Arquitetura Básica

Introdução

Arquitetura Básica

Pilha de Software

Implementação de Hosts e Switches

Switch Bare Metal

Introdução

A SDN é uma abordagem de construção de redes que favorece a programação de hardware comum.

Esses programas têm a inteligência necessária para para controlar a expedição de pacotes e outras operações de rede implementadas em software.

Realizar este design é agnóstico a pilha de protocolos. Requer um conjunto de APIs abertas e um conjunto de componentes de software de suporte.

Vamos definir uma arquitectura básica usando variados componentes que são open-source e estão disponíveis no gitub mas é possível usar outras opões.

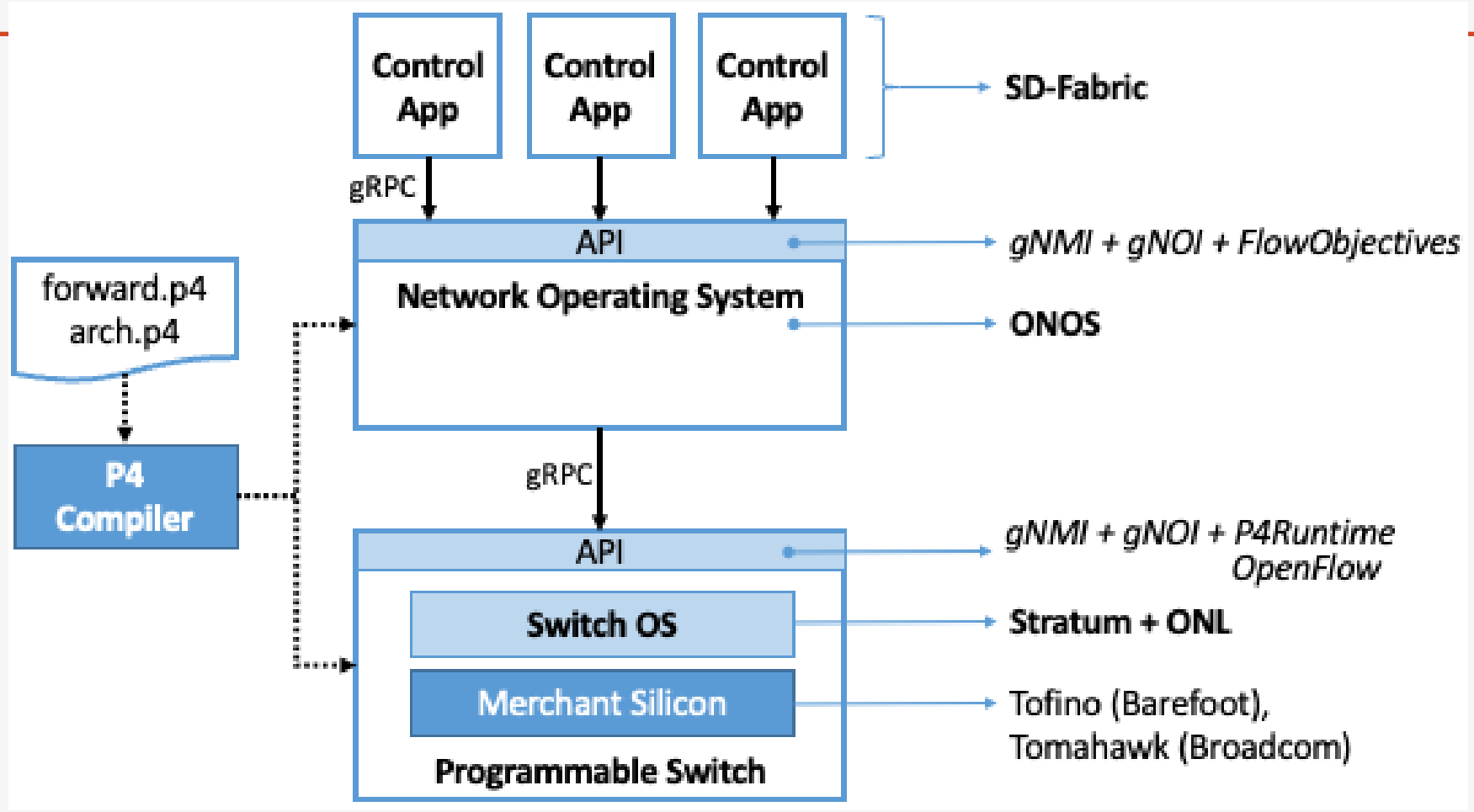
Pilha de Software

Uma visão geral da pilha de software aparece na Figura 5.1, que inclui um Switch Bare-Metal que corre um Switch OS local, controlado por um Network OS que hospeda uma coleção de Aplicações de Controlo.

A Figura 5.1 também mostra

- à direita: o conjunto correspondente de componentes exemplo de código aberto (SD-Fabric, ONOS e Stratum)
- à esquerda: um conjunto de ferramentas relacionadas com P4

Este capítulo introduz esses componentes, com capítulos posteriores fornecendo mais detalhes.



Arquitetura geral da pilha de software SDN.Implantação.

Pilha de Software

Observe a semelhança entre este diagrama e a Figura 3.2 no Capítulo 1. Ambas as figuras incluem duas interfaces: uma entre as aplicações de controle e o Sistema Operativo de Rede (NOS) e uma segunda entre o NOS e os switches programáveis subjacentes.

Essas duas interfaces são representadas como “calços de API” na Figura 5.1 e no contexto dos componentes exemplo, correspondem a uma combinação de gNMI, gNOI e FlowObject no primeiro caso, e uma combinação de gNMI, gNOI e P4Runtime ou OpenFlow no segundo caso.

gRPC, uma estrutura de chamada de procedimento remoto de código aberto, é mostrada como o protocolo de transporte para esses APIs – uma escolha de implementação, mas que geralmente assumiremos daqui em diante. (Observe que o OpenFlow, ao contrário de outros protocolos, não funciona em gRPC.) Discutimos todos esses acrônimos e interfaces em mais detalhes abaixo.

Pilha de Software

É importante ter em mente que os componentes de software listados na Figura 5.1 correspondem a projetos ativos de código aberto e, como consequência, continuam a evoluir (assim como suas APIs).

Versões específicas de cada componente — e suas APIs — foram integradas e desenvolvidas tanto em ambientes de teste como de produção.

Por exemplo, enquanto a figura mostra o P4Runtime como uma interface de controlo candidata a ser exportada pelo Switch OS, há soluções já desenvolvidas que usam o OpenFlow no seu lugar. (Isto inclui o Comcast).

Pilha de Software

Similarmente, enquanto a figura mostra gNMI/gNOI como interface de configuração ou operacional, há soluções que usam em vez disso o NETCONF.

No livro que foi usado como suporte aos slides destas aulas não é feita uma tentativa de apresentar todas as combinações de versões de components e APIs.

Em vez disso foca-se na única pilha apresentada na Figura 5.1 uma vez que é considerada no nosso julgamento a abordagem certa baseada na experiência passada com versões da pilha para cima e para baixo.

Switch versus Host Implementation

A visão de cima para baixo da pilha de software mostrada na Figura 5.1 é da perspectiva de um único switch, mas é importante também ter em mente a perspectiva da rede.

A Figura 5.2 dá essa perspectiva ao focando em um caminho ponta a ponta pela rede, conectando Máquinas Virtuais (VMs). Essa perspectiva destaca dois aspectos importantes do sistema.

O primeiro reforça o ponto que estivemos a fazer que é o sistema operacional de rede (por exemplo, ONOS) seja em toda a rede, enquanto o sistema operacional Switch (por exemplo, Stratum) É por switch. A segunda é que parte da pilha de software SDN é executada nos hosts finais.

Em particular, existe um Virtual Switch (vSwitch) – normalmente implementado em software como parte do hipervisor em execução no servidor – que é responsável por encaminhar pacotes de e para as VMs. É claro que nem todo host final executa VMs, mas uma arquitetura semelhante se aplica a hosts de contentores ou servidores bare-metal.

Switch vs Host Implementation

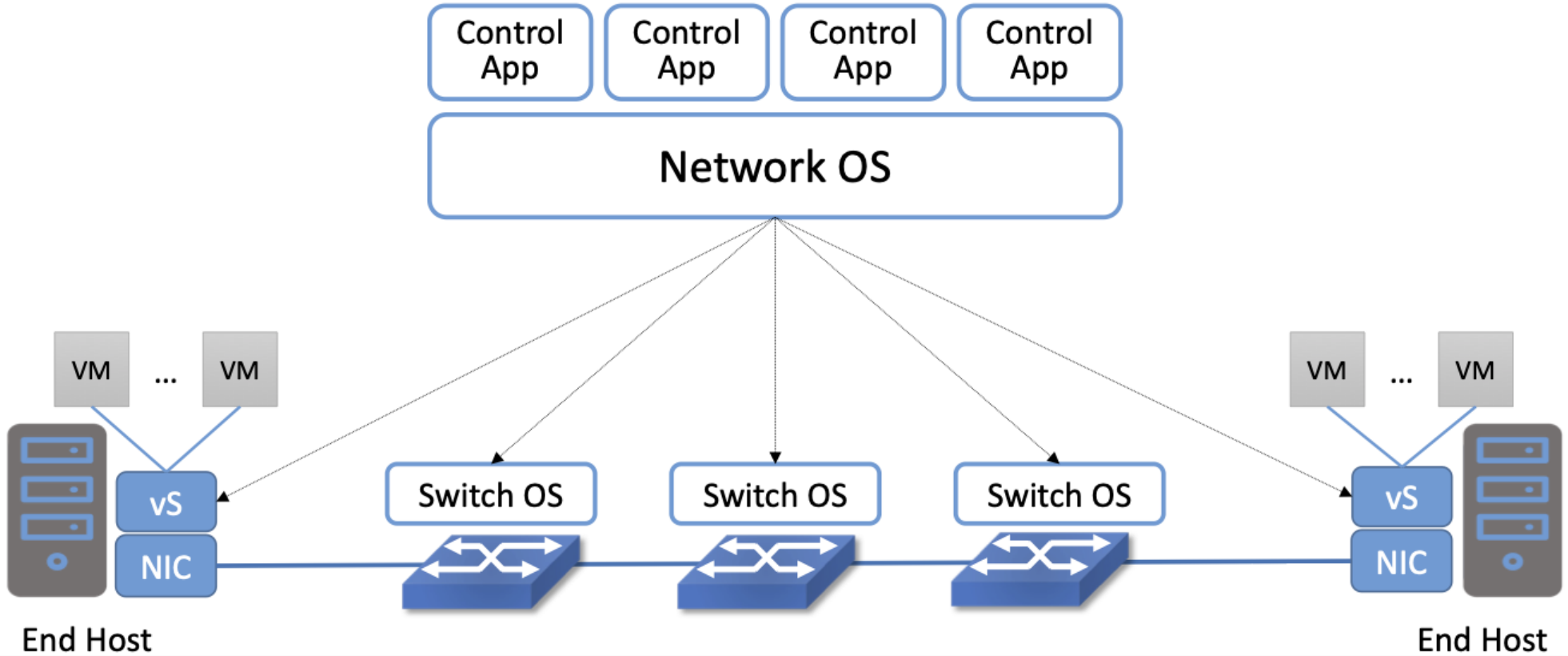
Em particular, existe um Virtual Switch (vSwitch) – normalmente implementado em software como parte do hipervisor em execução no servidor – que é responsável por encaminhar pacotes de e para as VMs.

É claro que nem todo host final executa VMs, mas uma arquitetura semelhante se aplica a hosts de contentores ou servidores bare-metal.

Assim como um switch físico, o vSwitch encaminha pacotes da porta de entrada e a de saída.

As portas do vSwitch são virtuais e estão conectadas a VMs (ou contentores).

Perspectiva centrada na Redes



Perspectiva de ponta a ponta de uma rede definida por software, incluindo os hosts finais e as máquinas virtuais (VMs) que eles hospedam.

Perspectiva de Rede para SDN

Este livro adota uma perspectiva de SDN orientada para rede, que trata o host final (tanto o host virtual quanto o host físico), o switch em execução no sistema operativo do host e a NIC que conecta o host à à rede como uma extensão do rede, rodando sob o controle de um sistema operativo de rede.

Uma perspectiva mais centrada no host é igualmente válida, e talvez mais importante, vem com um ecossistema robusto de software de código aberto que é executado como parte do sistema operativo do host.

DPDK é um exemplo, mas outro que está ganhando força é a combinação de eBPF (extended Berkeley Packe Filter) e XDP (eXpress Data Path).

Perspectiva de Rede para SDN

Quando usados juntos, eles fornecem uma maneira de programar Regras Match-Action no kernel do sistema operativo ou potencialmente até mesmo num SmartNIC.

Isto é semelhante em espírito ao OpenFlow e P4, exceto que permitem que a parte Action seja um programa arbitrário. Por outro lado, o OpenFlow define um conjunto fixo de Ações, e P4 é uma linguagem restrita para expressar Ações (por exemplo, não inclui loops).

Isto é necessário quando a Acção deve ser executada dentro de um orçamento de ciclo fixo, como é o caso para um pipeline de encaminhamento baseado em switch. Também permite a verificação formal do plano de dados, um método promissor cuja oportunidade é discutida no Capítulo 10.

. Implementação de Switches e Hosts

Felizmente, podemos ver um vSwitch a comportar-se exatamente como um switch físico, incluindo as APIs que ele suporta.

O fato de um vSwitch ser implementado em software um processador de uso geral, e não em um ASIC, é um detalhe de implementação.

Embora esta seja uma afirmação verdadeira, a mudança para software reduz drasticamente a barreira para a introdução de recursos adicionais, de modo que o conjunto de recursos é mais rico e dinâmico.

Implementação de Switches e Hosts

utilizado que oferece suporte a OpenFlow como uma API norte.

Ele formou o plano de dados para a plataforma original de virtualização de rede Nicira. O OVS foi integrado a uma variedade de ferramentas complementares, como o DPDK (Data Plane Development Kit), outro componente de código aberto que otimiza as operações de encaminhamento de pacotes em processadores x86.

Embora seja um tópico importante, este livro não explora toda a gama de possibilidades para um vSwitch como o OVS ou outras otimizações de host final, mas trata os vSwitches como qualquer outro switch ao longo do caminho ponta a ponta.

Implementação de Switches e Hosts

Outro detalhe de implementação mostrado na Figura 16 é que o host pode ter uma placa de interface de rede inteligente (SmartNIC) que auxilia (ou possivelmente até substitui) o vSwitch.

Os fornecedores têm um longo histórico de descarregamento da funcionalidade do kernel em NICs (por exemplo, tudo, desde o cálculo de somas de verificação TCP/IP até o suporte a VMs), mas no contexto SDN, a possibilidade interessante é replicar o pipeline de encaminhamento encontrado nos switches de rede.

Novamente, há uma série de opções de implementação possíveis, incluindo FPGA e ASIC, bem como se a NIC é de função fixa ou programável (usando P4).

Para nossos propósitos, trataremos tais NICs inteligentes como mais um elemento de comutação ao longo do caminho ponta a ponta.

Bare-Metal switches

Começando na parte inferior e subindo na pilha mostrada nas Figuras 15 e 16, o plano de dados da rede é implementado por um conjunto interconectado de switches bare-metal. Nosso foco por enquanto está em um único switch, onde a topologia geral da rede é ditada pelas aplicaçoes de controlo executadas no topo da pilha de software. Por exemplo, descreveremos uma aplicação de controle que gere uma topologia leaf-spine em uma seção posterior. A arquitetura é independente do fornecedor do switch, mas toda a pilha de software descrita neste capítulo é executada em switches construídos com chips de comutação Tofino e Tomahawk fabricados pela Barefoot (Intel) e pela Broadcom, respectivamente. O chip Tofino tem um pipeline de encaminhamento programável baseado no PISA, enquanto o chip Tomahawk (pipeline de função fixa).

Bare-Metal switches

No caso de ambos os chips, um par de programas P4 define o pipeline de encaminhamento.
O primeiro (forward.p4) especifica o comportamento de encaminhamento.
O segundo (arch.p4) especifica a arquitetura lógica do chip de encaminhamento de destino.
O compilador P4 gera arquivos de destino que são carregados no sistema operativo da rede e no switch.
Esses arquivos de destino não são nomeados na Figura 15 (retornaremos aos detalhes nos Capítulos 4 e 5).
Ambos os componentes precisam saber sobre a saída porque um implementa o comportamento de encaminhamento (o switch) e o outro controla o comportamento de encaminhamento. (o sistema operativo de rede)

Bare-Metal switches

Retornaremos aos detalhes da cadeia de ferramentas do compilador no Capítulo 4. Por enquanto, abordaremos apenas a questão de porque precisamos de um programa P4 no caso de um chip de comutação de função fixa (já que não estamos a usar o P4 para modificar o seu programa fixo).

O resumo rápido é que é necessária uma especificação formal do pipeline de encaminhamento para gerar a API para o plano de dados.

Os programas P4 são escritos num modelo abstrato do pipeline de encaminhamento e, independentemente de o pipeline de hardware real do chip ser fixo ou programável, ainda precisamos saber como mapear o pipeline abstrato no pipeline físico.

É aqui que arch.p4 desempenha um papel. Quanto à função de forward.p4, este programa na verdade prescreve o pipeline no caso de um chip programável, enquanto para o chip de função fixa, forward.p4 apenas descreve o pipeline.
Mas ainda precisamos de forward.p4 em ambos os casos porque o conjunto de ferramentas o utiliza, junto com arch.p4, para gerar a API que fica entre os planos de controle e o de dados.

Sistema Operativo do Switch

Partindo do hardware de base, cada switch executa um sistema operativo de switch (SOS) local. Não deve ser confundido com a NOS que gere uma rede de switches.

Este SOS é executado em um processador comum interno ao switch (não mostrado na Figura 5.1). Ele é responsável por lidar com chamadas de API emitidas para o switch, por exemplo, do NOS.

Isto inclui tomar as medidas apropriadas nos recursos internos do switch, que às vezesafetam o chip de comutação.

Switch OS

Vários sistemas operativos de switch de código aberto estão disponíveis (incluindo SONiC, originalmente desenvolvido no Microsoft Azure),mas usamos uma combinação de Stratum e Open Network Linux (ONL) como nosso exemplo principal.

O ONL é um distribuição de Linux pronta para switch (originalmente preparada pela Big Switch Networks), enquanto Stratum (originalmente desenvolvido no Google) é o principal responsável pela tradução entre a API externa e a API interna para trocar recursos. Por esse motivo, às vezes referimos-nos ao Stratum como SO Thin Switch.

SwitchOS

O Stratum faz a intermediação entre o switch e o mundo de fora . Isto inclui a carga o carregamento de ficheiros gerados pelo compilador P4, que define um contrato entre o plano de dados e o plano de controle. Este contrato substitui efetivamente a abstração de regras de fluxo do OpenFlow por uma especificação gerada automaticamente.

Este contrato substitui efetivamente a abstração de regras de fluxo do OpenFlow por uma especificação gerada automaticamente. O o restante da API gerenciada pelo Stratum é definido da seguinte forma:

• P4Runtime: Uma interface para controlar o comportamento de encaminhamento em tempo de execução. É a chave para povoar encaminhar tabelas e manipular o estado de encaminhamento. O P4Runtime é independente de qualquer Programa P4 e independente do hardware subjacente. Isso contrasta com o OpenFlow, que é bastante prescritivo sobre o modelo de encaminhamento e como o plano de controle interage com ele. (Para completar, A Figura 5.1 também lista o OpenFlow como uma interface de controle alternativa.)

SwichOS

• gNMI (gRPC Network Management Interface): Usado para definir e recuperar o estado de configuração. gNMI geralmente é combinado com modelos OpenConfig YANG que definem a estrutura da configuração e árvore de estado.

• gNOI (interfaces de operações de rede gRPC): usado para definir e recuperar o estado operacional, por exemplo suporte ao gerenciamento de certificados, testes de dispositivos, atualizações de software e solução de problemas de rede. Se você se lembrar da distinção entre Controle e Configuração apresentada no Capítulo 1, então você reconhecerá nize P4Runtime como a API de controle e a combinação gNMI/gNOI como uma versão moderna do tráfego de um switch API de configuração adicional. Esta última API tem sido historicamente chamada de interface OAM (para “Operações, Administração e Manutenção”), e na maioria das vezes tem sido implementado como uma interface de linha de comando (que obviamente não é realmente uma API).