P4 Programming

(Overview of P4 Programming Features and Architecture)

HITESH KUMAR SHARMA
Technical Instructor & Consultant

NobleProg

ABOUT ME

DR. HITESH KUMAR SHARMA

Technical Instructor & Consultant

NobleProg

- Industry/Academic Experience: **15 Years (6 years in DevOps & Python ,4 Years in RPA, 5 Years in Data Analytics)**
- Worked as **IBM Instructor**
- Worked as **Microsoft Instructor**
- Core Technical Domains: **UiPath RPA, DevOps, Cloud Computing, Data Analytics**
- Academic Qualifications**: Ph.D. (CSE), M.Tech (CSE)**
- Certifications:
  - **UiPath RPA Certified Associate**
  - **Docker Certified Associate**
  - **Neo4J Certified Associate**
  - **Maven Certified Professional**
- 5 Books Published
- 35 Patents Published
- 02 Copyright Published

COURSE OBJECTIVES

Introduction

Overview of P4 Programming Features and Architecture
- Function blocks and interfaces
- Top-down control and design

Programming with the P4 Language
- Protocol-independent switch architecture (PISA)
- Language elements
- Data types

Creating the Server
- Host configuration
- Basic forwarding
- Basic tunneling

Configuring the P4 Software Switch
- Building the P4 compiler
- Installing the software switch

Compiling the P4 Program
- Writing a P4 program over Ethernet packets
- Software switch target

Executing the P4 Program
- Starting the software switch
- Ethernet interface configuration
- CLI commands

Working with P4Runtime
- Runtime control
- Remote and local controls

Monitoring the Network
- Explicit congestion notification (ECN)
- Multi-route inspection

Troubleshooting

Summary and Conclusion

NobleProg

HITESH KUMAR SHARMA

DELIVERY MODE

- 4 Hours Daily of Instructor-Led Training
- Slides Based Content
- Whiteboard/ Pen Tab based Interaction
- Hands-on Live Demo
- Virtual Environment for Labs
- Daily Recap
- Course-End Assessment

PRE-REQUISITES

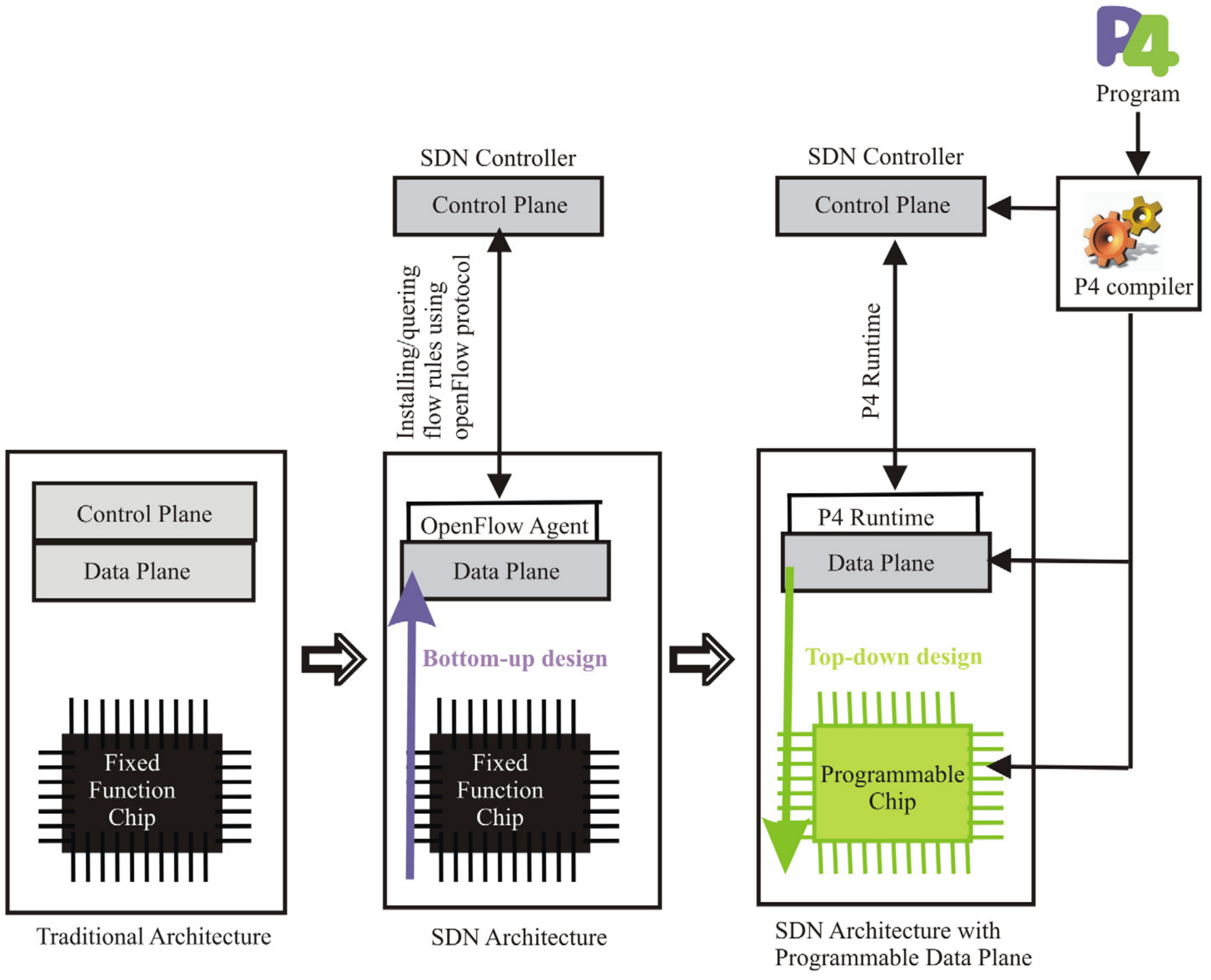
- Basic Knowledge of Networking
- Basic Knowledge of using IDE (VS Code)
- Basic Knowledge of Terminal
- Basic Knowledge of Project Architecture

P4 PROGRAMMING INTRODUCTION

WHAT IS P4 PROGRAMMING?

P4 (Programming Protocol-Independent Packet Processors) is a domain-specific programming language used for programming network devices like switches, routers, and network interface cards. P4 provides a high-level interface to define how packets are processed, parsed, and forwarded within network hardware. It allows network engineers and developers to customize the behavior of networking equipment, enabling the creation of efficient and tailored network protocols.

WHAT IS P4 PROGRAMMING?



NobleProg

KEY CONCEPTS

1. Packet Processing: P4 is focused on defining how packets are processed inside network devices. It allows you to specify how packets are parsed, how headers are extracted, and how forwarding decisions are made.

2. Protocol Independence: Unlike traditional networking languages, P4 is protocol-independent. This means you can define custom protocols or modify existing ones, regardless of the underlying hardware.

KEY CONCEPTS (CONTD..)

3. **Match-Action Paradigm:** P4 programs use a match-action paradigm, where packets are matched against rules, and corresponding actions are executed. This allows for flexible and granular control over packet processing.

4. **Runtime Switch Configuration:** P4 programs can be loaded onto network devices at runtime, allowing for dynamic configuration changes without requiring hardware updates.

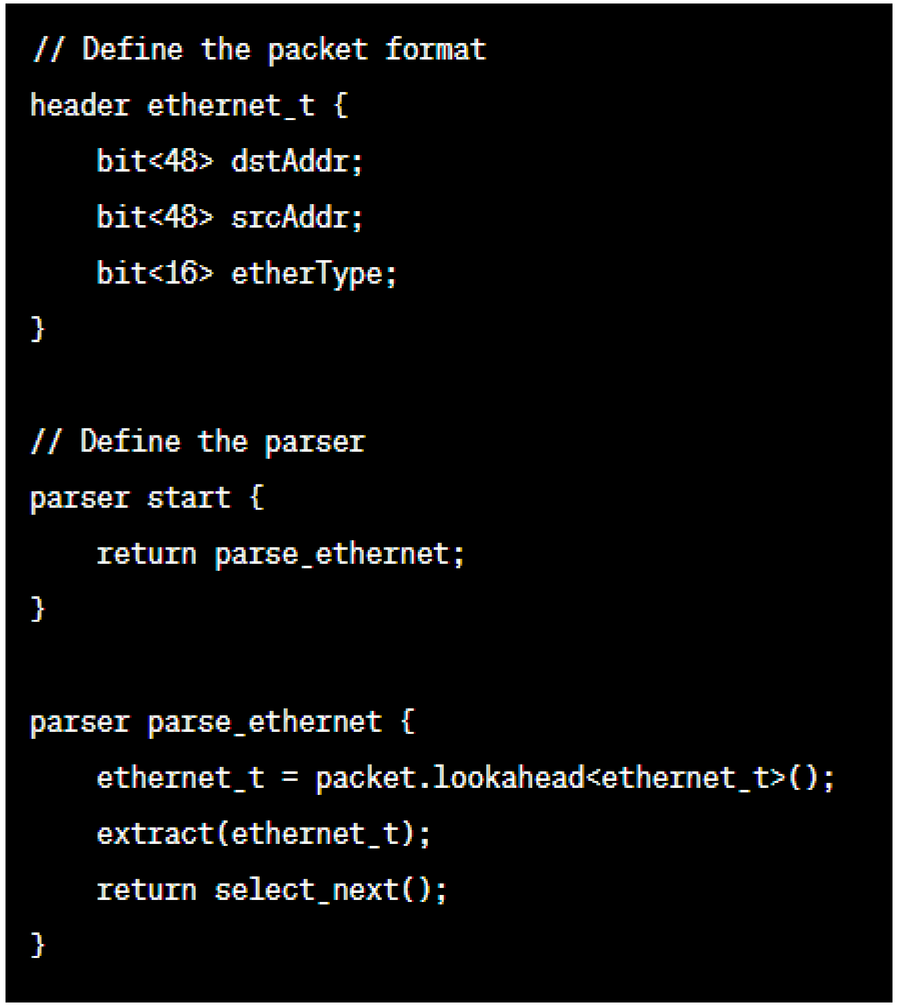
BASIC STRUCTURE

A P4 program typically consists of two main parts:

Parser: Defines how incoming packets are parsed. It specifies how different header fields are extracted from the raw packet data.

Control Blocks: Contains the logic for packet processing. This is where you define match-action tables, apply actions based on matches, and determine how packets are forwarded.

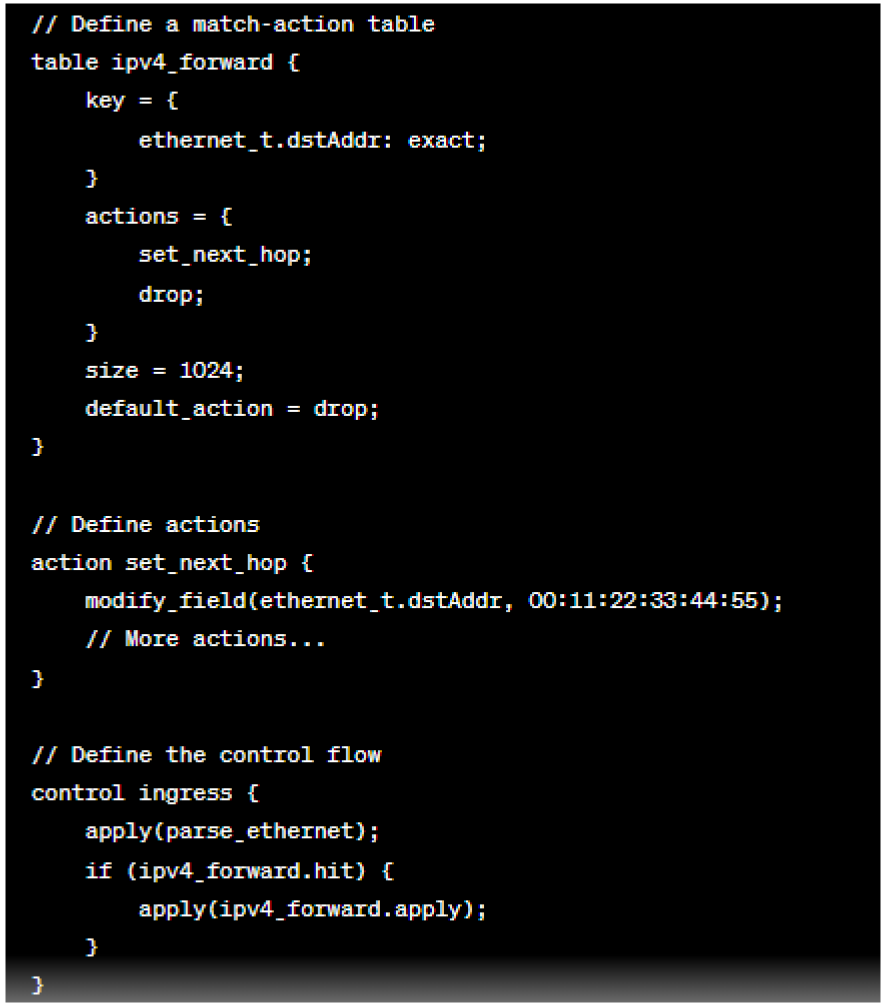
EXAMPLE P4 PROGRAM



```
// Define the packet format
header ethernet_t {
    bit<48> dstAddr;
    bit<48> srcAddr;
    bit<16> etherType;
}

// Define the parser
parser start {
    return parse_ethernet;
}

parser parse_ethernet {
    ethernet_t = packet.lookahead<ethernet_t>();
    extract(ethernet_t);
    return select_next();
}
```



```
// Define a match-action table
table ipv4_forward {
    key = {
        ethernet_t.dstAddr: exact;
    }
    actions = {
        set_next_hop;
        drop;
    }
    size = 1024;
    default_action = drop;
}

// Define actions
action set_next_hop {
    modify_field(ethernet_t.dstAddr, 00:11:22:33:44:55);
    // More actions...
}

// Define the control flow
control ingress {
    apply(parse_ethernet);
    if (ipv4_forward.hit) {
        apply(ipv4_forward.apply);
    }
}
```

NobleProg

FEATURES OF P4 PROGRAMMING

1. **Protocol Independence:** P4 is designed to be protocol-independent. It allows programmers to define new network protocols or modify existing ones without requiring changes to hardware. This enables flexibility and innovation in networking.

2. **Abstraction of Network Hardware:** P4 abstracts the underlying network hardware details, enabling developers to focus on high-level packet processing logic rather than dealing with hardware-specific intricacies.

NobleProg

FEATURES OF P4 PROGRAMMING (CONTD..)

3. Match-Action Paradigm: P4 follows a match-action paradigm, where packets are matched against defined rules, and corresponding actions are taken. This provides fine-grained control over packet processing and forwarding.

4. Runtime Programmability: P4 programs can be loaded onto network devices at runtime, allowing for dynamic reconfiguration of network behavior without hardware modifications.

FEATURES OF P4 PROGRAMMING (CONTD..)

5. Custom Parsing: P4 allows programmers to specify how packet headers are parsed, extracted, and organized. This is crucial for handling various protocol formats.

6. Flexible Table-Based Processing: P4 supports match-action tables, which enable efficient packet processing based on match criteria and corresponding actions. This table-based approach offers scalability and modularity.

7. Compile-Time Verification: P4 compilers analyze the programs for correctness and compatibility with the target hardware, providing a level of safety and reducing potential runtime errors

NobleProg

P4 ARCHITECTURE

1. Parser: The parser is responsible for extracting header fields from incoming packets. P4 allows you to define how different headers are recognized and parsed.

2. Header and Metadata: Headers represent the various layers of a packet (Ethernet, IP, TCP/UDP, etc.), while metadata holds additional information that may not be part of the packet itself.

3. Control Blocks: Control blocks contain the packet processing logic. They include match-action tables that determine how packets are processed and forwarded.

P4 ARCHITECTURE

4. Match-Action Tables: These are at the heart of P4's processing model. Match-action tables associate patterns (matching conditions) with corresponding actions. Packets are matched against these patterns, and the specified actions are executed.

5. Actions: Actions define the operations to be performed on packets, such as modifying fields, forwarding, or dropping packets.

6. Deparser: The deparser assembles packet headers and metadata back into the appropriate format before transmitting the packet.

P4 ARCHITECTURE

7. Pipeline: The control blocks, including the parser, match-action tables, and deparser, are organized in a pipeline to process incoming packets sequentially.

8. Runtime API: P4 programs can interact with the underlying system using the runtime API. This allows programs to access external information or perform dynamic updates.

9. P4 Compiler: P4 code is compiled into target-specific code that can be loaded onto network devices. The compiler also performs checks to ensure compatibility and correctness.

USE CASES OF P4 PROGRAMMING

1. Software-Defined Networking (SDN): P4 enables SDN controllers to customize the behavior of switches and routers dynamically, optimizing network performance based on specific needs.

2. Network Function Virtualization (NFV): P4 is used to program virtualized network functions to improve the efficiency and flexibility of network services.

3. Custom Network Protocols: P4 allows creating new protocols tailored to specific applications, devices, or environments.

4. Traffic Engineering: P4 enables fine-tuning of packet processing for traffic engineering purposes, optimizing network resources.

END OF MODULE-1

THANK YOU

NobleProg

HITESH KUMAR SHARMA