFlow

O objetivo deste experimento, como descrito na proposta de extensão apresentada no Apêndice C, é investigar a viabilidade do OMF 6 para controlar um teste completo com as tecnologias de uma *testbed* OpenFlow. Para tanto, o experimento reuniu diferentes experimentos, realizados de maneira independente na primeira fase dos microprojetos, em um único teste fim-a-fim. Um *script* executado pelo EC no Computador 1, realiza os seguintes passos:

No Computador 3:

1. Cria uma *bridge* no Open vSwitch, adiciona as portas eth1 e eth2 e configura o FlowVisor como contralodor da *bridge*;
2. Inicia o FlowVisor, cria um *slice* e configura um flowspace;

No Computador 2:

3. Inicia as máquinas virtuais *vm-openflow-test-1* e *vm-openflow-test-2*, conecta suas interfaces às portas eth1 e eth2, respectivamente, do *switch* OpenFlow;
4. Inicia a máquina virtual *vm-openflow-pox* e inicia o controlador OpenFlow POX para comunicar com o Open vSwitch através do FlowVisor;
5. Inicia o servidor *iperf* na *vm-openflow-test-2* e o cliente *iperf* na *vm-openflow-test-1*, fazendo-as se comunicar através do *switch* OpenFlow.

Para que fosse possível executar este experimento, precisamos realizar algumas modificações nos RCs que controlam o FlowVisor e o Open vSwitch. Como apresentado no relatório anterior, esses RCs estão desatualizados e, por isso, não funcionam com a versão mais nova do OMF 6. Além das modificações realizadas na primeira fase dos microprojetos, fizemos outras correções para que os ambos os RCs se comunicassem via AMQP. Incluímos também no RC do Open vSwitch a possibilidade de configuração do controlador da *bridge*, utilizada para configurar o FlowVisor como controlador no passo 1.

3.2 Investigação do Broker

Nesta seção, apresentamos conceitos da arquitetura do Broker e os experimentos realizados para verificar suas funcionalidades.

3.2.1 Arquitetura do Broker

O Broker é um arcabouço de gerenciamento de *testbeds* implementado pela equipe do NITlab [7] para gerenciar diferentes tipos de *testbed*. A arquitetura desse componente foi pensada para torná-lo

extensível e desacoplado de especificidades das *testbeds*. A Figura 3.2 mostra um diagrama da arquitetura, ilustrando a divisão em camadas e a comunicação entre elas. A explicação de cada uma das camadas, baseadas no artigo de Stavropoulos et. al. [7], será apresentada a seguir.

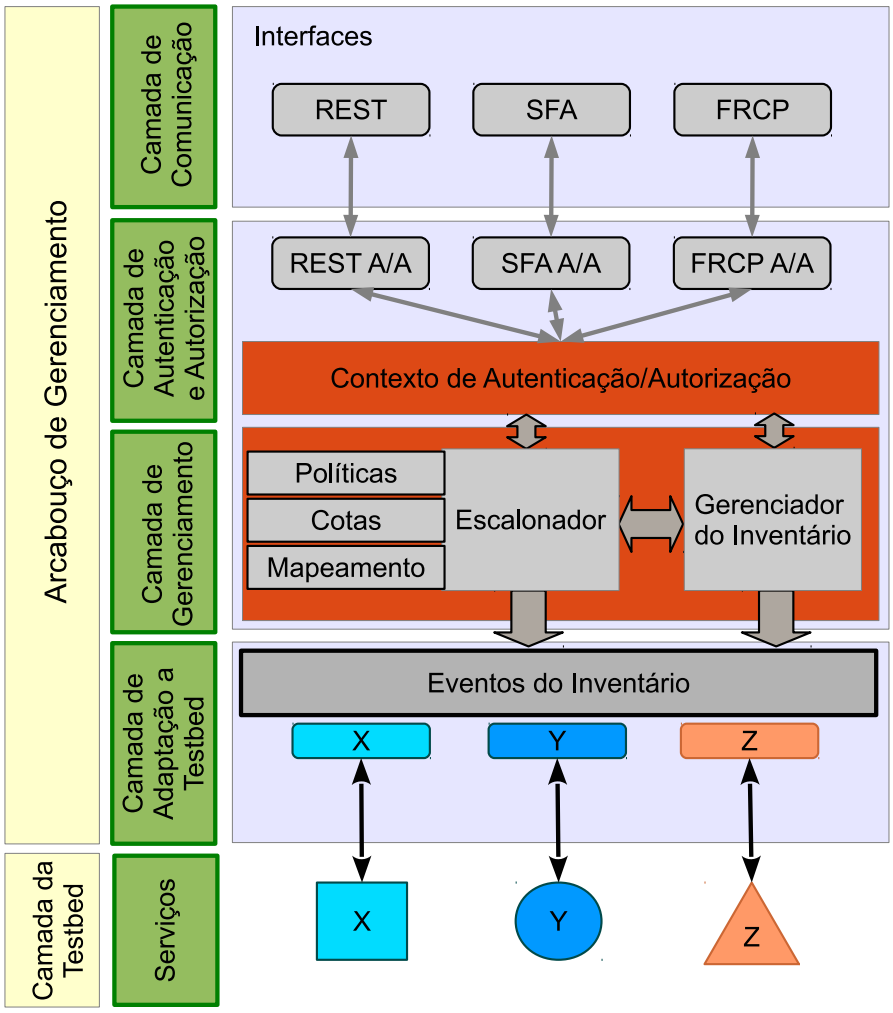

Figura 3.2: *Arquitetura do Broker.*

Camada de Comunicação

A Camada de Comunicação possui interfaces de comunicação para os protocolos SFA, REST e FRCP. A interface SFA suporta as versões v2 e v3[1] do protocolo SFA, o qual é extensamente utilizado para permitir a federação de *testbeds*. A interface REST é uma alternativa a SFA e que permite uma maior flexibilidade e a implementação de funcionalidades adicionais. A API REST também simplifica a comunicação com as tecnologias *Web* e a implementação de portais. A interface FRCP não tem o objetivo de realizar as responsabilidades da API SFA, como obter uma lista de recursos ou modificar um inventário. Ela é primariamente utilizada para interoperabilidade com os RCs, a fim de automatizar procedimentos relacionados a funcionalidades específicas de uma *testbed*.

[1] A versão v3 está em fase de desenvolvimento e será finalizada, possivelmente, este mês.

Camada de Autenticação/Autorização

Esta camada é responsável por aceitar ou negar uma requisição baseada em um contexto de autenticação e autorização. Cada uma das interfaces de comunicação recebe um conjunto de credenciais diferentes, o que requer a utilização de diferentes mecanismos para lidar com elas.

Inicialmente, o sistema confirma a identidade do usuário pelo processo de autenticação. O mecanismo de autenticação pode ser configurado para confiar e autenticar usuários de um conjunto específico de autoridades certificadoras. Após a autenticação, a autorização do usuário é verificada através de um conjunto de atributos que denotam as permissões dos usuários em relação a: (i) Recursos, (ii) Contas, (iii) Reservas. Essas permissões informam se o usuário pode: (i) Criar; (ii) Visualizar; (iii) Modificar; ou (iv) Liberar um recurso, conta ou reserva.

Em requisições SFA, existe um arquivo XML assinado que contém os privilégios do usuário, os quais são mapeados para o conjunto de atributos de permissões pelo módulo de Autenticação/Autorização (A/A). Em requisições REST e FRCP, os atributos de A/A são preenchidos baseado na identidade obtida do certificado e do papel do usuário.

Camada de Gerenciamento

A Camada de Gerenciamento é composta pelo Inventário e pelo Escalonador. O inventário é um banco de dados que armazena as informações sobre os recursos, contas e reservas da *testbed*. O Escalonador é o módulo responsável por tomar decisões a respeito da reserva/alocação de recursos, baseado na disponibilidade e em um conjunto de políticas. É nesse módulo que devem ser implementadas políticas de alocação e de cotas de acesso aos recursos. A implementação padrão utiliza a política *First-Come-First-Served* (FCFS) para atender as requisições de alocação, ou seja, ela aloca o recurso a quem solicitar primeiro. Contudo, devido a separação da arquitetura em módulos, é possível implementar políticas mais complexas apenas adaptando o Escalonador.

No Escalonador também pode ser implementada a funcionalidade de mapeamento de recursos abstratos para recursos físicos. Essa funcionalidade permite que os usuários solicitem recursos sem especificar quais, deixando que o escalonador decida quais recursos melhor completam a solicitação. Essa funcionalidade foi desenvolvida em um módulo separado do escalonador, permitindo que os operadores da *testbed* adicionem seus próprios algoritmos de mapeamento de recursos abstratos para recursos físicos.

Camada de Adaptação à *Testbed*

A Camada de Adaptação é responsável por integrar o arcabouço com os recursos e serviços da *testbed*. Seguindo os princípios do modelo arquitetural *Service Oriented Architecture* (SOA), quando uma tarefa específica é executada na *testbed*, a Camada de Adaptação é responsável por interagir com o serviço correspondente da *testbed*. Um exemplo dessa interação é mostrado no experimento apresentado na Seção 3.2.2.

A separação do Broker das peculiaridades da *testbed* oferece um melhor controle e manutenção da *testbed*, uma vez que as funcionalidades estão distribuídas em serviços e não acopladas ao AM. A interação

com os recursos da *testbed* a partir da Camada de Adaptação pode ser feita utilizando tanto REST, quanto FRCP. Teoricamente, também é possível criar uma hierarquia de Brokers e outros AMs, comunicando-se via FRCP ou mesmo via SFA.

3.2.2 Experimentos realizados com o Broker

Para realizarmos os testes com a interface SFA, utilizamos o cliente SFA *omni*, configurado para solicitar as credenciais da *clearinghouse gcf-ch*. Para autorizar o acesso do cliente ao Broker, é necessário gerar o certificado para o *gcf-ch* e inseri-lo no diretório .omf/trusted_roots da *vm-broker*. É necessário também criar um certificado assinado com a chave do certificado da *clearinghouse* para o cliente. A seguir apresentamos os experimentos realizados para testar a interface SFA do Broker.

Requisitar a versão do Broker

Para solicitar a versão do Broker, utilizamos o comando `omni -a https://ip-do-broker:8001/RPC2 getversion`, o qual retorna as informações cadastradas no arquivo omf_sfa/etc/omf-sfa/getVersion_ext.yaml do Broker [2]. As informações obrigatórias são o *hrn*, o *urn* e o *hostname*.

Listar os recursos do inventário

A lista de recursos pode ser solicitada via SFA ou via REST. A Lista 3.1 mostra a resposta ao comando `omni -a https://ip-do-broker:8001/RPC2 listresources`, o qual mostra os recursos cadastrados no inventário do Broker. No caso, os nós cadastrados são o ICARUS 5 e o ICARUS 6.

```
...
  <node component_id="urn:publicid:IDN+ufg.br+node+icarus5" component_manager_id="
      urn:publicid:IDN+ufg.br+authority+cm" component_name="icarus5" exclusive="true">
    <available now="true"/>
    <interface component_id="urn:publicid:IDN+ufg.br+interface+icarus5:if0"
        component_name="icarus5:if0" role="control">
      <ip address="10.0.0.5" type="ipv4" netmask="255.255.0.0"/>
    </interface>
    <interface component_id="urn:publicid:IDN+ufg.br+interface+icarus5:if1"
        component_name="icarus5:if1" role="experimental"/>
    <ol:lease_ref id_ref="ee58a9e7-ed65-4844-84df-44a0f78d1685"/>
  </node>
  <node component_id="urn:publicid:IDN+ufg.br+node+icarus6" component_manager_id="
      urn:publicid:IDN+ufg.br+authority+cm" component_name="icarus6" exclusive="true">
    <available now="true"/>
    <interface component_id="urn:publicid:IDN+ufg.br+interface+icarus6:if0"
        component_name="icarus6:if0" role="control">
      <ip address="10.0.0.6" type="ipv4" netmask="255.255.0.0"/>
    </interface>
    <interface component_id="urn:publicid:IDN+ufg.br+interface+icarus6:if1"
        component_name="icarus6:if1" role="experimental"/>
```

```
    <ol:lease_ref id_ref="ee58a9e7-ed65-4844-84df-44a0f78d1685"/>
  </node>
</rspec>
```

Lista 3.1: *Parte da resposta à solicitação de listagem de recursos via SFA.*

Solicitar informações de um recurso

Neste experimento, utilizamos a interface FRCP do Broker para solicitar informações a respeito de um recurso. A Figura 3.3 ilustra a execução do experimento. O *script omf6* foi executado para ligar o nó ICARUS 5, através do comando `omf6 tell -a on -t icarus5`. Esse comando, inicialmente, recupera as informações relativas ao ICARUS 5 cadastradas no inventário do Broker e publicadas no servidor XMPP. Utilizando a informação do endereço IP do Chassi Manager do ICARUS 5, o *omf6* envia uma requisição HTTP para ligar o ICARUS 5.

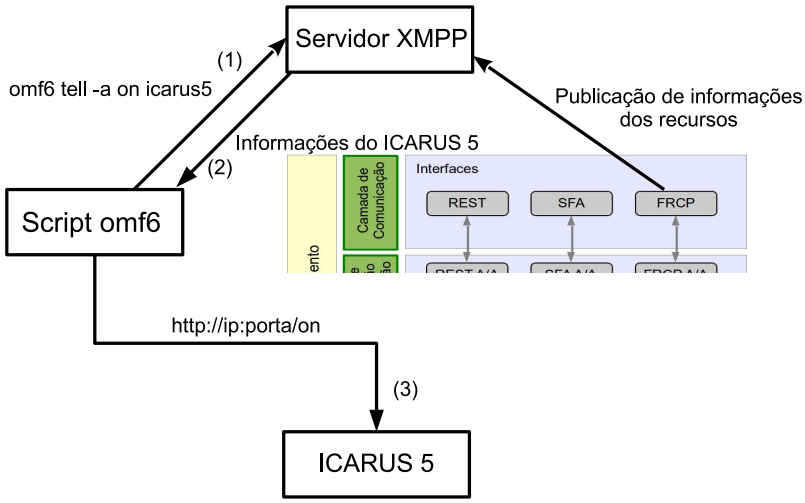

Figura 3.3: *Experimento com o script omf6.*

Criar um *sliver* no Broker

Para criar um *sliver* no Broker, podemos utilizar o comando `omni -a https://ip-do-broker:8001/RPC2 -q createsliver teste ~/omf-sfa/request.xml`. Esse comando cria um *slice* na *gcf-ch* e um *sliver* no Broker com as informações passadas no arquivo ~/omf-sfa/request.xml (em formato *rspec*), o qual contém uma especificação dos recursos a serem alocados. Na prática, um *sliver* no Broker equivale a uma conta de usuário com uma concessão (*lease*) de acesso a recursos. Como exemplo, observe o *rspec* apresentado na Lista 3.2, o qual concede acesso ao ICARUS 5 das 14:00 às 18:00 do dia 22/03/2016.

```
<rspec ...>
  <ol:lease client_id="lease_1234" valid_from="2016-3-22T14:00:00+03:00" valid_until=
     "2016-3-22T18:00:00+03:00"/>
  <node component_id="urn:publicid:IDN+ufg.br+node+icarus5" client_id="the_icarus5">
    <ol:lease_ref id_ref="lease_1234"/>
  </node>
```

```
</rspec>
```

Lista 3.2: *Parte do rspec com concessão ao ICARUS 5.*

A Figura 3.4 ilustra o processo de criação de um *sliver*: (1) o usuário executa o comando no cliente SFA; (2) o cliente SFA solicita a *clearinghouse* as credenciais do cliente e informações a respeito do Broker; (3) o cliente SFA envia uma solicitação de criação de *sliver* ao Broker; (4) o Broker cria uma conta e adiciona as concessões aos recursos no inventário; (5) o Broker interage com o User RC para criar uma conta Linux na *vm-broker*. O User RC insere a chave do cliente no arquivo ~/.ssh/authorized_keys e adiciona as configurações do *script omf6* ao diretório ~/.omf. O usuário pode, então, acessar a conta Linux via *ssh* e utilizar o comando *omf6* para operar o ICARUS 5 durante o período de concessão.

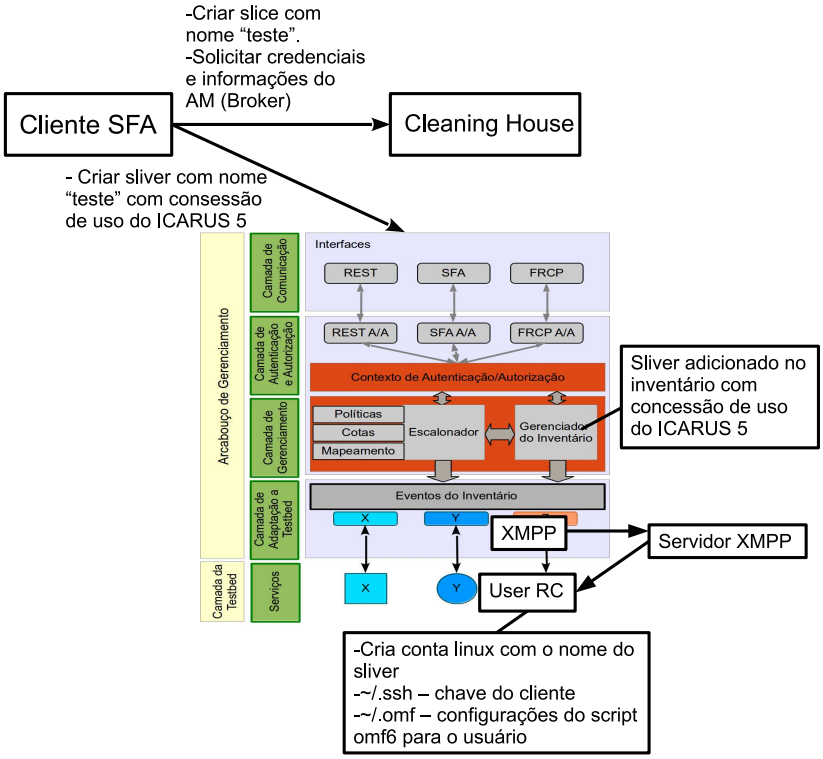

Figura 3.4: *Criação de um sliver no Broker, com concessão de acesso ao ICARUS 5.*

3.3 Considerações gerais

Os experimentos realizados no OMF 6 foram bem-sucedidos, mostrando o potencial do arcabouço para controlar diferentes tipos de recursos, incluindo recursos para a parte OpenFlow, mesmo em experimentos complexos que interagem com diversos RCs. A investigação do Broker revelou que sua arquitetura é madura, o que simplifica futuras extensões e adaptações. Por meio dos testes, validamos a funcionalidade de suas interfaces de comunicação, do escalonador e do inventário. Apesar de sua maturidade, ainda é necessário adaptar o Broker para utilizar o protocolo AMQP, a fim de simplificar a arquitetura da *testbed* que teria apenas um servidor *Publish/Subscribe*.

É preciso também alterar o EC para que ele solicite informações de reserva de recursos antes de iniciar os experimentos. Atualmente, o *script omf6* verifica as reservas antes de enviar os comandos, mas o EC não faz essa verificação. Dessa forma, é possível que um usuário execute um experimento em um nó que está alocado para outro usuário, interferindo assim nos resultados do usuário que possui a concessão.

CAPÍTULO 4

Conclusão e Avaliação

Durante a segunda etapa dos microprojetos realizamos experimentos mais complexos com o OMF 6 e validamos o poder do arcabouço para executar experimentos que utilizam diferentes recursos simultaneamente. Embora alguns RCs ainda apresentem limitações, sua correção e evolução e até mesmo a criação de novos RCs é uma tarefa relativamente simples devido à arquitetura madura do OMF 6.

Realizamos também, na segunda etapa, uma investigação das funcionalidades do Broker. Verificamos que esse componente possui uma arquitetura madura e planejada para ser extensível. O Broker possui interfaces de comunicação SFA, FRCP e REST, o que simplifica a federação e a integração com outras *testbeds*, além de oferecer vantagens para a criação de um portal. O Broker atualmente funciona com a versão v2 do SFA. Contudo, o desenvolvimento da comunicação via SFA v3 está em fase de conclusão e será liberado possivelmente este mês.

Os experimentos realizados no Broker validaram o funcionamento das três interfaces de comunicação, da camada de A/A, do escalonador, do inventário e da camada de adaptação à *testbed*. O arcabouço se mostrou estável em todas as avaliações, contudo, ainda é necessário adaptá-lo para utilizar o protocolo AMQP. Essa adaptação é importante para simplificar a arquitetura geral da *testbed* e ao mesmo tempo utilizando o protocolo indicado para o OMF 6.

A adoção do Broker como arcabouço de gerenciamento de recursos na *tesbed* do FIBRE é importante, uma vez que, o Broker foi desenvolvido e validado no contexto do OMF 6. O esforço estimado para a adaptação do Broker é de dois meses e sua implantação é relativamente simples em uma única ilha. Sua implantação em todo o FIBRE de maneira federada também deverá ser realizada. Todavia, ainda são necessárias mais discussões e informações para avaliar esse esforço. Essa e outras atividades estão sendo definidas e estimadas pelo Comitê Técnico do FIBRE.

Referências Bibliográficas

[1] CHOUMAS, K. **Omfrcopenflow - github**. https://github.com/kohoumas/omf_rc_openflow, 2015. [Último acesso: 29-Dezembro-2015].

[2] DOSTRAVO. **getversion_ext.yaml**. https://github.com/dostavro/omf_sfa/blob/master/etc/omf-sfa/getVersion_ext.yaml, 2016. [Último acesso: 01-Abril-2016].

[3] NICTA. **Omf: Unlock your experiments**. http://mytestbed.net/, 2015. [Último acesso: 29-Dezembro-2015].

[4] NICTA. **Performance comparison between using xmpp or amqp as the pubsub substrate for omf**. https://omf.mytestbed.net/projects/omf6/wiki/XMPPvsAMQP, 2015. [Último acesso: 29-Dezembro-2015].

[5] NICTA. **Omf: Unlock your experiments**. http://mytestbed.net/doc/omf/file.TUTORIAL_01.html, 2016. [Último acesso: 30-Março-2016].

[6] NITLAB. **Nitlab - network implementation testbed laboratory**. http://nitlab.inf.uth.gr/NITlab/, 2015. [Último acesso: 29-Dezembro-2015].

[7] STAVROPOULOS, D.; DADOUKIS, A.; RAKOTOARIVELO, T.; OTT, M.; KORAKIS, T.; TASSIULAS, L. **Design, architecture and implementation of a resource discovery, reservation and provisioning framework for testbeds**. IEEE, 2015.

Controle de Recursos Básicos de uma Testbed Usando OMF 6

Nome do orientador: Kleber Vieira Cardoso
Nome do bolsista: Vinícius Gonçalves Braga

Início do projeto: 15/10/2015
Término do projeto: 15/01/2016

Assunto/Tema: Atualização do arcabouço de controle e gerenciamento de *testbed*.

Objetivo:

Até a versão 5.4, utilizada no FIBRE, o OMF era composto por um conjunto de componentes, conforme ilustrado na Figura A.1, os quais implementavam as tarefas de controle e gerenciamento de uma infraestrutura de experimentação (*testbed*). Os principais componentes do OMF 5.4 são:

- *Experiment Controller* (EC) – é o componente com o qual o experimentador da *testbed* interage diretamente. O EC recebe uma descrição do experimento (escrita em OEDL – OMF Experiment Description Language) e interage com o AM para a configuração e execução do experimento. Também é através do EC que o experimentador recupera os resultados.
- *Aggregate Manager* (AM) – é o gerenciador central dos recursos da *testbed*. Ele é responsável pela alocação de nós para o experimento, início da execução, coleta dos resultados, armazenamento das imagens de disco, etc.
- *Resource Manager* (RM) – executa em cada um dos recursos da *testbed* e é responsável por receber uma imagem de disco e escrevê-la no recurso, ou gerar uma imagem do estado atual do recurso.
- *Resource Controller* (RC): assim como o RM, é executado no recurso e tem a finalidade de responder aos comandos fornecidos na descrição do experimento.

O OMF 6 introduziu o conceito de que tudo é um recurso e passou definir explicitamente apenas dois componentes: Resource Controller (RC) e Experiment Controller (EC). Enquanto o EC continua tendo um papel similar ao da versão anterior, o RC passou a ser uma entidade mais sofisticada. Um RC controla um ou múltiplos recursos, podendo ser executado dentro do recurso (por exemplo, em um PC) ou em um equipamento separado (por exemplo, para controlar um conjunto de nós sensores ou um *switch* OpenFlow). Assim, todas as funcionalidades de um AM (por exemplo, servidor de imagens, serviço de nomes, inventário de nós, etc.) podem ser controlados como recursos. Além disso, um recurso pode ser uma coleção de outros recursos. Por exemplo, uma *testbed* é um recurso composto por recursos como nós de experimentação, servidor de imagens, armazenamento de dados coletados, etc.

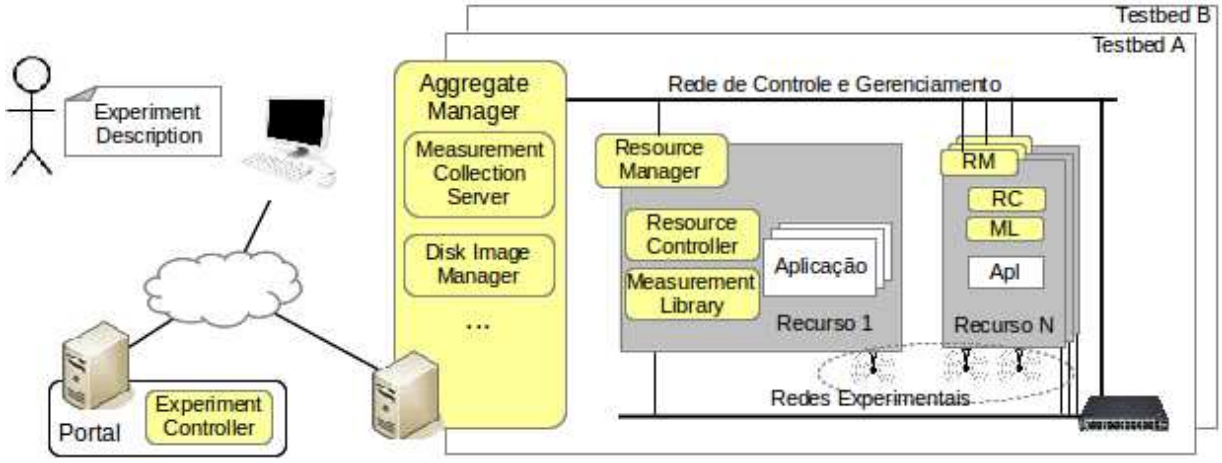

Figura A.1: *Componentes que integram a arquitetura do OMF 5.4.*

Atualmente, já há vários RCs (no OMF 6) para diferentes tipos de recursos. No entanto, há carência de documentação sobre a implantação e a integração de RCs suficientes para formar uma infra-estrutura de experimentação completa. Nesse contexto, o principal objetivo dessa proposta é investigar a disponibilidade de RCs suficientes para atender os requisitos básicos de uma *testbed* do FIBRE, assim como avaliar a viabilidade de adoção do OMF 6 como substituto do OMF 5.4.

Descrição do protótipo:

O protótipo consiste na implantação de uma *testbed* para prova de conceito composta por 2 nós sem fio como recursos para experimentação e demais recursos básicos de uma *testbed*: servidor de imagens, servidor de nomes, armazenamento de dados coletados, etc.

Resultados entregues:

1. *Testbed* para prova de conceito em estado operacional.
2. Relatório sobre a *testbed*, descrevendo os RCs implantados e sua integração.
3. Relatório sobre a implantação, descrevendo: principais problemas encontrados, eventuais RCs a serem implementados ou customizados e eventuais riscos de implantação em ambiente de produção.

Cronograma:

15/10 – 15/12: implantação e integração dos RCs para formar uma *testbed*.
15/11 – 15/12: interação com equipes de desenvolvimento do OMF 6 (em especial, UTH e NICTA) para identificar a disponibilidade e o estado mais recente dos RCs básicos para implantação de uma *testbed*.
15/12 – 15/01: testes de avaliação da *testbed* e elaboração dos relatórios.

Referências:

[1] http://mytetbed.net.
[2] Thierry Rakotoarivelo, Max Ott, Guillaume Jourjon, Ivan Seskar, “OMF: a control and management framework for networking testbeds”, in ACM SIGOPS Operating Systems Review 43 (4), 54-59, Jan. 2010.

Controle de recursos OpenFlow usando OMF 6

Nome do orientador: Kleber Vieira Cardoso
Nome do bolsista: Vinícius Gonçalves Braga

Início do projeto: 15/10/2015
Término do projeto: 15/01/2016

Assunto/Tema: Substituição do arcabouço de controle e gerenciamento de recursos OpenFlow.

Objetivo:

O controle e gerenciamento de uma *testbed* OpenFlow envolvem não apenas os *switches* com suporte a essa tecnologia, mas também outros recursos como máquinas virtuais, além da necessidade de manipular o conceito de *slices* que permite que múltiplos experimentadores utilizem concorrentemente os recursos OpenFlow.

O OMF 6 já oferece *Resource Controllers* (RCs) para gerenciar recursos como OpenvSwitch, FlowVisor e KVM. Além disso, *testbeds* como o da UTH (*University of Thessaly*) já disponibilizam acesso para realização de experimentos com recursos OpenFlow controlados e gerenciados pelo OMF. No entanto, há carência de documentação sobre a implantação e a integração desses RCs. Além disso, o repositório oficial do OMF 6 ainda não oferece RC para a plataforma de virtualização XEN, utilizada no FIBRE. Nesse contexto, o principal objetivo dessa proposta é investigar a disponibilidade de RCs suficientes para controlar e gerenciar recursos OpenFlow. É importante identificar o conjunto de funcionalidades disponíveis nos RCs existentes com intuito de avaliar a viabilidade de substituição do OCF pelo OMF 6. Adicionalmente, é necessário verificar se há iniciativas de desenvolvimento de RC para XEN ou se haveria necessidade de iniciar esse desenvolvimento.

Descrição do protótipo:

O protótipo consiste na implantação de uma *testbed* para prova de conceito composta por um *switch* OpenFlow em software (executando OpenvSwitch), o qual será virtualizado através do FlowVisor e conectado a um servidor com um sistema de virtualização, provavelmente baseado em KVM. O sistema de virtualização deve ser capaz instanciar máquinas virtuais tanto para geração de tráfego quanto para controlar o *switch* OpenFlow.

Resultados entregues:

1. *Testbed* para prova de conceito em estado operacional.
2. Relatório sobre a *testbed*, descrevendo os RCs implantados e sua integração.

3. Relatório sobre a implantação, descrevendo: principais problemas encontrados e eventuais RCs a serem implementados ou customizados.

Cronograma:

15/10 – 15/12: implantação e integração dos RCs para formar uma *testbed*.

15/11 – 15/12: interação com equipes de desenvolvimento do OMF 6 (em especial, UTH e NICTA) para identificar a disponibilidade e o estado mais recente dos RCs para recursos OpenFlow.

15/12 – 15/01: testes de avaliação da *testbed* e elaboração dos relatórios.

Referências:

[1] http://mytetbed.net.

[2] https://github.com/kohoumas/omf_rc_openflow.

[3] http://nitlab.inf.uth.gr/NITlab/index.php/news/313-openflow-extensions-for-omf-6.

[4] Thierry Rakotoarivelo, Max Ott, Guillaume Jourjon, Ivan Seskar, “OMF: a control and management framework for networking testbeds”, in ACM SIGOPS Operating Systems Review 43 (4), 54-59, Jan. 2010.

Avaliação das funcionalidades do Broker e realização de testes fim-a-fim em recursos OpenFlow e sem fio com o OMF 6

Nome do orientador: Kleber Vieira Cardoso
Nome do bolsista: Vinícius Gonçalves Braga

Início do projeto: 01/02/2016
Término do projeto: 31/03/2016

Assunto/Tema: Avaliação das funcionalidades do Broker e realização de testes fim-a-fim em recursos OpenFlow e sem fio com o OMF 6.

Objetivo:

Durante a execução dos microprojetos "Controle de recursos OpenFlow usando OMF 6" e "Controle de recursos básicos de uma *testbed* usando OMF 6" foi realizada a implantação de uma *testbed* funcional do OMF 6 e a avaliação do uso dos Resource Controllers (RCs) básicos e dos RCs da parte OpenFlow. A avaliação dos RCs foi feita isoladamente, comprovando suas funcionalidades e a a capacidade do OMF 6 de controlar diferentes tipos de recursos. Contudo, é importante também que testes fim-a-fim sejam realizados para verificar a capacidade do OMF 6 em experimentos mais avançados, controlando recursos OpenFlow e sem fio.

Como exemplo de um teste fim-a-fim para recursos OpenFlow, poderíamos definir um experimento em OEDL para:

1. Configurar uma bridge no Open vSwitch com duas portas (eth1 e eth2, por exemplo) configuradas;
2. Criar um *slice* no FlowVisor;
3. Criar duas máquinas virtuais, conectar suas interfaces às portas do *switch* OpenFlow (uma na porta eth1 e outra na porta eth2, por exemplo) e configurá-las na mesma rede;
4. Iniciar um controlador OpenFlow (o POX, por exemplo) para comunicar com o Open vSwitch através do FlowVisor;
5. Iniciar o servidor iperf em uma máquina virtual e o cliente iperf em outra, fazendo-as se comunicar através do *switch* OpenFlow.

E um exemplo de teste fim-a-fim para recursos sem fio, seria:

1. Configurar uma rede *adhoc* em dois nós ICARUS;

2. Iniciar o servidor iperf em uma nó e o cliente em outro, fazendo-os se comunicar através da rede sem fio.

Outro fato a se notar é que, com o contato mais próximo com o grupo NITlab [2], tivemos acesso ao Broker [1], um componente que age como um Aggregate Manager e também provê uma API SFA. Além disso, o Broker possui uma API REST, a qual pode ser utilizada para integração com uma interface Web. Durante o microprojeto, o Broker foi utilizado para cadastrar os nós ICARUS como recursos, permitindo o acesso a esses nós e o teste das funcionalidades de carregamento de imagem e controle do Chassi Manager. Contudo, não foi possível investigar outras funcionalidades do Broker, como descoberta, reserva e provisionamento de recursos.

Dessa forma, os objetivos dessa proposta são: (1) realizar testes fim-a-fim com o OMF 6, controlando tanto recursos OpenFlow, quanto recursos sem fio e coletando os resultados utilizando os recursos da OML; (2) investigar a parte de descoberta, reserva e provisionamento de recursos do Broker.

Descrição do protótipo:

O protótipo utilizado nesta proposta será a infraestrutura da *testbed* construída previamente durante a execução dos microprojetos.

Resultados entregues:

1. Relatório sobre os testes avançados, com recursos OpenFlow e sem fio, executados na *testbed*.
2. Relatório sobre o Broker, descrevendo sua funcionalidade de descoberta, reserva e provisionamento de recursos.

Cronograma:

01/02 – 10/02: execução dos testes avançados com recursos OpenFlow e sem fio.
11/02 – 31/03: investigação das funcionalidades do Broker, interagindo com a equipe do NITlab para eventuais dúvidas.
15/03 – 31/03: elaboração dos relatórios.

Referências:

[1] https://github.com/dostavro/omf_sfa/blob/master/Installation.md
[2] http://nitlab.inf.uth.gr