Redes de Data Center com filtro de Bloom nos pacotes

Carlos A. B. Macapuna , Christian Esteve Rothenberg , Maurício F. Magalhães (Orientador)

Departamento de Engenharia de Computação e Automação Industrial (DCA)
Faculdade de Engenharia Elétrica e de Computação (FEEC)
Universidade Estadual de Campinas (Unicamp)

{macapuna, chesteve, mauricio}@dca.fee.unicamp.br

Abstract – This paper describes a networking approach for cloud data centers architectures based on a novel use of in-packet Bloom filters to encode randomized network paths. We present the design principles and testbed implementation of a data center architecture governed by Rack Managers, which are responsible to transparently provide the networking and support functions to cost-efficiently operate the DC network. We evaluate the proposal in terms of state requirements, our claims of false-positive-free forwarding, and the load balancing capabilities.

1. Introdução

Com o advento de serviços em nuvem para Internet, o suporte às redes de *data center* (DCN - *data center networks*) tornou-se um assunto de intensa pesquisa para aumentar sua escala, desempenho e relação custo-eficiência [3]. A fim de cumprir essas metas, sem comprometer a qualidade do serviço, inovações são solicitadas em muitas áreas do ambiente de *data center*, incluindo a própria infraestrutura de hospedagem (por exemplo, eficiência energética, fiação, acondicionamento) e a engenharia de rede (roteamento, virtualização, monitoramento).

Pesquisas recentes em re-arquitetura de *data center* têm estimulado projetos inovadores para interligar servidores, incluindo encaminhamento baseado na posição de pseudo endereço MAC [7], topologias *fat-tree* [1], ou balanceamento de carga usando uma camada 2 virtual [2].

Neste trabalho, descrevemos o encaminhamento utilizando filtro de Bloom nos pacotes, uma proposta DCN motivada por mudanças na rede e impulsionado pelo baixo custo dos *switches* com um substrato de programabilidade (OpenFlow [6]). Nosso projeto empresta algumas características de uma nova geração de DCN, por exemplo, a incorporação de controladores logicamente centralizados (4D [4]).

Basicamente, a ideia é interligar qualquer par de nós conectados dentro da DCN através de codificação da rota na origem em um filtro de Bloom, adicionado nos campos MAC Ethernet. Os objetivos do projeto incluem a conservação da semântica IP e eliminação de falsos-positivos explorando os múltiplos caminhos disponíveis. A solução proposta permite uma melhor utilização do espaço de 96-bits de origem e destino dos campos MAC, evitando, assim, o encapsulamento e, ao mesmo tempo, conserva a boa propriedade de *plug and play* do endereçamento Ethernet.

O resto do artigo está organizado da seguinte forma: a Seção 2 introduz informações relacionadas à lógica sobre a arquitetura DCN; a Seção 3 apresenta os princípios de projeto adotados para a nossa solução e descreve os principais blocos funcionais; na Seção 4 detalhamos a implementação do protótipo e do ambiente de testes; a Seção 5 avalia a proposta em termos de requisitos e estado de rede, falsos positivos, e capacidades de balanceamento de carga e, finalmente, a Seção 5 conclui o artigo.

2. Arquitetura do Data Center

O *data center*, como uma rede de interconexão para realizar tarefas de processamento distribuído, tem três principais elementos dominantes que determinam o seu desempenho: (1) a arquitetura de rede, (2) o esquema de roteamento, e (3) a topologia de interconexão. Nesta seção, descrevemos a estratégia adotada para resolver (1) e (2) e que pode ser resumida como uma abordagem separação identificador/localizador, onde os endereços IP atuam apenas como identificadores, e o roteamento é fornecido pelo encaminhamento baseado em filtro de Bloom nos pacotes. Quanto a (3), assumimos uma topologia em 3 camadas, uma inferior de *switches* ToR (*Top Of Rack*), uma camada intermediária de *switches* de agregação (AGGR), e uma camada superior de *switches* CORE (ver Fig. 1).

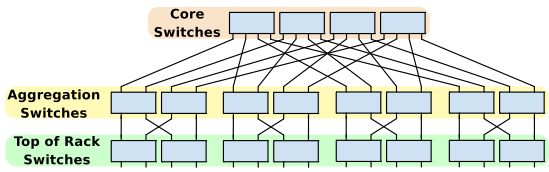

Figura 1. Arquitetura *fat tree* de 3 camadas.

2.1. Princípios de projeto

Com base na proposta de arquitetura de *data center*, nossos princípios de projeto são:

Separação identificador/localizador: A divisão entre identificador e localizador possui um papel fundamental para permitir o compartilhamento de recursos dos serviços com endereçamento IP. O IP é usado para identificar servidores físicos (e virtuais) dentro do DC, ou seja, não são impostas restrições de como os endereços são atribuídos ou utilizados para acesso externo (Internet), tornando-os não significativos para o roteamento de pacotes.

Rota na origem: Aproveitando o pequeno diâmetro das topologias de redes de *data center*, nossa abordagem utiliza o esquema de rota na origem (*strict source routing*). O roteamento nas topologias DCN em 3 camadas é bastante simplificado, ou seja, qualquer rota entre dois ToRs, tem uma trajetória ascendente em direção a um *switch* CORE e, em seguida, uma trajetória na direção do ToR destino, ambas passam pelos *switches* intermediários (AGGR). O encaminhamento é baseado em filtro de Bloom nos pacotes (iBF - *in-packet Bloom filter*) contendo apenas três elementos, nomeados de identificadores *Bloomed MAC* ($\langle CORE_i, AGGR_{down}, ToR_{dst} \rangle$) de *switches*. O ToR de origem calcula as rotas e envia para o AGGR mais próximo, com isso, três decisões são realizadas utilizando o iBF.

Diretório centralizado e controle de rede direto: Utilizamos a filosofia 4D [4] que simplifica o plano de dados e centraliza o plano de controle. Introduzimos o Gerenciador de *Rack* (RM) para tomar as decisões de roteamento e inserir as entradas dos fluxos nos *switches* programáveis.

Balanceamento de carga: A abordagem para fornecer o balanceamento de cargas é baseada no encaminhamento aleatório (*oblivious routing*). O RM é responsável pela seleção aleatória dos caminhos até o ToR destino.

2.2. Encaminhamento com identificadores *Bloomed MAC*

A originalidade, no caso, está associada à tomada de decisão nos *switches* AGRR e CORE. Inicialmente, as tabelas de encaminhamento estão vazias. Após a descoberta da topologia, as tabelas são preenchidas com uma entrada de fluxo para cada *switch* vizinho detectado. No lugar da tradicional combinação exata dos campos MAC, cada entrada de fluxo contém uma máscara de 96-bits gerada a partir de k funções de espalhamento (*hashing*) sobre o endereço MAC interno do *switch* vizinho. Um *ID Bloomed MAC* é um vetor de 96-bits onde apenas k bits são definidos com o valor igual a 1, como resultado da aplicação das k funções de *hashing*. Na chegada do pacote, apenas os 1s de cada *ID Bloomed MAC* são verificados quanto à presença nos campos MAC Ethernet (origem e destino) utilizados para transportar a rota na origem codificada no iBF. No caso de todos os 1s do *ID Bloomed MAC* corresponderem aos 1s definidos no iBF, o pacote é encaminhado para a interface correspondente.

2.3. Protocolo de descoberta

Uma questão não trivial é a descoberta da topologia da árvore e o papel de cada *switch* (isto é, ToR, AGGR ou CORE). Este conhecimento da topologia é um pré-requisito para viabilizar o roteamento na origem o que, além de reduzir os esforços operacionais, permite o encaminhamento correto e otimizado dos pacotes. Para este fim, criamos um protocolo (*Role Discovery Protocol*) que automatiza a inferência da árvore de *switches*, adicionando uma extensão no protocolo de descoberta LLDP. Nosso protocolo é bastante simples e requer apenas a identificação da camada em que o *switch* está situado.

3. Implementação e testbed do protótipo

A implementação do mecanismo de transmissão iBF é baseada em *switches* OpenFlow [6], enquanto o RM é implementado como uma aplicação adicionada ao controlador NOX [5]. A seguir, descrevemos as principais questões relacionadas à implementação do protótipo e do ambiente de testes.

3.1. OpenFlow

Um *switch* OpenFlow (OF) separa o encaminhamento de pacotes rápido (plano de dados) do nível de decisões de encaminhamento (plano de controle) de um roteador ou *switch*. Embora parte do plano de dados ainda encontre-se residente 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. Dispositivos com OF habilitado e o(s) respectivo(s) controlador(eres) comunicam-se através do

protocolo OF (OFP - *Open Flow Protocol*), que define mensagens como *packet-received, send-packet-out, modify-forwarding-table* e *get-stats*.

O aspecto principal do OF é definir uma interface de captação na forma de uma tabela de fluxos, com entradas que contêm um conjunto de campos de pacote que combina uma tupla formada por 10 elementos: $\langle inport, Eth_{src}, Eth_{dst}, VLAN, EthType, IP_{proto}, IP_{src}, IP_{dst}, TCP_{src}, TCP_{dst} \rangle$, e uma lista de ações suportadas em hardware como, por exemplo, encaminhar para uma porta, encapsular e transmitir para o controlador ou descartar o pacote. A fim de apoiar o encaminhamento com base no iBF, apenas uma pequena alteração foi necessária na implementação do OpenFlow (v. 0.89rev2).

3.2. Gerenciador de Rack (RM)

O RM (*Rack Manager*) atua como um controlador de *switches* e a sua implementação é instanciada através de um aplicativo que executa no contexto do controlador NOX [5]. O NOX está disponível gratuitamente traduzindo-se em um importante *framework* para construir novas aplicações para interação com os dispositivos que possuem o OF habilitado. Em poucas palavras, a interface de programação do NOX é construída sobre os eventos, seja por componentes principais do NOX (*core*), ou definidas por usuários, e gerenciados diretamente a partir de mensagens OF como `packet-in`, `switch join`, `switch leave`, etc.

3.3. Ambiente de Teste (Testbed)

O ambiente de teste é composto por 5 nós físicos, um deles hospeda o controlador NOX com o componente RM e os 4 restantes são compartilhados, cada um, por 9 máquinas virtuais: 5 instâncias de OF *switches* e 4 nós finais. A Figura 2 mostra o *testbed*, onde as linhas sólidas representam ligações diretas entre as máquinas virtuais e as linhas tracejadas representam as conexões entre as máquinas virtuais de diferentes máquinas físicas. A topologia em cada máquina física é configurada com o OpenFlowVMS, o qual dispõe de um conjunto útil de *scripts* para automatizar a criação de máquinas virtuais em rede utilizando o QEMU. *Scripts* adicionais foram desenvolvidos para distribuir o ambiente em diferentes máquinas físicas usando conexões SSH e *switches* virtuais desprovidos de inteligência e baseados no VDE (*Virtual Distributed Ethernet*). O conjunto de *scripts* desenvolvido neste trabalho permite definir rapidamente uma topologia e automatizar a iniciação dos nós virtuais e do OF *switch*, incluindo a configuração de IP, a criação de *data path*, a iniciação do módulo OFP e conexão ao controlador.

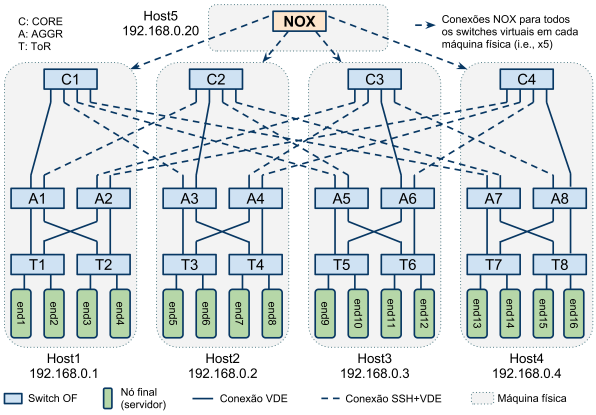

Figura 2. Ambiente de teste.

4. Avaliação

Depois de validar a implementação do protótipo através da verificação da conectividade total entre o conjunto de servidores (16 VMs), a próxima questão é avaliar o encaminhamento baseado no iBF em termos dos seguintes itens: (i) custo das informações de estado, (ii) os potenciais efeitos de falsos positivos, e (iii) a capacidade de balanceamento de carga. Devido às limitações de um ambiente virtualizado, os aspectos de desempenho não são considerados.

4.1. Análise do Estado

A configuração de rede é uma topologia em 3 camadas, com ToRs conectados a 2 servidores e dois AGGRs. As p_1 portas (p.e, p_1=4) do AGGRs são usados para conectar-se a $p_1/2$ ToRs e $p_1/2$ COREs. Em consonância com a literatura, assumimos uma média de 10 fluxos simultâneos por servidor (5 subindo e 5 descendo). Devido ao roteamento baseado na origem, nossa implemantação requer um estado mínimo nos COREs e AGGRs, ou seja, apenas uma entrada por interface (vizinho). Além disso, a escalabilidade no DCN não tem impacto sobre o número de entradas de fluxo nos *switches*, que é constante e igual ao número de vizinhos.

4.2. Falsos positivos

Avaliou-se o desempenho de falso positivo em filtro de Bloom de 96-bits quando se observa apenas

3 elementos, chamados de três endereços *Bloomed MAC*, que representam um caminho na rede na topologia DCN. A estimativa normalmente utilizada para a probabilidade de falso positivo de um filtro de Bloom de tamanho m, inserido com n elementos, com número de k funções *hash* é:

$$p^k = \left[1 - \left(1 - \frac{1}{m}\right)^{k*n}\right]^k \quad (1)$$

Temos que avaliar a viabilidade e a eficiência da nossa escolha de projeto para evitar falsos positivos, com base em descartar candidatos iBF propensos a falsos positivos antes da sua utilização. A nossa tese é que, dada a baixa fpr (*false positive rate*) de um iBF de 96-bits, há uma abundância de caminhos livres de falsos positivos entre quaisquer ToRs. A partir da teoria de filtros de Bloom, existe um número ideal de funções de *hash* ($k_{opt} = ln2 * m/n$, com $m = 96, n = 3$) que minimiza a probabilidade de falsos positivos que, no nosso caso, seria até 22 funções de *hashing*. Em nossa configuração prática, porém, o menor fpr foi obtido para k em torno de 7.

4.3. Balanceamento de carga

Dada uma matriz de tráfego (TM), o objetivo é avaliar como o tráfego é espalhado entre os enlaces disponíveis. Comparamos a utilização do enlace da nossa implementação com uma execução simplificada do *Spanning Tree Protocol* (STP) sobre a mesma topologia. Usamos ITG como gerador de tráfego, configurado com fluxos de TCP com duração de 10s, com tamanhos de carga exponencialmente distribuídos em torno de 850 bytes. Estes parâmetros são adequados para a maioria dos tráfegos DCN. A Figura 3 mostra a utilização normalizada após a repetição de dez experimentos. Como esperado, na STP as ligações de rede são muito utilizadas, enquanto o tráfego com iBF espalha adequadamente, com a utilização máxima e mínima normalizada de qualquer ligação desviando apenas cerca de 20% do valor ideal, ou seja, 1.

5. Conclusões

Apresentamos uma arquitetura de rede de *data center* com base em uma simples camada de plano de dados sob o IP, que encaminha pacotes baseado no

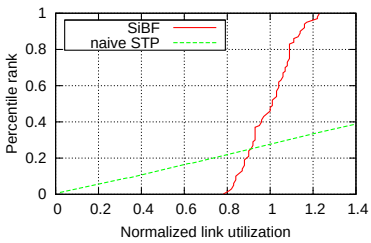

Figura 3. Testes.

conteúdo de um filtro de Bloom nos pacotes. A proposta DCN apresenta muitas características atraentes como, por exemplo, não exigir qualquer modificação nos nós finais, reutilizando o cabeçalho do pacote Ethernet, e um controle preciso sobre as rotas dos pacotes no *data center*. A avaliação sobre uma implementação de teste virtualizado em pequena escala, não só oferece uma prova de conceito, mas também lança luz sobre a capacidade de fornecer balanceamento de carga com iBFs. Em implementações futura, o protótipo será melhorado (por exemplo, para lidar com casos de falha) e estendido com características adicionais, como gerenciamento de banco de dados distribuídos.

Referências

[1] Mohammad Al-Fares, Alexander Loukissas, and Amin Vahdat. A scalable, commodity data center network architecture. *SIGCOMM CCR*, 38(4):63–74, 2008.

[2] Albert Greenberg, James Hamilton, Navendu Jain, Srikanth Kandula, Changhoon Kim, Parantap Lahiri, David Maltz, Parveen Patel, and Sudipta Sengupta. Vl2: a scalable and flexible data center network. *SIGCOMM CCR*, 2009.

[3] Albert Greenberg, James Hamilton, David Maltz, and Parveen Patel. The cost of a cloud: research problems in data center networks. *SIGCOMM CCR*, 2009.

[4] Albert Greenberg, Gisli Hjalmtysson, David Maltz, Andy Myers, Jennifer Rexford, Geoffrey Xie, Hong Yan, Jibin Zhan, and Hui Zhang. A clean slate 4d approach to network control and management. *SIGCOMM CCR*, 2005.

[5] Natasha Gude, Teemu Koponen, Justin Pettit, Ben Pfaff, Martín Casado, Nick McKeown, and Scott Shenker. Nox: towards an operating system for networks. *SIGCOMM CCR*, 2008.

[6] Nick McKeown, Tom Anderson, Hari Balakrishnan, Guru Parulkar, Larry Peterson, Jennifer Rexford, Scott Shenker, and Jonathan Turner. Openflow: enabling innovation in campus networks. *SIGCOMM CCR*, 2008.

[7] Radhika Mysore, Andreas Pamboris, Nathan Farrington, Nelson Huang, Pardis Miri, Sivasankar Radhakrishnan, Vikram Subramanya, and Amin Vahdat. Portland: a scalable fault-tolerant layer 2 data center network fabric. In *SIGCOMM '09: ACM SIGCOMM 2009 conference on Data communication*, 2009.