Redes Definidas por Software
Aula 5: SwitchOS

Sumário

1- Thin Switch OS

2- P4Runtime

3- gNMI and gNOI

4- SONiC

Introdução

Este capítulo descreve o sistema operativo executado em cada switch bare-metal. Um bom modelo mental é pensar nisso como análogo a um sistema operativo de servidor.

Há um processador de uso geral executando um sistema operativo baseado em Linux, além de um “acelerador de encaminhamento de pacotes” semelhante em espírito a uma GPU.

A base mais comum para este Switch OS é o Open Network Linux (ONL), um projeto de código aberto do OCP

O ONL começa com a distribuição Debian do Linux ampliada com suporte para hardware exclusivo para switches, incluindo o módulo de interface Small Form-factor Pluggable (SFP) mostrado na Figura seguinte.

Introdução

Este capítulo não entra nesses detalhes do driver de dispositivo de baixo nível, mas em vez disso concentra-se no Northbound Interface (NBI) exportada pelo sistema operativo do switch para o plano de controle, independentemente desse plano de controle ser executado no switch (como um programa executado no espaço do usuário na parte superior do sistema operacional do switch) ou fora do switch (como um controlador SDN como o ONOS).

E conforme apresentado no Capítulo 3, usaremos Stratum como nosso exemplo concreto da camada de software que implementa este NBI em cima do ONL. Stratum às vezes é chamado de Thin Switch OS, onde o sistema operacional é “thin” porque essencialmente implementa um calço de API.

O que é interessante sobre o calço é o conjunto de APIs que ele suporta e, correspondentemente, a grande maioria deste capítulo se concentra nessas APIs

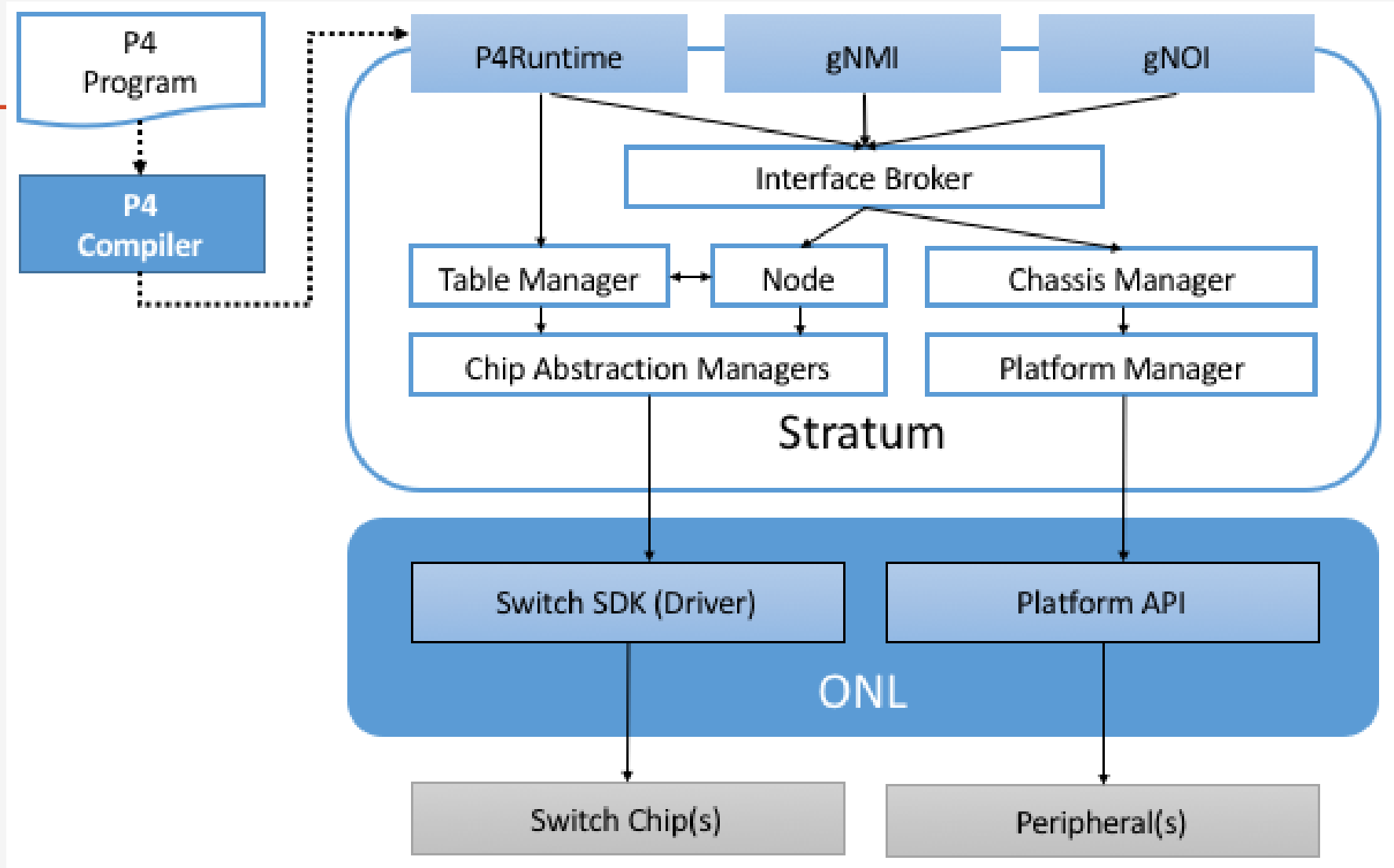
Thin Switch OS

Esta seção descreve o conjunto de componentes que implementam uma interface SDN Northbound pronta para o Switch, o sistema operativo em execução num switch bare-metal.

Os detalhes foram extraídos do Stratum, um projeto de código aberto da ONF que começou com código de qualidade de produção disponibilizado pelo Google.

A Figura seguinte fornece uma visão de alto nível geral do Stratum e, para enfatizar novamente, quais são as interfaces expostas -P4Runtime, gNMI e gNOI – que são as conclusões importantes deste capítulo.

Mostramos esses poucos detalhes de implementação apenas nesta seção como forma de fundamentar a descrição de um fluxo de trabalho ponta a ponta para desenvolvedores que implementam soluções.



Esquema de alto nível do Stratum, um Thin Switch OS rodando sobre o Open Networking Linux.

Thin Switch OS

Abaixo do Stratum, a arquitetura tira partido de dois componentes. O primeiro é um SDK para o(s) chip(s) de comutação integrado(s).

O Barefoot fornece um SDK semelhante para seu chip Tofino. Pode pensar nesses SDKs como semelhantes aos drivers de dispositivo em um sistema operativo tradicional: eles são usados para ler e gravar indiretamente locais de memória no chip correspondente.

O segundo é a Plataforma ONL (ONLP), que exporta a API da Plataforma mostrada na Figura anterior. Esta API fornece acesso a contadores de hardware, monitores, variáveis de status e assim por diante.

Thin Switch OS

Como um exemplo simples, que ajuda a ilustrar a diferença fundamental entre pipelines de função fixa e programáveis, o SDK da Broadcom define um método bcm_l3_route_create para atualizar a tabela de encaminhamento L3, enquanto o método independente de pipeline correpondente de Barefoot é bf_table_write.

Internos ao Stratum, o restante dos componentes mostrados na Figura 7.1 são projetados principalmente para tornar o Stratum independente de fornecedor.

No caso de um chip programável como o Tofino, o Stratum é amplamente passível de passagem: P4Runtime as chamadas provenientes de cima são passadas diretamente para o Barefoot SDK.

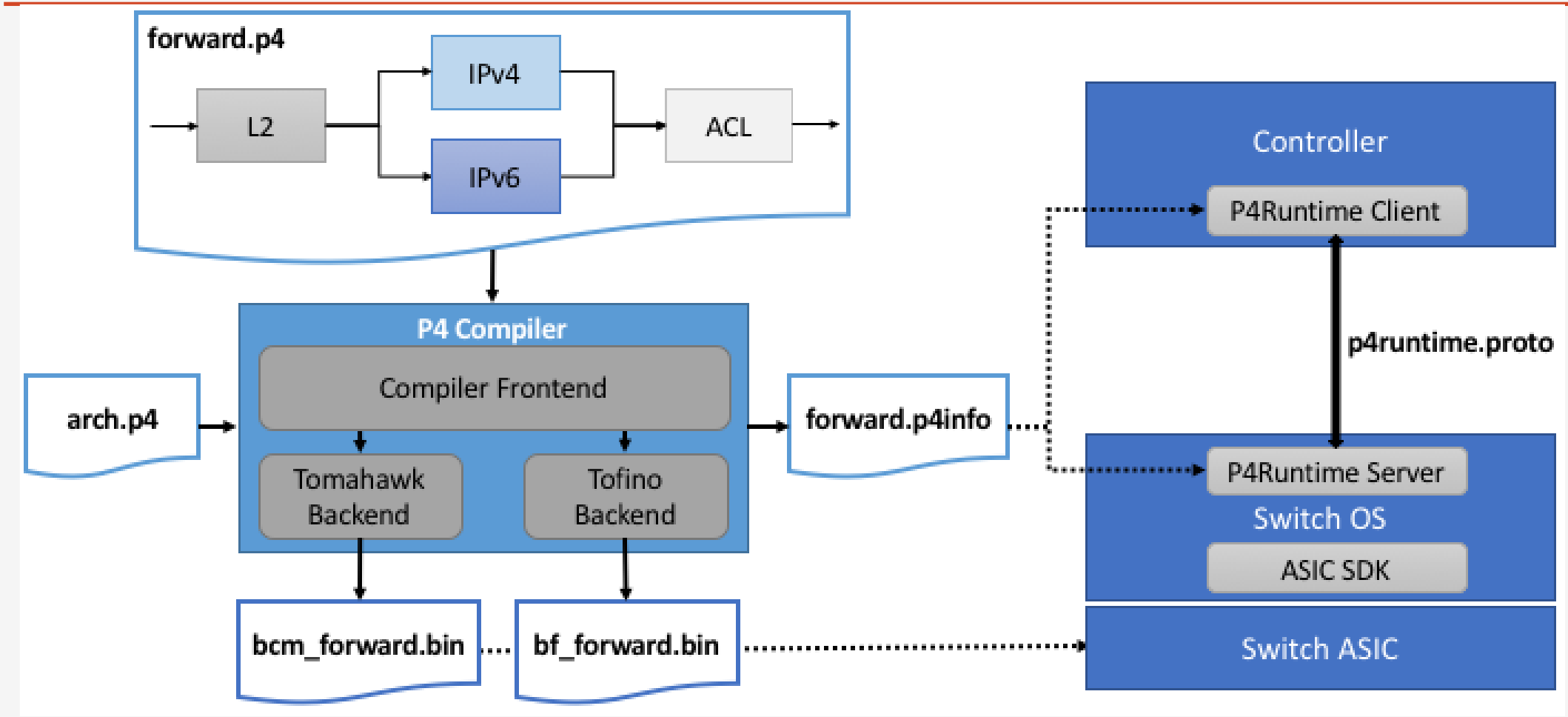
Thin Switch OS

No caso de chip de função fixa como o Tomahawk, o Stratum mantém o estado de tempo de execução necessário para traduzir as chamadas P4Runtime em seu equivalente Broadcom SDK.

Para uma primeira aproximação, isso implica mapear chamadas P4Runtime em switch.p4 em chamadas do Broadcom SDK

Por exemplo, uma chamada P4Runtime para atualizar entradas de tabela em um programa como switch.p4 seria mapeado em uma chamada do Broadcom SDK para atualizar entradas numa das Tabelas ASIC

P4Runtime



A cadeia de ferramentas P4 alcança a independência ASIC e gera automaticamente o contrato P4Runtime (representado como uma especificação de buffer de protocolo).

P4Runtime

Uma conclusão importante da Figura é que o compilador P4 gera o binário que é carregado em cada chip de comutação e a interface de tempo de execução usada para controlar o chip de comutação (indiretamente por meio do Switch OS).

O compilador faz isso com a ajuda de um backend específico do fornecedor, onde a Figura mostra dois possíveis exemplos.

Observe que esses backends específicos do fornecedor devem ser escritos para um modelo de arquitetura específico (como definido por arch.p4 neste exemplo)

P4Runtime

Você pode pensar na interface P4Runtime mostrada na Figura como um stub RPC do lado do servidor para controlar o comutador.

Há um stub correspondente do lado do cliente, que está incluído de forma semelhante no Controlador SDN.

Juntos, eles implementam o contrato P4Runtime entre o controlador e o switch.

O conjunto de ferramentas para gerar este contrato é mostrado na Figura , onde, como nas figuras anteriores, representamos o programa P4 original de expedição como um grafo abstrato em vez de código-fonte P4 real.

P4Runtime

Ou seja, hoje é uma combinação da linguagem P4, do backend específico do ASIC e do modelo de arquitetura que define o ambiente de programação para Injeção de funcionalidades no plano de dados.

A parte final da história de ponta a ponta é a conexão entre o contrato de tempo De execução e o programa original carregado no plano de dados.

Usando o programa de encaminhamento simples apresentado na Seção 4.4 como exemplo, vemos que forward.p4 define uma tabela de consulta, que reafirmamos aqui:

P4RunTime

```
table ipv4_lpm {
        key = {
                hdr.ipv4.dstAddr: lpm;
            }
        actions = {
                ipv4_forward;
                drop;
                NoAction;
                   }
       size = 1024;
       default_action = drop();
```

P4Runtime

```
table ipv4_lpm {
        key = {
            hdr.ipv4.dstAddr: lpm;
            }
        actions = {
            ipv4_forward;
            drop;
            NoAction;
           }
         size = 1024;
         default_action = drop();
```

Contrato P4Runtime

Da mesma forma, o arquivo forward.p4info gerado pelo compilador especifica o contrato P4Runtime.

Como mostrado no exemplo a seguir, ele contém informações suficientes para informar completamente o controlador e o switch sobre como formatar e interpretar o conjunto de métodos gRPC necessários para inserir, ler, modificar e excluir entradas nesta tabela.

Por exemplo, a definição da tabela identifica o campo a ser correspondido (hdr.ipv4.dstAddr) e o tipo de correspondência (LPM), juntamente com as três ações possíveis.

P4Runtime

```
actions {
        preamble {
                id: 16800567
                name: "NoAction"
                alias: "NoAction"
                 }
            }
actions {
         preamble {
                  id: 16805608
                  name: "MyIngress.drop"
```

P4Runtime

```
actions {
  preamble {
                  id: 16800567
                  name: "NoAction"
                   alias: "NoAction"
                  }
  }
actions {
      preamble {
                  id: 16805608
                  name: "MyIngress.drop"
                  alias: "drop"
       }

 }
```

P4Runtime

```
actions {
        preamble {
                id: 16799317
                 name: "MyIngress.ipv4_forward"
                alias: "ipv4_forward"
               }
         params {
                id: 1
                name: "dstAddr"
                bitwidth: 48
            }
```

```
params {
                id: 2
                name: "port“
                bitwidth: 9
            }
        }
tables {
         preamble {
                    id: 33574068
                   name: "MyIngress.ipv4_lpm"
                   alias: "ipv4_lpm"
                 }
```

SONiC

Da mesma forma que o SAI é uma abstração de switch para todo o setor (consulte a Seção 4.5), o SONiC é um fornecedor independente de Switch OS que está ganhando muito impulso na indústria.

Foi originalmente de código aberto pela Microsoft e continua a servir como Switch OS para a Nuvem Azure. SONiC aproveita o SAI como um fornecedor independente SDK e inclui uma distribuição Linux customizada por switch, ou seja, Stratum e SONiC tentam preencher a mesma necessidade.

Hoje, suas respectivas abordagens são amplamente complementares, com os dois softwares de código abertocomunidades que trabalham para uma solução do “melhor dos dois mundos”. Esse esforço é conhecido como PINS, que significapara pilha de rede integrada P4

Tanto SONiC quanto Stratum suportam uma interface de configuração, então unificá-los será uma questão de reconciliação seus respectivos modelos de dados e cadeias de ferramentas.