ISBN 857631411-8

617

CONSEGI 2012 V Congresso Internacional Software Livre e Governo Eletrônico

Inovando Serviços com Mobilidade Digital

FUNDAÇÃO ALEXANDRE DE GUSMÃO

Realização:

SERPRO Serviço Federal de Processamento de Dados

Ministério da Fazenda

Apoio:

SERPROS

CESUPA

SUCESU PARÁ

PRODEPA

Secretaria de Estado de Ciência, Tecnologia e Inovação

GOVERNO DO PARÁ

FUNDAÇÃO ALEXANDRE DE GUSMÃO

Bronze:

ITAIPU BINACIONAL

PREVIDÊNCIA SOCIAL GATAPREV

Prata:

CAIXA

Diamante:

SEBRAE

BNDES

GOVERNO FEDERAL BRASIL PAÍS RICO É PAÍS SEM POBREZA

consegi.gov.br

CONSEGI 2012

V Congresso Internacional
Software Livre e Governo Eletrônico

Inovando Serviços com Mobilidade Digital

MOBILIDADE DIGITAL

De 3 a 7 de dezembro
Belém/PA

consegi.gov.br

CONSEGI 2012

V Congresso Internacional
Software Livre e Governo Eletrônico

Inovando Serviços com Mobilidade Digital

MOBILIDADE DIGITAL

De 3 a 7 de dezembro
Belém/PA

CONSEGI 2012
V Congresso Internacional
Software Livre e Governo Eletrônico

Inovando Serviços com Mobilidade Digital

Ministério das Relações Exteriores

Ministro de Estado Embaixador Antonio de Aguiar Patriota
Secretário-Geral Embaixador Ruy Nunes Pinto Nogueira

Fundação Alexandre de Gusmão

Presidente Embaixador José Vicente de Sá Pimentel

Instituto de Pesquisa de Relações Internacionais

Centro de História e Documentação Diplomática

Diretor Embaixador Maurício E. Cortes Costa

A *Fundação Alexandre de Gusmão*, instituída em 1971, é uma fundação pública vinculada ao Ministério das Relações Exteriores e tem a finalidade de levar à sociedade civil informações sobre a realidade internacional e sobre aspectos da pauta diplomática brasileira. Sua missão é promover a sensibilização da opinião pública nacional para os temas de relações internacionais e para a política externa brasileira.

Ministério das Relações Exteriores
Esplanada dos Ministérios, Bloco H
Anexo II, Térreo, Sala 1
70170-900 Brasília, DF
Telefones: (61) 2030-6033/6034
Fax: (61) 2030-9125
Site: www.funag.gov.br

CONSEGI 2012
V Congresso Internacional
Software Livre e Governo Eletrônico

Inovando Serviços com Mobilidade Digital

Brasília, 2012

Ministério das Relações Exteriores
Esplanada dos Ministérios, Bloco H
Anexo II, Térreo
70170-900 Brasília – DF
Telefones: (61) 2030-6033/6034
Fax: (61) 2030-9125
Site: www.funag.gov.br
E-mail: funag@itamaraty.gov.br

Equipe Técnica:
Eliane Miranda Paiva
Fernanda Antunes Siqueira
Gabriela Del Rio de Rezende
Jessé Nóbrega Cardoso
Rafael Ramos da Luz
Wellington Solon de Souza Lima de Araújo

Programação Visual e Diagramação:
Gráfica e Editora Ideal

Impresso no Brasil 2012

C749

Congresso internacional software livre e governo eletrônico (5: 2012: Belém, Brasil). Congresso internacional software livre e governo eletrônico: inovando serviços com mobilidade digital: 3 a 7 de dezembro de 2012, Belém, Brasil. – Brasília-DF : FUNAG, 2012.

149 p.; 23 cm.

Apresentação de Marcos Vinicius Ferreira Mazoni. Textos de Benedicto Fonseca Filho, Alisson Wilker Andrade, Ronaldo Agra, Viviane Malheiros, Djamel F. H. Sadok, Fernando N. N. Farias, João J. Salvatti, Antônio J. G. Abelém, Flávio Gomes da Silva Lisboa.

Inclui bibliografia.

ISBN: 978-85-7631-411-0

1. Software livre. 2. Governo. .I. Serviço Federal de Processamento de Dados – SERPRO. II. Fundação Alexandre de Gusmão - FUNAG. III. CONSEGI.

CDU: 004.4::35

Ficha catalográfica elaborada pela bibliotecária Talita Daemon James – CRB-7/6078

Depósito Legal na Fundação Biblioteca Nacional conforme Lei n° 10.994, de 14/12/2004.

Apresentação

É com grande satisfação que apresento este livro cujas páginas reúnem um compêndio de textos relativos a tecnologias emergentes no mundo das Tecnologias da Informação e Comunicação (TICs). A descrição destas tecnologias são reflexões acerca de projetos e ações em curso nacional e internacionalmente. Elas compõem temas que serão amplamente discutidos durante o V Congresso Internacional de Software Livre e Governo Eletrônico – CONSEGI.

O CONSEGI 2012 terá vários eixos temáticos em torno do tema software livre, com destaque para o tema mobilidade digital em sistemas de governo. Inúmeros governos do mundo inteiro incrementaram, em 2012, a disponibilização de serviços de governo em plataformas móveis, principalmente em plataformas baseadas em sistema operacional Linux. A disponibilização de serviços cresce na mesma proporção que cresce o acesso à Internet móvel, sendo que o Brasil é o líder em crescimento de banda larga móvel na América Latina. Além disto, o Brasil é hoje o quarto maior mercado de mobilidade no mundo, com mais de 260 milhões de conexões móveis ativas, ou seja, a mobilidade é o maior vetor para disponibilização de serviços de governo eletrônico em banda larga no país.

A utilização de plataformas móveis tem a enorme capacidade de redefinir o tema governo eletrônico por meio de uma nova arquitetura, que pode ser resumida, no nível mais geral, a partir de três objetivos. São estes:

- mG2G, Governo móvel para Governo – resolutividade da máquina pública a partir de uma lista de serviços disponíveis em lojas virtuais;

- mG2E, Governo móvel para Empresas – redução do custo das obrigações por meio de aplicativos de governo;

- mG2C, Governo móvel para Cidadãos – provimento de direitos e cidadania a partir de acesso móvel pervasivo.

A inovação trazida pela mobilidade digital também está na capacidade de se criar novas interfaces humano-computador para serviços de governo já existentes em plataformas fixas, ou seja, uma utilização prática de reuso a partir de uma arquitetura orientada a serviços. Novas inovações em hardwares para dispositivos móveis são criadas a todo o instante e hoje vemos uma profusão de fabricantes e modelos que vão desde TV conectada a *smartphones*, passando por *ultrabooks* e *tablets*, criando uma diversidade de oferta que beneficia em muito a população.

Finalmente, espero que este material auxilie na difusão do conhecimento de temas tecnológicos emergentes e que seja um convite à participação colaborativa para o crescimento da comunidade de software livre.

Boa leitura!

Marcos Vinicius Ferreira Mazoni
Presidente do SERPRO

Sumário

Introdução

Benedicto Fonseca Filho

As Tecnologias da Informação e das Comunicações (TICs) vêm assumindo um papel cada vez mais central na sociedade, à medida que transformam o modo como os indivíduos se comunicam e interagem e se tornam ferramentas básicas, essenciais para a ampliação da produtividade, em praticamente todos os setores da economia. Nesse contexto, a intensificação das discussões sobre temas relacionados às TICs, tanto no plano interno quanto no plano internacional, é um desdobramento natural, dada a importância estratégica e o crescente interesse por essa área.

No plano interno, destacam-se os esforços para universalização do acesso à Internet; as iniciativas para modernizar a administração pública, conferindo maior transparência e abertura na relação com os cidadãos, por meio da Internet; bem como a utilização dessas tecnologias como ferramenta para modernizar e fortalecer a economia.

No plano internacional, a necessidade de coordenação de esforços para enfrentar os desafios dessa nova realidade é enfatizada na Agenda de Túnis para a Sociedade da Informação (documento final da Cúpula Mundial sobre Sociedade da Informação), a qual reconhece a "crescente importância do papel das TICs não apenas como um meio de comunicação, mas também como habilitadoras do desenvolvimento e ferramentas para o cumprimento de metas e objetivos acordados internacionalmente, dentre eles as Metas de Desenvolvimento do Milênio". Nesse contexto, a Agenda de Túnis reconhece que a superação do hiato digital requer investimentos adequados e sustentáveis na infraestrutura e nos serviços de TICs, bem como a construção de capacidades e a transferência de tecnologia.

O lançamento recente do Plano TI Maior pelo Ministério da Ciência, Tecnologia e Inovação evidencia a prioridade que o governo brasileiro confere a esses temas. O fomento às empresas "start-ups", a atração de centros globais de pesquisa e desenvolvimento no campo das TICs, com vistas à formação de uma rede local de desenvolvimento científico e tecnológico, bem como o estabelecimento de polos de negócios internacionais com o objetivo de contribuir para o desenvolvimento e a internacionalização de empresas brasileiras que atuam nessa área, são objetivos centrais do TI Maior.

Com enfoque diferente, porém igualmente relevante do ponto de vista do fortalecimento da sociedade da informação, o Plano Nacional de Banda Larga (PNBL), lançado em maio de 2010 pelo Ministério das Comunicações não apenas visa à massificação do acesso à Internet em banda larga no Brasil, por meio da ampliação da infraestrutura e da redução dos preços desse serviço, como também tem por objetivo promover a difusão dos serviços de governo eletrônico e a capacitação da população para o uso das tecnologias da informação.

Ambos os planos oferecem amplas possibilidades de cooperação na esfera internacional. No caso do PNBL, em particular, a ampliação da infraestrutura nacional coincide com esforço regional para integração das redes de fibra ótica entre os países da América do Sul. A formação do chamado "anel ótico sul-americano" permitirá a redução dos custos de interconexão internacional, o que, por sua vez, contribuirá para a oferta de serviços de banda larga a preços menores na região.

A concepção do Congresso Internacional Software Livre e Governo Eletrônico (CONSEGI) confirma, de um lado, a centralidade dessas questões, e evidencia, de outro, a necessidade de articular o debate nacional com a cooperação internacional. Nesta sua quinta edição, as discussões em torno do tema principal da mobilidade ensejarão o tratamento de questões que também vêm sendo amplamente discutidas no plano internacional, entre as quais mobilidade digital, segurança cibernética, informática e sociedade e governo eletrônico. A adoção de um país focal em cada edição do CONSEGI também contribui para o intercâmbio de experiências e para a exploração de convergências com outros países.

Com base nesse cenário e em linha com as estratégias e prioridades definidas internamente, a atuação internacional do Itamaraty busca, em coordenação com os órgãos competentes da administração pública, identificar áreas de interesse comum e explorar oportunidades de cooperação internacional em áreas como governança da Internet, inclusão digital, governo eletrônico e software livre.

No âmbito regional, cabe menção à Estratégia para a Sociedade da Informação na América Latina e no Caribe (eLAC), que em seu Plano de Ação mais recente estabelece metas e prioridades para a região em matéria de acesso, em especial a massificação da infraestrutura de acesso à Internet em banda larga; governo eletrônico, visto como direito dos cidadãos, devendo ser transacional e participativo; meio ambiente, em especial o uso das Tecnologias da Informação e Comunicação (TICs) para combater a mudança do clima e prevenir desastres naturais; desenvolvimento produtivo e inovação, incluindo fomento à pesquisa e ao desenvolvimento de TICs na região; e educação.

O Projeto Mercosul Digital, por sua vez, também contribui para a difusão das TICs entre os países que integram o bloco, a partir do objetivo de promover políticas e estratégias comuns na área da Sociedade da Informação, através da capacitação, da harmonização dos regulamentos, da implementação da infraestrutura técnica e do intercâmbio de conhecimentos. A criação da Escola Virtual do Mercosul, que oferece cursos de capacitação através de plataforma virtual, é um dos resultados concretos dessa iniciativa.

Valeria mencionar, ainda, outras iniciativas de cooperação bilateral que o Brasil vem desenvolvendo tanto com países desenvolvidos, como com países em desenvolvimento. Com a União Europeia mantemos Diálogo anual sobre Sociedade da Informação, com o objetivo de dar continuidade à cooperação e à troca de experiências em Pesquisa e Desenvolvimento na área de TICs, Regulação, Desenvolvimento da Banda Larga, Transmissão Digital, Governança da Internet, Conteúdos Digitais e Computação em Nuvem. Além de proveitoso intercâmbio em todas essas áreas, o lançamento de editais conjuntos para o desenvolvimento de projetos de cooperação em pesquisa e desenvolvimento na área de TICs tem permitido auferir resultados concretos da cooperação com o lado europeu. Afigura-se igualmente promissora a concertação bilateral no campo da governança da Internet, tema que, a partir deste ano, deverá figurar na agenda de discussões entre os dois parceiros.

Com os Estados Unidos, instituímos Grupo de Trabalho sobre Internet e TICs como o objetivo de promover intercâmbio de informações e explorar, onde cabível, oportunidades de cooperação, inclusive em temas relacionados à governança global da Internet.

O Canadá é outro país com o qual o Brasil tem buscado maior aproximação na área de TICs. O Plano de Ação conjunta em ciência e tecnologia adotado no início deste ano inclui, entre suas prioridades, o estabelecimento de cooperação bilateral em áreas como computação em nuvem e conteúdos digitais, bem como a organização, pelo Brasil, em

cooperação com o Canadá, da "Conferência Brasil 3.0", inspirada em evento anual semelhante organizado naquele país pelo qual se articulam os setores privado, acadêmico e governamental em busca de soluções digitais para demandas da sociedade. A primeira edição da "Conferência Brasil 3.0" será realizada em João Pessoa, nos dias 3 e 4 de dezembro próximo.

No caso do Uruguai, país focal do CONSEGI 2012, cabe ressaltar a criação, este ano, pela Presidenta Dilma Rousseff e pelo Presidente José Mujica, de um "Grupo de Alto Nível" encarregado de consolidar um Plano de Ação para o Desenvolvimento Sustentável e a Integração Brasil--Uruguai, o qual inclui, entre as áreas prioritárias, Ciência, Tecnologia e Inovação e Comunicação e Informação. No âmbito das TICs, os esforços estarão voltados para a implementação de plataforma de "e-learning", com vistas à formação de recursos humanos, à criação de Centro Binacional de Tecnologias da Informação e das Comunicações, bem como para a promoção da interconexão das redes acadêmicas avançadas de ambos os países.

Nesse contexto amplo, o Ministério das Relações Exteriores reafirma o apoio ao Congresso Internacional de Software Livre e Governo Eletrônico, no entendimento de que se encontra em sintonia com as prioridades nacionais nessa área e contribui para fomentar o debate sobre temas relevantes no campo das TICs, com ampla participação de todos os setores da sociedade brasileira, além de buscar fortalecimento da interação com parceiros-chave.

Mobilidade digital aplicada ao governo brasileiro

Alisson Wilker Andrade, Ronaldo Agra, Viviane Malheiros[1]

1. A mobilidade e sua influência no cotidiano dos brasileiros

O uso de dispositivos móveis possibilita o acesso a serviços e informações a qualquer momento, por meio de redes sem fio e de diversos recursos, como: texto, voz, vídeo, internet, GPS, câmera, música e televisão. Estes recursos integrados podem melhorar significativamente a prestação de serviços, e, em particular, de serviços de governo eletrônico. As possibilidades vão muito além da realização de chamadas telefônicas, e são aplicáveis a diversas áreas como: educação, bancária, entretenimento, turismo e saúde.

Explorar recursos de mobilidade digital é uma oportunidade para o governo oferecer serviços mais adequados ao cidadão brasileiro, além de ter serviços governo-governo mais eficientes e dinâmicos. Isto porque a disponibilidade de acesso à Internet e o uso de dispositivos móveis é cada vez maior no país. Em 2011, houve um crescimento de quase 100% no total de acessos à Internet móvel [1]. Em maio de 2012, a difusão de *smartphones* já atingia 14% da população, segundo relatório da Google sobre o mercado de mobilidade brasileiro [2].

O aumento do uso de *smartphones* no Brasil tem impactado diretamente o cotidiano dos brasileiros [2]: 42% dos usuários de *smartphones* acessam diariamente a Internet a partir de seus dispositivos móveis; 80% deles pesquisam um produto no dispositivo antes de comprar; e 31%

[1] {alisson-wilker.silva, jose-ronaldo.souza, viviane.malheiros}@serpro.gov.br.

fazem compras a partir do dispositivo [2]. Os *smartphones* são utilizados principalmente para: comunicar-se (redes sociais, e-mails), manter-se informado (jornais, blogs, revistas) e entreter-se (música, vídeo, jogos) [2].

A nova forma de executar atividades comuns do dia a dia influencia o relacionamento de empresas com seus clientes. Por exemplo, as empresas adaptam seus *sites* e propagandas para clientes, que agora chegam a eles através de um dispositivo móvel, tanto de dentro de um ônibus, como realizando exercícios físicos num parque da cidade. Da mesma forma, para assuntos relacionados ao governo, pode-se imaginar um cidadão consultando sua certidão eleitoral ou sua matrícula na universidade por meio de um dispositivo móvel.

Na perspectiva do Serpro, o desenvolvimento de aplicativos para dispositivos móveis é uma oportunidade de oferecer soluções diferenciadas para o governo e de manter-se como referência de tecnologia da informação, mostrando-se alinhado com as tendências de mercado e com as necessidades dos brasileiros.

Além de ser uma grande oportunidade, o uso abrangente de computação móvel para o governo é, também, um grande desafio. É fundamental ter infraestrutura e dominar o ambiente para explorar adequadamente questões relacionadas a particularidades dos dispositivos móveis, como: (i) limitações de memória, bateria, processamento, tamanho de tela e teclado, capacidade de armazenamento, largura de banda e consumo de energia; (ii) grande diversidade de aparelhos e plataformas de desenvolvimento (proprietárias e livres); (iii) segurança; e (iv) forma de publicação de aplicativos.

Alguns órgãos e empresas do governo já possuem aplicativos e conteúdo web adaptado para dispositivos móveis (especialmente, *tablets* e *smartphones*) [3] [4] [5] [6]. Em 2012, o Serpro desenvolveu e publicou aplicações móveis para a Receita Federal do Brasil [7] [8] [9] [10] e para o DNIT (Departamento Nacional de Infraestrutura de Transportes) [11] [12]. O desenvolvimento destas soluções suscitou o estudo das características do desenvolvimento para dispositivos móveis e discussões arquiteturais abrangentes. A experiência obtida com estes projetos está sendo utilizada na definição de ações e modelos corporativos para habilitar o Serpro a oferecer soluções inovadoras para o governo.

Neste contexto, este artigo apresenta: (i) características do desenvolvimento de aplicativos para dispositivos móveis; (ii) possíveis arquiteturas de aplicações para dispositivos móveis e como escolher a mais adequada a partir de seus requisitos; e (iii) dois estudos de caso de mobilidade para o governo desenvolvidos pelo Serpro.

2. Características do desenvolvimento para dispositivos móveis

Figura 1: *Novas possibilidades com o uso de dispositivos móveis.*

O desenvolvimento para dispositivos móveis deve levar em consideração uma experiência de usuário diferente daquela observada em estações de trabalho.

Com os novos recursos (como leitura de código, reconhecimento facial, GPS e comunicação por aproximação), processos podem ser eliminados e melhorados; e a interação homem-máquina pode ser repensada (Figura 1). Ao mesmo tempo, a relação tempo e espaço com os serviços é diferente.

Fatores a serem considerados no desenvolvimento para dispositivos móveis (extraídos de [13])	
Baixa largura de banda	Capacidades de cada plataforma (hardware e software)
Tempo de bateria limitado	Memória RAM e de armazenamento limitadas
Entrada de dados via toques, gestos e voz, mas lenta para textos	Custo de conectividade quando taxada por volume de dados
Telas de tamanho reduzido e aplicações que ocupam toda a tela	Pouca atenção do usuário pelo uso em contextos de movimento, agitação ou luz solar direta
Usabilidade de cada plataforma	Perda temporária de conectividade por baixo sinal da rede

Quadro 1: *Especificidades do desenvolvimento para dispositivos móveis.*

3. Arquiteturas de desenvolvimento para dispositivos móveis

O desenvolvimento para dispositivos móveis engloba dois cenários: (i) adaptação de sítios e portais para utilizar por meio de navegadores, a partir de qualquer *tablet* ou *smartphone;* e (ii) desenvolvimento de aplicações nativas adequadas para cada plataforma / tipo de dispositivo.

Cada um desses cenários requer uma arquitetura de solução específica, que valorize suas características principais. Ao mesmo tempo, é possível sugerir arquiteturas padrões, que facilitem o reuso de código e de infraestrutura. Assim, esta seção apresenta três arquiteturas: (i) uma arquitetura para desenvolvimento de conteúdo web móvel, (ii) outra para desenvolvimento de conteúdo nativo móvel, (iii) e uma terceira arquitetura que é chamada híbrida, pois engloba tanto o desenvolvimento nativo como o desenvolvimento web móvel, buscando valorizar o que há de melhor em cada cenário.

3.1. Arquitetura para desenvolvimento de conteúdo web móvel

O desenvolvimento de aplicações para dispositivos móveis engloba, dentre outros, a construção de conteúdo web adaptado para a interação via estes dispositivos (Figura 2). Nesse caso, o usuário acessa um sítio ou aplicação web através do navegador web de sua preferência, instalado em seu *smartphone* ou *tablet*. Através de uma conexão com a internet ou a intranet (2G, 3G, 4G, *Wi-Fi* ou *Bluetooth*), o navegador realiza uma requisição ao servidor web que contém o conteúdo web da aplicação ou sítio. Este, por sua vez, ao receber a requisição, identifica qual o dispositivo está solicitando o conteúdo web e realiza o redirecionamento para a página correta, seja um dispositivo móvel ou um computador pessoal.

Dessa forma, um único servidor web pode ser utilizado tanto para *desktops* como para dispositivos móveis. Por exemplo, o navegador *desktop* pode acessar o sítio <http://www.serpro.gov.br> enquanto o navegador móvel acessa <http://m.serpro.gov.br>, mesmo que ambos tenham feito uma requisição por <http://www.serpro.gov.br>. Nesse caso, a versão do conteúdo ou da aplicação web acessada pelo dispositivo móvel deve possuir HTML[2], CSS[3], JavaScript[4] e imagens adaptados para as características desse tipo de dispositivo (p. ex.: tamanho de tela, resolução e acesso à rede) e ao tipo de interação que o

[2] <http://www.w3.org/TR/html/>.
[3] <http://www.w3.org/Style/CSS/>.
[4] <http://www.w3.org/standards/techs/js>.

usuário tem com este (como toque na tela, uso do dispositivo em movimento e entrada de textos limitada).

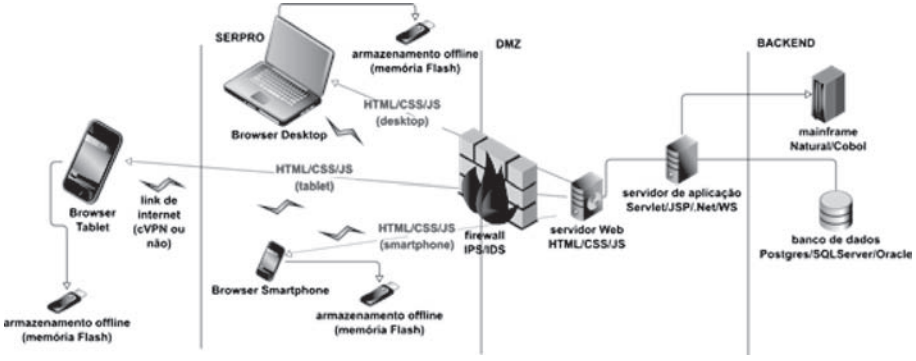


Figura 2: *Visão geral de arquitetura para conteúdo web móvel.*

Além do servidor web, que disponibiliza recursos estáticos do conteúdo ou aplicação web, um servidor de aplicação pode ser utilizado para gerar recursos dinâmicos ou processar as regras de negócio de uma ou mais aplicações. Para o navegador do dispositivo, essa comunicação entre servidores (web e de aplicação) é transparente, ficando assim sua responsabilidade restrita à renderização do conteúdo recebido.

Outra maneira de instanciar esta arquitetura é possuir apenas um conteúdo ou aplicação web e usar *layouts* diferentes para cada tipo de dispositivo, o que é chamado de design responsivo (*responsive design)*. Design Responsivo é uma técnica de estruturação HTML e CSS, na qual a página adequa-se ao navegador do usuário sem a necessidade de diversas folhas de estilos para cada resolução. Nesse caso, o CSS pode utilizar o recurso de "*media query*" para identificar o tamanho de tela do navegador e adaptar o *layout* do conteúdo ou aplicação web para um tamanho específico. A vantagem de utilizar essa abordagem é o menor custo de manutenção, se comparada à alternativa de manter uma versão móvel e outra para *desktop*. Porém, essa abordagem pode penalizar o desempenho e a navegabilidade da aplicação, visto que as imagens, por exemplo, não são otimizadas para a capacidade do dispositivo [14] [15] [16]. Uma alternativa de solução para adaptar as imagens do conteúdo web é utilizar imagens vetoriais, como as de formato SVG, por exemplo, as quais podem ser facilmente redimensionadas sem perder a qualidade [17]. A melhor escolha entre essas alternativas deve ser guiada pelo tipo de conteúdo web, os objetivos de sua publicação e o perfil dos usuários esperados.

É importante ressaltar também que a aplicação ou o conteúdo web desenvolvido para *tablet* pode ser bastante diferente daquele desenvolvido com foco em *smartphones*. Isto porque as dimensões e a resolução de tela, bem como a forma de uso, são bem distintas entre esses dispositivos. O tamanho de tela e a resolução maior permitem, e muitas vezes obrigam, que o conteúdo servido para um *tablet* seja mais completo e refinado do que aquele oferecido para um *smartphone*, onde a restrição de espaço disponível na tela é muito maior. Assim, normalmente, o conteúdo servido para *smartphones* é bem mais conciso, principalmente se comparado ao conteúdo servido para um navegador *desktop* [18]. Neste caso, também pode ser utilizada a técnica de Design Responsivo para fazer a adaptação do conteúdo para *tablets*.

Duas grandes vantagens do desenvolvimento de aplicações e de conteúdo web em relação à aplicação nativa (a ser detalhada nas próximas seções) são: a facilidade e a abrangência da liberação de novas versões da aplicação. Uma vez que uma nova versão da aplicação é disponibilizada no servidor web, automaticamente todos os usuários estarão utilizando a nova versão, sem que qualquer ação adicional do usuário ou do fornecedor daquela aplicação seja necessária. Além disso, como a linguagem de desenvolvimento é geralmente padronizada em HTML, CSS e JavaScript, o mesmo código pode ser executado em qualquer dispositivo que possua navegador compatível com essas linguagens, não importando os seus sistemas operacionais. Do ponto de vista organizacional, esta arquitetura reduz a complexidade operacional, pois não exige o domínio de diversas linguagens de programação e tecnologias específicas.

3.1.1. Vantagens

- Familiaridade do usuário com o contêiner da aplicação ou conteúdo web, que é o seu navegador web de preferência;
- Utilização de um único endereço (URL) para acessar conteúdo adaptado para *tablets*, *smartphones* e *desktops*, de forma transparente;
- Facilidade de implantar novas versões da aplicação, atualizando-se apenas os servidores;
- Reaproveitamento de código da interface através do desenvolvimento com linguagens multiplataforma (HTML, CSS e JavaScript);
- Reaproveitamento de código da lógica de negócio em aplicações para *desktop*, *smartphones* e *tablets*;

- Maior segurança para regras de negócio sensíveis, pois elas ficam armazenadas no servidor e não no dispositivo;
- Não depende de aprovação de terceiros para disponibilizar a aplicação em uma "store" (loja);
- Desnecessário instalar conteúdo no dispositivo móvel do usuário.

3.1.2. Desvantagens

- Possibilidade de falha ou restrição de desempenho na aplicação devido ao navegador web escolhido pelo usuário;
- Necessidade de testes criteriosos em diversas versões de navegadores antes da liberação;
- Limitação de acesso aos recursos de hardware do dispositivo móvel (ex.: câmera, acelerômetro) através do navegador web;
- Impossibilidade de acessar o banco de dados SQLite nativo dos ambientes Android e iOS;
- Inexistência de um ícone de aplicação na tela do dispositivo, embora esse fato possa ser contornado com a criação manual de um atalho para o conteúdo web na tela principal do dispositivo;
- Dependência maior de conectividade com a internet.

3.1.3. Ferramentas de apoio

Existem alguns *frameworks* que facilitam a construção de interfaces adaptadas à navegação via telas sensíveis ao toque e de tamanho limitado para aplicações web [19] [20]. Esses *frameworks* web trabalham, normalmente, com tecnologias padronizadas na Internet, como o HTML5, o JavaScript e o CSS3.

Usando essas tecnologias e diretrizes do paradigma de *Web Design* Responsivo, é possível construir aplicações web que se adaptam às características de tela do *tablet* ou do *smartphone*, como tamanho, resolução e profundidade de cores.

Um exemplo desse tipo de tecnologia é o jQuery Mobile [19]. Este *framework* facilita a criação de aplicações web que funcionam nas plataformas móveis e nos navegadores mais populares, como Android, iOS, Windows Phone e BlackBerry. Trata-se de um projeto de software livre com uma comunidade ampla e ativa e com código-fonte hospedado

no GitHub[5]. Seu objetivo é garantir a uniformidade, através de um conjunto de imagens e folhas de estilos, entre a aplicação web e o estilo visual da plataforma móvel em que está sendo executada.

No contexto da construção de portais web, o paradigma de *Web Design* Responsivo e o *framework* jQuery Mobile podem ser usados em conjunto com um sistema de gerenciamento de conteúdo (CMS[6]), como o Zope/Plone [21] ou o Liferay [22]. Ambos os CMS's possuem suporte nativo à construção de portais que se adaptam visualmente para possibilitar a navegação através de dispositivos móveis. Dessa forma, com pouco esforço e algum conhecimento de CSS, é possível entregar portais web compatíveis com as principais plataformas e navegadores móveis.

3.2. Arquitetura para aplicação nativa móvel

Outra possibilidade de desenvolvimento para dispositivos móveis é a construção de aplicações nativas para cada plataforma móvel, como Android, iOS, Blackberry ou Windows Phone (Figura 3). Nesse caso, o desenvolvedor utiliza o ambiente de desenvolvimento e as bibliotecas fornecidas pelo fabricante da plataforma para desenvolver a aplicação. Assim, no caso do Android, o desenvolvedor deve se familiarizar com a sintaxe Java e as bibliotecas do Android, assim como o desenvolvedor para iOS deve se familiarizar com a linguagem Objective-C e suas bibliotecas.

A aplicação nativa normalmente tem um visual mais integrado com a plataforma do dispositivo, visto que os componentes visuais como botões, menus e listas, por exemplo, são fornecidos e estilizados pela própria plataforma. O usuário pode, então, sentir-se mais confortável ao utilizar a aplicação nativa, visto que já está familiarizado com as cores, as formas e os comportamentos desses componentes.

Além disso, nesta arquitetura de desenvolvimento, é possível utilizar diversos recursos de hardware do dispositivo, como acelerômetro, GPS, bússola e câmera, que normalmente não podem ser acessados diretamente a partir de uma aplicação web executada em um navegador de Internet. Essa possibilidade leva à criação de aplicações muito mais ricas em termos de funcionalidades e integração com o dispositivo móvel. Uma aplicação que utiliza bússola e GPS, por exemplo, pode conectar-se a um servidor remoto para acessar um banco de dados geográficos, a fim de representar a posição e a direção do usuário em tempo real. Se estes

[5] <https://github.com/jquery/jquery-mobile>.

[6] Sigla em inglês para *Content Management System*.

recursos forem usados em conjunto com a câmera, é possível identificar objetos e edifícios que estejam à vista do usuário, fornecendo mais detalhes a respeito destes.

Para o armazenamento de informações, uma aplicação nativa pode realizar o armazenamento local tanto em arquivos como em banco de dados SQLite. O SQLite é um banco de dados transacional leve e multiplataforma utilizado pelas principais plataformas móveis, como Android e iPhone [23]. Ele permite o armazenamento de *strings*, números e dados binários e pode ser consultado através da linguagem SQL diretamente da aplicação ou via linha de comando.

Normalmente, uma aplicação desenvolvida nativamente será submetida para validação e publicação em uma *store*[7] de aplicações a partir da qual o usuário poderá instalá-la em seu dispositivo móvel e, em seguida, acessá-la através de um ícone no menu de aplicações ou na tela principal do aparelho.

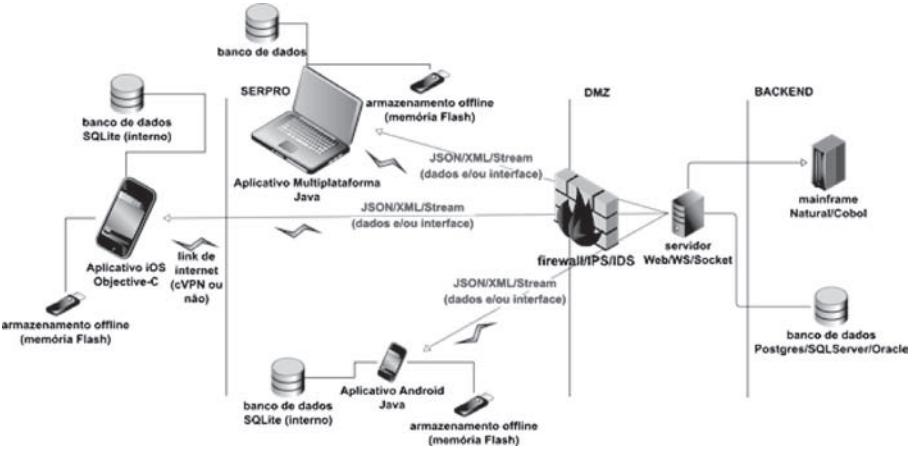


Figura 3: *Visão geral de arquitetura para aplicação nativa móvel.*

Uma desvantagem da *store* de aplicações é que toda versão da aplicação deve passar por um novo ciclo de validação e publicação. Este processo de validação que precede a publicação da aplicação pode levar horas (caso do Android) ou mesmo semanas (caso do iOS) e, portanto, não pode ser desprezado, uma vez que isto pode atrasar a entrega do aplicativo para o usuário.

Por outro lado, uma vantagem das lojas de aplicativos é que o fabricante da plataforma se responsabiliza por toda a infraestrutura para

[7] *Store*, ou loja de aplicativos, é um catálogo de aplicações que todas as plataformas utilizam para disponibilizar aos usuários os aplicativos existentes para a plataforma.

a publicação e distribuição das aplicações. Desta forma, o desenvolvedor pode focar seus esforços do desenvolvimento da aplicação em si, não tendo que se preocupar com a forma pela qual seu usuário irá adquirir o *software*. Além disso, também é possível monitorar os aplicativos com relação à quantidade de *downloads* efetuados em determinado período, assim como outras informações de usuários, como o índice de satisfação, por exemplo.

Por fim, como as linguagens e as orientações de desenvolvimento para cada plataforma móvel são diferentes, é importante ter uma equipe de desenvolvimento especializada em cada uma dessas plataformas. Uma alternativa para essa questão é usar alguma ferramenta para gerar código nas linguagens nativas de cada plataforma. Nesse caso, o desenvolvedor escreve o código da aplicação em uma linguagem padrão, como JavaScript, por exemplo, e depois a ferramenta transforma esse código em aplicações nativas das plataformas selecionadas pelo desenvolvedor [24]. Essa abordagem, porém, pode levar a outros problemas, como incompatibilidades, falhas de conversão da ferramenta, falta de suporte, entre outros.

3.2.1. Vantagens

- Integração da aplicação com o visual da plataforma em que ela é executada;
- Integração da aplicação com os recursos do dispositivo móvel, como acelerômetro, GPS, câmera e bússola, por exemplo;
- Familiaridade do usuário com os componentes visuais da aplicação, que são renderizados pela própria plataforma;
- Possibilidade de armazenamento local em arquivos ou em banco de dados SQLite;
- Monitoração das estatísticas de *download* e de satisfação dos usuários em relação a uma aplicação baixada da *store* de aplicações;
- Mais possibilidades de protocolos de comunicação com o servidor remoto, além daqueles suportados pelos navegadores web.

3.2.2. Desvantagens

- Demora para validar e publicar a aplicação em uma *store* de aplicações;
- A migração dos usuários para uma nova versão da aplicação não é controlada;

- Existência de um custo extra de publicação da aplicação na *store* de aplicações do fabricante;
- Especialização da equipe de desenvolvimento em uma ou mais plataformas móveis;
- Necessidade de disponibilizar o código-fonte da aplicação para o proprietário da *store* de aplicações durante o processo de validação e publicação.

3.2.3. Ferramentas de apoio

Para o desenvolvimento de aplicações nativas, o conjunto de tecnologias de apoio é diferente de acordo com a plataforma. No caso do iOS, a ferramenta principal de desenvolvimento é o software XCode [25] da Apple. Essa é a ferramenta oficial da Apple para desenvolvimento tanto de aplicativos para a plataforma OS X como para a plataforma móvel iOS, utilizada nos dispositivos iPhone, iPad e iPod. A ferramenta XCode é gratuita, mas, para publicação da aplicação na loja iTunes da Apple, é preciso ter uma conta de desenvolvedor Apple. A subscrição para ter a conta de desenvolvedor deve ser paga anualmente e depende do tipo da conta a ser adquirida (comum ou organizacional).

No caso do Android, a ferramenta principal utilizada para desenvolvimento é o Eclipse adicionado do *plugin* Android *Development Tools* (ADT) [26]. Esse pacote é utilizado em conjunto com a instalação do *Software Development Kit* (SDK) do Android [27]. Ambos os pacotes são gratuitos e, para publicação na loja Play Store da Google, é necessário o pagamento de uma taxa única, ou seja, não é necessária uma subscrição.

Para a plataforma Windows Phone, as ferramentas de desenvolvimento são baixadas através do pacote *Windows Phone* SDK [28]. Esse pacote de desenvolvimento é gratuito, mas a publicação do aplicativo também requer uma subscrição anual de desenvolvedor *Windows Phone*.

3.3. Arquitetura para aplicação nativa + conteúdo web móvel

A terceira abordagem para desenvolvimento de conteúdo para dispositivos móveis é a composição de conteúdo nativo com conteúdo web, também conhecida como aplicação híbrida. Nesse caso, uma parte da aplicação é escrita utilizando-se a linguagem padrão da plataforma (Java ou Objective-C, nos casos do Android e do iOS, respectivamente) e outra

parte é escrita utilizando-se linguagens de conteúdo web, como HTML, JavaScript e CSS, por exemplo.

Usualmente, as plataformas móveis disponibilizam objetos visuais nativos para renderizar e executar conteúdo web. Em linhas gerais, esses objetos funcionam como um navegador web, renderizando HTML + CSS e processando *scripts* escritos em JavaScript; permitindo assim embarcar conteúdo web em aplicações nativas. A aplicação construída com esses objetos tem todas as características de uma aplicação nativa, porém os componentes visuais com os quais o usuário interage são elementos da linguagem HTML, estilizados através de CSS, e cujo comportamento pode ser determinado via JavaScript.

Uma aplicação híbrida normalmente é construída quase completamente utilizando-se de linguagens padronizadas como HTML, CSS e JavaScript, e apenas uma pequena parte em linguagem nativa da plataforma. Dessa forma, boa parte do código de uma aplicação híbrida construída para Android, por exemplo, pode ser reaproveitada no desenvolvimento para outras plataformas. Nesse caso, apenas a pequena parte do código escrita em Java precisa ser alterada para a linguagem nativa da plataforma alvo.

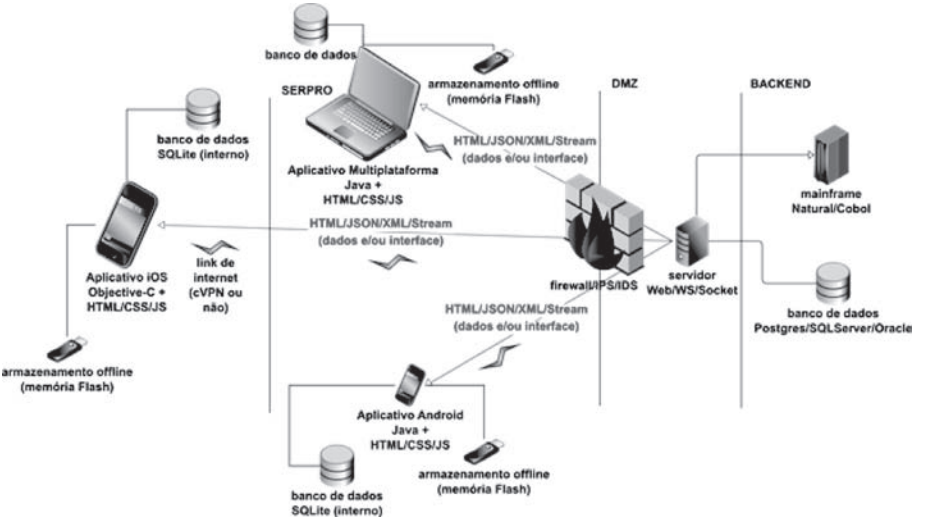


Figura 4: *Visão geral de arquitetura para aplicação nativa + conteúdo web móvel.*

Conforme Figura 4, a aplicação híbrida pode se comunicar com um servidor remoto através de um link de Internet (3G, por exemplo), ou com um servidor local através de uma rede *Wi-Fi* na intranet da organização.

Essa comunicação pode ocorrer via quaisquer protocolos de comunicação (p. ex.: HTTP, FTP, SMTP, WebDAV), seguros ou não, desde que compatíveis com as capacidades do aparelho e da rede de comunicação utilizada. As informações a serem trafegadas entre o dispositivo móvel e o servidor podem estar em diversos formatos de documento, como HTML, XML, JSON, ou mesmo em formato de *streaming* de dados.

As informações trafegadas entre aplicação móvel e servidor também podem ser dados ou descrições de componentes de interface a serem renderizados pela aplicação, assim como pode ser feito em aplicações nativas. Dessa forma, quando uma modificação afeta apenas a interface de usuário, essa alteração pode ser feita apenas no servidor remoto, evitando-se um novo ciclo de validação e publicação na *store* de aplicações.

A aplicação híbrida também permite tanto o armazenamento local em arquivos como o armazenamento em banco de dados SQLite. Normalmente, o acesso ao banco de dados SQLite é escrito na linguagem nativa da plataforma alvo. Porém, alguns *frameworks* de desenvolvimento de aplicações móveis híbridas oferecem API específica para que este acesso seja escrito em JavaScript [29]. Essas APIs específicas do *framework* devem ser usadas com cautela, visto que podem ser uma fonte de defeitos ou podem ser descontinuadas com o passar do tempo.

Aplicações que obedecem a essa arquitetura são instaladas no dispositivo móvel a partir de uma *store* de aplicativos. Portanto, também estão sujeitas ao ciclo de validação e publicação determinado por cada *store*. Além disso, assim como uma aplicação nativa, as aplicações híbridas são acessadas via um ícone da aplicação na tela do dispositivo e podem usar recursos do dispositivo, como acelerômetro, GPS, bússola e câmera.

A construção de aplicações híbridas possibilita não somente o reaproveitamento de código entre várias plataformas, como também pode reduzir a quantidade de desenvolvedores especializados em cada plataforma. Isto acontece, principalmente, quando o código nativo da aplicação híbrida se restringe apenas ao que não é conveniente ou não é possível tratar com tecnologia web, como questões específicas de segurança ou acesso a recursos do dispositivo, por exemplo.

A aplicação híbrida, porém, exige maior esforço na estilização dos elementos HTML através de CSS, a fim de deixar a aplicação mais parecida com os componentes visuais nativos da plataforma. De outra forma, os usuários podem não se sentir confortáveis com a aparência e o comportamento dos componentes visuais da aplicação. Esse esforço deve ser planejado e executado levando sempre em consideração que o principal objetivo de embarcar o conteúdo web em uma aplicação nativa é reaproveitá-lo em várias plataformas móveis.

Para construir aplicações móveis híbridas, o desenvolvedor pode utilizar algum *framework* que visa a minimizar o esforço de construção e manutenção desse tipo de aplicação [30]. Normalmente, esses *frameworks* são integrados ao ambiente de desenvolvimento padrão da plataforma para a qual a aplicação foi inicialmente desenvolvida. Em seguida, para portar essa aplicação para outra plataforma, o código web comum deve ser copiado para o ambiente de desenvolvimento da nova plataforma alvo e o código nativo deve ser reescrito na nova linguagem.

3.3.1. Vantagens

- Integração da aplicação com os recursos do dispositivo móvel, como acelerômetro, GPS, câmera e bússola, por exemplo;
- Escrita da maior parte do código em linguagem padrão multiplataforma (HTML, CSS e JavaScript), possibilitando o reaproveitamento de código;
- Maior flexibilidade na escolha e na estilização dos componentes visuais da aplicação (componentes nativos ou web);
- Possibilidade de redução da quantidade de desenvolvedores especializados em cada plataforma;
- Possibilidade de armazenamento local em arquivos ou em banco de dados SQLite;
- Possibilidade de monitorar os *downloads* de uma aplicação baixada da *store* de aplicações;
- Monitoração das estatísticas de *download* e de satisfação dos usuários em relação a uma aplicação baixada da *store* de aplicações;
- Mais possibilidades de protocolos de comunicação com o servidor remoto, além daqueles suportados pelos navegadores web;
- Possibilidade de reaproveitar a lógica de negócios da aplicação, visto que esta pode ser incorporada no servidor remoto, ao invés de estar no cliente.

3.3.2. Desvantagens

- Demora para validar e publicar a aplicação em uma *store* de aplicações;
- Maior esforço para customizar os componentes da aplicação para deixar sua aparência e comportamento similares ao da plataforma;

- A migração dos usuários para uma nova versão da aplicação não é controlada;
- Existência de um custo extra de publicação da aplicação na *store* de aplicações do fabricante;
- Necessidade de disponibilizar o código da aplicação para o proprietário da *store* de aplicações durante o processo de validação e publicação.

3.3.3. Ferramentas de apoio

Para o desenvolvimento de aplicações híbridas, a ferramenta mais utilizada é uma biblioteca de desenvolvimento chamada PhoneGap [30]. O PhoneGap é uma solução da Adobe Systems que provê um conjunto de recursos para que o desenvolvedor possa construir aplicações que serão distribuídas para diversas plataformas, porém utilizando-se de HTML, CSS e JavaScript em vez de linguagens específicas de cada plataforma.

Para construir uma aplicação com o PhoneGap, o desenvolvedor escolhe o ambiente de desenvolvimento mais conveniente (XCode, Eclipse, Visual Studio, entre outros) e cria inicialmente um projeto de desenvolvimento para a aplicação. No projeto da aplicação, o desenvolvedor referencia a biblioteca do PhoneGap, que lhe dará acesso a um conjunto de funções em JavaScript para que seja possível acessar recursos embarcados no dispositivo móvel, como bússola, acelerômetro, GPS, câmera, entre outros. Assim, o desenvolvedor constrói a aplicação utilizando tecnologias Web e os recursos do PhoneGap, de forma que possa reutilizar esse código-fonte em projetos criados em ambientes de outras plataformas.

O PhoneGap, então, não dispensa a necessidade de ter projetos diferentes para cada plataforma móvel. Porém, cada projeto subsequente precisa apenas referenciar a biblioteca do PhoneGap e utilizar o mesmo código-fonte que foi construído no primeiro projeto. Dessa forma, o custo de manutenção dos diversos projetos é bastante reduzido, uma vez que o código desenvolvido para um projeto é quase todo reutilizado nos projetos subsequentes.

3.4. Orientações para escolha da arquitetura

Conforme apresentado nas Seções 3.1, 3.2 e 3.3, cada arquitetura possui vantagens e desvantagens que devem ser consideradas durante a

construção de qualquer aplicação móvel. Nesta seção, são apresentados critérios para que, de acordo com os requisitos e o *roadmap* da aplicação móvel a ser construída, a equipe de desenvolvimento possa escolher a arquitetura mais adequada (Figura 5).

O primeiro critério diz respeito ao acesso a recursos de hardware do dispositivo móvel. Quando a aplicação requer acesso a recursos como acelerômetro, barômetro e bússola, por exemplo, as arquiteturas que envolvem código nativo devem ser priorizadas. Isto porque, apesar de existirem esforços para a construção de especificações de acesso a esse tipo de recurso via um navegador web, a maioria dessas especificações ainda está em fase embrionária [31]. Portanto, normalmente é bem mais fácil acessar esses recursos através de código nativo.

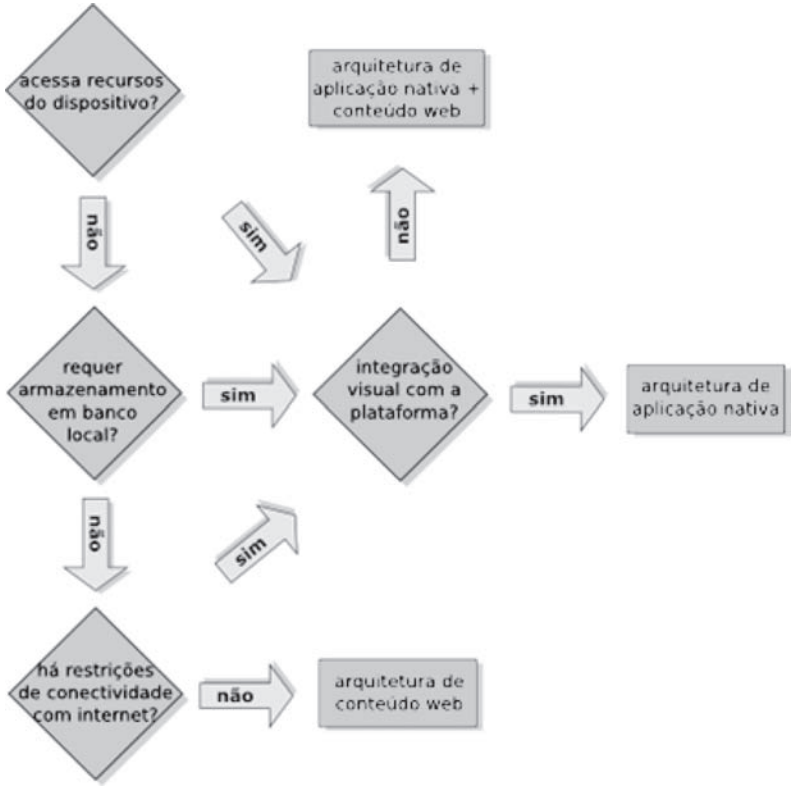


Figura 5: *Fluxo de decisão para escolha da arquitetura.*

Outro critério que aponta para o uso de arquiteturas que envolvem código nativo é o armazenamento local de informações. Se a aplicação requer

armazenamento e recuperação de informações estruturadas no dispositivo, as arquiteturas que envolvem código nativo devem ser priorizadas, visto que permitem a utilização de SQL para manipular um banco de dados local. Conforme explanado anteriormente, usando a arquitetura web, uma aplicação pode armazenar informações localmente no dispositivo através de arquivos. Porém, essa arquitetura não permite acessar o banco de dados SQLite, disponibilizado pelas principais plataformas móveis.

Outro critério muito importante que pode levar à escolha de uma das arquiteturas que envolvem código nativo é a conectividade com a Internet/intranet. Se a aplicação móvel a ser desenvolvida for utilizada em um contexto em que há restrições de conectividade com a Internet/intranet, a arquitetura web pode não ser a mais adequada. Isto porque a arquitetura web possui uma dependência muito forte de conectividade, visto que a interface e as informações da aplicação trafegam por meio da Internet/intranet para chegar ao navegador do dispositivo móvel. Essa necessidade, porém, pode ser amenizada com o uso de armazenamento local em arquivo.

Assim, se a aplicação móvel a ser construída não exigir acesso a recursos de hardware do dispositivo móvel ou a um banco de dados e não possuir restrições de conectividade, a arquitetura web móvel pode ser a mais recomendada.

Por outro lado, se a aplicação não corresponde a esses requisitos, é preciso considerar as arquiteturas que envolvem código nativo como principais opções. Nesse caso, a necessidade de se criar uma aplicação móvel com visual integrado à plataforma na qual será executada pode ser o critério mais importante para decidir entre a arquitetura nativa ou a arquitetura híbrida. Se esse for um requisito essencial da aplicação a ser construída, a arquitetura nativa pode ser a melhor opção, visto que os componentes visuais serão renderizados diretamente pela plataforma. Porém, se este não for o caso, a arquitetura híbrida pode ser uma alternativa suficiente e mais econômica.

Independentemente de todos os critérios apresentados na Figura 5, pode ser interessante avaliar também as habilidades e a quantidade de pessoas na equipe que irá desenvolver a aplicação. A arquitetura nativa pode requerer mais desenvolvedores especializados em cada plataforma móvel (p. ex.: Android e iOS), enquanto as arquiteturas web e híbrida podem requerer menos pessoal especializado, visto que boa parte do código pode ser desenvolvido em linguagem multiplataforma, como HTML, CSS e JavaScript.

4. Aplicações móveis

As arquiteturas apresentadas foram experimentadas em protótipos. Os protótipos desenvolvidos demonstraram que, para os casos estudados, a arquitetura para desenvolvimento de aplicações nativas se mostrou mais vantajosa. O Serpro desenvolveu e publicou aplicações móveis nativas para a RFB (Receita Federal do Brasil) e para o DNIT (Departamento Nacional de Infraestrutura de Transportes). As duas instituições inovaram em 2012, oferecendo soluções para dispositivos móveis, no âmbito do governo federal. Em junho de 2012, a RFB disponibilizou para os contribuintes o aplicativo móvel Pessoa Física e, desde julho, o Siesc Mobile está disponível para apoiar a fiscalização de obras públicas em andamento. As subseções a seguir apresentam as características dessas aplicações e analisam a seleção da arquitetura mais adequada para cada solução.

4.1. Aplicação Pessoa Física (RFB)

A aplicação Pessoa Física (Figura 6.a), disponibilizada pela RFB, permite que um contribuinte consulte a situação de sua restituição de Imposto de Renda (Figura 6.b). Além disso, um cidadão pode consultar a situação do seu Cadastro de Pessoa Física (CPF), identificando se está regular ou se existe alguma pendência. A aplicação também apresenta informações gerais sobre o Imposto de Renda e o processo de restituição.

Figura 6: *Aplicativo Pessoa Física: (a) tela principal; (b) consulta da restituição do IRPF[7]; avaliação do aplicativo (c) na loja Google Play [8] e (d) na loja App Store [7].*

O aplicativo Pessoa Física foi lançado no dia 06 de junho de 2012 e tem recebido avaliações muito positivas desde então. Na Google Play[8], a loja de aplicativos Android, a aplicação está avaliada com 4,6 estrelas, de um máximo de 5 estrelas, segundo consulta realizada em outubro de 2012 [8] (Figura 6.c). Já na App Store[9], a loja de aplicativos iOS, a aplicação está avaliada, por 85 usuários, com nota 4, em um total de 5 (Figura 6.d) . Além disso, o número de *downloads* em cada uma dessas lojas é bem similar e por volta de 98 mil [7][8]. Esses números indicam uma grande aceitação do aplicativo pelos contribuintes brasileiros.

A solução foi construída utilizando-se de linguagens e tecnologias disponíveis para cada plataforma específica, ou seja, um exemplo de uso da arquitetura de aplicações nativas.

A decisão pelo desenvolvimento com a arquitetura nativa foi baseada nos seguintes requisitos: (i) necessidade de integração visual com cada plataforma; (ii) necessidade de disponibilizar nas lojas de aplicativos para dispositivos móveis, para facilitar o descobrimento dos aplicativos, dado o amplo reconhecimento destas fontes pelos usuários de *smartphones*; (iii) necessidade de adequar as imagens por plataforma para seguir o padrão de qualidade visual da RFB (um aplicativo genérico, com todas as imagens, seria muito grande e o tempo de *download* seria muito grande). Como este foi o primeiro aplicativo para dispositivos móveis, lançado pela RFB, optou-se por investir em uma interface simples, fácil de usar e natural para os usuários, por isso a integração visual com cada plataforma era tão importante: para que os usuários se sentissem confortáveis com a interface. A equipe observou que os aplicativos nativos, via de regra, eram mais bem avaliados no quesito interface. As avaliações dos usuários demonstraram que a decisão foi acertada. Um dos primeiros usuários a registrar comentários avaliou: "Um bom começo para uma app do governo. Usabilidade tranquila. Poucos serviços. Direto ao ponto". Outro usuário registrou: "Simples, fácil de usar ". E, um terceiro: "Simples e funcional". Essa foi a tônica das avaliações positivas nas lojas.

Apesar da decisão mostrar-se acertada, a escolha da arquitetura nativa requereu o desenvolvimento de dois aplicativos: um para a plataforma Android e outro para Apple. A escolha das plataformas foi baseada na representatividade no mercado brasileiro, já que o aplicativo deveria ser disponibilizado para os cidadãos. As duas plataformas mais representativas no mercado foram selecionadas. Além disso, a plataforma Android tem

[8] Google Play é a "loja" de aplicativos Android. Esta "loja", além de disponibilizar o aplicativo, permite que o usuário o avalie com uma nota até 5.

[9] App Store é a "loja" de aplicativos da Apple. Esta "loja", além de disponibilizar o aplicativo, permite que o usuário o avalie com uma nota até 5.

a grande vantagem de estimular o uso de software livre. A versão para Android foi construída com a linguagem Java e o ambiente Eclipse, em conjunto com o Software Development Kit (SDK) da Google para Android. Já a versão para iOS (sistema operacional móvel da Apple) foi construída com a linguagem Objective-C e o ambiente de desenvolvimento XCode.

Com o desenvolvimento por plataforma, duas equipes foram mobilizadas, e após a fase de requisitos, as fases do desenvolvimento foram replicadas. Além do retrabalho, essa estratégia demandou sincronização constante entre as equipes para garantir a proximidade entre as duas versões. A complexidade do desenvolvimento e manutenção aumenta à medida em que novas plataformas são consideradas. Assim, esta deve ser uma preocupação ao optar-se pelo desenvolvimento nativo. Outra questão vivenciada foi que a publicação nas lojas pode impactar no cronograma de disponibilização dos aplicativos. A publicação, ou não, depende de uma avaliação de cada loja, e não há como priorizar publicações. Antes de publicar, a loja avalia alguns critérios para autorizar a publicação de um novo aplicativo, ou até mesmo de uma nova versão. Esses critérios exigem alguns cuidados com o design dos aplicativos e é recomendado o uso de uma lista de verificação na fase de requisitos e de testes para evitar problemas na publicação. Na loja da Apple, por exemplo, cada nova submissão, demanda ao menos 24 horas para avaliação.

4.2. Aplicação SIESC Mobile

Outro aplicativo móvel desenvolvido pelo Serpro foi o SIESC Mobile. Seu objetivo principal é melhorar a qualidade das fiscalizações das obras que são de responsabilidade do DNIT, proporcionando também redução de tempo e custo. Antes do SIESC Mobile, um engenheiro (que poderia ser de uma empreiteira contratada pelo governo) levava um formulário impresso à obra para preenchê-lo com todas as características observadas (localização, altitude, comprimento, tipo...) e com sua situação atual (falhas na estrutura, avanço da construção em relação à última medição...). Após o preenchimento desse formulário, as informações em papel eram transcritas para o Sistema de Acompanhamento de Contratos (SIAC). A partir do SIESC Mobile, as informações são captadas *in loco*. Recursos integrados, como GPS e câmera, permitem a complementação das informações com dados precisos de localização e fotos do andamento da obra, que mostram melhor a sua situação atual. Além disso, várias validações na entrada dos dados aumentam a qualidade das informações,

agilizam o processo e evitam retornos desnecessários à obra para correção de dados inválidos, que algumas vezes só eram identificados no escritório. Ou seja, o investimento no desenvolvimento de uma aplicação móvel foi importante para dar mais celeridade e confiabilidade ao processo de registro dos dados de fiscalização. Isso é particularmente importante, levando-se em consideração que as obras fiscalizadas, muitas vezes, estão em locais distantes da zona urbana e que o tempo de locomoção até o local e de volta até o posto de trabalho pode ser grande.

A expectativa é que a solução SIESC Mobile possibilite quitar os serviços contratados em um terço do tempo necessário anteriormente com os formulários de papel, e também que os valores dos contratos podem ser reduzidos mediante a redução dos custos de fiscalização para a empresa contratada.

Durante a execução de um piloto da solução realizado em 21 de junho de 2012, o engenheiro Marcos Antônio Borges, da Strata Engenharia, usou a aplicação SIESC Mobile e comentou: "Operei o SIESC no *tablet*, fiz fotos georreferenciadas, foi genial mesmo... a ferramenta vai facilitar o acompanhamento de cada fase de execução e vai nos ajudar a ter uma visão macro da obra" [32].

Assim como o Pessoa Física, o SIESC Mobile é um exemplo de aplicação construída usando a arquitetura nativa, ou seja, utilizando-se de tecnologias que executam apenas em uma plataforma específica. A decisão pelo desenvolvimento com a arquitetura nativa foi baseada nos seguintes requisitos: (i) necessidade de utilizar recursos específicos dos dispositivos, no caso, câmera e celular; (ii) restrições de conectividade com a internet, já que as obras estão espalhadas pelo Brasil, muitas vezes, longe de centros urbanos, como é o caso de estradas; e (iii) necessidade de armazenamento local, já que centenas de informações são capturadas pelo dispositivo *off-line*.

No caso do SIESC, a plataforma escolhida foi o Android. Essa escolha foi baseada no fato de que esta é uma plataforma aberta e que possui dispositivos a preços mais acessíveis. O fato de ser uma plataforma aberta demonstra o alinhamento da solução com a estratégia governamental de adoção e fomento ao software livre. Além disso, o preço mais acessível dos dispositivos que rodam o sistema operacional Android é importante para minimizar os custos de aquisição de tecnologia, visando à redução dos gastos públicos. Essa decisão foi possível porque, diferentemente da aplicação Pessoa Física, que foi desenvolvida para uso pelo cidadão, o SIESC Mobile foi construído para ser utilizado por engenheiros do DNIT e de empresas terceirizadas que fiscalizam obras de construção civil governamentais. À medida que o aplicativo for sendo utilizado, caso haja a

necessidade de uso em outra plataforma, o desenvolvimento de uma nova versão poderá ser considerado.

O SIESC Mobile (Figura 7) foi construído como uma aplicação para ser executada em *tablets* Android e possibilita a substituição dos formulários de fiscalização de obras por um dispositivo portátil com interface moderna e que agrega informações de diversas obras. Além disso, as informações registradas na aplicação podem ser enviadas diretamente do *tablet* para o DNIT a partir de uma conexão móvel 3G, por exemplo, diminuindo o prazo de pagamento dos contratos e aumentando o fluxo de caixa do contrato.

Figura 7: *Utilização do SIESC Mobile para efetuar uma medição de contrato.*

O aplicativo não foi disponibilizado na loja de aplicativos Google Play, já que seu público-alvo é específico. Em vez disso, a aplicação foi disponibilizada no próprio portal do DNIT, através de um link para *download* com instruções de instalação e uso (Figura 8). A distribuição de aplicativos por meio deste portal já é natural para o público-alvo do aplicativo.

Figura 8: *Página do Portal do DNIT que disponibiliza o SIESC Mobile para download.*

5. Trabalhos Futuros

O desenvolvimento de aplicações para dispositivos móveis e o estabelecimento de modelos de arquitetura a serem utilizados por esse tipo de aplicação são algumas das iniciativas que o Serpro apresenta para proporcionar inovação a seus clientes. Com o domínio das tecnologias disponíveis para dispositivos móveis, já está sendo possível oferecer soluções inovadoras para o governo brasileiro. O próximo passo é ampliar a oferta de soluções para estes dispositivos viabilizando uma aproximação cada vez maior entre os serviços de governo e o cidadão que utiliza estes serviços.

Com o objetivo de continuar a oferecer produtos eficientes e inovadores, o Serpro está constantemente estudando novas tecnologias. A partir de agora, os próximos passos estarão direcionados para o estudo de novas interfaces e formas de interação entre as soluções de software de governo e os cidadãos. Dentre essas tecnologias, pode-se citar o mapeamento de gestos corporais para controlar hardware e software e a utilização da TV digital como uma nova interface de usuário.

Além disso, o Serpro também tem prospectado tecnologias que permitam o gerenciamento e controle de dispositivos móveis e que

possibilitem a utilização de dispositivos particulares em atividades corporativas; este último vem despontando como uma tendência no mercado, e é conhecido pelo termo em inglês *BYOD (Bring Your Own Device)*.

6. Conclusão

Este artigo apresentou um consolidado de dados que mostram que a presença da mobilidade no Brasil tem crescido a cada ano e que, muitas vezes, o ritmo desse crescimento tem sido maior do que o observado em outros países. A presença da mobilidade no Brasil se evidencia principalmente através do crescimento do uso da internet móvel e do aumento significativo de vendas de dispositivos móveis, como *tablets* e *smartphones*, a cada ano.

Esse crescimento tem impacto significativo na vida dos brasileiros, principalmente pelo uso frequente de aplicativos que utilizam recursos embarcados nos dispositivos móveis, como: GPS, câmera, acelerômetro, entre outros. Esses aplicativos são utilizados principalmente para comunicação, informação e entretenimento, e podem ser construídos para plataformas específicas, como Android e iOS, ou como aplicação web para acesso via navegadores web móveis.

Diferentes tipos de arquiteturas para aplicações móveis foram discutidos, apresentando suas vantagens e desvantagens com base em requisitos e contexto de uso das soluções.

No âmbito do governo federal brasileiro, foram apresentados dois estudos de caso desenvolvidos pelo Serpro, para os clientes Receita Federal e DNIT. Esses estudos de caso mostraram exemplos de como a mobilidade pode ser utilizada para prover serviços de governo para o cidadão, como é o caso do aplicativo Pessoa Física, bem como para melhorar a eficiência das atividades de fiscalização executadas pelo Estado, como é o caso do aplicativo SIESC Mobile.

Os estudos de caso apresentados neste artigo são uma parcela pequena dos aplicativos desenvolvidos pelos governos das esferas federal, estadual e municipal que podem ser encontrados nas lojas virtuais de aplicativos móveis das plataformas. Isso mostra que cada vez mais serviços públicos são oferecidos pelos governos através de aplicativos para dispositivos móveis, facilitando a vida do cidadão brasileiro.

Além dos aplicativos desenvolvidos pelo governo, alguns aplicativos são construídos diretamente por cidadãos a partir de dados disponibilizados abertamente pelo governo. Dessa forma, o conjunto de serviços governamentais para plataformas móveis não depende apenas de

órgãos e empresas públicas, mas também pode ser ampliado pelos próprios cidadãos que se beneficiam com o uso desses serviços.

Apesar do crescimento já evidenciado, a tendência da mobilidade no Brasil ainda é de crescimento. Com *smartphones* e *tablets*, os cidadãos dispõem de um recurso poderoso para acessar serviços a partir de qualquer lugar e a qualquer momento. Por isso, o Serpro tem investido fortemente em preparar-se e aproveitar a oportunidade de prover soluções móveis para o governo e para o cidadão melhorando a qualidade dos serviços prestados e possibilitando uma aproximação maior dos serviços de governo com a população. Em especial, a Copa do Mundo 2014 e as Olimpíadas 2016 serão ótimas oportunidades para disponibilizar uma variedade grande de serviços móveis, que podem ajudar as pessoas que irão participar desses dois grandes eventos.

7. Referências

[1] "Acesso a Internet móvel no Brasil praticamente dobra", Jornal O Povo Online. Disponível em <http://www.opovo.com.br/app/opovo/tendencias/2012/01/27/noticiasjornaltendencias,2774145/acesso-a-internet-movel-no-brasil-praticamente-dobra.shtml. Acessado em 26/10/2012>.

[2] "Nosso Planeta Mobile: Brasil", Google Inc.. Disponível em <http://services.google.com/fh/files/blogs/our_mobile_planet_brazil_pt_BR.pdf>. Acessado em 26/10/2012.

[3] Aplicativo "Voos Online", Infraero. Disponível em <http://www.infraero.gov.br/fiquepordentro/#/download>. Acessado em 26/10/2012.

[4] Aplicativo "Banco do Brasil", Banco do Brasil S.A. Disponível em <https://play.google.com/store/apps/details?id=br.com.bb.android. Acessado em 26/10/2012>.

[5] "Celepar desenvolve novo site do Verão Paraná para plataformas móveis", Celepar. Disponível em <http://www.celepar.pr.gov.br/modules/noticias/article.php?storyid=871. Acessado em 26/10/2012>.

[6] "Brasil em Cidades é sucesso de downloads em um mês", Portal Brasil. Disponível em <http://www.brasil.gov.br/noticias/arquivos/2011/12/5/portal-brasil-em-cidades-oferece-informacoes-sobre-planejamento-urbano-aos-municipios-de-todo-o-pais>. Acessado em 26/10/2012.

[7] Aplicativo "Pessoa Física" na App Store. Disponível em <https://itunes.apple.com/br/app/pessoa-fisica/id529883041?mt=8>. Acessado em 26/10/2012.

[8] Aplicativo "Pessoa Física" na Google Play. Disponível em <https://play.google.com/store/apps/details?id=br.gov.fazenda.receita.pessoafisica>. Acessado em 26/10/2012.

[9] Aplicativo "Viajantes no Exterior" na App Store. Disponível em <https://itunes.apple.com/br/app/viajantes-no-exterior/id547724184?mt=8>. Acessado em 26/10/2012.

[10] Aplicativo "Viajantes no Exterior" na Google Play. Disponível em <https://play.google.com/store/apps/details?id=br.gov.fazenda.receita.viajantesexterior>. Acessado em 26/10/2012.

[11] Aplicativo "SIESC Mobile" no sítio do DNIT. Disponível em <https://gestao.dnit.gov.br/sistemas-gerenciais/siesc. Acessado em 26/10/2012>.

[12] Aplicativo "SGO Mobile" no sítio do DNIT. Disponível em <http://www.dnit.gov.br/sistemas-gerenciais/sgo>. Acessado em 26/10/2012.

[13] "Scope of Mobile Web Best Practices", W3C Standards. Disponível em <http://www.w3.org/TR/mobile-bp-scope/#s3>. Acessado em 26/10/2012.

[14] "Picking A Mobile Support Strategy For Your Website", Smashing Magazine. Disponível em <http://mobile.smashingmagazine.com/2011/07/11/picking-a-mobile-support-strategy-for-your-website/>. Acessado em 26/10/2012.

[15] "The Goldilocks Approach", Front. Disponível em <http://goldilocksapproach.com/>. Acessado em 26/10/2012.

[16] "CSS Media Queries & Using Available Space" CSS Tricks. Disponível em <http://css-tricks.com/css-media-queries/>. Acessado em 26/10/2012.

[17] "Resolution Independence With SVG", Smashing Magazine. Disponível em <http://coding.smashingmagazine.com/2012/01/16/resolution-independence-with-svg/>. Acessado em 26/10/2012.

[18] "Mobile Content: If in Doubt, Leave It Out", Jakob Nielsen. Disponível em <http://www.useit.com/alertbox/mobile-writing.html>. Acessado em 26/10/2012.

[19] "jQuery Mobile", The jQuery Foundation. Disponível em <http://jquerymobile.com/>. Acessado em 26/10/2012.

[20] "Sencha Touch", Sencha Inc. Disponível em <http://www.sencha.com/products/touch>. Acessado em 26/10/2012.

[21] "Plone", Plone Foundation. Disponível em <http://plone.org/>. Acessado em 26/10/2012.

[22] "Liferay", Liferay Inc. Disponível em <http://www.liferay.com/>. Acessado em 26/10/2012.

[23] "SQLite", SQLite. Disponível em <http://www.sqlite.org/features.html>. Acessado em 26/10/2012.

[24] "AppAccelerator Titanium", AppAccelerator. Disponível em <http://www.appcelerator.com/platform/titanium-sdk>. Acessado em 26/10/2012.

[25] "XCode", Apple. Disponível em <https://developer.apple.com/xcode/>. Acessado em 26/10/2012.

[26] "ADT Plugin", Google. Disponível em <http://developer.android.com/tools/sdk/eclipse-adt.html>. Acessado em 26/10/2012.

[27] "SDK Android", Google. Disponível em <http://developer.android.com/sdk/index.html>. Acessado em 26/10/2012.

[28] "Windows Phone SDK", Microsoft. Disponível em <http://www.microsoft.com/en-us/download/details.aspx?id=27570>. Acessado em 26/10/2012.

[29] "PhoneGap Storage", Adobe Systems. Disponível em <http://www.sqlite.org/features.html>. Acessado em 26/10/2012.

[30] "PhoneGap", Adobe Systems. Disponível em <http://phonegap.com/>. Acessado em 26/10/2012.

[31] "Standards for Web Applications on Mobile", W3C Standards. Disponível em <http://www.w3.org/wiki/Standards_for_Web_Applications_on_Mobile>. Acessado em 26/10/2012.

[32] "Na Palma da Mão", Revista Tema (pp. 14-19). Disponível em <http://tema.serpro.gov.br/pub/serpro//index.jsp?edicao=38>. Acessado em 26/10/2012.

Mobilidade digital: tecnologias e tendências

Djamel F. H. Sadok

Abstract

Mobilidade em qualquer contexto é sinônimo de liberdade. No mundo digital, governos e negócios reconhecem a importância de aproveitar novos canais de comunicação em busca de melhorar seus serviços e atingir novas massas. A mobilidade se manifesta hoje em três dimensões: usuário, conteúdo e dispositivo. A taxa de inovação em ambientes móveis claramente ultrapassa os desafios regulatórios, de segurança, organizacionais, etc. Exemplos de aplicações emergentes incluem médicos acessando registros de pacientes por meio de tecnologias móveis, conexão de veículos e seus usuários, passageiros acessando a Internet em pleno voo. Este *Keynote* mostrará as tecnologias por trás destas conquistas e como elas estão evoluindo, além de apontar desafios que devem ser considerados.

Introdução

A arquitetura Internet se baseia num conjunto de invariantes o suficiente para não sofrer mudanças profundas ao longo das últimas três décadas. Muitos destes princípios são reiteradamente ameaçados e questionados em nome de inúmeras iniciativas como: a Internet do futuro, a Internet das coisas, Redes definidas por Software, Redes centradas em Conteúdo, Novas Arquiteturas para a Internet, etc.

Duas escolas de pensamento surgiram entre os pesquisadores. A primeira corrente advoga que a arquitetura Internet está engessada por suas próprias regras e que uma alternativa deve considerar "recomeçar do zero" (*clean slate*). A segunda escolheu um caminho evolucionário ao contrário de uma revolução. Atualmente, a maioria dos pesquisadores reconhece a dificuldade que é propor um novo projeto totalmente desacoplado da arquitetura existente e admite a necessidade de mudanças gradativas. Nós reconhecemos hoje que o sucesso da Internet se deve a fatores inerentes a seu projeto original como: simplicidade de requisitos, abertura às contribuições do público em geral, apoio financeiro de um governo durante quase duas décadas, decisões baseadas em consenso aproximado e código que executa.

Hoje, quando visitamos este projeto podemos detectar várias falhas nele. Podemos ver que a Internet posiciona a inteligência das aplicações totalmente nos pontos de borda e não nos elementos retransmissores. A rede inteira é vista como uma linha de transmissão de acordo com o argumento fim a fim. A introdução de redes definidas por software e de redes centradas em conteúdo pode mudar este princípio e, consequentemente, o modelo de negócio por trás dele. Podemos, no futuro, enumerar todos os objetos de conteúdo e manipulá-los independentemente, ao invés de simplesmente manipular pacotes. A mobilidade da informação deve ser registrada e rastreada pela rede quando requisitada por seus usuários.

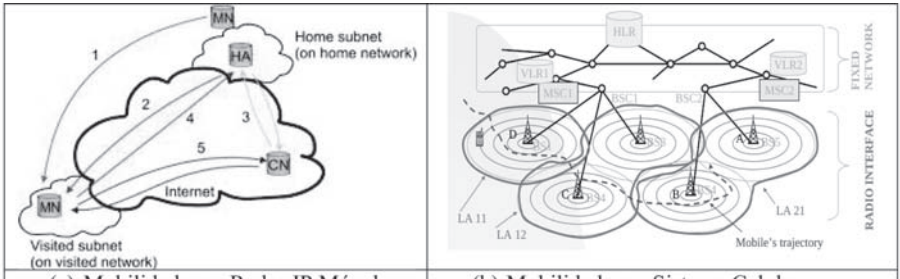
Não podemos simplesmente culpar os pioneiros deste projeto Internet. Muitos serviços que hoje consideramos essenciais num sistema de comunicação não eram conhecidos nem contemplados duas ou três décadas atrás. Os engenheiros de Telecom não pensavam nos anos 1970 e 1980 na existência de: a) usuários móveis; b) sistemas intermediários na rede como *proxies* e *firewalls*, na necessidade de desligar roteadores ou mover recursos de armazenamento dinamicamente na rede para economizar energia; c) identificação e mobilidade de conteúdo; d) segurança da informação; etc.

O espaço de endereçamento IP, versões 4 e 6, apresenta a identificação do elemento fim como também a sua localização na Internet. Descobrimos hoje que esta dupla responsabilidade do endereço atrapalha o roteamento da informação numa rede com destinos móveis já que enquanto a identificação do objeto fim não muda, a sua localização muda.

Pensando nos desafios impostos para a acomodação de usuários móveis na Internet, tivemos que revisitar a arquitetura Internet várias vezes, de modo a produzir soluções. Neste texto, vamos examinar estas tecnologias e soluções introduzidas na arquitetura Internet e em Redes de Telecom convergentes para suportar a mobilidade de usuários e do conteúdo.

Semelhança IP Móvel e Mobilidade Celular

Com o surgimento dos sistemas celulares e redes sem fio nos anos 1990, ficou evidente a necessidade de introduzir o suporte à mobilidade na arquitetura Internet. Esta nova comunidade de usuários sem fio demanda conectividade contínua quando se deslocam. A inspiração do mundo IP apareceu entre 1990 e 1995 com as primeiras pesquisas introduzindo esquemas que alteram o roteamento IP para contemplar cenários com mobilidade. Este esquema parte do princípio de que a arquitetura IP lida apenas com padrões nos níveis de rede (nível do IP), transporte (nível fim a fim com os protocolos TCP e UDP) e no nível de aplicação. Assim, a mudança no IP, identificada como IP Móvel, requer a introdução de dois servidores: o *home agent* (que fica na rede original do usuário) e o *foreign agent* (que fica na rede visitada). Estes dois processos interceptam a transmissão dos pacotes para usuários móveis e redirecionam estes pacotes para a nova localização do usuário.



(a) Mobilidade em Redes IP Móvel

(b) Mobilidade em Sistema Celular – *Roaming*

Figura 1: *Mobilidade Internet versus Sistema Celular.*

A solução IP Móvel é muito semelhante à solução adotada pelos padrões de telefonia celular para suportar *Roaming*. Examinando a Figura 1 podemos ver que o roteamento dos pacotes IP entre o nó correspondente (*correspondent node* – CN) e o nó móvel (*mobile node* – MN) passa sempre pela rede de origem deste no móvel e especificamente pelo agente *home agent* (HA). Diferentemente, as mensagens enviadas de MN para CN serão enviadas diretamente através da Internet. O leitor deve se perguntar o porquê desta diferença. Simplesmente porque MN mantém seu endereço enquanto muda de localização e a rede só está preparada para encaminhar os pacotes destinados à MN para a sua rede de origem. Além disso, os pacotes devem vir com o endereço da sua rede local, neste caso, o endereço do seu agente caseiro (HA), para evitar que outros usuários anunciem endereços

que não são deles e os hosts na Internet acreditem. No caso do IP Móvel, o HA da rede de origem do MN garante a confiabilidade da conexão. Este é mais um exemplo onde, em nome da segurança, sacrificamos a eficiência. Este problema não se repete no caso do IPv6, já que ao contrário do IP, este suporta segurança de maneira nativa.

Dupla Responsabilidade do Endereço IP

Com a queda dos investimentos nas empresas ".com" no início da primeira década dos anos 2000, o governo americano decidiu investir em novas pesquisas incluindo projetos com propostas para novas arquiteturas Internet. No caso do endereço IP, ficou claro que a sua dupla responsabilidade prejudica a mobilidade de usuários e que um desacoplamento entre identificação e localização se faz necessário. O *Host Identity Protocol* (HIP) é uma proposta que faz esta separação criando o localizador e o identificador (*Host Identity* – HI) de um sistema fim. Assim, o HIP permite que um nó tenha mais de um endereço. Este mecanismo é conhecido como *multihoming*. Para atingir seu objetivo, o HIP desacopla a camada de transporte da camada do IP como mostra a Figura 2 (a).

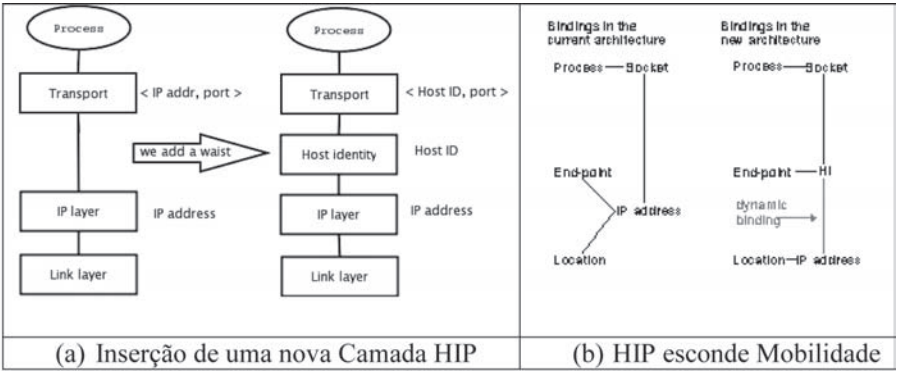


Figura 2: *Arquitetura HIP da Internet do Futuro.*

Vemos também no lado (b) da Figura 2 que o HIP entre o HI e a localização IP (parte em vermelho) esconde a mobilidade para os protocolos de alto nível como o TCP, fazendo com que estes continuem funcionando corretamente, de maneira transparente, inclusive em cenários móveis.

Micromobilidade

Ambas as soluções, Celular e IP Móvel, empregam servidores especializados no rastreamento dos usuários. No caso dos sistemas celulares, são duas bases de informação: a) os usuários da empresa de Telecom (*Home Location Register* – HLR) e b) o banco de dados de usuários visitantes (*Visitor Location Register* – VLR). Quando um usuário liga seu aparelho, ocorre uma operação de registro do aparelho e da localização do usuário. Esta informação de localização é atualizada à medida que o usuário se movimenta numa região, ver Figura 1. De maneira semelhante, o IP Móvel emprega dois agentes (HA) e (FA) para registrar a localização de usuários caseiros e visitantes, respectivamente. Fica claro que a atualização da localização do usuário nestas bases de informação deve acontecer cada vez que houver sua movimentação. Isso é feito através de mensagens especiais de sinalização.

Quando as áreas de cobertura são pequenas, a taxa de *roaming* cresce e o tráfego de atualização de localização pode ficar pesado para o sistema, implicando na perda de escalabilidade. Este problema é reconhecido por sistemas celulares e também pela comunidade Internet. Para sua solução, um sistema de cobertura hierárquico é adotado, ver exemplo na Figura 3. Podemos conectar usuários com alta mobilidade em células com cobertura grande para evitar uma alta frequência de atualizações. Temos hoje uma gama diversificada na hierarquia de células em termos de cobertura incluindo células macro, micro, pico e FemtoCells.

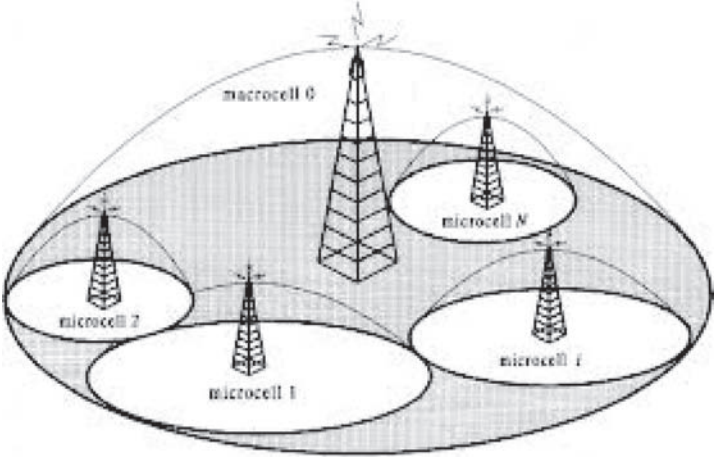


Figura 3: *Macro e microcélulas para Alta Mobilidade.*

No caso da Internet, várias soluções foram propostas, e uma delas coincidentemente é chamada *Cellular IP* (CIP). O IP Móvel é otimizado somente para mobilidade macro e com cenários com baixa mobilidade. O CIP é otimizado para hosts com alta mobilidade além de ser compatível com o IP Móvel como mostra a Figura 4. O CIP pode acomodar um alto número de usuários móveis separando os que são móveis dos outros.

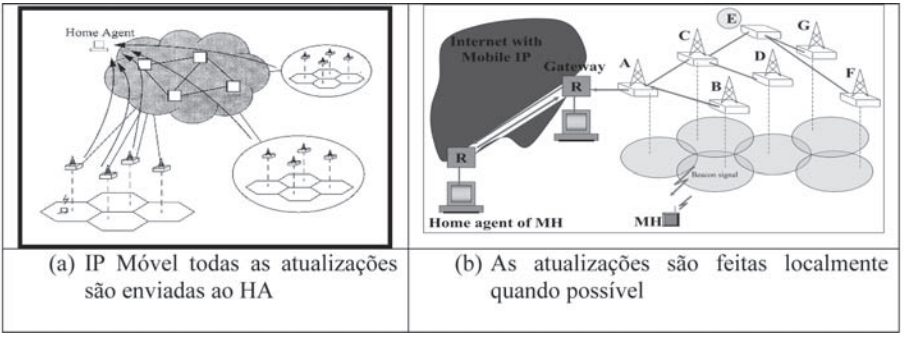


Figura 4: *Micromobilidade no Celular IP (CIP).*

Femtocells

Femtocells são pequenas células de baixa potência semelhantes ao ponto de acesso Wireless LAN e que podem ser interligadas em qualquer lugar na Internet para oferecer mobilidade. Imagine um evento ou uma conferência sendo organizados numa região do país sem cobertura celular. Muitos participantes vão se sentir isolados, neste caso, já que seus colegas e familiares não vão poder entrar em contato por meio da telefonia celular. As operadoras de telefonia celular oferecem Femtocell como uma estação rádio-base completa e portátil. Os organizadores do evento podem então conectar a estação Femtocell à Internet ou em outra linha dedicada, como um enlace de satélite. Assim, o resultado seria que os usuários estariam conectados através de uma cobertura celular normal emitida pela Femtocell que, por sua vez, estaria conectada com a Central de Comutação Celular (CCC) da operadora de Telecom através da Internet como mostra a Figura 5.

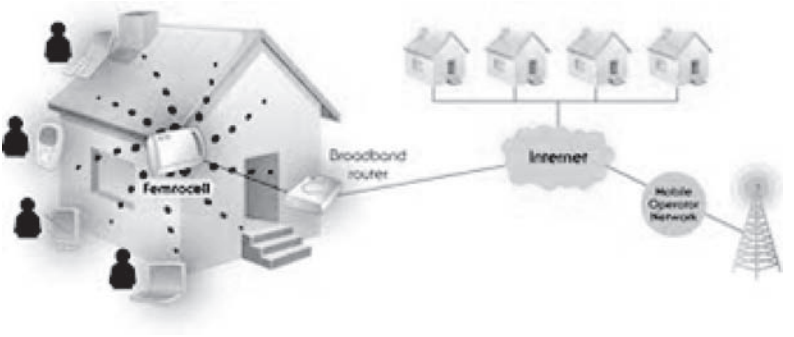
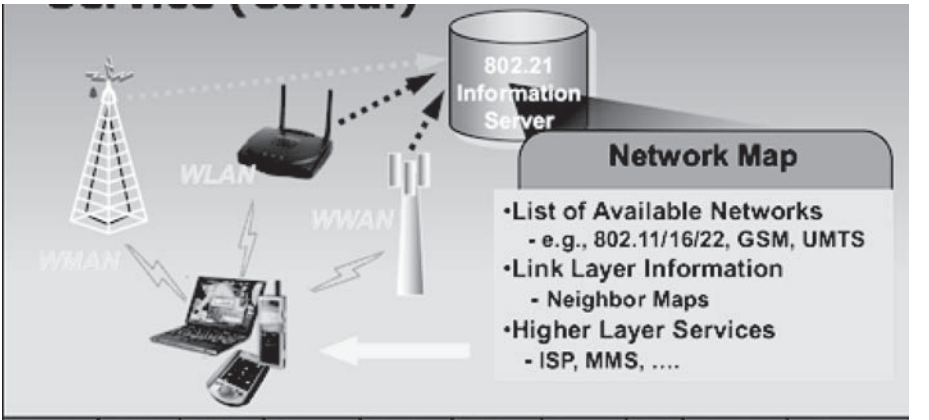


Figura 5: *Exemplo de uso de uma Femtocell.*

Para os interessados, hoje há uma implementação de *software* aberto de Femtocell, conhecida como OpenBTS. Este software precisa de outro *software* aberto chamado GNURadio, que implementa os módulos de comunicação sem fio e de rádio em *software*. As únicas partes que não são em *software* neste caso são a antena e os conversores analógico-digital (A/D/A).

IEEE 802.21

Em 2008 surgiu da IEEE o padrão internacional IEEE 802.21, que descreve como efetuar um *handoff* entre tecnologias sem fio diferentes como 802.11 (WLAN), 802.16 (WiMax) e 3G. Na verdade, até redes cabeadas, como a Ethernet (802.3), são contempladas. Além de reduzir o tempo de troca de tecnologia de acesso, o padrão permite aos usuários móveis a seleção entre as tecnologias disponíveis e a autenticação dos usuários. O 802.21 permite a continuidade de uma sessão de um usuário móvel de maneira transparente entre tecnologias diferentes.



Network Type	SSID/ Cell ID	BSSID	Operator	Security	EAP Type	Channel	QoS	Physical Layer	Data Rate
GSM	13389	N/A	Oper-1	NA	NA	1900	N/A	N/A	9.6 Kbps
802.11n	Enterprise	00:00:...	Oper-2	.11i	EAP-PEAP	6	.11e	OFDM	100 Mbps
802.16e	NA	NA	Oper-3	PKM	EAP-PEAP	11	Yes	OFDM	40 Mbps

Figura 6: *802.21 Mobilidade Independente da Tecnologia sem Fio Empregada.*

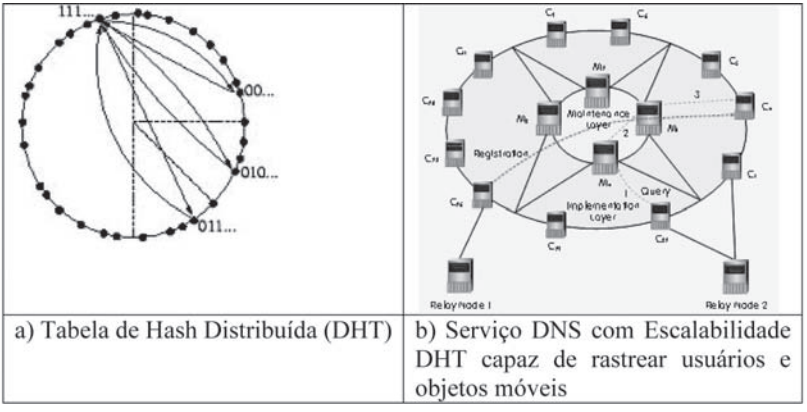
Mobilidade da informação na Internet do futuro

A Internet viveu recentemente uma mudança de paradigma de processamento e distribuição de conteúdo conhecida como comunicação Peer-to-Peer (P2P). Diferentemente do modelo cliente/servidor, o modelo P2P limita o poder de servidores intermediários para concentrá-lo nos hosts fim. Aplicações P2P, principalmente para a troca de conteúdo, surgiram. A localização da informação neste caso é baseada no uso de Tabelas Hash Distribuídas (DHT).

Algumas propostas para a próxima Internet identificaram as limitações do Domain Name System (DNS) na sua potência de escalar para redes que precisam de registros atualizados sobre usuários e conteúdos móveis. Obviamente, o DNS não foi otimizado para consultas complexas. Como solução, várias propostas propõem extensões para o DNS com sistemas mais rápidos de acesso, como o DHT que tem uma média de acesso de *log(n)*.

Podemos imaginar um cenário futuro no qual o e-mail de um usuário não seja ligado a sua localização. Assim, caso o usuário mude de emprego, por exemplo, este não precisará mudar sua identificação de e-mail. Podemos aplicar este raciocínio para as páginas da Web que poderiam ser identificadas por meio de endereços desacoplados da localização. Assim,

o sistema DHT terá a tarefa de localizar a entrada a partir da informação sobre um objeto, como um e-mail ou página Web. Cada vez que um objeto muda de localização ou quando é replicado na Internet, este deve avisar o sistema DHT para que haja atualização em seus dados.



a) Tabela de Hash Distribuída (DHT)

b) Serviço DNS com Escalabilidade DHT capaz de rastrear usuários e objetos móveis

Figura 7: *O Papel do DHT na Mobilidade.*

Mobilidade de conteúdo – redes centradas em conteúdo

Redes centradas em conteúdo (*Content-centric Networks* – CCN) pretendem oferecer uma disseminação eficiente da informação. Este novo paradigma quebra o argumento tradicional da Internet, conhecido como o argumento fim a fim. CCNs introduzem processamento nos elementos da rede com o pretexto de melhorar a mobilidade do conteúdo. Nestas propostas, objetos de informação são identificados de maneira única e roteados na Internet. A mobilidade da informação é característica principal deste novo tipo de arquitetura.

Mobilidade de recursos – OpenFlow

A fim de experimentar com a rede de um Campus Universitário, pesquisadores da Universidade de Stanford inventaram o OpenFlow. Podemos considerar que o OpenFlow é uma das tecnologias propostas para a Internet do futuro que mais atraiu interesse. Simplesmente, o OpenFlow permite a instalação em Switches e Roteadores de comandos e filtros gerenciados por um controlador central como mostra a Figura 8.

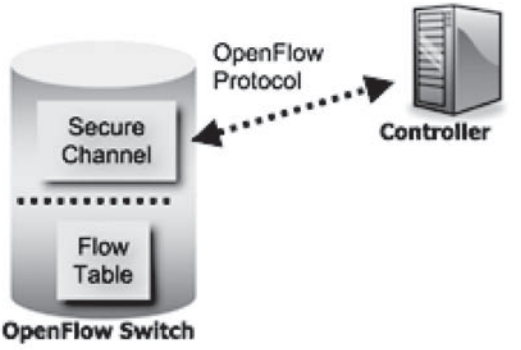


Figura 8: *Arquitetura e Protocolo OpenFlow.*

Usando o OpenFlow, um gerente de uma rede pode pedir que certos pacotes sejam encaminhados para um destino específico, tirados da rede, ter seus cabeçalhos alterados etc. Esta última ação, na forma de alteração dos cabeçalhos, permite a migração de fluxos, ou seja, a mobilidade de usuários.

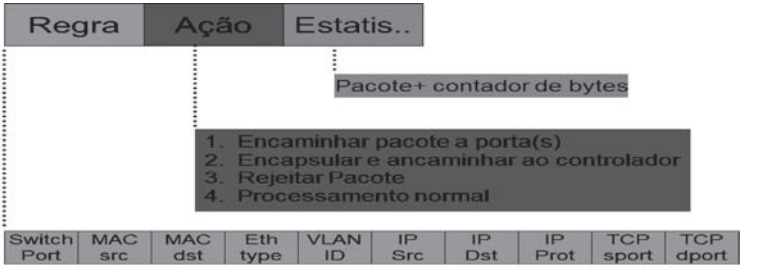


Figura 9: *Campos que Podem Alterados e Ações de um Switch OpenFlow.*

Assim, quando um pacote do experimento chega a um comutador ele é encaminhado para o Controlador, que é responsável por escolher a melhor rota para o pacote seguir na rede a partir dos mecanismos do protocolo de roteamento proposto. Após isso, o Controlador adiciona entradas na Tabela de Fluxos de cada comutador pertencente à rota escolhida. Os próximos pacotes desse fluxo que forem encaminhados na rede não necessitarão ser enviados para o Controlador.

Uma rede OpenFlow pode possuir pontos de acesso sem fio, permitindo que clientes móveis utilizem sua infraestrutura para se conectarem a Servidores ou à Internet. Assim, mecanismos de *handoff* podem ser executados no Controlador realizando alteração dinâmica das tabelas de fluxo dos comutadores de acordo com a movimentação do cliente,

permitindo a redefinição da rota utilizada. A Figura 10 mostra a migração de fluxos (em amarelo pontilhado) acompanhando a mobilidade do usuário da esquerda para a direita.

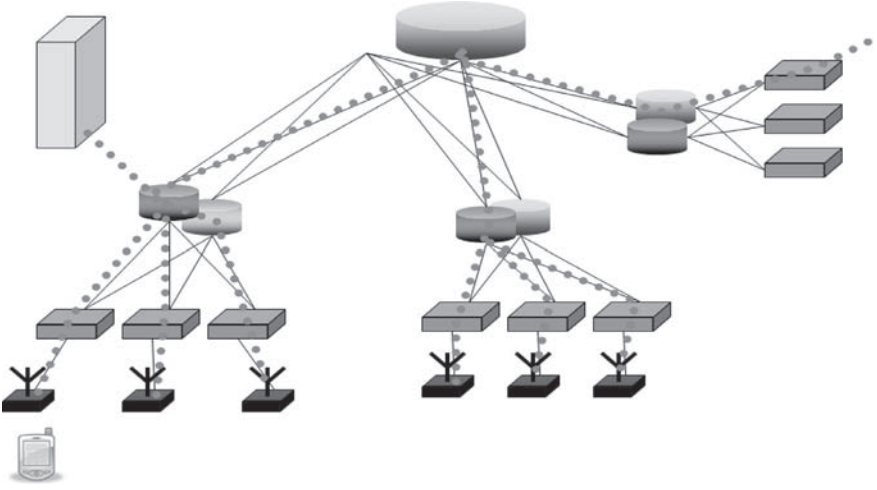


***Figura 10**: Exemplo de Mobilidade de Fluxos de Dados usando a Tecnologia OpenFlow.*

Podemos também visualizar um cenário em que desejamos mover um roteador para economizar energia ou para a manutenção do equipamento físico. Uma solução seria transferir todos os fluxos sendo roteados por este roteador para outro roteador e em seguida desligar este equipamento. Assim temos de forma virtual, uma mobilidade de roteadores alcançada através do uso da tecnologia OpenFlow.

Migração de máquinas virtuais

Uma das tecnologias-chave empregadas na Computação em Nuvens é a virtualização. Embora conhecida por muito tempo no âmbito de sistemas operacionais de *mainframe*, está sendo reavivada em ambientes de *Cloud Computing*. O motivo é simples: permitir um maior compartilhamento de recursos dividindo recursos físicos de CPU, memória, disco e de comunicação de maneira virtual.

A virtualização está sendo usada também por operadoras oferecendo *Clouds* na forma de infraestrutura. O software de virtualização é usado para compartilhar os recursos entre, por exemplo, operadores de distribuição de conteúdo (*Content Delivery Networks*).

Diferentemente de uma máquina física, o estado de uma máquina virtual (*Virtual Machine* – VM) pode ser descrito usando informação independente do hardware. Portanto, uma VM pode ser migrada para outra máquina física. Este tipo de operação pode ser aproveitada em diversos cenários. Por exemplo, à noite, quando o processamento diminui em alguns pontos da rede, as máquinas virtuais podem ser distribuídas entre estes pontos, desafogando pontos estressados. Em seguida, podemos desligar servidores ociosos e suas linhas de transmissão (caso possível), discos, etc. Assim é possível mudar a topologia da rede com seus servidores, o que pode favorecer a economia de energia, por exemplo.

A migração de máquinas virtuais tem sido uma alternativa comumente adotada em *clusters* e *datacenters* devido principalmente às suas vantagens no balanceamento de carga, tolerância à falha e economia de energia. Diferentes abordagens de migração de máquinas virtuais são disponibilizadas por *hypervisores*. No caso do Xen, estas são: *stop-and-copy* e pré-cópia.

Outras tecnologias com suporte à mobilidade

Neste texto, abordamos algumas tecnologias sendo usadas para prover mobilidade de recursos, usuários, informação, etc. Existem outras tecnologias desenvolvidas com o intuito de facilitar a mobilidade de usuários e seus recursos. Abaixo mencionamos algumas:

- OpenID: um sistema de suporte à autenticação única permitindo a portabilidade das credenciais de usuários para poder acessar os recursos de TIC em qualquer lugar federado à sua instituição. Este tipo de compartilhamento de controle de acesso está sendo usado em várias partes do mundo para o acesso de alunos e universitários às redes acadêmicas e de pesquisa. Evidentemente, a segurança neste caso deve ser dobrada já que a quebra de identidade significa o acesso em todos os serviços da federação OpenID.
- O uso de JavaCards e módulos de hardware seguro, como o Trusted Platform Module (TPM), permite a autenticação forte de usuários assim como seus equipamentos e controlar seu acesso a recursos de TIC. Passaportes e outros documentos importantes já estão incluindo chips seguros e TPMs permitindo uma troca de informação segura independentemente do site de acesso.

Bibliografia

[1] RFC 4423 – Host Identity Protocol (HIP) Architecture (early "informational" snapshot).

[2] A. T. Campbell, S. Kim, J. Gomez, C-Y. Wan, Z. Turanyi, A. Valko, "draft-ietf-mobileip-cellularip-00.tx", IETF mobile IP Working Group Document, December 1999.

[3] <http://www.ieee802.org/21>.

[4] KyoungSoo Park, Vivek S. Pai, Larry Peterson and Zhe Wang, *Improving DNS Performance* and. *Reliability via Cooperative Lookup*.

[5] <http://www.openflowswitch.org>.

Experimentação no futuro da Internet: pesquisa utilizando virtualização e *framework OpenFlow*

Fernando N. N. Farias[1,3], *João J. Salvatti*[2,3], *Antônio J. G. Abelém*[1,2,3]

Resumo

A Internet é um enorme sucesso mundial e vem mudando a forma como interagimos, trabalhamos e nos divertimos. Boa parte deste sucesso se deve à grande flexibilidade da tecnologia IP. Apesar de todo o sucesso da Internet, a tecnologia básica IP é a causa das suas próprias limitações, que se tornam cada vez mais evidentes. Um dos principais objetivos da atividade conhecida como Internet do Futuro (IF) é a formulação e avaliação de arquiteturas alternativas para substituir o protocolo IP. Nesse contexto, duas abordagens estão sendo discutidas e investigadas: a primeira denominada limpa (*Clean Slate*) que visa substituir a arquitetura atual por uma nova totalmente reconstruída e a outra chamada evolucionária (*Evolutionary*), que pretende evoluir a arquitetura atual sem perder a compatibilidade com a anterior. No entanto, o maior desafio agora é onde habilitar e testar as propostas das respectivas abordagens de modo a validá-las eficientemente, sem prejudicar a infraestrutura atual, já que haverá a necessidade de que roteadores e *switches* sejam reconfigurados, recursos sejam alocados e monitorados. Em suma, a rede deve ser a mais flexível possível, para que

[1] Instituto de Tecnologia – Programa de Pós-graduação em Engenharia Elétrica e de Computação – Universidade Federal do Pará (UFPA) 66.075-110 – Belém – PA – Brasil.

[2] Instituto de Ciências Exatas e Naturais – Programa de Pós-graduação em Ciência da Computação – Universidade Federal do Pará (UFPA) 66.075-110 – Belém – PA – Brasil.

[3] Grupo de pesquisa em Rede de Computadores e Comunicação Multimídia (GERCOM) – Universidade Federal do Pará (UFPA) – 66.075-110 – Belém – PA – Brasil.{fernnf,salvatti,abelem}@ufpa.br.

essa infraestrutura permita a coexistência de múltiplos modelos paralelos. Nesse sentido, a virtualização (de rede, dispositivos, enlace e etc.) e a solução OpenFlow vêm se tornando uma interessante alternativa para habilitar múltiplos experimentos com novas arquiteturas em um ambiente de produção. Este capítulo tem como principal objetivo apresentar os desafios e soluções para se criar um ambiente de testes para a Internet do Futuro utilizando técnicas de virtualização de redes e o *framework* OpenFlow.

1. Introdução

A Internet hoje está sendo usada para uma grande variedade de propósitos comerciais e não comerciais. Para muitas pessoas, é uma ferramenta de trabalho crucial, para outras, seu local de trabalho. Porém, para a grande maioria, é um eficiente meio de comunicação, entretenimento ou plataforma educacional. Ou seja, a Internet é um enorme sucesso mundial e vem mudando a forma como interagimos, trabalhamos e nos divertimos. Boa parte deste sucesso se deve à grande flexibilidade da tecnologia IP (*Internet Protocol*), que provê um mecanismo uniforme de transporte fim a fim, independente das tecnologias utilizadas nas camadas de mais baixo nível.

Apesar de todo o sucesso da Internet, a tecnologia básica IP é a causa das suas próprias limitações, que se tornam cada vez mais evidentes. A noção central de tamanho único, que requer tratamento idêntico para todos os fluxos de informação na Internet ao nível do pacote IP, não é desejável e nem necessariamente econômica, especialmente quando certas classes de aplicação, tais como de mídia interativa ou de acesso remoto a instrumentos científicos, requerem garantias de qualidade de serviço (*Quality of Service* – QoS) desnecessárias para a maioria de outras aplicações.

A atual arquitetura da Internet, inicialmente projetada há aproximadamente 30 anos, sofreu muitas extensões nos anos recentes para incluir novas funcionalidades, as quais não foram previstas no projeto inicial. Muitos especialistas de rede agora consideram que é necessário conduzir um estudo de arquiteturas alternativas para Internet do Futuro (IF), como uma maneira realmente eficiente de resolver muitos dos problemas prementes que atualmente afligem a Internet. Algumas das desvantagens da continuada persistência no uso da atual arquitetura incluem:

- exaustão iminente do espaço atualmente disponível de identificadores de *rede IP versão 4 (IPv4)*, causando uma "balcanização" da Internet, sem verdadeira conectividade global;

- custos elevados dos roteadores IP, restringindo o crescimento da rede, motivado principalmente pela natureza não escalonável das tabelas de roteamento, e dos requisitos de desempenho necessários para poder processar essas gigantescas tabelas na mesma velocidade que seus enlaces;
- investimentos imensos em medidas paliativas para conter problemas de segurança atualmente causados por *spam*, negação de serviço (*DoS – denial of service*) e crimes de roubo de informação;
- dificuldades de combinar transparência de acesso e desempenho de aplicação para usuários móveis.

Devido a isso, nos últimos anos, vem sendo aplicada à arquitetura da Internet uma série de modificações pontuais ou "remendos", para atender às limitações encontradas. Entre os "remendos", podem ser mencionados *Classless Internet Domain Routing (CIDR), Network Address Translation* (NAT), Serviços Integrados (*Intserv*), Serviços Diferenciados (*Diffserv*), IP Seguro, IP Móvel, *firewalls* e filtros de *spam;* estes são os mais conhecidos.

Por conta disso, há um entendimento crescente entre os pesquisadores em redes de computadores que as soluções para a maioria destes problemas dependerão de um redesenho fundamental da atual arquitetura da Internet, baseada em IP. E como aliada na busca dessas soluções, surge a atividade conhecida como Internet do Futuro, que inclui entre seus principais objetivos a formulação e avaliação de arquiteturas alternativas para substituir o IP, o qual é frequentemente ilustrado pelo conhecido modelo da ampulheta da Internet, apresentado na Figura 1.

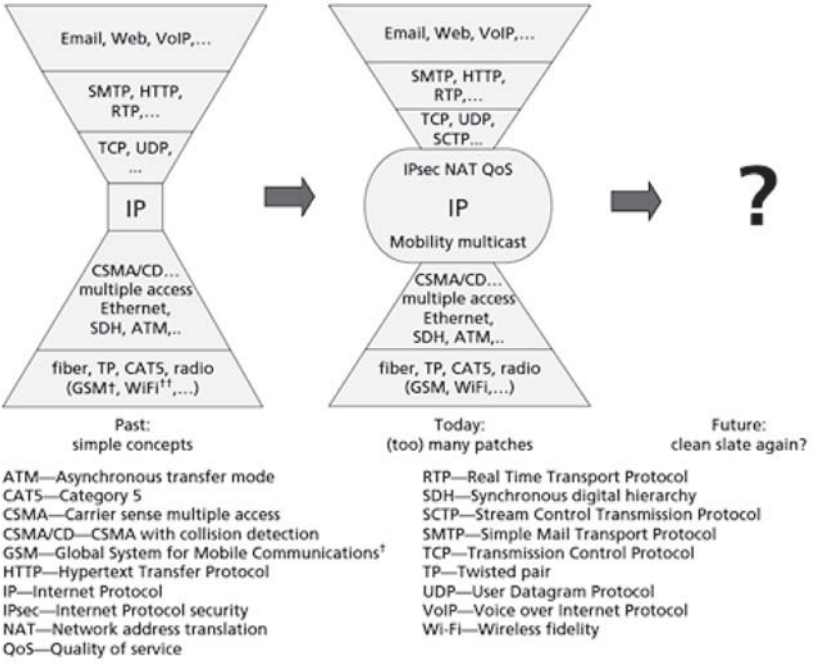


Figura 1.1: *A evolução do conceito da Internet [Banniza et. al., 2009]*

Nesse contexto, duas abordagens estão sendo discutidas e investigadas. A primeira denominada limpa (*Clean Slate*), que visa substituir a arquitetura atual por uma nova totalmente reconstruída. A outra, chamada evolucionária (*Evolutionary*), que pretende evoluir a arquitetura atual sem perder a compatibilidade com a anterior.

A proposta *Clean Slate*, que atualmente é um dos esforços para o desenvolvimento da Internet do Futuro, tem como principal objetivo direcionar como será efetuado o desenho da nova Internet [NICK, 2011].

Mas para que isso ocorra, a IF deve atender a um conjunto de requisitos importantes. Dentro desse contexto, David C. [2009] cita os caminhos e os requisitos fundamentais para o sucesso desta nova arquitetura, proposta que *Clean Slate* irá tomar para projetar a IF. Dentre esses requisitos, podemos destacar os principais desafios como: robustez e disponibilidade, suporte a nós móveis economicamente viável e rentável, gerenciabilidade com segurança e ser evolutiva.

As pesquisas baseadas em *Clean Slate* permitem explorar radicalmente uma nova arquitetura para Internet, mas isso não quer dizer que serão imediatamente aplicadas. Desse modo, algumas dessas soluções

podem ter ou não um caminho viável de implantação na Internet atual [ANJA F., 2007].

Por outro lado, tem-se a abordagem *Evolutionary* ou *Incremental,* que tem a missão, como pesquisadores da Internet, de primeiro medir e entender o estado corrente que se encontra a infraestrutura atual da Internet, para poder prever qual caminho ela seguirá e quais problemas enfrentará. Depois, com isso, criar o que pode ser referenciado como uma "inteligente mutação" [JENNIFER e CONSTATINE, 2010], ou seja, inovações que podem, primeiro, afastar ou resolver estes desafios e, segundo, que possam ser adaptadas para a corrente infraestrutura da Internet, de uma forma que seja compatível com as versões anteriores e possa ser incrementalmente expansível.

A adoção de qualquer uma das arquiteturas alternativas pode alterar a situação em que a Internet atual se encontra. Entretanto, um sério obstáculo para adoção efetiva de tais inovações tem sido a inabilidade de validá-las de maneira convincente. A redução no impacto do mundo real de qualquer inovação se deve à enorme base instalada de equipamentos e protocolos e à relutância em experimentar com tráfego de produção, o que tem criado uma barreira extremamente alta para a entrada de novas ideias. O resultado é que a maior parte das novas ideias da comunidade de pesquisa de rede não é testada, levando à crença comumente mantida de que a infraestrutura da Internet "ossificou" [PAUL, PAN e JAIN, 2011].

Tendo reconhecido o problema, a comunidade de pesquisa de rede está desenvolvendo soluções alternativas para pesquisa experimental em IF. Suas atividades inicias, usando redes para experimentação (*testbed*), começaram nos EUA com os programas *Global Environment for Network Innovations (GENI)[GENI, 2011]* e *Future Internet Design (FIND)[FIND, 2011]*, cujas principais propostas são testar e amadurecer um grande conjunto de pesquisas em comunicação de dados e sistemas distribuídos.

Além disso, ainda há o FIRE na UE (União Europeia) [FIRE, 2011] que é uma iniciativa de ambiente de pesquisa para experimentação e validação de ideias inovadoras e revolucionárias para novos paradigmas de redes e serviços. Nesta mesma linha, temos o projeto *AKARI* no Japão [AKARI, 2011], um programa de pesquisa que tem como objetivo implementar a nova geração de redes para 2015, baseada na proposta *Clean Slate.* E por fim, iniciativas semelhantes também foram lançadas mais recentemente em outras partes do mundo [CFI, 2011][JUN BI, 2011].

Neste sentido, redes de experimentação programáveis e virtualizadas vêm sendo utilizadas por esses projetos para diminuir a barreira de custo à entrada de novas ideias, aumentando a taxa de inovação

da infraestrutura de rede. Com isso, são oferecidas infraestruturas capazes de habilitar, controlar e testar as respectivas abordagens, de modo a validá-las eficientemente, sem prejudicar a infraestrutura atual.

No contexto de redes programáveis, o *framework* OpenFlow é a ferramenta que vem oferecendo a pesquisadores a possibilidade de testar seus protocolos experimentais em redes de produção como: redes de um campus, redes metropolitanas ou em um *backbone* de rede de ensino e pesquisa. O OpenFlow, além de oferecer o protocolo de controle, chamado de OpenFlow *protocol* para manipular a tabela de encaminhamento dos roteadores e switches, também oferece uma API (*Application Programming Interface*) simples e extensível para programar o comportamento dos fluxos de pacotes. Desta forma que, através desta API, pesquisadores podem rapidamente construir novos protocolos e aplicá-los em um ambiente de produção sem prejudicá-lo [MCKEOWN, 2008].

A virtualização de computadores tem sido usada há muito tempo e hoje está amplamente disponível em plataformas comuns. É realizada por meio do compartilhamento de processadores e dispositivos de E/S, utilizando técnicas de fatiamento de tempo e memória virtual. A virtualização de redes é mais recente e é conseguida pelo uso de switches e roteadores virtuais e a multiplexação de enlaces [CHOUDHURY, 2010].

Observando isso, o uso das técnicas de virtualização de redes (dispositivos, enlace, etc.) em conjunto com a solução OpenFlow vem se tornando uma excelente alternativa para habilitar múltiplos experimentos com novas arquiteturas em um ambiente de produção. Uma rede OpenFlow permite que se possa programar o comportamento da rede, além de criar ambientes virtuais de rede chamados de "fatias" (redes virtuais com nós, enlaces e recursos) para testar novas ideias e modelos de protocolos para IF.

Este capítulo tem como principal objetivo apresentar as iniciativas para o desenvolvimento da IF, seus requisitos necessários e o uso de virtualização e OpenFlow para se criar um ambiente de *testbed* para experimentação e desenvolvimento de protocolos na abordagem de IF.

Além desta seção introdutória, o capítulo está dividido da seguinte maneira:

- Seção 2 – são apresentadas as recentes experiências realizadas para Internet do Futuro, tanto no Brasil como em *backbones* da Europa e da América do Norte;
- Seção 3 – é detalhado o *framework OpenFlow*, incluindo os principais aspectos sobre virtualização;
- Seção 4 – são descritos os requisitos para construção de um *testbed* experimental;

- Seção 5 – tem-se um estudo de caso de criação de vários *slices* de redes para testes de protocolos;
- Seção 6 – são apresentadas as considerações finais e tendências futuras em relação à pesquisa experimental e Internet do Futuro.

2. Pesquisa experimental em Internet do Futuro no Brasil e no mundo

Atualmente, na comunidade acadêmica, existe um amplo consenso de que a Internet atual sofre várias limitações, relacionadas à escalabilidade, a suporte a redes móveis de vários tipos (*ad-hoc*, *multi-hop* e *mesh*), à mobilidade, ao consumo de energia, à transparência, à segurança e a outros [RAJ JAIN, 2006].

A partir disso, nesta seção, são levantados e comentados os principais desafios que estão relacionados à Internet do Futuro. Como também, são observados os projetos a respeito de investigação da IF como o caso do GENI (*Global Environment for Network Innovation*), financiado pela NSF (*National Science Foundation*), FIRE (Future Internet Research and Experimentation), pela União Europeia e FIBRE (Future Internet Experimentation between Brazil and Europe) no Brasil.

2.1. Desafios relacionados à Internet do Futuro

A ideia básica da arquitetura da Internet foi desenvolvida há mais de trinta anos. Neste tempo, tem-se aprendido bastante sobre redes de computadores e encaminhamento de pacotes. Também, durante este mesmo período, a Internet vem sofrendo contínuas adaptações, causadas pelo seu rápido crescimento e pela quantidade de usuários e de aplicações habilitadas sobre sua arquitetura. Essas adaptações ou "remendos" demonstram que o projeto inicial já não se ajusta às necessidades atuais na rede. Além disso, a arquitetura atual da Internet já apresenta inúmeros problemas ainda não solucionados, impedindo o atendimento dos requisitos destas novas aplicações e serviços.

Muitos especialistas em rede de computadores chegaram a um consenso de que agora consideram extremamente necessário conduzir um estudo de arquiteturas alternativas para Internet do Futuro como uma maneira realmente eficiente de resolver muitos dos problemas prementes que atualmente afligem a Internet. A seguir, são apresentados alguns dos principais desafios, que segundo Subharthi e Jianli e Raj [2011], a nova arquitetura da Internet deve atender.

2.1.1. Suporte a Novas Tecnologias de Rede

A Internet atual é projetada para tirar vantagem de uma grande gama de tecnologias de rede. No entanto, hoje há vários desafios no horizonte, entre elas as tecnologias sem fio e as novas soluções ópticas.

As redes sem fio abrangem uma série de tecnologias que vai do *wi-fi* de hoje até as *ultra-widebands* e redes de sensores sem fio de amanhã. Considera-se que as redes sem fio são talvez umas das maiores tecnologias de rede e com granularidade maior ou igual às redes locais (LAN). No entanto, uma consequência das redes sem fio é a mobilidade. Hoje, a mobilidade está à "borda" da rede, mas os mecanismos para torná-la mais eficiente não funcionam de maneira satisfatória, principalmente nas questões relacionadas à eficiência energética, à mudança de ponto de acesso e às aplicações. Portanto, identificam-se os seguintes desafios para suporte às tecnologias sem fio:

- A Internet do Futuro deve suportar a mobilidade como o objetivo primordial em sua construção. Os nós devem ser capazes de modificar seu ponto de acesso na Internet e, mesmo assim, continuarem sendo alcançáveis.
- A Internet do Futuro deve oferecer meios adequados para descobrir as características dos mais variados tipos de enlace de rede sem fio e se adaptar a eles, de maneira a prover as eficiências energéticas destes nós.
- A Internet do Futuro deve facilitar o processo pelo qual os nós móveis, fisicamente próximos, descubram-se uns aos outros.

Outra tecnologia revolucionária é a rede de transporte óptica. A comunidade de pesquisadores em redes ópticas está desenvolvendo maneiras de como usar as novas tecnologias como *switches* ópticos e elementos lógicos para encaminhar dados em desempenhos maiores que as redes puramente eletrônicas. Isso envolve mudanças em vários níveis de redes em anéis para redes em malha, de redes com comprimentos de ondas fixamente alocados para transmissores e receptores dinamicamente alocados, de redes sem *buffers* ópticos para redes com planos de controles inteligentes e com *buffers* ópticos suficientes e de redes ópticas que utilizam fixas larguras de bandas para as que permitem que a largura de banda seja dinamicamente alocada, acessadas e utilizadas. Logo se identificam os desafios para Internet do Futuro explorar a emergente capacidade óptica:

- A Internet do Futuro deve ser projetada para permitir aos usuários a utilização dessas novas capacidades de transporte ópticos, incluindo uma melhor confiabilidade através de diagnósticos entre camadas.
- A Internet do Futuro deve permitir que os nós que são dinamicamente configuráveis habilitem a camada eletrônica para o acesso dinâmico de usuários.
- A Internet do Futuro deve incluir softwares de controle e gerenciamento que permitam à rede formada por nós dinamicamente reconfiguráveis ser configurada por aplicações que necessitem de uma grande quantidade de recursos de rede, tal como largura de banda.

2.1.2. Segurança e robustez

Talvez a razão que mais estimulou a comunidade acadêmica a pensar em um redesenho da Internet foi ter uma rede com grandes avanços na segurança e robustez. Na Internet atual, não há nenhuma abordagem realmente abrangente para tratar questões de segurança. Ela possui muitos mecanismos, mas não uma "arquitetura segura" e muito menos um conjunto de regras determinando como esses mecanismos devem ser combinados para se obter uma boa segurança de maneira geral. A segurança em redes e principalmente a da Internet, atualmente se assemelha a um conjunto crescente de remendos, extremamente suscetível a falhas.

Para a construção de uma rede mais robusta e segura, identificam-se os seguintes desafios:

- Qualquer conjunto de *hosts* deve ser capaz de se comunicar entre si com alta confiabilidade e previsibilidade, e nós maliciosos ou defeituosos não podem ser capazes de interromper esta comunicação. Além disso, os usuários devem esperar por um nível de disponibilidade correspondente ou que exceda o sistema de telefonia de hoje.
- A segurança e robustez devem ser estendidas através de suas camadas, pois a segurança e a confiabilidade de um usuário final dependem da robustez, tanto nas camadas de comunicação quanto nas aplicações distribuídas.

2.1.3. Eficiência energética na comunicação

Atualmente, há uma preocupação mundial [WILLIAMS e CURTIS, 2008] por desenvolvimento de tecnologias que diminuam o consumo de energia, principalmente por redes, como as de sensores sem fio e as de dispositivos móveis *wi-fi*, em que recursos energéticos são fatores relevantes em sua comunicação. Com as atuais tecnologias, comunicando um bit sobre um meio sem fio, em intervalos curtos, consomem mais energia do que o seu processamento, e não há tendência de mudanças em um futuro próximo [Chen, 2006]. No entanto, a Internet atual não possui mecanismos que levem em consideração a comunicação com requisitos de eficiência de energia, de modo a adaptar o seu consumo mediante a observação do meio. Ou ainda, que adapte o uso de energia para oferecer uma comunicação eficiente, na transmissão de dados e no gerenciamento de rede, a baixos níveis energéticos.

Portanto, como desafio para o projeto da Internet do Futuro:

- Deve-se prover meios para uma eficiência energética generalizada tanto para dispositivos sem fio quanto os com fio;
- Desenvolver tecnologias com foco no uso eficiente da energia elétrica;
- Incluir em sua arquitetura mecanismos que ofereçam uma comunicação eficiente tanto na transmissão de dados como no gerenciamento de rede;
- Prover uma arquitetura para Internet do Futuro energeticamente otimizada.

2.1.4. Separação de Identidade e Endereço

Na Internet atual, um sistema é identificado pelo seu endereço IP. Como resultado, quando o sistema muda seu ponto de localização da interligação, seu endereçamento também muda. Ou seja, o endereço IP, neste caso, ao mesmo tempo, tem o papel de identificador e localizador. Devido a isso, os nós móveis se tornaram um desafio na Internet atual. Por essas características, os sistemas móveis têm dificuldades de serem atingidos.

Este é um problema bem conhecido na Internet e há um número considerável de tentativas e propostas para resolver esse problema, incluindo: *mobile IP*, *Internet indirection infraestructure* [Stoica et. al, 2002],

host identify protocol [MOSKOWITZ e NIKANDER, 2005] [MOSKOWITZ e NIKANDER e JOKELA, 2005] e outros [BALAKRISHNAN, 2004].

Assim, para a Internet do Futuro existem os seguintes desafios:

- Aplicar soluções que permitam a localização global de um determinado *host*, através da utilização de um sistema de endereçamento global;
- Definir novas formas de localização e identificação, além do desenvolvimento de endereçamentos coesos às necessidades da rede.

2.1.5. Qualidade de Serviço e Qualidade de Experiência

A natureza do IP, não orientado a conexão, dificulta qualquer aplicação de garantia de QoS (*Quality of Service*). Além disso, a característica de encaminhamento de pacote baseado em melhor esforço faz com que qualquer abordagem relacionada a questões de reservar recursos ou prioridade interfira no funcionamento estabelecido para Internet atual.

Desta maneira, embora a qualidade de serviço tenha sido extremamente pesquisada na comunidade acadêmica, ainda não há um modelo claro de como diferentes níveis de qualidade devem ser aplicados e de que forma ele se integrará à arquitetura de rede atual.

Desta forma, são desafios para a arquitetura da Internet do Futuro:

- Permitir uma variedade de garantias levando em consideração tanto a qualidade da aplicação na rede como a qualidade de experiência do usuário;
- Novos métodos de encaminhamentos de pacote baseados nas características das aplicações, principalmente as de tempo real.

2.1.6. Separação entre os planos de controle, gerenciamento e dados

Na Internet atual, os planos de controle, gerenciamento e dados são unificados. Ou seja, mensagens de controle, por exemplo, mensagens de conexão TCP (*Transmission Control Protocol*) ou mensagens de gerenciamento SNMP (*Simple Network Management Protocol*) seguem no mesmo canal em que trafegam os dados. Como consequência, isso possibilita um significante risco de segurança dos dados de controle, além do desperdício de recursos da rede.

Uma vantagem desta separação de planos é a adoção de novas tecnologias de plano de dados pela arquitetura de rede como: um comprimento de onda; *frame* SONET; ou uma linha de transmissão de energia (*Power Line Comunication* – PLC). Portanto, esta separação entre os planos deve ser parte integral desta próxima geração da arquitetura da Internet.

2.1.7. Isolamento

Para algumas aplicações críticas, o usuário demanda isolamento em um ambiente compartilhado como na Internet atual. Isolamento significa garantir que o desempenho desta aplicação crítica não seja afetado por outras aplicações que queiram compartilhar o mesmo recurso.

Com o uso da virtualização, o isolamento se tornou um fator ainda mais importante, principalmente para virtualização de redes, onde um roteador ou uma infraestrutura de rede é totalmente virtualizada, de forma que o encaminhamento de pacotes não pode, de maneira alguma, ser interferido ou interferir, em outras infraestruturas.

Assim, a próxima geração da Internet deve prover de modo eficiente um mix programável em sua arquitetura de isolamento e compartilhamento, às suas aplicações e serviços.

2.1.8. Mobilidade

Atualmente vem se observando uma crescente e diversificada quantidade de serviços na Internet, impulsionada principalmente pelo aumento do acesso de dispositivos sem fio. Com isso, amplia-se a necessidade de mobilidade pelos usuários.

Além disso, a forma de comunicação estabelecida originalmente pela arquitetura da Internet atual como conexão ponto a ponto e entrega imediata, faz com que esses tipos de usuários não sejam atendidos satisfatoriamente. Principalmente, por causa de uma questão comum relacionada à mobilidade, que são os *handovers*, criados com o objetivo de evitar que os nós móveis tenham a perda das suas conexões ativas. Os protocolos da Internet atual não preveem o tratamento desse comportamento. E ainda, protocolos como TCP também são prejudicados por não prever esse tipo de usuário.

Com isso, a Internet do Futuro enfrenta o grande desafio de como permitir a movimentação do nó entre diferentes pontos de acesso sem que as conexões ativas sejam perdidas.

2.1.9. Escalabilidade

Com o aumento exponencial do número de estações conectadas à Internet, alguns componentes da arquitetura atual têm sofrido com problemas de escalabilidade. Esse é o caso do sistema de roteamento, que sofre com o aumento e as atualizações frequentes de suas tabelas. Ainda há a questão do endereçamento que não é capaz de suportar todos os elementos conectados à rede como é o caso do IPv4.

Portanto, a Internet do Futuro terá o desafio de manter o sistema global de roteamento escalável, além de novos protocolos que mantenham essa escalabilidade, mesmo com o aumento do espaço de endereçamento. E também novas formas de endereçamento para evitar a escassez do recurso.

2.2. Iniciativas para investigação em Internet do Futuro

A comunidade acadêmica, observando os problemas atuais da Internet, começou uma corrida para solucionar os desafios da Internet do Futuro e, com isso, modelar a arquitetura que conduzirá a próxima geração da Internet.

No entanto, estas novas propostas e especulações teóricas na direção destas soluções devem ser suportadas por uma infraestrutura real de rede experimental e testadas em ambientes de larga escala. Estas instalações experimentais desempenham o papel de rede de teste ou *testbed*, conduzindo experimentos para a prova de conceitos de novas arquiteturas, protocolos, tecnologias e serviços.

Além disso, uma coexistência com a rede de tráfego de produção é essencial para observação e captura de certos aspectos e fenômenos perceptíveis apenas em instalações operacionais e assim permitir que sejam avaliados os seus impactos sobre a sociedade e a economia.

Observando essas necessidades, algumas iniciativas em pesquisa da Internet do Futuro começaram a surgir, principalmente nos Estados Unidos, Europa e no Brasil, conforme a seguir.

2.2.1. GENI (Global Environment for Network Inovation)

A *Global Environment for Network Inovation* (GENI) é a principal iniciativa americana em investigação e experimentação em Internet do Futuro. O GENI é um conjunto de infraestruturas de redes de pesquisa,

dos mais variados modelos tais como: sem fio, óptico e elétrico. Patrocinada pela *National Science Foundation* (NSF), atualmente está em fase de desenvolvimento e prototipagem.

O objetivo do projeto GENI é montar um grande laboratório virtual em larga escala para experimentações em rede de computadores, cuja maior importância é prever e criar novas possibilidades para Internet do Futuro.

Para o GENI, os conceitos principais relacionados à experimentação em Internet do Futuro e que fazem parte do projeto de sua arquitetura são [GENI Sys, 2008]:

- **Programabilidade**: Os pesquisadores poderão fazer *upload* de softwares aos nós do GENI para programar ou controlar o seu comportamento.
- **Virtualização e outras formas de compartilhamento de recursos**: Qualquer que seja a implementação de máquina virtual sobre um nó GENI, será permitido que múltiplos pesquisadores simultaneamente compartilhem a mesma infraestrutura. Cada experimento terá à sua disposição uma fatia ou *slice* isolado, com recursos fim a fim alocados dentro da infraestrutura do GENI.
- **Federação**: O GENI será composto de infraestruturas próprias e de outras de apoio, operadas por organizações parceiras ao projeto, criando o conceito de federações de recursos e nós, que na visão de um pesquisador, comporta-se como se fosse única infraestrutura.
- **Experimentos baseados em fatias (*Slices*):** Os experimentos no GENI serão interconectados a um conjunto de recursos reservados sobre uma plataforma em diversas localizações, dando a ideia de uma fatia ou *slice* de recursos. Dentro dessa fatia o pesquisador poderá remotamente descobrir, reservar, configurar, "debugar", operar e gerenciar seus experimentos e recursos disponíveis.

Para se ter uma ideia de como o GENI irá funcionar após a sua concepção, a Figura 2.1 ilustra o fluxo que um pesquisador percorrerá para habilitar seu experimento dentro da infraestrutura do GENI.

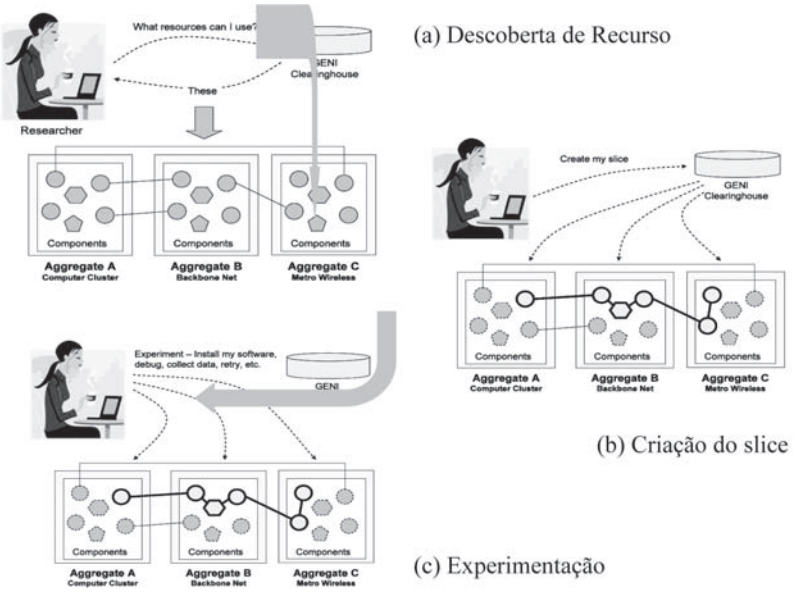


***Figura 2.1** – A criação de um slice sobre a infraestrutura do GENI [GENI Sys, 2008]*

Na Figura 2.1, é observado o processo que um pesquisador irá executar para solicitar e utilizar um *slice* para experimentação de seus modelos ou protocolos na arquitetura do GENI. Esse processo é formado por três estágios intermediários: descoberta de recurso, criação do *slice* e experimentação.

Na descoberta de recursos *(DR)*, Figura 2.1 (a), o pesquisador terá uma visão global dos recursos disponíveis para o seu experimento. A partir disso, poderá definir, de maneira simples, quais os recursos do GENI serão utilizados em seus experimentos.

Na criação do *slice* (CS), Figura 2.1 (b), o pesquisador recebe um *slice* ou um contêiner vazio onde serão instanciados os softwares e os recursos que foram definidos pelo DR. Neste caso, um *slice* é uma integração de dois agregados: um *cluster* de computadores e uma rede ou um conjunto de redes.

Por fim, na experimentação, Figura 2.1 (c), o pesquisador poderá instalar os seus softwares, executar, parar e coletar resultados do experimento.

O GENI é o conjunto de grandes *testbeds* com várias alternativas para experimentações. Dentre estes *testbeds*, podemos destacar: PlanetLab, EmuLab e ORBIT [Spiral2, 2010] [RAJ JAIN et. al, 2011].

PlanetLab

O PlanetLab [Peterson, 2006] é um consórcio formado por instituições acadêmicas, governamentais e industriais que visa à criação de uma grande malha de computadores espalhados pelo mundo em diversas redes. Essa malha serve como um grande laboratório, para testes e desenvolvimento de aplicações distribuídas.

A significância e abrangência do projeto permite o compartilhamento de aplicações em larga escala e uma grande integração entre as instituições de pesquisa participantes. Hoje, ele possui mais de 700 máquinas espalhadas em 350 locais diferentes, totalizando 25 países e mais de 275 projetos ativos. Na Figura 2.2, temos um mapa ilustrando os principais sites do PlanetLab.

***Figura 2.2** – Sites PlanetLab ao redor do mundo*

No projeto GENI, o PlanetLab [SPIRAL2, 2010], será uma alternativa de *testbed* aos seus pesquisadores. Atualmente, é dirigido pela Universidade de Princeton e tem como escopo integrar logicamente os componentes GENI aos serviços PlanetLab como: PLC (*PlanetLab Central Sofware*), responsável por criar a rede sobreposta definindo a topologia virtual que será usada para experimento; e o SFA (*Slice-Based Facility Architecture)*, responsável por localizar e alocar os recursos para os experimentos [PETERSON et. al, 2009].

ORBIT

O projeto ORBIT provê um flexível *testbed* sem fio, aberto à comunidade acadêmica. Ele foi desenvolvido para que pesquisadores entendessem as limitações do mundo real das redes sem fio, que muitas vezes não são percebidas em simulações por causa das simplificações aplicadas ao modelo ou por terem sido aplicadas em uma quantidade pequena de nós.

O ORBIT possui dois *testbeds*: o primeiro é *indoor*, constituído de 400 nós dispostos em uma grade 20x20 (ilustrado na Figura 2.3), separados por um metro de distância entre os nós adjacentes, onde cada nó é composto por uma plataforma PC com múltiplas interfaces sem fio e com fio. O segundo *testbed* está localizado no campus da Universidade de *Rutges*, dentro de uma área de 1,5 hectares quadrados. Entre eles existem nós fixos, e também nós móveis para prover mobilidade entre os roteadores.

Figura 2.3 – Testbed Indoor *do ORBIT*

No GENI, sua integração permitirá que pesquisadores gerenciem esse *testbed* de redes sem fio dentro do GENI, além de estender a capilaridade de nós do ORBIT para outros *testbed* sem fio. No GENI, o ORBIT será integrado dentro do ambiente OMF (*cOntrol and Management Framework*) [OMF, 2011].

EmuLab

EmuLab [Eide et. al, 2006] é um *testbed* em redes de computadores que oferece aos seus pesquisadores um *range* de ambientes aos quais eles podem desenvolver, analisar e avaliar seus sistemas. O nome EmuLab refere-se tanto ao *testbed* quanto ao software de interação do usuário ao *testbed*.

O projeto é gerenciado pela Universidade Utah, onde foram desenvolvidos os primeiros nós do *testbed*. Atualmente, há mais de doze países utilizando o EmuLab, totalizando um *testbed* com mais de cem nós.

Na Figura 2.4, temos os *clusters* de computadores utilizados pelo EmuLab. Os ambientes disponíveis no EmuLab são: Emulação (*Emulation*), neste ambiente o pesquisador define uma topologia arbitrária com *switches* ou roteadores e nós PC; *Live-Internet*, o Emulab oferece um ambiente federado para experimentos sobre Internet; No 802.11 Wireless, provê um ambiente com ponto de acesso, roteadores e cliente para experimentos em rede sem fio; e o ambiente *Software-Defined Radio*, o qual permite experimentações em camada 1 para análise de processamento de sinal.

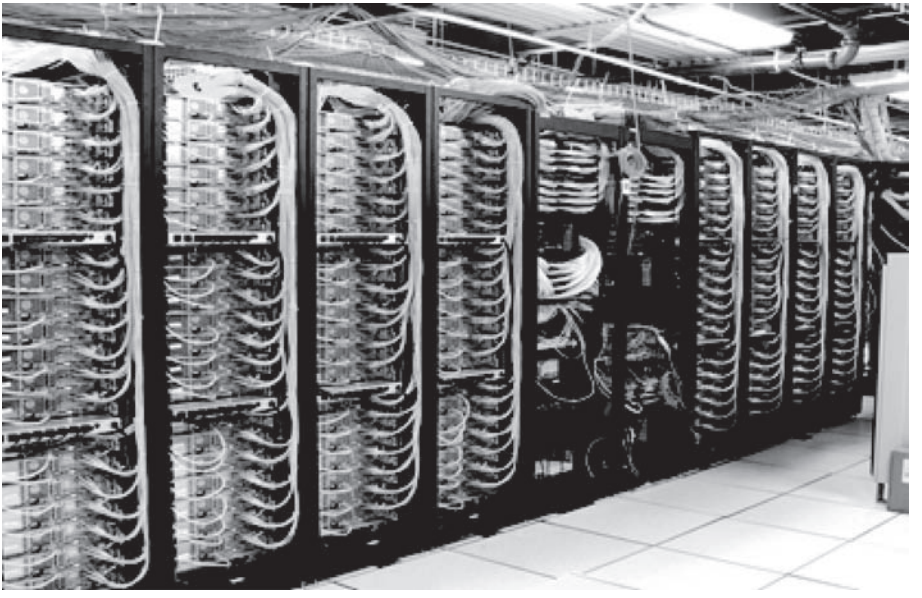


Figura 2.4 – Cluster *EmuLab*

No GENI, a integração do EmuLab ao projeto é conhecida de ProtoGENI [ProtoGENI, 2011], e permitirá ao EmuLab expandir ainda

mais seu *testbed*, além de controlar novos não oferecidos atualmente, por exemplo à infraestrutura de altíssima velocidade dos *backbones* da internet2.

2.2.2. Future Internet Research and Experimentation (FIRE)

A iniciativa FIRE (*Future Internet Research Experimentation*) na Europa visa à pesquisa experimental e ao financiamento de projetos que produzam infraestruturas para experimentação em Internet do Futuro. A meta é que as pesquisas em tecnologias para Internet do Futuro sejam direcionadas à rede ou a serviço e tenham a possibilidade de comparar as soluções correntes com as propostas futuras. Desta forma, afirma-se que o FIRE possui duas dimensões relacionadas que direcionam suas pesquisas e seus investimentos [FIRE, 2008]:

- Pesquisa Experimental: o objetivo é integrar a pesquisa multidisciplinar e a experimentação em larga escala. A partir daí, definir uma metodologia que direcionará pesquisa experimental na infraestrutura FIRE, baseada em um ciclo interativo que vai da pesquisa, passando pelo projeto e chegando à experimentação.
- Facilidades para Testes: o objetivo é oferecer um ambiente de testes, *testbed,* interconectados, suportando várias tecnologias e envolvendo uma granularidade de *testbeds* federados, de modo que ele seja sustentável, dinâmico e integrado em larga escala. E ainda, que facilite a pesquisa experimental na comunidade acadêmica, centros de pesquisa e indústria.

Para o FIRE, as experiências práticas são extremamente necessárias para dar credibilidade e levantar o nível de confiança na conclusão da pesquisa. Além disso, a experimentação deve ser executada em larga escala para que seja representativa, convincente, e para provar a escalabilidade da solução testada. Os projetos de *testbed* em destaque financiados pela iniciativa FIRE incluem [FIRE, 2010]: BonFIRE, CREW e OFELIA.

BonFIRE

A iniciativa BonFIRE (*Building Services Testbeds on FIRE*) [BonFIRE, 2011] é um projeto baseado em nuvens *multi-site* para o suporte à pesquisa de aplicações, serviços e sistemas, visando a "Internet de Serviços" dentro

da Internet do Futuro. O BonFIRE objetiva oferecer aos pesquisadores acesso a um ambiente experimental em larga escala para experimentação de seus sistemas e aplicações, avaliações do efeito "cross-cutting" da convergência dos serviços, infraestrutura de rede e a análise do impacto socioeconômico.

O BonFIRE irá operar uma nuvem baseada em IaaS (*Infrastructure as a Service*) com políticas e melhores práticas de experimentações. Ele irá se adaptar a uma abordagem multiplataforma federada provendo interconexão e interoperação entre os serviços e a rede no *testbed*. A plataforma irá oferecer serviços avançados e ferramentas para pesquisa de novos serviços, incluindo nuvens de federações, gerenciamento de máquinas virtuais, modelagem, gerenciamento do ciclo de vida a experimentação, monitoramento da qualidade de serviço e análise de desempenho. A Figura 2.5 ilustra o ambiente BonFIRE.

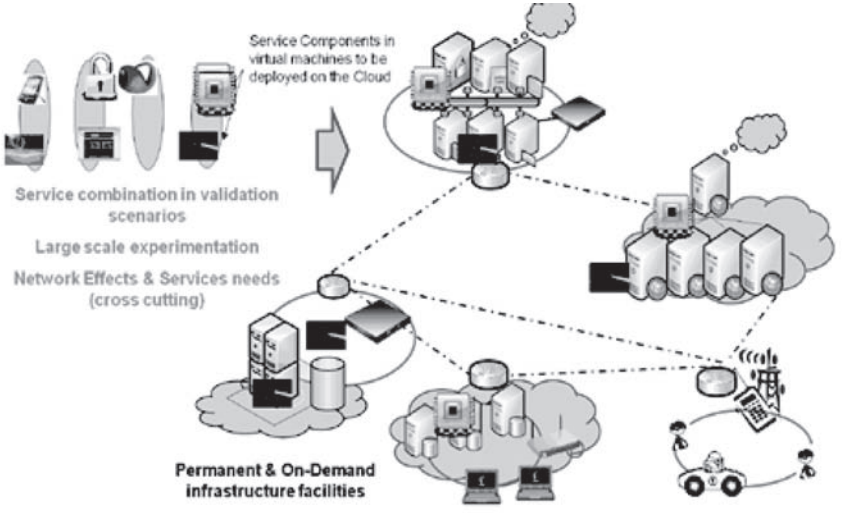


***Figura* 2.5** – *Infraestrutura disponível no BonFIRE [FIRE, 2010]*

CREW

O principal objetivo do CREW (*Cognitive Radio Experimentation World*) [CREW, 2011] é estabelecer uma plataforma de teste aberta e federada que facilite o avanço em pesquisa em avançados no sensoriamento de espectros, rádios cognitivos e estratégias de redes cognitivas em visões horizontais e verticais do espectro compartilhado em bandas licenciadas e não licenciadas.

O CREW incorpora quatros *testbeds* individuais sem fio utilizando um *ranger* de tecnologias sem fio como: heterogêneos ISM, Celular e Sensores sem fio, com o estado da arte de plataforma de sensoriamento cognitivo.

O CREW irá fisicamente e virtualmente federar componentes interligando entidades de software e hardware de diferentes padrões usando APIs padronizadas. Além disso, o CREW estabelecerá um *benchmark framework*, habilitará experimentos controlados, metodologias para análise automática de *performance* e reproduzirá condições para teste.

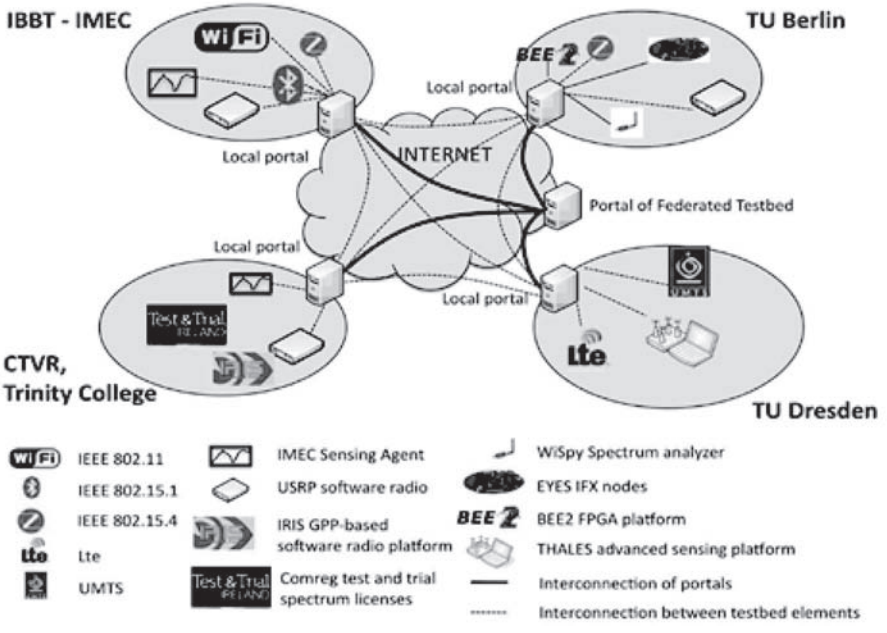


Figura 2.6 – *Ambiente federado CREW [FIRE, 2010]*

OFELIA

O projeto OFELIA [Ofelia, 2011] visa criar um ambiente experimental, *testbed*, que permita a seus pesquisadores não somente experimentação, mas também o controle a redes a sua maneira de forma precisa e dinâmica. Para alcançar isso, o *testbed* OFELIA é baseado em OpenFlow, atualmente uma emergente tecnologia de rede que permite virtualizar e controlar ambiente de rede, através de interfaces seguras e padronizadas. O OFELIA proverá

equipamentos OpenFlow de alto desempenho para habilitar experimentos em alta escala.

A infraestrutura federada pelo OFELIA, inclui "ilhas" OpenFlow na Bélgica, Alemanha, Espanha, Suíça e Reino Unido. Essas ilhas reúnem uma diversidade de infraestruturas baseadas em OpenFlow que permitirá experimentações em redes multicamadas e multitecnologias. Na Figura 2.7, é apresentado como a estrutura dessas ilhas será instalada em cada um desses países.

O OFELIA permitirá o teste de novos algoritmos de roteamento, tunelamento, protocolos e planos de controle. Além disso, novas aplicações poderão ser colocadas no controlador OpenFlow a qualquer momento na infraestrutura. Também permitirá serem investigadas novas estruturas e formatos de endereçamento e modelos de encaminhamentos.

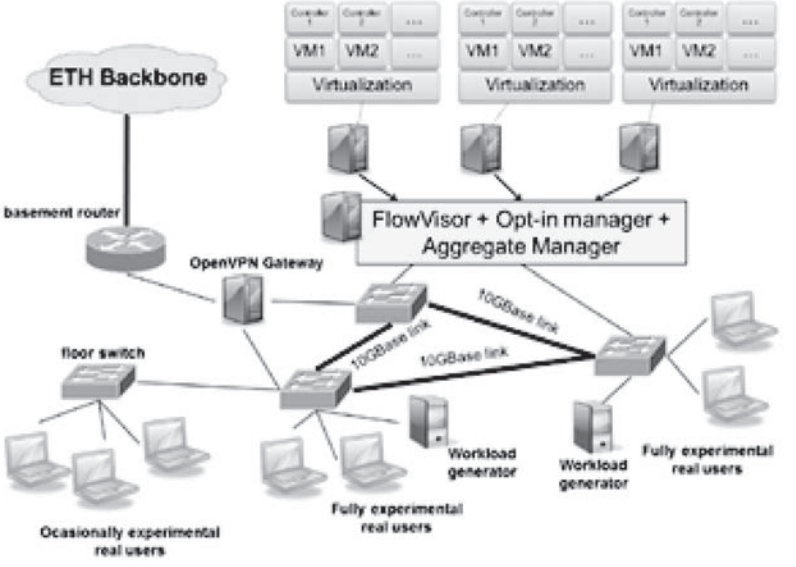


Figura 2.7 – *Estrutura de uma "ilha", baseada em OpenFlow no projeto OFELIA [Ofelia, 2011]*

2.2.3. Iniciativas Brasileiras

Os parceiros brasileiros contribuem no projeto com a experiência na implantação de instalações de experimentação locais e na participação em diferentes projetos de pesquisa experimental em Internet do Futuro,

embora com pouca ou nenhuma coalizão estratégica entre elas. O primeiro desses projetos a se destacar é o projeto de pesquisa e desenvolvimento GIGA e suas instalações experimentais em grande escala conhecido como rede GIGA, coordenado conjuntamente pelo CPqD e a RNP [SCARABUCCI, 2005].

O projeto GIGA atualmente se concentra em redes ópticas e definidas por software. Deverá ser atualizado em breve com comutadores OpenFlow de 10G (o primeiro na América do Sul) e uma solução aberta ("open-source") de roteamento de pacotes (o primeiro do mundo e neste momento disponível para parceiros selecionados nos EUA e no Brasil) que roda sobre o NOX para controlar o encaminhamento de pacotes em redes com tecnologia OpenFlow habilitada.

A rede GIGA conecta mais de 66 laboratórios de 23 instituições da Região Sudeste do Brasil e está conectada com a rede nacional de ensino e pesquisa (Rede Ipê). Através desta, interliga-se com redes de pesquisa e ensino em todo o mundo. A rede GIGA mantém um nó GENI (ou seja, servidores) no CPqD.

O segundo projeto de destaque é o projeto Web Science [WEBSCIENCE, 2010], apoiado pelo CNPq. O projeto Web Science começou efetivamente em 2010 e o subprojeto de Arquiteturas para a Internet do Futuro envolve a RNP e quatro Universidades parceiras com experiência em redes ópticas e sem fio, estudos de simulação e emulação e monitoramento de rede. Um dos primeiros objetivos do projeto é estabelecer ilhas experimentais nos laboratórios de cada parceiro e interligá-las em camada 2 através das redes de Ipê e GIGA.

FIBRE

Coordenado pela Universidade Federal do Pará, através do Grupo de Pesquisa em Redes de Computadores e Comunicação Multimídia (GERCOM) [22], o projeto FIBRE, aprovado no edital de Chamadas Coordenadas Brasil-União Europeia em TIC do CNPq e da CE no seu 7º Programa Quadro (FP7), representa a primeira colaboração direta em IF entre parceiros brasileiros e europeus, mas ambos os lados já têm entendimentos e colaborações anteriores com iniciativas internacionais em outras partes do mundo.

O objetivo principal do projeto FIBRE é a concepção, implementação e validação de uma infraestrutura compartilhada de pesquisa em Internet do Futuro que suporte a experimentação conjunta de pesquisadores europeus

e brasileiros. Para alcançar este objetivo o projeto executará cinco atividades principais:

- O desenvolvimento e a operação de uma nova instalação experimental no Brasil, incluindo a instalação de equipamentos de apoio à experimentação em diversas tecnologias (camada fixa 2 e camada 3, camada sem fio, camada óptica), bem como a concepção e a implementação de um arcabouço de controle para automatizar o uso e a operação da instalação experimental.
- O desenvolvimento e a operação de uma instalação experimental na Europa a partir de melhorias e da federação de duas infraestruturas experimentais existentes: OFELIA e OneLab. Duas ilhas OFELIA (i2CAT e UEssex) e a instalação experimental NITOS da UTH serão melhoradas pela i) inclusão de mais recursos físicos (servidores, switches OpenFlow e pontos de acesso sem fio) para poder lidar com um número maior de usuários e de casos de uso, ii) pela evolução dos seus arcabouços de controle (baseado no Expedient e no OMF) e iii) pelo incremento da força de trabalho para operar as instalações experimentais.
- A federação das instalações experimentais brasileiras e europeias, tanto no nível de conectividade física quanto no nível de arcabouço de controle, para apoiar o aprovisionamento de fatias de rede usando recursos das instalações das duas regiões.
- A concepção e a implementação de aplicações piloto de utilidade pública para demonstrar o potencial da instalação experimental em Internet do Futuro de larga escala compartilhada entre Europa e Brasil que foi desenvolvida.
- Aumentar a colaboração e troca de conhecimento entre pesquisadores europeus e brasileiros no campo de Internet do Futuro.

O ambiente de teste brasileiro, o qual será projetado e construído no projeto, incluirá localidades (também conhecidos como ilhas) em cada um dos nove parceiros brasileiros neste projeto. A Figura 2.8 mostra as localizações geográficas das ilhas brasileiras, das quais seis (incluindo a sede da RNP) são na Região Sudeste, e uma nas Regiões Norte, Nordeste e Centro Oeste. A Região Norte inclui 45,27% do território nacional, incluindo a floresta amazônica, onde as soluções cabeadas tradicionais de comunicações e transmissão de energia elétrica são frequentemente difíceis ou impossíveis de se usar. O GERCOM na UFPA tem estudado soluções

customizadas para comunicações no ambiente da floresta tropical e traz esta experiência e sua instalação experimental para este projeto.

Estas localidades serão interconectadas utilizando canais privados (de camada 2) por meio das redes de longa distância e metropolitanas à disposição da comunidade de pesquisa e educação brasileira. Estas incluem a rede *backbone* nacional Ipê da RNP e a rede para experimentação GIGA, mantida conjuntamente pela RNP e CPqD. As redes metropolitanas da RNP, entre elas a Rede MetroBel, serão usadas para acesso quando necessário e conexões internacionais RNP fornecerão acesso a outros ambientes de teste internacionais, tais como de OFELIA e de NITOS, para fins de federação.

Figura 2.8 – *Localização e interconexão das instalações no Brasil*

Cada ilha terá um núcleo comum de comutadores habilitados para OF, alguns baseados em NetFPGA e outros em comutadores de qualidade de produção, conjuntamente com seu(s) controlador(es), bem como um cluster de servidores de computação e armazenamento e outro cluster de nós sem fio Orbit, ambos apropriadamente virtualizados.

O projeto FIBRE pode ser visto como único a partir de diferentes perspectivas. Primeiro, pela inovação de prover federação entre instalações de menor escala entre a Europa e o Brasil visando o atendimento de uma escala de experimentação intercontinental. Segundo, porque o FIBRE habilita um conjunto de experimentações multitecnologia e multicamada numa escala também intercontinental. Este aspecto único torna-se possível pelos avanços no estado da arte em programabilidade, virtualização (ex.: camada óptica), monitoração e federação que o projeto FIBRE proverá.

3. Framework OpenFlow e Virtualização

3.1 Framework OPENFLOW

A pesquisa na área de redes de computadores possui diversos desafios em relação à implementação e experimentação de novas propostas em ambientes reais. Isso ocorre devido à dificuldade para o pesquisador possuir uma rede de testes próxima de uma rede real. Para isso, foi desenvolvido o OpenFlow [MCKEOWN, 2008] que propõe um mecanismo para permitir que redes reais sejam utilizadas como um ambiente de experimentos.

O OpenFlow é uma proposta tecnológica que, baseada no modelo de controle de rede logicamente centralizado, permite pesquisadores executarem seus experimentos em redes utilizadas no dia a dia, sem interferir no tráfego de produção, fundamentado em *switches* ethernet comerciais e define um protocolo padrão para controlar o estado dos switches (tabela de fluxos, estatísticas e etc.). O conceito de fluxo é o bloco fundamental que habilita os pesquisadores definir o plano de encaminhamento na rede, conforme os objetivos definidos pelas novas propostas de arquiteturas e protocolos de rede. O OpenFlow também define um novo elemento de rede, o controlador.

No OpenFlow, é proposto um mecanismo que é executado em todos os comutadores ou roteadores de forma que se possa haver isolamento entre os tráfegos, o experimental e o de produção. Assim, o OpenFlow possibilita que os pesquisadores reprogramem os comutadores, provocando interferência nas configurações da rede de produção. Outro objetivo dessa proposta é possibilitar que os fabricantes possam adicionar as funcionalidades do OpenFlow aos seus comutadores sem necessitarem expor o projeto desses equipamentos.

Além disso, tais equipamentos devem possuir um baixo custo e desempenho semelhante aos já utilizados na universidade, de forma que os administradores da rede aceitem a substituição dos equipamentos já existentes. Com base no exposto, o projeto do OpenFlow tem como objetivo ser flexível para atender aos seguintes requisitos:

- possibilidade de uso em implementação de baixo custo e de alto desempenho;
- capacidade de suportar uma ampla gama de pesquisas científicas;
- garantia de isolamento entre o tráfego experimental e o tráfego de produção;
- consistência com a necessidade de os fabricantes não exporem o projeto de suas plataformas.

O OpenFlow explora a tabela de fluxo que existe nos *switches* atuais, que normalmente são utilizadas para implementar serviços como NAT, *firewall* e VLANs. O *switch* OpenFlow é constituído de uma tabela de fluxos e um evento associado a cada entrada na tabela. Basicamente, a arquitetura do OpenFlow é composto por três partes:

i. Tabela de Fluxos: Cada entrada na tabela de fluxos contém uma ação associada, e consiste em Campos do cabeçalho (utilizado para definir um fluxo), ações (define como os pacotes devem ser processados e para onde devem ser encaminhados) e contadores (utilizados para estatísticas ou remoção de fluxos inativos).
ii. Canal Seguro: Para que a rede não sofra ataques de elementos mal intencionados, o *Secure Channel* assegura confiabilidade na troca de informações entre o switch e o controlador. A interface utilizada para acesso ao tráfego é o protocolo *Secure Socket Layer* (SSL). O NOX também suporta outras interfaces (passivas ou ativas), TCP e PCAP. Essas são bem úteis em ambientes virtuais, pela simplicidade de utilização, pois não necessitam de chaves criptográficas.
iii. Protocolo OpenFlow: Disponibiliza um protocolo aberto para estabelecer a comunicação entre o switch e o controlador. Ao fornecer uma interface externa que atue sobre os fluxos de um switch, o Protocolo OpenFlow (OFP – *OpenFlow Protocol*) evita a necessidade de um switch programável.

3.1.2. Aplicações

Como visto anteriormente, o OpenFlow permite o experimento de novas propostas na área de Redes de Computadores, utilizando uma infraestrutura de rede já existente. Dentre os possíveis experimentos permitidos, podem ser citados os seguintes [N. MCKEOWN et al. 2008]:

- Novos Protocolos de Roteamento: Os algoritmos de um protocolo de roteamento podem ser implementados para executarem no controlador de uma rede OpenFlow. Assim, quando um pacote do experimento chegar a um comutador, ele é encaminhado para o controlador, que é responsável por escolher a melhor rota para o pacote seguir na rede a partir dos mecanismos do protocolo de roteamento proposto. Após isso, o controlador adiciona entradas na Tabela de Fluxos de cada comutador pertencente à rota escolhida. Os próximos pacotes desse fluxo que forem encaminhados na rede não necessitarão serem enviados para o controlador.
- Mobilidade: Uma rede OpenFlow pode possuir pontos de acesso sem fio, permitindo que clientes móveis utilizem sua infraestrutura para se conectarem a Servidores ou à Internet. Assim, mecanismos de *handoff* podem ser executados no Controlador realizando alteração dinâmica das tabelas de fluxo dos comutadores de acordo com a movimentação do cliente, permitindo a redefinição da rota utilizada.
- Redes Não-IP: Como visto anteriormente, um comutador OpenFlow pode ser desenvolvido para analisar campos arbitrários de um pacote, permitindo flexibilidade na definição do fluxo. Assim, podem ser testadas, por exemplo, novas formas de endereçamento e roteamento. Nos comutadores do "tipo 0", os pacotes Não-IP podem ser definidos pelo endereço MAC de origem ou destino ou a partir de um novo Tipo de Ethernet ou IP.
- Redes com Processamento de Pacotes: Existem propostas de protocolos que realizam processamento de cada pacote de um fluxo como, por exemplo, os protocolos de Redes Ativas. Esse tipo de proposta pode ser implementado no OpenFlow forçando que cada pacote que um comutador receba seja enviado para o Controlador para ser processado. Outra alternativa é enviar esses pacotes para processamento por um comutador programável, como é o caso de um roteador programável baseado em Net-FPGA [J. LOCKWOOD et al. 2007].

3.1.2. Modos de funcionamento dos Switch

Existem dois tipos de comutadores OpenFlow. O primeiro consiste em um comutador OpenFlow dedicado, que não suporta encaminhamento comum de Nível 2 e Nível 3. O segundo tipo consiste em um comutador ou roteador comercial com OpenFlow habilitado que, além de suportar as funcionalidades do OpenFlow, realiza as funções comuns de um comutador ou roteador. Esses dois tipos são detalhados abaixo.

Comutador OpenFlow Dedicado

Esse tipo de comutador, exemplificado na Figura 3.1, é um elemento que apenas encaminha o tráfego entre as portas do comutador de acordo com a Tabela de Fluxos configurada remotamente, via controlador. O fluxo pode ser definido de diferentes formas. Por exemplo, um fluxo pode ser definido como todos os pacotes provenientes de certa porta TCP ou os pacotes com um determinado endereço MAC de destino. Além disso, alguns comutadores OpenFlow podem tratar pacotes que não utilizam o IPv4, verificando campos arbitrários de seu cabeçalho. Isso mostra a flexibilidade do OpenFlow para suportar diferentes tipos de experimentos.

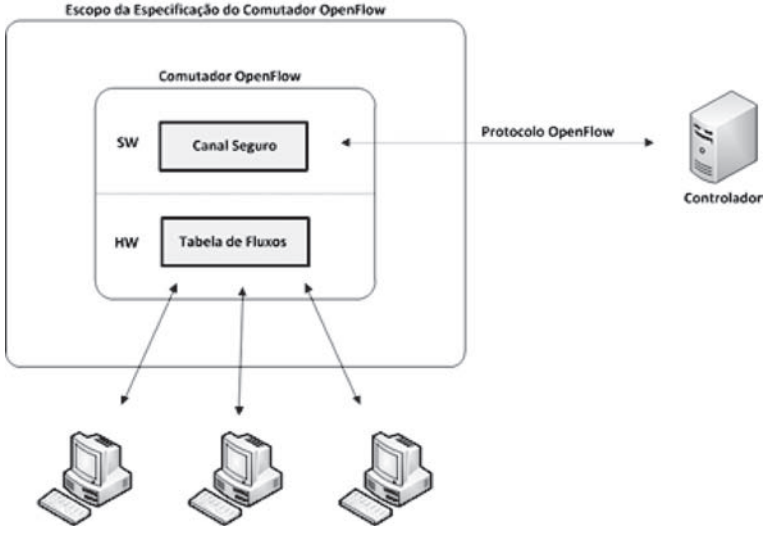


Figura 3.1 – *Comutador OpenFlow Dedicado*

Para cada entrada da Tabela de Fluxos é definida uma ação para ser tomada com os pacotes oriundos de um determinado fluxo. As três ações básicas que todos Comutadores OpenFlow devem suportar são as seguintes:

- encaminhar os pacotes do fluxo para uma (ou várias) porta(s) específica(s). Isso permite que os pacotes sejam roteados pela rede.
- encapsular os pacotes e encaminhá-los para o Controlador utilizando o Canal Seguro de comunicação. Essa ação pode ser executada no momento do envio do primeiro pacote de um fluxo, que ainda não tenha sido definido nas Tabelas de Fluxo dos comutadores, com o objetivo do Controlador configurar os comutadores com esse novo fluxo. Além disso, a ação pode ser realizada em um experimento no qual é necessário processamento adicional nos pacotes de um fluxo.
- descartar o pacote. Essa ação pode ser utilizada em aplicações de segurança para impedir, por exemplo, ataques de negação de serviço.

Uma entrada na Tabela de Fluxos possui três campos, como apresentado na Figura 3.2a. O primeiro identifica o cabeçalho que define o fluxo. Por exemplo, no cabeçalho dos comutadores OpenFlow (comutadores "tipo 0"), um fluxo pode ser definido por 10 parâmetros. Esses parâmetros podem ser o MAC e IP de origem ou destino, as portas TCP usadas, entre outros como mostra a Figura 3.2b. Em outras gerações de comutadores OpenFlow, esses parâmetros podem ser definidos arbitrariamente, permitindo o experimento de protocolos que não utilizam o IP.

Header Fields	Counters	Actions
Ether src	Received Packets	ALL
Ether dst	Transmitted Packets	CONTROLLER
IP src	Received Bytes	LOCAL
IP dst	Transmitted Packets	IN PORT
...	...	...

(a)

In Port	VLAN ID	Ethernet			IP			TCP	
		SA	DA	Type	SA	DA	Proto	Src	Dst

(b)

Figura 3.2 – *Cabeçalhos que definem um fluxo em comutadores "tipo 0"*

O segundo campo identifica a ação a ser tomada e o terceiro armazena as estatísticas do fluxo. Essas estatísticas podem ser o número de pacotes ou *bytes* referentes ao fluxo que passaram pelo comutador e o intervalo de tempo desde a última vez em que um pacote do fluxo foi identificado pelo comutador. Esse último é importante, por exemplo, para remoção de um fluxo da tabela caso esteja inativo.

Comutador com OpenFlow Habilitado

Esse tipo de comutador consiste nos equipamentos que já realizam encaminhamento normal de Nível 2 e Nível 3, mas adicionaram o OpenFlow com uma funcionalidade. Assim, o tráfego de produção pode ser encaminhado utilizando o encaminhamento normal e permanece isolado do tráfego experimental, que utiliza o mecanismo do OpenFlow. Para isso, esses comutadores podem adicionar outra ação além das três ações básicas citadas anteriormente:

- encaminhar os pacotes do fluxo pelo *pipeline* normal do comutador. Nessa ação, um fluxo identificado como não sendo referente ao OpenFlow será processado utilizando o encaminhamento normal.

Como alternativa ao uso da ação anterior, um comutador pode isolar o tráfego de produção do tráfego OpenFlow através do uso de VLANs. Alguns comutadores podem suportar tanto essa alternativa como a outra.

3.1.3. Controlador

O NOX [N. Gude et al. 2008] é uma proposta de sistema operacional para redes, possuindo como objetivo facilitar o gerenciamento de redes de grande escala. O conceito de sistema operacional de rede pode ser entendido pela analogia com os sistemas operacionais utilizados nos computadores. As ideias básicas dos sistemas operacionais de computadores são oferecer uma interface de alto nível para as aplicações utilizarem os recursos de *hardware* e também controlar a interação entre essas aplicações.

A interface de alto nível oferecida na primeira ideia tornou os programas mais fáceis de serem desenvolvidos e de executarem em diferentes plataformas de *hardware*. Isso ocorre porque os programadores

não precisam mais se preocupar com interações de baixo nível da aplicação com o *hardware* e escrevem seus programas utilizando primitivas genéricas que funcionam em diversas arquiteturas.

Em redes de computadores, o gerenciamento é realizado por configurações de baixo nível que depende do conhecimento da rede, análogo às aplicações desenvolvidas sem os sistemas operacionais. Como exemplo, o controle de acesso aos usuários de uma rede utilizando uma ACL (*Access Control List* – Lista de Controle de Acesso), necessita do conhecimento do endereço IP do usuário, que é um parâmetro de baixo nível dependente da rede.

Portanto, existe a necessidade de um sistema operacional de redes que forneça interfaces para controlar e observar uma rede, semelhantes às interfaces de leitura e escrita em diversos recursos oferecidos por um sistema operacional de computador [N. GUDE et al. 2008]. Assim, o sistema operacional de redes deverá fornecer uma interface genérica de programação que permite o desenvolvimento de aplicações de gerenciamento da rede. Esse sistema centraliza o gerenciamento da rede, necessitando haver uma grande preocupação com sua escalabilidade.

O NOX é um sistema operacional de rede que procura atender os requisitos já citados. Esse sistema foi desenvolvido para ser executado nos controladores de redes OpenFlow. Apesar do objetivo principal do OpenFlow ser o experimento de novas propostas, o NOX pode ser utilizado também no gerenciamento de redes de produção.

A Figura 3 ilustra os principais componentes de uma rede baseada no NOX. Esse tipo de rede possui um ou mais servidores executando o NOX. Cada um realiza o papel do controlador OpenFlow. Esses controladores possuem aplicações executando a partir do NOX. Todos os controladores compartilham uma única visão da rede, que é mantida na base de dados de um dos servidores.

A visão da rede é montada com base em informações da rede coletadas pelo NOX e é usada na tomada de decisões pelas aplicações de gerenciamento. As informações coletadas podem ser a topologia dos comutadores, a localização dos elementos da rede (ex. usuários, clientes, *middleboxes*) e os serviços oferecidos (ex. HTTP ou NFS).

As aplicações irão manipular o tráfego da rede a partir da configuração remota dos comutadores OpenFlow. Esse controle é realizado no nível de fluxo, ou seja, sempre quando um determinado controle é realizado no primeiro pacote de um fluxo todos os outros pacotes desse fluxo sofrerão a mesma ação.

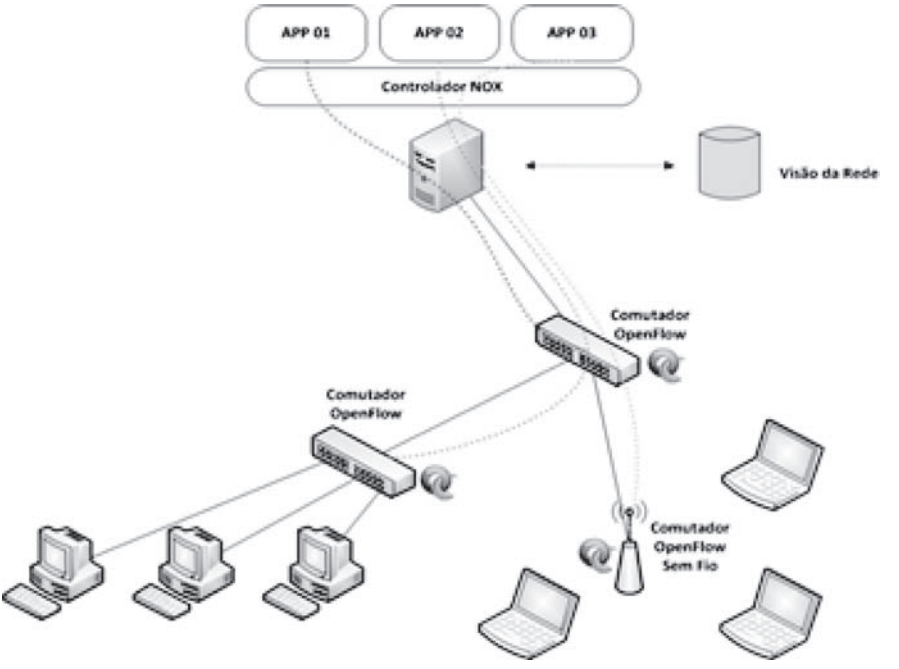


Figura 3.3 – *Componentes de uma rede baseada no NOX*

O funcionamento da rede baseada no NOX ocorre da seguinte forma: ao receber um pacote, o comutador verifica o cabeçalho do pacote para ver se ele está definido em sua Tabela de Fluxos. Se estiver definido, o comutador executa a ação especificada na Tabela de Fluxos e atualiza os contadores referentes à estatística do fluxo. Senão, encapsula o pacote e o envia para o NOX.

O NOX então será responsável por adicionar uma entrada na Tabela de Fluxos do comutador, identificando o fluxo referente àquele pacote. Dependendo da aplicação, o NOX pode não adicionar essa entrada, forçando que os comutadores enviem sempre para o NOX os pacotes que receberem.

Como exemplo de aplicação que pode ser desenvolvida para o NOX, pode ser citado o gerenciamento de consumo de energia [N. GUDE et al. 2008]. Por exemplo, nesse tipo de gerenciamento, pode ser reduzida a velocidade de enlaces subutilizados ou esses enlaces podem ser simplesmente desligados. Com a Visão da Rede do NOX, na qual é conhecida uma visão global da rede e as rotas em uso, pode-se adquirir conhecimento necessário para realizar tais ações. Além disso, essas ações podem ser auxiliadas por aplicações desenvolvidas para o NOX, como

redução de velocidade dos enlaces ou migração dos fluxos de um comutador que será desligado para outro comutador em uso.

3.2 Virtualização

A virtualização é uma técnica que permite que um sistema execute processos oferecendo a cada um deles a ilusão de executar sobre recursos dedicados. Inicialmente um mecanismo de isolamento, representa um fator de uso eficiente da crescente capacidade computacional disponível [EGI et al. 2007] e a integra a arquiteturas nas quais os elementos comuns a um conjunto de processos virtualizados possuem apenas uma cópia em execução, acessada de forma compartilhada [BHATIA et al. 2008].

O conceito foi estendido do âmbito de nós para os demais elementos de uma rede de computadores, dando origem à virtualização de redes [Chowdhury e Boutaba 2010]. Em um processo paralelo àquele descrito para a virtualização tradicional, a aplicação da virtualização de redes passou a permitir que os componentes de uma rede física particionassem sua capacidade de maneira a realizar simultaneamente múltiplas funções, estabelecendo infraestruturas lógicas distintas e mutuamente isoladas.

Assim como no caso da virtualização de sistemas, a virtualização de redes também permitiu que as arquiteturas de redes se tornassem mais eficientes. Funções que tradicionalmente eram gerenciadas de forma distribuída passaram a ser projetadas para uma execução e administração centralizadas. É o caso do encaminhamento do tráfego IP: podem ser encontradas arquiteturas do estado-da-arte [BOLLA et al. 2009] [NASCIMENTO et al. 2010] em que decisões de roteamento, originalmente tomadas de forma local por nós especializados, são encaminhados por comutadores a um sistema controlador, que executa em memória uma versão virtualizada da rede e dos respectivos elementos roteadores. Deriva decisões da base de informações de roteamento construída pela execução desta rede virtual e as transmite aos comutadores, que reagem de acordo.

3.2.1. Virtualização de Sistemas

Uma solução de virtualização fornece aos sistemas, executando sob sua supervisão, a abstração de um ambiente computacional exclusivo sobre o qual eles estão operando (nó "convidado"), quando de fato tem-se um computador ("anfitrião") cujos recursos foram partilhados entre esses mesmos sistemas.

Pode-se realizar a virtualização de sistemas de duas formas: a virtualização completa, em que cada convidado executa seu sistema operacional, e a virtualização baseada em containers, em que o sistema operacional do anfitrião distribui e isola os recursos disponíveis entre os sistemas convidados [BHATIA et al. 2008].

Ainda segundo Bhatia [2008], na virtualização completa, ilustrada na Figura 3.4, o anfitrião executa um Monitor de Máquinas Virtuais – MMV, também chamado *hipervisor*. É responsabilidade do MMV prover abstração de *hardware* (chamada de máquina virtual) para que seja possível executar o sistema operacional convidado, bem como mapear cada requisição dos convidados a seus respectivos dispositivos virtuais, para o elemento físico correspondente.

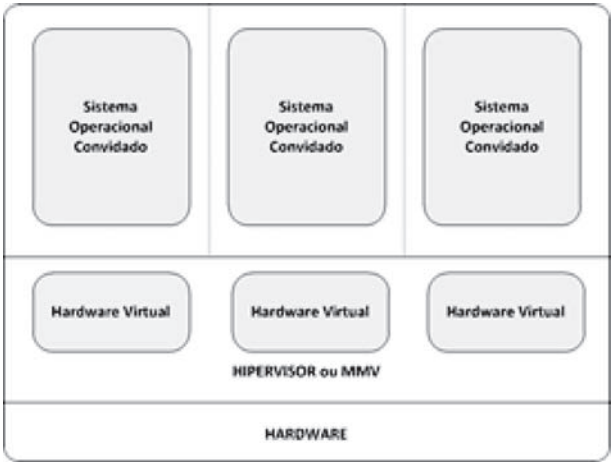


Figura 3.4 – *Virtualização Completa*

Também existe uma variação desta técnica chamada paravirtualização, que consiste em otimizar a emulação de *hardware* provida pelo MMV com vistas a um ganho de desempenho. Nesta abordagem, porém, os sistemas operacionais convidados devem ser modificados para que possam ser executados – o que não acontece na virtualização completa.

Em um sistema baseado em *containers*, apresentado na Figura 3.5, uma imagem de sistema operacional virtualizada é compartilhada entre todos os convidados – o que pode ser especialmente eficiente em cenários em que se deseja apenas executar de forma isolada diversas cópias do mesmo *software.* Ainda assim, os nós virtuais podem ser individualmente gerenciados, tal como na virtualização completa.

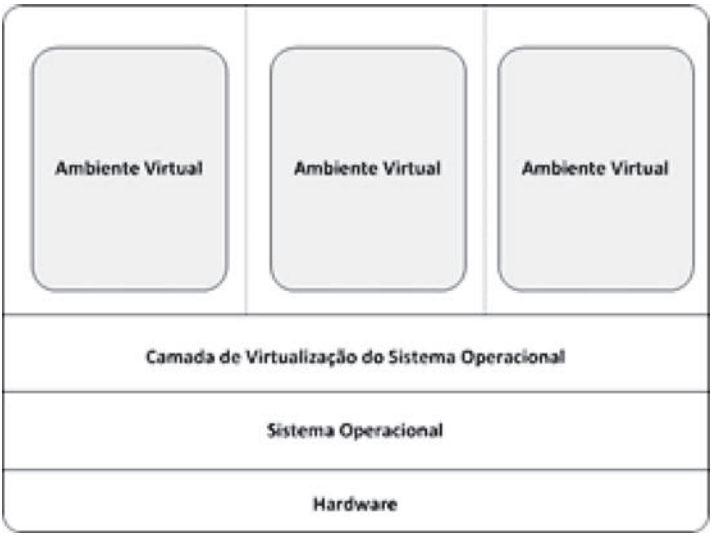


Figura 3.5 – *Virtualização utilizando container*

3.2.1.1 Principais ferramentas de virtualização de sistemas

A virtualização é empregada através de ferramentas que apresentam diferenças entre si e possuem suas vantagens e desvantagens. Atualmente, há uma gama enorme de softwares livres e empresas que fornecem soluções com esse conceito. Neste capítulo, comentaremos apenas as ferramentas Xen e o OpenVz, por serem as mais relevantes no contexto atual, bem como estabeleceremos uma comparação entre estas tecnologias.

XEN

O Xen é um monitor de máquina virtual (VMM ou *hypervisor*), em software livre, para arquiteturas x86. Originário de um projeto de pesquisa da Universidade de Cambridge, sua primeira versão foi criada em 2003, quatro anos antes de ser comprada pela Citrix System, em 2007. Ele apresenta uma solução para virtualização um pouco diferente das conhecidas.

Seu conceito consiste em criar um *hypervisor*, responsável por controlar os recursos das máquinas virtuais, mas que não possui drivers de dispositivos. Dessa forma, não é possível rodar um sistema operacional

diretamente no *hypervisor*. Assim, faz-se necessário que um sistema seja invocado para fazer a comunicação entre o *hypervisor* e os sistemas hóspedes.

A Figura 3.6 apresenta uma visão geral do Xen. Esse sistema inicial chama-se domínio 0. Ele consiste em uma máquina virtual que executa um núcleo Linux modificado e possui privilégios para acessar dispositivos de entrada e saída às outras máquinas virtuais, onde podem rodar outros sistemas operacionais, chamados domínio U. Elas são criadas, inicializadas e desligadas através do domínio 0. Possuem um *driver* virtual para acesso aos recursos de hardware.

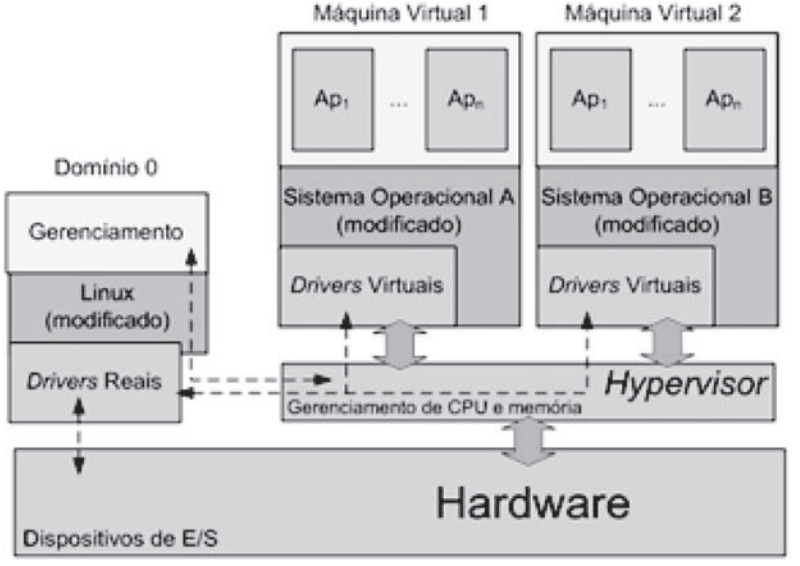


Figura 3.6 – *Visão geral do Arquitetura XEN*

O domínio 0 possui os *drivers* dos dispositivos da máquina física, além de dois *drivers* especiais que tratam as requisições de acesso à rede e ao disco enviadas pelas máquinas virtuais. Assim, toda requisição de uso da máquina real feita por uma máquina do domínio U deve ser tratada pelo domínio 0 antes de ser enviada ao *hypervisor*.

Originalmente, o Xen foi desenvolvido com o objetivo de implementar a técnica de paravirtualização, e, para isso, era necessário modificar os sistemas hóspedes para dar-lhes a consciência de rodarem sobre um *hypervisor*. Essa estratégia foi tomada visando ganhos em desempenho, mas limitou a difusão do Xen aos sistemas Unix, de código aberto.

OPENVZ

O *OpenVZ* é uma solução de virtualização em nível de sistema operacional. Cria ambientes virtuais isolados, que funcionam como servidores convencionais, porém, utilizando um único hardware em comum. Estes ambientes virtuais seguros são conhecidos como VE (*Virtual Environment*) ou como VPS (*virtual private server*).

VPS's podem ser reinicializados independentes uns dos outros. Todos possuem *hostname*, acesso de *root*, endereço IP e tudo mais que um servidor pode ter, sendo assim uma solução extremamente confiável e funcional de virtualização.

As capacidades básicas dos VPS's são:

- *Dynamic Real-time Partitioning*: Partição de um único servidor físico em dezenas de VPS's, cada um com funcionalidade total.
- *Resource Management*: Atribuição e controle dos recursos dos VPS's. Realocação de recursos em tempo real.
- *Mass Management*: Gerenciamento unificado de uma grande quantidade de servidores físicos e virtuais (VPS's).

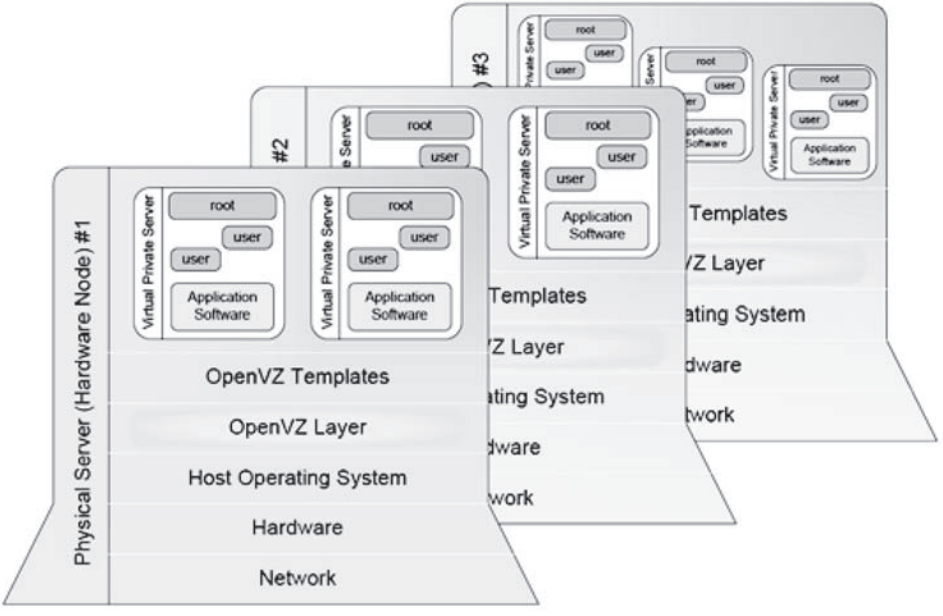


Figura 3.7 – *Arquitetura de virtualização utilizada pelo OpenVZ*

Como pode ser visto na Figura 3.7, um servidor físico pode conter diversos VPS's (*Virtual Private Servers*). Cada VPS possui isolamento total uns dos outros, inclusive com uma visão única de seus Ambientes Virtuais (VE's).

O OpenVZ provê uma solução para Provedor de Serviços de Hospedagem possibilitando que:

- centenas de usuários possuam seus próprios servidores privados (*Virtual Private Servers*) compartilhando um único servidor físico;
- provê para cada usuário uma garantia de qualidade de serviço;
- transferência transparente dos ambientes virtuais dos usuários para outros servidores físicos, sem nenhum tipo de reconfiguração manual.

O Virtual Private Server se comporta como um único servidor, em que:

- cada VPS possui seus próprios processos, usuários e provê acesso completo de *root* via shell;
- cada VPS possui seu próprio endereço IP, número de portas, firewall e roteamento;
- cada VPS possui seus próprios arquivos de configuração para o sistema e aplicações, como também suas próprias bibliotecas de sistema.

3.2.3. Virtualização de redes

Assim como a virtualização provê o compartilhamento de recursos de um nó computacional por múltiplos sistemas, a virtualização de redes (VR) é um método para que múltiplas arquiteturas de rede heterogêneas compartilhem o mesmo substrato físico – neste caso, componentes de uma rede como roteadores, comutadores, etc.

Segundo Chowdhury and Boutaba [2010], existem quatro abordagens para a implementação de VR: as Redes Locais Virtuais (VLANs), as Redes Privadas Virtuais (VPNs) e as redes de sobreposição (*Overlay*). Em Sherwood [2010], apresenta-se uma quarta, a partir do conceito de redes programáveis.

Redes Locais Virtuais (VLAN's)

Uma VLAN é um agrupamento lógico de dispositivos ou usuários que podem ser unidos por função, departamento ou aplicativo, independentemente da localização de seus segmentos físicos. A configuração de VLANs é feita no *switch*, e possivelmente no roteador, através de software proprietário do fabricante.

Com efeito, numa rede local a comunicação entre as diferentes máquinas é governada pela arquitetura física. Graças às redes virtuais (VLANs), é possível livrar-se das limitações da arquitetura física (constrangimentos geográficos, restrições de endereçamento, etc.), definindo uma segmentação lógica (software), baseada num agrupamento de máquinas com critérios como endereços MAC, números de porta ou protocolo.

Foram definidos vários tipos de VLAN, de acordo com o critério de comutação e o nível em que se efetua:

- VLAN de nível 1 (*Port-Based VLAN*) – define uma rede virtual em função das portas de conexão no comutador;
- VLAN de nível 2 (*MAC Address-Based VLAN*) – consiste em definir uma rede virtual em função dos endereços MAC das estações; e
- VLAN de nível 3 – distinguem-se vários tipos de VLAN de nível 3: VLAN por sub-rede (*Network Address-Based VLAN*), que associa sub-rede de acordo com o endereço IP fonte dos datagramas e VLAN por protocolo (em inglês *Protocol-Based VLAN*) que permite criar uma rede virtual por tipo de protocolo (por exemplo TCP/IP, IPX, AppleTalk, etc.), agrupando assim todas as máquinas que utilizam o mesmo protocolo numa mesma rede.

A VLAN permite definir uma nova rede acima da rede física e a esse respeito oferece mais flexibilidade para a administração e modificações da rede porque qualquer arquitetura pode ser alterada por simples parametrização dos comutadores. E ainda, um ganho em segurança, porque as informações são encapsuladas em um nível suplementar e são eventualmente analisadas. A VLAN oferece também redução da divulgação do tráfego sobre a rede.

Redes Privadas Virtuais (VPN's)

A ideia de utilizar uma rede pública como a Internet em vez de linhas privativas para implementar redes corporativas é denominada de *Virtual Private Network* (VPN) ou Rede Privada Virtual. As VPNs são túneis de criptografia entre pontos autorizados, criados através da Internet ou outras redes públicas e/ou privadas para transferência de informações, de modo seguro, entre redes corporativas ou usuários remotos.

A segurança é a primeira e mais importante função da VPN. Uma vez que dados privados serão transmitidos pela Internet, que é um meio de transmissão inseguro, eles devem ser protegidos de forma a não permitir que sejam modificados ou interceptados.

Outro serviço oferecido pelas VPNs é a conexão entre corporações (Extranets) através da Internet, além de possibilitar conexões *dial-up* criptografadas que podem ser muito úteis para usuários móveis ou remotos, bem como filiais distantes de uma empresa.

Uma das grandes vantagens decorrentes do uso das VPNs é a redução de custos com comunicações corporativas, pois elimina a necessidade de *links* dedicados de longa distância que podem ser substituídos pela Internet.

As LANs podem, através de *links* dedicados ou discados, se conectarem a algum provedor de acesso local e interligarem-se a outras LANs, possibilitando o fluxo de dados através da Internet. Esta solução pode ser bastante interessante sob o ponto de vista econômico, sobretudo nos casos em que enlaces internacionais ou nacionais de longa distância estão envolvidos. Outro fator que simplifica a operacionalização da WAN é que a conexão *LAN-Internet-LAN* fica parcialmente a cargo dos provedores de acesso.

Redes de Sobreposição (*Overlay*)

Uma rede *Overlay* é uma rede virtual que cria uma topologia virtual no topo da topologia física de outras redes. Os nós em uma rede de sobreposição são unidos por meio de ligações virtuais que correspondem em caminhos na rede subjacente. As redes de sobreposição são tipicamente programadas na camada de aplicação, embora várias implementações nas camadas inferiores da pilha de redes o faça existir.

As redes *Overlay* não são restritas geograficamente e são bastante flexíveis e adaptáveis a mudanças, se comparadas a qualquer outra rede. Como resultado, as redes sobrepostas têm sido usadas para implantar novos recursos e correções na Internet.

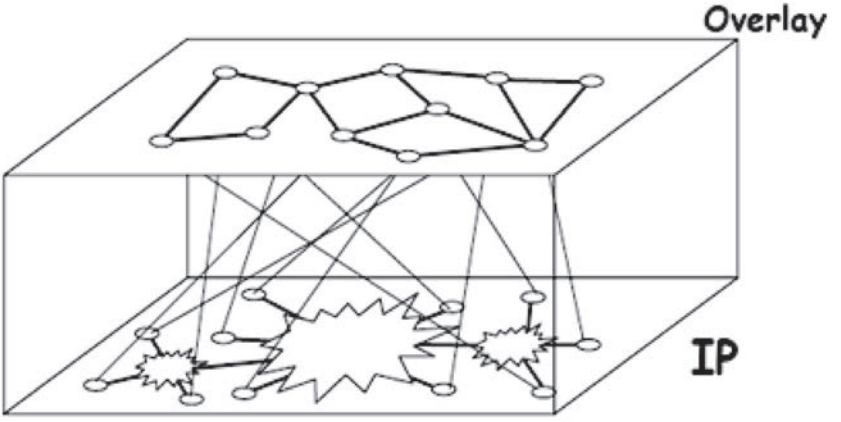


Figura 3.7 – *Modelo de rede de sobreposição*

Uma infinidade de modelos de sobreposição tem sido proposta nos últimos anos para tratar questões adversas que incluem: desempenho assegurado [SAVAGE,1999], disponibilidade de roteamento na internet [ANDERSEN, 2001], *multicast* [JANNOTTI, 2000][CHU, 2001], garantias de qualidade de serviço [SUBRAMANIAN, 2004], ataque de negação de serviços [ANDERSON, 2003], distribuição de conteúdo [KRISHNAMURTHY, 2001], compartilhamento de arquivos [LUA, 2005] e até sistemas de armazenamento [DABEK, 2001]. *Overlay* também está sendo usada como *testbed*, por exemplo PlanetLab [PETERSON, 2009], Federica [CAMPANELLA, 2009] e o GENI [GENI, 2011] para desenvolvimento e avaliação de novas arquiteturas.

3.2.4. Virtualização de Rede com OpenFlow

Similar à virtualização de computadores, a virtualização de redes promete melhorar a alocação de recursos, permitir que seus operadores possam checar suas redes antes de eventuais mudanças e também que clientes compartilhem o mesmo equipamento de forma controlada e isolada.

Portanto, analogamente, a rede em si deve ter uma camada de abstração de hardware, similar ao que acontece na virtualização de computadores. Esta camada deve ser facilmente "fatiável", para

que múltiplas redes completamente diferentes possam ser executadas simultaneamente em cima, sem interferir uns aos outros, sobre uma variedade de hardwares diferentes abaixo, incluindo switches, roteadores, pontos de acesso e assim por diante. Ou seja, acima desta camada de abstração de hardware, têm-se novos protocolos e formatos de endereçamento rodando independentemente de sua própria "fatia", uma mesma rede física, e na parte de baixo da camada de virtualização, novos hardwares, podendo ser desenvolvidos para diferentes ambientes e diferentes velocidades, mídia (com fio e sem fio) e energia.

De acordo com Rob [2010a], a camada de abstração de hardware é provida pelo OpenFlow e como camada de virtualização se tem FlowVisor. Similar à virtualização de computadores, o FlowVisor é uma cada de virtualização que reside entre o hardware e o software dos componentes da arquitetura, conforme ilustrado na Figura 3.8. Portanto, a integração FlowVisor e OpenFlow permite que em uma rede OpenFlow possam ser criadas várias fatias de recursos de redes rodando simultaneamente e isoladamente.

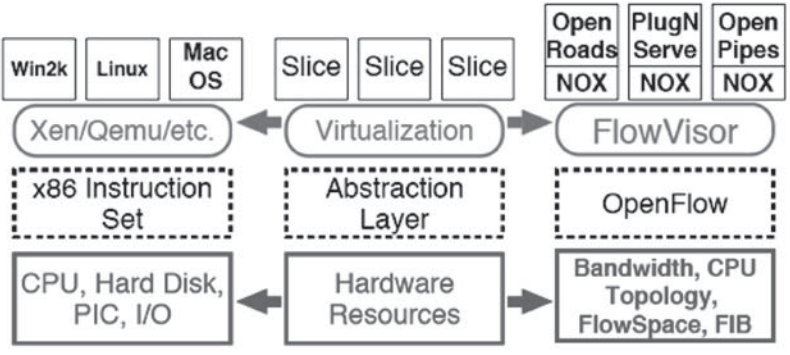


Figura 3.8 – *Comparação entre o FlowVisor como camada de virtualização com a virtualização computacional [Sherwood, 2009]*

O FlowVisor é um controlador especializado que atua como um *proxy* transparente entre os *switches* de uma rede OpenFlow e seus múltiplos controladores. Todas as mensagens do protocolo OpenFlow tanto aquelas originárias dos *switches* para os controladores quanto as dos controladores para os *switches*, são interceptadas através do FlowVisor. Desse modo, os controladores não necessitam de modificações e, devido à interceptação transparente do FlowVisor, os mesmos acreditam estar se comunicando diretamente com os dispositivos da rede que constitui o plano de dados [ROB, 2010b].

Devido à existência da interface entre os planos de dados e o de controle, a técnica empregada permite a partição do plano de dados em "fatias" (ou *slices*) que estão sob a gerência do FlowVisor [SHERWOOD, 2009].

Cada *slice* está vinculado a um controlador. O FlowVisor define um *slice* como um conjunto de fluxos também chamado de "espaço de fluxo" (ou *flowspace*). Devido à versão atual do protocolo OpenFlow permitir a definição de um fluxo como uma combinação de dez campos do cabeçalho de pacote incluindo informações das camadas física, de enlace, de rede e de transporte, o FlowVisor possibilita que se implemente *slices* com um elevado nível de granularidade, no que diz respeito à caracterização do *flowspace* [ROB, 2010a]. Além disso, os *slices* podem ser definidos por ações de negação, união e intersecção.

A Figura 3.9 ilustra o particionamento da rede em slices por meio do FlowVisor. Nesta figura, além do slice da rede de produção, têm-se mais dois slices identificados por "Alice" e "Bob". Os círculos enumerados em ordem crescente indicam a dinâmica de execução durante o processamento das mensagens interceptadas pelo FlowVisor. Inicialmente, o FlowVisor intercepta as mensagens provenientes de um determinado controlador "convidado" no ambiente de controle (1). Utilizando a política do espaço de fluxos definida para aquele slice (2), o FlowVisor reescreve a mensagem do controlador transparentemente para o slice da rede que compõe o plano de dados (3). As mensagens originárias dos switches para o plano de controle (4), por sua vez, são novamente interceptadas pelo FlowVisor e reescritas para o respectivo controlador de acordo com a política que define o espaço de fluxos.

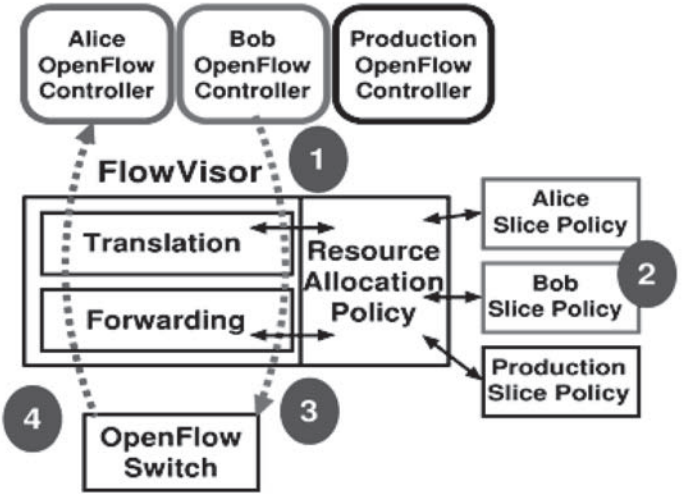


Figura 3.9 – *Gerenciamento de* slices *por meio do FlowVisor. [Sherwood, 2009]*

O particionamento da rede possibilita que as ações desenvolvidas em um de seus *slices* não interfiram negativamente nos demais, mesmo que estes estejam compartilhando a mesma infraestrutura física. Em arquiteturas mais tradicionais, a rede é "fatiada" através da técnica de VLAN (*Virtual Local Area Network*), porém, com a granularidade das redes, a estrutura de VLANs torna os experimentos, como IP *Mobility* ou *Wireless Handover,* por exemplo, bastante difíceis de gerenciar.

As características inerentes ao FlowVisor, como a virtualização transparente, o forte isolamento entre os *slices* e a sua rica política de definição de *flowspaces* tornam mesmo uma ferramenta extremamente eficiente no que diz respeito à virtualização e implementação de redes programáveis orientadas a software.

4. Requisitos para construção de um ambiente para experimento e virtualização de redes

Iniciativas [FIRE, 2010][GENI, 2011][AKARI, 2009] vêm construindo infraestruturas para testar novos protocolos, serviços e aplicações em ambientes de larga escala, visando solucionar esses desafios da Internet atual e compor a arquitetura da chamada Internet do Futuro.

Por essas razões, definir estratégias corretas para construção e operação destes *testbeds* para Internet do Futuro é considerado uma etapa

vital. Além disso, o ambiente deve combinar flexibilidade a um conjunto mínimo de restrições e um completo controle do ambiente para seus pesquisadores [YOUNG et. al, 2009]. Deste modo, o OpenFlow vem se destacando como framework capaz de habilitar experimentos de novas tecnologias utilizando redes reais de produção. Nesta mesma linha, a virtualização vem como uma ferramenta essencial, para permitir o acesso compartilhado de recursos, seja de rede ou computacional.

Portanto nesta seção, observam-se as informações importantes para construção de um *testbed*, levando em consideração a utilização de OpenFlow e virtualização. Verificam-se o modo que se descrevem, os hardwares utilizados dentro desta infraestrutura, tipos de enlaces, software de virtualização e ferramentas para análise de tráfegos e geração de tráfego.

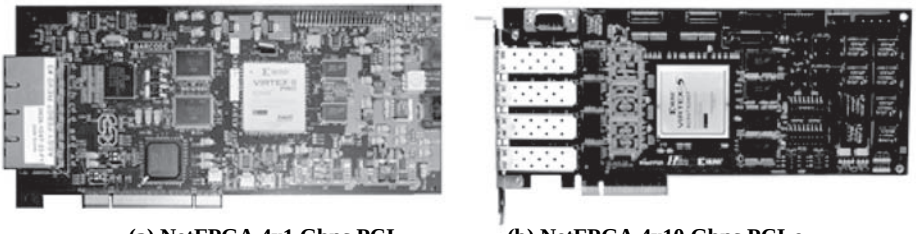
4.1. Tipos de Hardwares

Para construção do *testbed*, são necessários principalmente hardwares que suportem virtualização e OpenFlow. Para isso, são necessários dispositivos especiais para que o desempenho da rede virtual não tenha uma influência nos experimentos realizados e possibilite sua otimização de forma programável. Entre esses dispositivos estão interfaces de redes programáveis e *switches* programáveis com OpenFlow habilitado.

Interface de Rede Programável

O NetFPGA [LOCKWOOD, 2007][BILAL, 2009] é uma plataforma aberta que permite estudantes e pesquisadores desenvolverem protótipos de sistemas de redes em alta velocidade e sistemas de aceleração de redes via hardware, além de protótipos de switches ou roteadores IP para Internet do Futuro.

A NetFPGA consiste em uma placa de desenvolvimento reconfigurável, interface de ligação ao PC, utilizando PCI e PCI-e, dois processadores PowerPC, SDK para o desenvolvimento de novas funcionalidades, um *troughput* que varia de 8Gbps até 80 Gbps, dependendo da versão da placa, e quatro interfaces de 1 Gbps ethernet RJ-45 ou quatro interfaces 10 Gbps ethernet SFP+. A Figura 4.1 ilustra as duas versões de placas NetFPGA, modelos de 1 (Figura 4.1a) e 10 Gbps (Figura 4.1b). Além disso, é totalmente compatível com Linux e OpenFlow.



(a) NetFPGA 4x1 Gbps PCI **(b) NetFPGA 4x10 Gbps PCI-e**

Figura 4.1 – *Modelos de NetFPGA disponíveis*

Switches Programáveis

O OpenFlow é um framework que permite o gerenciamento da tabela de encaminhamento de switches Ethernet, desacoplando a lógica de controle do equipamento do hardware de encaminhamento de pacotes, baseado no conceito chamado de *Software-Defined Networking* (SDN) [GREENE, 2009]. Esta separação não só permite um modelo de inovação aberta tanto no plano de controle quanto no plano de encaminhamento, mas também permite a virtualização do plano de encaminhamento em fatias ou redes lógicas. Deste modo, são necessários equipamentos que possuam o OpenFlow habilitado.

No entanto, o protocolo Openflow ainda está em fase de padronização nos *switches* e alguns fabricantes já disponibilizam versões de switches com OpenFlow habilitado, como é o caso dos modelos Pronto 3290 para soluções até 1GbE e o Pronto 3790 para 10 GbE e SFP+. Também há o projeto Indigo [INDIGO, 2011] que é uma implementação aberta do OpenFlow que permite rodar sobre switches físicos baseados em chipsets *Broadcom*. A cada switch suportado é desenvolvido um firmware para essa linha de produto. Este firmware sobrescreve o original do switch, instalando a nova imagem com OpenFlow habilitado. Dentre os switches compatíveis, temos, além dos modelos da linha Pronto, os modelos Netgear GSM7328 (24 x 1 GbE) e GSM7352 (48 x 10 GbE).

Servidores de alto desempenho

Os servidores também são uma parte importante dentro do substrato, pois deles são virtualizados recursos como: CPU, armazenamento e I/O. Para isso, são necessários servidores de alto poder computacional

para que não haja perda de desempenho no nível virtualizado e suporte a quantidade de usuários locais e também os federados. Observando esses requisitos, as linhas de servidores do tipo *BLADE* surgem como uma excelente solução para oferecer esse poder computacional.

Blade é uma nova forma de tecnologia computacional que permite a alta densidade de componentes e recursos incluindo servidores, armazenamento e interfaces de comunicação integradas em um mesmo chassi.

A vantagem do uso de servidores *blades* é a possibilidade do crescimento incremental mediante a demanda de usuários, devido a sua característica modular, baseada em contêineres de servidores, switches e armazenamento. O seu poder computacional pode ser incrementado com introdução de novas lâminas *blades*, que são servidores que inseridos no chassi com recurso de processamento e memória. Além disso, outra vantagem é a sua otimização para uso de virtualização com barramento em altíssimas taxas de velocidade e implementação de grupos de recursos dentro de conjunto virtualizado. O uso de *blades* está ligado principalmente a virtualização de servidores e a computação em nuvem.

4.2. Arcabouços de Controle

Os arcabouços de controle é o coração de qualquer infraestrutura de experimentação baseada em virtualização. Porém, muitos arcabouços são limitados a tipo de recurso e a um tipo de comunidade de pesquisa. Com base nisso, são apresentados alguns frameworks de controle que devem ser selecionados de acordo com o tipo de infraestrutura que se está oferecendo:

ProtoGENI

O ProtoGENI [PROTOGENI, 2011] é atualmente dirigido pela Universidade de Utah. É um framework que pretende controlar a integração em larga escala de infraestruturas existentes e em construção no GENI. Esses *testbeds* são compostos principalmente por elementos embarcados e programáveis no *backbone*, PCs com hardwares programáveis e uma variedade de redes sem fio, redes de acesso e *clusters* programáveis. Atualmente, o ProtoGENI está usando RSpec para descrever seus nós, enlaces e interfaces. Esses recursos são mapeados para o Emulab que aplica os mapas virtuais de recursos em nós locais e vlans.

FEDERICA

O FEDERICA [SEZGEDI et. al, 2009] é uma iniciativa composta de roteadores, *switches* e computadores para experimentações em Internet do Futuro. Ele é baseado em redes de Ensino & Pesquisa (*NRENs – National Research and Education Network*) e no *backbone* da GEANT2.

O Arcabouço FEDERICA é capaz de alocar recursos para múltiplos *slices* e diferentes redes a serem utilizadas pelos pesquisadores que possuem o controle completo dos recursos de seu *slice*. Ele é composto de pequenas ferramentas e outros recursos de instrumentação que vão desde a gerência e controle de máquinas virtuais à alocação de circuito fim a fim camada 2, camada 1 ou MPLS.

ORCA

O ORCA [ORCA, 2011] [BEN, 2011] é *framework* de código fonte aberto, utilizado para gerenciar e controlar a programabilidade de substratos (hardware) compartilhados, que podem incluir, servidores, *storages*, redes e outros componentes. O ORCA foi desenvolvido como um protótipo arcabouço GENI.

No ORCA, há uma coleção dinâmica de controladores de interação que trabalham juntas para o provisionamento e configuração de recursos para os *slices* requisitados pelos seus usuários de acordo com as políticas dos participantes.

Atualmente, o ORCA está sendo estendido para incluir recursos ópticos disponíveis em *metro-scale optical testbed* na internet2. Também está em processo de integração a seleção para utilização de protótipos para redes de sensores e sem fio, de modo que o ORCA ofereça a maior variedade de *testbeds* possíveis.

PII

O objetivo do projeto PII [RAJ JAIN et. al, 2011] é criar uma federação de instalações experimentais entre diferentes pólos de inovação regional na Europa. Isso permite que as empresas participantes possam testar novos serviços de comunicação e aplicações na Europa como um todo.

O *framework* também terá como esforço integrar federados *testbeds* distribuídos em laboratórios de redes espalhados pela Europa provendo um

realístico ambiente de teste para novos conceitos de serviços, tecnologias de rede e modelos de negócios. Os mecanismos de seu framework incluem regras e procedimentos para alcançar níveis de teste e colaboração dentro dessas federações de infraestruturas dentro da Europa.

4.3. Monitoração e Medição

Medição e monitoramento são atividades que determinam o estado operacional da infraestrutura ou *testbed* que está sendo observada, de modo que se possa analisar informações como: os desempenhos dos recursos disponíveis no *testbed*, verificar a sua disponibilidade do recurso ou componente e medir as características dos componentes ou da rede que estão sendo disponibilizadas. O objetivo disso é oferecer a outros sistemas uma visão dos recursos da infraestrutura, para auxiliar decisões como: a possibilidade de alocação de recurso, bem como sua escolha e disponibilização.

Portando, dentro dos requisitos para construção dessas infraestruturas de *testbed*, esses elementos são essenciais. A seguir, são apresentados alguns softwares relacionados a características de medição e monitoramento.

PerfSONAR

O perfSONAR (*PERFormance Service Oriented Network Monitoring Architecture*) é uma arquitetura orientada a serviços (SOA) que foi projetada para o monitoramento de redes em ambientes interdomínios. No perfSONAR, existe um conjunto de serviços bem definidos que realizam ou disponibilizam resultados de medições de desempenho em um ambiente federado. Esses serviços atuam como uma camada intermediária entre as ferramentas de medições e as ferramentas de visualização [TIERNEY, 2009].

Além disso, o perfSONAR também definiu um protocolo comum entre os serviços para permitir que sejam criados novos serviços seguindo a padronização deste protocolo, dando flexibilidade à arquitetura. Também, criou-se uma representação unificada para definir, armazenar e arquivar os dados relativos às medidas do experimento que é mais um componente chave, de modo que certo nível de concordância é necessário para fazer convergir esforços e melhorar a experiência, fidelidade e usabilidade das infraestruturas de experimentação em escala global.

Real-Time Unified Measurement Framework

O UMF (*Unifed Measurament Framework*) é um framework desenvolvido para oferecer capacidade de medições em tempo real a avaliações de desempenho sobre vários cenários de infraestrutura, interfaces de integração dos recursos de medição com os substratos e os framewoks de controle dos protótipos de GENI e outros sistemas que necessitem de suas medidas e a monitoração das condições dos recursos de um *slice* durante um experimento [UMF, 2011].

Inicialmente, o UMF está sendo desenvolvido para alimentar informações dos demais arcabouços de controle do GENI e de visualização dos usuários pesquisadores. Na Figura 4.2, mostra o esquema de interação dos frameworks com o UMF.

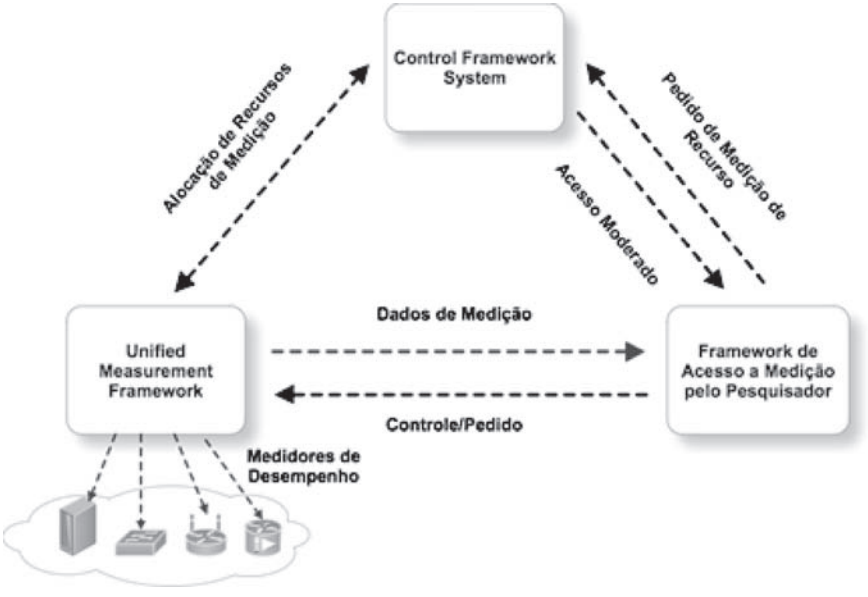


Figura 4.2 – *Esquema de interação do UMF com outros frameworks do GENI*

UMF é um arcabouço genérico para gerenciamento de *testbed* e coleta de métricas de experimentos. Quando o experimento está sendo executado, os dados são medidos e coletados de acordo com a descrição do usuário. Pontos de medição são definidos dentro de uma aplicação C/C++ ou automaticamente, baseados em arquivos de configuração XML.

Os dados medidos são transmitidos usando codificação XDR e salvos em um banco de dados para análise posterior. Um novo banco de

dados de medição é criado para cada experimento. O SQLite pode ser usado para manipular os dados ou estes podem ser exportados para geração de gráficos ou tabelas.

No UMF, há medidores de desempenho (*Performance Monitors*) para os mais variados tipos de substratos físicos, como: unidades externas de hardware, por exemplo, servidores com capacidade de armazenamento (*storages*), interface de ethernet NetFPGA (10G - NetFPGA). Outras interfaces ethernet seriam GPIB e wireless, e protocolos padrões para medição como TL1, SNMP e NetConf.

PS-Performance ***Toolkit***

O pS-NPToolkit [Ps-Performance, 2011] é uma coleção de ferramentas de medida de desempenho de redes. Foi desenvolvida para coletar, armazenar e analisar o desempenho da rede. Esse toolkit é usado à parte para criar uma completa análise em um determinado nó, bem como para fazer o monitoramento periódico na infraestrutura do *testbed*. Além disso, essas ferramentas podem ajudar a identificar mudanças de desempenhos na rede ou pontos que necessitam de uma atualização em suas políticas para manter o desempenho esperado.

BWCTL

O BWCTL [HU, 2010] é uma aplicação cliente/servidor que permite o agendamento de testes com políticas de medidas utilizando IPERF [MAZHAR, 2008], THRULAY [Shalunov, 2011] e NUTTCP [Fink, 2011]. Estes testes podem medir, por exemplo, a máxima largura de banda para TCP ou testes UDP para medir o atraso, a variação do atraso ou nível de perda de datagrama na rede. BWCTL tem sido utilizado para testar as qualidades das reservas de circuitos. Neste caso, o seu cliente solicita e agenda um circuito, verifica se foi criado ou não, e caso seja criado, ele verifica se os desempenhos solicitados estão realmente disponíveis.

4.4. Ferramentas para Gerência de Virtualização

As ferramentas de gerência são essenciais no desenvolvimento dos ambientes de experimentação, pois elas têm o papel de manipular, criar,

configurar e remover recursos em *testbed*. Portanto, abaixo, são apresentadas as principais ferramentas para gerenciamento de virtualização, computação em nuvem e manipulação de *slices*.

LIBVIRT

A LIBVIRT [WU, 2010] existe como um conjunto de APIs projetadas para serem usadas como um aplicativo de gerenciamento. Por meio de um mecanismo específico de *hypervisores*, a LIBVIRT comunica-se com um *hypervisor* disponível para executar as solicitações da API. Deste modo, a LIBVIRT provê uma comum genérica e estável camada para gerenciamento. A API pode acessar esse *hypervisor* permitindo o provisionamento, criação, modificação, controle, monitoramento, migração e inicialização e paralisação de VM (*virtual machine*). No entanto, o suporte a todas essas operações irá depender da tecnologia de virtualização utilizada.

Com a LIBVIRT, têm-se dois meios de controle distintos. O primeiro é demonstrado na Figura 4.3, no lado esquerdo, em que o aplicativo de gerenciamento e os domínios existem no mesmo nó. Nesse caso, o aplicativo de gerenciamento trabalha por meio da biblioteca *libvirt* para controlar os domínios locais. Outros meios de controle existente são quando o aplicativo de gerenciamento e os domínios estão em nós separados, que é demostrado na Figura 4.3, lado direito. Este modo usa um *daemon* especial chamado *libvirtd*, que é executado em nós remotos. Tal *daemon* é iniciado automaticamente quando a *libvirt* é instalada em um novo nó e pode determinar, de forma automática, os *hypervisores* locais e configurar *drivers* para eles. O aplicativo de gerenciamento se comunica por meio da *libvirt* local com o *libvirt* remoto através de um protocolo customizado [BOLTE, 2010].

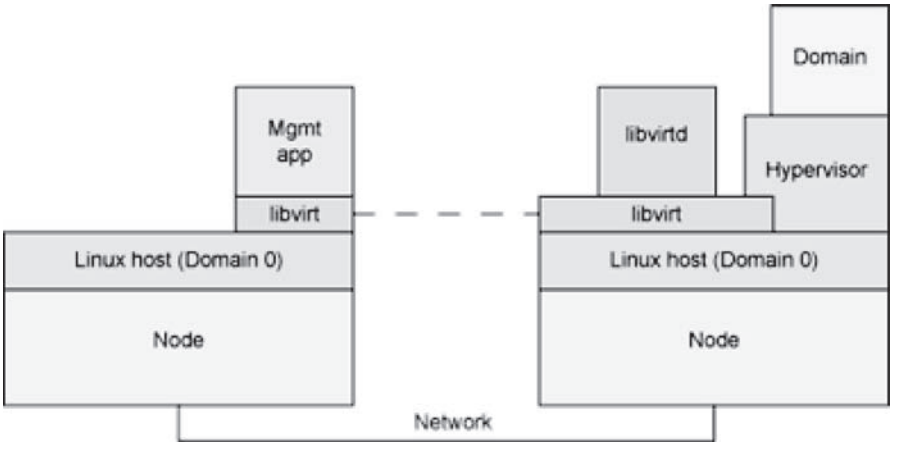


***Figura 4.3** – Modos de controle de hypervisor [Wu, 2010]*

A API LIBVIRT foi construída para trabalhar através de múltiplos ambientes de virtualização. Dentre as tecnologias de virtualização temos: QEMU, LXC, OpenVZ, *User Mode Linux*, KVM, *VirtualBox* e *VMWare*. Além disso, também consegue gerenciar as seguintes tecnologias de armazenamento: *Storage* IDE/SCSI/USB, *FiberChannel*, LVM, iSCSI e NFS.

Eucalyptus

Eucalyptus [Nurmi, 2009] é framework aberto para montagem e gerência de ambientes de computação em nuvem utilizando virtualização sendo o seu foco para pesquisa acadêmica. Ele provê uma implementação baseada em IaaS (*Infraestructure as a Service*), ou seja, infraestrutura como recurso nuvem. Os usuários do Eucalyptus são capazes de iniciar, controlar, acessar e terminar máquinas virtuais (VMs). A atual versão do *Eucalyptus* suporta VMs, rodando sobre uma XEN *hypervisor*, mas há planos de utilizar KVM/QEMU e VMWare.

O projeto Eucalyptus apresenta quatro características que o diferenciam de outras soluções de computação em nuvem e virtualização: o *Eucalyptus* foi projetado para ser simples, de modo que não há a obrigatoriedade da dedicação de recursos; o framework foi projetado para encorajar extensões de softwares de terceiros; possui uma interface externa que é baseada na Amazon API EC2; e o *Eucalyptus* provê uma rede sobreposta virtual que isola o tráfego de rede de diferentes usuários, permitindo que clusters remotos pareçam partes da mesma rede local.

A gerência do *Eucalyptus* é toda baseada em WebService que oferece alguns benefícios à arquitetura como a exposição da API em forma bem conhecida e funcional como o WSDL (*Web Service Description Language*), definindo tanto as suas operações aos quais seus serviços podem desempenhar quanto as estruturas de dados de entrada e saída utilizadas. A arquitetura define três níveis de gerenciamento da infraestrutura de nuvem:

- *Instace Manager (IM)*: Controla a execução, inspeção e finalização das instâncias de VMs nos hospedeiros que estão rodando.
- *Group Manager (GM)*: Suas funcionalidades são as seguintes: escalonamento e gerenciamento de execução de operações sobre os IM, controle de operações sobre a rede virtual sobreposta e reportar informações sobre um conjunto de IMs.
- *Cloud Manager (CM)*: Interage com os usuários e gerenciadores reportando informações a respeito dos estados dos recursos da nuvem.

O Eucalyptus é flexível e pode ser instalado em uma mínima infraestrutura (no mínimo duas máquinas), como também em milhares de núcleos e *terabytes* de armazenamento. Tudo totalmente integrado sobre uma rede sobreposta de interligação das infraestruturas de nuvem.

E-GENI

O Enterprise-GENI [E-GENI, 2011] é uma arquitetura/*framework* utilizada para gerenciar o uso de *slice* em uma infraestrutura baseada em OpenFlow e FlowVisor. O objetivo do *framework* é criar uma interface para visualização e gerências de múltiplos experimentos em rede OpenFlow. A Figura 4.4 mostra como está definida a arquitetura do E-GENI.

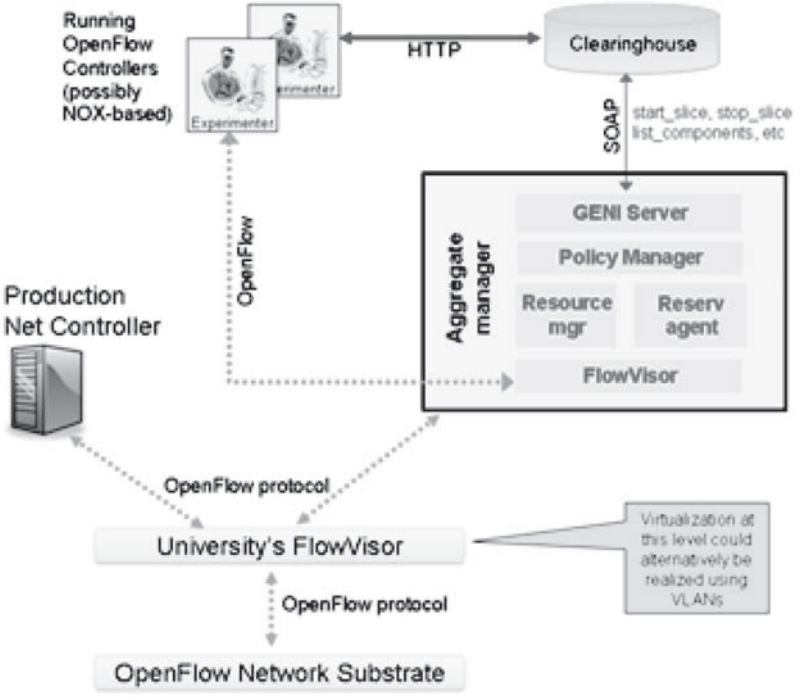


Figura 4.4 – *Arquitetura do E-GENI [E-Geni, 2011]*

A arquitetura do E-GENI é dividida em três partes:

- *OpenFlow-Based Substrate:* Composto de switches que se comunicam utilizando o protocolo OpenFlow para uma aplicação do controlador.
- *FlowVisor:* Customizado para controlar slice de rede pelo isolamento e controle de tráfego de experimentos individuais.
- *Aggregate Manager:* Uma aplicação integra o clearinghouse, responsável pela manipulação dos experimentos, ao E-GENI FlowVisor utilizando o protocolo SOAP, baseado no GENILight Protocol.

O E-GENI vem sendo desenvolvido para integrar redes do tipo OpenFlow ao *framework* GENI como mais uma alternativa de experimento dentro da infraestrutura de *testbed* para Internet do Futuro da GENI.

5. Estudo de Caso

Com o propósito de criação de ambientes experimentais em Internet do Futuro, a comunidade acadêmica vem orientando esforços consideráveis na criação de *testbeds* baseados em redes programáveis. Baseada em tráfegos reais de usuários, a criação de *testbeds* programáveis permite uma abordagem mais realística com relação à escalabilidade da rede e seu comportamento interativo. Por outro lado, a construção e manutenção destes ambientes possui custos elevados e a realização de testes de novos protocolos, por exemplo, pode ser bastante dispendiosa e gerar muitos problemas devido a sua interferência no tráfego de produção.

De maneira complementar a esta abordagem com *testbeds*, este estudo de caso tem por objetivo demonstrar como é possível implementar uma rápida prototipagem de protocolos de rede que seja facilmente aplicável a redes reais. Ou seja, por meio de um ambiente local, é possível implementar uma funcionalidade que possa ser imediatamente aplicada para testes num ambiente global possibilitando uma maior inovação em redes programáveis.

Como importante proposta no contexto de redes programáveis, neste estudo de caso utiliza-se do *framework* OpenFlow devido à possibilidade de criação de uma infraestrutura de experimentação sobre uma rede de produção. Além disso, por meio da utilização do OpenFlow associado à virtualização, demonstra-se como é possível criar, utilizar e gerenciar vários *slices* sobre uma infraestrutura de rede compartilhada, possibilitando que vários experimentos de protocolos possam ser executados de maneira simultânea.

Por meio deste estudo de caso demonstra-se que se pode implementar uma nova funcionalidade, ou mesmo uma nova arquitetura de rede em ambientes locais baseados em software. Estas funcionalidades podem então ser testadas por seus usuários por meio de largas topologias e com tráfegos específicos. Posteriormente, os mesmos códigos que implementam estas funcionalidades podem ser empregados em ambientes de *testbeds* reais.

5.1. Definição do Ambiente

Para exemplificar a utilização do OpenFlow associado à virtualização, esse estudo de caso tem como principal objetivo apresentar como se pode desenvolver um ambiente capaz de suportar simultaneamente vários experimentos de protocolos compartilhando a mesma infraestrutura física.

5.1.1. Dados de Implementação do Ambiente

O estudo de caso é formado por um laboratório virtualizado que implementa o plano de controle e o plano de dados baseados em uma rede programável. Com o objetivo de implementar um ambiente com suporte ao *framework* OpenFlow, utiliza-se dois servidores: um com *Xen Server* para criar os controladores virtualizados que serão utilizados nos experimentos e outro servidor utilizando apenas o sistema operacional Linux e o software Mininet. O Mininet utiliza virtualização baseada em contêineres para criar uma instância virtual de cada *switch* (ou host) que seja utilizado no experimento. Além disso, utiliza-se um switch *SummitX150* para fazer a ligação entre o plano de controle e plano de dados de maneira fora da banda. A Figura 5.1 ilustra o ambiente que foi desenvolvido para realização dos testes.

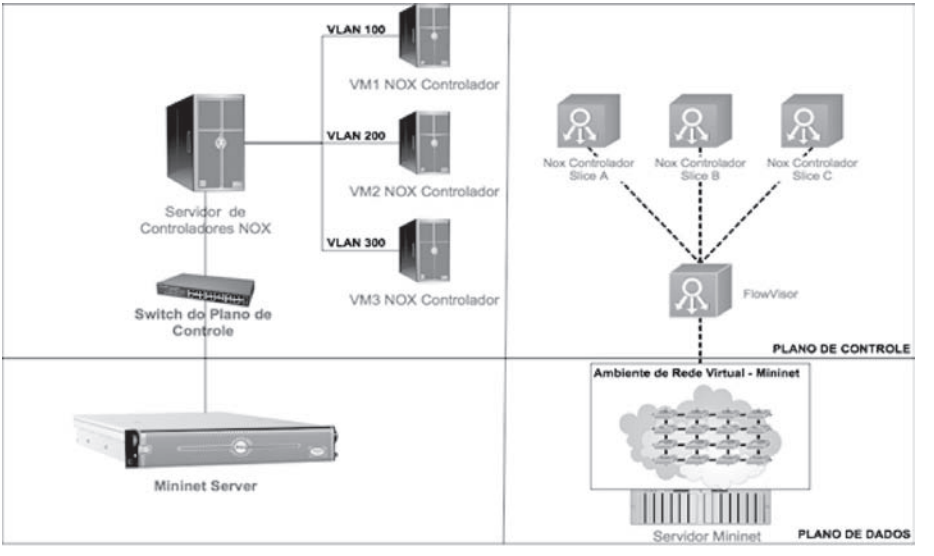


Figura 5.1 – *Ambiente de teste do estudo de caso*

A Tabela 5.2 apresenta as configurações dos servidores utilizados nos estudos de caso.

Tabela 5.1 – *Dados dos servidores*

Dados	Servidor MiniNet	Servidor de Controladores NOX
Sistema Operacional	Debian Lenny 5.0	Citrix XenServer 5.6
Memória	4096 MB	2048 MB
Interface de Rede 1GbE	2	2
Armazenamento	320 GB SAS	500 GB SATA
Virtualização	Contêiner	XEN

No servidor de controladores virtuais têm-se três VMs responsáveis por conter cada servidor controlador NOX. Estes controladores gerenciam os *slices* criados no plano de dados da infraestrutura do estudo de caso. Cada controlador se comunica com FlowVisor usando 3-tuplas de informações: VLAN, IP e porta TCP do processo. Com o objetivo de isolar o tráfego de cada controlador para eventuais análises do comportamento das informações de controle entre os dois, as ligações entre o controlador e o FlowVisor são divididas por VLAN. A Tabela 5.2 contém um resumo dos dados de configuração de cada VM contendo os controladores.

Tabela 5.2 – *Configurações das VMs com os Controladores NOX*

Dados	VM 1	VM 2	VM3
Sistema Operacional	Debian Lenny 5.0	Debian Lenny 5.0	Debian Lenny 5.0
Memória	512 MB	512 MB	512 MB
Interface de Rede 1GbE	1	1	1
Endereçamento IP	192.168.100.1	192.168.200.1	192.168.300.1
VLAN	100	200	300
Software Controlador	NOX 0.5	NOX 0.5	NOX 0.5
Porta de Conexão	1515	1516	1517
Armazenamento	4 GB	4 GB	4 GB
Aplicação	Switch	Switch e SpanningTree	Switch e Routing

No Servidor Mininet, têm-se dois ambientes distintos virtualizados. O primeiro contém o aplicativo Mininet virtualizando a infraestrutura física de switches e hosts, ou seja, ele é responsável pela

criação do plano de dados. No segundo ambiente tem-se o FlowVisor como o software que realiza a criação, remoção e gerência dos *slices* no plano de dados do Mininet. Assim, todos switches se comunicam ao FlowVisor utilizando endereçamento IP e porta TCP. A Tabela 5.3 resume as configurações do servidor Mininet.

Tabela 5.3 – *Dados do servidor MiniNet*

Dados	Servidor MiniNet
Sistema Operacional	Debian Lenny 5.0
Memória	4096 MB
Interface de Rede 1GbE	2
Armazenamento	320 GB SAS
Virtualização de Rede	FlowVisor 0.6
Virtualização	Contêiner Virtuais LXC
Porta FlowVisor	6633

Mesmo considerando a infraestrutura de switches Openflow como virtualizada, os procedimentos aqui executados são compatíveis com aqueles realizados em ambientes reais. Portanto, as aplicações utilizadas no plano de controle neste estudo de caso são aplicáveis em uma rede com switches ou roteadores com protocolo Openflow habilitado. Desse modo, o ambiente desenvolvido aqui serve como um ponto de partida no desenvolvimento de uma nova solução que depois pode ser aplicado em um ambiente Openflow real.

5.1.2. Plano de dados

Para construção do plano de dados, o software Mininet utiliza um *script* que contém a descrição de nós (*switches* ou roteadores) e *hosts* (clientes) que são criados por meio de sua API. Além disso, por meio deste *script* é possível definir a topologia da rede a ser utilizada. Neste estudo de caso, utiliza-se a topologia representada pela Figura 5.2. O *script* desenvolvido chamado "shortCourse_topology.py" está descrito detalhadamente no Apêndice A.

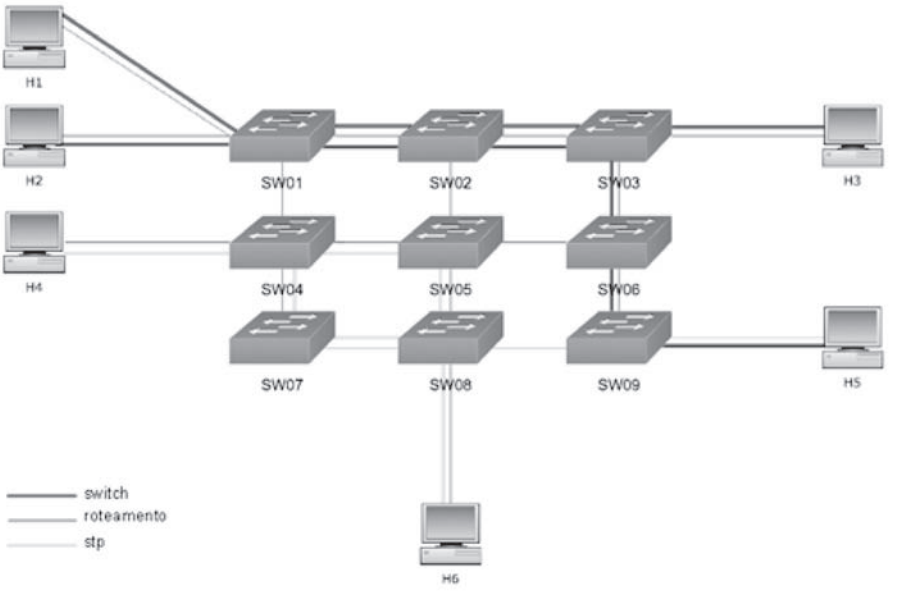


Figura 5.2 – *Topologia do plano de dados*

Para este trabalho optou-se pelo emprego de uma topologia em malha 3x3. O objetivo é ter um cenário com vários nós no qual seja possível alocar vários *slices* com pontos em comum entre si e com isso demonstrar o compartilhamento de recursos e o isolamento entre eles. Além disso, o cenário oferece um ambiente fortemente sujeito a problemas de *loops* que podem ser tratados via aplicação no plano de controle de cada *slice*. Para este cenário, alocou-se três *slices* com as aplicações de *switch*, roteamento e o algoritmo *SpanningTree*(stp). Nas extremidades dos nós, os hosts são utilizados para geração de tráfego no plano de dados por meio de requisições ICMP ou utilizando fluxos TCP ou UDP com uso da aplicação *Iperf*.

5.1.3. Plano de controle

O servidor Mininet que implementa o plano de dados também abriga o FlowVisor que é responsável pela instância de *slices* para cada experimento. Para este estudo de caso, como forma de demonstrar a implementação de regras de controle lógico sobre a rede virtualizada realizou-se a implementação de três *slices*, conforme apresentado na Figura 5.3.

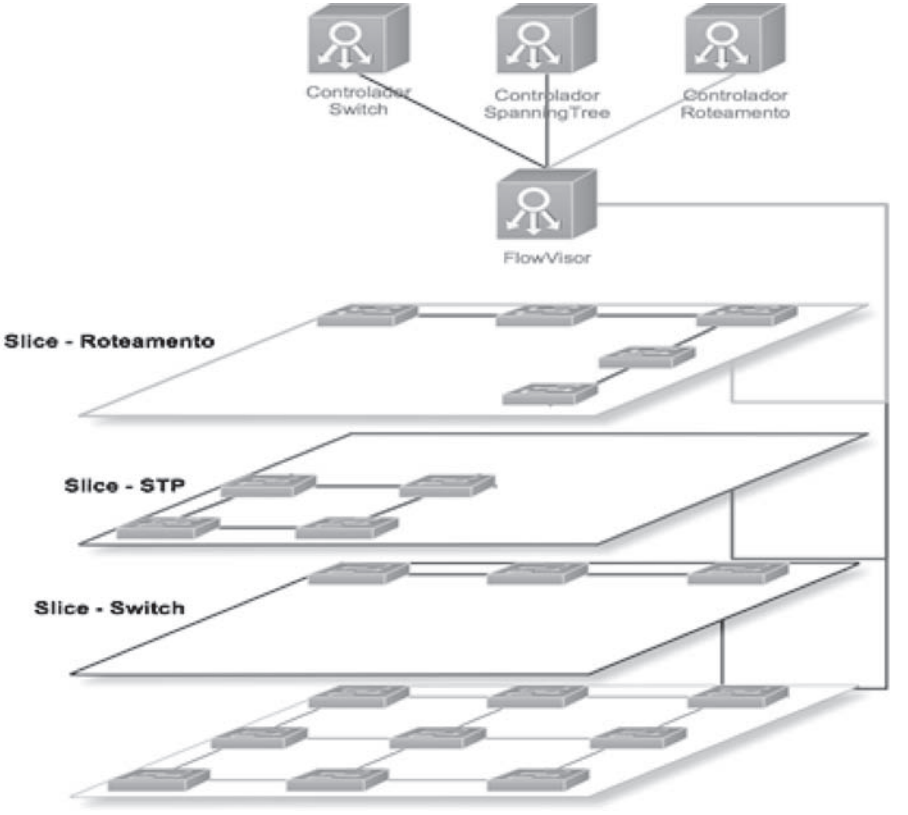


Figura 5.3 – *Slice dos experimentos*

Desse modo, representando o controle lógico implementado pelos pesquisadores, criou-se três instâncias do controlador NOX no servidor correspondente ao plano de controle. Associado a esses controladores NOX têm-se uma aplicação específica a ser executada que implementa o comportamento de encaminhamento no plano de dados.

Para o primeiro *slice*, o experimento da aplicação *switch* simula o comportamento de encaminhamento em camada 2 e faz com que os nós se comportem como um *switch*. No segundo *slice*, a aplicação *STP* aplica o algoritmo *SpanningTree* como solução para o problema gerado pela presença de *loops* no *slice*. Já no terceiro *slice*, a aplicação roteador faz com que os nós encaminhem pacotes via camada 3, empregando aos mesmos o comportamento de roteadores. No ambiente controlador NOX, em cada uma das VMs designada a um *slice*, executou-se respectivamente os seguintes comandos, invocando a aplicação correspondente.

```
# nox_core –v –u –i ptcp:1515 switch
# nox_core –v –u –i ptcp:1516 switch spanning_tree
# nox_core –v –u –i ptcp:1517 switch route
```

No ambiente onde se encontra a aplicação do FlowVisor, foram criados os *slices* intitulados como "**switch**", "**switch_stp**" e "**switch_router**". Cada um destes *slices* destina-se a conectar a sua respectiva instância de execução do controlador NOX. As instruções a seguir, utilizando o comando *fvctl,* efetivam a criação dos *slices*.

```
# fvctl createSlice switch tcp:192.168.100.1:1515 research_switch@
ufpa.br
# fvctl createSlice switch_stp tcp:192.168.200.1:1516 research_
switch_stp@ufpa.br
# fvctl createSlice switch_router tcp:192.168.300.1:1517 research_
router@ufpa.br
```

Ao final de cada entrada será solicitada uma senha específica para aquele *slice*. Como forma de prover um controle de acesso, esta senha pode ser utilizada para obter informações básicas relacionadas aos *slices* no FlowVisor. O comando a seguir e sua saída exemplificam este controle e os *slices* existentes.

```
# fvctl listSlices
Enter root's password:
Slice 0: root
Slice 1: switch
Slice 2: switch_stp
Slice 3: switch_router
```

5.2. Aplicação Prática

A aplicação prática apresenta o comportamento do que foi determinado nas aplicações que estão sendo executadas no controlador NOX. A primeira aplicação apenas efetua encaminhamento seguindo regras de *switch*; a segunda além de efetuar o encaminhamento, trata do aparecimento de *loops* na rede através do protocolo SpanningTree; e, a terceira, por fim, habilita o roteamento de pacotes dentro do *slice* alocado.

Slice Switch

Representando a solicitação de recursos realizada por um pesquisador, para o *slice switch*, foi designada a topologia representada pela Figura 5.4. Nesta topologia, ao fluxo de pacotes entres os *hosts* que estão ligados aos elementos de rede são implementadas as ações de um *switch*.

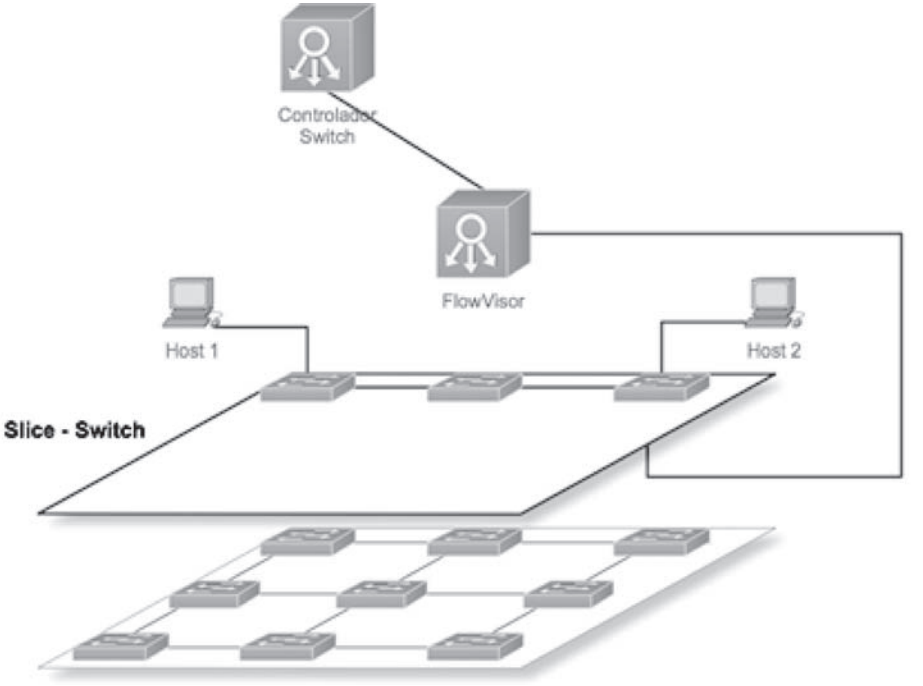


Figura 5.4 – *Slice Switch*

Nesta aplicação o comportamento de um switch consiste em analisar cada pacote e "aprender" sobre sua porta de origem, associando o endereço MAC de origem à porta onde o mesmo está conectado. Caso o destino do pacote já esteja associado a uma dada porta, o pacote será enviado diretamente, caso contrário, o *switch* realizará a ação de "flood" em todas as portas aguardando que o destino responda à requisição.

Para efetivar a alocação de recursos para o *slice*, deve-se implementar via linha de comando o espaço de fluxos que o compõe. A definição inclui a especificação dos nós que compõem o plano de dados e a caracterização do fluxo ali corrente. Os comandos a seguir implementam esta ação.

```
# fvctl addFlowSpace 00:00:00:00:00:01 10 dl_vlan=100
"Slice:switch=4"
# fvctl addFlowSpace 00:00:00:00:00:02 10 dl_vlan=100
"Slice:switch=4"
# fvctl addFlowSpace 00:00:00:00:00:03 10 dl_vlan=100
"Slice:switch=4"
```

Nos comandos acima se tem a adição dos nós SW01, SW02 e SW03 da topologia subjacente ao *slice* switch. Os nós são identificados pelos endereços MAC (*datapath id*) 00:00:00:00:00:01, 00:00:00:00:00:02 e 00:00:00:00:00:03, respectivamente. Este endereçamento é atribuído por meio do script que define a topologia cujo código-fonte está localizado no Apêndice A. Além disso, os comandos aplicam o isolamento de recursos por meio da identificação de *vlans*, neste caso igual a 100. Desse modo outros *slices* podem ser criados em paralelo utilizando os mesmos elementos do plano de dados, porém com uma caracterização de fluxos diferente.

Após a inclusão dos fluxos no FlowVisor os *switches* passam a ser reconhecidos pelo NOX e a executar as ações da aplicação instanciada no mesmo. Para ilustrar o funcionamento da aplicação *switch*, aplica-se um fluxo ICMP do tipo ECHO_REQUEST e ECHO_REPLY entre os hosts do *slice*. Na primeira tentativa de comunicação entre eles são trocadas informações de controle entre o *slice* e o controlador NOX. O controlador impõe então as políticas de fluxo que são implementadas nas tabelas dos nós pertencentes neste *slice* (Figura 5.5a).

Depois do primeiro estabelecimento de conexão entre os *hosts*, o tráfego de dados não flui mais através do controlador. Neste momento os dados são transmitidos com as informações contidas em cada *switch* e estes *switches* atualizam a tabela de fluxo do NOX com o status do enlace estabelecido (Figura 5.5a).

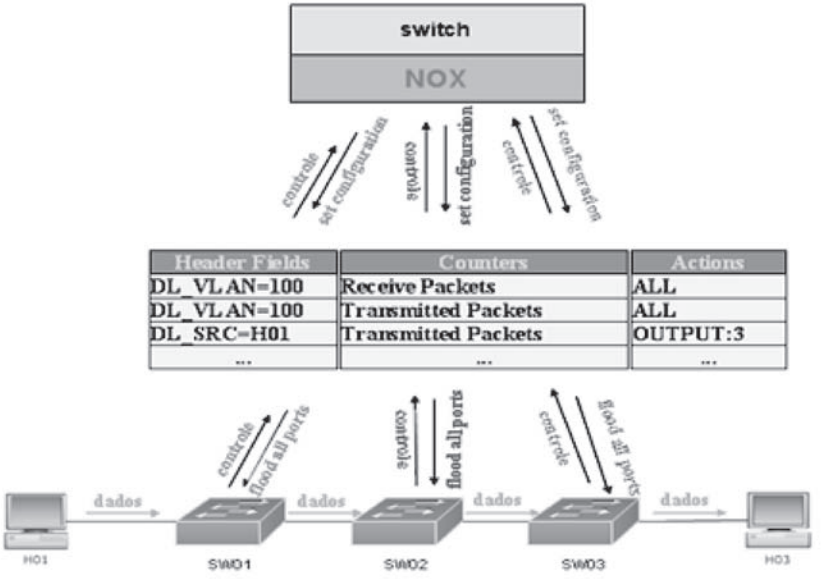


Figura 5.5a – *Troca de mensagens entre plano de dados e plano de controle*

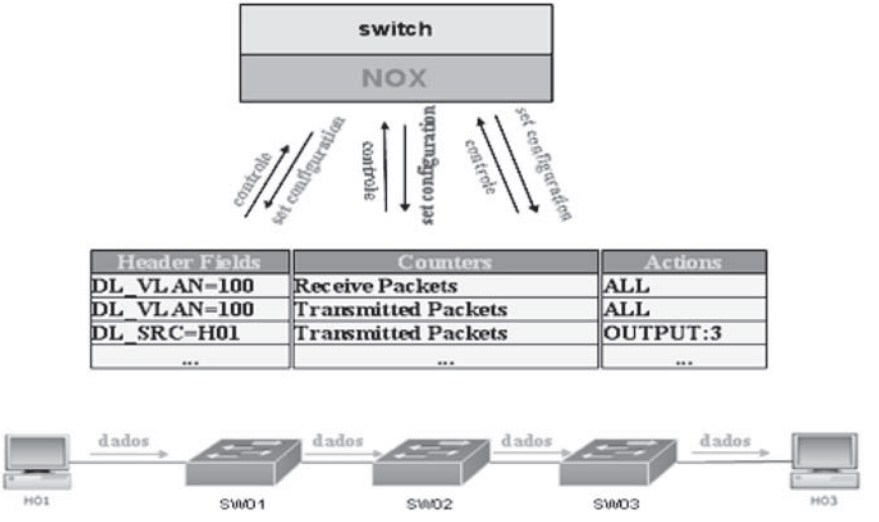


Figura 5.5b – *Encaminhamento sem troca de mensagens entre plano de dados e plano de controle*

Slice STP

Para o correto funcionamento de uma rede Ethernet, deve haver somente um caminho ativo entre dois sistemas finais. O protocolo STP, como um protocolo de gerenciamento de enlace, além de prover redundância por meio de caminhos alternativos entre estes sistemas, previne a existência de *loops* indesejáveis na rede. Para o *slice* denominado switch_stp, optou-se por implementar a topologia representada pela Figura 5.7. Nesta topologia tem-se um cenário no qual há presença de *loops* no caminho entre os *hosts* finais. O objetivo, neste caso, é tratar esta problemática por meio da implementação do algoritmo *SpanningTree* no ambiente controlador que é proprietário deste *slice*.

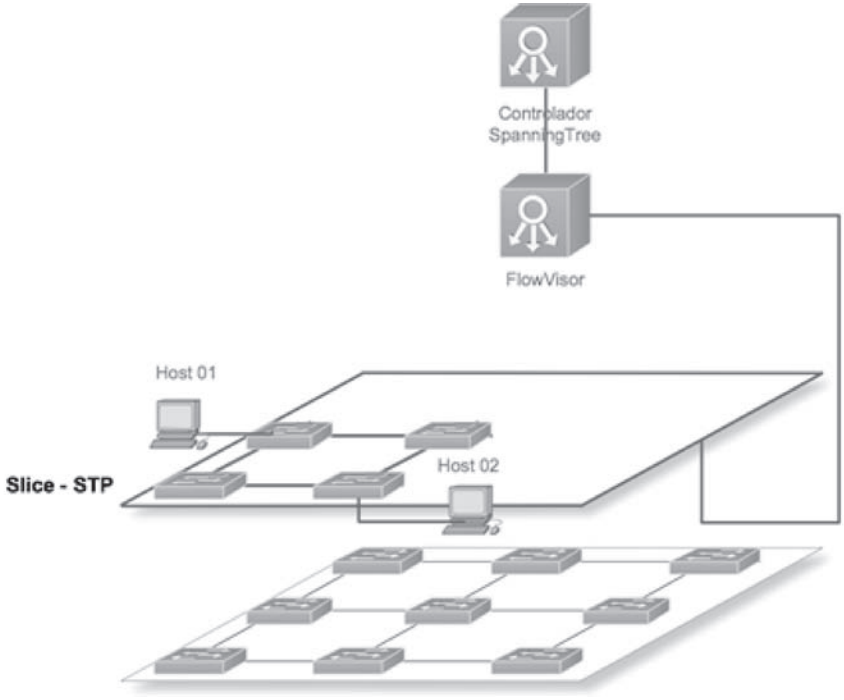


Figura 5.6 – *Slice STP*

No módulo STP implementado, quando um novo switch conecta-se ao controlador Nox, o módulo registra este momento e desabilita a operação de *flood* em todas as suas portas. Periodicamente o módulo realiza consultas a uma lista com todos os enlaces. Esta lista é utilizada para a construção da árvore do *Spanning Tree*. Os seguintes passos constroem esta estrutura:

- Todos os switches candidatos são alocados em uma lista e ordenados (crescentemente) por meio de seu DPID;
- O primeiro elemento (com o menor DPID) é removido da lista e definido como "switch raiz";

Nesse momento, todos os enlaces do switch raiz são examinados para que se identifique os switches de destino ligados a ele. O enlace para o qual o switch de destino não tem sido processado ainda nesta rodada é habilitado (adicionado à árvore *SpanningTree*). Somente um enlace deve levar a um dado destino. Caso haja múltiplos enlaces que levem a um switch de destino, o enlace mais rápido é escolhido e, considerando a existência de múltiplos enlaces com a mesma velocidade, o primeiro deles é escolhido.

Este procedimento é então executado para todos os switches candidatos que compõem a lista. O módulo *SpanningTree* utiliza-se do módulo de descoberta, chamado *Discovery*, para localizar as ligações dos switches com os seus nós vizinhos.

A alocação de recursos ao *slice* switch_stp é feita de maneira similar ao que foi implementado para *slice* switch. Para este fim, deve-se executar a seguinte sequência de comandos:

```
# fvctl addFlowSpace 00:00:00:00:00:04 10 dl_vlan=200
"Slice:switch_stp=4"
# fvctl addFlowSpace 00:00:00:00:00:05 10 dl_vlan=200
"Slice:switch_stp=4"
# fvctl addFlowSpace 00:00:00:00:00:07 10 dl_vlan=200
"Slice:switch_stp=4"
# fvctl addFlowSpace 00:00:00:00:00:08 10 dl_vlan=200
"Slice:switch_stp=4"
```

Neste experimento é possível observar a aplicação do *Spanning Tree* nos switches SW04, SW05, SW07 e SW08. Novamente os fluxos aplicados neste slice são caracterizados pela identificação de vlan cujo valor corresponde a 200. Pode-se observar que por meio da remoção do *loop* na topologia, há comunicação entre os hosts finais ligados diretamente aos *switches*.

Como realizado no experimento anterior, foi gerado um fluxo ICMP entre os hosts que utilizam este *slice*. Uma vez encontrada a topologia o algoritmo *SpanningTree* irá calcular a melhor forma de desfazer o *loop*, desabilitando umas das portas do switch que foi escolhida no cálculo. A Figura 5.8 mostra o momento que o controlador começa a efetuar *flood* em

portas e *non-flood* em outras para montar a árvore de nós. Nesta figura, os itens numerados de 1 a 5 mostram o processo de reconhecimento de características dos *switches* que compõem o *slice*, além da convergência da aplicação na habilitação (e não habilitação) de portas.

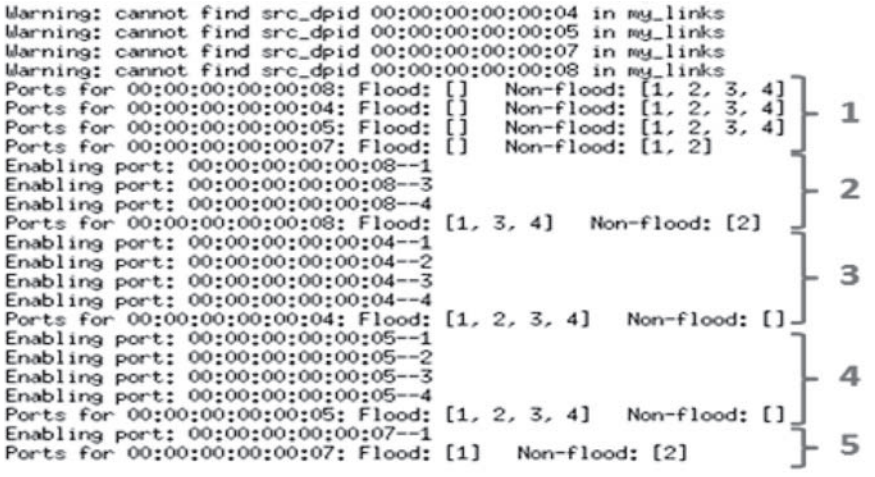


Figura 5.7

O item "1" corresponde à descoberta dos *switches*, neste momento o comando de *non-flood* é enviado em todas as suas portas para que se possa iniciar o mapeamento da árvore de nós. No item "2", o primeiro *switch* é selecionado, tem suas portas 1, 3 e 4 habilitadas com o envio de um *food* e a porta 2 desabilitada por meio de um *non-flood*. Deste modo o algoritmo segue para os próximos switches candidatos conforme itens "3", "4" e "5" até que a árvore esteja formada e o protocolo convergido completamente, habilitando assim o tráfego entre os dois *hosts*. Toda a comunicação pode ser verificada como descrita no primeiro *slice* através do software Wireshark.

Slice Roteamento

Neste último experimento os recursos alocados no *slice* switch_router são controlados via NOX utilizando a aplicação de roteamento. Esta aplicação possui a capacidade de aplicar aos nós o comportamento de roteadores, efetuando encaminhamento camada 3. A topologia ilustrada na Figura 5.11 exibe como os recursos de rede são visualizados pelo controlador NOX.

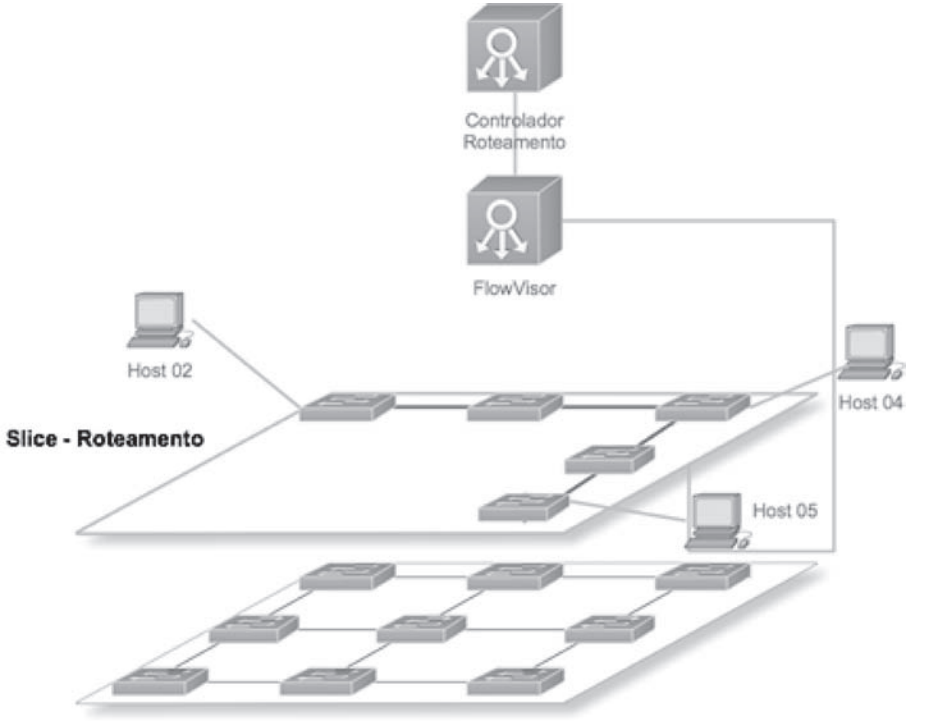


Figura 5.8 – *Slice Roteamento*

O módulo roteador é baseado em roteamento estático. Durante o funcionamento do módulo, cada nó da rede terá uma sub-rede configurada. Quando um pacote é destinado a um *host* que está nesta mesma sub-rede, o nó age como um *switch*, encaminhando o pacote sem alterações para uma porta conhecida ou de difusão. Caso um pacote seja destinado para um endereço IP no qual o roteador conhece o próximo salto, o mesmo modifica o cabeçalho e envia o pacote para a porta correta.

Para aplicar os nós que fazem parte do *slice* neste experimento, a configuração do espaço de fluxos para a aplicação switch_router recebe a seguinte implementação:

```
# fvctl addFlowSpace 00:00:00:00:00:01 10 dl_vlan=300
"Slice:switch_router=4"
# fvctl addFlowSpace 00:00:00:00:00:02 10 dl_vlan=300
"Slice:switch_router=4"
# fvctl addFlowSpace 00:00:00:00:00:03 10 dl_vlan=300
"Slice:switch_router=4"
```

fvctl addFlowSpace 00:00:00:00:00:04 10 dl_vlan=300 “Slice:switch_router=4”

fvctl addFlowSpace 00:00:00:00:00:06 10 dl_vlan=300 “Slice:switch_router=4”

Neste caso, aplicam-se os switches SW01, SW02, SW03, SW04 e SW06 ao *slice* switch_route. Cada host está endereçado com sub-redes diferentes por meio das seguintes configurações de endereçamento IP listadas pela Tabela 5.4:

Tabela 5.4 – *Endereçamento IP dos hosts ligados ao* slice

Host	Endereço IP	Sub-rede
Host01	0.0.0.0	0.0.0.0
Host02	0.0.0.0	0.0.0.0
Host03	0.0.0.0	0.0.0.0

Para fins de testes, por meio da aplicação do Iperf, utilizam-se dois fluxos de pacotes, sendo o primeiro UDP e o segundo TCP. Desta forma, entre os hosts 01 e 02 têm-se um fluxo TCP na porta 200 e entre os hosts 01 e 03 têm-se um fluxo UDP na porta 100. Na primeira tentativa de estabelecimento de comunicação entre os hosts são trocadas informações de controle, assim como informações dos dados a serem transmitidos de modo que a tabela de fluxo nos elementos do *slice* seja construída. Diferentemente do *slice switch*, os campos utilizados desta vez correspondem a informações da camada 3, correspondente a aplicação *routing* instanciada no controlador NOX (Figura 5.12a). Após a definição da tabela de fluxo, os dados são trafegados entre os hosts com informações contidas nos próprios switches e estes ficam responsáveis por atualizar a tabela de fluxo junto ao NOX com o status do enlace (Figura 5.12b).

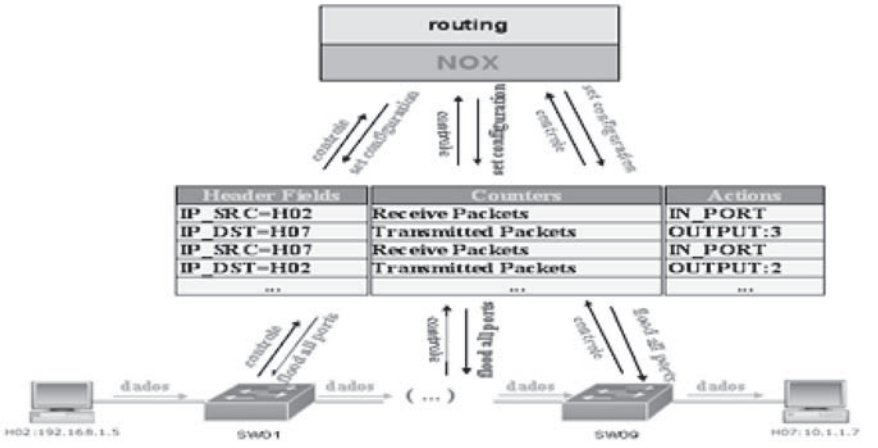


Figura 5.9a – *Slice Roteamento*

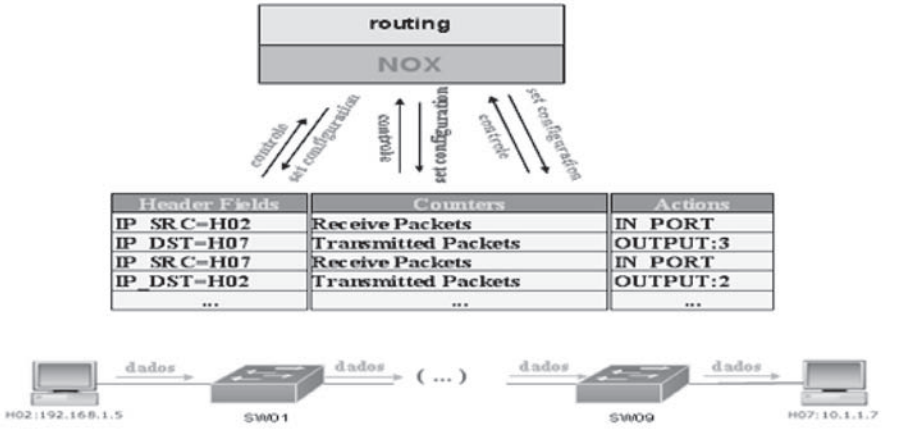


Figura 5.9b – *Slice Roteamento*

Esta aplicação de roteamento não é baseada em algoritmos específicos tais como de vetor de distância ou estado de enlace. Para que isso ocorra, tornando a aplicação mais sofisticada, faz-se necessário a implementação destes algoritmos via ambiente controlador. Como exemplos de esforços neste sentido, existe a aplicação QuagFlow [Nascimento et. al, 2011]. Por meio do QuagFlow é possível aplicar algoritmos de roteamento em modo de compatibilidade com o conjunto de aplicações Quagga [Quagga, 2011]. Desse modo, os mesmos algoritmos implementados para Quagga podem ser utilizados pelo NOX de modo que se possa atender aos requisitos mais exigentes de um ambiente real de execução.

5.3. Conclusão

O estudo de caso mostrou que de uma maneira rápida e simplificada é possível construir infraestruturas prontas para experimento de protocolos baseadas em redes OpenFlow. Além disso, é uma excelente ferramenta para a experimentação em Internet do Futuro (IF). Observou-se que com pequenos recursos é possível iniciar estes estudos para IF virtualizando todo plano de dados alinhados às necessidades de um ambiente real.

Além disso, observa-se que o Openflow associado ao FlowVisor oferece de uma maneira muito flexível a construção desse ambiente oferecendo uma API extensível para a criação de aplicações que modelem o comportamento dos fluxos no plano de dados.

Um ponto fraco da solução seria a não disponibilidade de interfaces de mais alto nível, como interfaces web que facilitem a interação com FlowVisor. O que deixa a sua manipulação fortemente dependente do administrador da rede para criação do plano de dados virtual.

6. Considerações finais e o futuro em Pesquisa Experimental em Internet do Futuro

6.1. Considerações finais

Observa-se que a Internet atual não conseguirá sustentar todos os desafios que lhe são impostos, por isso a remodelagem de sua arquitetura é um fato nesta próxima década. O processo de análise e desenvolvimento desta nova arquitetura já começou em muitos países com iniciativas GENI, FIRE e FIBRE.

Neste contexto, a virtualização tornou-se a principal ferramenta para pesquisa de desenvolvimento e habilitação da Internet do Futuro em todas as comunicações ou camadas computacionais dessas grandes infraestruturas. Com o uso da virtualização na construção desses *testbeds* foi possível desacoplar a complexidade dos recursos físicos dos serviços oferecidos e apresentar simples interfaces com usuário, em localizações geográficas distintas e independentes de dispositivo de acesso.

Além disso, a virtualização terá um papel importante, pois ela permitirá a comparação das novas tecnologias que estão sendo desenvolvidas para Internet do Futuro, com as tecnologias atuais. Também garantirá que tecnologias desenvolvidas no passado possam ser disponíveis nessa moderna infraestrutura.

Com isso, a construção dos frameworks para esses *testbeds* está sendo desenvolvida com base no conceitos de redes definidas por software (*SDN*) e virtualização, seja ela baseada na virtualização do substrato, seja na infraestrutura física que contém todos os hardwares e softwares para inicializar os recursos virtuais, bem como para a criação da infraestrutura virtual contendo os recursos virtuais e a topologia lógica, interligando-as.

O framework OpenFlow é uma dessas soluções, pois provê um protocolo aberto para programação do comportamento de fluxos de pacote em diferentes switches e roteadores. A rede contendo OpenFlow pode ter o tráfego particionado entre tráfego de produção e tráfego de experimentação, de maneira que pesquisadores possam controlar seus próprios fluxos. A rede de produção permanecerá isolada e processada da mesma maneira que antes do uso do OpenFlow.

Ainda, o OpenFlow integrado com a ferramenta FlowVisor é capaz de criar e virtualizar elementos de redes de modo que uma mesma infraestrutura física possa ser compartilhada entre várias topologias lógicas, onde cada um possui sua própria lógica de encaminhamento e tratamento de fluxos de pacotes distintos.

Por fim, o OpenFlow possibilita de maneira rápida e simples a criação de uma infraestrutura para concepção de novos protocolos, como apresentada na seção de estudos de casos onde em um único servidor, conseguimos simular uma infraestrutura de switches e roteadores OpenFlow. Também conseguimos aplicar os conceitos de virtualização baseada em slice de recursos e redes programáveis.

Este capítulo apresentou o andamento da pesquisa experimental em Internet do Futuro no mundo e observou duas grandes tecnologias que estarão presentes nessas infraestruturas montadas em escala mundial. São elas: a virtualização e o framework OpenFlow.

6.2. Tendências futuras

Em uma visão geral sobre as tendências futuras, observa-se que essa corrida pela tão sonhada "Internet 3.0" ainda está em sua fase inicial, porém se expandindo extremamente rápida entre os continentes no mundo. No topo da corrida, estão a América do Norte, Europa e Japão, seguidos do Brasil.

Na américa do norte, o projeto GENI está iniciando a sua terceira fase, chamada espiral 3 (*Spiral* 3). As tendências, nesta terceira fase, serão iniciar o suporte de pesquisadores nas infraestruturas de experimentação

e a incrementação de recursos disponíveis para uso entre seus usuários. Além disso, ainda irá continuar os esforços na implementação, integração e início das operações com switches OpenFlow e estações base WiMax.

Já na Europa, o projeto FIRE, que está entrando na chamada "first wave", está investindo 40 bilhões de euros para criação de *testbed* sustentáveis em internet do futuro. Neste momento as convergências do projeto estão na busca pela estabilização das suas infraestruturas de experimentação, além de iniciar as chamadas por parceiros para a criação de estudos de casos a serem aplicados em seus ambientes.

No Japão, representado pelo projeto AKARI [AKARI, 2009], que está em seu segundo período de desenvolvimento, a tendência futura é iniciar as construções de seus *testbeds* e as primeiras demonstrações experimentais. O objetivo para o final de 2016 é oferecer um protótipo da nova geração de rede para disponibilidade aos pesquisadores.

Por fim, no Brasil, a pesquisa em Internet do futuro iniciou-se através de projetos isolados, como o projeto *Horizon* [HORIZON, 2011] e o projeto *Webscience* [WEBSCIENCE, 2011], mas efetivamente se fortaleceu através do projeto FIBRE, de cooperação entre o Brasil e a União Europeia, que prevê a construção de uma infraestrutura experimental brasileira compartilhada de larga escala, que permita a experimentação em infraestrutura de rede e aplicações distribuídas. Isso consistirá em um novo ambiente de teste baseado em OpenFlow no Brasil e uma melhoria da instalação do projeto OFELIA, atualmente em desenvolvimento na Europa.

Além disso, possibilitará a federação de ambientes de teste dos parceiros brasileiros e europeus para permitir aos pesquisadores que usem recursos de ambos em um mesmo experimento. Tal iniciativa demonstrará o potencial das instalações ao mostrar aplicações baseadas em OpenFlow, estabelecidas sobre os recursos das instalações federadas. Portanto, aumentará a colaboração e a troca de conhecimentos entre pesquisadores europeus e brasileiros no campo de Internet do Futuro.

Referências

David C. Clack (2009). Toward the design of a future Internet. Technical Report: National Science Fundation. Disponível em: <http://groups.csail.mit.edu/ana/People/ DDC/Future%20Internet%207-0.pdf>.

Anja Feldmann. Internet clean-slate design: what and why?. *SIGCOMM Comput. Commun. Rev.* 37, 3 (July 2007), 59-64.

Banniza, T.-R., Boettle, D., Klotsche, R., Schefczik, P., Soellner, M. and Wuenstel, K. (2009), A European approach to a clean slate design for the future Internet. Bell Labs Technical Journal, 14: 5–22.

Jennifer Rexford and Constantine Dovrolis. 2010Es. Future Internet architecture: clean-slate versus evolutionary research. *Commun. ACM* 53, 9 (September 2010), 36-40.

Subharthi Paul, Jianli Pan, Raj Jain. Architectures for the Future Networks and the Next Generation Internet: A Survey Computer Communications, Vol. 34, No. 1. (15 January 2011), pp. 2-42.

GENI. Global Environment for Network Innovations, Disponível em: <http://www.geni.net. Acessado em 20/10/2012>.

FIND. Future Internet Design. Disponível em: <http://www.nets-find.net>. Acessado em: 20/10/2012.

FIBRE. Future Internet Experimentation between Brazil and Europe. Disponível em: <http://www.fibre.org.br>. Acessado em 20/10/2012.

FIRE. Future Internet Research and Experimentation. Disponível em: <http://cordis.eur opa.eu/fp7/ict/fire/>. Acessado em 20/10/2012.

AKARI project (2009), "New Generation Network Architecture: AKARI Conceptual Design" (ver1.1), Disponível em: <http://akari-project.nict.go.jp/eng/concept-design/>. AKARI <_fulltext_e_preliminary.pdf>. Acessado em 20/02/2011.

CFI 2010: 5th International Conference on Future Internet Technologies (em conjunto com 2nd Future Internet Testbed & Research Workshop), June 16-18, 2010, Seoul, Korea, <http://as.kaist.ac.kr/cfi10/>. Acessado em 3/1/2011.

Jun Bi, "Future Internet related Research Activities in China", 30th APAN Meeting, August 2010, Hanoi, Vietnam. Disponível em: <http://www.apan.net/meetings/Hanoi 2010/Session/Slides/FutureInternet/3-1.pdf>. Acessado em 20/02/2011.

Choudhury, N.M.M.K., and Boutaba, R. (2010), "A survey of network virtualization", In: Computer Networks, Volume 54, Março 2010, pp. 862-876.

Nick McKeown, Tom Anderson, Hari Balakrishnan, Guru Parulkar, Larry Peterson, Jennifer Rexford, Scott Shenker, and Jonathan Turner. 2008. OpenFlow: enabling innovation in campus networks. *SIGCOMM Comput. Commun. Rev.* 38, 2 (March 2008), 69-74.

R. Jain. Internet 3.0: Ten Problems with Current Internet Architecture and Solutions for the Next Generation. *Military Communications Conference, 2006. MILCOM 2006*, pages 1-9, October 2006.

Michael Stanton; Iara Machado. Redes de Pesquisa e Educação: Cenário do Futuro. Disponível em: <http://www.ic.uff.br/~michael/futuro-da-Internet/redes-de-pesquisa-2009.pdf>. Acessado em: 20/02/2011.

Subharthi Paul, Jianli Pan, and Raj Jain, "Architectures for the Future Networks and the Next Generation Internet: A Survey," Computer Communications, UK, Volume 34, Issue 1, 15 January 2011, Pages 2-42.

Joseph Williams, Lewis Curtis, "Green: The New Computing Coat of Arms?", IT Professional, pp. 12-16, January/February, 2008.

I. Stoica, D. Adkins, S. Zhuang, et al, "Internet Indirection Infrastructure," ACM SIGCOMM, Pittsburgh, PA, 2002, Disponível em: <http://i3.cs.berkeley.edu/publications/papers/i3-sigcomm.pdf>.

H. Balakrishnan, et al, "A Layered Naming Architecture for the Internet," SIGCOMM 2004, pp. 343-352.

R. Moskowitz, P. Nikander, "Host Identity Protocol Architecture," Internet Draft, August 1, 2005, draft-ietf-hip-arch-03, 24 pp.

R. Moskowitz, P. Nikander, P. Jokela, T. Henderson, "Host Identity Protocol," Internet Draft, October 24, 2005, draft-ietf-hip-base-04, 99 pp.

GENI (2008). GENI system overview. Relatório Técnico. Disponível em: <http://groups.geni.net/geni/attachment/wiki/GeniSysOvrvw/GENISysOvrvw092908.pdf>.

Spiral 2 (2010), GENI Spiral 2 Overview. Relátorio Técnico. Disponível em: <http://groups.geni.net/geni/attachment/wiki/SpiralTwo/GENIS2Ovrvw060310.pdf>.

B Boehm. 1986. A spiral model of software development and enhancement. *SIGSOFT Softw. Eng. Notes* 11, 4 (August 1986), 14-24.

FIRE (2010). Future Internet Research and Experimentation: An Overview of European FIRE Initiative and its projects. Disponível em: http://cordis.europa.eu/fp7/ict/fire/docs/fire-brochure-2010_en.pdf.

FIRE (2008). Future Internet Research and Experiment: AREA 6. Disponível em: <http://www.future-Internet.eu/fileadmin/documents/bled_documents/experiemental_fac ilities/FIRE-Issues-paper-2.2.pdf>.

WebScience. Brazilian Institute for Web Science Research. Disponível em: <http://webscience.org.br/files/INCTportugues2010.pdf>. Acessado em: 20/02/2011.

Scarabucci, R.R., et al. (2005), "Project GIGA – High-speed Experimental Network", In: First International Conference on Testbeds and Research Infrastructures for the DEvelopment of NeTworks and COMmunities (TRIDENTCOM'05), Trento, Itália, 02/2005, p. 242-251.

D.Young Kim, L.Mathy, M. Campanella, R. Summerhill, J. Williams, S. Shimojo, Y. Kitamura, H. Otsuki, "Future Internet: Challenges in Virtualization and Federation" aict, pp.1-8, 2009 Fifth Advanced International Conference on Telecommunications, 2009.

M.Campanella, "The FEDERICA Project: creating cloud infrastructures", In Proceedings of Cloudcomp 2009 (CloudComp), October 19-21, 2009, Munich, Germany.

Thrasyvoulos S.; Serge F.; Scott K. 2007. Future Internet: fundamentals and measurement. *SIGCOMM Comput. Commun. Rev.* 37, 2 (March 2007).

P. Sezgedi, S. Figuerola, M. Campanella, V. Maglaris, C. Cervello-Pastor: "With Evolution for Revolution: Managing FEDERICA for Future Internet Research", IEEE Communications Magazine, Vol.47, No.7, pp. 34-39, July 2009.

G. Di Stasi, R. Bifulco, F. P. D'Elia, S. Avallone, R. Canonico, A. Apostolaras, N. Giallelis, T. Korakis and L. Tassiulas. Experimenting with P2P traffic optimization for Wireless Mesh Networks in a federated OMF-PlanetLab environment. WCNC 2011, IEEE Wireless Communications & Networking Conference. Cancun (Mexico): March 28-31, 2011.

Eric Eide, Leigh Stoller, Tim Stack, Juliana Freire, and Jay Lepreau. 2006. Integrated scientific workflow management for the Emulab network testbed. In *Proceedings of the annual conference on USENIX '06 Annual Technical Conference* (ATEC ‹06). USENIX Association, Berkeley, CA, USA, 33-33.

L. Peterson, S. Sevinc, J. Lepreau, et al., "Slice-Based Facility Architecture, Draft Version 1.04," April 7, 2009, <http://svn.planet-lab.org/attachment/wiki/GeniWrapper/ sfa.pdf>.

ProtoGENI. Disponível em: <http://groups.geni.net/geni/wiki/ProtoGENI>. Acessado em: 20/02/2011.

ORCA, Open Resource Control Architecture Disponível em: <http://groups.geni.net/ geni/wiki/ORCABEN>. Acessado em: 20/02/2011.

BEN. Breakable Experimental Network. Disponível em: <https://geni-orca.renci.org/trac>. Acessado em: 20/02/2011.

OMF. Control and Management Framework. Disponível em: <http://omf.mytestbed.net>. Acessado em: 20/02/2011.

P. Antoniadis, T. Friedman, X. Cuvellier, "Resource Provision and Allocation in Shared Network Testbed Infrastructures," ROADS 2007, Warsaw, Poland, July 11-12, 2007.

John W. Lockwood, Nick McKeown, Greg Watson, Glen Gibb, Paul Hartke, Jad Naous, Ramanan Raghuraman, and Jianying Luo. 2007. NetFPGA--An Open Platform for Gigabit-Rate Network Switching and Routing. In *Proceedings of the 2007 IEEE International Conference on Microelectronic Systems Education* (MSE ‹07). IEEE Computer Society, Washington, DC, USA, 160-161.

Muhammad Bilal Anwer and Nick Feamster. 2009. Building a fast, virtualized data plane with programmable hardware. In *Proceedings of the*

1st ACM workshop on Virtualized infrastructure systems and architectures (VISA ‹09). ACM, New York, NY, USA, 1-8.

Indigo, Disponível em: <http://www.OpenFlowhub.org/display/Indigo/Indigo+-+Open Flow+for+Hardware+Switches>. Acessado em: 20/02/2011.

Kate Greene (2009-04), "TR10: Software-Defined Networking", MIT Technology Review.

Brian Tierney, Jeff Boote, Eric Boyd, Aaron Brown, Maxim Grigoriev, Joe Metzger, Martin Swany, Matt Zekauskas, Yee-Ting Li, and Jason Zurawski, Instantiating a Global Network Measurement Framework,LBNL Technical Report LBNL-1452E. Disponível em: <http://acs.lbl.gov/~tierney/papers/perfsonar-LBNL-report.pdf>. Acessado em janeiro de 2009.

Y. P. Chen, A. L. Liestman and J. Liu. A Hierarchical Energy-Efficient Framework for Data Aggregation in Wireless Sensor Networks, *IEEE Transactions on Vehicular Technology*, 55(3):789-796, May 2006.

Larry Peterson, Steve Muir, Timothy Roscoe, Araon Klingaman. PlanetLab Architecture: An Overview. Documento Técnico, maio 2006. Disponível em: <http://www.planet-lab.org/files/pdn/PDN-06-031/pdn-06-031.pdf>.

BonFIRE. Building service testbeds on Future Internet Research and Experimentation. Disponível em: <http://www.bonfire-project.eu/project>. Acessado em: 20/02/2011.

S. Savage, T. Anderson, A. Aggarwal, D. Becker, N. Cardwell, A. Collins, E. Hoffman, J. Snell, A. Vahdat, G. Voelker, J. Zahorjan, Detour: a case for informed internet routing and transport, IEEE Internet Computing 19 (1) (1999) 50–59.

D. Andersen, H. Balakrishnan, F. Kaashoek, R. Morris, Resilient overlay networks, SIGOPS Operating Systems Review 35 (5) (2001), 131–145.

J. Jannotti, D.K. Gifford, K.L. Johnson, M.F. Kaashoek, J. James, W. O'Toole, Overcast: reliable multicasting with an overlay network, in: Proceedings of the Fourth Conference on Symposium on Operating System Design & Implementation (OSDI'00), 2000, pp. 197–212.

Y. Chu, S. Rao, S. Seshan, H. Zhang, Enabling conferencing applications on the internet using an overlay multicast architecture, SIGCOMM Computer Communication Review 31 (4) (2001) 55–67.

L. Subramanian, I. Stoica, H. Balakrishnan, R. Katz, OverQoS: an overlay based architecture for enhancing internet QoS, in: Proceedings of the First Symposium on Networked Systems Design and Implementation (NSDI), 2004, pp. 71–84.

A. Keromytis, V. Misra, D. Rubenstein, SOS: secure overlay services, in: Proceedings of the ACM SIGCOMM Conference (SIGCOMM'02), 2002.

D.G. Andersen, Mayday: distributed filtering for internet services, in: Proceedings of the Fourth Conference on USENIX Symposium on Internet technologies and Systems (USITS'03), USENIX Association, Berkeley, CA, USA, 2003.

B. Krishnamurthy, C. Wills, Y. Zhang, On the use and performance of content distribution networks, in: Proceedings of the First ACM SIGCOMM Workshop on Internet Measurement (IMW'01), ACM, New York, NY, USA, 2001, pp. 169–182.

E.K. Lua, J. Crowcroft, M. Pias, R. Sharma, S. Lim, A survey and comparison of peer-to-peer overlay network schemes, IEEE Communications Surveys & Tutorials 7 (2) (2005) 72–93.

F. Dabek, M.F. Kaashoek, D. Karger, R. Morris, I. Stoica, Wide-area cooperative storage with CFS, in: Proceedings of the 18th ACM Symposium on Operating Systems Principles (SOSP'01), ACM, New York, NY, USA, 2001, pp. 202–215.

L. Peterson, T. Anderson, D. Culler, T. Roscoe, A blueprint for introducing disruptive technology into the Internet, SIGCOMM Computer Communication Review 33 (1) (2003) 59–64.

Sherwood, Rob; Glen, Gibb; Kobayashi Masayoshi. Carving Rearch Slices Out of Your Production Networks with OpenFlow. ACM SIGCOMM Computer Communication Review, Volume 40, Janeiro 2010.

Rob Sherwood, Glen Gibb, Kok-Kiong Yap, Guido Appenzeller, Martin Casado, Nick McKeown, and Guru Parulkar. 2010. Can the production network be the testbed?. In *Proceedings of the 9th USENIX conference on Operating systems design and implementation* (OSDI›10). USENIX Association, Berkeley, CA, USA, 1-6.

R. Sherwood, G. Gibb, K. Yap, G. Appenzeller, N. McKeown, and G. Parulkar. "FlowVisor: A network virtualization layer", Relatório Técnico, 2009. Disponível em: <http://OpenFlowswitch.org/downloads/technicalreports/OpenFlow-tr-2009-1flowvisor.pdf>.

CREW. Cognitive Radio Experimentation World. Disponível em: <http://www.crew-project.eu/>. Acessado em: 20/02/2011.

OFELIA. OpenFlow in Europe – Linking Infrastructure and Applications. Disponível em: <http://www.fp7-ofelia.eu/>. Acessado em: 20/02/2011.

UMF. Embedding real-time substrate measurements for cross-layer communications. Disponível em: <http://groups.geni.net/geni/wiki/Embedded%20Real-Time%20Measurements>. Acessado em: 20/02/2011.

Matthias Bolte, Michael Sievers, Georg Birkenheuer, Oliver Niehérster, and André; Brinkmann. 2010. Non-intrusive virtualization management using libvirt. In *Proceedings of the Conference on Design, Automation and Test in Europe* (DATE ‹10). European Design and Automation Association, 3001 Leuven, Belgium, Belgium, 574-579.

Song Wu, Li Deng, Hai Jin, Xuanhua Shi, Yankun Zhao, Wei Gao, Jianyin Zhang, and Jin Peng. 2010. Virtual Machine Management Based on Agent Service. In *Proceedings of the 2010 International Conference on Parallel and Distributed Computing, Applications and Technologies* (PDCAT '10). IEEE Computer Society, Washington, DC, USA, 199-204.

Mazhar B. Tayel and Ashraf A. Taha. 2008. Effect of TCP and UDP parameters on the quality of video streaming delivery over the internet. *WTOC* 7, 6 (June 2008), 653-662.

S. Shalunov. Thrulay – Network capacity tester. Disponível em: <http://shlang.com/thrulay/>. Acessado em: 20/02/2011.

B. Fink, R. Scott. NUTTCP. Disponível em: <ftp://ftp/lcp.nrl.navy.mil/pub/nuttcp/2006>. Acesado em: 20/02/2011.

en-Wei Hu, Hui-Min Chen, Te-Lung Liu, Hui-Min Tseng, Daniel Lin, Chu--Sing Yang, and C. Eugene Yeh. 2010. Implementation of Alarm Correlation System for Hybrid Networks Based upon the perfSONAR Framework. In *Proceedings of the 2010 IEEE 24th International Conference on Advanced Information Networking and Applications Workshops* (WAINA ‹10). IEEE Computer Society, Washington, DC, USA, 893-898.

Ps-Performace. pS-Performace ToolKit. Disponível em: <http://www.internet2.edu/ performance/toolkit>. Acessado em: 20/02/2011.

Daniel Nurmi, Rich Wolski, Chris Grzegorczyk, Graziano Obertelli, Sunil Soman, Lamia Youseff, and Dmitrii Zagorodnov. 2009. The Eucalyptus Open-Source Cloud-Computing System. In*Proceedings of the 2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid* (CCGRID ‹09). IEEE Computer Society, Washington, DC, USA, 124-131.

E-GENI. Enterprise GENI. Disponível em: <http://OpenFlow.org/wk/index.php/E-GENI>. Acessado em: 20/02/2011.

Nick McKeown. *Clean Slate Research Program*. Disponível em: <http://cleanslate.stanford.edu/>. Acessado em: 20/02/2011.

Horizon. Horizon Project: A New Horizon to The Internet. Disponível em: <http://www.gta.ufrj.br/horizon/>. Acessado em: 20/02/2011.

Marcelo Nascimento, Christian Rothenberg, Marcos Salvador, and Mauricio Magalhães. 2010. *QuagFlow: Partnering Quagga with OpenFlow*. SIGCOMM'10 New Delhi, India.

Quagga Routing Suite. Disponível em: <http://www.quagga.net/>. Acessado em: 14/03/2011.

Beyonder – mobilidade digital livre com PHP

Flávio Gomes da Silva Lisboa

Resumo

Beyonder é um projeto de software livre, poliglota, multiplataforma, aderente a padrões, voltado para o réuso de soluções e integrado com as comunidades de software livre e aberto. Seu objetivo é disponibilizar um módulo acoplável de reconhecimento de dispositivos móveis para seleção da interface com o usuário em aplicações Web utilizando soluções livres e um módulo que permita a execução local de aplicações Web utilizando recursos de HTML 5 e Javascript.

1.1. Introdução

Um dos principais destaques da pesquisa TIC Domicílios 2011, realizada pelo Comitê Gestor da Internet do Brasil (CETIC.BR, 2012) foi o avanço das tecnologias móveis. Alguns fatos apurados pela pesquisa foram os seguintes:

- o telefone fixo apresenta estabilidade desde 2008;
- o celular passa o rádio e se torna a segunda mídia mais presente no domicílio;
- os notebooks avançam nos domicílios;
- o aumento da penetração dos notebooks nos domicílios sugere um processo de substituição;

- o portátil já é o primeiro computador em 9% dos domicílios com computador;
- o aumento da posse dos computadores portáteis se deu em todas as classes sociais;
- Em 2011, a proporção de portáteis e computadores se iguala na classe A;
- Em 2011, a banda larga móvel ultrapassa o acesso discado pela primeira vez;
- A banda larga móvel é a principal responsável pelo crescimento da banda larga.
- O celular caminha para ser uma mídia universalizada entre a população;
- As proporções de posse e uso do celular estão cada vez mais próximas;
- O envio de mensagens de texto e fotos pelo celular apresenta estabilidade;
- Acesso à Internet via telefone celular cresce de forma expressiva;
- A diferença na proporção de pré-pagos e pós-pagos apresenta pequena redução;
- Os planos de Internet para pré-pagos são responsáveis por 81% do crescimento do uso da Internet nos celulares pré-pagos.

A ascensão do uso das tecnologias móveis pela população em geral indica que os dispositivos móveis tornam-se um meio preferencial de comunicação. O governo deve observar esse fato e procurar disponibilizar serviços, para o cidadão, que sejam acessíveis por dispositivos móveis.

Devemos observar que não estamos tratando do governo disponibilizar novos serviços para o cidadão, mas sim de oferecê-los por meio de um novo canal de comunicação. Dessa forma, coloca-se a questão da necessidade ou não de refazer sistemas de informação para operarem nesse novo paradigma, e o custo envolvido nisso.

Este é o cenário dentro do qual será apresentado o projeto *Beyonder*.

1.2. O projeto

O projeto *Beyonder* tem em seu escopo dois objetivos a serem alcançados:

- criar um módulo acoplável de reconhecimento de dispositivos móveis para seleção da interface com o usuário em aplicações Web utilizando soluções livres.

- criar um módulo que permita a execução local de aplicações Web utilizando recursos de HTML 5 e Javascript.

A motivação para alcançar esses dois objetivos têm um fundamento econômico, que se justifica pela austeridade que a administração pública deve praticar na condução de suas atividades, com orientação de utilizar sempre os recursos públicos da melhor forma: atendendo a população com o mínimo de gastos possíveis.

Com relação à provisão de reconhecimento de dispositivos móveis em aplicações Web, temos de considerar o fato de que adaptar aplicações para serem acessíveis por dispositivos móveis evita gastos com novos desenvolvimentos.

Com relação à execução local de aplicações Web em dispositivos móveis, isso permite democratizar o acesso aos serviços, não limitando os aparelhos que podem ser utilizados e evitando linhas de produção paralelas para os mesmos aplicativos.

Como os dois objetivos envolvem produtos distintos, o projeto foi subdividido em dois subprojetos, denominados Cyborg e Omens. Eles serão descritos com mais detalhes a seguir.

1.2.1. Cyborg

O subprojeto *Cyborg* consiste em um ambiente de desenvolvimento livre para criação de aplicações embarcadas em dispositivos móveis. Seu objetivo é ir além das iniciativas atuais de desenvolvimento de aplicações embarcadas para governo, como os aplicativos disponíveis para Android e iOs da Receita Federal.

Os desenvolvimentos atuais de soluções instaladas em celulares e *tablets*, como os citados anteriomente, limitam-se a dois sistemas operacionais, o que obriga o cidadão a ter de usar um determinado produto de software se quiser ter acesso aos serviços móveis. Não existe liberdade de escolha.

O objetivo do projeto *Beyonder Cyborg* é criar uma solução geral para desenvolvimento de aplicações que possam ser executadas localmente sem que tenham de ser nativas de determinado(s) sistema(s) operacional(s), dando ao cidadão a liberdade de escolha, pois ninguém deveria ser obrigado a adquirir determinada marca de produto tecnológico para ter acesso a serviços públicos. Além de democratizar o acesso, evita-se o custo de manter linhas de produção paralelas, onde o mesmo aplicativo

tem de ser desenvolvido para plataformas diferentes, porque eles não são portáveis.

1.2.2. Omens

Enquanto o subprojeto Cyborg trata da execução local de aplicações Web, e está em um contexto em que pode se pensar em baixa conectividade, o subprojeto *Omens* trata de um contexto mais amplo, ao lidar com aplicações acessíveis por dispositivos móveis largamente conectados.

Uma vez que os dispositivos móveis possuem navegadores e conexão com Internet, as aplicações Web também podem ser utilizadas a partir de celulares e *tablets*. Ocorre, porém, que as aplicações existentes não foram originalmente pensadas para isso. O resultado é que muitos sítios e aplicativos são inoperáveis por determinados dispositivos móveis.

Não é necessário criar novas aplicações Web para serem acessíveis por dispositivos móveis. A questão aqui está na interface. É necessário identificar qual o aparelho que está solicitando dados das aplicações e apresentar as telas, ou melhor, páginas, mais adequadas àquele aparelho, que inclusive saibam quais recursos do navegador podem ser utilizados.

O objetivo deste projeto é prover um componente que faça a identificação do dispositivo solicitante e selecione uma interface adequada, sem que seja necessário alterar as demais camadas da aplicação Web.

Novamente, não se trata de criar algo do zero, mas de estender e integrar projetos de software livre que já existem. O projeto Tera-WURFL provê uma acurada detecção de dispositivos móveis com rápida distinção entre Desktop e Mobile, utilizando uma base de dados de alta performance. Esse projeto pode ser integrado com outros que tratam de criação de interfaces específicas para dispositivos móveis, como o Dojo Toolkit e o Jquery.

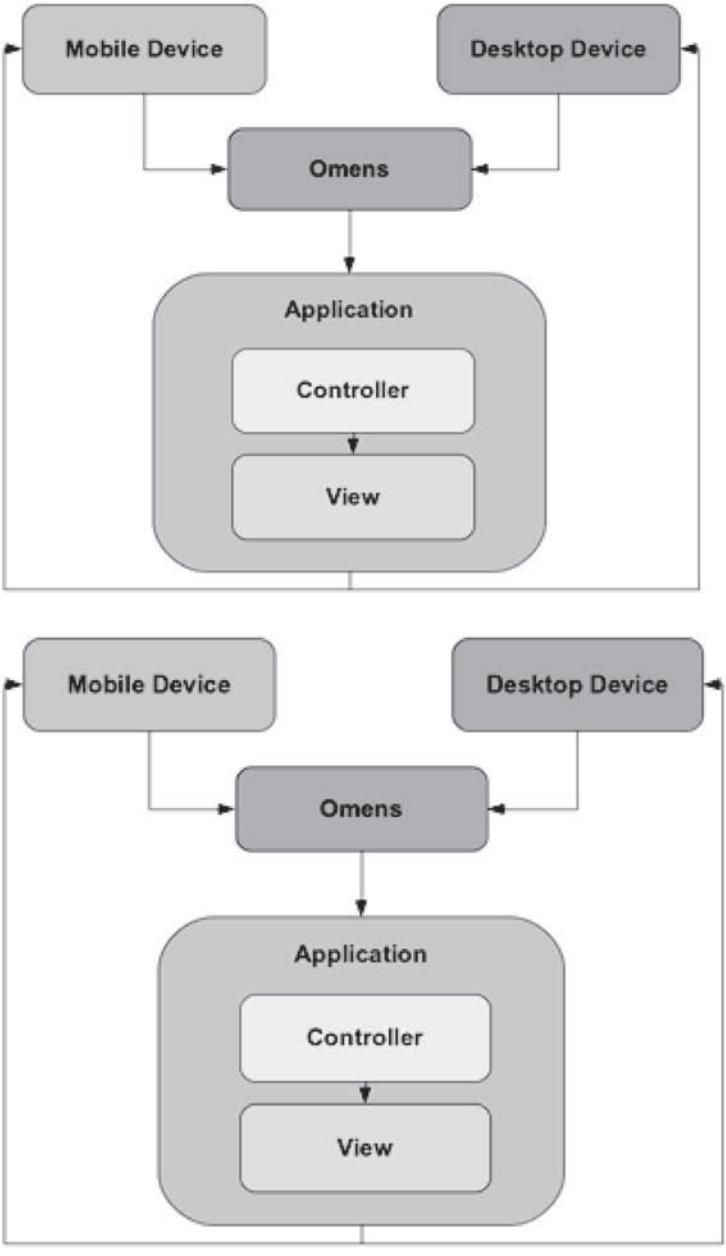


Figura 1 – *Arquitetura do Omens*

1.3. O ponto de Intersecção

Embora tenham focos diferentes, os dois projetos têm um ponto de intersecção: nenhuma aplicação móvel hoje pode ser uma ilha. Existe comunicação entre o dispositivo e algum servidor que recebe ou provê dados. Nós continuamos em rede, e ficaremos cada vez mais conectados.

Nada impede que uma aplicação Web mobile seja híbrida, quer dizer, ela tenha processamento realizado no servidor e processamento realizado no cliente. Assim com a aplicação embarcada pode realizar processamento local e enviar dados para o servidor, que realizará outro processamento.

Por isso, além dos projetos já mencionados, o Beyonder também irá incluir no seu escopo o estudo de plataformas livres para criação de aplicações independentes de dispositivos que sejam executadas tanto no cliente quanto no servidor. Um exemplo é o projeto Mojito, um framework Javascript disponibilizado pelo Yahoo! que explora as funcionalidades do HTML 5 e do Node.js, uma plataforma para construção de aplicações em redes rápidas e escaláveis.

1.4. Infraestrutura necessária

Um processo de construção de software envolve codificação e testes. Neste caso, notadamente, os testes são parte intrínseca de um ciclo iterativo de desenvolvimento. Ao contrário de uma aplicação Desktop ou Web, que poderia ser colocada em execução na própria estação de trabalho do desenvolvedor, esse projeto exige a execução em dispositivos móveis. E mais, em dispositivos móveis com diferentes características, desde o tamanho físico até os recursos disponíveis pelos seus navegadores.

Embora pudesse ser viável solicitar o empréstimo de dispositivos dos respectivos fabricantes, que com certeza têm interesse de que seus aparelhos sejam compatíveis com a maior quantidade possível de serviços, isso possivelmente iria implicar em algum custo. Mas os dados expostos na introdução desse trabalho já indicavam um outro caminho, muito mais simples e econômico para homologar os produtos do projeto em uma gama diversa de dispositivos móveis.

Dados os números expostos na introdução, pode-se imaginar que em qualquer empresa a maioria dos funcionários contém um celular. Se considerarmos que o corpo gerencial tem necessidade de dispositivos móveis com amplos recursos, podemos imaginar que uma parcela dos funcionários deve portar um *tablet*.

Ou seja, já existe um exército de testadores e homologadores para o projeto. Tudo o que eles precisam fazer é requisitar uma página de seus dispositivos móveis e responder se a interface exibida está de acordo com o tamanho de seu dispositivo e se a aplicação é completamente funcional. A validação pode ser feita por meio de um formulário eletrônico, de fácil preenchimento e submissão.

Dessa forma, a infraestrutura necessária para se alocar a esse projeto consiste tão somente em uma estação de trabalho que funcione como ambiente de teste e homologação, uma máquina acessível por Internet, que receba requisições de dispositivos móveis e faça a seleção das visões adequadas de acordo com o dispositivo. Essa máquina também fará o *download* dos scripts a serem processados pelos dispositivos.

Uma única máquina atuando como servidora de dispositivos móveis permite que voluntários acessem a aplicação de teste e relatem os resultados.

Em suma, a infraestrutura necessária para o desenvolvimento consiste em uma máquina dedicada, conectada à Internet e em voluntários com celulares e *tablets* 3G.

Essa estrutura simples permite a adesão de testadores e homologadores de outras organizações interessadas e até da comunidade de usuários em geral, que tem interesse em que os serviços digitais de informação sejam acessíveis plenamente pelos seus dispositivos móveis.

1.5. Sustentabilidade

O projeto tem como alicerce outros projetos de software livre, com suas respectivas comunidades.

A orientação não é criar um produto novo, e ter de manter uma equipe interna de manutenção e suporte dedicada.

Uma vez lançada a primeira versão estável, o objetivo é manter o projeto como um conjunto de contribuições aos projetos originais, atualizando a parte genérica e mantendo apenas o que for particular ao projeto.

Assim é possível ter apenas um desenvolvedor que atue como coordenador e de acordo com a demanda, e que envolva os recursos necessários, de forma pontual, nunca contínua.

O modelo de negócio não é manter uma equipe dedicada, mas ter contribudores envolvidos nos projetos que formam a infraestrutura do projeto Beyonder.

O projeto já foi hospedado desde o princípio no Github, um repositório de projetos de software que permite fácil clonagem e ramificação, o que torna mais simples contribuir com o software. Ele está aberto a contribuições de quaisquer voluntários e seus resultados e produtos estão disponíveis para todos que estiverem interessados.

A abertura do código-fonte permite que ele seja auditado, testado e melhorado de uma forma que não é possível para uma só equipe em uma única empresa.

1.6 Os nomes

O nome Beyonder vem do inglês e significa algo como aquele que está além, acima ou fora do alcance, inclusive se referindo a alguém de outro mundo. No caso deste projeto, o nome Beyonder tem dois propósitos: o primeiro, de usar a língua franca de nosso tempo, e comunicar rapidamente a ideia de algo a distância, o segundo de fazer uma homenagem a uma figura importante na história das comunicações no Brasil.

O Marechal-do-Ar Casimiro Montenegro Filho, patrono da Área de Engenharia da FAB e da Academia Nacional de Engenharia, foi criador do ITA e CTA, instituições de ensino e pesquisa que permitiram ao Brasil criar uma indústria nacional de aviação.

Mas antes das honrarias e dos empreendimentos em ciência e tecnologia, quando ainda era tenente do Exército, Casimiro realizou o primeiro voo do CAN (Correio Aéreo Nacional), levando a bordo de um biplano monomotor correspondências do Rio de Janeiro para São Paulo. Ele foi pioneiro no uso do avião como instrumento de comunicação. O Correio Aéreo Nacional encurtou as distâncias entre pessoas e empresas. Podemos dizer que o pioneirismo do Marechal Casimiro levou a comunicação além dos limites a que estava presa, ao usar os ares como meio de transporte.

Hoje, os celulares e *tablets* também usam os ares para interagirem com redes de comunicação, e não só a distância entre cidades foi superada, como a distância entre países e continentes.

O nome Beyonder portanto é uma homenagem à contribuição do Marechal Casimiro ao desenvolvimento das comunicações no Brasil.

Os nomes dos subprojetos também tem motivos inspirados.

O subprojeto Cyborg tem esse nome para lembrar algo híbrido. Um ciborgue é uma criatura metade homem, metade máquina. A aplicação

Web executada localmente é algo híbrido, pois utiliza parcialmente processamento local e parcialmente conexão com um servidor.

O subprojeto Omens tem esse nome para lembrar algo que está à distância e que é alcançado por meio de um artefato com poderes para tal. Omens significa presságio em inglês, e é uma referência à espada de Lion-o, líder dos Thundercats, que conferia a ele a "visão além do alcance". A frase dita ao utilizar esse poder era "*Sword of Omens give me sight beyond sight*". O nome da espada no original é *Sword of Omens*.

1.7 Onde estamos agora

O projeto Expresso 3 (https://github.com/explivre) possui sincronia com dispositivos móveis, mas não uma versão mobile para suas aplicações. Por isso, ele foi escolhido como "cliente" para o Beyonder Omens, o primeiro subprojeto a ter um produto entregável. Foi realizada a prospecção de APIs de reconhecimento de dispositivos com bancos de dados de características, e foram selecionados dois produtos livres: Browser Capabilities (BCP) e TERA WURFL.

Com base nessas APIs, foi criada uma biblioteca de classes chamada Beyonder, seguindo o modelo de componentes do Zend Framework, que é utilizado no backend do Expresso. A figura 2 mostra a estrutura dessa biblioteca.

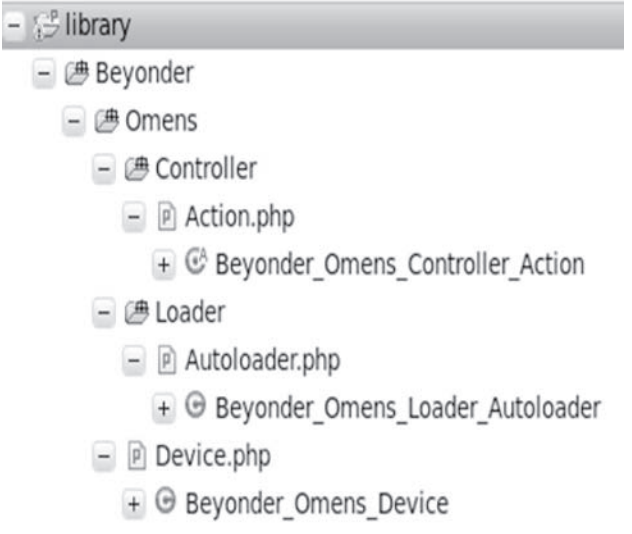


***Figura 2** – Biblioteca Beyonder*

Quatro aplicações de exemplo foram criadas para testar essa biblioteca, que pode ser utilizada em uma aplicação PHP com Zend Framework. Elas estão disponíveis dentro do site do Fórum de Tecnologia em Software Livre nos seguintes endereços:

- http://ftsl.org.br/bcomens
- http://ftsl.org.br/bcomens2
- http://ftsl.org.br/twomens
- http://ftsl.org.br/twomens2

É possível acessar esses endereços e conferir páginas diferentes para acesso mobile e desktop.

Todo o código do projeto e das aplicações criadas está disponível em <https://github.com/fgsl/beyonder>. O ambiente também permite colaboração, estando aberto a sugestões e críticas.

1.8. Reflexões

Temos um paradigma de desenvolvimento tecnológico baseado em competição, que possui como grande referência as guerras, mesmo as frias como a que teve como grande campo de batalha a corrida espacial entre Estados Unidos da América e a extinta União Soviética.

No entanto, esse modelo de desenvolvimento em que uma das partes sempre perde acaba sendo danoso, porque a responsabilidade por manter algo sozinho é muito grande. E sabemos da fragilidade de depender de um único vencedor quando lembramos da Crise de 1929.

Segundo Stephen Hawking, Sir Isaac Newton disse que, se havia visto mais longe, era porque estava sobre ombros de gigantes. Com isso ele queria dizer que não poderia ter criado o cálculo diferencial e integral nem descoberto a lei da gravitação universal se não tivesse acesso aos trabalhos de outros pesquisadores, como Copérnico, Galileu, Kepler e Descartes. O avanço da ciência e da tecnologia só é realmente possível, com benefício para todos, pela agregação de conhecimento, feita com compartilhamento.

1.9. Conclusões

O projeto Beyonder é totalmente realizável. Ele está baseado em reuso de soluções, tecnologia aberta e custo mínimo. Mas principalmente, está baseado no desenvolvimento de software em comunidade.

Esse projeto tem todas as condições de ser mantido com o mínimo custo por uma grande comunidade de desenvolvedores e usuários, dado que comunicação móvel é algo que já faz parte do dia a dia do cidadão, e os dispositivos móveis tornam-se um meio simples de levar serviços públicos de informação com presteza a todos.

1.10. Referências

CETIC.BR. *TIC DOMICÍLIOS e USUÁRIOS 2011 – TOTAL BRASIL.* Disponível em <http://www.cetic.br/usuarios/tic/2011-total-brasil/analises.htm> Acesso em 06/08/2012.

HAWKING, STEPHEN. *Os Gênios da Ciência: Sobre os Ombros dos Gigantes*. 1.ed. São Paulo. Campus, 2004.

THE DOJO FOUNDATION. *Mobile*. Disponível em <http://dojotoolkit.org/features/mobile>. Acesso em 08/08/2012.

THE JQUERY FOUNDATION. *JQuery Mobile Framework.* Disponível em <http://jquerymobile.com/>. Acesso em 08/08/2012.

YAHOO!. *Mojito.* Disponível em <http://developer.yahoo.com/cocktails/mojito/>. Acesso em 08/08/2012.

Formato	15,5 x 22,5 cm
Mancha gráfica	12 x 18,3cm
Papel	pólen soft 80g (miolo), cartão supremo 250g (capa)
Fontes	Verdana 13/17 (títulos), Book Antiqua 10,5/13 (textos)

ISBN 857631411-8
9 788576 314110

Realização:

SERPRO Serviço Federal de Processamento de Dados

Ministério da Fazenda

Apoio:

SERPROS Previdência é ter um plano

CESUPA Centro Universitário do Estado do Pará

SUCESU PARÁ Sociedade de Usuários de Informática e Telecomunicações Regional Pará

PRODEPA

Secretaria de Estado de Ciência, Tecnologia e Inovação

GOVERNO DO PARÁ

FUNDAÇÃO ALEXANDRE DE GUSMÃO

Bronze:

ITAIPU BINACIONAL

PREVIDÊNCIA SOCIAL DATAPREV

Prata:

CAIXA

Diamante:

SEBRAE

BNDES

GOVERNO FEDERAL BRASIL PAÍS RICO É PAÍS SEM POBREZA