UNIVERSIDADE FEDERAL DE GOIÁS
INSTITUTO DE INFORMÁTICA

VINÍCIUS GONÇALVES BRAGA

Controle de Recursos Básicos de uma Testbed e Controle de Recursos OpenFlow Usando OMF 6

Goiânia
2016

Sumário

Lista de Figuras

Lista de Tabelas

CAPÍTULO 1

Introdução

O OMF é um arcabouço de controle, medição e gerenciamento de plataformas experimentais (*testbeds*) [3]. Ele busca oferecer as ferramentas e o ambiente necessário para a criação de infraestruturas de pesquisa em redes de computadores, visando principalmente o avanço de trabalhos sobre a Internet do Futuro. Inicialmente, criado pelo *Wireless Information Network Laboratory* (WinLab) para a *Open-Access Research Testbed for Next-Generation Wireless Networks* (ORBIT), suportava apenas equipamentos específicos dessa *testbed*. Nos últimos anos, os esforços de desenvolvimento resultaram em um arcabouço mais maduro e que suporta diferentes tipos de dispositivos.

Atualmente, o FIBRE utiliza a versão 5.4 do OMF para controlar experimentos e coletar resultados de diferentes tipos de dispositivos e o *OFELIA Control Framework* (OCF) para experimentação em projetos relacionados a OpenFlow [2]. O OMF 5.4 possui quatro componentes principais:

- *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.

Os papéis desses componentes foram revistos na nova versão. O OMF 6 simplifica a arquitetura do arcabouço e mantém apenas dois desses componentes: o Experiment Controller (EC) e o Resource Controller (RC). O EC continua sendo o responsável por receber e executar uma descrição do experimento em OEDL, mas agora se comunica diretamente com os RCs. Os RCs se tornaram uma entidade mais sofisticada, podendo controlar um ou múltiplos recursos. Um RC pode ser instalado dentro do recurso (por exemplo, em um PC) ou em um equipamento separado, controlando o recurso através da rede (via HTTP, por exemplo). No OMF 6, a API permite que qualquer tipo de hardware ou *software* seja um recurso. Desde que exista uma interface de comunicação[1] é possível implementar um RC para controlar esse recurso.

Com o novo papel do RC, todas as funcionalidades de um AM (por exemplo, servidor de imagens, serviço de nomes, inventário de nós, etc.) podem ser controlados como recursos. Dessa forma, foi criado o

[1] A interface de comunicação pode ser oferecida como comandos locais ou remotos, por meio de uma API REST, por exemplo

Broker, o qual herdou as responsabilidades do AM e possui RCs para controlar contas de usuários Linux, o carremento de imagens e os Chassis Manager dos nós. Em geral, o Broker é responsável pela descoberta, reserva e provisionamento de recursos.

Atualmente, o OMF 6 possui um conjunto de RCs para controle de diferentes tipos de recursos. Além de RCs para controlar aplicações e recursos de rede (cabeada e sem fio), ele oferece RCs para controle do Open vSwitch, do FlowVisor e do KVM. A existência desses RCs sugere que o OMF 6 pode substituir o OMF 5.4 e, ainda, pode substituir o OCF na infraestrutura do FIBRE. Apesar disso, há carência de documentação sobre esses RCs e uma avaliação mais profunda deve ser realizada para possibilitar uma tomada de decisão. Nesse contexto, os principais objetivos das propostas relacionadas a este relatório são:

- 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; e
- identificar o conjunto de funcionalidades disponíveis nos RCs relacionados a parte OpenFlow com intuito de avaliar a viabilidade de substituição do OCF pelo OMF 6.

Este relatório descreve a *testbed* criada e outras atividades realizadas a fim de cumprir as propostas de projeto *Controle de Recursos Básicos de uma Testbed Usando OMF 6*, descrita no Apêndice A, e *Controle de recursos OpenFlow usando OMF 6*, apresentada no Apêndice B. No Capítulo 2, descrevemos a configuração do ambiente da *testbed*, mostrando todos os programas instalados para possibilitar a avaliação do OMF 6. No Capítulo 3, apresentamos os experimentos realizados na *testbed*, destacando as funcionalidades existentes, as limitações e os problemas encontrados no OMF 6. Por fim, no Capítulo 4, descrevemos nossa avaliação geral sobre o OMF 6 e sobre sua adoção na infraestrutura do FIBRE e estimamos o esforço necessário para a implantação desse arcabouço em uma ilha.

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.

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.

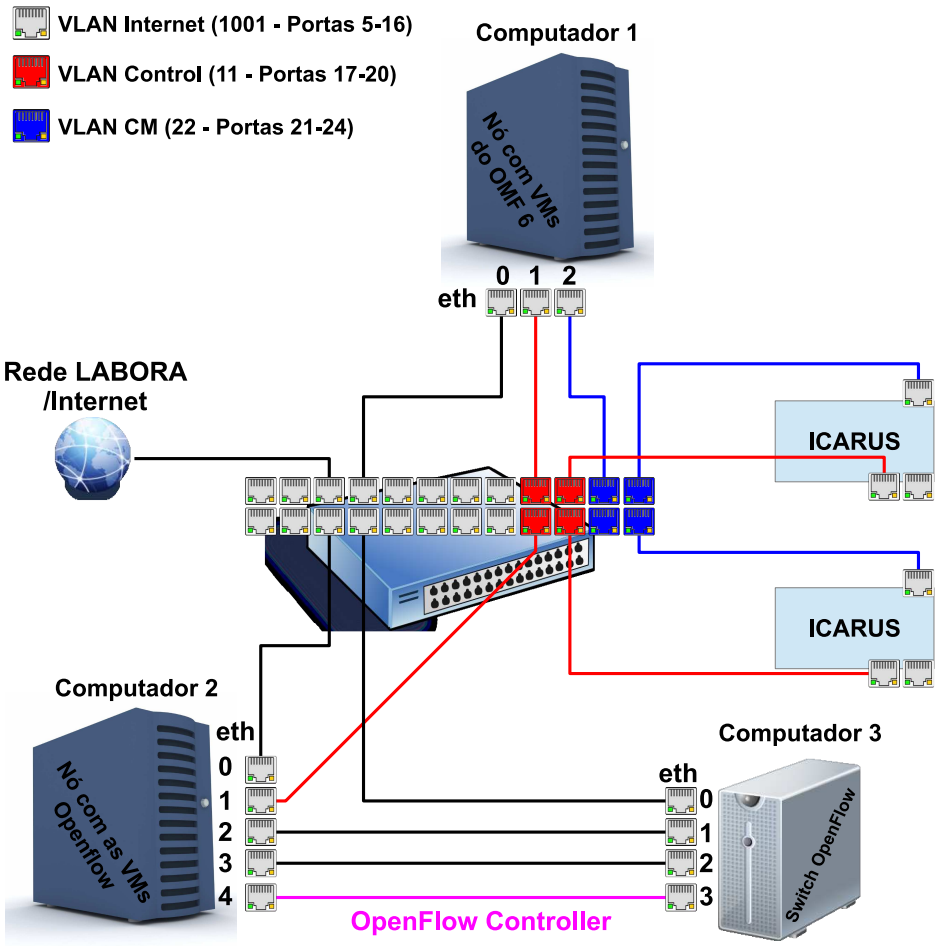

Figura 2.1: *Modelo de implantação da testbed.*

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.

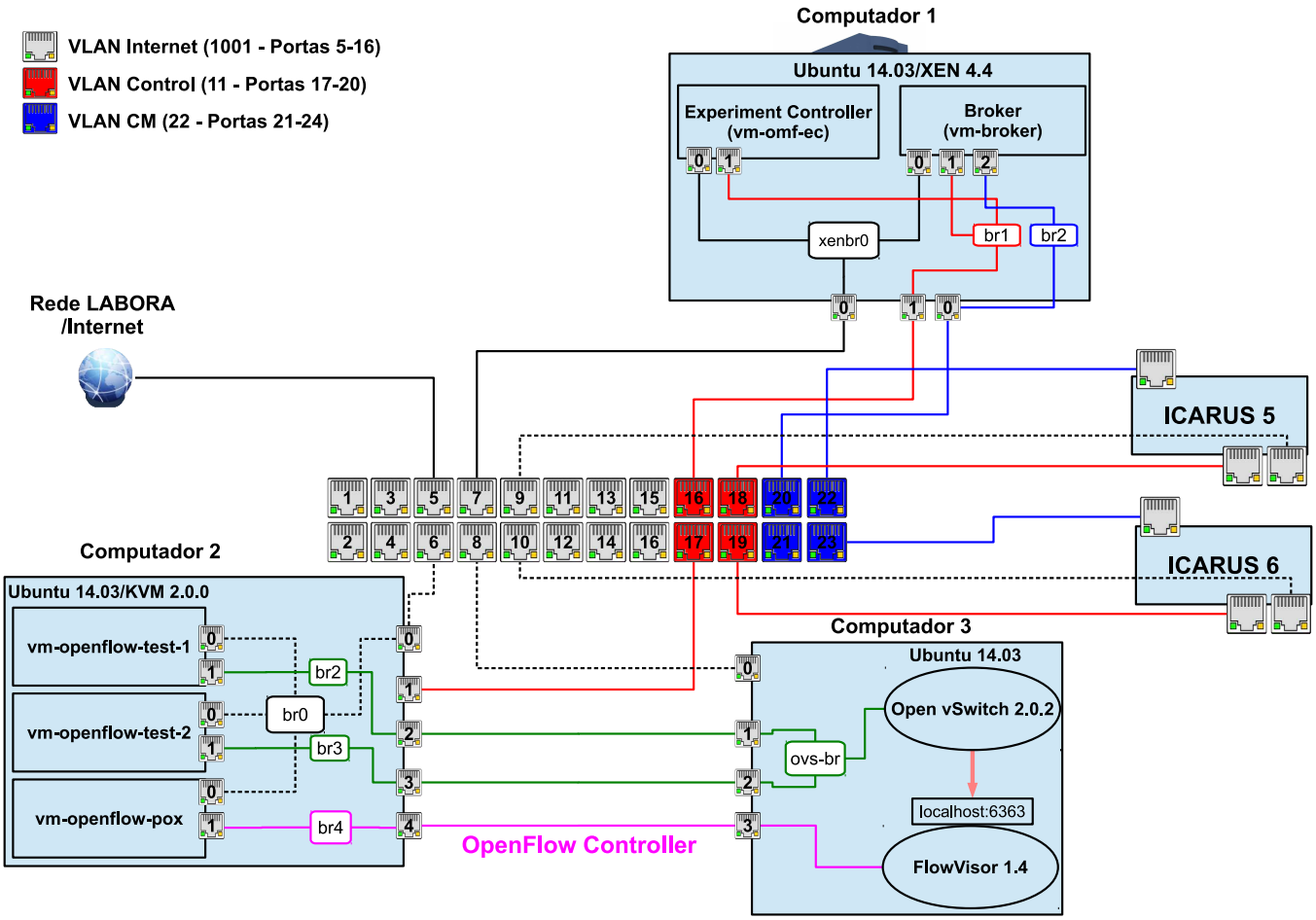

Figura 2.2: *Modelo de implantação da testbed.*

No C1, foram criadas duas máquinas virtuais: (1) *vm-omf-ec*, na qual instalamos o EC, 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. 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
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, como o Broker, por exemplo.

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 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

Boa parte do conhecimento conquistado nos microprojetos foi em relação a instalação e configuração dos diferentes módulos que formam uma *testbed*. Alguns desses módulos, como o Broker, são formados por vários programas e a configuração de todos é bastante complexa, para quem não tem uma experiência prévia. Por esse motivo, a comunicação com os grupos desenvolvedores, como o NITlab, é essencial nesse processo. Além do conhecimento da parte de implantação da *testbed*, adquirimos conhecimento em relação aos RCs existentes através de uma série de experimentos, que serão descritos a seguir.

3.1 Experimentos realizados

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 1: executar o iperf em 3 computadores

Neste experimento, nós configuramos 3 máquinas com o o RC do tipo *node*, o qual representa uma máquina do tipo PC. A Figura 3.1 mostra o cenário do experimento. Nós utilizamos o EC para controlar a aplicação *iperf* nas três máquinas, executando o módulo cliente e configurando-as para enviarem requisições para outro computador com o servidor *iperf* executando. O experimento foi bem sucedido, mostrando o potencial do OMF 6 para controlar aplicações.

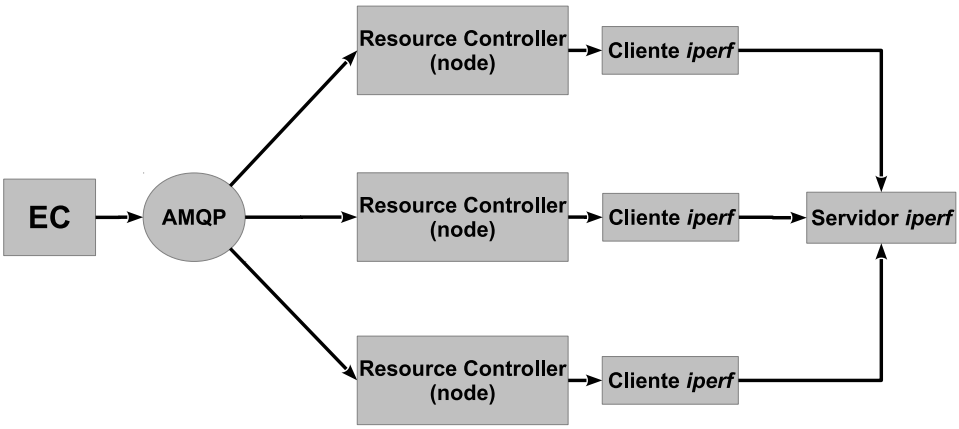

Figura 3.1: *Cenário do Experimento 1.*

3.1.2 Experimento 2: criar uma *bridge* no Open vSwitch

O RC para controlar o Open vSwitch (virtual_openflow_switch [1]) não está entre os RCs padrões do OMF 6. Ele foi desenvolvido pela equipe do NITlab e deve ser instalado a parte. Sua última atualização

foi em 2013 e ele não funciona com as versões mais novas do OMF 6. Após modificações no código, fomos capazes de executá-lo.

Um experimento descrito em OEDL foi executado no EC, o qual se comunica, via XMPP, com o RC do Open vSwitch, cria uma *bridge* chamada “test” no *switch* e, após isso, configura duas portas nessa *bridge*. A Figura 3.2 ilustra a execução do experimento. Embora seja recomendado utilizar o protocolo AMQP no OMF 6, o RC do Open vSwitch ainda apresenta problemas na utilização desse protocolo. Por esse motivo, utilizamos o XMPP nesse experimento.

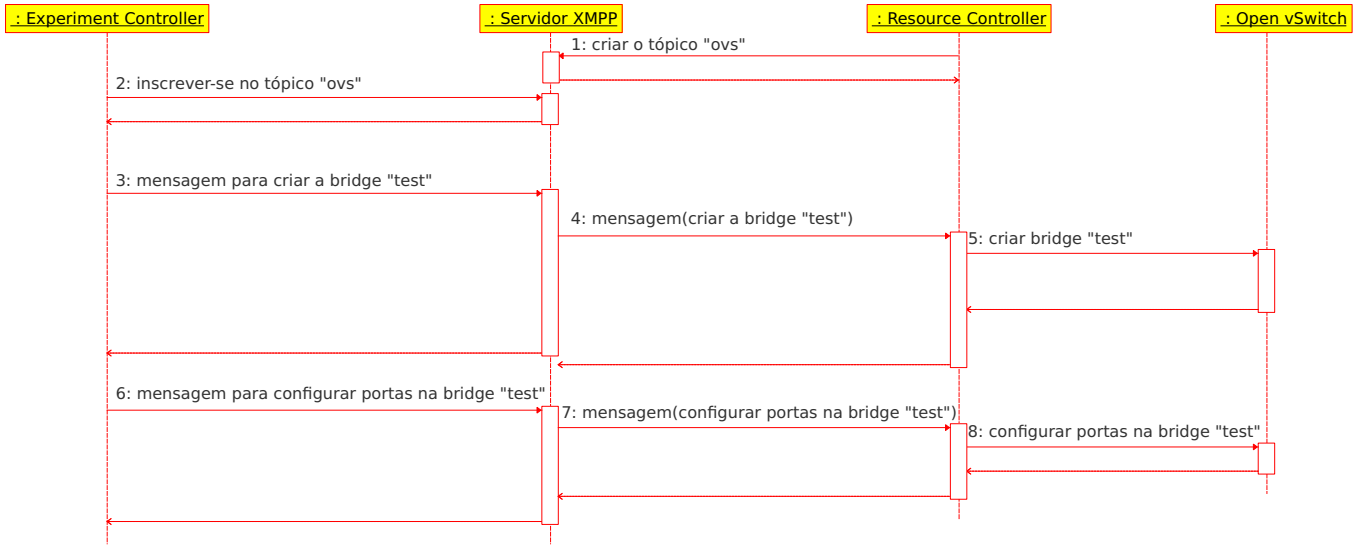

Figura 3.2: *Diagrama de sequências do experimento com o Open vSwitch.*

3.1.3 Experimento 3: criar um slice no FlowVisor

O RC para controlar o FlowVisor (openflow_slice [1]) está no mesmo pacote do RC para controlar o Open vSwitch e, portanto, apresenta os mesmos problemas de versão e com o uso do AMQP. Dessa forma, foi necessário fazer algumas modificações no código para que ele funcionasse corretamente.

A Figura 3.3 apresenta o diagrama de sequências que descreve a execução do Experimento 3. O experimento foi descrito em OEDL e executado no EC, o qual se comunica, via XMPP, com o RC para criar um *slice* (passo 5) e, em seguida, adicionar um *flowspace* a esse *slice* (passo 8).

3.1.4 Experimento 4: controlar o KVM

Neste experimento, utilizamos o RC do tipo virtual_machine, o qual é capaz de se comunicar com um hipervisor para controlar todo o ciclo de vida de uma máquina virtual. A implementação padrão do RC virtual_machine é capaz de se comunicar apenas com o KVM, contudo, ele foi desenvolvido de maneira a facilitar sua extensão para controlar outros hipervisores. Para isso, basta implementar um conjunto de métodos descrito em sua documentação e alterar o valor de duas propriedades do RC [5].

No Experimento 4, utilizando o protocolo AMQP, enviamos comandos para o RC virtual_machine de maneira a executar as seguintes tarefas:

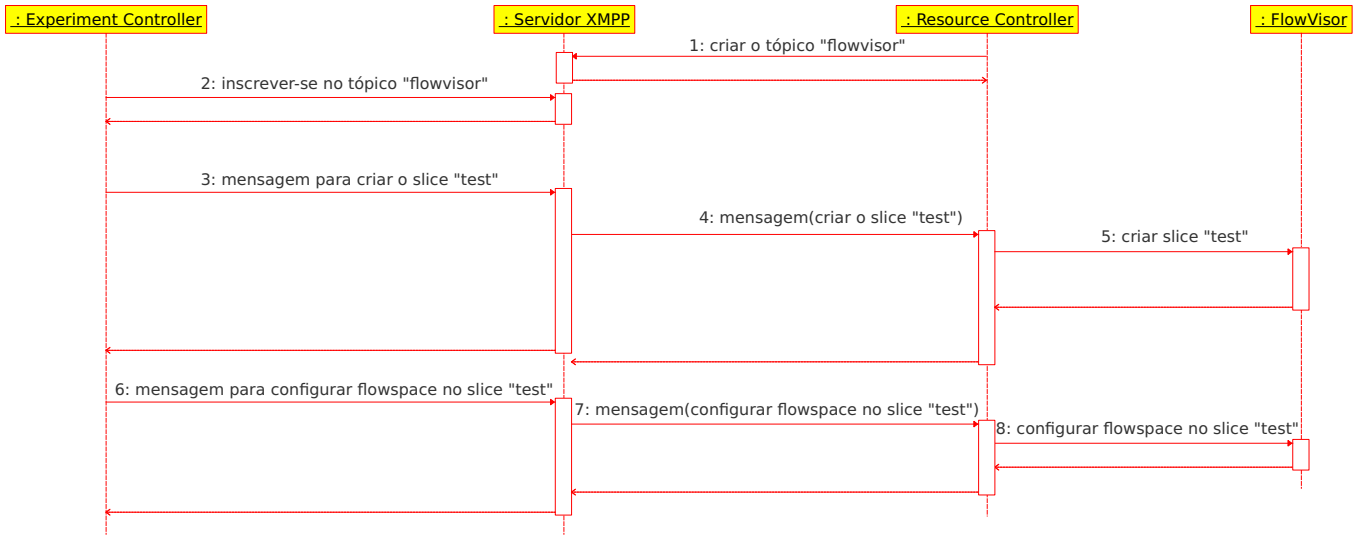

Figura 3.3: *Diagrama de sequências do experimento com o FlowVisor.*

- Criar uma nova máquina virtual
- Clonar uma máquina virtual
- Iniciar uma máquina virtual
- Desligar uma máquina virtual
- Conectar uma máquina virtual já existente ao RC
- Excluir uma máquina virtual

Apesar do RC virtual_machine ter funcionado corretamente, encontramos alguns pontos de sua implementação que devem ser melhorados. Ele armazena o estado e outras informações da última máquina virtual acessada em propriedades do RC. Dessa forma, por exemplo, quando se liga a máquina virtual A, a variável *state* (que representa o estado) recebe o valor *running* (executando). Ao tentar ligar uma máquina virtual B, uma exceção é gerada informando que ela já está ligada, pois o RC observa o valor da propriedade *state*. Assim, para se ligar a máquina virtual B, é necessário primeiro configurar a propriedade *state* para *stopped* (parada) e em seguida chamar o método para ligá-la.

Essa implementação apresenta problemas de concorrência. Além disso, o RC não tem conhecimento do estado de todas as máquinas virtuais existentes e ainda pode apresentar inconsistências no valor do estado, como mostrado no exemplo do parágrafo anterior. A maneira correta, seria implementar um método para consultar o estado da VM diretamente no hipervisor e consultá-lo antes de requisitar uma alteração de estado.

3.1.5 Experimento 5: utilizar os RCs da NITOS Testbed

Os RCs da NITOS Testbed foram desenvolvidos pela equipe do NITlab e são responsáveis pela parte de carregamento de imagens e de controle dos CMs. Eles se comunicam com o Broker para solicitar informações dos recursos no inventário e, assim, enviar seus comandos aos recursos. Por meio de um *script* desenvolvido pelo NITlab e que é instalado juntamente com os RCs, nós realizamos os seguintes experimentos:

- Carregar imagens nos nós ICARUS
- Salvar imagens dos nós ICARUS
- Ligar os nós ICARUS
- Desligar os nós ICARUS
- Reiniciar os nós ICARUS

Como dito no capítulo anterior, o firmware dos nós ICARUS foi atualizado para que funcionasse corretamente com os RCs da NITOS Testbed.

3.2 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. Todos os RCs testados funcionaram corretamente, apesar de que alguns RCs ainda apresentam deficiências na implementação e precisam ser atualizados para criar uma *testbed* robusta e que opere totalmente com AMQP. No entanto, a API do OMF 6 oferece uma base consistente para que essas modificações sejam realizadas e para que novos RCs sejam implementados. A princípio, um dos RCs necessários será um para se comunicar com Xen, visto que esse hipervisor é utilizado nas ilhas do FIBRE.

CAPÍTULO 4

Conclusão e Avaliação

A implantação da *testbed* foi bem sucedida e, conseguimos, a partir disso, obter uma melhor percepção das funcionalidades, do potencial e das limitações do OMF 6. Analisamos tanto os RCs básicos, quanto os RCs da parte OpenFlow e, ainda, analisamos os RCs da NITOS Testbed. Obtivemos neste período uma grande experiência com a instalação e configuração de todos os módulos do OMF 6 e de outros programas que compõem uma *testbed* operacional. Trabalhamos com o Broker, utilizando a parte de inventário para cadastrar recursos, contudo, ainda existem funcionalidades a serem exploradas, como a reserva de recursos, por exemplo.

O OMF 6 possui uma arquitetura madura e uma API que possibilita e simplifica a criação de controladores para qualquer tipo de recurso de *software* e *hardware*. Essa API abstrai a camada de comunicação e, com diretivas bem definidas, permite instanciar, configurar e requisitar informações sobre os recursos. O OMF 6 suporta diferentes servidores *Publish/Subscribe*, não sendo acoplado ao XMPP, como era o OMF 5.4. Outra vantagem da versão 6 é que ela contempla controladores para a parte OpenFlow de uma *testbed* e está sendo evoluída. Não menos importante, o suporte por parte das equipes desenvolvedoras é efetivo, o qual já não ocorre com a versão legada. A equipe do NITlab se mostrou bastante solícita, acompanhando e ajudando em várias etapas da implantação da *testbed*.

Por todas as características e vantagens citadas, acreditamos que o OMF 6 tenha potencial para substituir o OMF 5.4 e o OCF na infraestrutura do FIBRE, embora ajustes ainda sejam necessários. Como apresentado no Capítulo 3, alguns RCs apresentam limitações e precisam ser evoluídos. Contudo, a nova arquitetura está madura e simplifica as correções dessas limitações e a implementação de novos RCs. A adoção do OMF 6 poderá simplificar também a arquitetura do FIBRE, pois, a retirada do OCF representaria um módulo complexo a menos.

A implantação do OMF 6 em uma ilha do FIBRE exigirá um esforço prévio para corrigir problemas em alguns RCs e para padronizar a comunicação de todos os módulos utilizando o protocolo AMQP. Para isso, seriam necessários desenvolvedores com experiência na linguagem de programação *ruby* e em sistemas distribuídos, principalmente com o padrão *Publish/Subscribe*. O OMF 6 é todo implementado em *ruby* e sua comunicação é realizada através de servidores *Publish/Subscribe*, o que justifica os requisitos apresentados. É importante também que os desenvolvedores tenham experiência com Linux e em configuração de infraestruturas distribuídas. Muitos dos módulos do OMF 6 exigem um grande esforço para configuração e é essencial que o desenvolvedor tenha facilidade para resolver problemas utilizando a linha de comando. Por fim, é interessante que os desenvolvedores tenham habilidade para se comunicar em inglês, pois a comunicação com as equipes mantenedoras do OMF 6, como NITlab e NICTA, é realizada nesse idioma e é essencial para a evolução do projeto.

Com o conhecimento adquirido nos microprojetos, estimamos que para implantar o OMF 6 em uma ilha do FIBRE no período de 1 ano seriam necessários 2 desenvolvedores que, de preferência, atendam aos requisitos informados no parágrafo anterior. Na Tabela 4.1, apresentamos o tempo estimado das

principais atividades necessárias para a implantação. Durante a fase de adaptação do OMF 6 ao FIBRE será necessária a criação de alguns RCs, cujo tempo de desenvolvimento pode variar dependendo da complexidade. O RC para controlar o Xen, por exemplo, que pode ser considerado um RC de média complexidade, poderá levar cerca de duas semanas para ser implementado. Por outro lado, um RC mais complexo, como de controle de acesso de usuários, poderia levar até um mês ou mais para ser finalizado.

Tarefa	**Estimativa**
Ambientação com o OMF 6 e com os outros programas da *testbed*	3 meses
Evolução dos RCs que apresentaram problemas	1 mês
Evolução do Broker para funcionar com AMQP	2 meses
Adaptação ao FIBRE e testes	6 meses

Tabela 4.1: *Estimativa das atividades necessárias para implantação do OMF 6 em uma ilha do FIBRE.*

Referências Bibliográficas

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

[2] FIBRE. **Ofelia control monitoring framework**. http://www.fibre-ict.eu/cmf/ofelia, 2015. [Último acesso: 29-Dezembro-2015].

[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. **virtual_machine - github**. https://github.com/mytestbed/omf/blob/master/omf_rc/lib/omf_rc/resource_proxy/virtual_machine.rb, 2015. [Último acesso: 29-Dezembro-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.

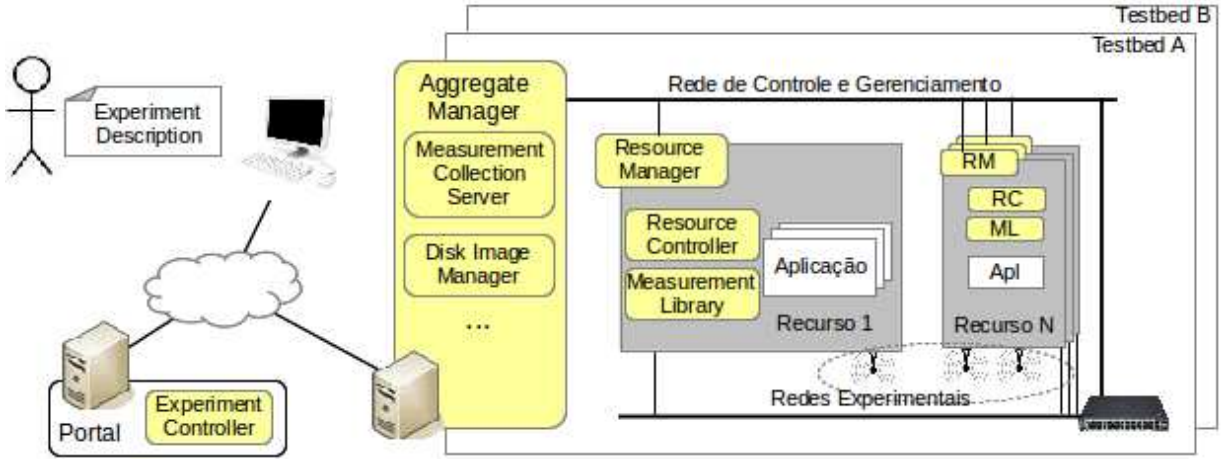

Figura A.1: *Componentes que integram a arquitetura do OMF 5.4.*

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.