OpenFlow e redes definidas por software: um novo paradigma de controle e inovação em redes de pacotes

Christian Esteve Rothenberg*, Marcelo Ribeiro Nascimento, Marcos Rogério Salvador, Maurício Ferreira Magalhães**

Há algum tempo observa-se o surgimento da necessidade de maior abertura e flexibilidade dos equipamentos de rede, especialmente com o propósito de pesquisa e inovação. Os roteadores atuais implementam uma arquitetura composta por uma camada de software fechada, que é executada em um hardware proprietário. Esse modelo resulta em soluções de alto custo, dificulta a inserção de novas funcionalidades e inviabiliza a experimentação de novas ideias. Com o avanço da padronização do protocolo OpenFlow para programar o plano de encaminhamento dos equipamentos de redes de pacotes, o conceito de redes programadas por software traz um novo paradigma de inovação cujo potencial disruptivo assemelha-se ao da introdução de sistemas operacionais em computadores e, mais recentemente, em dispositivos móveis. Este artigo introduz os princípios da tecnologia OpenFlow e apresenta a arquitetura RouteFlow como exemplo de inovação para a função de roteamento em um ambiente aberto (open source), executado sobre hardware comercial (commodity).

Palavras-chave: Redes de pacotes. Roteamento. Encaminhamento. Software livre. Virtualização.

Introdução

A infraestrutura das redes de pacotes é composta, atualmente, por equipamentos proprietários, fechados e de alto custo, cujas arquiteturas básicas são concebidas a partir da combinação de circuitos dedicados, responsáveis por garantir alto desempenho (Application Specific Integrated Circuit – ASIC), ao processamento de pacotes. A infraestrutura é complementada por uma camada de software de controle, responsável pelo suporte a uma extensa pilha formada por um número elevado de protocolos. No entanto, torna-se evidente a necessidade de especialização da lógica de controle de acordo com cada tipo e objetivo de rede. Porém, qualquer mudança de configuração avançada do equipamento ou especialização da lógica de controle e tratamento dos pacotes ou, ainda, inserção de novas funcionalidades estão sujeitas a ciclos de desenvolvimento e de testes restritos ao fabricante do equipamento, resultando em um processo demorado e custoso (HAMILTON, 2009). No âmbito da pesquisa, essas exigências são ainda maiores pois requerem experimentações de novos protocolos e funcionalidades da rede em condições reais, idealmente em ambientes "espalhados" da rede em operação (SHERWOOD et al., 2010). A ausência de flexibilidade no controle do funcionamento interno dos equipamentos assim como o alto custo da infraestrutura vigente são barreiras para a evolução das arquiteturas e para a inovação decorrente da oferta de novos serviços e aplicações de rede (ANWER et al., 2010).

Este artigo faz um levantamento do estado da arte da tecnologia OpenFlow como quebra de paradigma de controle em software (remoto) dos equipamentos de rede, habilitando o conceito de redes definidas por software. Como exemplo de utilização, apresenta-se um trabalho em desenvolvimento, voltado para uma arquitetura de roteamento denominada RouteFlow (NASCIMENTO et al., 2011), cujo controle é realizado remotamente via software. Essa arquitetura é implementável em produtos comerciais e utiliza ambientes de software de domínio público para implementação das funções de controle.

1 Redes definidas por software

Apesar da evolução formidável da Internet, em termos de penetração e de aplicações, sua tecnologia, representada pela arquitetura em camadas e pelos protocolos do modelo TCP/IP, não evoluiu suficientemente nos últimos vinte anos. A Internet tornou-se comercial e os equipamentos de rede tornaram-se "caixas pretas", ou seja, implementações integradas verticalmente baseadas em software fechado sobre hardware proprietário. O resultado desse modelo é o já reconhecido engessamento da Internet (CHOWDHURY; BOUTABA, 2009).

Em contraste com o desenho da arquitetura da Internet, caracterizada por um plano de controle (PC) distribuído, os avanços na padronização de APIs (Application Programming Interface) independentes do fabricante do equipamento

*Autor a quem a correspondência deve ser dirigida: esteve@cpqd.com.br.
** Faculdade de Engenharia Elétrica e de Computação – Unicamp.

(OPENFLOW, 2010; IETF, 2011) permitem mover grande parte da lógica de tomada de decisão dos dispositivos de rede para controladores externos, que podem ser implementados com o uso da tecnologia de servidores comerciais (PCs), um recurso abundante, escalável e barato. Essa "lobotomia" da inteligência do equipamento da rede para controladores logicamente centralizados possibilita a definição do comportamento da rede em software não apenas pelos fabricantes do equipamento, mas também por fornecedores ou pelos próprios usuários, como, por exemplo, operadores de rede.

1.1 Arquiteturas de roteamento

Uma análise da arquitetura atual dos roteadores (Figura 1) permite observar que se trata de um modelo formado basicamente por duas camadas bem distintas: o software de controle e o hardware dedicado ao encaminhamento de pacotes (JUNIPER NETWORKS, 2010). O primeiro, encarregado de tomar as decisões de roteamento, transfere essas decisões para o plano de encaminhamento através de uma API proprietária. A única interação da gerência com o dispositivo ocorre através de interfaces de configuração (Web, SNMP, CLI, por exemplo), limitando o uso dos dispositivos às funcionalidades programadas pelo fabricante.
É coerente pensar que se a arquitetura é, atualmente, composta por duas camadas autocontidas, elas não precisam estar fechadas em um mesmo equipamento. Para isso, basta que exista uma forma padrão de se programar o dispositivo de rede remotamente, permitindo que a camada de controle possa ser movida para um servidor dedicado e com alta capacidade de processamento. Desse modo, mantém-se o alto desempenho no encaminhamento de pacotes em hardware aliado à flexibilidade de se inserir, remover e especializar aplicações em software por meio de um protocolo aberto para programação da lógica do equipamento (Figura 1). Com esse propósito, nasceu o consórcio OpenFlow (MCKEOWN et al., 2008), dando origem ao conceito de software-defined networking – as redes definidas por software (GREENE, 2009).

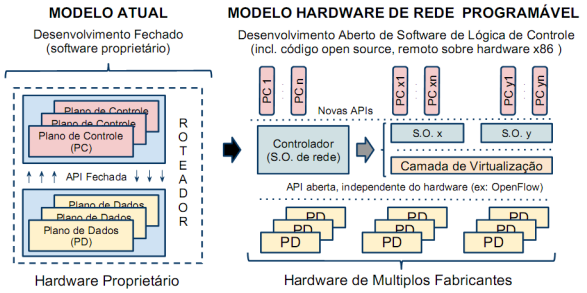


Figura 1 Arquiteturas de roteadores: modelo atual *mainframe* e modelo programável OpenFlow

1.2 OpenFlow

O OpenFlow foi proposto pela Universidade de Stanford para atender à demanda de validação de novas propostas de arquiteturas e protocolos de rede (incluindo as abordagens *clean slate*) sobre equipamentos comerciais. OpenFlow define um protocolo-padrão para determinar as ações de encaminhamento de pacotes em dispositivos de rede, como, por exemplo, comutadores, roteadores e pontos de acesso sem fio. As regras e ações instaladas no hardware de rede são responsabilidade de um elemento externo, denominado controlador, que pode ser implementado em um servidor comum, conforme Figura 2.

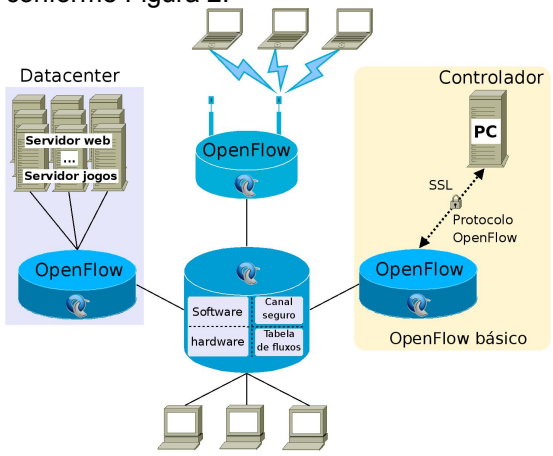


Figura 2 Exemplo de uma rede com OpenFlow habilitado

A principal abstração utilizada na especificação OpenFlow é o conceito de fluxo. Um fluxo é constituído pela combinação de campos do cabeçalho do pacote a ser processado pelo dispositivo, conforme Figura 3. As tuplas podem ser formadas por campos das camadas de enlace, de rede ou de transporte, segundo o modelo TCP/IP. Deve-se enfatizar que a abstração da tabela de fluxos ainda está sujeita a refinamentos, com o objetivo de oferecer uma melhor exposição dos recursos do hardware e, nesse caso, permitir a concatenação de várias tabelas já disponíveis, como, por exemplo, tabelas IP/Ethernet/MPLS. Nesse sentido, a contribuição mais importante do paradigma do OpenFlow é a generalização do plano de dados – qualquer modelo de encaminhamento de dados baseado na tomada de decisão fundamentada em algum valor, ou combinação de valores, dos campos de cabeçalho dos pacotes pode ser suportado.

inport	Ethernet			VLAN		IP				TCP/UDP	
	src	dst	type	id	pri	src	dst	proto	ToS	src	dst

Figura 3 Cabeçalhos disponíveis no OpenFlow para a especificação de fluxos

De forma pragmática, a especificação OpenFlow (OPENFLOW, 2010) procura reutilizar as funcionalidades do hardware existente (por

exemplo, Access Control List – ACL em *switches* e roteadores para implementar serviços como NAT, *firewall* e VLANs) através da definição de um conjunto simples de regras e das ações associadas: encaminhar, descartar, enviar para o controlador, reescrever campos do cabeçalho, etc.

1.2.1 Componentes

Basicamente, uma rede programável com OpenFlow consiste em equipamentos de rede habilitados para que o estado das tabelas de fluxos possa ser instalado através de um canal seguro, conforme as decisões de um controlador em software:

1.2.1.1 Tabela de fluxos

Cada entrada na tabela de fluxos do hardware de rede consiste em regra, ações e contadores. A regra é formada com base na definição do valor de um ou mais campos do cabeçalho do pacote. Associa-se a ela um conjunto de ações que definem o modo como os pacotes devem ser processados e para onde devem ser encaminhados. Os contadores são utilizados para manter estatísticas de utilização e para remover fluxos inativos. As entradas da tabela de fluxos podem ser interpretadas como decisões em cache (hardware) do plano de controle (software), sendo, portanto, a mínima unidade de informação no plano de dados da rede.

1.2.1.2 Canal seguro

Para que a rede não sofra ataques de elementos mal-intencionados, o canal seguro garante confiabilidade na troca de informações entre o *switch* e o controlador. A interface de acesso recomendada é o protocolo SSL (Secure Socket Layer). Interfaces alternativas (passivas ou ativas) incluem TCP e pcap (*packet capture*), e são especialmente úteis em ambientes virtuais e experimentais pela simplicidade de utilização, pois não necessitam de chaves criptográficas.

1.2.1.3 Protocolo OpenFlow

Protocolo aberto para comunicação que usa uma interface externa, definida pelo OpenFlow para a troca de mensagens entre os equipamentos de rede e os controladores. Essas mensagens podem ser simétricas (*hello, echo vendor*), assíncronas (*packet in, flow removed, port status, error*) ou, ainda, iniciadas pelo controlador (*features, configuration, modify state, send packet, barrier*).

1.2.1.4 Controlador

É o software responsável por tomar decisões e adicionar e remover as entradas na tabela de fluxos, de acordo com o objetivo desejado. O controlador exerce a função de uma camada de abstração da infraestrutura física, facilitando a criação de aplicações e serviços que gerenciem as entradas de fluxos na rede. Esse modelo assemelha-se a outros sistemas de software que proveem abstração do hardware e funcionalidade reutilizável. Dessa forma, o controlador OpenFlow atua como um sistema operacional (SO) para gerenciamento e controle das redes, e oferece uma plataforma com base na reutilização de componentes e na definição de níveis de abstração (comandos da API). Contudo, novas aplicações de rede podem ser desenvolvidas rapidamente (GUDE et al., 2008). A programabilidade do controlador permite a evolução em paralelo das tecnologias nos planos de dados e as inovações na lógica das aplicações de controle. A Figura 4 mostra uma abstração de uma rede com OpenFlow e o controlador NOX executando inúmeras aplicações que necessitam de uma visão do estado da rede (*network view*). Essa visão pode ser armazenada, por exemplo, em um simples banco de dados executado localmente ou em um servidor remoto.

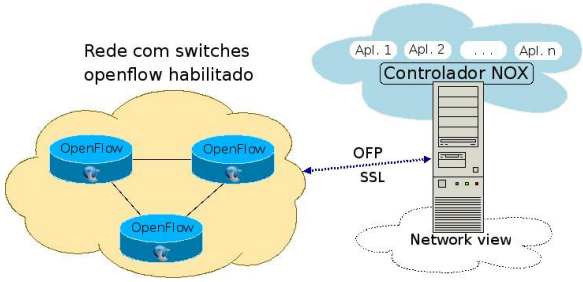


Figura 4 Elementos de uma rede OpenFlow

1.2.2 OpenFlow em ação

Quando um pacote chega a um equipamento com OpenFlow habilitado, os cabeçalhos do pacote são comparados às regras das entradas das tabelas de fluxos, os contadores são atualizados e as ações correspondentes são realizadas. Se não houver correspondência entre o pacote e alguma entrada da tabela de fluxos, o pacote é encaminhado, por completo, ao controlador. Alternativamente, apenas o cabeçalho é encaminhado ao controlador mantendo o pacote armazenado no buffer do hardware.

Normalmente, os pacotes que chegam ao controlador correspondem ao primeiro pacote de um novo fluxo ou, em função do tipo de pacote e da aplicação, o controlador pode optar por instalar uma regra no *switch* para que todos os pacotes de determinado fluxo sejam enviados para o controlador para serem tratados individualmente. Esse último caso corresponde, em geral, a pacotes de controle (ICMP, DNS, DHCP) ou de protocolos de roteamento (OSPF, BGP).

Como exemplo de fluxo, têm-se todos os pacotes

em uma faixa de endereços IP (roteamento IP tradicional), uma conexão TCP em uma porta específica ou, ainda, todos os pacotes pertencentes a uma mesma VLAN. As ações associadas aos fluxos incluem:

a) encaminhar o fluxo de pacotes para determinada porta (ou portas);
b) modificar os campos do cabeçalho;
c) encapsular e transmitir o pacote para o controlador;
d) descartar os dados, como medida de segurança, com a implementação de *firewalls*, ou ainda para inibir ataques de negação de serviço;
e) encaminhar o pacote para o processamento normal do equipamento nas camadas 2 ou 3.

O último item garante que o tráfego experimental não interfira no processamento-padrão do tráfego de produção. Conforme ilustrado na Figura 5, outra forma de garantir isso é definir um conjunto de VLANs para fins experimentais. Essa segmentação do tráfego permite ter equipamentos híbridos que processem tráfego legado conforme os protocolos e as funcionalidades embarcadas no equipamento e, ao mesmo tempo e de forma isolada, obter tráfego baseado na programabilidade do OpenFlow.

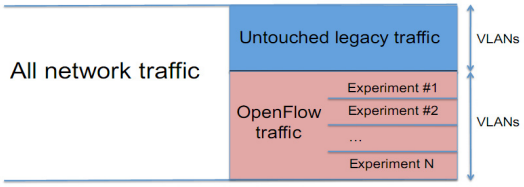


Figura 5 Modelo de operação híbrido com VLANs isolando tráfego legado e OpenFlow

Na versão 1.1 do protocolo OpenFlow (ainda sob especificação), está sendo considerado o suporte a múltiplas tabelas de fluxos concatenadas (Figura 6) mediante um novo conjunto de instruções (*Go-to-Table*, *Write-Metadata*), bem como novas ações (*copy/decrement TTL, push/pop tag, QoS*) e campos de cabeçalho (*VLAN priority, MPLS label/traffic class, SCTP port*) para uma definição ainda mais completa dos fluxos e do conjunto de ações associadas.

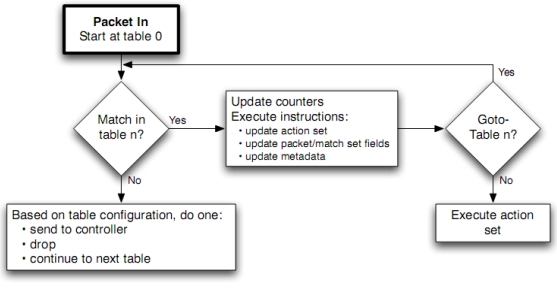


Figura 6 Diagrama detalhando o tratamento de um pacote no *pipeline* de um *switch* OpenFlow

1.2.3 Fatores críticos de sucesso

Existem quatro razões fundamentais que contribuem para a aceitação da tecnologia OpenFlow:

a) o OpenFlow pode ser incorporado em equipamentos de rede (roteadores, *switches*, pontos de acesso Wi-Fi) comerciais, atualmente, em operação, sem modificação do hardware e mediante uma atualização do *firmware*, garantindo o desempenho de tecnologias consolidadas no encaminhamento de pacotes IP/Ethernet, como, por exemplo, ASICs, FPGAs, etc.;
b) o protocolo OpenFlow separa o plano de controle do plano de dados, permitindo a utilização de controladores remotos baseados em servidores com sistemas operacionais e linguagens de programação comuns na indústria de TI. O software do controlador é responsável por definir o modo como os fluxos de pacotes são encaminhados e processados na rede. Isso permite que o controle sobre o plano de dados seja "terceirizado" a pesquisadores, sistemas de gerência, desenvolvedores, operadores de rede, plataformas de serviços e, até mesmo, às próprias aplicações finais, como, por exemplo, servidores de conteúdo ou serviços em nuvem. Dessa forma, o controle da rede deixa de estar embarcado nos equipamentos e limitado por implementações e padrões com mais de 15 anos de existência;
c) o OpenFlow não dependente da forma como o software controla a rede, e oferece um simples serviço de encaminhamento multicamada (L1-L4) orientado a fluxos definidos por qualquer combinação de mais de 20 cabeçalhos-padrão. Uma rede OpenFlow permite a definição de fatias de rede (*slices* ou *flow-spaces*), com garantia de isolamento entre os diferentes controladores que operam sobre a rede, permitindo que o tráfego operacional (conforme os protocolos tradicionais) e o tráfego experimental (conforme definido pelo usuário/operador da rede) operem em paralelo;
d) o OpenFlow é compatível com a Internet atual, cujo tráfego pode continuar em operação em uma ou mais fatias da rede OpenFlow.

Todos os argumentos citados, anteriormente, apontam o OpenFlow como uma tecnologia inovadora que abre uma série de oportunidades de desenvolvimento tecnológico na área das redes de pacotes. Com a consolidação das tecnologias de equipamentos programáveis em software no padrão OpenFlow, ou tecnologias

similares, o conceito de redes definidas por software envolve novos paradigmas de gerência integrada, inovação em protocolos e serviços baseados em recursos de redes virtualizados, como, por exemplo, *Network as a Service* (CHOWDHURY; BOUTABA, 2009; KELLER; REXFORD, 2010).

1.3 Mercado e cenários de aplicação

Os fatores apontados anteriormente e o potencial disruptivo da tecnologia OpenFlow têm atraído a atenção da indústria, o que tem se traduzido:

a) no desenvolvimento dos primeiros protótipos com suporte ao OpenFlow (HP, NEC, Extreme, Arista, Ciena, Juniper, Cisco);
b) no suporte dos fornecedores de processadores em silício (Broadcom, Marven);
c) na criação de novas empresas (Nicira, Big Switch Networks); e
d) no interesse de operadoras, como Deutsche Telekom e Docomo, e provedores de serviços em nuvem, como Google, Facebook e Amazon).

Entre os vários cenários de rede em que a adoção da tecnologia OpenFlow traz aspectos promissores, vale destacar:

a) **redes corporativas:** novos mecanismos de controle de acesso e segurança, gerência integrada de rede cabeada e sem fio, configuração de VLANs, suporte à mobilidade, etc. (CASADO et al., 2007);

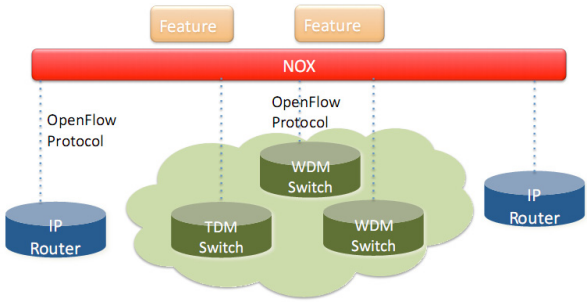
b) ***backbone*:** convergência de redes de pacotes e circuitos, (Figura 7), como, por exemplo, agregação e gerência dinâmica e flexível do tráfego, novos mecanismos de roteamento e engenharia de tráfego e recuperação de falhas; balanceamento do tráfego Web; mobilidade de máquinas virtuais; etc. (GUDLA et al., 2010);

c) **redes celulares:** uso transparente de diversas redes de acesso (Wi-Fi/3G/WiMAX), separação do provedor de infraestrutura do provedor de serviços (por exemplo, virtual network operators), etc. (YAP et al., 2010);

d) **data center:** técnicas de conservação de energia, engenharia de tráfego, roteamento plano e multicaminho, suporte à virtualização de *hosts* e *software switches* (KOPONEN et al., 2010);

e) **redes domésticas:** terceirização (*outsourcing*) da gerência de rede, compartilhamento da rede com vários provedores de serviços e usuários, como, por exemplo, Open Wi-Fi, e gerência de energia com medidores inteligentes, como *smart grid*;

f) **redes de ensino e pesquisa:** infraestrutura de experimentação de novas arquiteturas de rede e paradigmas de encaminhamento de pacotes, compartilhamento de uma infraestrutura experimental multicamada, multidomínio e multitecnologia, validação sobre hardware comercial, etc.



Fonte: GUDLA et al., 2010

Figura 7 Rede híbrida com *switches* TDM, WDM e roteadores IP controlados por OpenFlow (NOX), que implementa as funcionalidades em software

2 Arquitetura RouteFlow

O RouteFlow é uma proposta de oferta de serviços de roteamento IP remoto de forma centralizada, e que visa um desacoplamento efetivo entre o plano de encaminhamento e o plano de controle (ROUTEFLOW, 2011). O objetivo é tornar as redes IP mais flexíveis pela facilidade de adição, remoção e especialização de protocolos e algoritmos. O RouteFlow armazena a lógica de controle dos *switches* OpenFlow na infraestrutura de rede, através de uma rede virtual composta por máquinas virtuais (MV), cada uma executando um código (*engine*) de roteamento de domínio público (*open source*). Essas MVs podem ser interconectadas de maneira a formar uma topologia lógica, espelhando a topologia de uma rede física correspondente ou uma topologia virtual simplificada. O ambiente virtual é armazenado em um servidor externo, ou um conjunto deles, que se comunicam com os equipamentos do plano de dados através de um controlador OpenFlow, que transporta para o plano de encaminhamento as decisões tomadas pelos protocolos de roteamento no plano de controle (OSPF, BGP, RIP). A Figura 8 ilustra uma sub-rede com *switches* programáveis, em que a lógica de roteamento é implementada no servidor RouteFlow.

O resultado consiste numa solução flexível de alto desempenho e comercialmente competitiva, a partir da combinação de recursos disponíveis, como, por exemplo:

a) *switches* programáveis de baixo custo e software embarcado reduzido (OpenFlow);
b) pilhas de protocolos de roteamento *open source* (QUAGGA, 2009; XORP, 2011); e

c) servidor de prateleira de alto poder de processamento e, também, de baixo custo.

Cabe ressaltar que, apesar de o controle estar fisicamente centralizado, ele continua distribuído logicamente. Dessa forma, não é necessária qualquer alteração dos protocolos de roteamento existentes. Além disso, a solução pode tornar-se mais escalável no futuro, com o uso de vários servidores de alto desempenho.

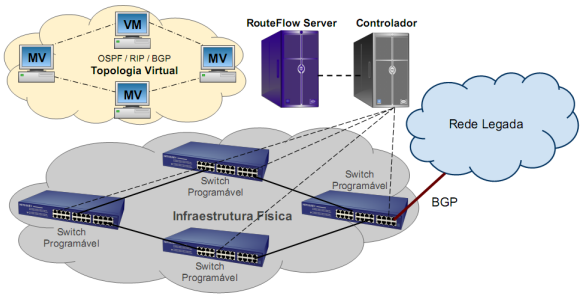


Figura 8 Visão geral do RouteFlow

2.1 Blocos funcionais

Uma visão global dos principais componentes do RouteFlow pode ser observada na Figura 9. A seguir, apresenta-se a descrição de cada um dos componentes da arquitetura proposta.

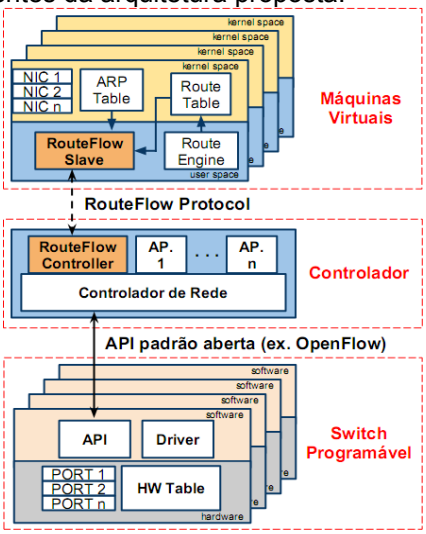


Figura 9 Componentes da solução RouteFlow

2.1.1 RouteFlow-Slave (RF-S)

Cada MV do plano de controle executa um processo (*daemon*) responsável pelas seguintes funções:

a) registrar a MV no controlador como recurso da topologia virtual, sendo que a MV se torna um recurso disponível para ser utilizado na representação de um roteador;
b) gerenciar as interfaces de rede do sistema (portas do roteador);
c) detectar as atualizações das tabelas Address Resolution Protocol (ARP) e de roteamento do sistema; e
d) converter rotas em fluxos a serem instalados no plano de dados por meio da API do OpenFlow.

Pacotes de protocolos – e outros, cujos destinos sejam o próprio roteador – são encaminhados e recebidos pela MV para processamento (ARP, ICMP, Telnet, SSH, OSPF, RIP, etc.). Pacotes ARP e ICMP, por exemplo, são tratados pela pilha TCP/IP do Linux, enquanto pacotes de protocolos são utilizados pela *engine* de roteamento para cálculo de rotas. O processamento no caminho inverso ocorre do mesmo modo. Concomitantemente ao processamento de pacotes da MV, um mecanismo de *polling* executa a tarefa de verificar as tabelas ARP e ROUTE do sistema em busca de atualizações que devem ser reproduzidas no plano de dados. Quando atualizações são detectadas, as modificações correspondentes são convertidas na instalação ou remoção de fluxos da tabela do elemento de encaminhamento atribuído à MV.

2.1.2 RouteFlow-Controller (RF-C)

É o componente central da arquitetura proposta. Como o próprio nome sugere, trata-se de uma função de controle que conecta todos os elementos do plano de dados e as MVs do plano de controle. As principais funções do RF-C são:

a) registrar-se no controlador de rede para eventos de recebimento de pacotes, conexão e desconexão de *switches*;
b) registrar os recursos (MVs) disponíveis na topologia virtual;
c) fazer a configuração do *software switch* para montar a topologia lógica da rede;
d) gerenciar o mapeamento entre *switches* OpenFlow e MVs;
e) instalar/remover fluxos OpenFlow dos *switches* do plano de dados.

A vinculação das máquinas virtuais aos *switches* do plano de dados pode ser feita estaticamente através de um arquivo de configuração, que permite configurar uma topologia virtual independentemente da topologia física. Alternativamente, a configuração pode ser dinâmica, com base em um mecanismo de descoberta de topologia governado pelo controlador de rede. Nesse último caso, a rede virtual será uma réplica da topologia física. Outra possibilidade consiste em agregar um conjunto de nós físicos, como, por exemplo, *switch stacking/trunking,* em uma única MV no plano virtual, ou ter várias *engines* de roteamento atuando sobre um mesmo elemento físico – o que remete ao conceito de roteadores virtuais, em que múltiplos planos de controle com objetivos diferentes compartilham o mesmo substrato hardware de rede (CHOWDHURY; BOUTABA, 2009).

2.1.3 RouteFlow-Protocol (RF-P)

Protocolo desenvolvido na arquitetura proposta para a comunicação entre os componentes do RouteFlow. Nele, estão definidas as mensagens e os comandos básicos para conexão e configuração das MVs e, também, gerenciamento das entradas de roteamento em hardware. Entre os campos da mensagem-padrão estão: identificação do controlador, identificação da MV, tipo da mensagem, comprimento e dados.

3 Avaliação do protótipo RouteFlow

O RF-Controller foi implementado em C++ como uma aplicação do controlador *open source* NOX (2011). Como *engine* de roteamento, foi utilizado o Quagga (2009), uma conhecida solução de roteamento *open source* com suporte aos principais protocolos de roteamento (RIP, OSPF, BGP). Como plataforma para programação remota dos equipamentos de rede, utilizou-se a versão 1.0 da tecnologia OpenFlow, cuja vantagem é sua abordagem multifornecedor. Isso significa que ele não depende do fabricante e do modelo do equipamento que compõe a rede. Desde que haja suporte ao protocolo OpenFlow, não são necessárias mudanças no controlador nem nas aplicações de rede que executam sobre ele. A infraestrutura do plano de dados do protótipo é formada por NetFPGAs, que são hardware programáveis com quatro interfaces de rede Gigabit e com módulo OpenFlow. A escolha da plataforma NetFPGA se deu pela flexibilidade de programação do plano de encaminhamento e pela taxa de processamento de pacotes em Gigabits. Também foi usado o *switch* L2/L3 CPqD Enterprise, baseado em um ASIC da Broadcom com 24 portas de 1 Gbit/s e 2 de 10 Gbit/s. Devido ao número de equipamentos disponíveis com suporte ao OpenFlow, por se tratar de um ambiente real e não simulado, as redes testadas apresentam um número limitado de nós. Entretanto, a quantidade de nós é suficiente para uma avaliação de viabilidade da proposta, através da análise detalhada das trocas de mensagens e dos tempos relacionados à convergência da rede em caso de falha, como, por exemplo, o tempo de processamento das mensagens RouteFlow e OpenFlow. Essas mensagens não existem em uma arquitetura clássica de roteador com pilha de protocolos embarcada.

3.1 Tempo de convergência com tráfego *line rate*

Para os testes de convergência, foi utilizado um gerador e um analisador de tráfego configurados para transmissão de pacotes IP de 1500 bytes em ambos os sentidos (*full duplex),* a uma taxa de 1 Gbit/s. Dessa forma, garante-se que essa solução de roteamento remota sobre uma rede programável é capaz de operar com tráfego na taxa de transmissão da interface (*line rate*), uma vez que os pacotes são processados em hardware (ARGYRAKI et al., 2008).

A Figura 10 apresenta um gráfico que mostra o tempo total de convergência da rede após falha de um enlace, utilizando o protocolo OSPFv3 configurado com o parâmetro *hello time* em 1 segundo. Esse é o principal parâmetro do protocolo diretamente relacionado ao tempo de detecção de falhas. O *hello time* foi configurado para 1, 5 e 10 segundos, correspondendo aos valores típicos da indústria, que apresentam *default* de 10 segundos em enlaces Ethernet. O *dead interval* utilizado foi o recomendado correspondendo a quatro vezes o *hello time*.

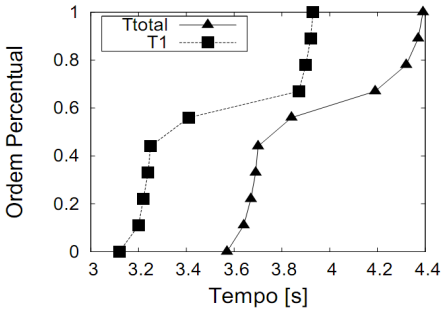


Figura 10 Tempo total de convergência (OSPF *hello interval* = 1 s)

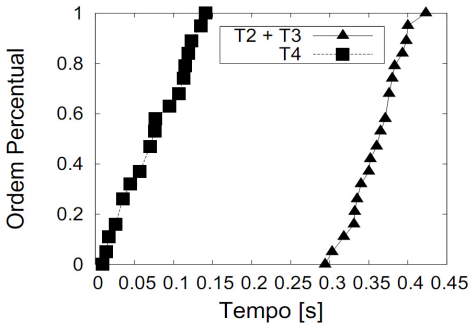


Figura 11 Fragmentação do tempo de convergência (OSPF *hello interval* = 1 s)

Com o intuito de estudar os componentes que contribuem com o tempo total de convergência, foram realizadas análises das mensagens enviadas e recebidas pelo hardware OpenFlow para o controlador NOX. Nos resultados, está incluso o tempo gasto pelas mensagens para transitar entre o dispositivo de encaminhamento, o controlador e a MV. T1 pode ser obtido a partir do intervalo entre o instante em que ocorre a interrupção do tráfego e a primeira mensagem de LS-Update gerada pelo RouteFlow ao detectar a falha. Em seguida, leva-se T2 + T3 para que o RF-Slave inicie o envio da primeira mensagem de alteração de fluxo e, por último, T4 finaliza o processo com o restabelecimento do tráfego.

Tabela 1 Tempos de convergência após falha

Hello Time	T_1 [s]		$T_2 + T_3$ [s]		T_4 [s]		T_{total} [s]	
OSPF	$T_{med.}$	$T_{90\%}$	$T_{med.}$	$T_{90\%}$	$T_{med.}$	$T_{90\%}$	$T_{med.}$	$T_{90\%}$
1 s	3,25	3,92	0,36	0,40	0,07	0,12	3,70	4,37
5 s	16,7	18,9	0,32	0,39	0,06	0,10	17,1	19,3
10 s	36,4	37,8	0,36	0,49	0,04	0,11	36,8	38,3

Na Figura 11 e na Tabela 1, apresentam-se os tempos conforme a fragmentação proposta anteriormente, em que cada um deles possui o valor de tempo abaixo do qual se encontram metade dos testes e o valor de tempo abaixo do qual se encontram 90% dos testes, num total de 20 repetições para cada uma das configurações.
Conforme os dados apresentados, o tempo total de convergência é bem próximo de T1 (tempo de detecção da falha). Portanto, o tempo restante, gasto com o cálculo de rotas, detecção de modificações da tabela de roteamento pelo RF-Slave e instalação do fluxo tem pouco impacto no tempo de convergência. Apesar de T2 e T3 não terem sido analisados separadamente, sabe-se que o tempo de *polling* para verificar atualizações da tabela de roteamento e ARP do Linux é fixo em 100 ms, ou seja, da soma T2 + T3, T3 representa até 100 milissegundos, no pior caso. Vale ressaltar que foi utilizada a estratégia de *polling* por simplificação; no entanto, podem ser feitas otimizações para reduzir substancialmente T3. Uma opção seria fazer com que a aplicação se registrasse no Zebra (módulo central do Quagga) para receber notificações sobre a RIB. Dessa forma, a informação chegaria ao RF-Slave de forma muito mais rápida e eficiente.
Dada a baixa representatividade dos tempos T3 e T4 sobre o tempo total, é razoável afirmar a viabilidade da solução RouteFlow no contexto proposto.

3.2 Encaminhamento em *slow path* e *fast path*

Outra forma de se avaliar o impacto de uma pilha de protocolos de roteamento remota é comparar os tempos para os modos de encaminhamento de pacotes: *slow path* e *fast path*. A primeira forma de encaminhamento ocorre quando o roteador precisa se comunicar com um nó de uma rede diretamente conectada a ele, mas antes do aprendizado do endereço MAC do destino, necessário para o encaminhamento em hardware. O *slow path* normalmente ocorre com os nós que entraram na rede ou que ficaram sem comunicação por um longo período. Por essa razão, essa operação nos roteadores, apesar de necessária, é considerada de baixa relevância.

O *fast path* refere-se ao encaminhamento em hardware que ocorre após o aprendizado dos endereços dos nós e o preenchimento das tabelas em hardware. Portanto, em uma transmissão de dados, o *slow path* ocorre, quando necessário, apenas para o primeiro pacote, a partir do qual será realizado o aprendizado, sendo que o restante dos pacotes serão processados em *line rate*.
Através desse teste, é possível avaliar a diferença de desempenho entre um encaminhamento por software e por hardware, de um roteador tradicional com protocolos embarcados, e do RouteFlow, com o plano de controle remoto. Foram realizados testes de PING (Internet Control Message Protocol – ICMP) entre dois *hosts* diretamente conectados a portas distintas, de um mesmo roteador, e configuradas em redes diferentes. No início, os *hosts* não têm conhecimento do endereço MAC dos *gateways,* e o roteador também não contempla ambos os *hosts* em suas tabelas. Após o aprendizado dos endereços, são obtidos os valores de *fast path*. Foram utilizados os seguintes *switches layer 3*: CISCO 3560-e Catalyst, Extreme x450-e, CPqD Enterprise (protótipo) e RouteFlow. Os resultados dos testes estão apresentados na Tabela 2.
Foi observado nos testes com o RouteFlow que além do primeiro pacote ICMP *request,* o *reply* também é encaminhado no *slow path* em software. O motivo desse comportamento está relacionado principalmente aos 100 ms de *polling* para que as atualizações nas tabelas do Linux sejam detectadas. No momento da resposta ICMP, os fluxos ainda não estão instalados e o pacote precisa novamente ser direcionado ao controlador para ser, então, encaminhado por software. Portanto, é válido pensar em um tempo consideravelmente menor para esse teste, com uma otimização da forma com que o RF-Slave passa a ter conhecimento das atualizações, como exemplificado anteriormente.
Outro ponto a ser destacado sobre o resultado de *slow path*, consideravelmente superior quando comparado aos dispositivos com software embarcado, é o desempenho do controlador OpenFlow (NOX) utilizado nos testes, cuja proposta é a de prover uma plataforma simples de desenvolvimento de aplicações de rede sem o compromisso de manter o foco em desempenho. Nesse sentido, já foram identificadas possíveis otimizações que devem trazer ganhos significativos no tempo de processamento das mensagens no controlador, como, por exemplo, tornar o controlador multitarefa, de forma a realizar o processamento paralelo das mensagens e, também, a implementação de mensagens que agreguem mais informação, resultando em um menor chaveamento de contexto da aplicação para processar cada

pacote recebido.
Em contrapartida, pode-se observar o melhor desempenho de *fast path* do RouteFlow, que é a forma de encaminhamento mais relevante. Nesse modo, a tabela de fluxos é encaminhada mais rapidamente, se comparado ao modo *longest prefix match*, que é utilizado nas tabelas de encaminhamento IP.

Tabela 2 Tempo de resposta ICMP

Equipamento	*Slow path* [ms]		*Fast path* [ms]	
	$T_{med.}$	$T_{90\%}$	$T_{med.}$	$T_{90\%}$
CISCO 3560-e C.	5,46	7,75	0,100	0,130
Extreme x450-e	11,30	14,00	0,106	0,141
CPqD Enterprise	14,20	17,30	0,101	0,147
RouteFlow	116,00	138,00	0,082	0,119

3.3 Discussão dos resultados e trabalhos futuros

Os resultados alcançados na avaliação do protótipo sugerem o grande potencial do RouteFlow como solução de roteamento para redes de campus/corporativas, mediante o ganho de flexibilidade e o poder de inovação proporcionados. As questões pertinentes ao desempenho não inviabilizam a proposta, uma vez que já foram identificadas otimizações e outras melhorias decorrentes da maturidade do protocolo OpenFlow.
Como em qualquer abordagem baseada em controladores centralizados, tornam-se desafios os aspectos de escalabilidade, resiliência a falhas e segurança. Vale a pena notar que a centralização do modelo OpenFlow é somente lógica. Não existe nenhuma restrição a uma implementação distribuída dos elementos controladores para atender aos requisitos de escala, desempenho e confiabilidade da rede. Esforços atuais no aprimoramento da arquitetura nesses aspectos incluem a distribuição do plano de controle virtual em múltiplos servidores, a adoção de técnicas avançadas de virtualização para o balanceamento e migração das MVs, e outras melhores práticas da programação de sistemas em nuvem (*cloud computing*) para tornar a plataforma escalável e tolerante a falhas (ex.: BD distribuída do tipo NoSQL, replicação do estado em controladores *backup)* (KOPONEN et al., 2010).
Outros aspectos da proposta, que merecem considerações adicionais e são objeto de estudo, incluem o isolamento entre as MVs (FERNANDES et al., 2010) e a materialização do conceito de roteamento como serviço (KELLER; REXFORD, 2010). Dessa forma, pode-se obter topologias lógicas independentes sobre uma mesma rede, que executem protocolos distintos, resultando em uma abordagem de virtualização com um melhor aproveitamento dos recursos da infraestrutura, que, agora, pode ser compartilhada com diferentes propósitos (CASADO et al., 2010).
Outra questão pertinente é o gargalo observado nas implementações disponíveis do OpenFlow quanto ao tamanho da tabela de fluxos (entre 2 e 4 mil) e à capacidade de instalação de novas entradas por segundo (valor em torno de 100 fluxos por segundo). Essas limitações são transitórias e serão contornadas com o avanço das especificações. Por exemplo, a partir da versão 1.1 do protocolo OpenFlow é possivel expor e controlar as tabelas L2/L3 do hardware. Outras técnicas para reduzir o consumo das entradas em hardware incluem algoritmos que escolhem aquelas entradas mais utilizadas (SARRAR et al., 2010). Essas e outras questões encontram-se atualizadas na agenda de P&D do projeto (ROUTEFLOW, 2011).

Conclusão

O conceito de redes definidas por software – decorrente da disponibilidade de um protocolo aberto, como o OpenFlow – promete um modelo disruptivo de inovação tecnológica de potencial impacto nos cenários de convergência ampla (consolidação de planos de controle de redes heterogêneas) e de computação em nuvem (integração da rede com recursos de TI). Equipamentos de rede programáveis poderão servir de base para o início de um ciclo de inovação aberta e rápida, envolvendo pesquisadores, engenheiros e desenvolvedores de software da academia, dos institutos de ciência e tecnologia, dos fabricantes de equipamentos, das operadoras de telecomunicações e dos provedores de serviços de Internet. Essas inovações se darão tanto no âmbito dos controladores como das aplicações que são executadas nesses controladores. Por aplicações entenda-se algoritmos, protocolos, serviços e aplicações de controle e gerência de redes e serviços. Além disso, o movimento de inovação aberta, decorrente do modelo OpenFlow, poderá alavancar e induzir um grande desenvolvimento na área de redes e tecnologias da informação e computação de uma forma considerável, como ocorreu com os computadores pessoais com o advento dos sistemas operacionais DOS, do Windows, do Linux e da consolidação destes por meio de técnicas de virtualização que dominam, atualmente, o mundo das tecnologias da informação.
Em resumo, os principais benefícios e impactos do advento da tecnologia OpenFlow incluem:

1. Capacidade de inovação (possivelmente aberta) em soluções de redes e serviços para os proprietários de infraestrutura, os provedores de serviços e a comunidade

de pesquisa.
2. Oportunidade para que novas empresas possam competir e inovar na área de aplicações avançadas para gerenciamento e controle de redes de pacotes.
3. Novos modelos de negócio que promovem redução de CAPEX e OPEX por meio de novos serviços – através da alocação dinâmica de fatias/recursos da rede, por exemplo – e de reaproveitamento de ativos – virtualização/consolidação.
4. Redução do custo de manutenção dos equipamentos, através, por exemplo, da incorporação de novos protocolos, da atualização do hardware do equipamento, etc.
5. Redução do tempo de implementação de novas funcionalidades nos equipamentos e de integração de soluções de redes especializadas às demandas do cliente final.
6. Simplificação e barateamento dos equipamentos de rede pela diminuição dos requisitos mínimos do software embarcado e das pilhas de protocolos proprietárias.
7. Consolidação dos planos de controle e da gerência de infraestruturas de rede, facilitando a migração para novos protocolos-padrão e tecnologias de rede de transporte.

Como exemplo de inovação sobre redes programáveis com OpenFlow, foi apresentada a proposta do RouteFlow, que representa uma abordagem expansível (*scale-out)* para arquiteturas de redes de pacotes. Com o plano de controle externo ao dispositivo de rede, passa a existir maior independência entre esse plano e o de encaminhamento, de forma que ambos escalem e evoluam separadamente. Nesse sentido, o RouteFlow possibilita o surgimento de redes mais baratas e flexíveis, mantendo compatibilidade com redes legadas e apoiando uma evolução das redes em que a conectividade IP possa se tornar um produto (*commodity*) ofertado em um modelo de plataforma como serviço (KELLER; REXFORD, 2010) e a diferenciação dos serviços dos operadores possa se dar pelas potenciais inovações do paradigma de redes definidas por software.

Agradecimentos

Os autores agradecem o apoio dado a este trabalho, desenvolvido no âmbito do projeto "Arquitetura de Redes para Comunicações Móveis IP" que contou com recursos do Fundo para o Desenvolvimento Tecnológico das Telecomunicações – FUNTTEL, do Ministério das Comunicações, através do Convênio nº 002/2007 com o Ministério das Comunicações.

Referências

ANWER, M. B. et al. Switchblade: a platform for rapid deployment of network protocols on programmable hardware. In: SIGCOMM '10. **Proceedings...** New York, USA: ACM, 2010. p.183-194.

ARGYRAKI, K. et al. Can software routers scale? In: PRESTO '08. **Proceedings...** New York, USA: ACM, 2008. p. 21-26.

CASADO, M. et al. Ethane: Taking Control of the Enterprise. In: SIGCOMM '07. **Proceedings...** New York, USA: ACM, 2007.

_____. Virtualizing the network forwarding plane. In: PRESTO '10. **Proceedings...** Philadelphia, USA, 2010.

CHOWDHURY, M.; BOUTABA, R. Network Virtualization: State of the Art and Research Challenges. **IEEE Communications Magazine**, v. 47, n. 7, p. 20-26, 2009.

FERNANDES, N. et al. Virtual networks: isolation, performance, and trends. **Annals of Telecommunications**, p. 1-17. 2010.

GREENE, K. TR10: Software-Defined Networking. **MIT Technology Review**. 2009. Disponível em: <http://www.technologyreview.com/communications/22120/>. Acesso em: 31 jan. 2011.

GUDE, N. et al. NOX: towards an operating system for networks. **SIGCOMM Computer Communication Review**, 38(3), p.105-110, 2008.

GUDLA, V. et al. Experimental Demonstration of OpenFlow Control of Packet and Circuit Switches. In: OPTICAL FIBER CONFERENCE (OFC/NFOEC'10). **Proceedings...** San Diego, USA, 2010.

HAMILTON, J. **Networking: The last bastion of mainframe computing.** 2009. Disponível em: <http://perspectives.mvdirona.com/2009/12/19/NetworkingTheLastBastionOf-MainframeComputing.aspx>. Acesso em: 31 jan. 2011.

INTERNET ENGINEERING TASK FORCE (IETF). **Forwarding and Control Element Separation (ForCES)**, WG, 2011. Disponível em: <http://datatracker.ietf.org/wg/forces/charter/>. Acesso em: 31 jan. 2011.

JUNIPER NETWORKS. **Network Operating System Evolution**. White paper, Oct. 2010.

KELLER, E.; REXFORD, J. The 'Platform as a Service' model for networking. In: INM/WREN. **Proceedings...** San Jose, USA, 2010.

KOPONEN, T. et al. Onix: A distributed control platform for large-scale production networks. In: OSDI'10. **Proceedings...** Vancouver, Canada,

Oct. 2010.

MCKEOWN, N. et al. OpenFlow: enabling innovation in campus networks. **SIGCOMM Computer Communication Review**, 38, p.69-74, 2008.

NASCIMENTO, M. R. et al. RouteFlow: Roteamento Commodity Sobre Redes Programáveis. In: XXIX SIMPÓSIO BRASILEIRO DE REDES DE COMPUTADORES - SBRC'2011, Campo Grande, MS, Brasil, 2011. **Proceedings...**

NOX. **An open-source OpenFlow controller.** 2011. Disponível em: <http://noxrepo.org>. Acesso em: 31 jan. 2011.

SARRAR, N. et al. FIBIUM – towards hardware accelerated software routers. In: EUROVIEW 2010 (poster session).

SHERWOOD, R. et al. Can the production network be the testbed? In: OSDI'10. **Proceedings...** USENIX Association, Vancouver, Canada, 2010.

The OPENFLOW Specification. 2010. Disponível em: <http://www.openflowswitch.org/documents/openflow-wp-latest.pdf>. Acesso em: 31 jan. 2011.

The QUAGGA Project. 2009. Disponível em: <http://www.quagga.net>. Acesso em: 31 jan. 2011.

The XORP Project. **Extensible open router platform**. 2011. Disponível em: <http://www.xorp.org>. Acesso em: 31 jan. 2011.

The ROUTEFLOW Project. 2011. Disponível em: <https://sites.google.com/site/routeflow/>. Acesso em: 31 mai. 2011.

YAP, K. K. et al. Blueprint for Introducing Innovation into Wireless Mobile Networks. In: VISA'10, New Delhi, India, 2010. **Proceedings...** New York: ACM, 2010.

Abstract

For some time we have seen the need for greater openness and flexibility of networking equipment, not only for research purposes, but also in search of in-house innovation. Today's networking gear follows the model of computer mainframes, with closed software running on proprietary hardware. This approach results in expensive solutions and prevents equipment owners to put new ideas into practice. Advances in the standardization of OpenFlow as a hardware-independent, open protocol to control the networking gear of packet networks introduces the notion of software-defined networks, a new paradigm for innovation with a disruptive potential comparable to the emergence of operating systems in the computer and mobile device industries. This paper introduces the principles of OpenFlow and presents the RouteFlow as an example of innovative ongoing work on open-source routing services over commodity hardware.

Key words: *Packet networks. Routing. Switching. Free Software. Virtualization.*