Capítulo 3

Novas Arquiteturas de Data Center para *Cloud Computing*

Fábio Luciano Verdi[1], Christian Esteve Rothenberg[2], Rafael Pasquini[2] e Maurício Ferreira Magalhães[2]

[1]Universidade Federal de São Carlos - UFSCar
[2]Faculdade de Engenharia Elétrica e Computação (FEEC) - Unicamp

Abstract

Recently, there has been a change in the way we interact with services and applications. The paradigm (still under evolution) of cloud computing provides services and applications through the Internet intending to offer infinity capacity by using the pay-as-you-go model. The purpose of this work is to present the impacts on the network architectures caused by this new communication model which, on one hand is a good allied of companies such as Google, Microsoft and Yahoo! but, on the other hand, impose technical and market challenges to support a growing demand for cloud services. We will discuss the challenges and limitations of the current network infraestructure, the network requirements for the next generation data centers and the architectures that pursue to attend such requirements.

Resumo

Recentemente tem-se observado uma mudança na forma como interagimos com os serviços e as aplicações. O paradigma (ainda sob evolução) de cloud computing (ou computação em nuvem) fornece serviços e aplicações através da Internet com a promessa de capacidade infinita e modelos de serviço do tipo pay-as-you-go. Este minicurso tem como objetivo apresentar os impactos nas arquiteturas de rede deste novo modelo de comunicação que, por um lado se apresenta como um aliado importante de empresas como Google, Microsoft e Yahoo! mas que, por outro lado, impõe desafios técnicos e de mercado para suportar uma crescente demanda por cloud services. Serão discutidos os desafios e as limitações da infraestrutura tradicional, os requisitos de rede para data centers de nova geração e as arquiteturas de data centers que buscam atender tais requisitos.

3.1. Introdução

Um novo modelo de computação tem surgido e alterado a forma como interagimos com a rede e com os serviços e aplicações. Os *cloud services* [Greenberg 2009] são oferecidos por grandes empresas tais como Google, Yahoo!, Facebook e Microsoft e permitem que aplicações sejam hospedadas e executadas remotamente em um grande *data center*.

O surgimento de serviços populares tais como e-mail baseado na Web, busca (*searching*) e redes sociais, associados ao aumento da conectividade através de banda larga e redes ópticas, impulsionaram o modelo de comunicação centrado no servidor (*server-side*). Cada vez mais o processamento e o armazenamento estão sendo movidos dos PCs para grandes provedores de serviços. Fotos, vídeos e aplicações que antes eram armazenadas e processadas nos PCs, agora migram para serem hospedadas e processadas por provedores sob o formato de serviços Web.

Esta mudança para um modelo centrado no servidor não traz vantagens apenas para o usuário que fica liberado de toda a gerência local necessária para manter as aplicações (intensas configurações e grandes quantidades de *backups*), mas também traz vantagens para os produtores de software [Hoelzle and Barroso 2009]. O desenvolvimento de aplicações é mais simples pois as mudanças e as melhorias nos softwares são feitas em um único local, ao invés de serem feitas em milhões de clientes que possuem diferentes configurações de hardware. Além disso, uma vez que o uso do software hospedado no provedor passa a ser compartilhado por diversos usuários, o custo também se divide entre estes usuários, fazendo com que os valores pela utilização dos serviços sejam reduzidos.

Em termos gerais, a computação se torna mais barata quando vista como um serviço compartilhado. Esta premissa tem sido discutida durante muito tempo mas, até o momento, o modelo centrado no usuário, com máquinas e aplicações individuais, tem sido dominante.

O modelo computacional dos *data centers* começou com os *mainframes* em 1960. Posteriormente, os microcomputadores surgiram no mercado e uma busca constante por altas capacidades de armazenamento se estabeleceu. As estruturas computacionais baseadas nos *mainframes* e nos PCs podem ser vistas como modelos de *data center*s. O primeiro, como um modelo mais concentrado e o segundo, como um modelo distribuído. Esta fase foi seguida pelos sistemas distribuídos baseados no modelo cliente/servidor e, subsequentemente, pelo crescimento explosivo da Internet e da Web. Mais recentemente, a evolução das técnicas de virtualização tem propiciado o desenvolvimento de aplicações que compartilham a mesma infraestrutura de hardware, alavancando o surgimento de soluções para serviços em nuvem.

A diversidade de aplicações que podem ser hospedadas usando o modelo em nuvem inclui desde aplicações comerciais, aplicações de TI e aplicações Web tradicionais até aplicações científicas para processamento paralelo em *batch* e aplicações móveis. Estas diferentes aplicações requerem diferentes arquiteturas de *data centers* e isto tem motivado a pesquisa e o desenvolvimento de soluções que atendam a esta demanda, no sentido de criar mecanismos escaláveis, com alto desempenho e custos mínimos. Isto inclui a pesquisa em infraestrutura voltada para a eficiência energética, cabeamento, resfriamento e, principalmente, infraestrutura de interconexão dos servidores. Neste trabalho enfatizamos os desafios atuais grandes *data centers* em termos de infraestrutura de rede.

Esta seção apresenta as definições, as características essenciais de *cloud computing* e

um panorama geral da área. A Seção 3.2 caracteriza os *data centers* e apresenta a arquitetura atual utilizada nos *data centers* tradicionais destacando as suas limitações e derivando os requisitos de rede para os *data centers* de próxima geração. A Seção 3.3 discute algumas propostas atuais para *data centers* que tentam atender estes requisitos. Finalmente, a Seção 3.4 conclui o capítulo apontando as tendências atuais, os desafios em *cloud computing* e direcionando futuras linhas de pesquisa nesta área.

3.1.1. Definições e terminologias

O que é *cloud computing*?

O termo *cloud computing* (computação em nuvem) possui, como qualquer novo termo[1], várias definições possíveis, muito embora todas relativamente parecidas. O mais importante é entendermos que a definição do conceito está ainda em evolução e novas definições poderão surgir. O trabalho [Vaquero et al. 2009] faz uma análise das definições utilizadas na literatura atual e adota a seguinte opção:

"Cloud computing é um conjunto de recursos virtuais facilmente usáveis e acessíveis tais como hardware, plataformas de desenvolvimento e serviços. Estes recursos podem ser dinamicamente re-configurados para se ajustarem a uma carga variável, permitindo a otimização do uso dos recursos. Este conjunto de recursos é tipicamente explorado através de um modelo pay-per-use com garantias oferecidas pelo provedor através de acordos de nível de serviço (Service Level Agreements-SLAs)."

A Figura 3.1 ilustra o conceito.

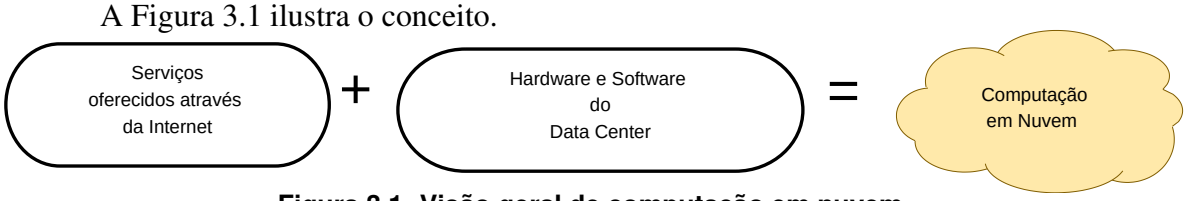


Figura 3.1. Visão geral de computação em nuvem.

Os serviços oferecidos também são conhecidos como *Software as a Service* (o termo será melhor explicado na sequência) e o *data center* engloba todo o hardware utilizado para processar e, também, armazenar os dados associados a eles. Sendo assim, o *data center* e os softwares necessários para gerenciar a oferta de serviços são denominados *cloud*. É importante não confundir os serviços oferecidos pela Internet para os usuários com os softwares e sistemas utilizados nos *data centers* para gerenciamento e fornecimento de tais serviços.

O modelo de computação em nuvem é composto, tipicamente, por cinco características essenciais, três modelos de serviços e quatro modelos de implantação da nuvem [Mell and Grance 2009], conforme descrito abaixo.

- Características Essenciais
  - Serviço sob-demanda: as funcionalidades computacionais são providas automaticamente sem a interação humana com o provedor do serviço;
  - Amplo acesso aos serviços: os recursos computacionais estão disponíveis através da Internet e são acessados via mecanismos padronizados, para que possam ser utilizados por dispositivos móveis e portáteis, computadores, etc.;

[1]Neste trabalho, tanto o termo em inglês quanto o termo em português serão utilizados sem distinção.

- *Resource pooling*[2]: os recursos computacionais (físicos ou virtuais) do provedor são utilizados para servir a múltiplos usuários, sendo alocados e realocados dinamicamente conforme a demanda do usuário. Neste cenário, o usuário do serviço não tem a noção da localização exata do recurso, mas deve ser capaz de definir a localização em um nível mais alto (país, estado, região);
- Elasticidade: as funcionalidades computacionais devem ser rápida e elasticamente providas, assim como, rapidamente liberadas. O usuário dos recursos deve ter a impressão de que ele possui recursos ilimitados, que podem ser adquiridos (comprados) em qualquer quantidade e a qualquer momento;
- Medição dos serviços: os sistemas de gerenciamento utilizados para computação em nuvem controlam e monitoram automaticamente os recursos para cada tipo de serviço (armazenamento, processamento e largura de banda). Este monitoramento do uso dos recursos deve ser transparente para o provedor do serviço, assim como, para o consumidor do serviço utilizado.

• Modelos de serviços

- Software como um serviço (*Software as a Service* - SaaS): aplicações de interesse para uma grande quantidade de usuários passam a ser hospedadas na nuvem como uma alternativa ao processamento local. As aplicações são oferecidas como serviços pelos provedores e acessadas pelos clientes através de aplicações como o *browser*. Todo controle e gerenciamento da infraestrutura de rede, sistemas operacionais, servidores e armazenamento é feito pelo provedor do serviço. O Google Apps [Google 2010b] é um exemplo de SaaS. A Forrester concluiu que ao usar o Google Apps, as empresas podem economizar de 50% a 70% em comparação com outras soluções de e-mail [Forrester 2010];
- Plataforma como um Serviço (*Plataform as a Service* - PaaS): é a capacidade oferecida pelo provedor para o usuário desenvolver aplicações que serão executadas e disponibilizadas em nuvem. Neste sentido, surge um outro conceito conhecido como *utility computing*[3], utilizado para denominar toda a plataforma de suporte ao desenvolvimento e fornecimento de aplicações em nuvem. Qualquer aplicação quando desenvolvida precisa seguir um modelo de computação, um modelo de armazenamento e um modelo de comunicação. As plataformas para desenvolvimento de aplicações em nuvem fornecem tais modelos e permitem utilizar conceitos implícitos tais como virtualização e compartilhamento de recursos. As *Utility Computing* mais relevantes atualmente são a AppEngine da Google [Google 2010b] e a plataforma Azure da Microsoft [Microsoft 2010]. Na Seção 3.1.3 apresentaremos as três classes de *Utility Computing* e suas diferenças;
- Infraestrutura como um Serviço (*Infraestructure as a Service* - IaaS): é a capacidade que o provedor tem de oferecer uma infraestrutura de processamento e armazenamento de forma transparente. Neste cenário, o usuário não tem o cont-

[2]O termo *resource pooling* é utilizado para definir um conjunto de recursos que se comportam como se fossem um único recurso [Wischik et al. 2008]. O objetivo desta técnica é aumentar a confiabilidade, flexibilidade e eficiência do sistema como um todo.

[3]Uma tentativa de tradução para este termo seria computação vista como um utilitário. Entretanto, neste caso, os autores preferem usar o termo em inglês.

role da infraestrutura física mas, através de mecanismos de virtualização, possui controle sobre os sistemas operacionais, armazenamento, aplicações instaladas e, possivelmente, um controle limitado dos recursos de rede. Um exemplo de *Utility Computing* disponibilizada como uma IaaS é a Amazon EC2.

- Modelos de implantação
  - Nuvem privada (*private clouds*): compreende uma infraestrutura de nuvem operada unicamente por uma organização. Os serviços são oferecidos para serem utilizados internamente pela própria organização, não estando disponíveis publicamente para uso geral;
  - Nuvem comunidade (*community cloud*): fornece uma infraestrutura compartilhada por uma comunicade de organizações com interesses em comum;
  - Nuvem pública (*public cloud*): a nuvem é disponibilizada publicamente através do modelo *pay-per-use*. Tipicamente, são oferecidas por companhias que possuem grandes capacidades de armazenamento e processamento;
  - Nuvem hibrida (*hybrid cloud*): a infraestrutura é uma composição de duas ou mais nuvens (privada, comunidade ou pública) que continuam a ser entidades únicas porém, conectadas através de tecnologia proprietária ou padronizada.

Ao analisarmos as definições expostas acima, é possível destacar três novos aspectos em relação ao hardware, introduzidos em *cloud computing*. São eles:

- A ilusão de recurso computacional infinito disponível sob-demanda;
- A eliminação de um comprometimento antecipado por parte do usuário;
- A capacidade de alocar e pagar por recursos usando uma granularidade de horas.

Esta elasticidade para obter e liberar recursos é um dos aspectos chaves da computação em nuvem, sendo uma das principais diferenças quando comparada com computação em grade. A próxima seção faz uma breve comparação entre computação em nuvem, computação em grade e *High Performance Computing* (HPC).

3.1.2. Grades, Computação em Nuvem e HPC

Muitas das características encontradas em grades computacionais também são encontradas na computação em nuvem. Isto ocorre porque ambos os modelos possuem objetivos comuns tais como: redução dos custos computacionais, compartilhamento de recursos e aumento de flexibilidade e confiabilidade. Entretanto, existem algumas diferenças que precisam ser enfatizadas. Estas semelhanças e diferenças têm causado confusão e sobreposição de características e funcionalidades. O trabalho [Vaquero et al. 2009] realiza um estudo das diferentes características associadas com computação em nuvem e as compara com as características associadas com grades computacionais. Apesar do trabalho apontar 12 itens comparativos, abaixo elencamos apenas os mais importantes.

- Modelo de pagamento e origens: as grades computacionais surgiram através de financiamento público, na maioria das vezes patrocinadas por projetos dentro de universidades. O modelo *cloud computing* é motivado por aspectos comerciais onde grandes empresas criam estratégias de mercado com interesses nos lucros. Tipicamente, os serviços em grade são cobrados usando uma taxa fixa por serviço, enquanto que os

usuários dos serviços oferecidos nas *clouds* são cobrados pelo modelo *pay-per-use*. Muitas aplicações não usam a mesma capacidade computacional de armazenamento e recursos de rede. Sendo assim, o modelo de cobrança deve considerar o pagamento separado para cada tipo de recurso utilizado;

- Compartilhamento de recursos: as grades computacionais compartilham os recursos entre as organizações usuárias através do modelo "mais justo possível". A computação em nuvem fornece a quantidade de recursos desejados para cada usuário dando a impressão de recurso dedicado. Esta noção de recurso dedicado é possível através do uso de virtualização, aspecto ainda pouco explorado pelas grades;
- Virtualização: as grades computacionais usam interfaces para esconder a heterogeneidade dos recursos computacionais. A virtualização utilizada em grades computacionais é ainda muito simplista. A virtualização utilizada em *cloud computing* ocorre de forma plena, possibilitando que usuários instalem máquinas virtuais e sistemas operacionais específicos nestas máquinas virtuais. A migração/mobilidade de máquinas virtuais também é um aspecto comum dentro da nuvem e permite a otimização do uso de recursos de energia e resfriamento;
- Escalabilidade e gerenciamento: a escalabilidade em grades ocorre através do aumento no número de nós de processamento. A escalabilidade em *cloud computing* ocorre através de um redimensionamento do hardware virtualizado. O gerenciamento das grades computacionais é dificultado pois não há tipicamente uma única entidade proprietária de todo o sistema. Por outro lado, as *clouds* encontradas atualmente são controladas por uma única entidade administrativa, muito embora exista uma tendência em se criar federações de nuvens;
- Padronização: a maturidade das grades computacionais fez com que vários fóruns fossem criados para a definição de padronização. Neste sentido, esforços para padronização de interfaces para os usuários assim como padronização de interfaces internas alavancaram a interoperabilidade de grades computacionais. Em *cloud computing*, as interfaces de acesso aos serviços são muito parecidas com as interfaces das grades, entretanto, as interfaces internas são proprietárias e dificultam a criação de federação de nuvens. Atualmente há várias iniciativas para definição de padrões para computação em nuvem [Cloud Standards 2010]. Um dos desafios principais é a padronização do formato das imagens virtuais e APIs de migração.

Em termos gerais, grade computacional se refere ao processamento distribuído e paralelo, ou seja, quebrar uma tarefa em várias, distribuir em nós para processamento e então unir as partes para obter o resultado final. Na maioria das vezes, isto significa executar a mesma tarefa em diferentes conjuntos de dados para agilizar o resultado. Para isso, distribui-se a atividade no número máximo de unidades de processamento possível, enquanto que em *cloud computing*, obtém-se a quantidade de recursos suficientemente necessária para a realização da tarefa computacional em um determinado tempo.

Há também distinções entre HPC e computação em nuvem. Tipicamente, HPC se concentra em executar uma única aplicação com alto desempenho, muito similar às grades computacionais. As aplicações HPC podem ser executadas em grade ou utilizar uma nuvem, entretanto, devem seguir as premissas temporais destes modelos. O desenvolvimento de aplicações HPC faz uso de um modelo de programação de baixo nível, enquanto o desenvolvimento de aplicações em nuvem ocorre através da utilização de plataformas com linguagens e

ambientes em alto nível (PaaS) focando na implementação de serviços que serão executados continuamente no *data center*.

Ian Foster em seu blog [Foster 2010] faz uma pequena discussão que compara a execução de uma aplicação em um supercomputador e em uma nuvem. Para os testes, o supercomputador escolhido foi o sistema "Abe" do *National Center for Supercomputing Applications* [NCSA 2010] e a nuvem escolhida foi a Amazon EC2 [Amazon 2010a].

Os resultados mostraram que a execução da aplicação no supercomputador foi extremamente mais rápida. Entretanto, a análise feita por Ian Foster considera o fato de que, na maioria das vezes, a obtenção de recursos computacionais em um supercomputador não ocorre de maneira imediata. Neste sentido, executar uma determinada aplicação em nuvem pode trazer vantagens quando consideramos o tempo total (momento da submissão até o momento da obtenção dos resultados) para concluir a tarefa.

O teste mostrou que a aplicação executada em um supercomputador com 32 processadores precisou de 25 segundos para ser finalizada. Enquanto que a mesma aplicação precisou de 100 segundos na Amazon EC2. O tempo necessário para se obter 32 nós (imagens de máquinas virtuais) no *data center* da Amazon foi de 300 segundos. Sendo assim, o tempo total para concluir a tarefa na Amazon EC2 foi de 100 + 300 = 400 segundos. Por outro lado, a probabilidade de se obter em 400 segundos 32 processadores utilizáveis por 20 segundos no supercomputador mencionado é de 34%, uma taxa relativamente baixa.

Neste sentido, aplicações HPC precisam considerar a possibilidade de utilizarem *cloud computing* ao invés de apostarem unicamente em supercomputadores. Além de obterem um bom tempo de resposta, na maioria das vezes o custo é reduzido.

3.1.3. Classes de computação utilitária (*Utility Computing*)

Atualmente existem três principais plataformas, cada uma disponibilizando níveis diferentes de funcionalidades dependendo do grau de abstração oferecido ao programador: a Amazon EC2 [Amazon 2010a], a AppEngine da Google [Google 2010b] e a plataforma Azure da Microsoft [Microsoft 2010].

Dentre as três plataformas mencionadas anteriormente, a Amazon EC2 é a que oferece a maior liberdade de acesso aos recursos. Uma instância EC2 oferece ao usuário desenvolvedor um conjunto de recursos físicos como se fosse uma máquina real, podendo controlar praticamente todo o software a partir do *kernel*. A API oferecida pela Amazon EC2 é pequena, possuindo poucas chamadas para configuração do hardware virtualizado e não há, em princípio, limite quanto aos tipos de aplicações que podem ser hospedadas pela nuvem da Amazon. Se, por um lado, toda esta flexibilidade oferece ao usuário uma elevada capacidade de desenvolvimento, para a Amazon, tanta flexibilidade dificulta a manutenção da escalabilidade e a gerência de falhas.

Os AWS (*Amazon Web Services*) [Amazon 2010b] oferecem um conjunto de ferramentas e serviços prontos para facilitar a vida dos desenvolvedores de aplicações na nuvem, que podem ou não ser utilizados em conjunto com o EC2. Tais ferramentas e serviços incluem serviços de armazenamento (S3 - *Simple Storage Service*), bases de dados (RDS - *Relational Database Service*, *SimpleDB*), processamento massivo de dados (*Elastic MapReduce*), faturamento (*DevPay*), entre outros.

No outro extremo, encontra-se a AppEngine da Google que oferece uma plataforma para desenvolvimento de aplicações específicas para determinados domínios. A AppEngine é voltada para o desenvolvimento de aplicações Web tradicionais, forçando que tais aplicações sejam estruturadas em duas camadas: camada sem estado e a camada com estado. As aplicações desenvolvidas utilizando a AppEngine devem usar o modelo *request-reply* e, para isso, é de extrema importância levar em conta a quantidade de CPU utilizada pelas requisições realizadas. A AppEngine possui um mecanismo bastante escalável para armazenamento de dados. As aplicações desenvolvidas pela AppEngine usam a MegaStore, uma solução proprietária da Google para armazenamento de dados baseada na BigTable [Google 2010a][4]. Atualmente, as aplicações podem ser desenvolvidas utilizando as linguagens Java e Python. A linguagem Java se integra ao Google Web Toolkit [Google 2010c] e traz um *plug-in* para o Eclipse que permite o desenvolvimento completo de aplicações AJAX.

A plataforma Azure da Microsoft é um ponto intermediário entre a flexibilidade total da EC2 e a especificidade da AppEngine. As aplicações desenvolvidas na plataforma Azure utilizam as bibliotecas .NET e são compiladas para uma linguagem independente de ambiente denominada de *Common Language Runtime*. A plataforma Azure suporta o desenvolvimento de aplicações de objetivo geral tal como a EC2, ao invés de uma única categoria de aplicações como no caso da AppEngine. Permite ainda um certo grau de escolha nas linguagens (PHP, .NET, proprietária) mas não permite o controle do sistema operacional. As bibliotecas fornecem a possibilidade de configuração automática da rede e gerência de falhas, mas requerem que o desenvolvedor defina isto dentro de suas aplicações. A Figura 3.2 apresenta algumas *utility computing* e o modelo de camadas organizado em Infraestrutura (IaaS), Plataforma (PaaS) e Aplicação (AaaS).

3.1.4. Benefícios e oportunidades de novas aplicações

Embora seja possível imaginar a quantidade de aplicações que podem ser desenvolvidas utilizando as plataformas descritas acima e vislumbrar um universo de novos serviços, espera-se que esta diversidade de aplicações cresça muito mais. Abaixo listamos algumas aplicações que podem se beneficiar com *cloud computing* [Armbrust et al. 2009].

- Aplicações móveis interativas: estes serviços tendem a utilizar o modelo em nuvem, pois necessitam estar sempre disponíveis e requerem armazenamento de grandes quantidades de dados. Sensores e, mais recentemente, o conceito de *Internet of Things*, farão com que a produção de dados aumente e necessite ser armazenada em *data centers* geograficamente próximos ao local onde serão utilizados;
- Processamento paralelo em *batch*: muito embora as aplicações tipicamente utilizadas em nuvem sejam interativas, o processamento em *batch* de grandes quantidades de dados é um forte candidato para ser realizado nos *data centers*;
- Computação analítica de dados: um caso especial do processamento em *batch* é a análise de dados de negócios. Neste sentido, uma grande tendência, atualmente, nas empresas é entender o comportamento e o perfil de seus clientes a fim de oferecer novos produtos e serviços. Este tipo de análise permite fundamentar as tomadas de decisões na empresa e organizar estratégias de mercado;

[4]A BigTable é um sistema distribuído de armazenamento desenvolvido pela Google capaz de armazenar grandes quantidades de dados. Atualmente, a BigTable é responsável por armazenar dados de mais de 60 produtos da Google, incluindo Google Analytics, Google Finance, Orkut e Google Earth.

- Aplicações que requerem computação intensiva: vários pacotes matemáticos como Matlab e Mathematica requerem uma capacidade de processamento bastante intensa. Renderização em 3D também exige alta capacidade de processamento. Tais aplicações podem fazer uso do modelo em nuvem para realizarem cálculos complexos, evitando atualizações constantes de hardware em cada *desktop*.

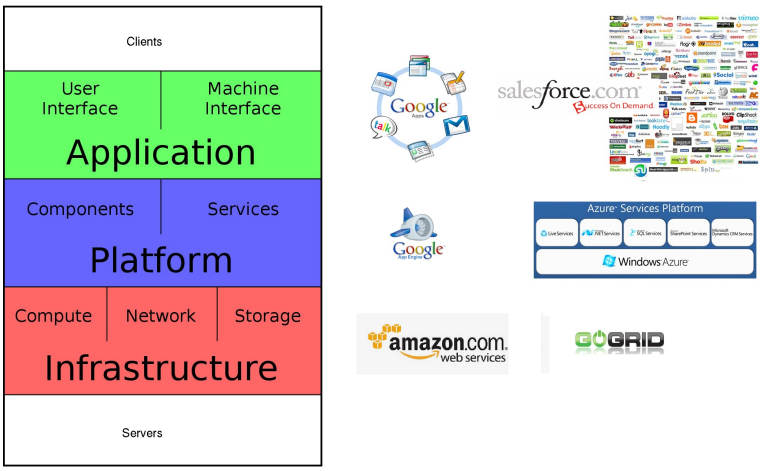


Figura 3.2. Modelo em camadas da computação em nuvem com exemplos de serviços oferecidos. Extraída de [Johnston 2009].

3.1.5. Distribuição de custos

A busca por redução de custos é um dos itens dominantes no projeto das infraestruturas dos serviços em nuvem. Muitos esforços são dedicados na obtenção de soluções mais eficientes, desde a modularização da infraestrutura do *data center*, até componentes *green* que atenuem o consumo da energia em função da carga [Barroso and Hölzle 2007], passando por propostas de encaminhamento de pacotes para aqueles *data center*s onde a energia é mais barata [Qureshi et al. 2009]. A quantificação dos custos associados a um *data center* é uma tarefa complexa que tem sido objeto de recentes estudos.

Segundo a análise de Greenberg [Greenberg et al. 2009a] e os argumentos de James Hamilton [Hamilton 2008], os principais custos de um *data center* da ordem de 50.000 servidores construído seguindo as melhores práticas da indústria (qualidade, alta disponibilidade, etc.), podem ser estruturados conforme a Figura 3.3. Os custos são amortizados assumindo um tempo de vida razoável para os equipamentos adquiridos e a infraestrutura instalada, assim como, um custo de 5% para o dinheiro. Desta forma, pode-se obter uma métrica de custo comum que pode ser aplicada na fase inicial do projeto (por exemplo, aquisição de equipamentos) e durante a operação do *data center* (por exemplo, energia, manutenção, etc.).

Ao observarmos o gráfico acima, constatamos que 62% do custo pertence à infraestrutura de TI (44% em servidores e 18% em equipamentos de rede). Os números específicos podem ser discutidos e variar de um caso para outro. Porém, levando em conta a tendência atual relativa ao aumento dos custos com energia e infraestrutura, enquanto o custo de servidores (medido em trabalho realizado por dólar investido) continua a cair, a conclusão é que os custos totais associados à energia (soma dos custos com a energia consumida, para efetuar a refrigeração e a infraestrutura necessária para distribuição de energia) são hoje comparáveis aos custos dos equipamentos de TI e poderão dominar os custos de um *data center*.

Embora a infraestrutura de comunicação não represente o maior custo, cabe destacar a importância da inovação nas arquiteturas de redes e nos sistemas distribuídos de gerência para a redução dos custos e a obtenção do máximo retorno para cada dólar investido. Uma maneira de se obter isso é oferecendo agilidade aos mecanismos de rede, para que qualquer máquina virtual possa ser instanciada em qualquer servidor físico disponível, independentemente da sua localização na rede, conforme discutiremos mais a frente neste minicurso (veja Seção 3.2.3). Outras propostas de novas arquiteturas de rede para *data center*s [Brandon 2009] sugerem ligar/desligar *switches* que, em função do tráfego, formam caminhos redundantes, apontando para reduções no consumo de energia dos equipamentos de rede na ordem de 50%.

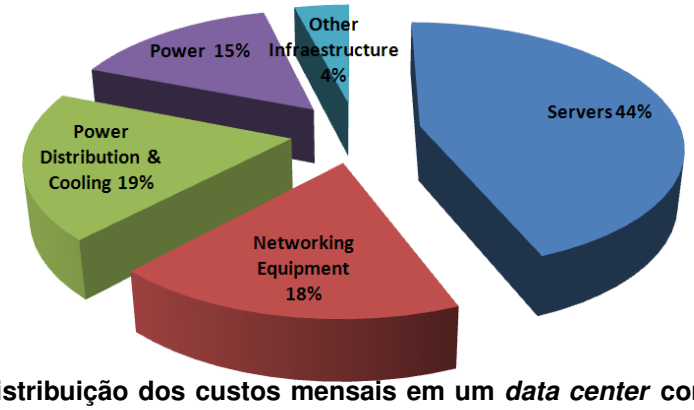


Figura 3.3. Distribuição dos custos mensais em um *data center* com 50 mil servidores. Extraída de [Hamilton 2009a].

3.1.5.1. Eficiência energética

A energia e a refrigeração são dois dos maiores problemas (fontes de despesa) que as organizações de TI enfrentam, de maneira que esses custos devem ser controlados para permitir a expansão e proliferação de mais e maiores *data centers*. *Data centers* que conseguem uma economia eficiente da energia podem gerenciar melhor o aumento de processamento computacional, da rede e as demandas de armazenamento. A redução dos custos de energia se traduz em um menor custo total de infraestrutura, mantendo a oferta de serviços competitiva e capaz de atender às necessidades de futuros negócios.

O Consórcio Green Grid [Christian Belady (ed) 2007] tem reconhecido a importância do estabelecimento de métricas para a eficiência do *data center*, com o objetivo de orientar sobre as tecnologias capazes de melhorar o desempenho por Watt. Idealmente, essas métricas ajudam a determinar se os *data centers* existentes podem ser otimizados antes da ampliação e construção de novas infraestruturas. O Green Grid define duas métricas relacionadas (ver fórmulas abaixo): (1) *Power Usage Effectiveness* (PUE) e (2) a eficiência da infraestrutura do *data center* (*Datacenter infrastructure Efficiency*-DCiE).

PUE = Total Facility Power/IT Equipment Power (1)

DCiE = 1/PUE = IT Equipment Power/Total Facility Power x 100% (2)

Note que nas equações 1 e 2, o "*Total Facility Power*" é definido como a energia dedicada exclusivamente ao *data center*. O "*IT Equipment Power*" é definido como a energia consumida por todo o equipamento que é usado para gerenciar, processar, armazenar, ou encaminhar os dados dentro do *data center*.

O valor PUE ideal é de 1.0, correspondendo a um *data center* onde toda a energia fornecida pela rede elétrica é dedicada para equipamentos de TI, não havendo gasto de energia com refrigeração e nem com distribuição de energia. Um PUE < 1 seria possível com a geração local de energia com base em fontes térmicas residuais, mas atualmente este modelo é comercialmente impraticável de se implementar.

Embora o PUE e o DCiE sejam essencialmente equivalentes, eles podem ser usados para ilustrar a alocação de energia no *data center* de forma diferente. Por exemplo, se um PUE está determinado a ser 3.0, isso indica que a demanda do *data center* é três vezes maior do que a energia necessária para alimentar o equipamento de TI. Além disso, a relação pode ser usada como um multiplicador para o cálculo do impacto real das demandas de energia. Por exemplo, se um servidor exige 500 watts e o PUE para o *data center* é 3.0, então a energia que deverá ser disponibilizada para entregar 500 watts para o servidor é de 1500 watts. Reciprocamente, um valor DCiE de 33% (equivalente a um PUE de 3.0) sugere que os equipamentos de TI consomem 33% da energia no *data center*.

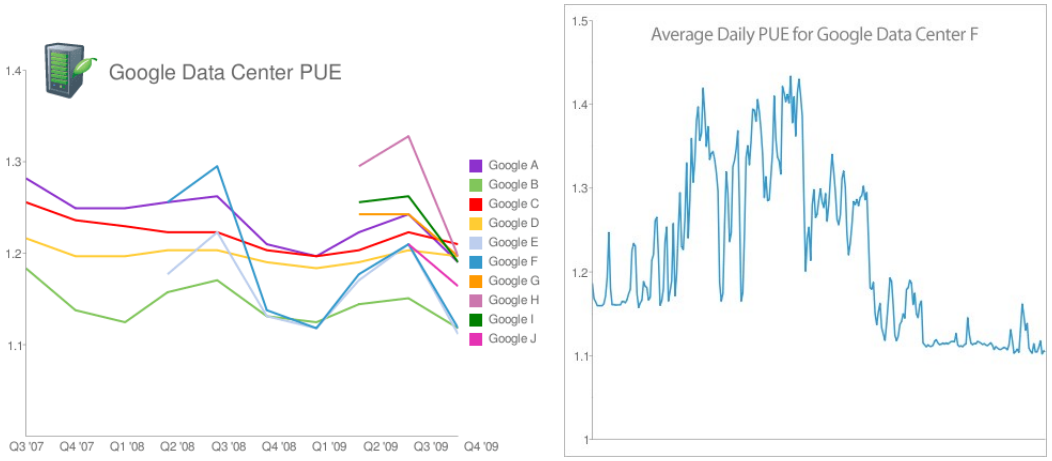
Infelizmente, o PUE médio dos *data centers* convencionais é vergonhosamente elevado. Segundo estudos de 2006 [Lim et al. 2008], 85% dos *data center*s atuais possuem um PUE estimado superior a 3.0, isto é, os sistemas mecânicos e elétricos do *data center* consomem o dobro de energia quando comparados ao equipamento de TI. Estudos mais otimistas mostram um PUE médio de aproximadamente 2.0 para um conjunto de 22 *data centers* pesquisados [Greenberg et al. 2006].

O especialista James Hamilton discute no seu blog [Hamilton 2010a], de forma regular, desafios e tendências no mundo dos *data center*s, incluindo discussões sobre utilização do PUE e estratégias para melhorar a eficiência energética [Hamilton 2009b]. Na realidade, o PUE é um valor dinâmico, que muda com as alterações de carga de trabalho no *data center*. A Figura 3.4(a) apresenta os valores médios de PUE de 10 *data center*s de grande escala da Google. Já a Figura 3.4(b) ilustra uma amostra da variação diária do PUE em um desses *data center*s. O melhor PUE é atingido quando todos os servidores no *data center* estão funcionando perto da capacidade máxima. Sendo assim, desligar os servidores com o intuito de economizar energia acaba, na verdade, aumentando o PUE.

3.1.6. *Comoditização* e consolidação das tecnologias

A infraestrutura dos *data centers* é composta por servidores e redes de conexão, sendo que o custo de manutenção e reposição desta infraestrutura tem se tornado um grande desafio para empresas que oferecem serviços em nuvem considerando que os *data centers* possuem na ordem de 100 mil servidores[5]. Para reduzir tais custos, estas empresas têm se beneficiado do barateamento de equipamentos hoje disponíveis no mercado. Em um primeiro momento, houve uma substituição de servidores de alto desempenho e alto custo por servidores de pequeno porte, porém com baixo custo. Em um segundo momento, a infraestrutura de rede seguiu esta tendência, realizando a substituição dos equipamento de rede. Neste sentido, os provedores de serviços em nuvem estão fazendo uso de um conceito conhecido como *scale-out* ao invés de *scale-up*. *Scale-out* significa instalar equipamentos baratos ao invés de substituir os equipamentos em uso por equipamentos cada vez mais caros. Além disto, o modelo de *scale-out* se beneficia das economias de escala na aquisição massiva de equipamentos, assim

[5]Já em 2006 o Google possuía 450 mil servidores distribuídos em 30 *data centers* [Guo et al. 2008].



(a) Valores médios de PUE de 10 *data center*s. Extraída de [Google 2010d].

(b) Variação diária do PUE do *data center* F. Extraída de [Google 2010d].

Figura 3.4. Valores médios e valores específicos do PUE.

como da consolidação de plataformas por meio das tecnologias de virtualização.

Todo este modelo de utilização de equipamentos menores e mais baratos é conhecido como "comoditização" (*commoditization*). Um dos principais motivadores da *comoditização* dos equipamentos de rede é a disponibilidade de *switches* com preços abaixo de U$$ 100 por porta de 1Gbps e U$$ 1,000 por porta de 10 Gbps [Greenberg et al. 2008]. Apesar destes *switches* não possuírem processamento sofisticado de pacotes e grandes *buffers*, eles oferecem um plano de dados básico operando em altas velocidades. Um outro motivador para utilização deste tipo de equipamento é a capacidade de se criar um plano de controle adaptado às necessidades específicas dos *data centers*, permitindo maior flexibilidade para interconexão dos servidores.

Um outro problema em *data centers* é a capacidade que a infraestrutura de rede oferece para comunicação entre servidores. Estima-se que atualmente os servidores de grandes *data centers* utilizam em média de 5% a 20% de sua capacidade de processamento [Armbrust et al. 2009]. Considera-se um cenário ótimo quando a utilização alcança 30% da capacidade [Greenberg 2009]. Estes números estão consistentes com a realidade uma vez que se tem observado que para muitos serviços o pico de carga excede a média em fatores de 2 a 10, exigindo um super-dimensionamento que suporte tais picos. Esta subutilização da capacidade de processamento dos *data centers* se deve muito a infraestrutura de rede que acaba sendo um ponto de congestionamento para a comunicação entre servidores (mais detalhes na análise de *oversubscription* na Seção 3.2.2.2).

Esta limitação na conectividade entre servidores ocorre devido à infraestrutura de rede hierárquica dos *data centers* atuais, reduzindo a utilização do uso dos recursos computacionais e dificultando o que se denominou de agilidade (*agility*), capacidade de realocar dinamicamente servidores entre os serviços oferecidos pela nuvem, conceito que abordaremos com mais detalhes na Seção 3.2.3.

Neste sentido, a utilização de *switches* "comoditizados" permite que o encaminhamento no plano de dados ocorra somente na camada 2, eliminando a necessidade de

roteadores em vários pontos da rede. Técnicas de engenharia de tráfego tais como *Equal-Cost Multi-Path* (ECMP) são utilizadas permitindo que vários caminhos alternativos entre servidores sejam disponibilizados aumentando assim a capacidade de comunicação e, consequentemente, a capacidade de processamento dos servidores.

3.2. Caracterização dos *data centers* para serviços em nuvem

Com objetivo principal de apresentar os requisitos de rede demandados pelas arquiteturas dos *data centers*, primeiramente apresentamos as características de infraestruturas deste cenário e, então, caracterizamos os serviços e as cargas de trabalho de aplicações típicas. Finalmente, serão discutidas as limitações das abordagens atuais e os requisitos de rede subjacentes em termos de escalabilidade, custo, suporte à virtualização, agilidade, distribuição de carga, tolerância a falhas e segurança.

3.2.1. Infraestrutura dos *data centers*

A infraestrutura da nuvem é formada pelos *data centers* que abrigam os servidores que, mediante diferentes níveis de organização e técnicas de virtualização, oferecem os serviços em nuvem. Portanto, os *data centers* são a manifestação física da computação em nuvem, sendo a infraestrutura de rede a base de comunicações que habilita o paradigma de *cloud computing*, interligando servidores físicos em grande escala. Dependendo do tamanho da própria infraestrutura física e sua localização, os *data centers* podem ser classificados como mega, micro, nano ou baseados em contêineres:

- Mega *data centers*: os chamados mega *data centers* têm na ordem de dezenas de milhares de servidores consumindo dezenas de Mega-Watts de potência. O local para construir estes gigantescos "armazéns" de servidores é escolhido com atenção especial à disponibilidade de recursos como: (1) energia, perto de subestações de grande porte com energia barata e em abundância, ou onde houver fontes de energia naturais, por exemplo, rios ou climatologia adequada no caso de energias renováveis e (2) boa conectividade à Internet. Outros critérios com foco na otimização do custo podem incluir fatores de regulamentação, como incentivos fiscais por emissão reduzida de carbono. Estes mega *data centers* são um ajuste natural para aquelas aplicações de análise de dados (por exemplo, cálculo dos índices de buscadores Web, classificação de documentos ou codificação de arquivos de mídia) ou outras aplicações que requerem enormes quantidades de memória RAM, e/ou exigem um alto número de ciclos de CPU e necessitam de uma grande capacidade de armazenamento em disco rígido. Estes problemas computacionais são atacados de forma distribuída e requerem normalmente uma comunicação intensiva entre os servidores, de modo que o tempo total de computação diminua conforme o atraso entre os servidores é reduzido. Além disso, os custos totais aumentariam se os servidores fossem espalhados por vários centros de dados separados por ligações de longa distância. As aplicações na nuvem são comumente desenvolvidas com base em outros serviços existentes, seguindo uma abordagem de arquiteturas orientadas a serviço (SOA). Um grande número de servidores no mesmo local facilita o projeto dos sistemas e reduz o custo para suportar aplicações com múltiplas dependências e necessidades de comunicação associadas;
- Micro *data centers*: uma área de rápida inovação na indústria é a concepção e implantação de micro *data centers*, com milhares de servidores consumindo centenas de quilowatts. Estes locais são especialmente atrativos para suportar aplicações alta-

mente interativas (por exemplo, e-mail e aplicações de escritório como Google Docs). A localização dos micro *data centers*, também conhecidos como *data centers* satélite, é escolhida estrategicamente em pontos próximos de grandes centros populacionais, para oferecer uma diversidade geográfica que irá minimizar a latência e os custos da rede de trânsito. Hoje, estes micro *data centers* são utilizados principalmente como nós de redes de distribuição de conteúdo (CDNs) localizados em pontos de presença (PoP) de provedores de serviços de Internet (ISP). As principais aplicações são aquelas facilmente distribuídas sem dependências de grandes volumes de trocas entre muitos servidores. Um exemplo típico é o e-mail, onde um fator chave é a interação com o usuário mediante respostas rápidas dos servidores *front-end*. A distribuição geográfica oferece uma opção de projeto com benefícios interessantes em termos de custos, escala, desempenho e confiabilidade. Com os avanços no desenvolvimento e gerência de sistemas de software em nuvem, espera-se que mais *front-ends* de aplicações sejam migrados para estes *data centers*, mantendo os *back-ends* nos centros de maior porte;

- Nano *data centers*: este conceito extremo de *data center* surge da adoção do paradigma *peer-to-peer* (P2P) e da possibilidade de considerar os equipamentos dos usuários finais (por exemplo, os *set-top-boxes*) como uma extensão natural do *data center*, usando os recursos do usuário para migrar a execução de tarefas, armazenar dados e prover serviços através da instanciação de máquinas virtuais. Desta forma, o provedor da infraestrutura da nuvem reduz os custos de adquisição e manutenção dos equipamentos, fazendo com que os serviços e dados fiquem mais perto dos usuários finais. Porém, os desafios de gerenciar um sistema tão distribuído com certas garantias de desempenho, confiabilidade e segurança não são poucos. Isto tem motivado projetos de pesquisa específicos [Nanodatacenters 2010] para este cenário de *cloud computing*, com o apoio de operadoras que já têm este vínculo com o cliente e presença no domicílio do mesmo;

- *Data centers* baseados em contêineres: um outro modelo de disponibilização de *data centers* modularizados é usar contêineres com até 2000 servidores, formando um bloco computacional (ver Figura 3.5) que "só"[6] precisa de energia, água (para refrigeração) e conectividade de rede (fibra óptica) para ser colocado em funcionamento. Esta abordagem modular tem vários atrativos, tanto para os provedores da infraestrutura, quanto para os fornecedores do equipamento, sendo uma forma eficaz de aumentar a capacidade de sua infraestrutura de *data center*. Este tipo de *data center* é uma excelente opção para implantação em locais remotos ou temporários, por exemplo, nas proximidades dos locais de eventos esportivos ou culturais, de forma similar a como estações de telefonia celular são instaladas sob-demanda. Atualmente, a razão mais convincente para sua implantação é a economia de energia (mais do que o aumento de capacidade *ad-hoc*), pois os sistemas de energia são altamente eficientes, podendo economizar de 40% a 50% das despesas operacionais de *data centers* tradicionais. Exemplos comerciais incluem o Portable Modular *data center* da IBM e o Blackbox da Sun. Empresas como Google ou Microsoft já anunciaram o uso deste tipo de infraestrutura.

3.2.2. Visão geral da arquitetura

Independentemente do fator de escala, a infraestrutura do *data center* é constituída por milhares de servidores com suas correspondentes redes de comunicação, subsistemas de ar-

[6] Assumindo os sistemas de software apropriados.

mazenamento, distribuição de energia e extensos sistemas de refrigeração. Há ainda o foco na eficiência dos custos, o fator número um nas decisões de projeto, implementação e procura de novas abordagens na construção e operação desses sistemas. No final das contas, um *data center* é uma fábrica que transforma e armazena bits, tentando maximizar o trabalho útil para cada dólar investido.

Figura 3.5. Um modelo de *data center* baseado em contêineres. Extraída de [Enterprise Control Systems 2010].

3.2.2.1. Infraestrutura de rede

Os servidores físicos são tipicamente montados em conjunto dentro de um *rack* e interligados através de um *switch* Ethernet de acesso. Este *switch* é denominado *Top-of-Rack* (ToR) e usa interfaces de *uplink* de 1 ou 10 Gbps para se interligar com um ou mais *switches* IP/Ethernet de agregação (AGGR), também conhecidos como *switches End-of-Row* (EOR), que agregam os *clusters* de servidores. Este segundo nível de domínio de comutação pode, potencialmente, abranger mais de dez mil servidores individuais. Finalmente, um terceiro nível de *switches* IP/Ethernet é categorizado como *switches*/roteadores de *Core* (CORE), conforme mostra a topologia em camadas da Figura 3.6. A abordagem de topologia em camadas é um fundamento básico dos *data centers* para prover escalabilidade, alto desempenho, flexibilidade, resiliência e facilidade de manutenção. Porém, como veremos nesta seção, a abordagem tradicional (topologia, protocolos de roteamento, separação por VLANs), que tem funcionado razoavelmente bem para os requisitos de *data centers* de redes corporativas, não é suficiente para atender os requisitos dos *cloud data centers* em termos de escala, custo, agilidade e desempenho.

A Figura 3.6 que serve como referência de projetos para *data centers* [Cisco 2007], faz uso de uma sub-camada de serviços representados por um conjunto de ícones e accessíveis pelos *switches* de agregação. Estes serviços são módulos integrados nos *switches* de agregação ou disponibilizados em equipamentos dedicados (*middleboxes* ou serviços externos na Figura 3.6), que proveem serviços tais como balanceadores de carga, roteadores de conteúdo, aceleradores de aplicações (*SSL offload*), segurança (*firewall*, detecção de intrusão, proteção contra ataques de DDoS), análise e monitoramento da rede, etc.

Não estão contemplados na Figura 3.6 os denominados *switches* virtuais rodando nos servidores finais. Em ambientes virtualizados, cada servidor físico pode conter múltiplas máquinas virtuais (VM) gerenciadas por um software de virtualização (*hypervisor*). Este software introduz um *switch* virtual para interligar máquinas virtuais dentro de um mesmo servidor físico e pode estender suas configurações para criar domínios virtuais no nível de *rack*. Esta nova geração de *switches* (por exemplo, Cisco v1000, Open vSwitch) em software é desenvolvida com funcionalidades e interfaces comparáveis aos *switches* físicos, mas com a flexibilidade e velocidade de desenvolvimento de uma implementação em software. Isto facilita extremamente a gerência de redes em ambientes virtuais, além de oferecer numerosas

opções de configuração, isolamento e funcionalidades que seriam intratáveis com técnicas tradicionais de gerência dos *switches* em hardware.

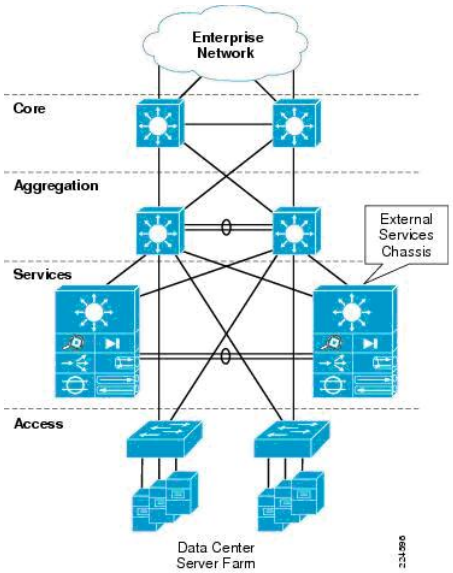


Figura 3.6. Topologia em camadas de um *data center*. Extraída de [Cisco 2007].

Como em outras partes da arquitetura, a escolha da configuração topológica e funcional da camada de rede envolve um compromisso entre capacidade, escala e custo. As propostas de novas topologias de *switches* e mecanismos para o encaminhamento de pacotes diferentes das hierarquias tradicionais serão detalhadas na Seção 3.3. As Figuras 3.7 e 3.8 mostram diferentes topologias em árvore com dois e três níveis de hierarquia, respectivamente. Os modelos de 3 níveis são tipicamente utilizados quando é necessário suportar um grande número de servidores, como no caso dos mega *data centers*.

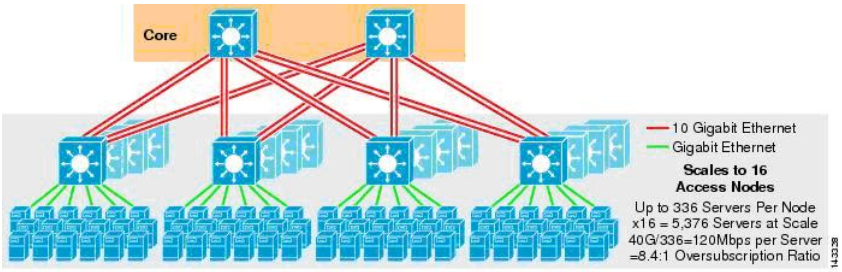


Figura 3.7. Topologia hierárquica de dois níveis e fator de *oversubscription*. Extraída de [Cisco 2007].

Deve ser destacado que os *switches* considerados *commodity*[7] são os que oferecem a melhor relação de custo por porta e por isso são usados como ToRs. *Switches* com um *fan-out*[8] maior, comumente usados na camada de agregação, não mantêm a mesma estrutura de preços. Normalmente, estes *switches* oferecem largura de banda agregada (*bi-section bandwidth*) 10 vezes superiores, mas com um aumento do custo desproporcional, por exemplo, 100 vezes mais caros. Esta descontinuidade nos preços tem motivado a realização de projetos hierárquicos, assim como a utilização exclusiva de *switches commodity* proposto por Al-Fahres [Al-Fares et al. 2008].

[7]Nos dias atuais seria um *switch* Ethernet com até 48 portas de 1-Gbps.

[8]Capacidade que as portas de saída tem de alimentar uma porta de entrada de maior capacidade.

3.2.2.2. Fator de *oversubscription*

Oversubscription no contexto de redes de computadores refere-se à prática de realizar a multiplexação dos recursos de banda para fazer um dimensionamento adequado, que economize a quantidade de enlaces e equipamentos, sem comprometer o desempenho da mesma. Tomemos o seguinte exemplo para ilustrar o fator de *oversubscription* utilizado em *data centers* atuais. Assumindo um *rack* com 40 servidores, cada um interligado a uma porta de um 1Gbps em um *switch* Ethernet de 48-portas de 1Gbps, restarão apenas 8 portas disponíveis para se interligar com a camada de *switches* de agregação. Se todos os servidores quisessem transmitir no máximo da capacidade da interface de rede (40Gbps no total), o tráfego agregado nos *uplinks* seria, no melhor dos casos, de apenas 8 Gbps, o que corresponde a um fator 5 de *oversubscription* para comunicações entre *racks*. Neste tipo de rede, os programadores de aplicações distribuídas devem incorporar estas limitações de recursos da rede na inteligência da aplicação e maximizar o uso de comunicações entre máquinas locais (dentro de um mesmo *rack*), o que complica o desenvolvimento do software e penaliza a eficiência na utilização dos recursos.

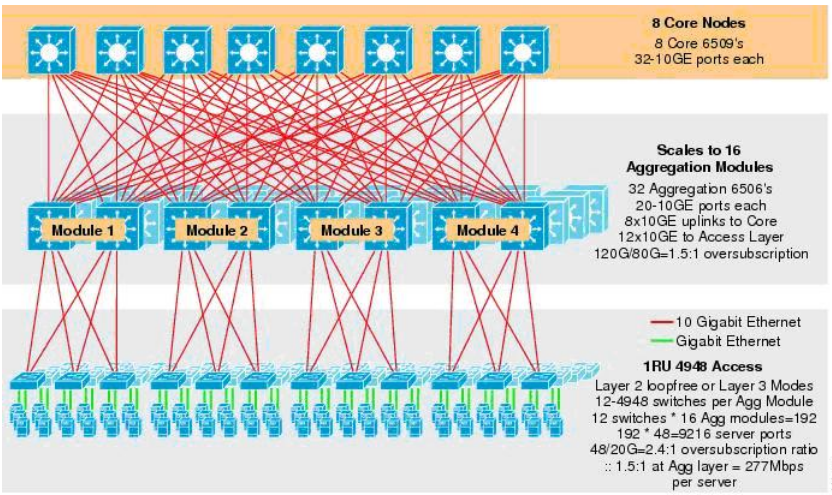


Figura 3.8. Topologia hierárquica de três níveis e fator de *oversubscription*. Extraída de [Cisco 2007].

Aplicando agora o conceito de *oversubscription* ao exemplo da Figura 3.8, o fator total de *oversubscription* será a soma dos valores nos domínios de acesso e de agregação. Neste exemplo, o *switch* ToR é um Catalyst 4948 da Cisco com 48 portas de 1Gbps conectando os servidores e 20 Gbps para *uplink* (cada *switch* possui dois enlaces de 10 Gbps cada), o que corresponde a um fator de *oversubscription* de 2.4:1 (48G/20G) e uma largura de banda real nos servidores de aproximadamente 416Mbps (1Gbps/2.4). Os *switches* de agregação correspondem ao modelo Catalyst 6506 com 20 interfaces de 10Gbps em uma configuração com 8 interfaces para interligar *switches Core* e as outras 12 interligando *switches* ToR, o que resulta em um fator de *oversubscription* de 1.5:1 (120G/80G). Somando ambos os fatores obtemos um fator de *oversubscription* de 3.9:1, o que em termos práticos, limita a largura de banda disponível nos servidores a aproximadamente 277 Mbps (416Mbps/1.5).

Independentemente da capacidade das interfaces de rede dos servidores, por exemplo com a chegada das interfaces de 10Gbps, ao se ter um fator de *oversubscription* superior a 1, a rede se torna um gargalo para matrizes de tráfego onde uma grande parte dos servidores transmitem no máximo da sua capacidade, causando congestionamento nos *switches* e o consequente descarte de pacotes. Arquiteturas de interconexão sem *oversubscription* são bem

conhecidas nas redes de telefonia e nas arquiteturas de HPC, que têm usado topologias de tipo *Clos* ou *fat tree* que garantem a probabilidade de bloqueio zero para qualquer tráfego de entrada [Leighton , Yuan et al. 2007]. Conforme discutiremos nas propostas de novas arquiteturas na Seção 3.3, há interesse em uma rede sem *oversubscription*, porém, os custos associados aos *switches* e cabeamento devem ser justificados pelas demandas de tráfego.

3.2.3. Endereçamento e roteamento IP

Tipicamente, cada aplicação é hospedada em um conjunto específico de servidores (muitas vezes um conjunto de máquinas virtuais). Um *data center* suporta dois tipos de tráfego: (1) o tráfego que entra e sai do *data center* e (2) o tráfego que é gerado e flui apenas internamente ao *data center*. Normalmente, os dois tipos de tráfego são gerados pelas aplicações com algumas especificidades dependendo da aplicação. Por exemplo, em aplicações de busca (*searching*) o tráfego interno domina devido à necessidade de realizar indexações e sincronizações. Por outro lado, em aplicações de vídeo sob-demanda, o tráfego externo prevalece.

Para receber as requisições externas da Internet, uma determinada aplicação possui um ou mais endereços IPs visíveis e válidos publicamente, a fim de que os clientes enviem suas requisições e recebam as respostas. Estes endereços são aqueles obtidos pela resolução de um nome para um endereço IP realizada pelo DNS. Dentro do *data center*, as requisições que chegam da Internet são distribuídas para um conjunto de servidores responsáveis pelo processamento. Normalmente, as comunicações com os usuários externos são atendidas por servidores denominados de *Front-Ends*[9]. Para completar o serviço, estes servidores Web *Front-Ends* tipicamente se comunicam com outros servidores denominados *Back-Ends* para disparar novos (sub-)serviços e acessar os meios de armazenamento distribuídos.

A distribuição das requisições externas aos servidores *Front-Ends* ocorre através do uso de um hardware especializado conhecido como Balanceador de Carga (LB - *Load Balancer*). Neste sentido, uma nova terminologia surge das necessidades de balanceamento de carga: os endereços IPs virtuais (VIPs - *Virtual IPs*) e os endereços IPs diretos (DIPs - *Direct IPs*). Os VIPs representam os endereços públicos, externamente acessíveis na Internet e os DIPs são os endereços internos dos servidores que irão receber e tratar as requisições. A Figura 3.9 ilustra a organização típica de um *data center*.

As requisições de camada 3 (IP) que chegam da Internet utilizando os VIPs são roteadas através dos roteadores de borda e roteadores de acesso até um domínio de camada 2, conforme ditado pelos protocolos de roteamento intra-domínio (tipicamente, IGRP, OSPF). Tais requisições chegam até os LBs que possuem uma lista de endereços DIPs internos que serão utilizados para distribuir cada requisição VIP. Como mostrado na Figura 3.9, todos os servidores que estão conectados a um par de roteadores de acesso pertencem a um único domínio de camada 2. Como há necessidade de rápida convergência durante a ocorrência de falhas e devido às limitações dos protocolos atuais, um único domínio de camada 2 limita-se a possuir no máximo 4000 servidores. Além disso, limitações do protocolo ARP que gera *broadcast* para resolução de endereços, faz com que, tipicamente, o tamanho de uma sub-rede IP não passe de algumas centenas de servidores. Sendo assim, o domínio de camada 2 é dividido em várias VLANs de camada 2, sendo uma sub-rede IP por VLAN, o que fragmenta

[9]Note que, em função do provedor e as características do serviço, os *Front-Ends* podem também estar localizados em micro *data center* satélites para reduzir o tempo de latência com o usuário final.

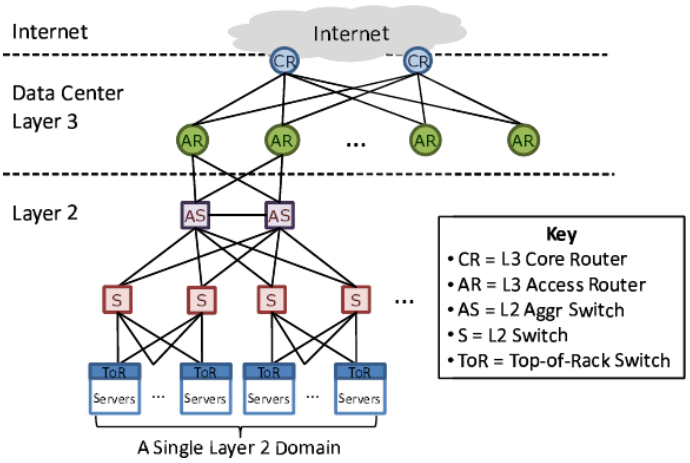


Figura 3.9. Organização de um *data center* tradicional. Extraída de [Greenberg et al. 2008].

os recursos e a capacidade de alocar livremente qualquer endereço IP para qualquer servidor disponível. Neste sentido, surge o termo agilidade definido abaixo:

Agilidade: *é a capacidade de realocar dinamicamente servidores entre os serviços oferecidos pela nuvem com o propósito de otimizar a alocação da carga de trabalho [Greenberg et al. 2009b].*

Sem agilidade, cada serviço deve pré-alocar a quantidade de servidores necessários para atender a demanda em momentos de picos de utilização de um determinado serviço ou em momentos de falhas. Quando um operador de *data center* faz uso da agilidade, é possível atender as flutuações e demandas por serviços individuais através da utilização e distribuição dos serviços entre um conjunto de servidores disponíveis.

Sendo assim, a agilidade em *data centers* tem como objetivo organizar a rede de forma a dar a noção de que cada serviço pode ser alocado em qualquer servidor. Neste sentido, os serviços deveriam utilizar endereços independentes da localização totalmente desacoplados do endereço do servidor. Além disso, se qualquer servidor pode ser alocado para qualquer serviço, é importante que haja isolamento entre os serviços de forma que o desempenho de um serviço não afete o desempenho de outro.

Algumas propostas de novas arquiteturas de *data centers* utilizam este mecanismo. A arquitetura VL2 (apresentada em detalhes na Seção 3.3.2) usa duas famílias de endereços para identificação de *switches* e aplicações. A identificação das aplicações não muda e através de roteamento plano (*flat* ou independente da topologia), a re-alocação de aplicações em diferentes servidores é facilitada.

3.2.4. Software das aplicações em nuvem

A arquitetura típica das aplicações Web é distribuída, seguindo um modelo em camadas: (1) Lógica da aplicação (Ruby on Rails, Scala); (2) caching (Memcache, SQL *query caching*) e (3) *backend* da base de dados (*clusters* RDBMS, CouchDB, BigTable da Google ou Dynamo da Amazon). As interações entre os componentes é realizada através da troca de mensagens assíncronas. Neste ambiente altamente distribuído, cada camada consiste tipicamente de um conjunto de servidores dedicados em executar as respectivas tarefas. A abordagem para au-

mentar a escala da aplicação consiste em aumentar o número de servidores em cada camada, também conhecido como *sharding*.

Esta arquitetura de software funciona bem nas aplicações Web tradicionais, onde as cargas de trabalho, o estado da aplicação e os dados do usuário são facilmente divisíveis. Exemplos deste tipo de aplicação são os serviços de e-mail, onde os usuários acessam os próprios dados, ou dados que são comuns e facilmente armazenáveis em cache, e tipicamente apenas 1-2% dos usuários estão ativos de forma simultânea. Soluções baseadas em DHTs funcionam muito bem para distribuir as cargas de trabalho entre os servidores.

No caso das aplicações de redes sociais, devido ao grande número de usuários, é muito difícil encontrar e gerenciar a quantidade de dependências entre dados e eventos [Pujol et al. 2009]. Por este motivo, este tipo de aplicação é dificilmente escalável, apresentando desafios na distribuição e atualização dos dados entre os servidores, o armazenamento em cache das informações em RAM para serem acessíveis de forma rápida e na propagação dos eventos de atualizações (por exemplo, *Push-on-Change, Pull-on-Demand*). Serviços como Facebook, MySpace, Orkut, Digg ou Twitter são complicados de gerenciar e escalar [Hoff 2009b]. Para termos uma noção da ordem de magnitude, o Facebook possui mais de 30 mil servidores, 300 milhões de usuários ativos e 80 bilhões de fotos. Seus servidores processam 600 mil fotos por segundo, possuem no total 28 TB de cache e produzem 25TB de dados de logs por dia.

Em um site social como o Orkut, considerando um usuário com 200 amigos, cada vez que ele entra na página principal, a aplicação deve coletar simultaneamente o estado dos 200 amigos e apresentar as atualizações. Isto se traduz em 200 requisições simultâneas, mais o processamento das respostas e outra série de serviços que devem ser invocados para obter informações e serviços adicionais. Após o processamento coletivo, o servidor Web *front-end* pode apresentar a página inicial em um tempo razoável.

Usuários com um grande número de seguidores geram bastante processamento nos *data centers* em cada *post*. O grafo da rede social de cada usuário deve ser percorrido para avaliar quem deve receber o *post* (isso ocorre com base nas preferências e configurações do usuário). Atravessar tal grafo em tempo real é uma tarefa que exige processamento. Por exemplo, um usuário da rede social Digg que possui 100 seguidores, gera 300 milhões de "diggs" por dia, 3 mil operações de escrita por segundo, 7GB de dados armazenados por dia e 5TB de dados espalhados entre os 50-60 servidores.

O site social MySpace [Hamilton 2010b] possui 130 milhões de usuários ativos. Aproximadamente 40% da população dos EUA tem conta no MySpace e uma média de 300 mil novos usuários são criados por dia. A infraestrutura é formada por 3 mil servidores Web, 800 servidores de cache e 440 servidores SQL (hospedando mais de 1.000 bancos de dados). Cada servidor SQL roda em um HP ProLiant DL585 (4 processadores *dual core* AMD, 64 GB de RAM). O sistema de armazenamento possui 1.100 discos em uma SAN (*Storage Area Network*), totalizando 1PB de dados dos servidores SQL.

Nessa escala, toda otimização é pouca e já existem trabalhos na área de arquiteturas de aplicações distribuídas otimizadas para redes sociais. Por exemplo, o *One-Hop Replication* (OHR) [Pujol et al. 2009], que considera o particionamento dos dados levando em conta a relação semântica e a importância de cada usuário neste tipo de rede.

Também há um debate intenso sobre novas tendências tecnológicas que não utilizam o modelo tradicional de bases de dados baseadas no modelo SQL. Os atributos das RDBMS com semânticas transacionais do tipo ACID trazem uma rigidez na definição dos esquemas de dados que dificultam a escalabilidade horizontal destes sistemas. Isto tem motivado o surgimento de sistemas "No-SQL", tais como o SimpleDB da Amazon, a BigTable da Google e outros vários projetos No-SQL [No-SQL 2010].

No que se refere a rede, esta deve garantir uma latência baixa e constante para qualquer par de nós que queiram se comunicar, independentemente da sua localização (*intra-rack*, *inter-rack*, *inter-switch*), uma característica que simplifica o projeto e a escalabilidade das aplicações em nuvem.

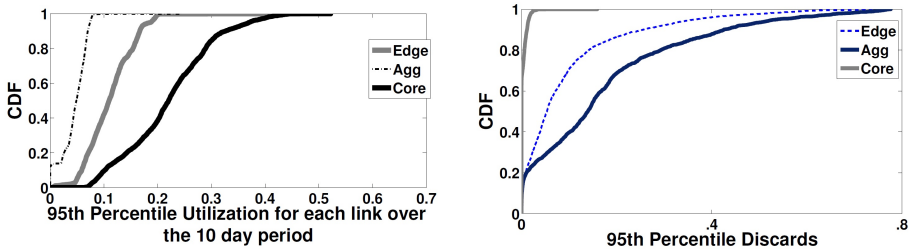
3.2.5. Caracterização do tráfego

Estudos recentes [Benson et al. 2009a, S. Kandula and Patel 2009] têm revelado detalhes sobre as características do tráfego de *data centers* em operação. A conclusão principal é que a matriz de tráfego é altamente variável e dominada por rajadas (picos). Estes fatores dificultam a previsão do comportamento e a aplicação de técnicas tradicionais de engenharia de tráfego, demandando novos mecanismos para reduzir os problemas de congestionamento e prover garantias de uniformidade nas taxas de transferência entre qualquer par de servidores. Além disto, a taxa de utilização dos recursos deve se manter alta e os custos sob controle. Outro dado interessante, referre-se à distribuição do volume, sendo que apenas 20% do tráfego total tem destino/origem na Internet pública. O restante é fruto do desdobramento de novas requisições internas e outros serviços de suporte às operações. Em média, cada servidor contribui com 10 fluxos de tráfego simultâneos.

Com base em dados coletados em 19 *data centers* em funcionamento, o trabalho [Benson et al. 2009a] apresenta um estudo macroscópico das variações temporais e espaciais do tráfego, assim como, as perdas observadas na malha interna dos *switches*. Entre as observações, convém destacar que a carga média nos *switches* do *core* é superior e diminui progressivamente à medida que descemos em direção aos *switches* de acesso (ver Figura 3.10(a)). Por outro lado, as maiores perdas, em média, são superiores nos ToRs e mais reduzidas no *core* (ver Figura 3.10(b)), o que sugere que o tráfego nos enlaces de acesso e agregação possuem um comportamento típico de rajadas. Outra importante observação é que uma pequena fração dos enlaces apresenta perdas muito maiores. Isto indica que seria possível achar rotas por caminhos alternativos que evitem a maior parte das perdas resultantes de *switches* congestionados. Trabalhos recentes que exploram esta abordagem incluem o conceito de *Flyways* de Kandula [Kandula et al. 2009] onde enlaces sem-fio de 60 GHz entre os ToRs são estabelecidos dinamicamente após a detecção de congestionamento. Já explorando um plano ótico reconfigurável, o trabalho [Wang et al. 2009] propõe o estabelecimento dinâmico de caminhos óticos para aliviar os pontos de congestionamento.

Outra observação é que o tráfego nos ToRs segue padrões do tipo ON-OFF, onde os diferentes períodos e o intervalo de tempo entre pacotes podem ser caracterizados como uma distribuição *log-normal*. Os períodos ON-OFF são tipicamente da ordem de milissegundos e, em conjunto, são observados picos de tráfego intensos de duração inferior a 10 segundos. Embora técnicas de balanceamento de carga e re-roteamento possam ser definidas para evitar estes períodos de congestionamento, para serem efetivas, tais técnicas devem ser capazes

de tomar decisões com uma frequência da ordem de segundos. Desta forma, novos mecanismos de engenharia de tráfego para *data centers* devem ser desenvolvidos, para garantir o balanceamento da carga e reagir de forma mais sensível (maior granularidade) que as abordagens tradicionais de redes WAN, uma linha de pesquisa explorada em [Benson et al. 2009b].



(a) Utilização dos enlaces nas diferentes camadas dos *data centers* estudados. Extraída de [Benson et al. 2009a].

(b) Perda de pacotes nas diferentes camadas dos *data centers* avaliados. Extraída de [Benson et al. 2009a].

Figura 3.10. Utilização média e perda de pacotes nas camadas.

Já o estudo apresentado em [S. Kandula and Patel 2009] consiste na instrumentação de 1500 servidores em um *cluster* de *data center* em operação, monitorado durante 2 meses para obter os perfis de tráfego e os padrões e condições de congestionamento. Seguindo uma abordagem baseada na instrumentação dos próprios servidores, ao invés de coletar dados dos *switches*, os autores argumentam que podem ser obtidas métricas mais ricas e confiáveis, evitando a falta de precisão das estatísticas de amostragens por fluxos dos *switches* e o *overhead* de técnicas baseadas em *Deep Packet Inspection* (DPI). A carga de trabalho no *cluster* é fundamentalmente de aplicações de processamento de dados baseadas no modelo MapReduce, onde basicamente os dados a serem processados são divididos em pequenas sub-tarefas, distribuídas entre diferentes servidores escravos, conforme agendado por um servidor mestre. Finalmente, os resultados são agregados e retornados para a aplicação de origem.

A Figura 3.11 mostra o tráfego em $log_e(Bytes)$ trocado entre pares de servidores durante períodos de 10 segundos. A ordenação dos servidores nos eixos replica a localização física dos servidores nos *racks* (20 servidores/*rack*). O trabalho ressalta a identificação de dois padrões de tráfego: (1) *Work-Seeks-Bandwidth* e (2) *Scatter-Gather*:

- *Work-Seeks-Bandwidth*: as formações quadráticas na diagonal do gráfico correspondem ao tráfego trocado entre servidores dentro do mesmo *rack*, como consequência de projetos de programação voltados para a otimização dos recursos da rede. Sendo assim, a distribuição de cargas de trabalhos que envolvem a troca de um volume alto de dados é preferencialmente realizada nas áreas onde a largura de banda disponível entre servidores é maior (dentro do *rack*). Esta é uma prática comum em topologias do tipo árvore com fatores de *oversubscription*, que reduzem a largura de banda efetiva entre os servidores, forçando as aplicações a alocarem o processamento primeiramente em servidores dentro do mesmo *rack*, depois dentro da mesma VLAN e, de menor preferência, em servidores mais distantes. É por este motivo que os autores chamam este padrão de tráfego de *work-seeks-bandwidth*;
- *Scatter-Gather*: o segundo padrão observável da figura vem representado pelas linhas verticais e horizontais, onde um servidor empurra (ou puxa), dados a vários servidores dentro de todo o *cluster*. Este é um claro indicativo das operações de *map* e *reduce* de software de processamento distribuído, onde os dados correspondentes às requisições

são divididos em pequenos pedaços, cada um dos quais é processado por diferentes servidores e os resultados são posteriormente agregados. Por isso o nome de *scatter-gather* ou (dispersão-recolhimento).

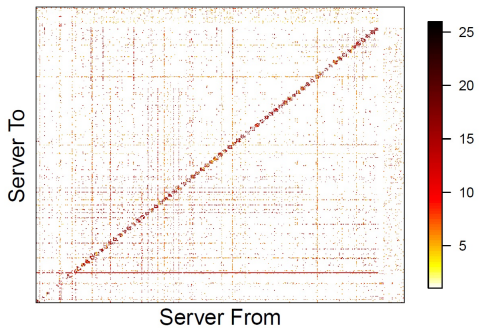


Figura 3.11. Os padrões *Work-Seeks-Bandwidth* e *Scatter-Gather* analisados entre pares de servidores de um *data center*. Extraída de [S. Kandula and Patel 2009].

Desta análise do perfil de tráfego, fica evidente outro requisito das arquiteturas de *data center*: alta largura de banda com taxa de transferência uniforme, independentemente da localização do par de servidores dentro da topologia da rede. Desta forma, (1) qualquer servidor disponível pode ser alocado para as tarefas de computação, liberando a programação do software da complexidade de tratar com a localização dos servidores e (2) o tempo total para conclusão das operações em paralelo não é afetado pelo atraso na obtenção de resultados parciais devido à capacidade da rede.

A Figura 3.12 destaca como o perfil de tráfego agregado varia com uma alta frequência. Os dados da figura correspondem a 165 milhões de fluxos, o total de um dia de operação do *cluster*, revelando que a maior parte dos fluxos são de curta duração (80% duram menos de 10s) e há poucos de longa duração (menos de 0.1% duram mais de 200s). Além disso, mais de 50% dos bytes estão em fluxos que duram menos de 25 segundos.

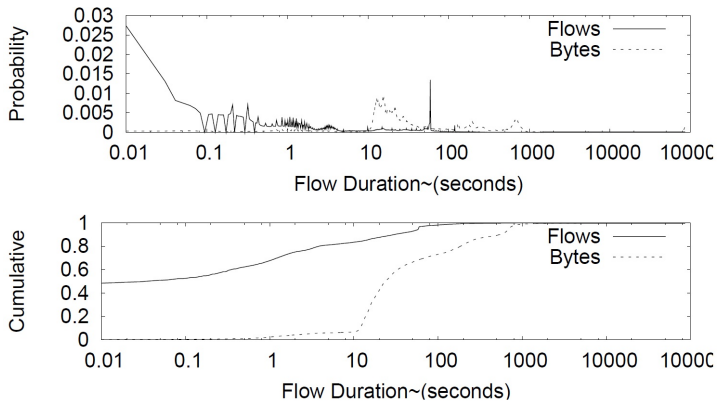


Figura 3.12. Características de tráfego dos *data centers* avaliados. Extraída de [S. Kandula and Patel 2009].

Como já foi observado anteriormente, isto tem implicações interessantes para as técnicas de engenharia de tráfego. Soluções baseadas em decisões centralizadas com visão global da rede são difíceis de implementar, por motivos de escala (quantidade de novos fluxos por segundo) e pela rapidez com que as decisões devem ser tomadas para não atrasar o início dos fluxos. Contribuindo para este desafio, os fluxos de larga duração não são correspondidos com aqueles de maior volume de tráfego. Desta forma, não é suficiente fazer um agendamento especial para este tipo de fluxo "problemático".

A Figura 3.13 ilustra como varia o tráfego no *data center* com o tempo. A sub-figura superior mostra o tráfego agregado de todos os pares de servidores durante um período de 10 horas. Podemos observar que o tráfego muda muito rapidamente, com picos que correspondem a mais da metade da capacidade total de banda da rede. Comunicações *full-duplex* entre os pares de servidores não são a regra, já que tipicamente os papéis de fonte e consumidor dos dados permanecem fixos. Isto indica que durante vários momentos durante um dia típico, os enlaces são utilizados perto da sua capacidade máxima. A outra dimensão na mudança do perfil de tráfego é em relação aos participantes atuais. Embora o tráfego agregado total se mantenha constante, o conjunto de servidores que está se comunicando pode sofrer variações, conforme indica a sub-figura inferior.

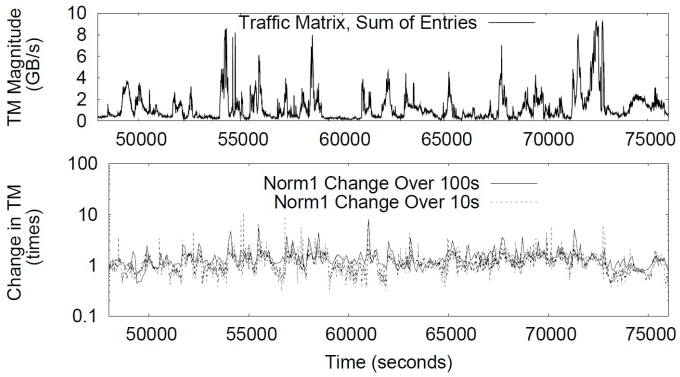


Figura 3.13. Variação do tráfego em termos de quantidade de dados e número de participantes. Extraída de [S. Kandula and Patel 2009].

Em relação aos eventos de congestionamento observados na rede (*hot-spots*), foi avaliada a utilização média dos enlaces com base em uma constante C[10]. A Figura 3.14 ilustra a evidência de numerosos períodos onde a utilização dos enlaces é muito alta (maior do que C). Dentre os 150 *switches* que interligam as 1500 máquinas, 86% dos enlaces sofreram congestionamento de pelo menos 10 segundos e 15% apresentaram períodos de congestionamento de pelo menos 100 segundos. Os períodos curtos de congestionamento (círculos azuis, 10s de alta utilização) estão correlacionados com dezenas de enlaces congestionados, o que indica que são consequência de picos de demanda da aplicação. Por outro lado, períodos largos de congestionamento tendem a se localizar em um número mais reduzido de enlaces.

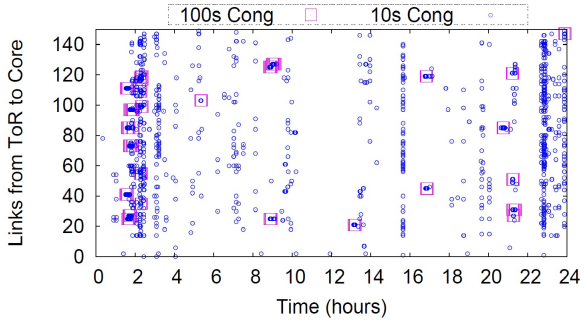


Figura 3.14. Quando e onde o congestionamento ocorre. Extraída de [S. Kandula and Patel 2009].

A Figura 3.15 mostra que a maioria dos períodos de congestionamento tendem a ser de curta duração. De todos os eventos superiores a um segundo, mais de 90% não superam os 2

[10]No artigo foi utilizado um valor de C = 70%, mas a escolha de um limite de 90% ou 95% tem resultados qualitativamente semelhantes.

segundos. Períodos longos de congestionamento também foram observados, contabilizando um total de 665 episódios únicos de congestionamento por mais de 10s. Alguns poucos duraram várias centenas de segundos e o maior durou 382 segundos.

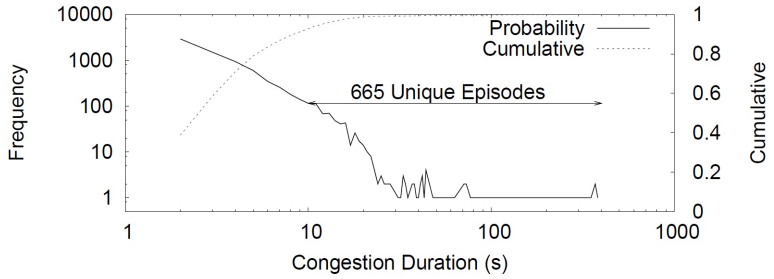


Figura 3.15. Tamanho dos eventos de congestionamento. Extraída de [S. Kandula and Patel 2009].

Idealmente, a rede deve ser operada no maior grau de utilização possível sem prejudicar a taxa de transferência entre os nós. Períodos prolongados de baixa utilização da rede podem significar (1) que a aplicação exige mais de outros recursos, como CPU e disco de rede e está esperando por respostas de algum servidor ou (2) que a aplicação não está otimizada para fazer um melhor (maior) uso dos recursos da rede (banda). Trabalhos recentes [Sonnek and Chandra 2009] têm tentado encaixar VMs com requisitos de CPU, disco (I/O) e rede (I/O) de forma ortogonal, a fim de otimizar a utilização dos recursos no *data center*, uma área de muito interesse e ainda com oportunidades de pesquisa.

3.2.6. Limitações das arquiteturas de rede tradicionais

As práticas atuais de engenharia do *data center* e, em especial, a organização hierárquica L2/L3 da arquitetura de rede, causam uma série de problemas que dificultam a agilidade do sistema (Seção 3.2.3) e apresentam limitações que são resumidas da seguinte forma:

- Fragmentação dos recursos: o mecanismo de roteamento das arquiteturas convencionais atribui, normalmente, endereços IP que possuem significado topológico (aderentes à organização da rede) e são divididos em VLANs, criando um cenário de elevada fragmentação dos servidores. Tal fragmentação introduz rigidez na migração de máquinas virtuais entre VLANs, uma vez que a migração exige a troca do endereço IP para aderir à nova posição topológica. Se uma determinada aplicação cresce e necessita de mais recursos (servidores), tal aplicação não pode utilizar servidores disponíveis e ociosos localizados em outro domínio (sub-rede) de camada 2, causando uma subutilização dos recursos do *data center*;
- Alocação Estática de serviços: as aplicações ficam mapeadas a determinados *switches* e roteadores e necessitam utilizar VLANs para poderem acessar os servidores dedicados a uma aplicação. Apesar das VLANs fornecerem isolamento e segurança, na prática elas fazem com que o tráfego se concentre nos enlaces do topo da hierarquia, causando congestionamento. Além disso, esta alocação estática dificulta a re-alocação dinâmica de serviços. Quando um determinado serviço enfrenta sobrecarga, a intensa utilização da rede faz com que os serviços que compartilham a mesma sub-rede do serviço sobrecarregado também sejam afetados;
- Vazão e latência entre servidores: as arquiteturas convencionais não oferecem capacidade de transmissão suficiente entre os servidores. Organizadas em topologias em

forma de árvores, apresentam taxas de sobrecarga no encaminhamento de tráfego entre diferentes ramos da árvore na ordem de 1:5, ou mais, e concentração de tráfego no mais alto nível da árvore variando entre 1:80 e 1:240 [Greenberg et al. 2009b]. Como consequência, a latência e a vazão se tornam não uniformes dependendo do par de servidores, o que afeta negativamente o desempenho das aplicações nos padrões de tráfego observáveis no *data center*. Além disso, devido à natureza hierárquica da rede, a comunicação entre servidores localizados em diferentes domínios de camada 2 (diferentes VLANs) deve ocorrer através de roteamento em camada 3. Para isso, os roteadores do topo da hierarquia devem possuir alta capacidade de processamento e grandes *buffers*, aumentando ainda mais os custos de TI do *data center*;

- Escalabilidade: os protocolos atuais de roteamento, encaminhamento e gerenciamento não oferecem a escalabilidade necessária para atingir a ordem de grandeza necessária nos *data centers*. As soluções atuais de camada 2 utilizam *broadcast*, criando um cenário não escalável devido ao elevado nível de sinalização. Por outro lado, as soluções de camada 3 exigem a configuração dos equipamentos, seja na definição de sub-redes nas quais os equipamentos estão incluídos ou mesmo na sincronização de servidores DHCP para efetuar a correta atribuição de endereços. Para se ter uma idéia do tamanho destas redes e da complexidade envolvida, considere um *data center* com 100.000 servidores, cada um executando 32 máquinas virtuais. Isto se traduz em mais de três milhões de endereços IP e MAC em um único *data center*;
- Custo dos equipamentos de rede: os balanceadores de carga (LB) tradicionais são equipamentos caros, utilizados em pares em uma configuração 1+1 para tolerância a falhas. Quando estes LBs já não suportam mais a carga, eles são substituídos por novos com mais capacidade, utilizando o modelo *scale-up* ao invés do modelo *scale-out* (mais barato). Este é o modelo seguido também na utilização de *switches* tipo *high-end* de alto desempenho em termos de encaminhamento e "bufferização", com capacidade para suportar picos de processamento e alto consumo de banda;
- Eficiência energética: os *data centers* tradicionais usam mecanismos básicos de refrigeração seguindo a premissa de que se o *data center* cresce, instala-se mais equipamentos de refrigeração, causando um impacto significativo no consumo de energia mensal. Além dos aspectos de refrigeração e sistemas de distribuição da energia na infraestrutura, ainda há uma margem alta de eficiência energética que não é atingida pelos servidores e os equipamentos de rede atuais. Especialmente em relação à proporcionalidade do consumo de energia em função da carga de trabalho pois, nos projetos atuais, o consumo dos equipamentos (e da rede como um todo) trabalhando com uma carga mínima, não consomem proporcionalmente menos energia do que trabalhando ao máximo da sua capacidade.

Conclui-se, baseado nos comentários anteriores, que as técnicas convencionais (*Spanning Tree*, VLAN, criação de sub-redes) não oferecem o desempenho (alta taxa de transferência uniforme fim-a-fim e baixa latência para qualquer par de nós da estrutura) nem a agilidade (flexibilidade na gerência do endereçamento IP/Ethernet) necessários. Vale a pena destacar que estas limitações não são resolvidas simplesmente com um aumento da capacidade de transmissão dos enlaces. Embora uma baixa latência entre servidores seja um dos objetivos, uma latência determinística e consistente entre qualquer par de servidores é igualmente importante para suportar as aplicações em nuvem.

3.2.7. Objetivos e requisitos das arquiteturas de rede para *data centers*

Além dos requisitos óbvios de confiabilidade e desempenho, uma série de novos requisitos em termos de arquitetura e sistema são fundamentais para aumentar a escalabilidade e a eficiência e reduzir os custos dos *cloud data centers*. Um fator chave é habilitar o desacoplamento das aplicações da própria infraestrutura e transformar o *data center* (como um grande computador) em um sistema ágil, de alto desempenho e eficiente. Estes *data centers*, porém, não devem ser confundidos arquiteturalmente com a Internet, ou simplesmente reduzidos a técnicas de virtualização (servidor, armazenamento, rede) e segurança avançadas.

As diferenças fundamentais em relação à infraestrutura de rede em um *data center* comparadas com redes tradicionais locais ou intra-domínio são: (1) a topologia é definida pelo arquiteto da rede; (2) o número de nós pode ser alto, mas é conhecido; (3) serviços de *broadcast/multicast* não são requeridos; (4) os aspectos de segurança ficam simplificados (não há ataques maliciosos, apenas pequenos erros de configuração) e (5) há opções de controle sobre a distribuição das aplicações para minimizar a possibilidade de gargalos.

As redes para *data centers* de grande escala, como as utilizadas em infraestruturas para computação em nuvem, possuem uma contribuição importante para o custo total da infraestrutura. Especialmente os *switches*/roteadores de grande porte têm custos por porta muito altos comparativamente. O software dos roteadores tem evoluído de forma fechada e proprietária, gerando um código legado desnecessariamente grande e complexo sendo apontado como a causa comum de muitos dos problemas de confiabilidade. O serviço de manutenção, assim como, as requisições de personalização e novas funcionalidades são providas apenas pelo fornecedor do equipamento e sempre requerem que o equipamento seja enviado até o local de atendimento.

Estes fatores associados às limitações da arquitetura de rede tradicionais, têm motivado a procura por novos projetos que atendam melhor os requisitos, aumentem o desempenho, melhorem a confiabilidade e reduzam custos. Desta forma, os novos projetos podem ser feitos na medida em que as necessidades particulares do provedor da infraestrutura aumentem e novos serviços sejam ofertados. Isto evita que os provedores paguem por funcionalidades dos equipamentos que não são necessárias e que ficam ociosas na maior parte do tempo. Desta forma, podemos definir algumas metas de projeto das arquiteturas de rede para *cloud data centers*:

- Redução da dependência de *switches* de grande porte no *core* da rede;
- Simplificação do software de rede (plano de controle, pilha de protocolos);
- Aumento da confiabilidade do sistema como um todo;
- Evitar que a rede seja o gargalo do sistema e simplificar o trabalho do desenvolvimento de aplicações;
- Redução dos custos de capital e operacionais.

Com base nas limitações e metas apontadas acima, compilamos os requisitos da arquitetura de rede e os dividimos da seguinte forma:

- Agilidade: endereçamento e encaminhamento de pacotes que permitam levantar (e mover) qualquer máquina virtual em qualquer servidor físico;

- Escalabilidade: roteamento e endereçamento escaláveis, tabelas de encaminhamento (L2 Ethernet) com tamanho gerenciável e tempos de convergência aceitáveis. O objetivo final é poder escalar a uma ordem de centenas de milhares de servidores e milhões de máquinas virtuais. Sendo assim, dois aspectos precisam ser considerados: o tamanho das tabelas de encaminhamento e os *broadcasts*, dois grandes problemas para redes de grande escala;
- Desempenho: vazão e latência uniformes e constantes para qualquer padrão de tráfego (1:1, 1:M, N:N) e entre qualquer par de servidores. Garantir o isolamento entre tráfegos e evitar pontos de congestionamentos. A capacidade de comunicação entre servidores deve ser limitada apenas pela capacidade das interfaces;
- Controle: flexibilidade para inserção de serviços de *middleboxes*, monitoramento e resolução de problemas. A infraestrutura também deve ser capaz de incorporar novos servidores sem a necessidade de considerar a topologia. O controle dos aspectos de segurança deve estar incorporado aos sistemas de gerência;
- Confiabilidade: o projeto da infraestrutura deve suportar falhas de servidores, *switches*, etc. Devido ao grande número de equipamentos, as chances de falhas são maiores e precisam ser contornadas para que o sistema se mantenha operacional;
- Custo: baixo custo de capital e operacional. Melhorar os esforços de configuração através de mecanismos automatizados ajudam a reduzir o custo de manutenção dos sistemas. Usar hardware *comoditizado* reduz o custo de capital. Isto envolve também novos métodos de estruturação e cabeamento, procurando modularidade da infraestrutura do *data center*, que contribuam para uma redução dos custos. Além disso, melhorar a eficiência energética é fazer com que, idealmente, o consumo de energia seja proporcional à carga de trabalho, um problema bastante desafiador.

Cada um destes requisitos pode ser resolvido separadamente com as tecnologias de rede atuais. Porém, satisfazer todos eles ao mesmo tempo é uma tarefa de grandes desafios, principalmente porque atender a um determinado requisito implica, muitas vezes, em não conseguir atender a outro. Por exemplo, um fator de *oversubscription* alto resulta em uma redução do desempenho real entre os servidores em relação a algumas demandas de tráfego. Ao mesmo tempo, um projeto de rede sem *oversubscritpion* implica em um alto custo devido à quantidade necessária de elementos de rede.

A próxima seção apresenta algumas das principais arquiteturas de *data centers* atuais que buscam atender aos requisitos de rede listados acima. Algumas arquiteturas privilegiam certos requisitos em detrimento de outros. Entretanto, como veremos, a maioria das arquiteturas atende em maior ou menor grau a todos os requisitos citados.

3.3. Novas propostas de arquiteturas para *data centers*

Atualmente existe um grande esforço da comunidade (indústria e acadêmica) em busca do desenvolvimento de novas arquiteturas de *data centers* voltadas para a resolução dos problemas citados nas seções anteriores. Nesse sentido, apresentamos quatro arquiteturas que também podem ser consideradas como as mais significativas e que representam o universo das pesquisas realizadas nesta área. Estas quatro propostas envolvem as principais características necessárias para suportar os *cloud data centers* e servem como base para compreendermos o que de fato é necessário neste tipo de infraestrutura.

3.3.1. Monsoon

Conforme discutido na Seção 3.2, as aplicações contidas nos *data centers* atuais sofrem com a fragmentação interna dos recursos, rigidez e limitação de banda que são impostos pela arquitetura de rede que conecta os servidores. Por exemplo, arquiteturas convencionais de rede utilizam mecanismos estáticos para mapear os serviços oferecidos pelo *data center* em VLANs, limitadas em algumas centenas de servidores devido ao nível de sinalização (*overhead*) gerado no plano de controle da rede. Além disso, a utilização de equipamentos no nível IP para efetuar a distribuição de tráfego entre diversas VLANs, em conjunto com os balanceadores de carga necessários para espalhar as requisições entre os servidores, elevam o custo de implantação dos *data centers*. Basicamente, este cenário é caracterizado pela concentração de tráfego em alguns poucos equipamentos, resultando em frequentes substituições de hardware para atender a nova demanda do *data center*.

A filosofia defendida pelo Monsoon [Greenberg et al. 2008] é a *comoditização* da infraestrutura como forma de obter escalabilidade e baixo custo, ou seja, o Monsoon adiciona equipamentos novos e baratos (*scale-out*) para atender a nova demanda, ao invés da constante troca por equipamentos mais potentes (*scale up*). O Monsoon estabelece uma arquitetura de *data center* organizada em formato de malha com o objetivo de comportar 100.000 ou mais servidores. Para criar esta malha são utilizados *switches* programáveis de camada 2 (*comoditizados*) e servidores. Modificações no plano de controle dos equipamentos são necessárias para suportar, por exemplo, roteamento com rota na origem. As modificações no plano de dados, por sua vez, visam suportar o encaminhamento de dados por múltiplos caminhos através do *Valiant Load Balancing* (VLB). Nesta proposta, a função de balanceamento de carga é compartilhada por um grupo de servidores. Consequentemente, os equipamentos responsáveis pelo balanceamento de carga podem ser distribuídos pelos *racks* do *data center*, oferecendo maior agilidade e menor fragmentação na utilização dos recursos disponíveis.

3.3.1.1. Arquitetura

Os aspectos principais do Monsoon podem ser definidos como: (1) engenharia de tráfego utilizando uma malha de camada 2; (2) escalabilidade para criar grandes domínios de camada 2 e (3) espalhamento de carga ubíquo[11].

A Figura 3.16 apresenta uma visão geral da arquitetura do Monsoon. Os dois principais aspectos da arquitetura são: (1) a definição de uma única rede de camada 2 na qual todos os 100.000 servidores são conectados e (2) a flexibilidade pela qual as requisições podem ser distribuídas entre os diversos conjuntos de servidores.

Se observadas a partir de uma perspectiva arquitetural de mais alto nível, as diferenças entre as camadas 2 (Ethernet) e 3 (IP) estão diminuindo, especialmente para redes contidas dentro de um mesmo prédio. Entretanto, existem alguns fatores práticos que levaram o Monsoon a optar pela camada 2 como tecnologia para criar um único domínio no qual todos os servidores estão conectados: (1) cortar custos; (2) eliminar a fragmentação de servidores e (3) diminuir o distúrbio às aplicações. Considerando estes fatores, o Ethernet é claramente a tecnologia eleita, pois está abaixo do IP e já apresenta custo e desempenho otimizados para o encaminhamento baseado em endereços planos. Atualmente, uma porta Ethernet custa entre

[11] Espalhamento de carga ubíquo, neste contexto, significa distribuir todo o tráfego presente no interior do *data center* através de sua estrutura total de *switches* para melhor aproveitamento da infraestrutura.

10% e 50% do valor de uma porta de velocidade equivalente de camada 3 e um *data center* possui centenas de milhares de portas.

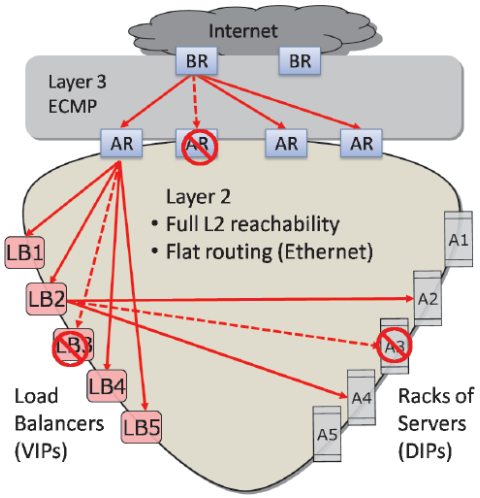


Figura 3.16. Visão geral da arquitetura do Monsoon. Extraída de [Greenberg et al. 2008].

Como observado na Figura 3.16, o domínio de camada 2 provê total conectividade entre os servidores do *data center*. A comunicação entre os servidores utiliza a taxa máxima das interfaces de rede, sem que ocorram enlaces sobrecarregados, ou seja, todos os servidores podem comunicar entre si a 1 Gbps. Ainda na Figura 3.16, a porção da rede que utiliza a camada 3 é necessária para conectar o *data center* à Internet, utilizando roteadores de borda (BRs - *Border Routers*) e o *Equal Cost MultiPath* (ECMP) para espalhar as requisições igualmente entre os roteadores de acesso (ARs - *Access Routers*).

Assim que as requisições adentram o *data center*, os roteadores de acesso utilizam técnicas de *hash* consistente para distribuir as requisições, uniformemente, entre os diversos balanceadores de carga (LBs - *Load Balancers*) que estão associados a um endereço IP virtual (VIP - *Virtual IP*) de uma determinada aplicação visível publicamente. Finalmente, os balanceadores de carga utilizam funções de distribuição específicas de cada aplicação para espalhar as requisições no conjunto de servidores, identificados pelos endereços IP diretos (DIPs - *Direct IPs*), que estão executando a aplicação desejada.

A Figura 3.16 indica a existência de um mecanismo de recuperação de falhas, sejam elas em qualquer dos roteadores de acesso, balanceadores de carga ou servidores. Um serviço de recuperação (*health service*) continuamente monitora o estado de todos os equipamentos que constituem o *data center*. Por exemplo, um servidor que venha a falhar é imediatamente removido do conjunto de servidores associados a uma determinada aplicação, evitando que novas requisições sejam encaminhadas para o servidor em falha.

3.3.1.2. Encaminhamento Servidor-a-Servidor

O Monsoon utiliza tunelamento MAC-in-MAC para encaminhar pacotes entre os servidores que constituem o *data center*. Para tanto, é utilizado um serviço de diretório no qual a lista de endereços MAC dos servidores responsáveis pelas requisições, bem como o endereço MAC dos *switches* nos quais os servidores estão conectados, é mantida.

A Figura 3.17 apresenta a pilha de rede implementada pelos servidores do Monsoon. Nesta pilha, a tradicional função de ARP é desativada e substituída por um processo executado em espaço de usuário (*Monsoon Agent*) e uma nova interface MAC virtual (*Encap-*

sulator). Note que estas mudanças são imperceptíveis para as aplicações. Basicamente, o encapsulador recebe pacotes vindos da camada de rede (IP) e consulta seu cache de resoluções MAC em busca da informação de encaminhamento. Caso não exista, o encapsulador solicita ao agente uma nova resolução através do serviço de diretório.

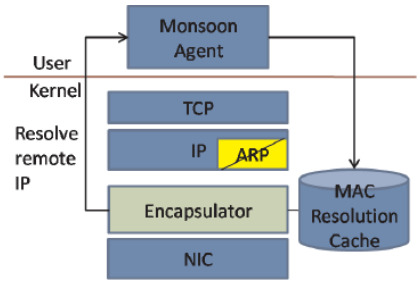


Figura 3.17. Pilha de rede implementada pelos servidores do Monsoon. Extraída de [Greenberg et al. 2008].

O serviço de diretório resolve o endereço IP de destino e retorna uma lista contendo todos os endereços MAC dos servidores associados ao serviço solicitado. Com base nesta lista, o encapsulador escolhe um servidor (seleciona seu MAC), encontra o MAC do *switch* de topo de *rack* (TOR) no qual o servidor está conectado e, finalmente, escolhe um *switch* intermediário no qual os pacotes serão enviados (*bounce off*) por questões de balanceamento de carga do VLB. Este conjunto de informações corresponde a um único fluxo de dados e é mantido em cache para que todos os quadros do fluxo recebam o mesmo tratamento. A Figura 3.18 detalha o encaminhamento dos quadros entre origem e destino.

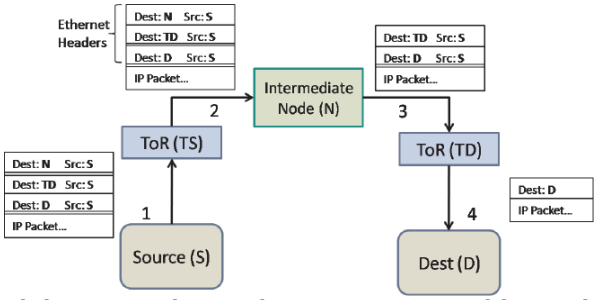


Figura 3.18. Encaminhamento de quadros entre os servidores de origem e destino via encapsulamento MAC-in-MAC. Extraída de [Greenberg et al. 2008].

3.3.1.3. Conectividade externa

A Figura 3.19 apresenta o caminho utilizado pelas conexões que são originadas ou destinadas de/para a Internet. Basicamente, todo o tráfego entra ou sai do *data center* através dos roteadores de borda que, por sua vez, estão conectados a um conjunto de roteadores de acesso utilizando uma rede de camada 3, na qual o ECMP é implementado. Entretanto, os roteadores de acesso não suportam as primitivas de espalhamento e encapsulamento do Monsoon. Sendo assim, cada roteador de acesso possui um servidor de ingresso associado a ele e todo tráfego externo que chega ao roteador de acesso é encaminhado para este servidor de ingresso para receber o tratamento adequado do Monsoon.

Cada servidor de ingresso possui duas interfaces de rede, uma conectada diretamente ao roteador de acesso e a outra conectada à rede do *data center* através de um *switch* TOR. No caso dos pacotes vindos da Internet, o servidor de ingresso utiliza o serviço de diretório para resolver o endereço IP e encaminha o tráfego dentro da rede do *data center* como qualquer outro servidor. Para os pacotes na direção da Internet, o serviço de diretório mapeia o endereço MAC do *gateway default* para o endereço MAC dos servidores de ingresso, possibilitando assim a saída dos pacotes para a Internet.

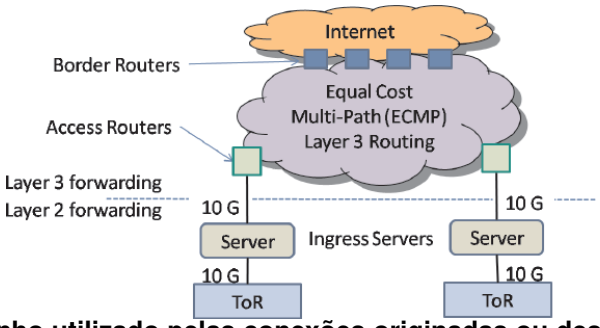


Figura 3.19. Caminho utilizado pelas conexões originadas ou destinadas à Internet. Extraída de [Greenberg et al. 2008].

3.3.1.4. Topologia de *switches*

A Figura 3.20 apresenta um exemplo concreto de uma topologia capaz de conectar 103.680 servidores em um único domínio de camada 2, na qual o VLB seria bem suportado. Neste cenário, cada *switch* TOR possui 2 portas Ethernet de 10Gbps que são conectadas a dois *switches* diferentes de ingresso/egresso localizados na região central (*core*) do *data center*, por questões de tolerância a falhas.

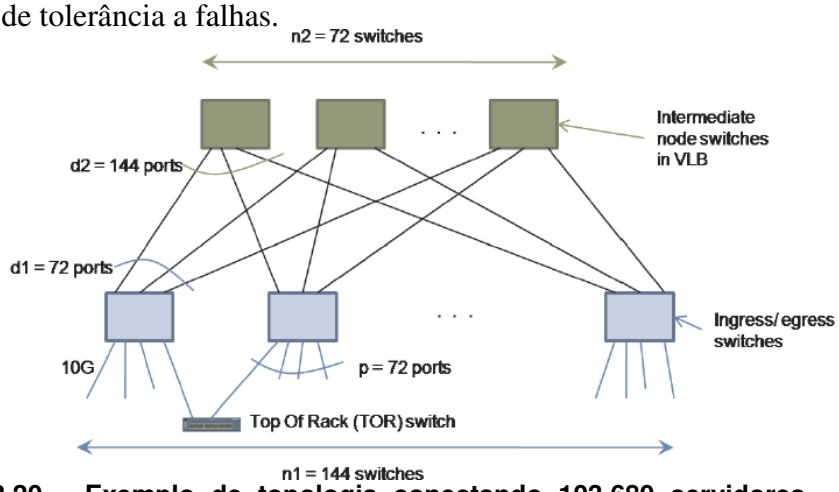


Figura 3.20. Exemplo de topologia conectando 103.680 servidores. Extraída de [Greenberg et al. 2008].

A região central é organizada em dois níveis (*n1* e *n2*) de *switches*. Neste exemplo, o nível *n1* possui 144 *switches*, representados pelos *switches* cinza claro, que não possuem qualquer ligação entre eles, mas cada um deles está conectado a todos os *switches* intermediários (nível *n2*) através de portas Ethernet de 10 Gbps. Sendo assim, existem 72 *switches* no nível *n2* (intermediários), representados na figura em cinza escuro, tornando a topologia interessante para o VLB, pois os fluxos podem escolher os *switches* intermediários nos quais eles irão alcançar a partir deste conjunto com 72 *switches*.

Finalmente, considerando-se o número de portas e *switches* presentes no nível *n1*, esta topologia é capaz de conectar 5184 *racks* com 20 servidores cada, totalizando 103.680 servidores em um único domínio de camada 2, possibilitando que cada servidor transmita à taxa máxima de sua interface (1 Gbps).

3.3.2. VL2

O VL2 é uma proposta de nova arquitetura de *data center* e pertence ao mesmo grupo de pesquisa da Microsoft que originou o Monsoon. Sendo assim, o VL2 pode ser considerado como uma evolução do Monsoon, introduzindo alguns refinamentos.

O VL2 [Greenberg et al. 2009b] trata as limitações citadas na Seção 3.2.6 através da criação de uma camada 2 virtual (VL2 - *Virtual Layer 2*). Esta camada virtual provê aos serviços do *data center* a ilusão de que todos os servidores associados a eles, e apenas os servidores associados a eles, estão conectados através de um único *switch* de camada 2 (livre de interferências) e mantém esta ilusão mesmo que o tamanho de cada serviço varie entre 1 ou 100.000 servidores. Para atingir esta meta, é preciso construir uma rede que honre três objetivos: (1) a comunicação servidor-a-servidor só pode ser limitada pela taxa de transmissão das interfaces de rede de cada servidor (1 Gbps); (2) o tráfego gerado por um serviço deve ser isolado de tal forma a não afetar o tráfego de outros serviços e (3) o *data center* deve ser capaz de alocar qualquer servidor para atender a qualquer serviço (agilidade), possibilitando a atribuição de qualquer endereço IP àquele servidor, de acordo com as exigências do serviço e independente da sua localização no *data center*.

Além de propor uma arquitetura que busca prover agilidade na alocação de servidores, o VL2 investiga que tipo de solução poderia ser construída utilizando os recursos disponíveis atualmente, evitando qualquer tipo de modificação no hardware de *switches* e servidores, oferecendo ainda, um cenário transparente para aplicações legadas. Sendo assim, uma prática comum em *data centers* é a modificação do software (sistemas operacionais) utilizados nos servidores. Neste contexto, o VL2 propõe uma reorganização nos papéis desempenhados tanto pelos servidores quanto pela rede, através da introdução de uma camada de software (*shim*) na pilha de protocolos implementada pelos servidores, de tal forma a contornar as limitações impostas pelos dispositivos legados de rede.

3.3.2.1. Arquitetura

A Figura 3.21 apresenta a topologia utilizada pelo VL2, um *backbone* com elevada conectividade entre os *switches* de agregação e intermediários. Os *switches* ToR são conectados a dois *switches* de agregação e, devido ao elevado número de conexões disponível entre qualquer par de *switches* de agregação, a falha de qualquer um dos *n switches* intermediários reduz em apenas $1/n$ a largura de banda disponível, garantindo uma lenta degradação do serviço oferecido pelo *data center*.

A rede é constituída por duas classes de endereços, os endereços com significado topológico (LAs - *Locator Addresses*) e os endereços planos de aplicação (AAs - *Application Addresses*). Neste cenário, a infraestrutura de rede utiliza endereços LA, que são atribuídos para todos os *switches* e suas interfaces. Além disso, todos os *switches* executam um protocolo de roteamento baseado no estado do enlace para disseminar estes LAs, oferecendo uma visão global da topologia formada pelos *switches* e possibilitando o encaminhamento de pacotes encapsulados em LAs através de caminhos mais curtos. Por outro lado, as aplicações utilizam AAs que permanecem inalterados, independentemente da maneira como os servidores migram no interior do *data center*. Para todo AA é associado o LA atribuído ao *switch* ToR no qual o servidor está conectado. Este mapeamento é mantido por um serviço de diretório do VL2.

A malha de camada 3 formada pelo VL2 cria a ilusão de um único domínio de camada 2 para os servidores no interior do *data center*, uma vez que os servidores imaginam pertencer a uma mesma VLAN. Note que todos os servidores na Figura 3.21 possuem um endereço AA alocado a partir da faixa 20/8. Neste cenário de camada 3, as requisições vindas da Internet podem fluir diretamente até os servidores, sem serem forçadas através

de *gateways* específicos nos quais os pacotes são re-escritos como ocorre, por exemplo, no Monsoon [Greenberg et al. 2008]. Para tanto, endereços LA adicionais são atribuídos aos servidores a partir de uma faixa de IPs válidos (alcançáveis) na Internet.

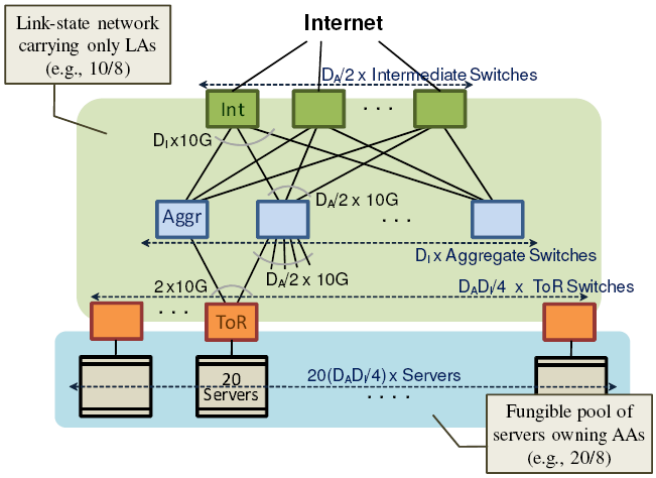


Figura 3.21. Exemplo de *backbone* utilizado pelo VL2. Extraída de [Greenberg et al. 2009b].

3.3.2.2. Endereçamento e roteamento

A Figura 3.22 apresenta um exemplo de encaminhamento de pacotes através da estrutura do VL2. Basicamente, para rotear tráfego entre servidores identificados por endereços AA em uma rede que possui rotas formadas a partir de endereços LA, um agente VL2 executando em cada um dos servidores intercepta os pacotes originados e os encapsula em pacotes endereçados ao LA do *switch* ToR associado ao servidor de destino. Existe ainda um terceiro cabeçalho encapsulando todos os pacotes por razões de espalhamento de tráfego que será detalhado na Seção 3.3.2.3. Basicamente, o sucesso do VL2 está associado ao fato dos servidores acreditarem compartilhar uma única sub-rede IP, devido ao encapsulamento efetuado.

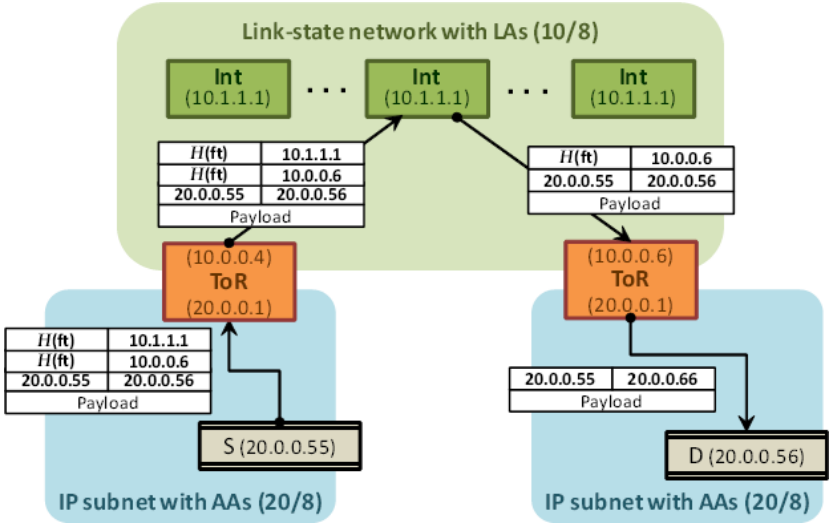


Figura 3.22. Exemplo de encaminhamento de pacotes em uma rede VL2. Extraída de [Greenberg et al. 2009b].

Como forma de contornar o *broadcast* gerado pelo protocolo ARP, na primeira vez em que um servidor envia pacotes para um determinado endereço AA, a pilha de rede original do servidor gera uma requisição ARP e a envia para a rede. Neste instante, o agente VL2

presente no servidor intercepta a requisição ARP e a converte em uma pergunta *unicast* para o serviço de diretório do VL2. O serviço de diretório, por sua vez, responde com o endereço LA do ToR de destino no qual os pacotes devem ser tunelados e o agente VL2 armazena este mapeamento, em um procedimento similar ao cache original do ARP, evitando novas perguntas ao serviço de diretório.

3.3.2.3. Espalhamento de tráfego por múltiplos caminhos

O VL2 combina o VLB e o ECMP como forma de evitar áreas de concentração de tráfego. O VLB é utilizado para distribuir o tráfego entre um conjunto de *switches* intermediários e o ECMP é utilizado para distribuir o tráfego entre caminhos de custo igual. A combinação de ambos os mecanismos cria um cenário de distribuição de tráfego mais eficaz, uma vez que as limitações presentes em um mecanismo são tratadas pelo outro.

A Figura 3.22 ilustra como o agente VL2 utiliza o encapsulamento para implementar o VLB e enviar o tráfego através de um *switch* intermediário escolhido aleatoriamente. Em suma, entre a origem e o destino, o pacote é enviado aos *switches* intermediários, desencapsulado por este *switch*, encaminhado para o ToR de destino, desencapsulado novamente e, finalmente, enviado ao destinatário. Este processo de encapsulamento de pacotes na direção de um *switch* intermediário satisfaz o VLB. Entretanto, eventuais falhas nestes *switches* intermediários poderiam levar a um cenário no qual um elevado número de agentes VL2 teriam de ser atualizados para convergir ao novo estado da rede.

O VL2 contorna esta situação atribuindo o mesmo endereço LA para todos os *switches* intermediários (10.1.1.1 na Figura 3.22). Note que na topologia adotada pelo VL2, todos os *switches* intermediários estão a exatos três saltos de distância dos servidores de origem, criando o cenário adequado para a utilização do ECMP. Sendo assim, o ECMP assume a responsabilidade de entregar os pacotes para um dos *switches* intermediários e, em caso de falhas, o ECMP reage, enviando os pacotes para um *switch* que esteja operacional, eliminado a necessidade de avisar os diversos agentes VL2.

3.3.2.4. Serviço de diretório

O serviço de diretório do VL2 provê três serviços principais: (1) consultas; (2) atualizações de mapeamentos entre AAs e LAs e (3) um mecanismo de atualização de cache reativo para atualizações sensíveis a atrasos (por exemplo, a atualização entre um AA e um LA durante o processo de migração de uma máquina virtual).

Considerando os requisitos de desempenho e os padrões de consultas e atualizações, o VL2 define uma arquitetura de dois níveis conforme ilustra a Figura 3.23. O primeiro nível, considerando-se um *data center* com aproximadamente 100.000 servidores, possui entre 50 e 100 servidores de diretório (DS - *Directory Servers*) otimizados para leitura (consultas), utilizados para replicar os dados do serviço de diretório e responder a consultas dos agentes VL2. No segundo nível, existe um número pequeno, entre 5 e 10, de servidores otimizados para escrita (RSM - *Replicated State Machine*), que oferecem um serviço consistente de armazenamento.

Cada servidor DS possui em cache todos os mapeamentos AA-LA disponíveis nos servidores RSM e responde às consultas feitas pelos agentes VL2 de forma independente. Para estes servidores DS, não há a necessidade do oferecimento de elevados níveis de consistência. Sendo assim, as sincronizações com os servidores RSM ocorrem a cada 30 segun-

dos. Em caso de atualizações na rede, o sistema envia a nova informação para um servidor DS que, por sua vez, encaminha a atualização para um servidor RSM. O servidor RSM replica esta informação entre todos os servidores RSM e, finalmente, envia uma mensagem de confirmação para o servidor de diretório que originou a atualização.

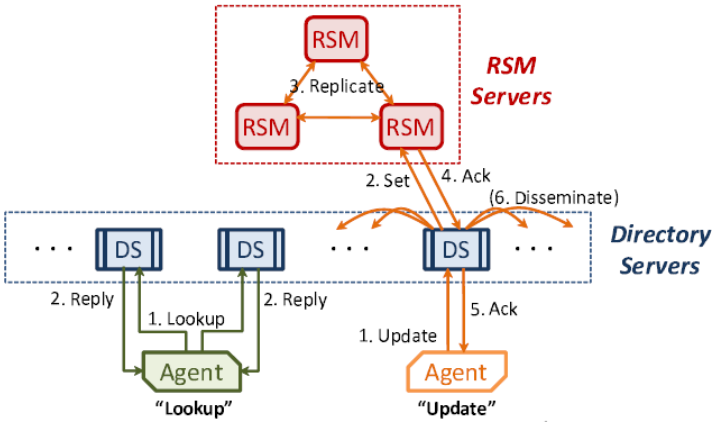


Figura 3.23. Arquitetura do serviço de diretório do VL2. Extraída de [Greenberg et al. 2009b].

3.3.3. Portland

Esta seção introduz o PortLand [Mysore et al. 2009], um conjunto de protocolos compatíveis com Ethernet para efetuar roteamento, encaminhamento e resolução de endereços. O PortLand é desenvolvido considerando-se a estrutura organizacional comumente encontrada em *data centers*, ou seja, uma árvore com múltiplos nós na raiz (denominada *fat tree*). O Portland pode ser considerado como uma versão mais viável e refinada da proposta anterior de um dos autores [Al-Fares et al. 2008], baseada também em uma topologia *fat-tree* e *switches comoditizados*, mas fundamentada em um esquema de roteamento IP customizado.

Com base neste cenário, o PortLand propõe: (1) a utilização de um protocolo para possibilitar que os *switches* descubram sua posição (topológica) na rede; (2) a atribuição de Pseudo endereços MAC (PMAC) para todos os nós finais, de forma a codificar suas posições na topologia; (3) a existência de um serviço centralizado de gerenciamento da infraestrutura de rede (*Fabric Manager*) e (4) a implantação de um serviço de *Proxy* para contornar o *broadcast* inerente ao ARP.

3.3.3.1. Arquitetura

A Figura 3.24 ilustra uma topologia *fat tree* em três níveis utilizada pelo PortLand. Para construir uma topologia em três níveis como esta, é preciso estabelecer o parâmetro k que define o número de portas em cada *switch* (k=4 neste exemplo). Em geral, a topologia em três níveis constituída de *switches* com k portas suporta comunicação não-bloqueante entre $k^3/4$ servidores utilizando $5k^2/4$ *switches* de k portas. A topologia como um todo é organizada em k conjuntos de servidores (chamados de *pods*), nos quais é possível prover comunicação não-bloqueante entre $k^2/4$ servidores através de técnicas de *hash* e distribuição do ECMP.

Do ponto de vista organizacional, a rede *fat tree* é relativamente fixa, possibilitando a construção e manutenção de *data centers* modulares, nos quais a filosofia de expansão é adicionar mais equipamentos (*racks* ou colunas de servidores) sob demanda. Obviamente, toda expansão requer uma fase anterior de planejamento na qual a estrutura de *switches* é definida de forma a suportar tais evoluções.

3.3.3.2. *Fabric Manager*

O PortLand utiliza um gerenciador de infraestrutura de rede centralizado (denominado *Fabric Manager*) para manter estados relacionados à configuração da rede, tal como a sua topologia. O *Fabric Manager* é um processo executado no espaço do usuário em uma máquina dedicada responsável pelo auxílio às requisições do ARP, tolerância a falhas e operações de *multicast*. De acordo com a especificação do PortLand, o *Fabric Manager* pode ser desenvolvido como um servidor conectado de forma redundante à estrutura do *data center* ou, ainda, ser executado em uma rede de controle separada.

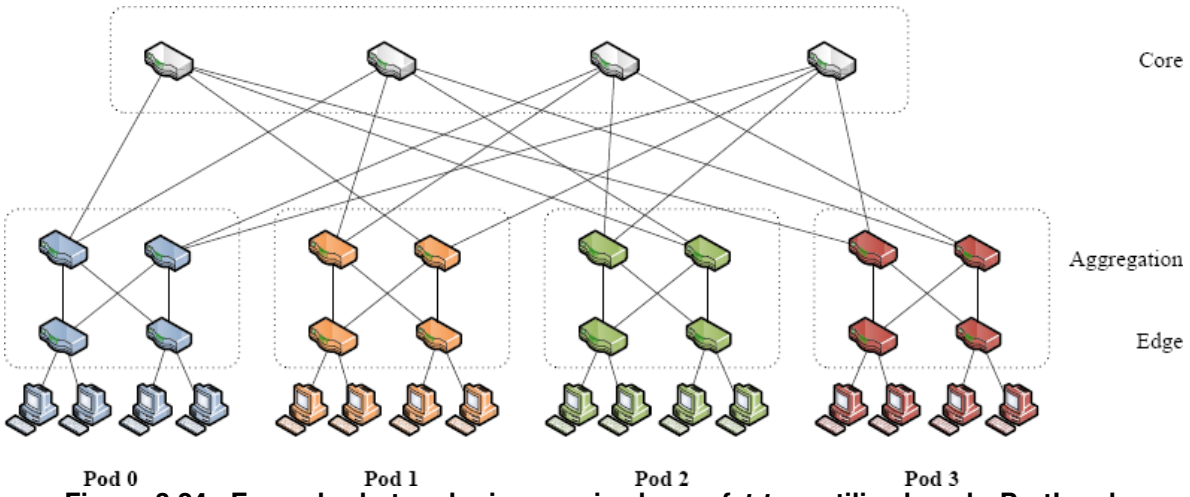


Figura 3.24. Exemplo de topologia organizada em *fat tree* utilizada pelo PortLand. Extraída de [Mysore et al. 2009].

3.3.3.3. Endereços PMAC (Pseudo MAC)

A base para um mecanismo de encaminhamento e roteamento eficiente, bem como suportar a migração de máquinas virtuais no Portland, vem da utilização dos endereços hierárquicos chamados Pseudo MAC (PMAC). O PortLand atribui um único PMAC para cada nó final, representando a localização de cada nó na topologia. Por exemplo, todos os nós finais localizados em um determinado *pod* compartilham um mesmo prefixo em seu PMAC. Entretanto, os nós finais permanecem inalterados, ou seja, eles acreditam ser identificados por seus endereços MAC atuais (AMAC - *Actual MAC*). As requisições de ARP feitas pelos nós finais são respondidas com o PMAC do nó de destino. Sendo assim, todo processo de encaminhamento de pacotes ocorre através da utilização dos PMAC. Os *switches* de egresso são responsáveis pelo mapeamento PMAC para AMAC e re-escrita dos pacotes para manter a ilusão de endereços MAC inalterados no nó de destino.

Os *switches* de borda (*Edge*) aprendem um único número de *pod* e uma única posição dentro deste *pod*. Para todos os nós diretamente conectados, os *switches* de borda atribuem um PMAC de 48 bits sob o formato *pod*.posição.porta.*vmid*, onde *pod* possui 16 bits e refere-se ao número do *pod* onde os nós estão localizados, posição possui 8 bits e indica a posição do *switch* dentro do *pod* e porta possui 8 bits para representar a porta na qual o nó final está ligado ao *switch*. O campo *vmid* possui 16 bits e é utilizado para multiplexar máquinas virtuais em uma mesma máquina física.

No instante em que um *switch* de ingresso observa a existência de um novo endereço MAC, os pacotes com este endereço são desviados para o plano de controle do *switch*, que

cria uma nova entrada na tabela de mapeamento e, na sequência, encaminha este novo mapeamento para o *Fabric Manager* para futuras resoluções, conforme ilustra a Figura 3.25.

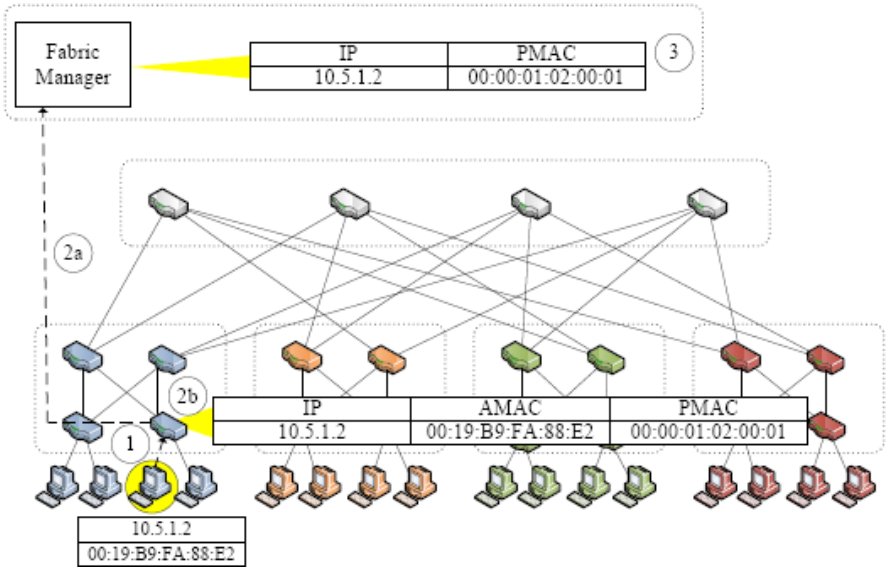


Figura 3.25. Mapeamento entre AMAC e PMAC. Extraída de [Mysore et al. 2009].

Basicamente, o PortLand efetua a separação entre localizador/identificador de forma totalmente transparente aos nós finais e compatível com o hardware dos *switches comoditizados* disponíveis no mercado. Outra característica importante do PortLand refere-se a não utilização de técnicas de tunelamento para encaminhar os pacotes, sendo apenas necessário a re-escrita de endereços PMAC/AMAC nas bordas do *data center*.

3.3.3.4. Mecanismo de *proxy* para requisições ARP

As requisições ARP, originalmente, efetuam *broadcast* e atingem todos os nós localizados em um mesmo domínio de camada 2. O PortLand utiliza o *Fabric Manager* para contornar o *overhead* de sinalização causado pelo ARP conforme ilustra a Figura 3.26. No passo 1, o *switch* de ingresso detecta a chegada de uma mensagem ARP requisitando um mapeamento IP para MAC, intercepta esta mensagem e a encaminha para o *Fabric Manager* no passo 2. O *Fabric Manager* consulta sua tabela de PMACs em busca do mapeamento e retorna o PMAC para o *switch* requisitante no passo 3. O *switch* de borda, por sua vez, cria uma mensagem de resposta do ARP e a retorna para o nó que originou a requisição no passo 4.

Finalmente, existe um detalhe adicional para prover suporte à migração de máquinas virtuais. Assim que a migração é completada, ou seja, a máquina virtual acaba sua transição entre um servidor físico e outro, a máquina virtual envia um ARP gratuito contendo seu novo mapeamento entre endereços IP e MAC. Este ARP gratuito é encaminhado até o *Fabric Manager* pelo *switch* de borda. Infelizmente, os nós que estavam comunicando com esta máquina virtual antes da migração manterão o mapeamento antigo em sua memória cache e terão de esperar até que o mapeamento expire para prosseguir com a comunicação. Entretanto, o *Fabric Manager* pode encaminhar uma mensagem de invalidação de mapeamento ao *switch* no qual a máquina virtual estava associada. Desta maneira, o *switch* seria capaz de replicar o ARP gratuito aos nós que continuam a originar pacotes na direção da máquina virtual que migrou, atualizando seus mapeamentos.

3.3.3.5. Protocolo de descoberta de posição na topologia

Os *switches* utilizam informações relativas às suas posições na topologia global do *data center* para efetuar encaminhamento e roteamento mais eficientes através da comunicação em pares, ou seja, encaminhamento considerando apenas os vizinhos diretamente conectados. Com o objetivo de criar um cenário *plug-and-play*, o PortLand propõe a utilização de um protocolo para descobrir a localização dos *switches* de forma automática.

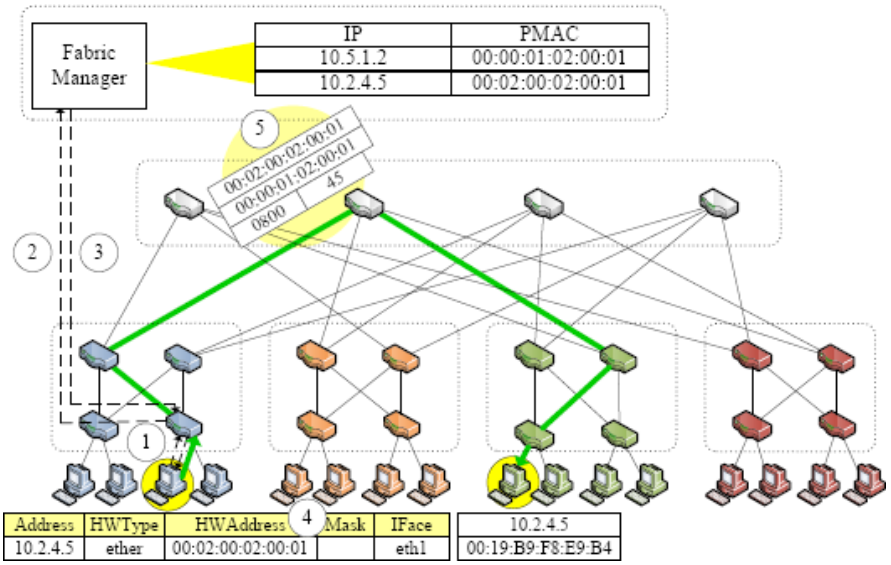


Figura 3.26. Resoluções ARP utilizando mecanismo de *Proxy*. Extraída de [Mysore et al. 2009].

Neste protocolo chamado LDP (*Location Discovery Protocol*), os *switches* enviam, periodicamente, LDMs (*Location Discovery Messages*) em todas as suas portas para: (1) definir suas posições e (2) monitorar o estado de suas conexões físicas. Em resumo, o LDP consegue definir quais são os *switches* de borda (*Edge*), uma vez que eles recebem LDMs em apenas uma fração de suas portas, as conectadas aos *switches* de agregação (*Aggregation*), pois os nós finais não geram LDMs.

A partir do momento que os *switches* de borda descobrem sua localização (nível) na topologia, as LDMs disseminadas subsequentemente passam a conter informação referente ao seu nível. Desta forma, o restante dos *switches* são capazes de aferir suas respectivas posições. Os *switches* de agregação aferem sua posição, uma vez que eles recebem LDMs em todas as suas portas, sendo algumas originadas pelos *switches* de borda (contendo informação de nível) e o restante originadas pelos *switches* de núcleo (sem informação de nível). Finalmente, os *switches* de núcleo (*core*) aferem sua posição, uma vez que todas as suas portas passarão a receber LDMs originadas por *switches* de agregação (contendo informação de nível). Uma vez definido o nível de todos os *switches*, o *Fabric Manager* é utilizado para atribuir o mesmo número de *pod* para os *switches* de borda pertencentes ao mesmo grupo.

3.3.4. BCube e MDCube

Uma nova maneira de construir e implantar *data centers* é através da sua modularização em contêineres, denominados *data centers* modulares (MDCs - *Modular data centers*). Em um MDC, alguns milhares de servidores são conectados utilizando pequenos *switches* para formar a infraestrutura do *data center* e, então, empacotados em contêineres padronizados de

20 ou 40 pés comumente utilizados para transportes mercantes. Desta forma, os *data centers* não são mais restritos a uma localização fixa, possibilitando que as organizações coloquem seus *data centers* em qualquer lugar e, também, possam realocá-los na medida em que seus requisitos mudam. Além da mobilidade, um MDC possui menor tempo de implantação, oferece um sistema com maior densidade de equipamentos e menores custos de resfriamento e produção. Entretanto, é difícil, ou mesmo impossível, efetuar manutenção no interior do contêiner uma vez que ele está implantado.

Esta seção apresenta o BCube [Guo et al. 2009], uma arquitetura robusta e de alto desempenho para a construção de MDCs e, também, apresenta o MDCube [Wu et al. 2009], uma estrutura para a construção de mega *data centers*. Neste cenário, cada contêiner desenvolvido de acordo com a arquitetura do BCube é uma peça dentro de um mega *data center* organizado segundo a estrutura do MDCube.

A arquitetura do BCube adota o modelo *server-centric*, ou seja, toda a inteligência necessária para o funcionamento de um MDC é inserida nos servidores, possibilitando a utilização de *switches comoditizados*. Cada servidor é equipado com um pequeno número de interfaces de rede (tipicamente não mais do que quatro) e múltiplas camadas de *switches* pequenos (poucas portas) são utilizadas para conectar todos os servidores.

A proposta do MDCube é considerar cada contêiner como um nó virtual dentro de um mega *data center*. Cada contêiner possui um conjunto de portas de alta velocidade que são utilizadas para conectá-lo a outros contêineres, criando uma malha de interconexão constituída por enlaces de fibra ótica de baixo custo. O MDCube ainda encontra-se em fase de desenvolvimento e, basicamente, estende a metodologia desenvolvida no BCube para um cenário de mega *data centers*. Os objetivos principais são: (1) prover elevada capacidade de transmissão entre contêineres; (2) diminuir os custos de infraestrutura e (3) simplificar a estrutura de cabeamento necessária.

3.3.4.1. Arquitetura BCube

Existem dois tipos de dispositivos no BCube: servidores com múltiplas interfaces de rede e *switches* que se conectam a um número (pequeno) constante de servidores. A estrutura do BCube é definida através de recursividade, um BCube_0 nada mais é que um conjunto de n servidores conectados a um *switch* de n portas. Um BCube_1 é constituído por n BCubes_0 e n *switches* de n portas. De forma genérica, um BCube_k ($k \geq 1$) é constituído de n BCubes_{k-1} e n^k *switches* de n portas. Cada servidor em um BCube_k possui $k + 1$ portas, as quais são numeradas a partir do nível 0 até o nível k. Com base nestas definições, é trivial perceber que um BCube_k possui $N = n^{k+1}$ servidores e $k + 1$ níveis de *switch*, onde cada nível possui n^k *switches* de n portas.

A Figura 3.27 apresenta um BCube1 com $n = 4$, constituído por quatro BCubes_0 e quatro *switches* de 4 portas. Todos os enlaces do BCube são bidirecionais e a construção garante que os *switches* apenas se conectam a servidores, ou seja, *switches* nunca se conectam a outros *switches* diretamente. A título de exemplificação, utilizando *switches comoditizados* de 8 portas, é possível construir um BCube_3 com 4096 servidores.

3.3.4.2. Mecanismo de Rota na Origem do BCube

O BCube requer um protocolo de roteamento capaz de utilizar de forma eficiente a diversidade de caminhos disponíveis na sua topologia, efetuando uma distribuição de carga ade-

quada. Sendo assim, é proposto o *BCube Source Routing* (BSR). Basicamente, no BSR o servidor de origem decide qual o caminho que o fluxo de pacotes deve atravessar através de mensagens de teste (*probing*) transmitidas na rede e, consequentemente, a rota na origem é inserida no cabeçalho dos pacotes. Duas razões foram consideradas para efetuar a opção pelo mecanismo de rota na origem: (1) o servidor de origem escolhe toda a rota para os pacotes sem a intervenção dos servidores intermediários e (2) os servidores intermediários apenas efetuam o encaminhamento dos pacotes com base na rota disponível nos cabeçalhos. Finalmente, a utilização de mensagens de teste (*probing*) cria um cenário no qual a topologia da rede é descoberta reativamente, evitando a utilização de protocolos baseados no estado do enlace, que sofrem de problemas de escalabilidade comumente associados ao *broadcast*.

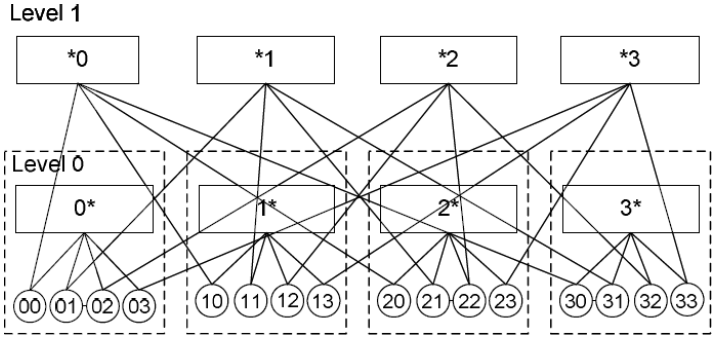


Figura 3.27. Exemplo de um BCube1 com *n* = 4. Extraída de [Guo et al. 2009].

Um fluxo é uma cadeia de pacotes identificados por cinco informações (origem, porta de origem, destino, porta de destino, protocolo). Quando um novo fluxo precisa ser transmitido, a origem envia mensagens de teste através de múltiplos caminhos e os servidores no caminho processam estas mensagens para verificar as necessidades do fluxo. Por exemplo, verificar se atendem as exigências referentes à largura de banda. Assim que as mensagens de teste atingem o destino, ele envia a mensagem de volta ao servidor de origem que, por sua vez, decide que caminho utilizar, baseando-se, por exemplo, na largura de banda disponível.

3.3.4.3. Roteando para redes externas no BCube

O BCube suporta comunicação externa com a Internet e, também, com outros BCubes, propondo a utilização de agregadores e *gateways*. Um agregador é um *switch comoditizado* de camada 2 com interfaces de 10 Gbps e os *gateways* são os servidores conectados nestes *switches* agregadores. Quando um servidor interno envia pacotes para um endereço IP externo, ele escolhe um dos *gateways* disponíveis. O pacote então é roteado para este *gateway* utilizando o BSR e, assim que o *gateway* recebe o pacote, ele remove o cabeçalho com a rota do BSR e encaminha o pacote para a rede externa através das interfaces de 10Gbps dos *switches* agregadores. A comunicação a partir das redes externas atingem o BCube através da utilização de endereços IPs dos servidores, externamente visíveis. Assim que os pacotes adentram o *data center*, os *gateways* são responsáveis pela definição da rota pela qual os pacotes atingirão os servidores.

3.3.4.4. BCube parcial

Em alguns casos, pode ser difícil ou desnecessário a construção de um BCube completo devido a restrições de espaço ou mesmo financeiras. Por exemplo, quando temos n=8 e k=3, é possível inserir 4096 servidores em um BCube_3. Para efetuar a construção de um BCube parcial, todos os BCubes_{k-1} são criados e, então, estes BCubes_{k-1} são conectados utilizando

uma camada k completa de *switches*. Sendo assim, o BSR é capaz de funcionar como se o BCube estivesse completo. A desvantagem desta abordagem é que os *switches* presentes na camada k (completa) não serão totalmente utilizados.

3.3.4.5. Arquitetura do MDCube

O MBCube é desenvolvido para conectar diversos BCubes através das interfaces de alta velocidade disponíveis nos *switches* de agregação destes elementos. Para suportar centenas de contêineres em um mega *data center* e prover uma vazão eficiente de dados, o MDCube utiliza fibra ótica na construção destes enlaces entre BCubes. Neste cenário, as interfaces de alta velocidade disponíveis nos diversos BCubes são vistas como uma única interface virtual. Por exemplo, um *switch comoditizado* que possua quatro interfaces de 10 Gbps é visto como sendo uma interface virtual com a capacidade de 40 Gbps. De acordo com a especificação do MDCube, um BCube é visto como um nó dentro da estrutura de um mega *data center* e o número de interfaces virtuais contidas neste nó é definido pelo número de *switches* de agregação contidos no BCube.

Para a construção de um MDCube $(D + 1)$ dimensional, temos $M = \Pi_{d=0}^{D} m_d$, onde m_d é o número de contêineres em uma dimensão d. Cada contêiner é identificado por um $cid = c_D c_{D-1} ... c_0 (c_d \in [0, m_d - 1], d \in [0,D])$. Cada contêiner armazena $\Sigma_{d=0}^{D}(m_d - 1)$ *switches*, sendo $(m_d - 1)$ o número de *switches* em uma dimensão d. Dentro de um MDCube, cada *switch* é identificado através do seu contêiner ID e seu *switch* ID dentro do BCube: cid, $bwid$, $cid \in [0, M - 1]$, $bwid \in [0, \Sigma_{d=0}^{D}(m_d - 1) - 1]$. Existe um enlace para cada par de contêineres em uma dimensão d. A construção de um MDCube de 2 dimensões é exemplificada na Figura 3.28 para um cenário formado por nove BCubes.

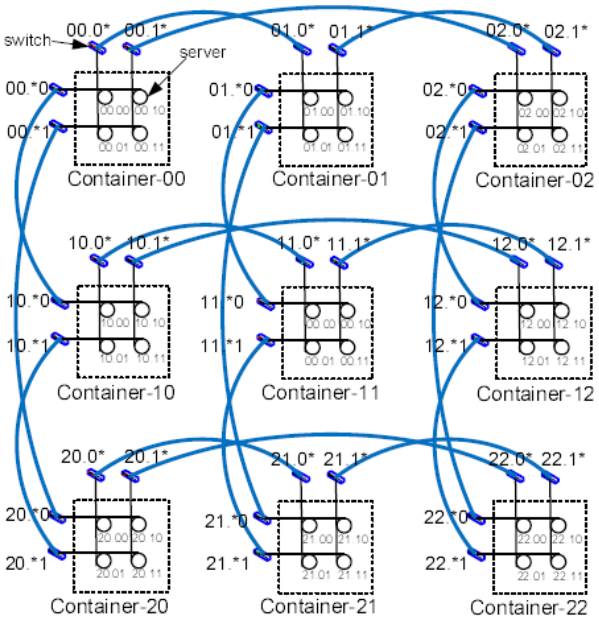


Figura 3.28. Exemplo de um MDCube de 2 dimensões formado a partir de nove (3x3) BCubes. Extraída de [Wu et al. 2009].

O modelo de comunicação do MDCube é *server-centric* como no BCube e a estrutura de roteamento é estendida para contemplar o mega *data center*. Sendo assim, cabe ao servidor de origem enviar as mensagens de teste através da estrutura do mega *data center* de tal forma a descobrir as rotas até o servidor de destino. O roteamento entre os diversos contêineres é baseado na correção das tuplas que identificam os contêineres em cada nó

atravessado pelos pacotes. Esta correção é controlada por uma permutação Π_D e uma nova função introduzida nos servidores possibilita o descobrimento dos enlaces existentes entre os diversos contêineres. A mesma abordagem do BCube para a comunicação externa é utilizada no MDCube. Entretanto, no MDCube as interfaces de alta velocidade dos contêineres são conectadas a roteadores responsáveis pela agregação de tráfego na entrada ou na saída do mega *data center*.

3.3.5. Resumo comparativo das abordagens

A Tabela 3.1 apresenta, de forma compilada, uma análise comparativa das propostas descritas nesta seção, considerando alguns dos requisitos levantados na Seção 3.2.

Tabela 3.1. Análise comparativa das novas arquiteturas de *data center*.

	Monsoon	VL2	Portland	BCube e MDCube
Realocação dinâmica de servidores (agilidade)	sim	sim	sim	sim
Transmissão à taxa máxima das interfaces/ *oversubscription*	sim (1Gbps) / 1:1	sim (1Gbps) / 1:1	sim (1Gbps) / 1:1	sim (1Gbps) / 1:1
Topologia	*fat tree*	*fat tree*	*fat tree*	hipercubo
Mecanismo de Roteamento/ Encaminhamento	tunelamento MAC-in-MAC	tunelamento IP-in-IP	baseado na posição hierárquica dos nós (PMAC)	rota na origem gerada por mensagens de *probing*
Balanceamento de Carga	VLB + ECMP	VLB + ECMP	Não especificado	trocas periódicas de *probing* modificam a rota
Modificação nos Nós Finais	sim	sim	não	sim
Modificação nos *Switches*	sim	não	sim	não
Serviço de diretório	sim	sim	sim	não

3.4. Tendências e conclusões

3.4.1. Tendências

Nesta seção destacamos de forma resumida uma série de tendências tecnológicas e áreas de pesquisa relevantes para o desenvolvimento de *data centers* e a evolução do modelo de computação em nuvem, não necessariamente restritas aos aspectos de arquitetura de rede.

- ***Software-Defined Networking* com OpenFlow:** o paradigma OpenFlow [McKeown et al. 2008] propõe uma generalização do plano de dados que

habilita a programabilidade e a virtualização dos recursos de rede, permitindo a separação entre o plano de dados e plano de controle. O OpenFlow permite que as decisões de manipulação de pacotes em alto-nível sejam realizadas por um controlador separado e centralizado fazendo com que o plano de dados possa ser implementado em *switches comoditizados*, sem possuir uma pilha de protocolos complexa.

Esta proposta tecnológica é especialmente interessante para as redes de *data centers*, de forma que os próprios provedores possam definir a funcionalidade da rede, com um baixo custo do equipamento e, especialmente customizado para as necessidades de escala e controle das características de *data centers* para computação em nuvem. Portland é um dos exemplos apresentados neste minicurso que usa a tecnologia OpenFlow. Espera-se que novas propostas (incluindo também os grandes provedores como Microsoft e Google) adotem a tecnologia OpenFlow (já) para as redes de *clusters* no *data center* [Tavakoli et al. 2009].

Basicamente, um *switch* OpenFlow separa o encaminhamento rápido de pacotes (plano de dados) do nível de decisões de encaminhamento (plano de controle) em um roteador ou *switch*. Embora parte do plano de dados ainda resida no *switch* e execute sobre o mesmo hardware (portas lógicas, memória), as decisões de manipulação de pacotes em alto-nível são movidas para um controlador separado (por exemplo, NOX [Gude et al. 2008]). A Figura 3.29 mostra uma abstração de uma rede com OpenFlow e um controlador NOX.

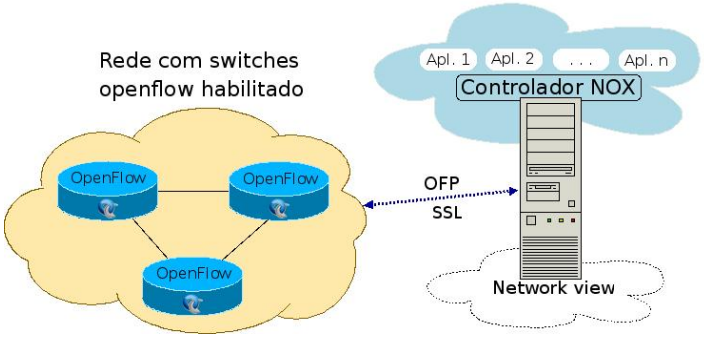


Figura 3.29. Uma rede com OpenFlow e controlador NOX.

- **Arquiteturas baseadas em memória:** com o objetivo de superar o tempo de acesso lento de discos e aumentar o desempenho das aplicações em nuvem, muitos esforços adicionam uma camada de cache e tentam manter o banco de dados e todas as aplicações em sintonia com o cache. A pergunta natural neste sentido é se realmente essa nova camada extra de cache é necessária quando os dados poderiam muito bem ser mantidos na memória desde o início. Com a proliferação de aplicações com requisitos de "tempo real" e sistemas que requerem escalabilidade massiva, os avanços em hardware e software sugerem que a memória volátil pode se tornar o novo disco rígido e os discos rígidos serem relegados às tarefas de *back-up* (tipo fitas).

  Este modelo já foi discutido na comunidade das grades computacionais, onde o avanço em hardware e redes em torno do acesso à memória rápida têm permitido a criação de *clusters* com melhor desempenho do que *clusters* baseados em discos. A memória volátil (por exemplo, RAM) é várias ordens de grandeza mais rápida do que um acesso aleatório em disco.

A fim de preencher a enorme lacuna entre a latência de memória RAM e os discos, novos sistemas de armazenamento (*flash*) SSDs (*Solid-State Drive*), estão sendo introduzidos pelos vendedores. Com a tecnologia *flash* continuando a tendência de redução de preço, esta pode até mesmo substituir as unidades tradicionais para algumas aplicações. Vários projetos já estão trabalhando nessa linha (com ênfase em sistemas de baixo consumo), tais como FAWN [Andersen et al. 2009], Gordon [Caulfield et al. 2009], ou o projeto RAMCloud [Ousterhout et al. 2009] da Universidade de Stanford.

- **Bases de dados NoSQL:** ao longo dos últimos anos, temos assistido o surgimento de sistemas de armazenamento de dados que diferem de forma bastante significativa em relação ao modelo de RDBMS. Estes sistemas são também conhecidos como NoSQL [Leavitt 2010] e entre os exemplos mais notáveis se encontram BigTable da Google, o Dynamo da Amazon, Cassendra, CouchDB, MongoDB, etc.

  Estas soluções têm um número de características em comum, tais como armazenamento de pares chave-valor, a execução em um grande número de máquinas *comoditizadas*, a divisão e replicação dos dados entre as máquinas e o relaxamento do requisito de consistência dos dados. O modelo de dados subjacente pode ser considerado como uma grande tabela *Hash* onde a forma básica de acesso à API é tipicamente da forma *Get (key), Put (key, value), Delete (key)*. Embora os sistemas NoSQL não substituirão os sistemas RDBMS tradicionais, não há dúvida que eles terão um papel importante em muitas das aplicações em nuvem. Nesse sentido, ainda são muitos os desafios que devem ser resolvidos como, por exemplo, o aperfeiçoamento de mecanismos para garantir a confiabilidade, a consistência dos dados e a resiliência a falhas. Vale a pena notar que há interessantes oportunidades de pesquisa na combinação de sistemas NoSQL e as arquiteturas baseadas em memória, pois ambas formam uma parceria natural para os requisitos das aplicações de grande escala nos *data centers* da nuvem.

- **Padrões para computação em nuvem:** a computação em nuvem se insere na Internet como uma nova camada de sistemas distribuídos com APIs fornecidas por múltiplos provedores. Como toda nova camada, oferece vantagens pela abstração dos recursos e oportunidades de inovação. Porém, também requer trabalhos em conjunto para garantir a interoperabilidade das soluções e permitir uma evolução saudável do sistema global como um todo. Vint Cerf, um dos pioneiros da Internet original, tem advertido [Cerf 2010] sobre essa necessidade e aponta os passos que a indústria deve dar para atingir a meta de uma interoperabilidade global na nuvem. Na mesma direção, o *Open Cloud* Manifesto [Open Cloud Manifesto 2010] é uma declaração de princípios para a manutenção da computação em nuvem como um sistema aberto, atualmente apoiada por mais de 250 organizações. Há ainda a iniciativa de padrões para computação em nuvem [Cloud Standards 2010] que atua em conjunto com o *Open Cloud* Manifesto na busca pela definição de padrões para este novo modelo de interação.

- **Interação entre nuvens (*Inter-Cloud*):** até o momento, temos focado nossa discussão nos *data centers* operados de forma isolada, sem entrar em detalhes de cenários (1) onde múltiplos *data centers* operam de forma unificada (por exemplo nos modelos de micro- e nano-*data centers*); (2) onde múltiplos provedores de serviços em nuvem estabelecem acordos para se federar ou (3) onde o cliente (corporativo) dos serviços em

nuvem tem relacionamento com múltiplos provedores (conhecido como nuvem *multi-homing*). Os trabalhos de padronização discutidos anteriormente são um pré-requisito para os cenários 2 e 3. Para tornar estes cenários realidade, ainda há muitos outros desafios e oportunidades para serem abordados, quando analisamos o que pode se chamar de *Inter-Cloud*.

Para começar, podemos imaginar que na comunicação entre nuvens deverá haver um suporte natural do movimento massivo de cargas de trabalho na forma de máquinas virtuais migrando junto a um grande volume de dados. Esta migração pode ocorrer, por exemplo, quando houver a necessidade de aumento de escalabilidade de serviços populares sob-demanda ou a oferta de um provedor oferecendo descontos de recursos computacionais, fazendo com que esta migração entre nuvens seja economicamente rentável. Nestes casos de uso, a comunicação entre nuvens deve garantir segurança dos dados nas múltiplas dimensões conhecidas (integridade, confiabilidade, etc.), transparência nos sistemas de gerência e flexibilidade nos sistemas de endereçamento e roteamento IP/Ethernet de forma que a migração aconteça de modo natural, tanto nas comunicações com as redes virtuais privadas estabelecidas quanto nas comunicações com usuários da Internet pública. A criação de nuvens privadas virtuais [Wood et al. 2009] (fatias de redes virtuais) pode ser realizada como uma rede *overlay* sobre IP ou como uma rede *underlay* (por exemplo mediante o estabelecimento de circuitos óticos multi-domínios) garantindo não só transparência nos protocolos de endereçamento mas também uma grande capacidade de transferência de dados[12].

Finalmente, podemos especular que o desenvolvimento da *Inter-Cloud* pode trazer ainda mais mudanças na infraestrutura central (*core*) da Internet, com novos acordos de *peering* entre os provedores de serviços em nuvem e o surgimento da denominada "*dark Internet*". Também se espera o surgimento de um mercado de recursos na nuvem operado por agregadores e *brokers* (negociadores) para serviços em nuvem (por exemplo, Transit Portal [Valancius et al. 2009]). Serviços de conectividade flexível (multi-homing, balanceamento de carga) e avançada (segurança, QoS) com os provedores de serviços em nuvem podem trazer novos incentivos para a adoção de protocolos, tais como as versões seguras de BGP, DNS, IP *multicast*, MPLS, IPv6 e propostas baseadas na separação identificador/localizador como o LISP.

Em uma evolução extrema da *Inter-Cloud*, pode emergir o conceito de *Ambient Cloud* [Hoff 2009a], motivado pelos custos das mega-infraestruturas e as demandas de aplicações, onde os recursos de qualquer dispositivo (*set-top-box*, PDAs, celulares, PCs) podem ser utilizados em uma escala global.

3.4.2. Conclusões

O modelo convencional de rede IP/Ethernet não atende os requisitos de custo, escala e controle dos provedores de computação em nuvem. É por este motivo e pelas características especiais das redes dos *data centers* que novos projetos e propostas têm emergido para atender os objetivos específicos dos *cloud data centers*, que são criticamente diferentes dos *data centers* tradicionais e das redes locais e metropolitanas dos provedores de serviços.

[12]Estima-se que o volume de tráfego resultante desta migração de máquinas virtuais pode alcançar ordens de magnitude comparáveis à quantidade de vídeo sobre Internet atualmente.

Tratando os *data centers* como um sistema e fazendo uma customização e otimização completa, as novas propostas de arquiteturas de rede prometem atingir uma redução dos custos operacionais e de capital, uma maior confiabilidade, um modelo de escala sob-demanda sustentável e uma maior capacidade de inovação.

Não se pode negar que o modelo de computação em nuvem é evolucionário pois surge de uma construção histórica, baseada na forma como a própria Internet surgiu e cresceu. As demandas por novos serviços e o barateamento de recursos computacionais fizeram com que grandes empresas como Microsoft e Google, que já possuíam um grande patrimônio computacional instalado para atender suas próprias necessidades, percebessem que vender ou alugar tais recursos computacionais poderia ser um negócio rentável. Sendo assim, estas empresas são as que lideram este modelo e continuam investindo cada vez mais para aumentar a disponibilidade de serviços em nuvem. Entretanto, pequenas empresas podem se inserir no mercado como provedores de serviços utilizando uma infraestrutura (IaaS) alugada através do modelo *pay-as-you-go*.

O modelo de computação em nuvem tem atraído vários setores incluindo empresas provedoras de serviços Web, provedores de conectividade, indústria, setor bancário e mais recentemente governos federais de alguns países. Um exemplo disso é o Governo Federal Americano que tem investido no desenvolvimento de soluções em nuvem, utilizando plataformas de código livre (*open source*) para atender às suas próprias necessidades [NASA 2010]. Espera-se que a comunidade brasileira apoiada por órgãos de pesquisa e instituições federais e privadas comece a avaliar as soluções baseadas em serviços em nuvem como uma forma de reduzir custos de TI, aumentar a oferta de serviços e se inserir em um modelo global, inovador e emergente.

Referências

[Al-Fares et al. 2008] Al-Fares, M., Loukissas, A., and Vahdat, A. (2008). A Scalable Commodity Data Center Network Architecture. *SIGCOMM Comput. Commun. Rev.*, 38(4):63–74.

[Amazon 2010a] Amazon (2010a). Amazon Elastic Compute Cloud. Disponível online em http://aws.amazon.com/ec2/.

[Amazon 2010b] Amazon (2010b). Amazon Web Services. Disponível online em http://aws.amazon.com/.

[Andersen et al. 2009] Andersen, D. G., Franklin, J., Kaminsky, M., Phanishayee, A., Tan, L., and Vasudevan, V. (2009). FAWN: A Fast Array of Wimpy Nodes. In *SOSP '09: Proceedings of the ACM SIGOPS 22nd symposium on Operating systems principles*, pages 1–14, New York, NY, USA. ACM.

[Armbrust et al. 2009] Armbrust, M., Fox, A., Griffith, R., Joseph, A. D., Katz, R. H., Konwinski, A., Lee, G., Patterson, D. A., Rabkin, A., Stoica, I., and Zaharia, M. (2009). Above the Clouds: A Berkeley View of Cloud Computing. Technical Report UCB/EECS-2009-28, EECS Department, University of California, Berkeley.

[Barroso and Hölzle 2007] Barroso, L. A. and Hölzle, U. (2007). The Case for Energy-Proportional Computing. *Computer*, 40(12):33–37.

[Benson et al. 2009a] Benson, T., Anand, A., Akella, A., and Zhang, M. (2009a). Understanding Data Center Traffic Characteristics. In *WREN '09: Proceedings of the 1st ACM workshop on Research on enterprise networking*, pages 65–72, New York, NY, USA. ACM.

[Benson et al. 2009b] Benson, T. A., Akella, A., and Zhang, M. (2009b). The Case for Fine-Grained Traffic Engineering in Data Centers. Technical Report 1666, University of WisconsinMadison. Disponível online em http://www.cs.wisc.edu/techreports/2009/TR1666.pdf.

[Brandon 2009] Brandon, H. e. a. (2009). ElasticTree: Saving Energy in Data Center Networks. Technical report. Disponível online em http://www.openflowswitch.org/wk/images/2/2c/ElasticTree-TechReport2009.pdf.

[Caulfield et al. 2009] Caulfield, A. M., Grupp, L. M., and Swanson, S. (2009). Gordon: Using Flash Memory to Build Fast, Power-efficient Clusters for Data-intensive Applications. In *ASPLOS '09: Proceeding of the 14th international conference on Architectural support for programming languages and operating systems*, pages 217–228, New York, NY, USA. ACM.

[Cerf 2010] Cerf, V. (2010). Vint Cerf e computação em nuvem. Disponível online em http://news.techworld.com/virtualisation/3209948/vint-cerf-calls-for-cloud-computing-standards/.

[Christian Belady (ed) 2007] Christian Belady (ed) (2007). The Green Grid Data Center Power Efficiency Metrics: PUE and DCiE. Technical report.

[Cisco 2007] Cisco (2007). Cisco Data Center Infrastructure 2.5 Design Guide. Disponível online em http://www.cisco.com/en/US/docs/solutions/Enterprise/Data_Center/DC_Infra2_5/DCI_SRND_2_5_book.html.

[Cloud Standards 2010] Cloud Standards (2010). Padrões para Computação em Nuvem. Disponível online em http://cloud-standards.org.

[Enterprise Control Systems 2010] Enterprise Control Systems (2010). Vendor Neutral Modular and Container Based Data Centers. Disponível online em http://www.datacenterexperts.com/containerbaseddatacenters.html.

[Forrester 2010] Forrester (2010). Should Your Email Live In The Cloud? A Comparative Cost Analysis. Disponível online em http://www.google.com/a/help/intl/en/admins/pdf/forrester_cloud_email_cost_analysis.pdf.

[Foster 2010] Foster, I. (2010). Blog do Ian Foster. Disponível online em http://ianfoster.typepad.com/blog/.

[Google 2010a] Google (2010a). BigTable. Disponível online em http://labs.google.com/papers/bigtable.html.

[Google 2010b] Google (2010b). Google Apps. Disponível online em http://www.google.com/apps/.

[Google 2010c] Google (2010c). Google Web Toolkit. Disponível online em http://code.google.com/intl/pt-BR/webtoolkit/.

[Google 2010d] Google (2010d). Medidas de PUE da Google. Disponível online em http://www.google.com/corporate/green/datacenters/measuring.html.

[Greenberg 2009] Greenberg, A. (2009). Networking The Cloud. ICDCS 2009 keynote. Disponível online em http://www.cse.ohio-state.edu/icdcs2009/Keynote_files/greenbergkeynote.pdf.

[Greenberg et al. 2009a] Greenberg, A., Hamilton, J., Maltz, D. A., and Patel, P. (2009a). The Cost of a Cloud: Research Problems in Data Center Networks. *SIGCOMM Comput. Commun. Rev.*, 39(1):68–73.

[Greenberg et al. 2009b] Greenberg, A., Hamilton, J. R., Jain, N., Kandula, S., Kim, C., Lahiri, P., Maltz, D. A., Patel, P., and Sengupta, S. (2009b). VL2: A Scalable and Flexible Data Center Network. *In Proceedings of the ACM SIGCOMM 2009 Conference on Data Communication, Barcelona, Spain*.

[Greenberg et al. 2008] Greenberg, A., Lahiri, P., Maltz, D. A., Patel, P., and Sengupta, S. (2008). Towards a Next Generation Data Center Architecture: Scalability and Commoditization. *In Proceedings of the ACM Workshop on Programmable Routers For Extensible Services of Tomorrow, Seattle, WA, USA*.

[Greenberg et al. 2006] Greenberg, S., Mills, E., Tschudi, B., Rumsey, P., and Myatt (2006). Best Practices for Data Centers: Results from Benchmarking 22 Data Centers. In *Proceedings of the 2006 ACEEE Summer Study on Energy Efficiency in Buildings.*

[Gude et al. 2008] Gude, N., Koponen, T., Pettit, J., Pfaff, B., Casado, M., McKeown, N., and Shenker, S. (2008). NOX: Towards an Operating System for Networks. *SIGCOMM Comput. Commun. Rev.*, 38(3):105–110.

[Guo et al. 2009] Guo, C., Lu, G., Li, D., Wu, H., Zhang, X., Shi, Y., Tian, C., Zhang, Y., and Lu, S. (2009). BCube: A High Performance, Server-centric Network Architecture for Modular Data Centers. *In Proceedings of the ACM SIGCOMM 2009 Conference on Data Communication, Barcelona, Spain*.

[Guo et al. 2008] Guo, C., Wu, H., Tan, K., Shi, L., Zhang, Y., and Lu, S. (2008). Dcell: A Scalable and Fault-tolerant Network Structure for Data Centers. In *SIGCOMM '08: Proceedings of the ACM SIGCOMM 2008 conference on Data communication*, pages 75–86, New York, NY, USA. ACM.

[Hamilton 2008] Hamilton, J. (2008). Diseconomies of Scale. Blog post, Disponível online em http://perspectives.mvdirona.com/2008/11/28/CostOfPowerInLargeScaleDataCenters.aspx.

[Hamilton 2009a] Hamilton, J. (2009a). Data Center Networks Are in My Way. *Stanford Clean Slate CTO Summit*.

[Hamilton 2009b] Hamilton, J. (2009b). PUE and Total Power Usage Efficiency (tPUE). Blog post, Disponível online em http://perspectives.mvdirona.com/2009/06/15/PUEAndTotalPowerUsageEfficiencyTPUE.aspx.

[Hamilton 2010a] Hamilton, J. (2010a). Perspectives. Disponível online em http://perspectives.mvdirona.com.

[Hamilton 2010b] Hamilton, J. (2010b). Scaling at MySpace. Blog post, Disponível online em http://perspectives.mvdirona.com/2010/02/15/ScalingAtMySpace.aspx.

[Hoelzle and Barroso 2009] Hoelzle, U. and Barroso, L. A. (2009). *The Datacenter as a Computer: An Introduction to the Design of Warehouse-Scale Machines*. Morgan and Claypool Publishers.

[Hoff 2009a] Hoff, T. (2009a). Building Super Scalable Systems: Blade Runner Meets Autonomic Computing in the Ambient Cloud. Disponível online em http://highscalability.com/blog/2009/12/16/buildingsuperscalablesystemsbladerunnermeetsautonomic.html.

[Hoff 2009b] Hoff, T. (2009b). Why are Facebook, Digg, and Twitter so Hard to Scale? Blog post, Disponível online em http://highscalability.com/blog/2009/10/13/why-are-facebook-digg-and-twitter-so-hard-to-scale.html.

[Johnston 2009] Johnston, S. (2009). Introduciong the Cloud Computing Stack. Disponível online em http://samj.net/2009/04/introducing-cloud-computing-stack-2009.html.

[Kandula et al. 2009] Kandula, S., Padhye, J., and Bahl, P. (2009). Flyways To De-Congest Data Center Networks. In *Proc. of workshop on Hot Topics in Networks (HotNets-VIII)*.

[Leavitt 2010] Leavitt, N. (2010). Will NoSQL Databases Live Up to Their Promise? *Computer*, 43:12–14.

[Leighton] Leighton, F. T. *Introduction to Parallel Algorithms and Architectures: Arrays, Trees, Hypercubes*. Morgan Kaufmann Publishers, 1 edition.

[Lim et al. 2008] Lim, K., Ranganathan, P., Chang, J., Patel, C., Mudge, T., and Reinhardt, S. (2008). Understanding and Designing New Server Architectures for Emerging Warehouse-Computing Environments. In *ISCA '08: Proceedings of the 35th International Symposium on Computer Architecture*, pages 315–326, Washington, DC, USA. IEEE Computer Society.

[McKeown et al. 2008] McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., Shenker, S., and Turner, J. (2008). OpenFlow: Enabling Innovation in Campus Networks. *SIGCOMM Comput. Commun. Rev.*, 38(2):69–74.

[Mell and Grance 2009] Mell, P. and Grance, T. (2009). The NIST Definition of Cloud Computing. *National Institute of Standards and Technology, Information Technology Laboratory*.

[Microsoft 2010] Microsoft (2010). Windows Azure Platform. Disponível online em http://www.microsoft.com/windowsazure/.

[Mysore et al. 2009] Mysore, R. N., Pamboris, A., Farrington, N., Huang, N., Miri, P., Radhakrishnan, S., Subramanya, V., and Vahdat, A. (2009). PortLand: A Scalable Fault-Tolerant Layer 2 Data Center Network Fabric. *In Proceedings of the ACM SIGCOMM 2009 Conference on Data Communication, Barcelona, Spain*.

[Nanodatacenters 2010] Nanodatacenters (2010). Disponível online em http://www.nanodatacenters.eu.

[NASA 2010] NASA (2010). Nebula. Disponível online em http://www.datacenterknowledge.com/archives/2009/12/02/nasas-nebula-the-cloud-in-a-container/.

[NCSA 2010] NCSA (2010). National Center for Supercomputing Applications. Disponível online em http://www.ncsa.illinois.edu/.

[No-SQL 2010] No-SQL (2010). Disponível online em http://nosql.net.

[Open Cloud Manifesto 2010] Open Cloud Manifesto (2010). Disponível online em http://opencloudmanifesto.org.

[Ousterhout et al. 2009] Ousterhout, J., Agrawal, P., Erickson, D., Kozyrakis, C., Leverich, J., Mazières, D., Mitra, S., Narayanan, A., Parulkar, G., Rosenblum, M., Rumble, S. M., Stratmann, E., and Stutsman, R. (2009). The case for RAMClouds: Scalable High-performance Storage Entirely in DRAM. *SIGOPS Oper. Syst. Rev.*, 43(4):92–105.

[Pujol et al. 2009] Pujol, J. M., Siganos, G., Erramilli, V., and Rodriguez, P. (2009). Scaling Online Social Networks without Pains. NetDB 2009, 5th International Workshop on Networking Meets Databases, co-located with SOSP 2009.

[Qureshi et al. 2009] Qureshi, A., Weber, R., Balakrishnan, H., Guttag, J., and Maggs, B. (2009). Cutting the Electric Bill for Internet-scale Systems. In *SIGCOMM '09: Proceedings of the ACM SIGCOMM 2009 conference on Data communication*, pages 123–134, New York, NY, USA. ACM.

[S. Kandula and Patel 2009] S. Kandula, Sudipta Sengupta, A. G. and Patel, P. (2009). The Nature of Data Center Traffic: Measurements and Analysis. In *ACM SIGCOMM Internet Measurement Conference (IMC)*.

[Sonnek and Chandra 2009] Sonnek, J. and Chandra, A. (2009). Virtual Putty: Reshaping the Physical Footprint of Virtual Machines. In *Proc. of Workshop on Hot Topics in Cloud Computing (HotCloud'09)*.

[Tavakoli et al. 2009] Tavakoli, A., Casado, M., Koponen, T., and Shenker, S. (2009). Applying NOX to the Datacenter. In *Proc. of workshop on Hot Topics in Networks (HotNets-VIII)*.

[Valancius et al. 2009] Valancius, V., Mundada, Y., Feamster, N., Rexford, J., and Nakao, A. (2009). Transit Portal: Bringing Connectivity to The Cloud. SIGCOMM 2009 Poster/Demo Session.

[Vaquero et al. 2009] Vaquero, L. M., Rodero-Merino, L., Caceres, J., and Lindner, M. (2009). A Break in the Clouds: Towards a Cloud Definition. *SIGCOMM Comput. Commun. Rev.*, 39(1):50–55.

[Wang et al. 2009] Wang, G., Andersen, D. G., Kaminsky, M., Kozuch, M., Ng, T. S. E., Papagiannaki, K., Glick, M., and Mummert, L. (2009). Your Data Center Is a Router: The Case for Reconfigurable Optical Circuit Switched Paths. In *Proc. of workshop on Hot Topics in Networks (HotNets-VIII)*.

[Wischik et al. 2008] Wischik, D., Handley, M., and Braun, M. B. (2008). The resource pooling principle. *SIGCOMM Comput. Commun. Rev.*, 38(5):47–52.

[Wood et al. 2009] Wood, T., Alexander, Ramakrishnan, K., Shenoy, P., and Merwe, J. V. (2009). The Case for EnterpriseReady Virtual Private Clouds. In *Proc. of Workshop on Hot Topics in Cloud Computing (HotCloud'09)*.

[Wu et al. 2009] Wu, H., Lu, G., Li, D., Guo, C., and Zhang, Y. (2009). MDCube: A High Performance Network Structure for Modular Data Center Interconnection. *In Proceedings of the ACM CONEXT 2009, Rome, Italy*.

[Yuan et al. 2007] Yuan, X., Nienaber, W., Duan, Z., and Melhem, R. (2007). Oblivious Routing for Fat-tree Based System Area Networks with Uncertain Traffic Demands. *SIGMETRICS Perform. Eval. Rev.*, 35(1):337–348.