Προγραμματισμός Δικτυακών Συσκευών σε Επίπεδο Δεδομένων: Η Γλώσσα P4 (8η Άσκηση)

Διαχείριση Δικτύων - Ευφυή Δίκτυα

9ο Εξάμηνο, 2022-2023

9/1/2023

Legacy Δικτυακές Συσκευές

- Line-rate επεξεργασία πακέτων μέσω fixed-function chips στο data plane
- Καθορισμός των δυνατοτήτων της συσκευής από τα features των chips

Προσθήκη νέων δυνατοτήτων σε μια συσκευή:

- Επεξεργασία πακέτων στο control plane (CPU offloading) – μείωση απόδοσης
- Αίτημα για νέο feature στον κατασκευαστή του chip – απαιτεί μήνες/έτη

Οι δικτυακές λειτουργίες (network functions) της συσκευής περιορίζονται από τις δυνατότητες που προσφέρουν οι κατασκευαστές των chips

Software Defined Networking (SDN)

Πλεονεκτήματα:

- Διαχωρισμός data plane και control plane δικτυακών συσκευών
- Match/Action tables: Υλοποίηση λογικής προώθησης πακέτων βασισμένη στις τιμές των πεδίων (match fields) στις L2-L4 επικεφαλίδες
- Ευελιξία μέσω κεντρικοποιημένης διαχείρισης, π.χ. υπολογισμός συντομότερων διαδρομών

OpenFlow: Επικρατέστερο, standardized πρωτόκολλο για υλοποίηση SDN

Βασικό μειονέκτημα:

- Η υποστήριξη νέων δυνατοτήτων στο data plane απαιτεί κοινά αποδεκτές αλλαγές στο OpenFlow header και ανάγκη για standardization

  (Εξέλιξη OpenFlow, 2009 – 2014: Αύξηση των match fields από 12 σε πάνω από 40)
- Ανάγκη για επικοινωνία με εξωτερικό controller: Εισαγωγή καθυστερήσεων

Προγραμματισμός στο Data Plane

Ο διαχειριστής δικτύων προγραμματίζει τη λογική με την οποία η δικτυακή συσκευή προωθεί τη διερχόμενη κίνηση σε πραγματικό χρόνο (line-rate)

- **Programmable chips:** Προσθήκη νέων δυνατοτήτων/πρωτοκόλλων μέσω προγραμματισμού των chips στο data plane
- **Εξοικονόμηση πόρων:** Αγορά δικτυακών συσκευών (whiteboxes) που μπορούν να υποστηρίξουν μελλοντικά πρωτόκολλα
- **Μείωση πολυπλοκότητας**: Αφαίρεση πρωτοκόλλων/δυνατοτήτων που δε χρησιμοποιούνται
- **Network visibility:** Νέες δυνατότητες παρακολούθησης δικτυακής κίνησης και συλλογής μετρικών (telemetry) στο data plane

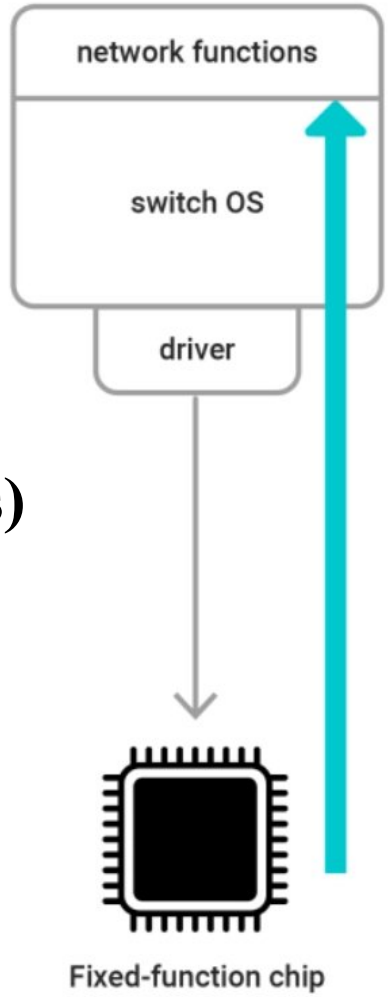
Γλώσσα P4: Επικρατέστερη γλώσσα προγραμματισμού στο data plane

Άλλες λύσεις για programmable data planes: XDP/eBPF, DPDK

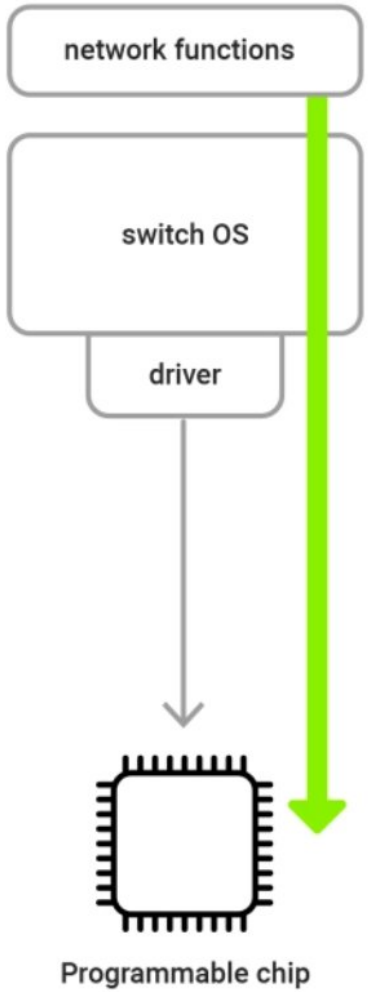
Legacy Networks vs Programmable Data Planes

Bottom-up approach

(legacy network devices)



vs



Top-down approach

(programmable data planes – P4, XDP)

Το chip καθορίζει τις δυνατότητες της συσκευής (network functions)

Ο διαχειριστής δικτύων προγραμματίζει τις δυνατότητες του chip

Σχήματα: https://codilime.com/blog/p4-network-programming-language-what-is-it-all-about/

Η Γλώσσα P4

Programming
Protocol-independent
Packet
Processors

} **P4**

- Κατάλληλη για τον προγραμματισμό της λογικής προώθησης δικτυακών συσκευών
- Compiled programs, παρόμοιο συντακτικό με τη γλώσσα C
- Δύο versions: $P4_{14}$ και $P4_{16}$ (τρέχουσα)
- Domain-Specific Language (DSL): Δεν αποτελεί general-purpose programming language
  Περιλαμβάνει περιορισμούς για την line-rate προώθηση πακέτων:

  - Δεν υπάρχουν loops
  - Δεν υπάρχουν δείκτες (pointers)
  - Fixed-size variables

Σχήμα: https://upload.wikimedia.org/wikipedia/commons/1/12/P4-programming-language-logo.png

P4 Targets (1/2)

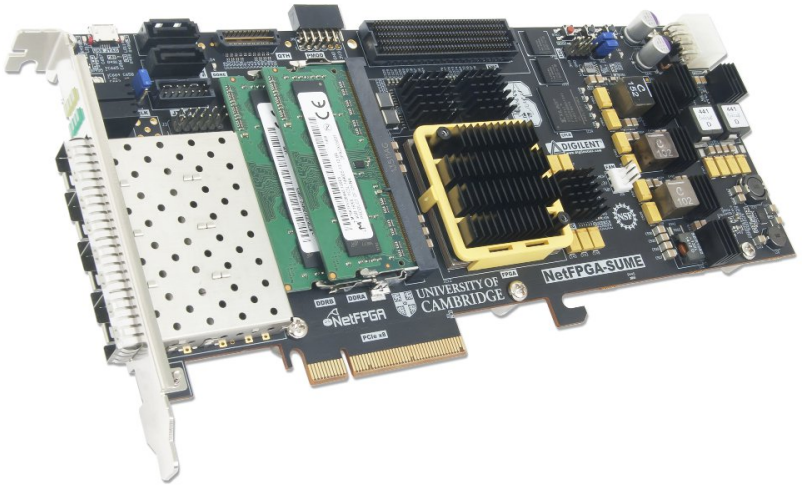
Εκεί φορτώνεται και εκτελείται το compiled πρόγραμμα P4

Διάφορα P4 targets:

- P4 Switch
  - Πολύ μεγάλο throughput
  - Πολλές θύρες
  - Πολύ υψηλό κόστος
  - Περισσότερες δυνατότητες από τα υπόλοιπα P4 targets

π.χ. - Edgecore Wedge 100BF-32X, 32 θύρες των 100Gbps ~ $10,000

- Intel Tofino X532P-T, 32 θύρες των 100Gbps (σύνολο 3.3Tbps)



- FPGA
  - Μεγάλο throughput
  - Ανάγκη για P4-to-VHDL compiler
  - Υψηλό κόστος
  - Ικανοποιητικός αριθμός θυρών

π.χ. NetFPGA-SUME Virtex-7, 4 θύρες των 10 Gbps ~ $7,000

Σχήματα: https://colfaxdirect.com/store/pc/viewPrd.asp?idproduct=3485, https://www.xilinx.com/products/boards-and-kits/1-6ogkf5.html

P4 Targets (2/2)

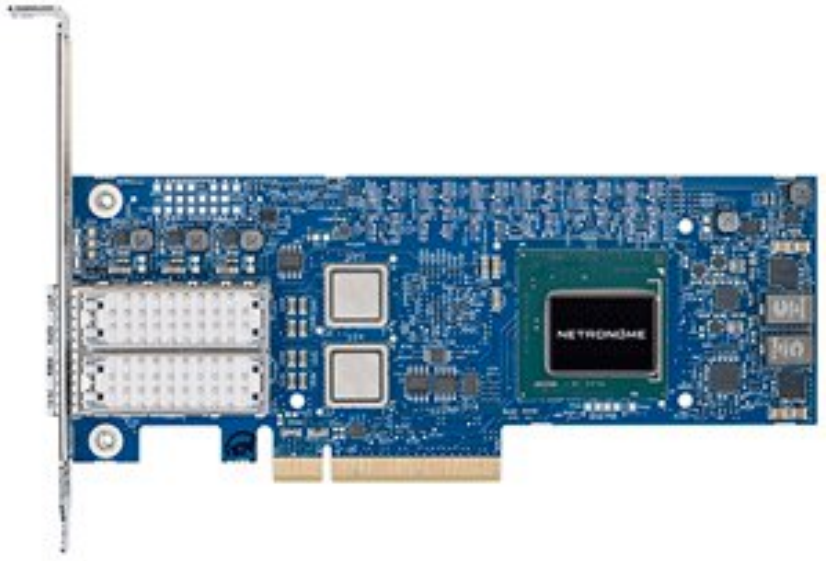
Άλλα P4 targets:

- BMv2 switch (Behavioral Model version 2)
  - Open-source software switch σε πλατφόρμα Linux
  - Χαμηλό throughput
  - Χρησιμοποιείται για πειραματικούς σκοπούς, debugging και εκμάθηση γλώσσας P4

  Repository: https://github.com/p4lang/behavioral-model

- P4-programmable SmartNIC's
  - Μεγάλο throughput
  - Λίγες θύρες
  - Χαμηλό κόστος

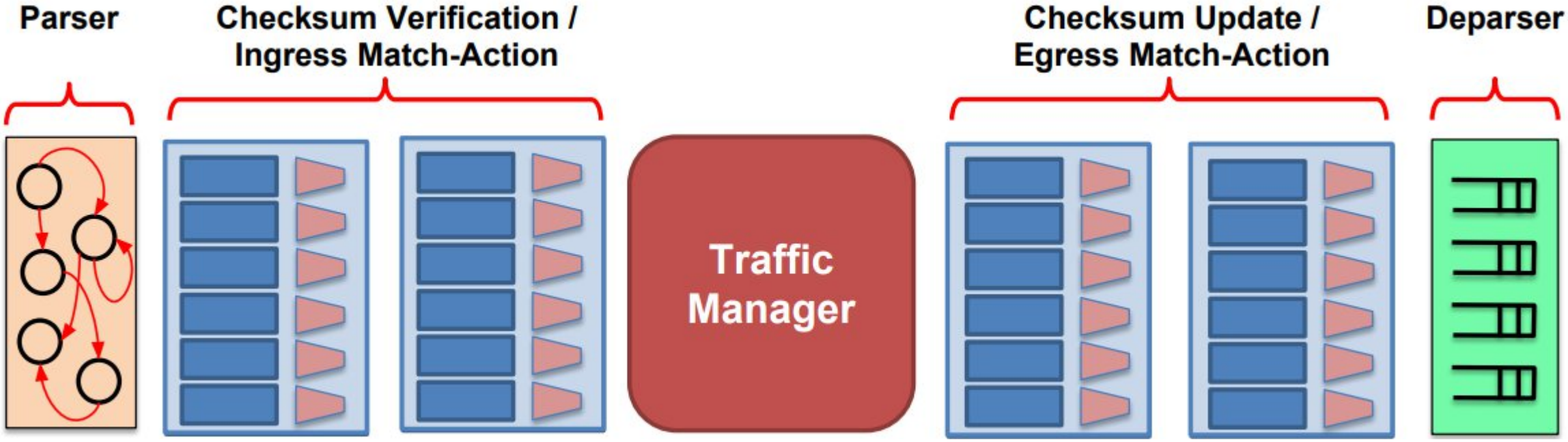
  π.χ. Netronome SmartNIC, 2 θύρες των 25 Gbps ~ $600



Σχήμα: https://www.colfaxdirect.com/store/pc/viewPrd.asp?idproduct=3144

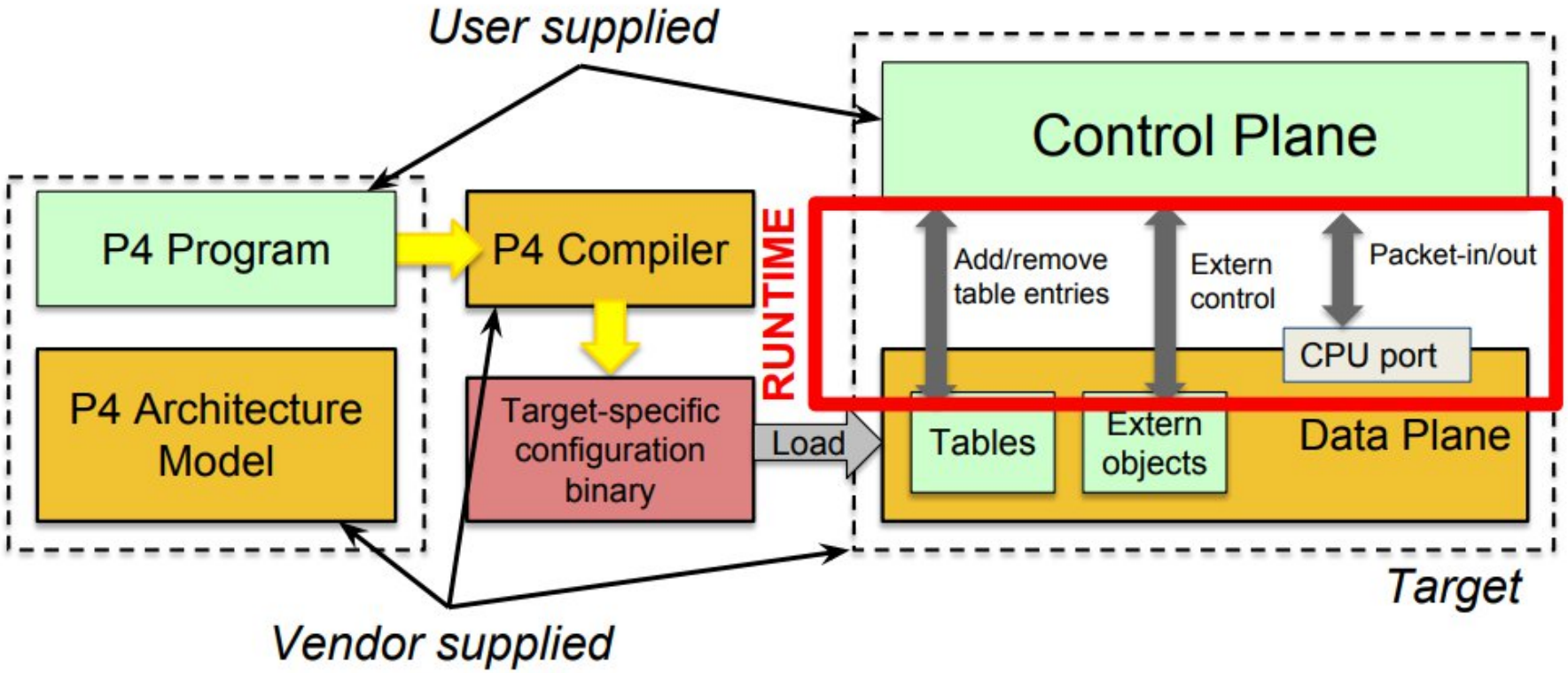
Αρχιτεκτονική P4 (P4 Architecture)

- Ορίζει τα function blocks ενός P4 target, καθώς και τις διεπαφές μεταξύ τους
- Function blocks:
  - Programmable: Προγραμματίζονται από τον P4 developer (parser & control blocks)
  - Fixed-function: Δεν μπορούν να προγραμματιστούν, ορίζονται από τον κατασκευαστή
- Αρχιτεκτονική **V1Model** για BMv2 software switch:



Traffic Manager: Fixed-function στο V1Model (programmable σε άλλες αρχιτεκτονικές, π.χ. PISA architecture σε P4 switches)

Προγραμματισμός P4 Target



- Συγγραφή προγράμματος P4
- Μεταγλώττιση προγράμματος P4 βάσει της αρχιτεκτονικής P4
- Φόρτωση αποτελέσματος στον P4 target (data plane)
- Ανανέωση match/action rules από το control plane (P4Runtime)

Σχήμα: https://opennetworking.org/wp-content/uploads/2020/12/P4_D2_East_2018_01_basics.pdf

Data Types και Headers στην P4

```
typedef bit<48> macAddr_t;
typedef bit<32> ip4Addr_t;
header ethernet_t {
  macAddr_t dstAddr;
  macAddr_t srcAddr;
  bit<16>   etherType;
}
header ipv4_t {
  bit<4>    version;
  bit<4>    ihl;
  bit<8>    diffserv;
  bit<16>   totalLen;
  bit<16>   identification;
  bit<3>    flags;
  bit<13>   fragOffset;
  bit<8>    ttl;
  bit<8>    protocol;
  bit<16>   hdrChecksum;
  ip4Addr_t srcAddr;
  ip4Addr_t dstAddr;
}
```

Βασικοί τύποι δεδομένων (data types):

- bit<n>: μη προσημασμενος (unsigned) ακέραιος n bits
- int<n>: προσημασμενος (signed) ακέραιος n bits
- typedef: custom ορισμοί τύπων δεδομένων

Επικεφαλίδες (headers):

- Διατεταγμένη (ordered) συλλογή από μεταβλητές
- Σχήμα: Παράδειγμα ορισμού Ethernet και IPv4 headers

Επιτρέποντας τον ορισμό των headers, η γλώσσα P4 επιτρέπει την υποστήριξη custom πρωτοκόλλων. Έτσι, ακόμα και οι επικεφαλίδες για κλασικά πρωτόκολλα, π.χ. Ethernet και IPv4, πρέπει να οριστούν στο πρόγραμμα P4

Πηγή: https://opennetworking.org/wp-content/uploads/2020/12/P4_D2_East_2018_01_basics.pdf

Standard Metadata και User Metadata

```
/* Architecture */
struct standard_metadata_t {
  bit<9>  ingress_port;
  bit<9>  egress_spec;
  bit<9>  egress_port;
  bit<32> clone_spec;
  bit<32> instance_type;
  bit<1>  drop;
  bit<16> recirculate_port;
  bit<32> packet_length;
  ...
}

/* User program */
struct metadata {
  ...
}
struct headers {
  ethernet_t   ethernet;
  ipv4_t       ipv4;
}
```

Structs: Μη διατεταγμένη (unordered) συλλογή μεταβλητών

Standard Metadata:

- Καθορίζονται από την αρχιτεκτονική P4
- Δεν μπορούν να διαγραφούν από τον προγραμματιστή
  - ingress_port: η θύρα στην οποία έφτασε το πακέτο
  - egress_spec: η θύρα από την οποία θα εξέλθει το πακέτο

User Metadata:

- Καθορίζονται από τον προγραμματιστή
- Παράγονται κατά τη διάρκεια εκτέλεσης του προγράμματος

Πηγή: https://opennetworking.org/wp-content/uploads/2020/12/P4_D2_East_2018_01_basics.pdf

Parser

```
state start {
  transition parse_ethernet;
}

state parse_ethernet {
  packet.extract(hdr.ethernet);
  transition select(hdr.ethernet.etherType) {
    0x800 : parse_ipv4;
    default : accept;
  }
}

state parse_ipv4 {
  packet.extract(hdr.ipv4);
  transition accept;
}
```

Πηγή: https://cornell-pl.github.io/cs6114/lecture05.html

Parser:

- Finite state machine
- Εντοπίζει τα πεδία της επικεφαλίδας κάθε πακέτου
- Ξεκινάει από την κατάσταση *start*
- Η εντολή *transition* καθορίζει την επόμενη μετάβαση
- Η εντολή *transition accept* σταματά το parsing
- *Extract*: Λέει στον parser να εξάγει ένα τμήμα επικεφαλίδας

Select statement:

- Αντίστοιχο με την εντολή *switch* της γλώσσας C
- Συνέχεια εκτέλεσης του parser ανάλογα με την τιμή της μεταβλητής που δίνεται ως όρισμα στο *select*

Control Blocks - Tables

- Βασική μονάδα του match/action pipeline στην P4
- Key/action data stores: Ορίζουν ποια ενέργεια θα πραγματοποιηθεί ανάλογα με μια τιμή
- Κανόνας ταιριάσματος κλειδιού: (i) exact (ακριβής), (ii) lpm (Longest Prefix Match)

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

Παράδειγμα ορισμού πίνακα για δρομολόγηση IPv4:
- Key: destination IP address
- Actions: Μία από 3 δυνατές ενέργειες θα πραγματοποιηθούν (ορισμός στη συνέχεια)

Control Blocks - Actions

Οι ενέργειες που θα εκτελεστούν ανάλογα με το key του πίνακα πρέπει να οριστούν.

Για το προηγούμενο παράδειγμα:

```
/* core.p4 */
action NoAction() {
}

/* basic.p4 */
action drop() {
  mark_to_drop();
}

/* basic.p4 */
action ipv4_forward(macAddr_t dstAddr,
                    bit<9> port) {
  ...
}
```

Πηγή: https://opennetworking.org/wp-content/uploads/2020/12/P4_D2_East_2018_01_basics.pdf

Control Blocks - Apply

Για την εφαρμογή του πίνακα πρέπει να χρησιμοποιηθεί η εντολή apply, όπως παρακάτω:

```
control MyIngress(inout headers hdr,
                  inout metadata meta,
                  inout standard_metadata_t standard_metadata) {
  table ipv4_lpm {
    ...
  }
  apply {
     ...
     ipv4_lpm.apply();
     ...
  }
}
```

Πηγή: https://opennetworking.org/wp-content/uploads/2020/12/P4_D2_East_2018_01_basics.pdf

Παράδειγμα Προγράμματος σε P4 – V1Model (1/2)

```
#include <core.p4>
#include <v1model.p4>
struct metadata {}
struct headers {}

parser MyParser(packet_in packet,
   out headers hdr,
   inout metadata meta,
   inout standard_metadata_t standard_metadata) {

    state start { transition accept; }
}

control MyVerifyChecksum(inout headers hdr, inout metadata
meta) {   apply {  }   }

control MyIngress(inout headers hdr,
   inout metadata meta,
   inout standard_metadata_t standard_metadata) {
apply {
        if (standard_metadata.ingress_port == 1) {
            standard_metadata.egress_spec = 2;
        } else if (standard_metadata.ingress_port == 2) {
            standard_metadata.egress_spec = 1;
        }
    }
}
```

```
control MyEgress(inout headers hdr,
   inout metadata meta,
   inout standard_metadata_t standard_metadata) {
    apply {  }
}

control MyComputeChecksum(inout headers hdr, inout metadata
meta) {
     apply { }
}

control MyDeparser(packet_out packet, in headers hdr) {
    apply { }
}

V1Switch(
   MyParser(),
   MyVerifyChecksum(),
   MyIngress(),
   MyEgress(),
   MyComputeChecksum(),
   MyDeparser()
) main;
```

Προωθεί πακέτα από τη θύρα 1 στη θύρα 2 και αντίστροφα (λογική hardcoded στο data plane)

Πηγή: https://opennetworking.org/wp-content/uploads/2020/12/P4_D2_East_2018_01_basics.pdf

Παράδειγμα Προγράμματος σε P4 – V1Model (2/2)

```
#include <core.p4>
#include <v1model.p4>
struct metadata {}
struct headers {}

parser MyParser(packet_in packet, out headers hdr,
   inout metadata meta,
   inout standard_metadata_t standard_metadata) {
    state start { transition accept; }
}

control MyIngress(inout headers hdr, inout metadata meta,
   inout standard_metadata_t standard_metadata) {
    action set_egress_spec(bit<9> port) {
        standard_metadata.egress_spec = port;
    }
    table forward {
        key = { standard_metadata.ingress_port: exact; }
        actions = {
            set_egress_spec;
            NoAction;
        }
        size = 1024;
        default_action = NoAction();
    }
    apply {   forward.apply();   }
}
```

```
control MyEgress(inout headers hdr,
   inout metadata meta,
   inout standard_metadata_t standard_metadata) {
    apply {  }
}

control MyVerifyChecksum(inout headers hdr, inout metadata
meta) {   apply { }   }

control MyComputeChecksum(inout headers hdr, inout metadata
meta) {   apply { }   }

control MyDeparser(packet_out packet, in headers hdr) {
    apply { }
}

V1Switch( MyParser(), MyVerifyChecksum(), MyIngress(),
MyEgress(), MyComputeChecksum(), MyDeparser() ) main;
```

Key	Action Name	Action Data
1	set_egress_spec	2
2	set_egress_spec	1

Προωθεί πακέτα από τη θύρα 1 στη θύρα 2 και αντίστροφα (λογική δοσμένη στο control plane)

Πηγή: https://opennetworking.org/wp-content/uploads/2020/12/P4_D2_East_2018_01_basics.pdf

P4 Control Plane - Πίνακες

Στην άσκηση που θα εκτελέσετε θα χρησιμοποιήσετε ως P4 target το BMv2 switch

Για την εισαγωγή εντολών από το control plane του BMv2 switch, πρέπει να χρησιμοποιήσετε την εντολή *simple_switch_CLI*. Στη συνέχεια, θα εμφανιστεί το prompt του *RuntimeCmd*

Χρήσιμες εντολές σχετικές με τους πίνακες που έχετε ορίσει είναι οι παρακάτω:

- Για να δείτε τους πίνακες που έχετε ορίσει στο πρόγραμμα P4:

  RuntimeCmd: show_tables
- Για να δείτε πληροφορίες σχετικές με τον ορισμό ενός πίνακα:

  RuntimeCmd: table_info table_name
- Για να εισαγάγετε μία τιμή στον πίνακα:

  table_add table_name given_action key => value

Πηγή: https://opennetworking.org/wp-content/uploads/2020/12/P4_D2_East_2018_01_basics.pdf (διαφάνεια 48)

P4 Counters

Για τη συλλογή στατιστικών σε σχέση με την κίνηση που διέρχεται από ένα BMv2 switch μπορείτε να ορίσετε μετρητές (counters)

Για να μετρήσετε τα πακέτα και τα bytes που διέρχονται από τις θύρες ενός switch μπορείτε να χρησιμοποιήσετε το παρακάτω τμήμα κώδικα:

```
control MyIngress(...) {

    counter(512, CounterType.packets_and_bytes) port_counter;

    apply {
        port_counter.count((bit<32>)standard_metadata.ingress_port);
    }
}
```

use the ingress port as counter index

Πηγή: https://adv-net.ethz.ch/2019/pdfs/03_stateful.pdf

Η απαραίτητη εντολή για να διαβαστεί η τιμή ενός counter από το Control Plane είναι:

RuntimeCmd: counter_read ctl_ingress.counter_name index

π.χ. για να διαβαστεί η τιμή του παραπάνω μετρητή για τη θύρα 1:

RuntimeCmd: counter_read MyIngress.port_counter 1

Άλλες Μέθοδοι Προγραμματισμού στο Data Plane: The eXpress Data Path (XDP) Framework

Establishes a programmable data path in the Linux Kernel

- **XDP Hook:**
  - Ingress traffic detained & processed before any memory allocation
  - Delivers packets to an extended Berkeley Packet Filter (**eBPF**) Program

- **eBPF Verifier:** Ensures that XDP will not compromise kernel safety:
  - No unbounded loops present
  - Maximum eBPF program size limited
  - No data are read out of bounds

- **eBPF Maps:** Data structures used for communication among user space and eBPF programs

Note: No requirements for specific hardware, e.g. **P4** switches
XDP-enabled NIC's: 25 Gbps Netronome SmartNIC's (~ $600)