P4を使ってみて分かったこと

～現実的になってきたデータプレーンのプログラマビリティ～

APRESIA Systems株式会社
エンタープライズ事業部 技術部
熊谷 渉
wataru.kumagai.qq@apresiasystems.co.jp

P4って何？

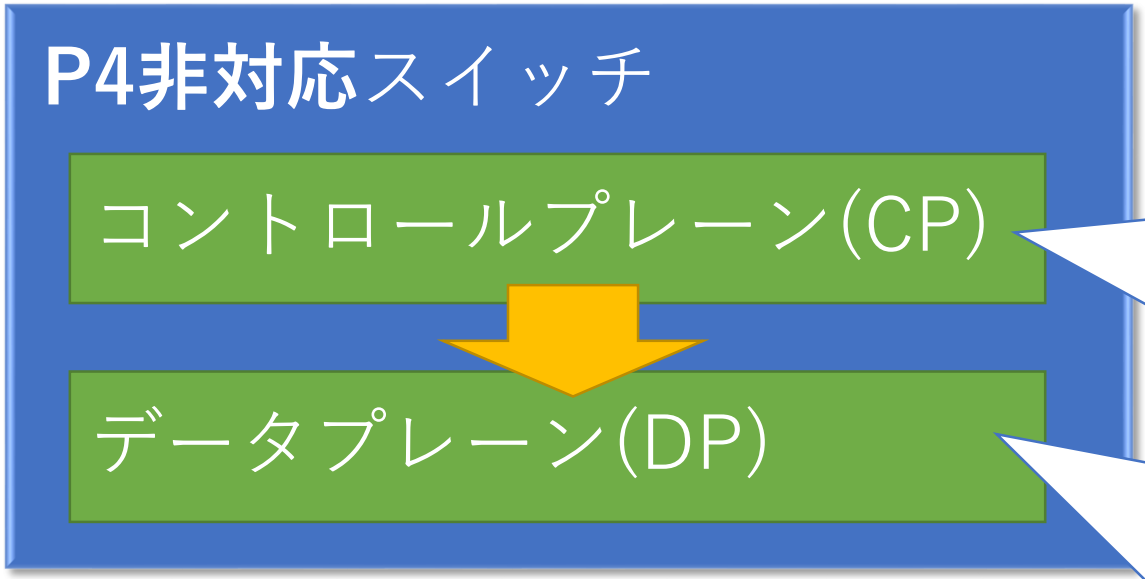
- P4 (Programming Protocol-Independent Packet Processors)

  - データプレーンをプログラミング可能な、プログラム言語

  - p4.orgには、大手のデータセンター/通信事業者、機器ベンダー、ASICベンダーがメンバーとして参加しており、大変注目されている

  →で、何が嬉しいの？

P4の無い世界



CLI、プロトコル等によりDPへ設定
・テーブルマッチ
・カプセル化、データ書き換え設定
→プログラミング可能

予めハードウェアに組み込まれ、特定のプロトコルに特化したテーブルマッチ、データ書き換え、カプセル化等 (例: IP/TCP/UDP/VLAN...)
→**プログラミング不可**

◆新しいプロトコルに対応した、データプレーンのテーブルマッチ、データ書き換えは実現できない(ハードウェア対応待ち)

P4のある世界

P4対応スイッチ

コントロールプレーン(CP)

P4対応データプレーン(DP)

CLI、プロトコル等によりDPへ設定
・テーブルマッチ
・カプセル化、データ書き換え設定
→プログラミング可能

ハードウェア上で**ユーザー定義による**テーブルマッチ、データ書き換え、カプセル化等を実現
→**プログラミング可能**

◆新しいプロトコルに対応した、データプレーンのテーブルマッチ、データ書き換え等を、既設のスイッチ上で実現可能

→ネットワーク運用の自由度、拡張性が飛躍的に向上する！？

P4ってどうなの？→使ってみました

どうやって
動かすの？

P4で何が
できるの？

プログラミング
は難しい？

実際に使ってみて
確かめました！

P4の特徴

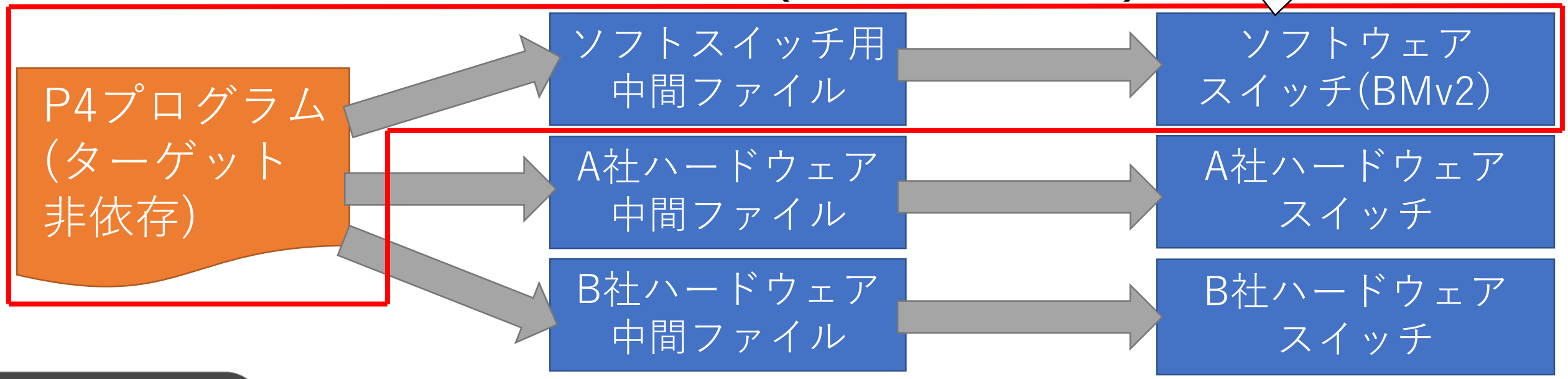
◆特徴

◇データプレーンを記述可能

- パケットマッチング、書き換え、ヘッダー追加/削除、カウンタ・・・
- 任意のフィールドに対し操作可能

◇ターゲットデバイス非依存

◇現時点で2種類のVersionが存在(P4_14とP4_16)

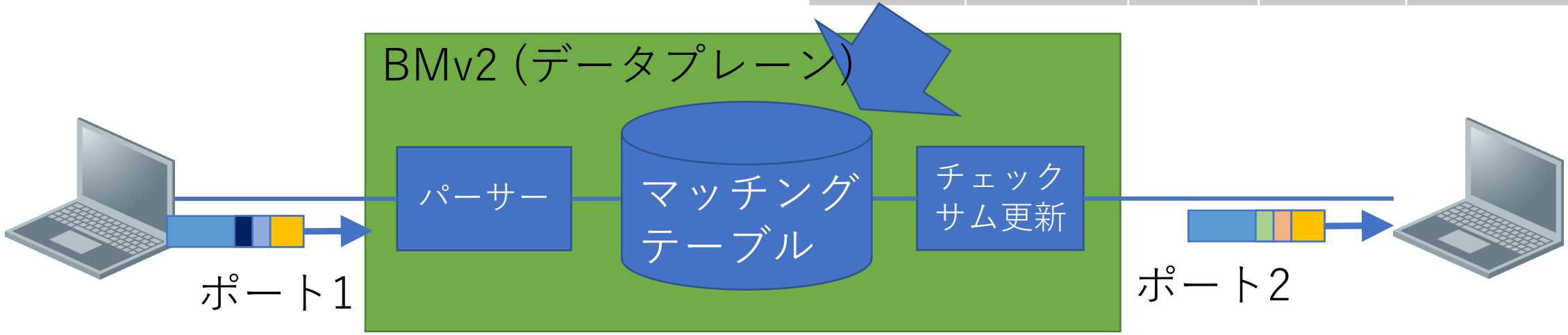


P4プログラミング例

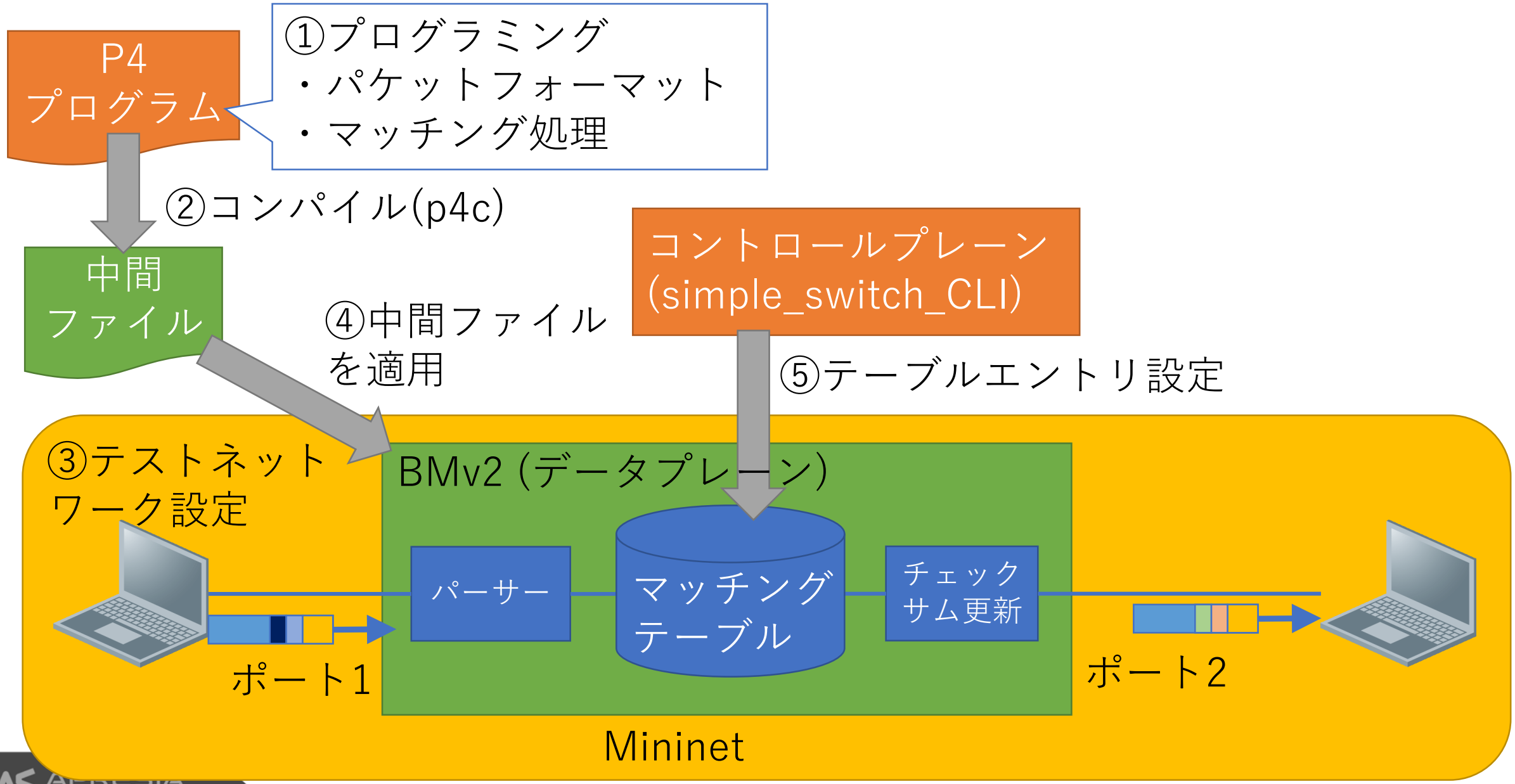
- VXLANのパケット変換
  - UDP送信ポート書き換え
  - VNID書き換え
  - 上記を書き換えた値でUDPチェックサム再計算

ユーザー定義のVXLANパケットマッチングテーブル

Match		Action		
UDP	VNID	UDP	VNID	PortNo
8472	100	4789	200	2
4789	200	8472	100	1



プログラミング、環境構築の手順



プログラミング概要

P4プログラムのブロック構成

```
/************************ H E A D E R S  ************************************
...

/************************ P A R S E R  ************************************
...

/**************  I N G R E S S   P R O C E S S I N G  ********************
...

/****************  E G R E S S   P R O C E S S I N G  ********************
...

/*************   C H E C K S U M   C O M P U T A T I O N  **********
...

/************************  D E P A R S E R  ****************************/
...
```

パケット
フォーマット

ヘッダー解析

パケット
マッチング処理

チェックサム更新

パケットフォーマットの定義(VXLANフレーム)

```
header ethernet_t {
    bit<48> dstMac;
    bit<48> srcMac;
    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;
}
```

```
header udp_t {
    bit<16>   src_port;
    bit<16>   dst_port;
    bit<16>   length;
    bit<16>   checksum;
}
header vxlan_t {
    bit<8>      flag;
    bit<24>     rsvd1;
    bit<24>     vnid;
    bit<8>      rsvd2;
}
struct metadata {
    bit<16>     pseudoLength;
}
struct headers {
    ethernet_t   ethernet;
    ipv4_t       ipv4;
    udp_t        udp;
    vxlan_t      vxlan;
}
```

フィールド毎に
ビット長と名前を定義
例：UDP受信ポート番号
を16ビット長とする

内部でのみ使用
(UDPチェックサム
計算のため)

ヘッダー解析の定義

```
...
    state parse_ipv4 {
        packet.extract(hdr.ipv4);
        meta.pseudoLength = hdr.ipv4.totalLen - 16w20; /*
For re-calculating UDP checksum */
        transition select(hdr.ipv4.protocol) {
            IP_PROTOCOL_UDP: parse_udp;
            default: accept;
        }
    }

    state parse_udp {
        packet.extract(hdr.udp);
        transition select(hdr.udp.dst_port) {
            UDP_PORT_VXLAN : parse_vxlan;
            UDP_PORT_OLD_VXLAN : parse_old_vxlan;
            default: accept;
        }
    }

    state parse_vxlan {
...
```

パケットからヘッダーデータを取り出す
例：IPv4ヘッダーを取り出す

ヘッダーの解析
例：IPv4プロトコル番号がUDP(17番)であった場合、UDPヘッダーのデータ取り出し処理へ移行する

パケットマッチング処理の定義

```
. . .
    action vnid_translate(l4Port_t dst_port,
vxlanVnid_t vnid, egressSpec_t port) {
        hdr.udp.dst_port = dst_port;
        hdr.vxlan.vnid = vnid;
        standard_metadata.egress_spec = port;
    }

    table vnid_translation_table {
        key = {
            hdr.udp.dst_port: exact;
            hdr.vxlan.vnid: exact;
        }
        actions = {
            vnid_translate;
            NoAction;
        }
        size = 1024;
        default_action = NoAction();
    }
    apply {
        if (hdr.vxlan.isValid()) {
            vnid_translation_table.apply();
        }
```

アクション動作の定義
例：ヘッダーの値をテーブルの □ 内にある変換後の値に置き換え

送信ポート番号の指定

Match		Action vnid_translate		
UDP	VNID	UDP	VNID	PortNo

テーブルマッチの定義
例：テーブルの □ 内にあるマッチ要素と、取りうるアクションの定義

チェックサム更新の定義

```
. . .
control ComputeChecksum(inout headers hdr, inout metadata
meta) {
    apply {
        update_checksum(hdr.ipv4.isValid(),
            { hdr.ipv4.version, hdr.ipv4.ihl,
hdr.ipv4.diffserv, hdr.ipv4.totalLen, hdr.ipv4.identification,
hdr.ipv4.flags, hdr.ipv4.fragOffset, hdr.ipv4.ttl,
hdr.ipv4.protocol, hdr.ipv4.srcAddr, hdr.ipv4.dstAddr },
hdr.ipv4.hdrChecksum,
            HashAlgorithm.csum16);
        update_checksum_with_payload(
            hdr.udp.isValid(),
            { hdr.ipv4.srcAddr, hdr.ipv4.dstAddr, 8w0,
hdr.ipv4.protocol, meta.pseudoLength, hdr.udp.src_port,
hdr.udp.dst_port, hdr.udp.length, 16w0 }, hdr.udp.checksum,
HashAlgorithm.csum16);
    }
}
```

IPv4チェックサム計算

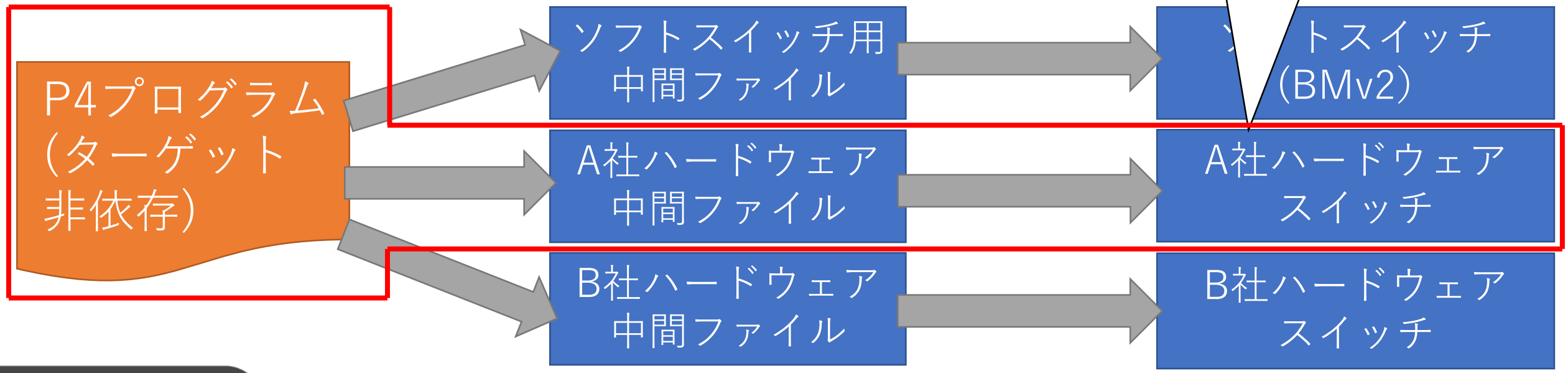
UDPチェックサム計算

デモ

実際の手順をデモの動画にてご覧いただきます

P4の特徴

- 特徴
  - データプレーンを記述可能
    - パケットマッチング、書き換え、ヘッダー追加/削除、カウンタ・・・
    - 任意のフィールドに対し操作可能
  - ターゲットデバイス非依存
  - 現時点で2種類のVersionが存在(P4_14とP4_16)

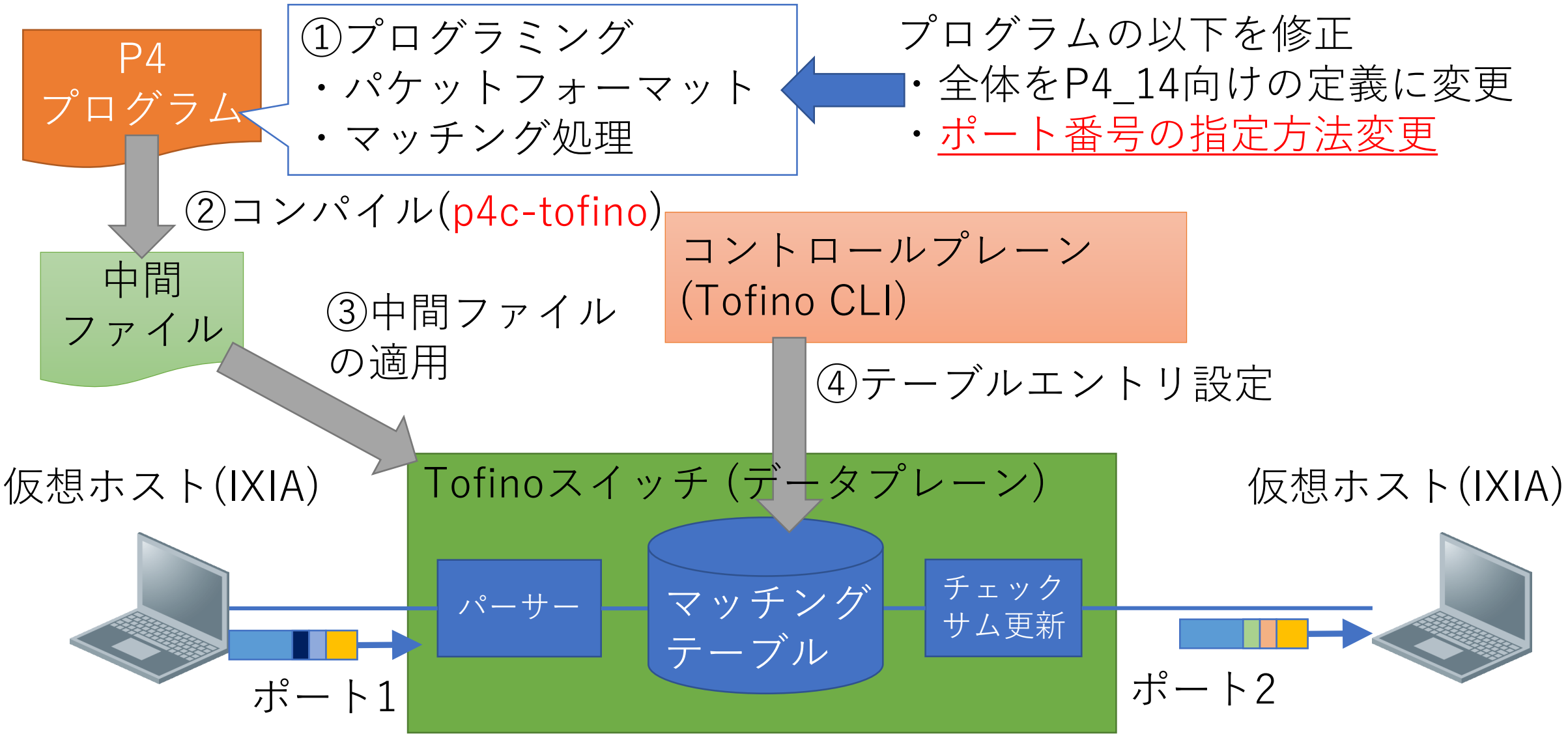


P4対応ハードウェアスイッチ上での確認

- Barefoot Tofino ASIC搭載スイッチ
  - ハードウェア上でプログラミングされたデータプレーンを実現可能
  - https://www.barefootnetworks.com/products/brief-tofino/
  - 現時点ではP4_16に対応しておらず、P4_14のみ対応
- ハードウェアスイッチでも、同様の動作が行えることを確認

※Barefoot Networks社提供

P4対応ハードスイッチ上での運用の流れ



わかったこと(1)

- P4のプログラミングは、簡単です！！
  - サンプルが多数あり、これを参考にカスタマイズも出来る
  - デバッグのための仕組みも充実している(BMv2)

- データプレーンを自由に変更できます！！
  - パケット内の任意のフィールドを書き換え可能
  - L3/L4チェックサムの再計算が可能

わかったこと(2)

- 「ターゲットデバイス非依存」ではない場合がある
  - ポート番号等、デバイスに関係する変数を使用する箇所は、プログラムの変更が必要となる場合がある

- コントロールプレーンの対応が必要
  - 定義したデータプレーンに対応したコントロールプレーンを用意しないといけない
  - コントロールプレーンからデータプレーンを制御する仕組みとしてP4Runtimeが用意されている

ご清聴ありがとうございました！

参考資料

APRESIA SYSTEMS

参考資料

- P4_16の言語仕様
  - https://p4.org/p4-spec/docs/P4-16-v1.0.0-spec.html

各種ソフトウェアの入手先(1)

◆P4関連

◇サンプルプログラム

- https://github.com/p4lang/tutorials
  - exercises ディレクトリ以下に、いくつかサンプルがある
  - こちらはチュートリアルになっているので、READMEの手順に従うことで、BMv2への設定、Mininetの準備から、動作確認まで一通り行うことが可能
- https://github.com/p4lang/switch/tree/master/p4src
  - サンプルコード多数あり

◇コンパイラ

- https://github.com/p4lang/p4c
- READMEに従いインストール

各種ソフトウェアの入手先(2)

- その他
  - ソフトスイッチ(BMv2)
    - https://github.com/p4lang/behavioral-model
    - READMEに従いインストール
  - Mininet
    - https://github.com/mininet/mininet
    - INSTALLに従いインストール