データプレーンプログラミング “P4” の次の一歩

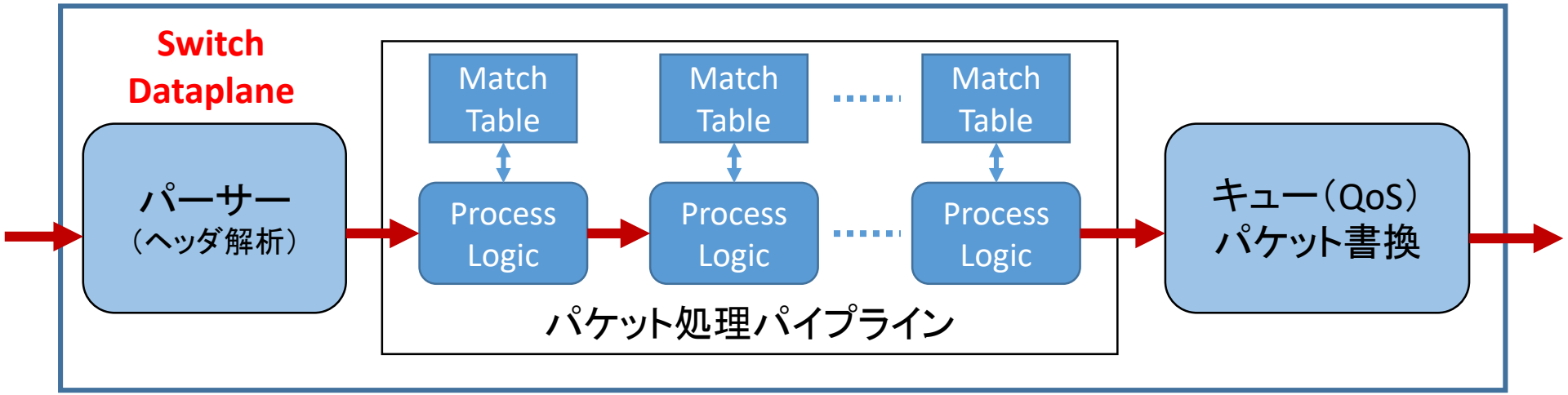
コントロールプレーンとコミュニティと

トヨタ自動車株式会社
コネクティッド先行開発部 InfoTech DCインフラG
プリンシパル・リサーチャー 海老澤 健太郎

P4とは？

データプレーンの挙動をプログラムできる言語

パーサー（ヘッダ解析）
マッチテーブル（レイアウト・メモリ配分）
アクション（処理内容）



データプレーンをプログラマブルにする目的

ユースケースに合わせたリソースの配分

１種類のハードウェアで
複数のユースケースに対応

スケーラビリティ向上（リソース適正配置）

汎用なACLマッチテーブル（数千ルール）
↓
特定フィールドのExact Match（～数百万ルール）

新しいアプリケーション・機能の実装

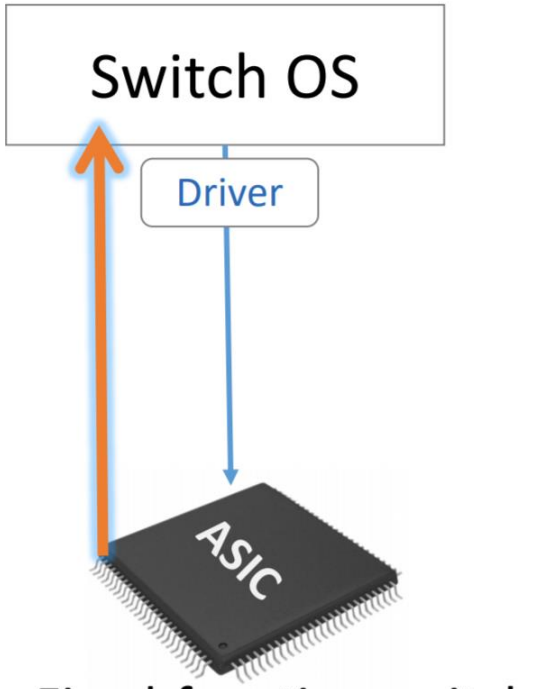
パケットへの情報の埋め込み
独自ヘッダの定義（ex: Telemetry, OAM）

新しいプロトコルのサポート（ex: VxLAN）
新しい領域への適応（ex: GTP, SFC）

ASIC改修には1.5～3年程度必要

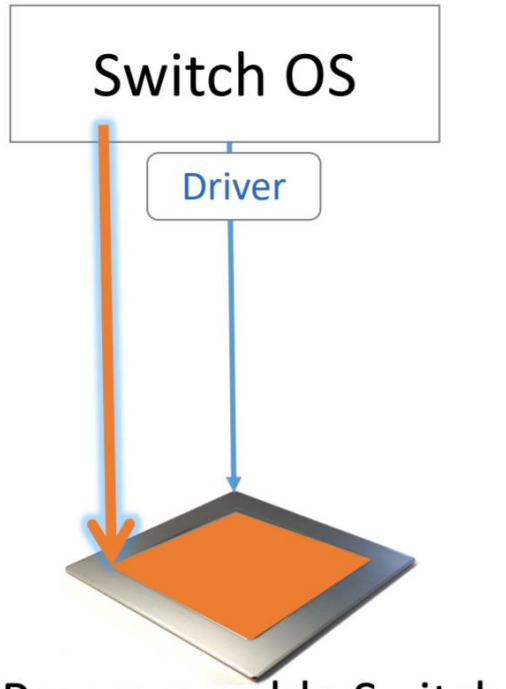
「ボトムアップ」から「トップダウン」へ

ASIC の機能 ⇒ スイッチが実現可能な機能



Fixed-function switch

スイッチで実現したい機能 ⇒ ASICの機能



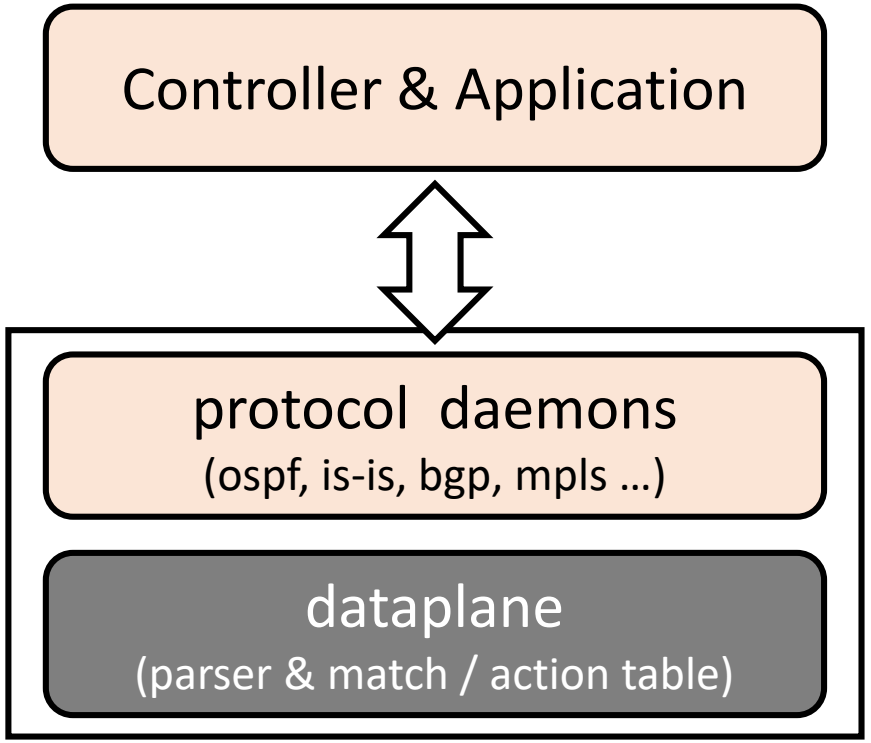
Programmable Switch

"How We Might Get Humans Out of the Way - Keynote by Nick McKeown", ONF Connect 2019
https://www.opennetworking.org/onf-connect-2019-resources/
https://www.opennetworking.org/wp-content/uploads/2019/09/Connect-2019-Nick-McKeown.pdf

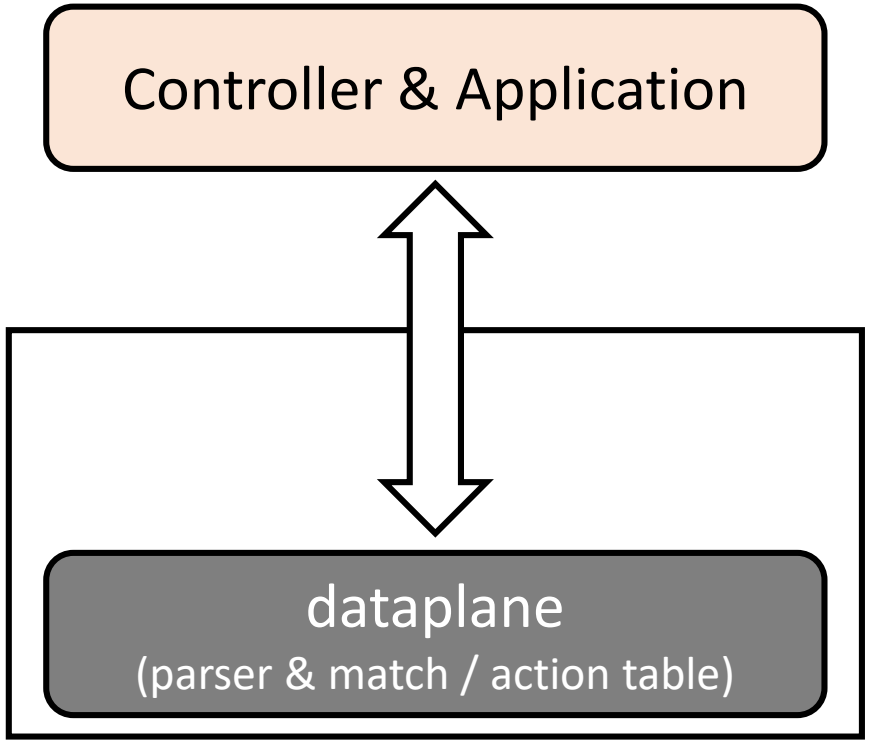
プログラマブルデータプレーンにおける コントロールプレーン

コントロールプレーンの実装場所

プロトコルデーモンを介した制御

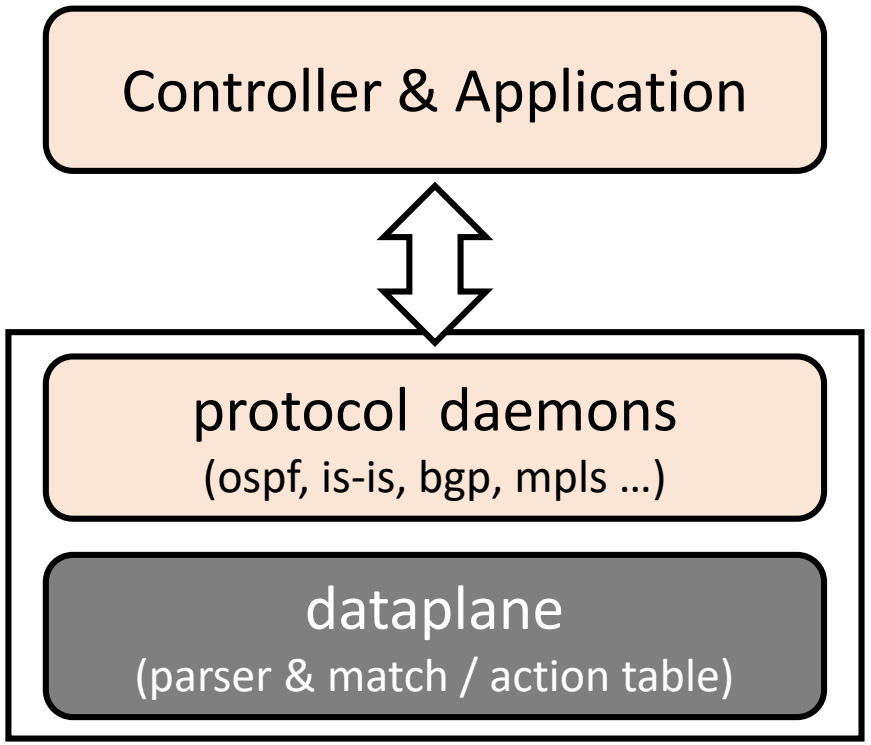


コントローラーからの直接制御



コントロールプレーンの実装場所

プロトコルデーモンを介した制御



良いところ

- コントローラーとの接続が切れても動作
- 各ノードの自律的な動作が可能
- コントローラーがシンプルに

苦労するところ

- 多機能＆安定なプロトコル実装
- データプレーンとのインテグレーション（開発）
- データプレーン変更へのプロトコル実装の追従
- データプレーン変更時のAPI変更

コントロールプレーンの実装場所

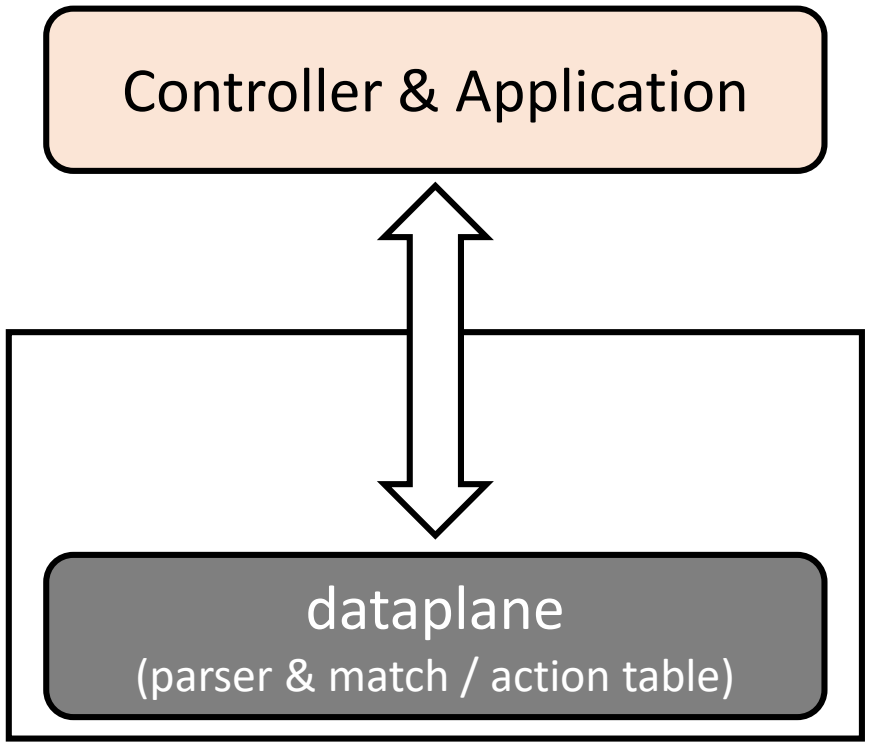
コントローラーからの直接制御

良いところ

- ユースケースに応じたコントロールプレーンとデータプレーン両方の変更が容易
- データプレーン変更時もAPIの変更は不要（データのみ変更）

苦労するところ

- コントローラーが複雑に（パス計算など）
- コントローラーとの接続断時の冗長化



“カスタマイズされたデータプレーン”における“コントロールプレーン”

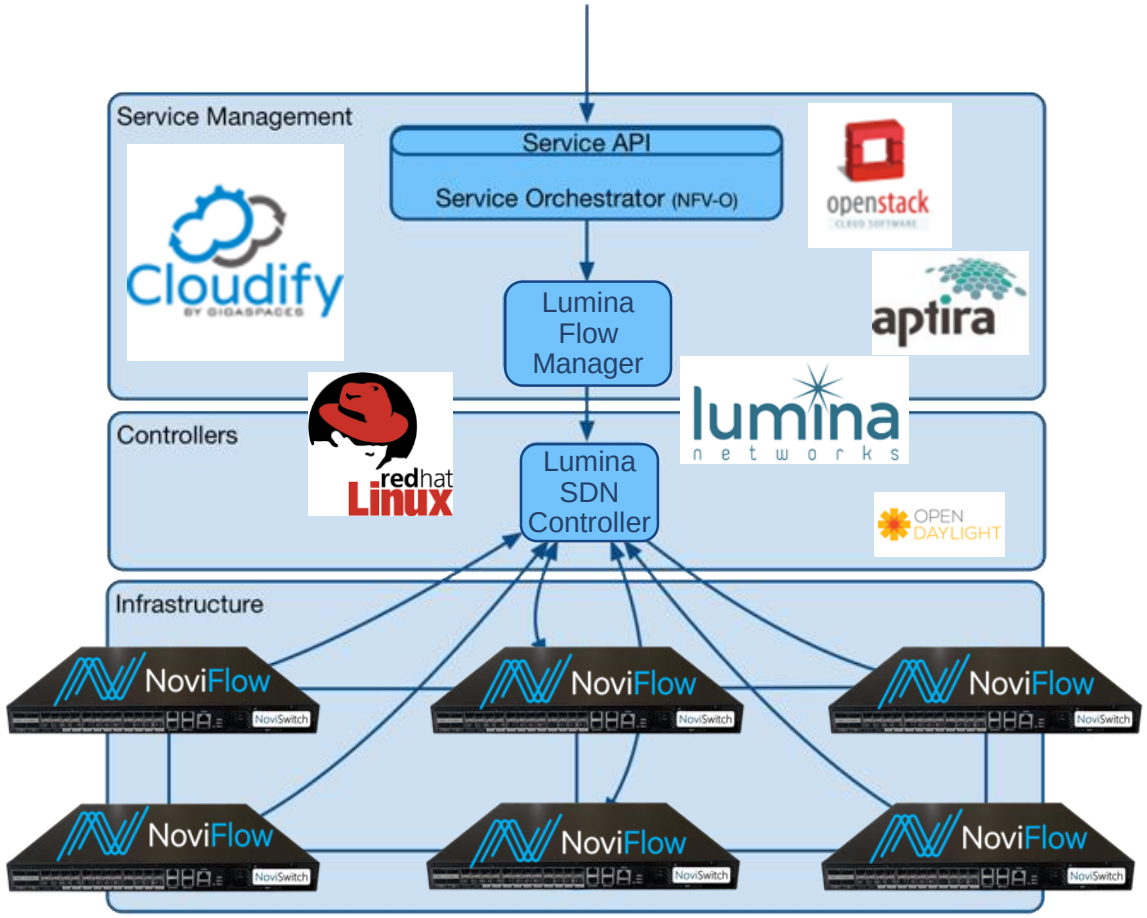
実装例

NoviFlow

https://noviflow.com/

WAN Edge & CORE (SD-CORE)

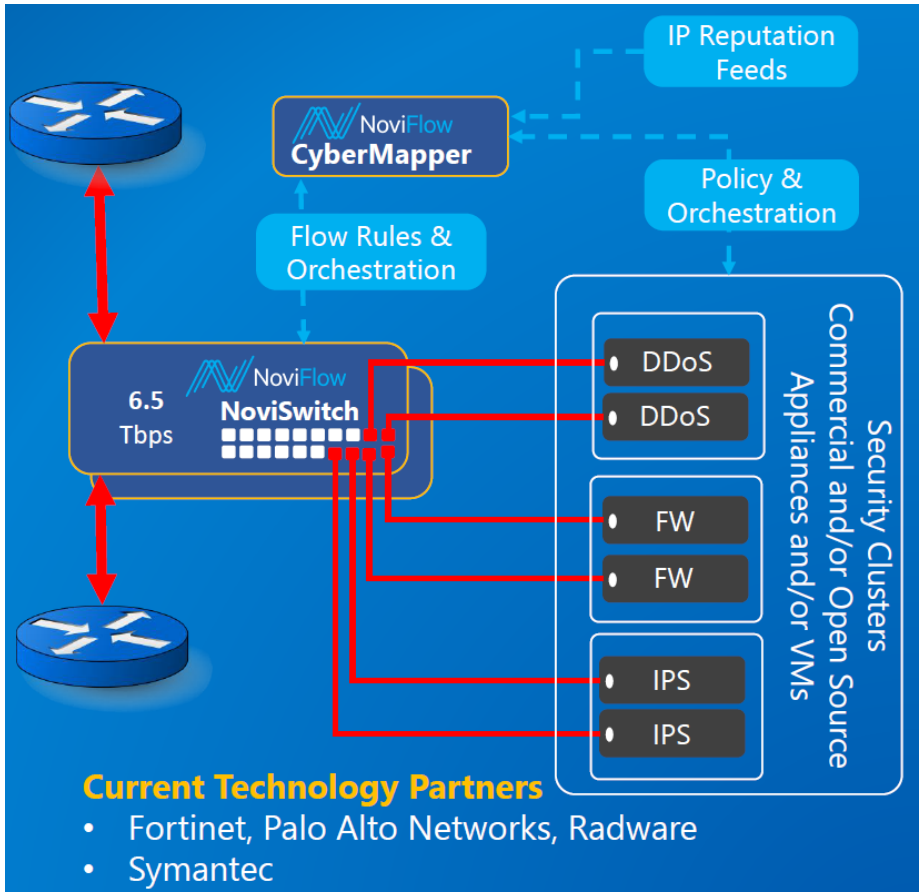
- MPLS VPN ベースのサービスを短期間・低コストで実現
  - MPLS forwarding
  - Segment Routing
  - Traffic Engineering
  - Streaming Telemetry
  - Service Automation（マルチドメイン）
- Tofino (P4) ベースの White Box Switch
- OpenDaylight（SDN Controller）



NoviFlow

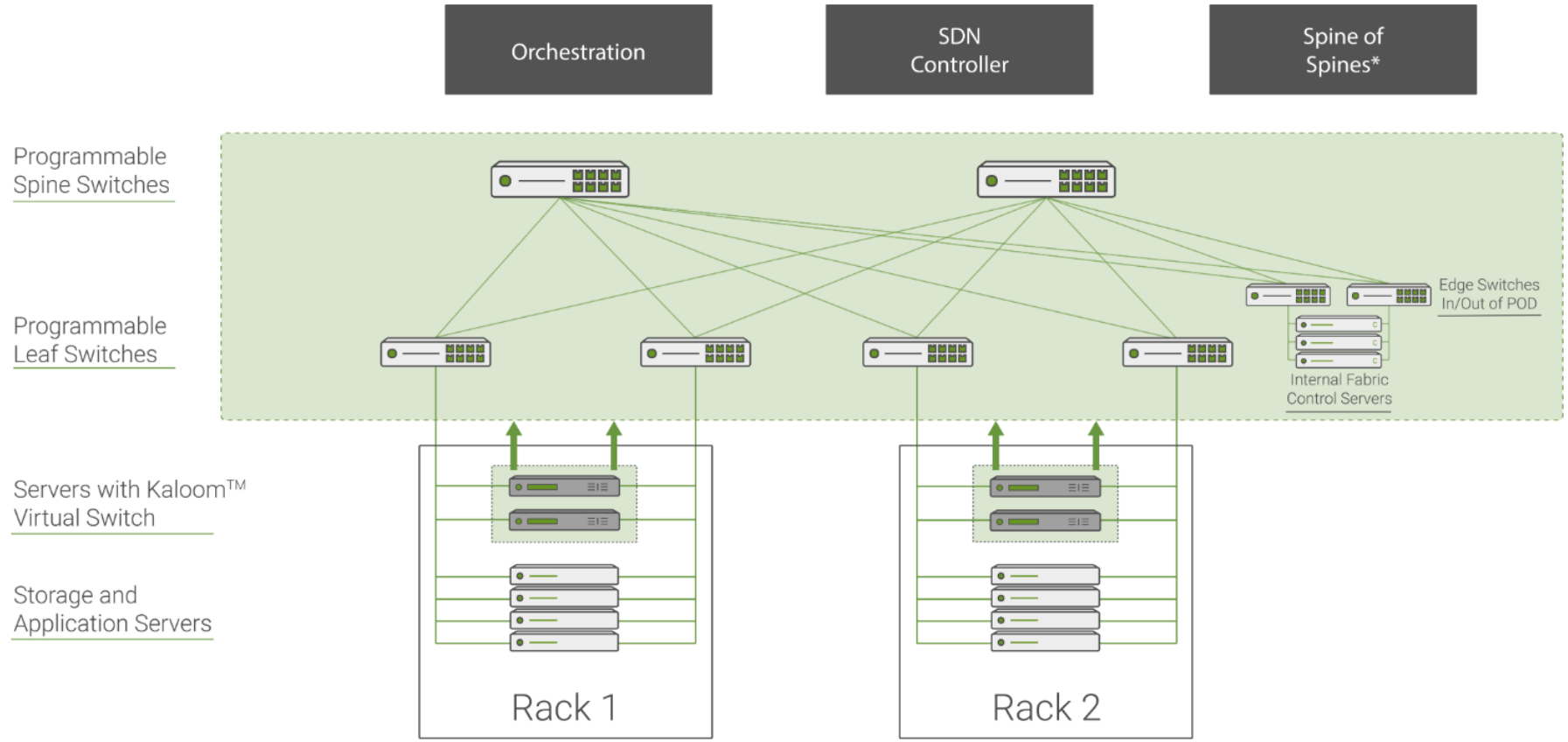
https://noviflow.com/

- ロードバランサー、パケットブローカー、INT（遅延測定）など
- コントローラー（CyberMapper）を通じ REST API で操作
- OpenFlow や P4 Runtime を用いたアクセスも可能



https://www.kaloom.com/

- Programmable Spine / Leaf Switch を利用した "Software Defined Fabric"
- サーバーの仮想スイッチと連携し ASIC へとオフロード
- 低遅延や拡張性を実現

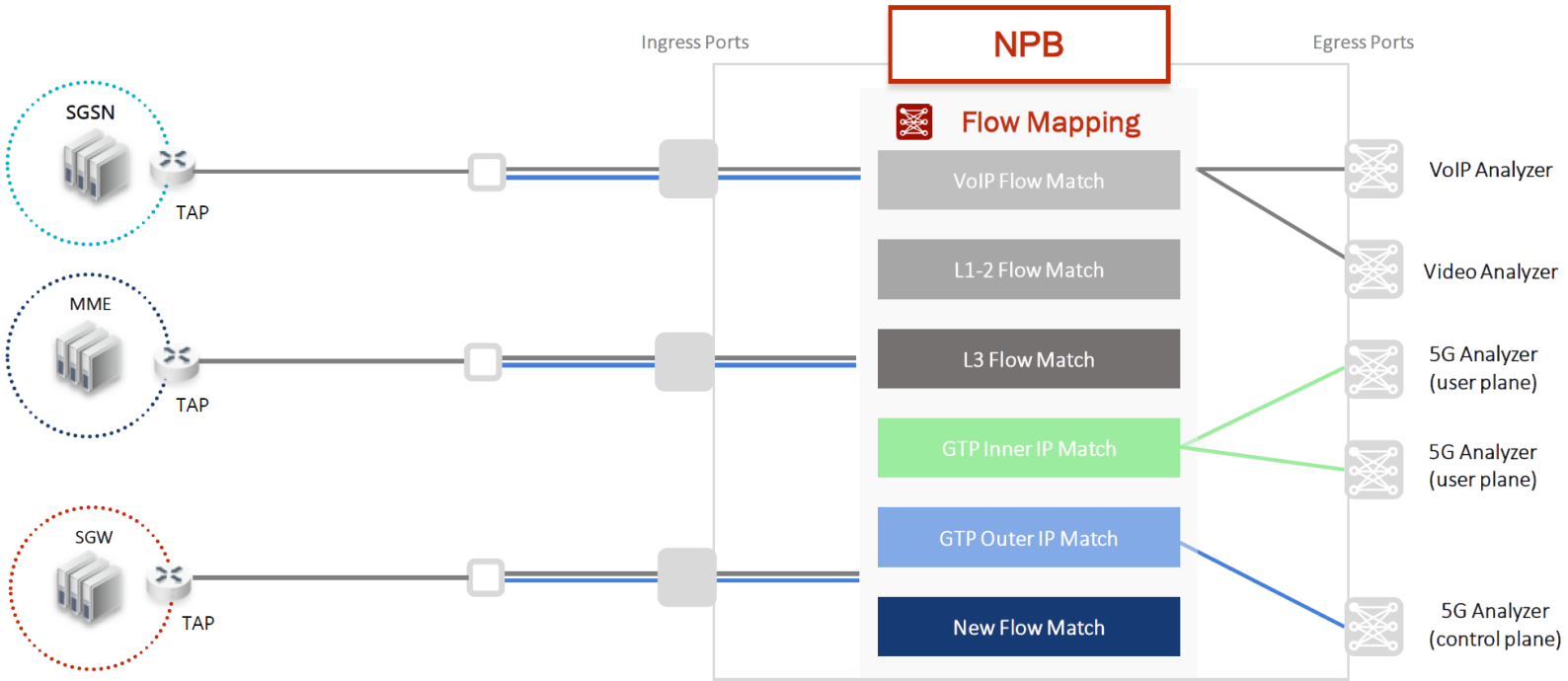


KulCloud

http://www.kulcloud.com/

"Using Programmable Chip and Open Source SW Toward Disaggregated Network Packet Broker and 5G UPF", P4 Workshop, May 1st, 2019

- "Prism Controller" を用い P4 で作ったデータプレーンを管理
- モバイルプロトコル（GTP）にも対応
- Network Packet Broker (NPB), 5G UPF (with N4 interface)



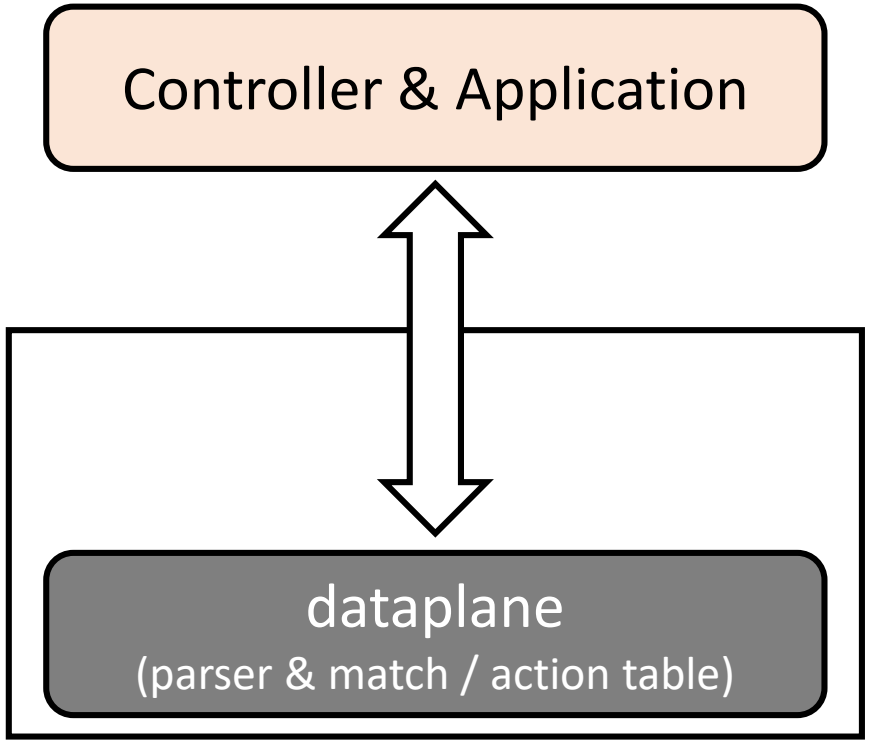
コントロールプレーンの実装場所

コントローラーによる独自データプレーン
直接制御によるユースケースの拡大

「データプレーンの変化に追従可能」な
プロトコル実装の不在

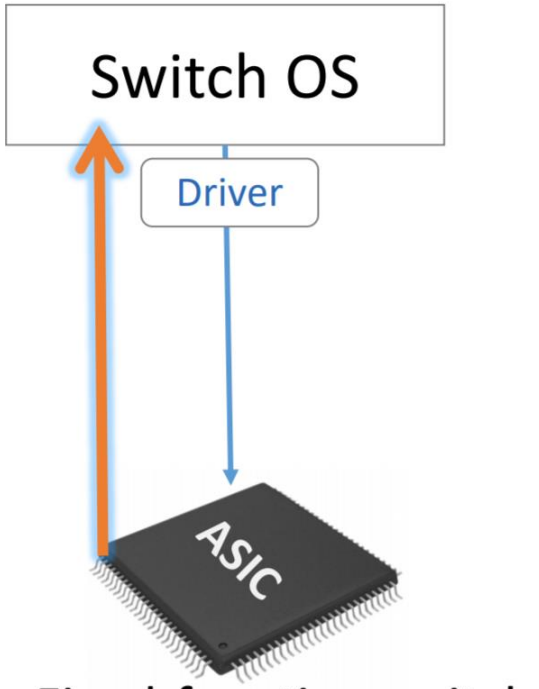
（少なくても現時点では）
コントローラーからのデータプレーン直接制御が
より多くの恩恵をもたらたす

コントローラーからの直接制御



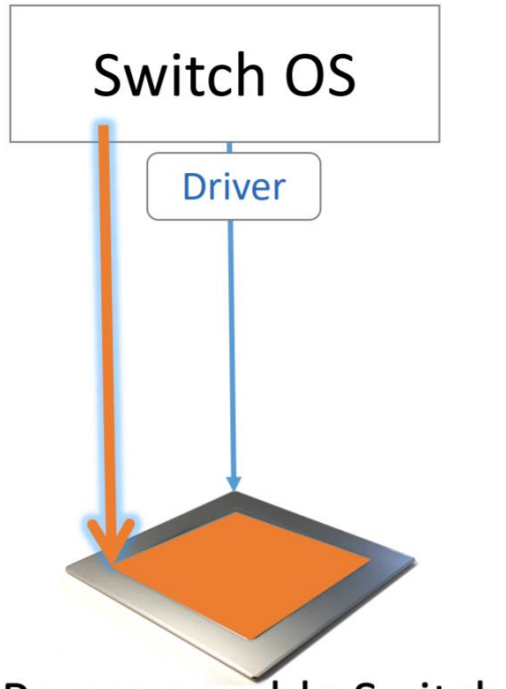
「ボトムアップ」から「トップダウン」へ

ASIC の機能 ⇒ スイッチが実現可能な機能



Fixed-function switch

スイッチで実現したい機能 ⇒ ASICの機能



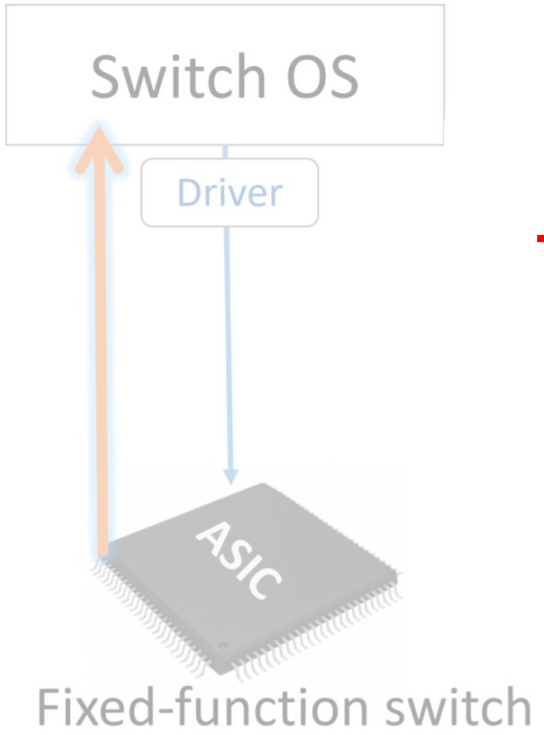
Programmable Switch

"How We Might Get Humans Out of the Way - Keynote by Nick McKeown", ONF Connect 2019
https://www.opennetworking.org/onf-connect-2019-resources/
https://www.opennetworking.org/wp-content/uploads/2019/09/Connect-2019-Nick-McKeown.pdf

「ボトムアップ」から「トップダウン」へ

ASIC の機能 ⇒ スイッチが実現可能な機能

スイッチで実現したい機能 ⇒ ASICの機能

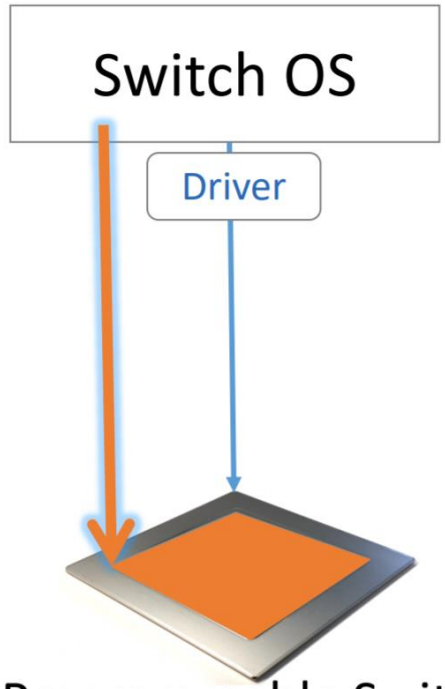


Fixed-function switch

実現可能な
ユースケースの数

⇓

トップダウンの力



Programmable Switch

"How We Might Get Humans Out of the Way - Keynote by Nick McKeown", ONF Connect 2019
https://www.opennetworking.org/onf-connect-2019-resources/
https://www.opennetworking.org/wp-content/uploads/2019/09/Connect-2019-Nick-McKeown.pdf

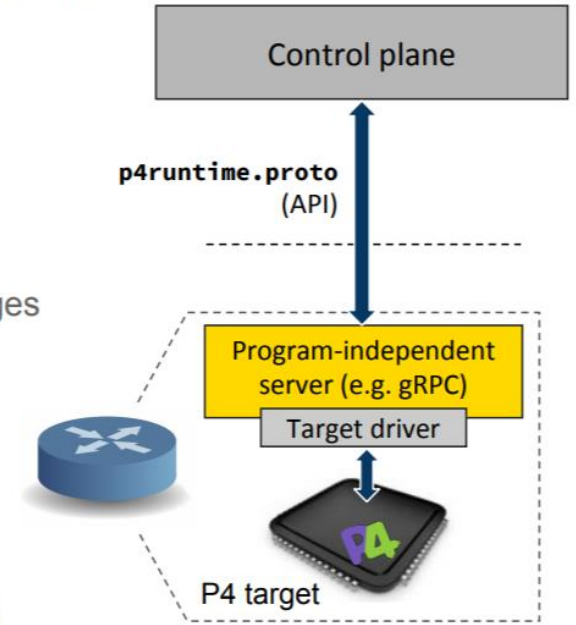
データプレーン ＆
コントロールプレーン を繋ぐ API

APIの標準化・コントロールプレーンとの連携

P4Runtime overview

19

- **Protocol for runtime control of P4-defined switches**
  - Designed around PSA architecture but can be extended to others
- **Work-in-progress by the p4.org API WG**
  - Initial contribution by Google and Barefoot
  - Draft of version 1.0 available: https://p4.org/p4-spec/
- **Protobuf-based API definition**
  - Automatically generate client/server code for many languages
  - gRPC transport
- **P4 program-independent**
  - API doesn't change with the P4 program
- **Enables field-reconfigurability**
  - Ability to push new P4 program, i.e. re-configure the switch pipeline, without recompiling the switch software stack



Slide courtesy P4.org

P4Runtime

Releases for P4Runtime

- v1.0.0 [HTML | PDF] (Jan 2019)
- *working draft*: [HTML | PDF]

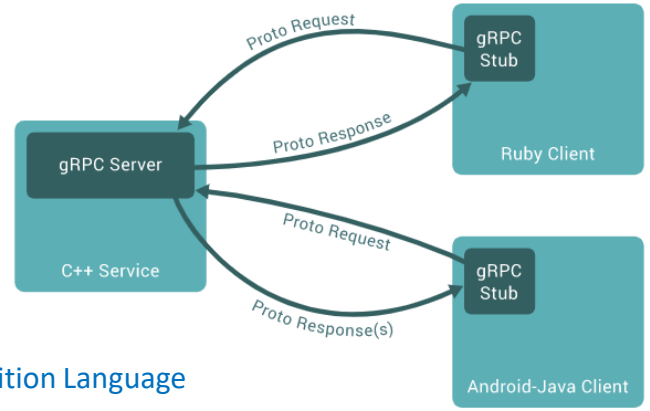
https://p4.org/specs/

ONS 2018, "Tutorial: P4 and P4Runtime Technical Introduction and Use Cases for Service Providers"
https://events.linuxfoundation.org/wp-content/uploads/2017/12/Tutorial-P4-and-P4Runtime-Technical-Introduction-and-Use-Cases-for-Service-Providers-Carmelo-Cascone-Open-Networking-Foundation.pdf

P4Tuntime == gRPC + protobuf

gRPC

- RPCを実現するプロトコル
- HTTP/2.0 や TLS をトランスポートに利用
- インターフェース定義言語（IDL）を用いて、サービスとメッセージを定義
- 各種プログラム言語のスタブコードを自動生成
- 同期＆非同期通信に対応（言語依存）



IDL: Interface Definition Language

protobuf

- gRPC デフォルトのIDLとして以下を定義
  - サービスインターフェース
  - メッセージ・ペイロードの構造
- 構造化データ ⇔ シリアライズ化されたバイナリ

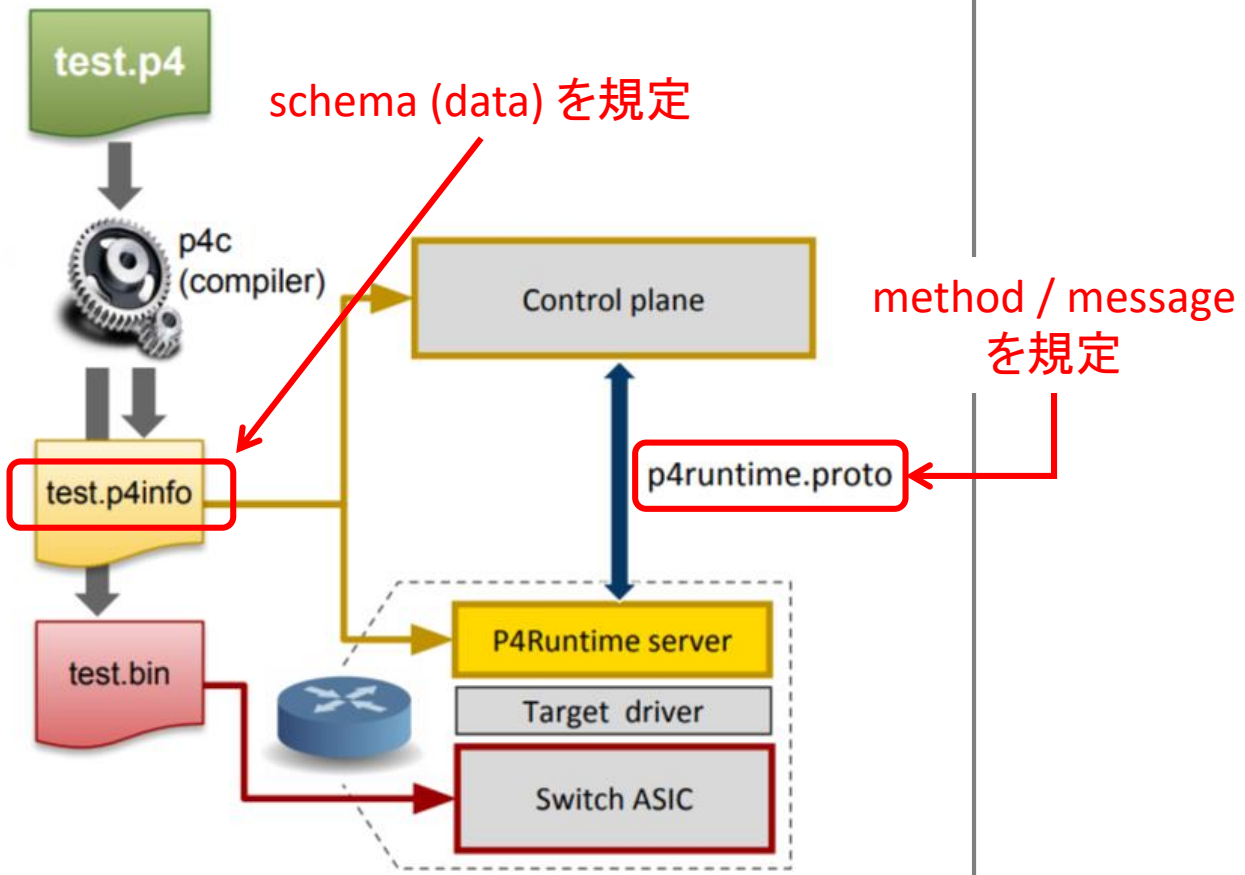
```
syntax = "proto2";

package tutorial;

message Person {
  required string name = 1;
  required int32 id = 2;
  optional string email = 3;

  enum PhoneType {
    MOBILE = 0;
    HOME = 1;
    WORK = 2;
  }

  message PhoneNumber {
    required string number = 1;
    optional PhoneType type = 2 [default = HOME];
  }

  repeated PhoneNumber phones = 4;
}

message AddressBook {
  repeated Person people = 1;
}
```

P4 compiler workflow

P4 compiler generates 2 files:

1. **Target-specific binaries**
   - Used to configure switch pipeline
     (e.g. binary config for ASIC, bitstream for FPGA, etc.)
2. **P4Info file**
   - Describes “schema” of pipeline for runtime control
   - Captures P4 program attributes
     - Tables, actions, parameters, etc.
   - Protobuf-based format
   - Target-independent compiler output
     - Same P4Info for SW switch, ASIC, etc.

Full P4Info protobuf specification:
https://github.com/p4lang/p4runtime/blob/master/proto/p4/config/v1/p4info.proto



ONS 2018, "Tutorial: P4 and P4Runtime Technical Introduction and Use Cases for Service Providers"
https://events.linuxfoundation.org/wp-content/uploads/2017/12/Tutorial-P4-and-P4Runtime-Technical-Introduction-and-Use-Cases-for-Service-Providers-Carmelo-Cascone-Open-Networking-Foundation.pdf

p4runtime.proto

https://github.com/p4lang/p4runtime/blob/master/proto/p4/v1/p4runtime.proto

P4Runtime "service" の "method" と "message" を定義

```
service P4Runtime {
  // Update one or more P4 entities on the target.
  rpc Write(WriteRequest) returns (WriteResponse) {
  }
  // Read one or more P4 entities from the target.
  rpc Read(ReadRequest) returns (stream ReadResponse) {
  }

  // Sets the P4 forwarding-pipeline config.
  rpc SetForwardingPipelineConfig(SetForwardingPipelineC
      returns (SetForwardingPipelineConfigResponse) {
  }
  // Gets the current P4 forwarding-pipeline config.
  rpc GetForwardingPipelineConfig(GetForwardingPipelineC
      returns (GetForwardingPipelineConfigResponse) {
  }
```

```
  // Represents the bidirectional stream between the controller and the
  // switch (initiated by the controller), and is managed for the following
  // purposes:
  // - connection initiation through master arbitration
  // - indicating switch session liveness: the session is live when switch
  //   sends a positive master arbitration update to the controller, and is
  //   considered dead when either the stream breaks or the switch sends a
  //   negative update for master arbitration
  // - the controller sending/receiving packets to/from the switch
  // - streaming of notifications from the switch
  rpc StreamChannel(stream StreamMessageRequest)
      returns (stream StreamMessageResponse) {
  }
}
```

p4runtime.proto

https://github.com/p4lang/p4runtime/blob/master/proto/p4/v1/p4runtime.proto

P4Runtime "service" の "method" と "message" を定義

```
message WriteRequest {
message WriteResponse {
message ReadRequest {
message ReadResponse {
message Update {
message Entity {
message ExternEntry {
message TableEntry {
message FieldMatch {
message TableAction {
message Action {
message ActionProfileActionSet {
message ActionProfileAction {
message ActionProfileMember {
message ActionProfileGroup {
message Index {
message MeterEntry {
```

```
message DirectMeterEntry {
message MeterConfig {
message CounterEntry {
message DirectCounterEntry {
message CounterData {
message PacketReplicationEngineEntry {
message Replica {
message MulticastGroupEntry {
message CloneSessionEntry {
message ValueSetMember {
message ValueSetEntry {
message RegisterEntry {
message DigestEntry {
message StreamMessageRequest {
message PacketOut {
message DigestListAck {
message StreamMessageResponse {
```

```
message PacketIn {
message DigestList {
message PacketMetadata {
message MasterArbitrationUpdate {
message Role {
message IdleTimeoutNotification {
message StreamError {
message PacketOutError {
message DigestListAckError {
message StreamOtherError {
message Uint128 {
message SetForwardingPipelineConfigRequest {
message SetForwardingPipelineConfigResponse {
message ForwardingPipelineConfig {
message GetForwardingPipelineConfigRequest {
message GetForwardingPipelineConfigResponse {
message Error {
```

P4Info example

basic_router.p4

```
...

action ipv4_forward(bit<48> dstAddr,
                    bit<9> port) {
   /* Action implementation */
}

...

table ipv4_lpm {
   key = {
       hdr.ipv4.dstAddr: lpm;
   }
   actions = {
       ipv4_forward;
       ...
   }
   ...
}
```

P4 compiler

basic_router.p4info

```
actions {
  id: 16786453
  name: "ipv4_forward"
  params {
    id: 1
    name: "dstAddr"
    bitwidth: 48
    ...
    id: 2
    name: "port"
    bitwidth: 9
  }
}
...
tables {
  id: 33581985
  name: "ipv4_lpm"
  match_fields {
    id: 1
    name: "hdr.ipv4.dstAddr"
    bitwidth: 32
    match_type: LPM
  }
  action_ref_id: 16786453
}
```

Slide courtesy P4.org

P4Runtime table entry example

basic_router.p4

```
action ipv4_forward(bit<48> dstAddr,
                    bit<9>  port) {
   /* Action implementation */
}
table ipv4_lpm {
   key = {
       hdr.ipv4.dstAddr: lpm;
   }
   actions = {
       ipv4_forward;
       ...
   }
   ...
}
```

Control plane generates

Logical view of table entry

```
hdr.ipv4.dstAddr=10.0.1.1/32
      -> ipv4_forward(00:00:00:00:00:10, 7)
```

Protobuf message

```
table_entry {
  table_id: 33581985
  match {
    field_id: 1
    lpm {
      value: "\n\000\001\001"
      prefix_len: 32
    }
  }
  action {
    action_id: 16786453
    params {
      param_id: 1
      value: "\000\000\000\000\000\n"
    }
    params {
      param_id: 2
      value: "\000\007"
    }
  }
}
```

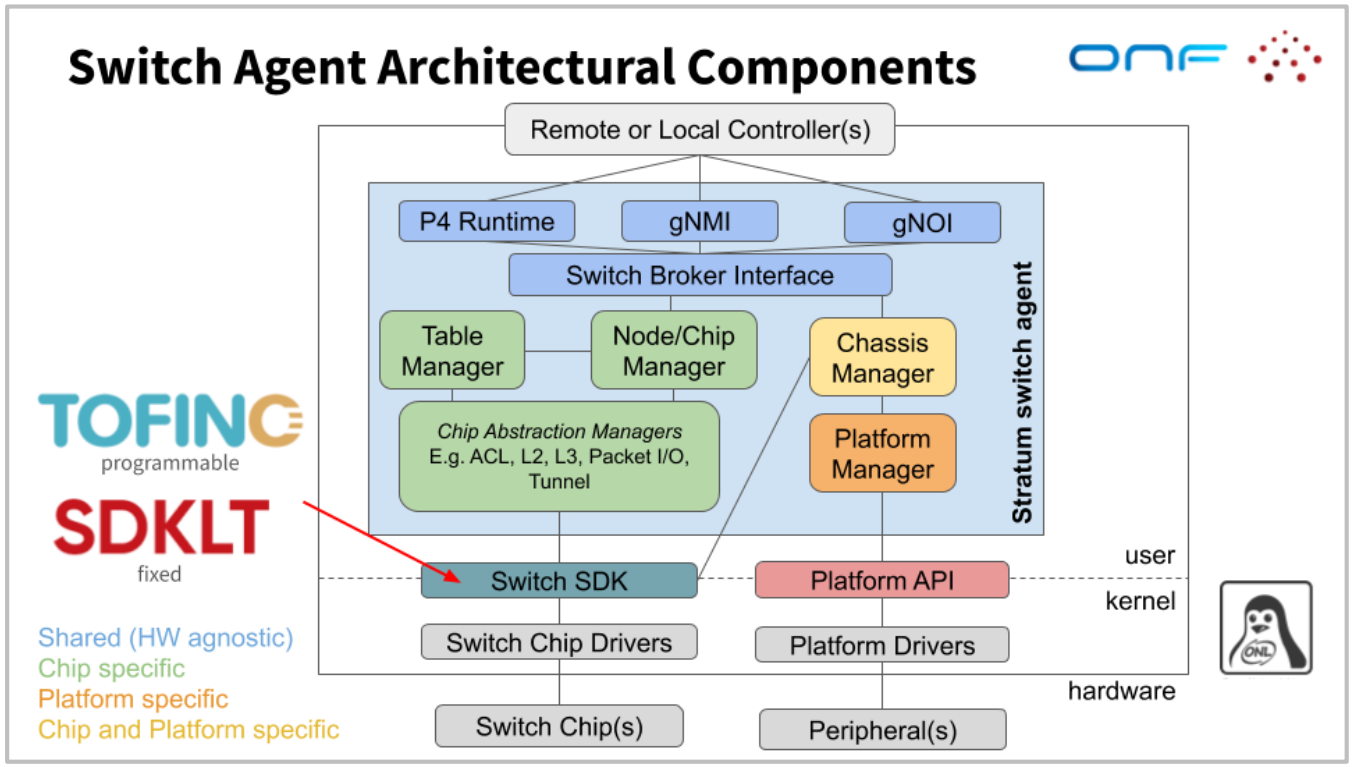
ONS 2018, "Tutorial: P4 and P4Runtime Technical Introduction and Use Cases for Service Providers"
https://events.linuxfoundation.org/wp-content/uploads/2017/12/Tutorial-P4-and-P4Runtime-Technical-Introduction-and-Use-Cases-for-Service-Providers-Carmelo-Cascone-Open-Networking-Foundation.pdf

	パイプラインのデータ構造（Table, Match Field, Action）
P4Runtime	ユーザーによる拡張可能 • API仕様では未定義（method/message と schema定義の分離） • P4から自動生成し読み込み
OpenFlow	ユーザーによる拡張は部分的に可能 • API仕様で定義済み • ベンダー独自フィールドの利用により Match Field の拡張が可能
SAI (Switch Abstraction Interface)	ユーザーによる拡張は不可 • API仕様で定義済み • Layer 2/3 スイッチに特化したパイプライン

独自機能（*1）を、デファクト標準なAPIを使ってコントロールできる
（*1）パケットフォーマット＆パイプライン

P4 で記述できないスイッチ機能へのアクセス

P4の記述と同期して動作する必要がある設定・統計の管理（Port ID, Stats etc.）



Open Sourced on 2019年9月10日

https://github.com/stratum/stratum

README.md

STRATUM

Stratum - Enabling the era of next-generation SDN

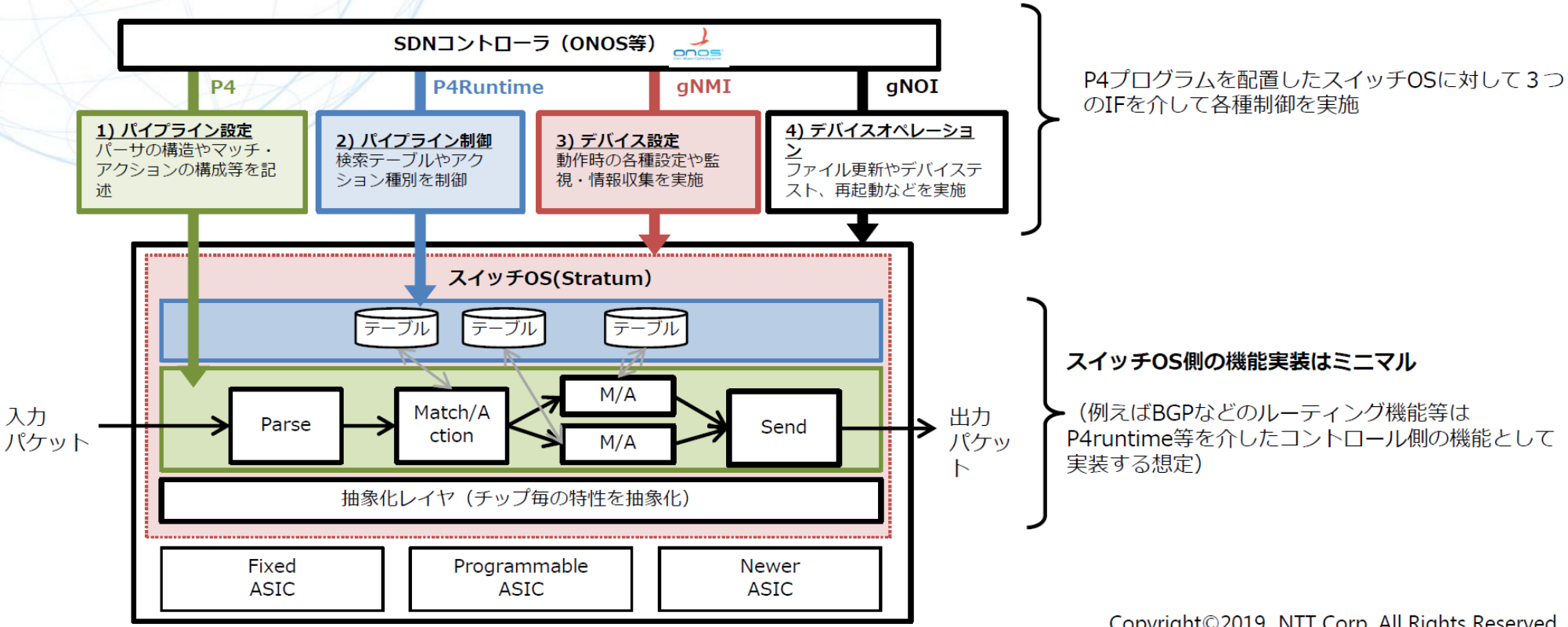
Copyright 2018 Google LLC
Copyright 2018-present Open Networking Foundation

Stratum is an open source silicon-independent switch operating system for software defined networks. It is building an open, minimal production-ready distribution for white box switches. Stratum exposes a set of next-generation SDN interfaces including P4Runtime and OpenConfig, enabling interchangeability of forwarding devices and programmability of forwarding behaviors. Current support includes Barefoot Tofino and Broadcom Tomahawk devices, as well as the bmv2 software switch.

P4Runtime provides a flexible mechanism for configuring the forwarding pipeline on a network switch.
gNMI is a framework for network device management that uses gRPC as the transport mechanism.

Stratum (2/3)

- SDN集中制御を前提とした、ミニマルな機能実装を指向したスイッチOSである。
- 異なるベンダチップを搭載したスイッチであっても統一的な手段で制御するために P4Runtime・gNMI・gNOIと呼ぶインタフェースを規定



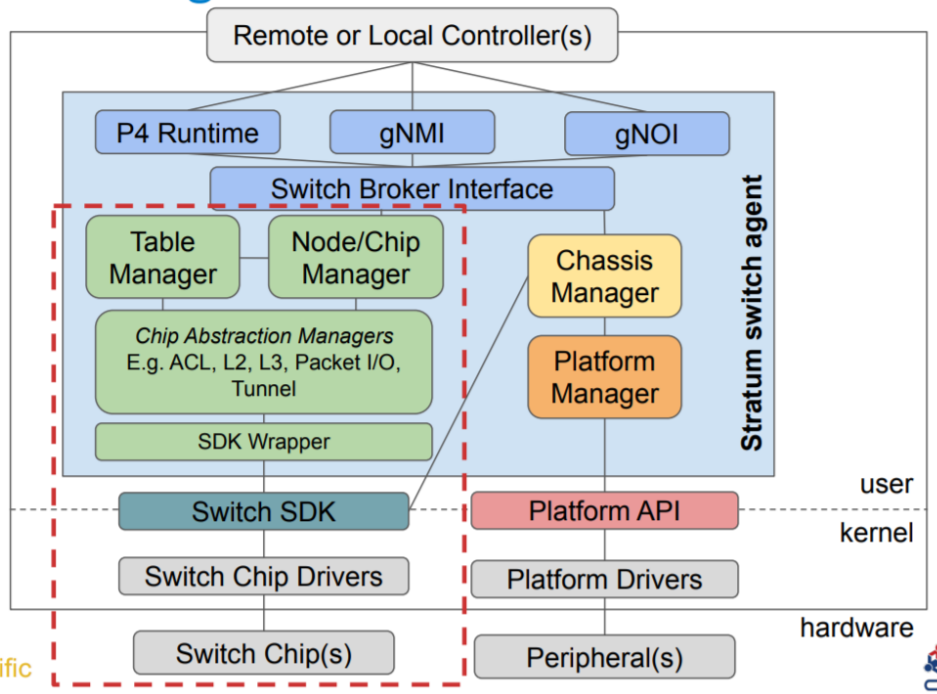
日本 P4 ユーザ会 2019 https://p4users.org/2019/07/16/event2019/
“キャリアにおける P4 ユースケースの紹介” NTTネットワークサービスシステム研究所 武井 勇樹

P4 on Fixed Function Switch ASIC

P4 program as an unambiguous *contract* describing the complete network behaviour in *machine-readable* format.

機械可読（API自動生成可能）なネットワーク挙動を記述する契約としてP4を利用

Enabling P4 on BCM Tomahawk

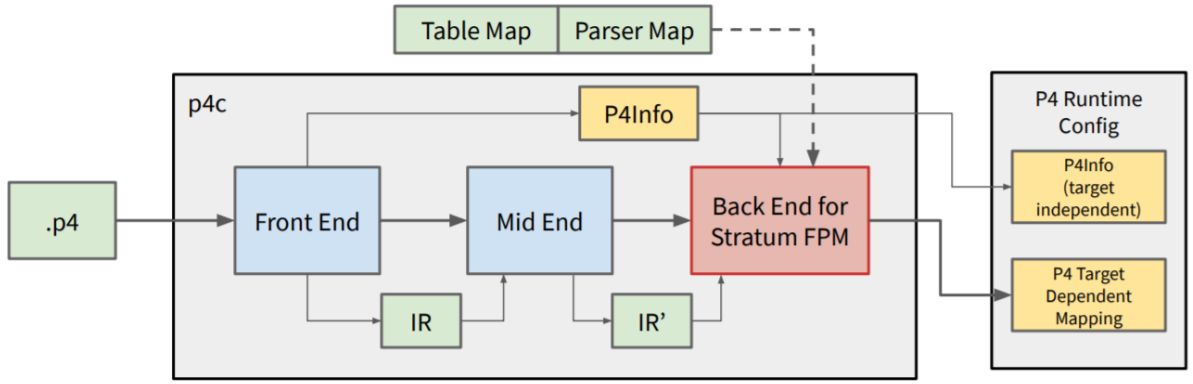


TOFINO programmable

BROADCOM fixed

Shared (HW agnostic)
Chip specific
Platform specific
Chip and Platform specific

P4C FPM Compiler



SDKLT is used to program fixed-pipeline switches using the Tomahawk chip from Broadcom.

ONF Connect 2019 "P4 on FPM Switches with Stratum"
https://www.opennetworking.org/wp-content/uploads/2019/09/3.30pm-Max-Pudelko-Stratum-FPM-Compiler.pdf

APRESIA SYSTEMS Technical Blog　　ソリューション

APRESIA Technical Blog > オープンネットワーク > Stratum on BMv2をGNS3上で動かしてみた

Stratum on BMv2をGNS3上で動かしてみた

Mr.APRESIA

2019/10/07 12:00

オープンネットワーク

はじめに

https://www.apresiatac.jp/blog/201910071861/

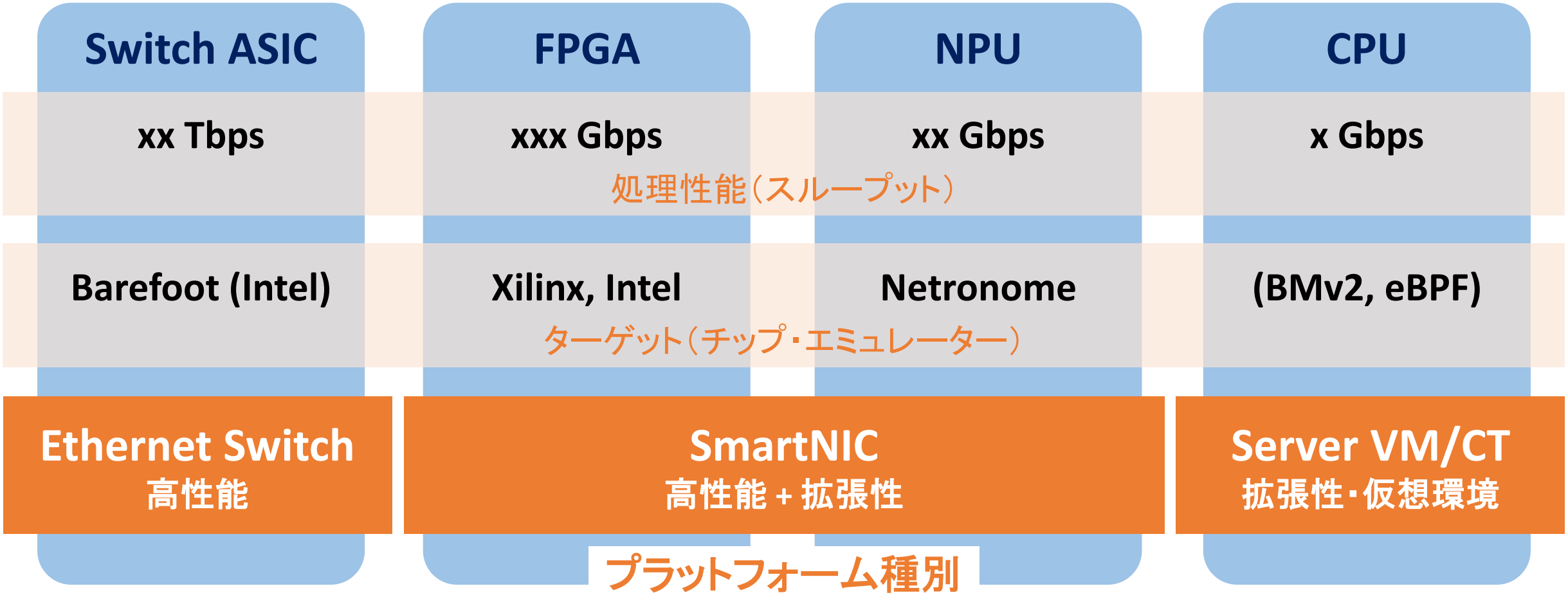
Open Network Foundation (ONF)は、以前より Google と共同で次世代型の SDN インターフェースと呼ばれている「P4Runtime」「gNMI」「gNOI」(いずれも gRPC で通信)をベースにした新たな Network OS「Stratum」の開発を進めていました。先日、このプロジェクトが OSS として Github に公開 されましたので、早速触っていきたいと思います。
今回は、ONF Connect '19 にて開催されていたチュートリアルをベースに進めていきたいと思います。チュートリアルでは Stratum の環境を Mininet で構築しており、Mininet はとても便利ですが構成が見えづらいため今回は GNS3 で構築してみました。

ターゲットとプラットフォーム

- P4 で記述したプログラムが動作する環境（チップ・エミュレーター）
- P4を利用できるプラットフォーム

ターゲット（P4 Target）

P4 で記述したプログラムが動作する環境（チップ・エミュレーター）



ターゲット（P4 Target）

P4 で記述したプログラムが動作する環境（チップ・エミュレーター）

Switch ASIC	FPGA	NPU	CPU
xx Tbps	xxx Gbps	xx Gbps	x Gbps
	処理性能（スループット）		
Barefoot (Intel)	Xilinx, Intel	Netronome	(BMv2, eBPF)
	ターゲット（チップ・エミュレーター）		
P4 Studio	P4-SDNet (Xilinx) Netcope NP4	Agilio P4C SDK	p4lang/p4c
	コンパイラ		

P4 on ODM Ethernet Switch (Tofino)

様々なハードウェア構成のスイッチ

Original Device Manufacturer (ODM)

Barefoot Networks is pleased to work with leading ODM whitebox and britebox vendors to further disaggregate networking by decoupling datapath intelligence from switch silicon. Please select one of the vendors to learn more:

Edge-corE NETWORKS

Interface Masters TECHNOLOGIES Innovative Network Solutions

Inventec

STORDIS The Open Networking Expert

WNC Wistron NeWeb Corp.

https://barefootnetworks.com/partners/

Edge-corE NETWORKS

Wedge100BF-32X/65X

32/65 x QSFP28 ports

CPU Modules
Intel x86 Broadwell-DE
Pentium D-1517

Memory (RAM)
4/8/16 GB SO-DIMM DDR4

Storage
32 GB M.2 SSD

注:参考情報です。現在の仕様は
各メーカーにお問い合わせ下さい

Interface Masters
TECHNOLOGIES
Innovative Network Solutions

Tahoe 2624

24 ports of 100G/40G QSFP28
& 20 ports of 25G/10G SFP 28

Two Rear-Facing I/Os Supporting
Powerful XEON-D Offload I/O
Xilinx Virtex UltraScale FPGA I/O

Tahoe 2860

32 ports of QSFP28

Control plane processor options
x86 and Power PC
Data plane processors options
MIPS and Power PC

Edge-Core: https://www.edge-core.com/productsInfo.php?cls=1&cls2=180&cls3=181&id=335 | Interface Masters: https://interfacemasters.com/products/switch-appliances/

STORDIS BF2556X-1T

Networks Ports

48x25G + 8x100G in 1RU Chassis
Port 1 – Port 16: Support 1/10/25GbE
Port 17 – Port 48: Support 10/25GbE

ASIC

Barefoot Tofino 2.0Tbit

CPU & Core

Broadwell-DE 8-core @2.0GHz
32G DDR4
128G SSD

Timesync option

1588v2 PTP Time Synchronization

STORDIS BF6064X-T

Networks Ports

64x 40/50/100GbE in 2RU Chassis
256x 10/25GbE via breakout
128x 50GbE via breakout

ASIC

Barefoot Tofino 6.4Tbit

CPU & Core

8-core x86 CPU
32G DDR4
128G SSD

Timesync option

1588v2 PTP Time Synchronization

注:参考情報です。現在の仕様は
各メーカーにお問い合わせ下さい

STORDIS: https://www.stordis.com/products/stordis-bf6064x-t/

P4 on Fixed Function Switch ASIC

Stratum Switch Support Today

Switching ASIC \ Switch Vendor	DELL	DELTA	Edge-corE NETWORKS	Inventec	QCT	STORDIS
BAREFOOT NETWORKS Tofino *Up to 6.5 Tbps*		**AG9064v1** *64 x 100 Gbps*	**Wedge100BF-32X** *32 x 100 Gbps* **Wedge100BF-65X** *65 x 100 Gbps*	**D5054** *6 x 100 Gbps + 48 x 25 Gbps*		**BF6064X** *64 x 100 Gbps*
BROADCOM Tomahawk *Up to 3.2 Tbps*	**Z9100** *32 x 100 Gbps*		**AS7712** *32 x 100 Gbps*	**D7032** *32 x 100 Gbps*	**T7032-IX1** *32 x 100 Gbps*	

\+ 2 software switches: **bmv2** (functional software switch) & **dummy switch** (used for API testing)

Near-term future platforms:

- Additional platforms for existing targets
  - Existing vendors + Asterfusion, ...
- Mellanox SN2700 (Spectrum)
- Datacom platforms (PowerPC-based)

Asterfusion Mellanox TECHNOLOGIES DATACOM

ONF Connect 2019, "Stratum Overview & Update"
https://www.opennetworking.org/wp-content/uploads/2019/09/2.00pm-Brian-OConnor-Stratum.pdf

P4 on SmartNIC (FPGA/NPU)

NETCOPE TECHNOLOGIES

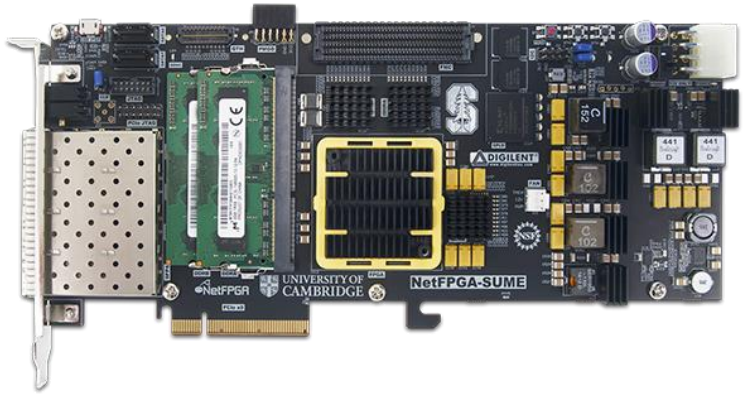
INTEL MAP

2x40G card by Intel. This SmartNIC features Arria 10 FPGA connected as a "bump in the wire".

NFB-200G2QL

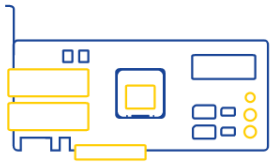
2x100G card by Netcope. This datacenter-ready card offers guaranteed 200 Gbps packet processing.

NetFPGA-SUME Virtex-7 FPGA Development Board

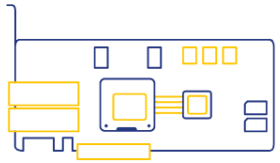


NETRONOME

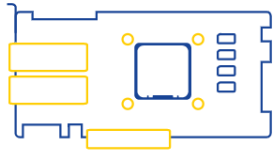
Agilio CX
for Compute Nodes



Agilio FX
for Bare Metal Servers



Agilio LX
for Service Nodes



注:参考情報です。現在の仕様は
各メーカーにお問い合わせ下さい

P4 on CPU

BMv2

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

README.md

BEHAVIORAL MODEL REPOSITORY

build passing

This is the second version of the P4 software switch (aka behavioral model), nicknamed bmv2. It is meant to replace the original version, p4c-behavioral, in the long run, although we do not have feature equivalence yet. Unlike p4c-behavioral, this new version is static (i.e. we do not need to auto-generate new code and recompile every time a modification is done to the P4 program) and written in C++11. For information on why we decided to write a new version of the behavioral model, please look at the FAQ below.

P4 to eBPF compiler

https://github.com/p4lang/p4c/tree/master/backends/ebpf

README.md

eBPF Backend

The back-end accepts only P4_16 code written for the `ebpf_model.p4` filter model. It generates C code that can be afterwards compiled into eBPF (extended Berkeley Packet Filters https://en.wikipedia.org/wiki/Berkeley_Packet_Filter) using clang/llvm or bcc (https://github.com/iovisor/bcc.git).

An older version of this compiler for compiling P4_14 is available at https://github.com/iovisor/bcc/tree/master/src/cc/frontends/p4

Identifiers starting with ebpf_ are reserved in P4 programs, including for structure field names.

Stratum Tutorials: Container of BMv2 + Stratum

https://github.com/stratum/tutorial

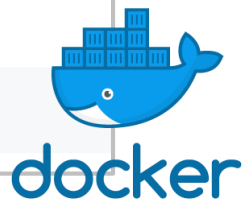
Starting a Mininet topology of Stratum switches

This tutorial uses Docker. If it is not installed, follow these instructions to install it.

There is a Docker container with Mininet and the Stratum BMv2 switch preinstalled: `opennetworking/mn-stratum`

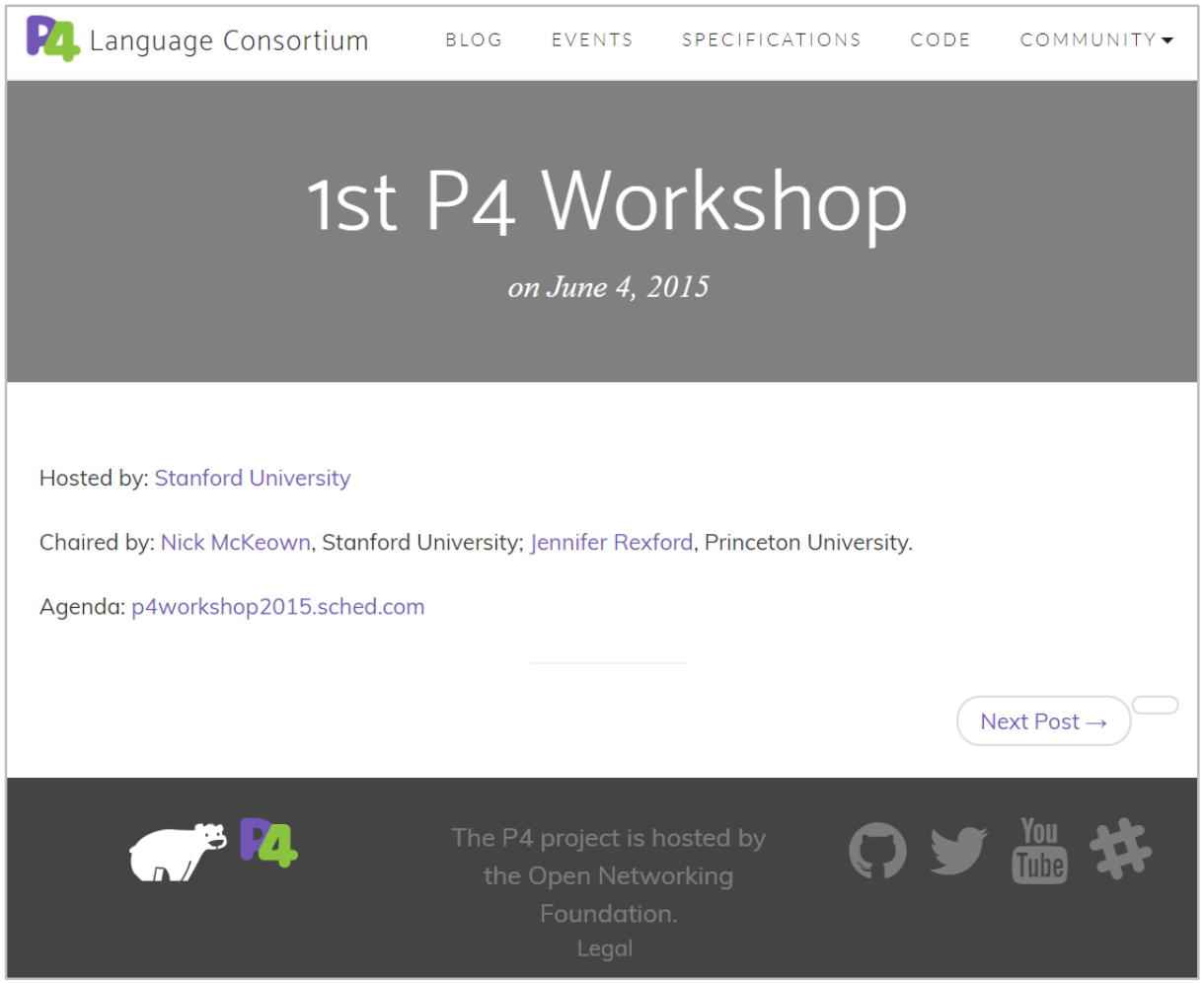
We can use the following command to start the container:

```
docker run --privileged --rm -it -p 50001:50001 opennetworking/mn-stratum
```

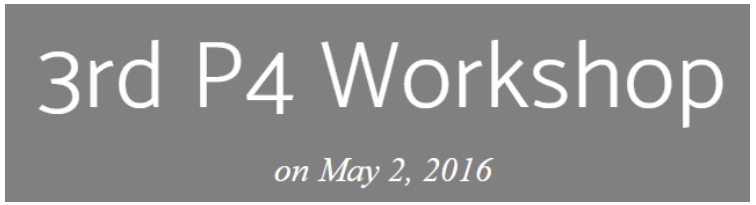


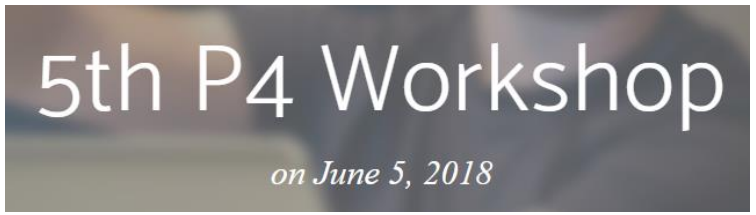
P4 コミュニティの近況

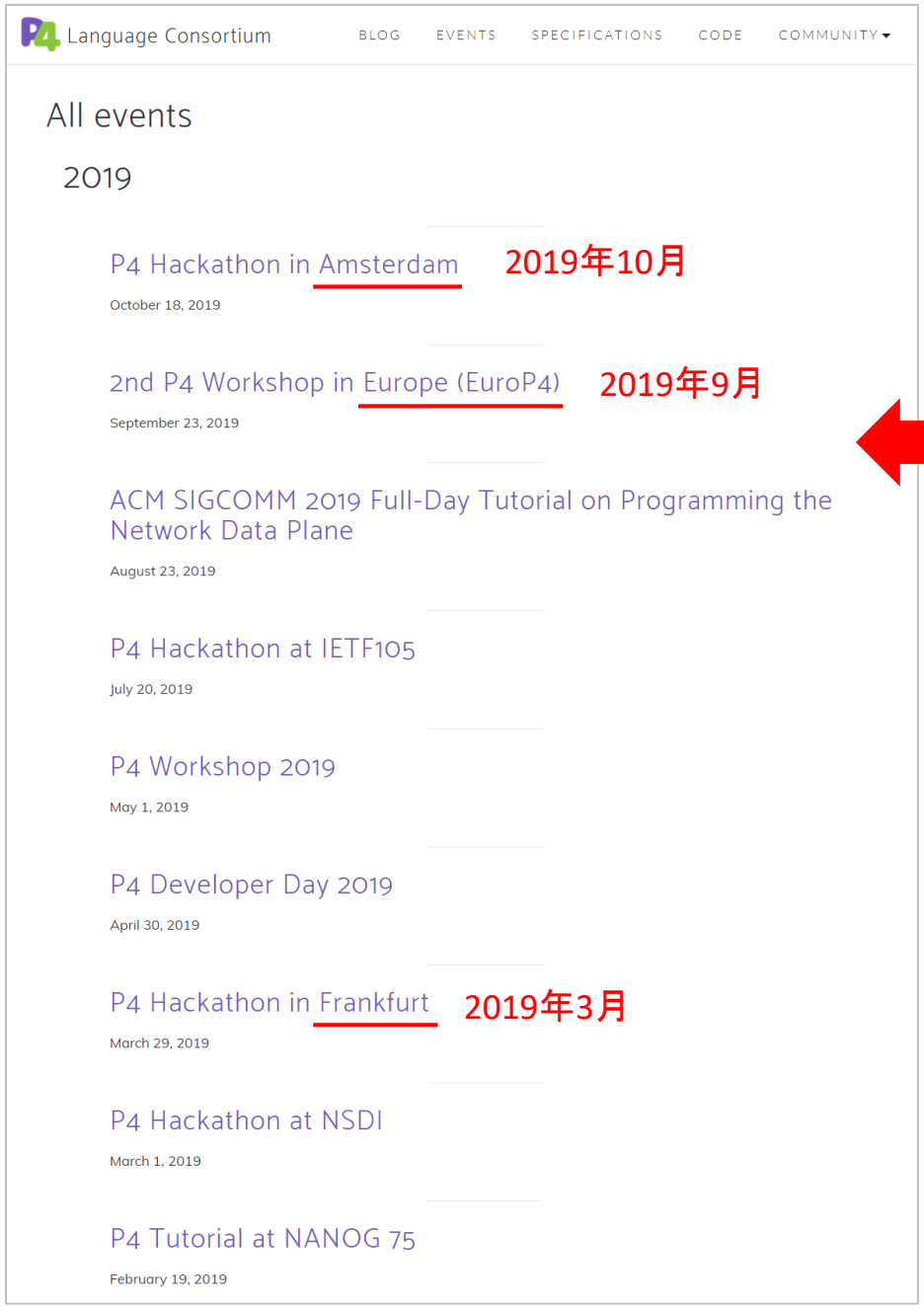
本家 P4 Workshop ＠Stanford, California

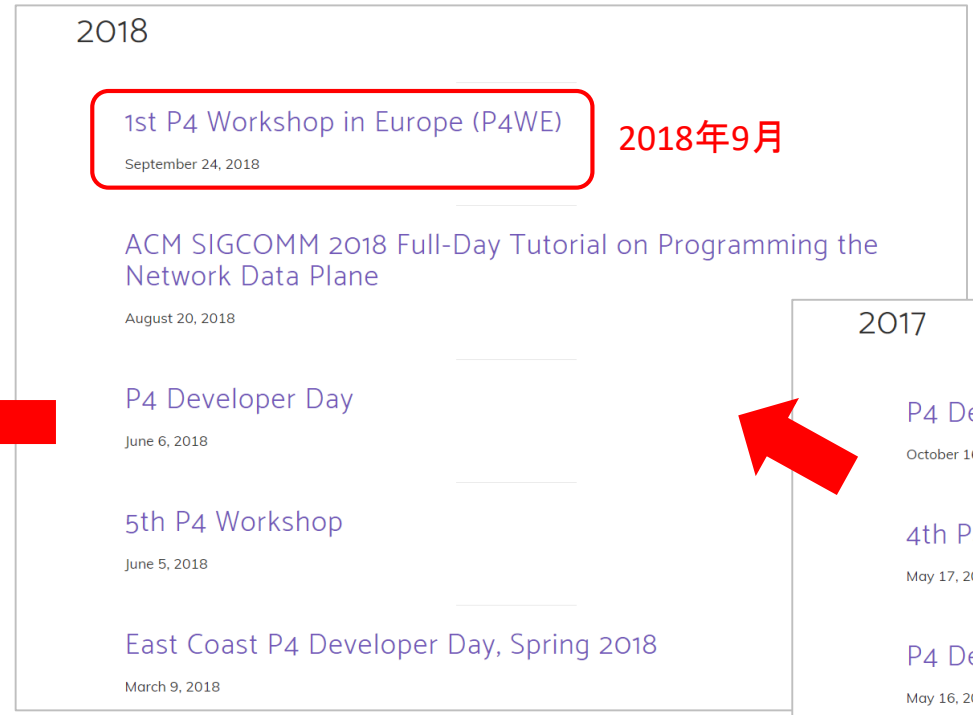


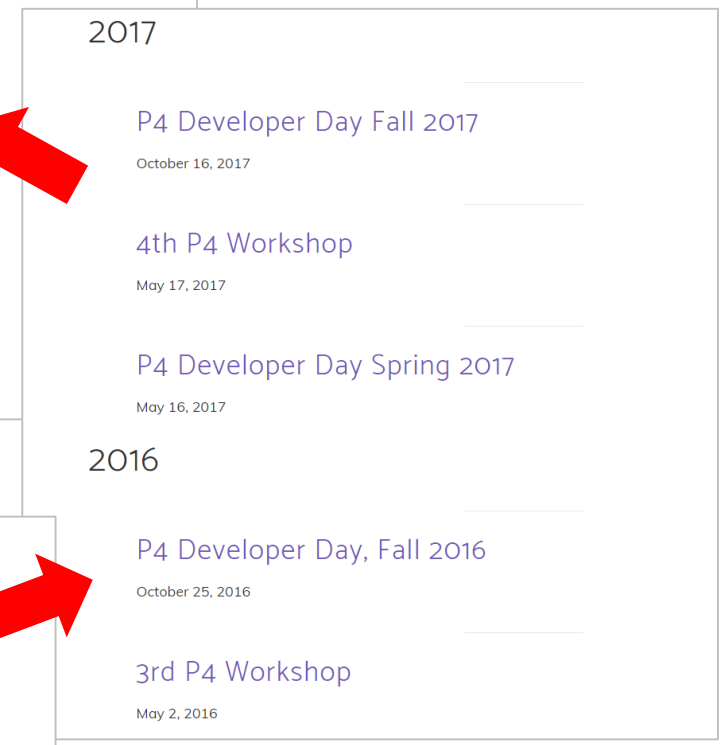
https://p4.org/events/

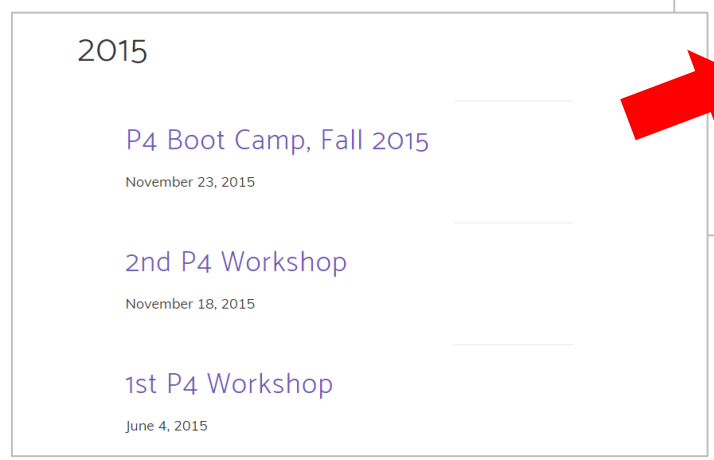





Language Consortium
BLOG
EVENTS
SPECIFICATIONS
CODE
COMMUNITY
All events
2019
P4 Hackathon in Amsterdam
2019年10月
October 18, 2019
2nd P4 Workshop in Europe (EuroP4)
2019年9月
September 23, 2019
ACM SIGCOMM 2019 Full-Day Tutorial on Programming the Network Data Plane
August 23, 2019
P4 Hackathon at IETF105
July 20, 2019
P4 Workshop 2019
May 1, 2019
P4 Developer Day 2019
April 30, 2019
P4 Hackathon in Frankfurt
2019年3月
March 29, 2019
P4 Hackathon at NSDI
March 1, 2019
P4 Tutorial at NANOG 75
February 19, 2019


2018
1st P4 Workshop in Europe (P4WE)
2018年9月
September 24, 2018
ACM SIGCOMM 2018 Full-Day Tutorial on Programming the Network Data Plane
August 20, 2018
P4 Developer Day
June 6, 2018
5th P4 Workshop
June 5, 2018
East Coast P4 Developer Day, Spring 2018
March 9, 2018

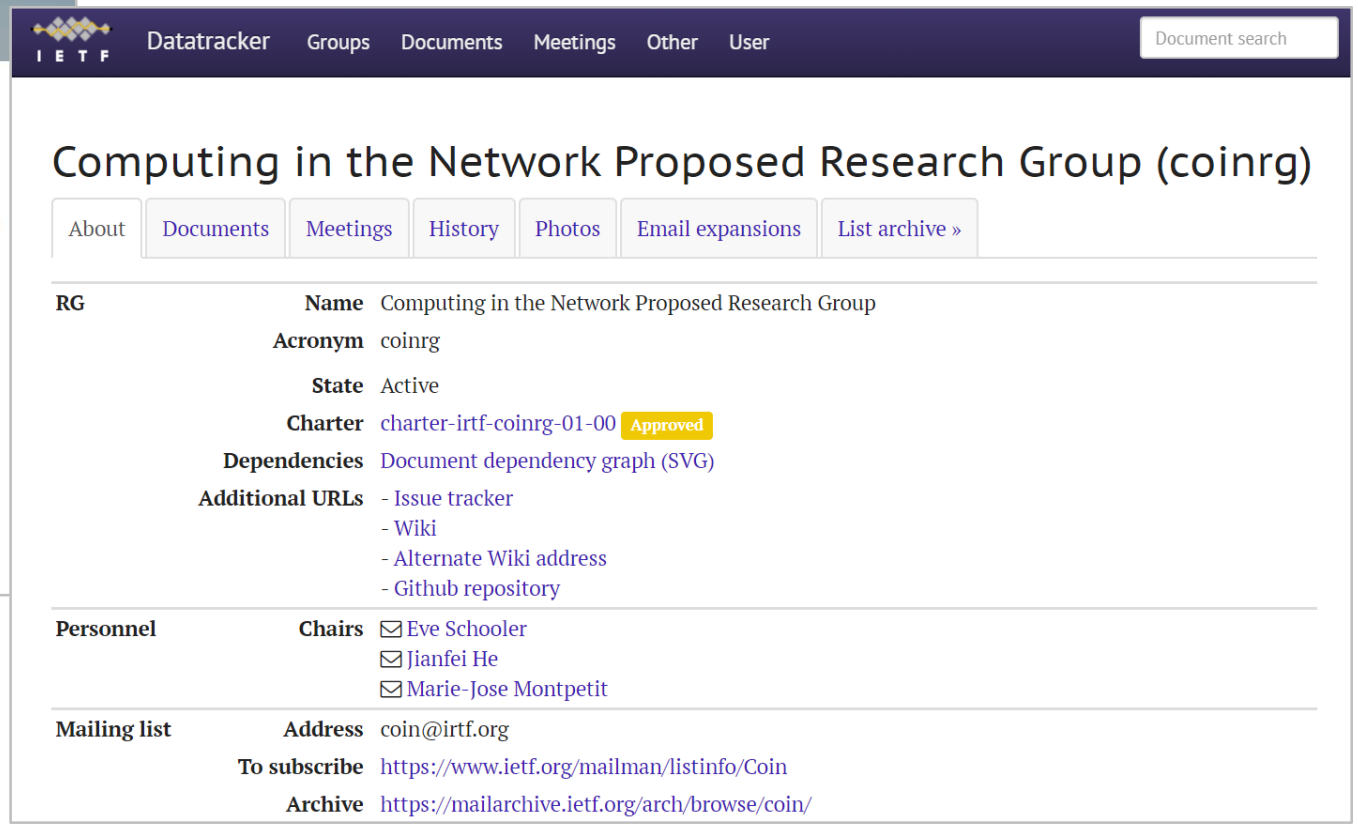

2017
P4 Developer Day Fall 2017
October 16, 2017
4th P4 Workshop
May 17, 2017
P4 Developer Day Spring 2017
May 16, 2017
2016
P4 Developer Day, Fall 2016
October 25, 2016
3rd P4 Workshop
May 2, 2016


2015
P4 Boot Camp, Fall 2015
November 23, 2015
2nd P4 Workshop
November 18, 2015
1st P4 Workshop
June 4, 2015

P4 @ IETF Hackathon

https://p4.org/events/2019-07-20-ietf105/

https://datatracker.ietf.org/rg/coinrg/about/



BoF や ユーザー会での活動

B3 ソフトウエアルータ・スイッチBoF

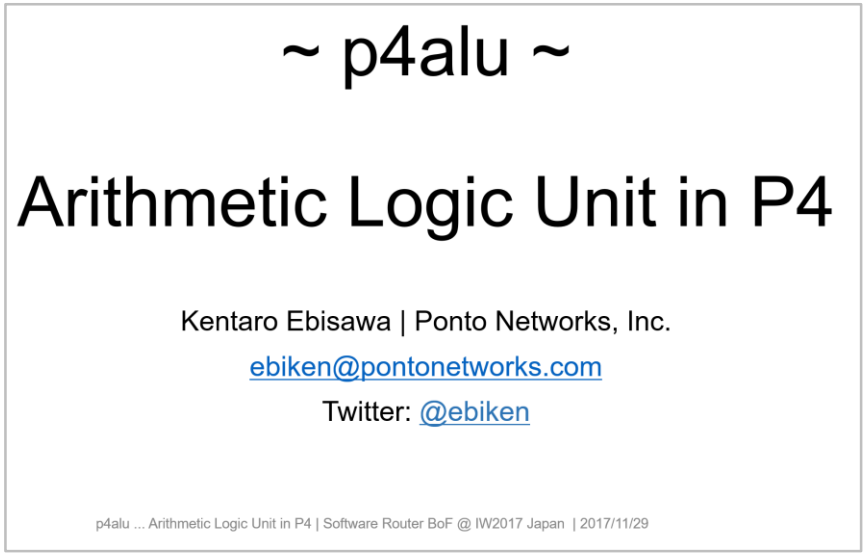
いいね！ 41 Pocket B! ツイート

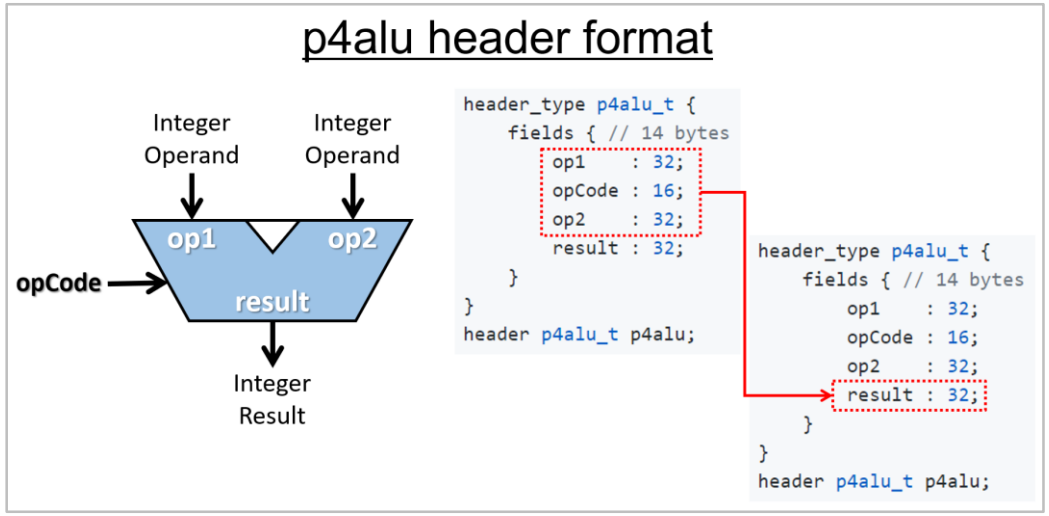
日時	2017年11月29日(水) 19:00 ～ 20:30
場所	3F Room0
主催	ソフトウエアルータ・スイッチ勉強会
参加料金	＜無料、当日受付のみ＞
内容	ソフトウェアスイッチ/ルータの開発者、 ユーザから技術トレンドについて共有してもらい、 BoF参加者とともに今後のソフトウェアスイッチ/ルータの方向性について議論する。

※時間割、内容、講演者等につきましては、 予告なく変更になる場合がございます。 あらかじめご了承ください。

Internet Week 2017

https://www.nic.ad.jp/iw2017/program/b3/





https://www.slideshare.net/kentaroebisawa/p4alu-arithmetic-logic-unit-in-p4

BoF や ユーザー会での活動

2018年7月12日

https://www.janog.gr.jp/meeting/janog42/program/SP-P4

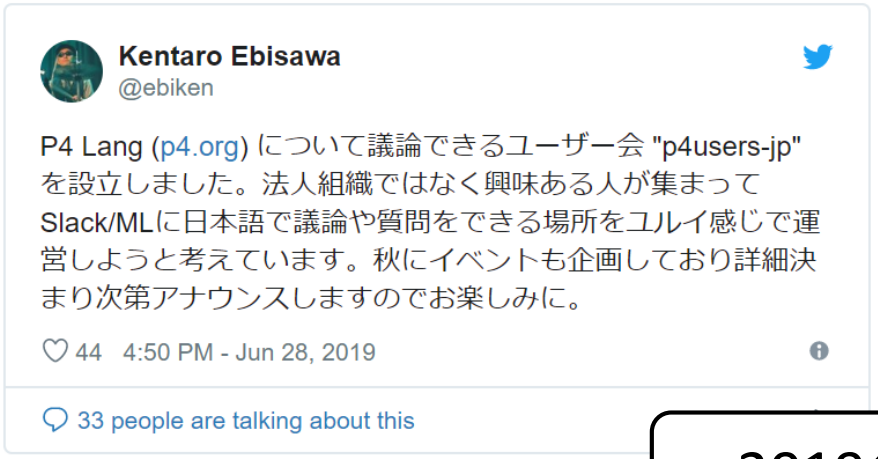
P4 Workshop 2018 in Tokyo
【データプレーンプログラミングの世界】

主催：P4.org、Barefoot Networks、ネットワンシステムズ株式会社(協賛)

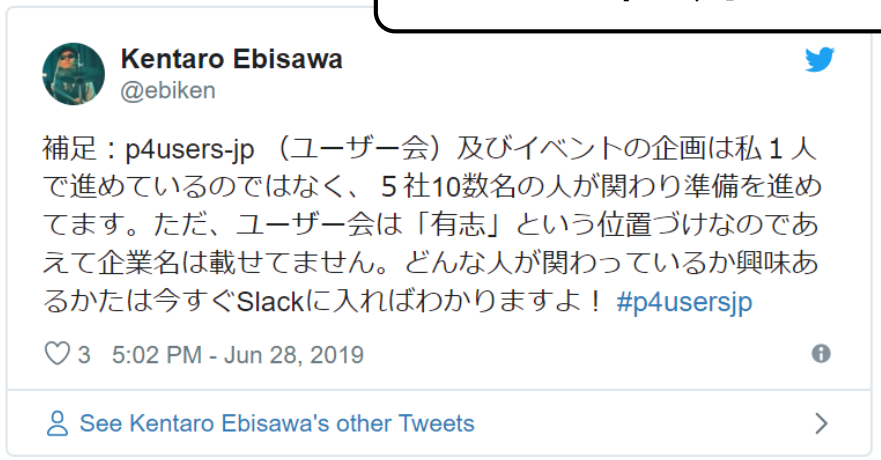
講演企業

Barefoot Networks
Net One Systems
P4.org
Arista
Cisco
Apresia & Edgecore
Kaloom Software

「日本 P4 ユーザ会」



2019年6月28日



日本 P4 ユーザ会 運営委員

氏名	役職	所属
ハディ ザケル Zaker Hadi	会長	ネットワンシステムズ株式会社 Net One Systems Co., Ltd.
海老澤 健太郎 Kentaro Ebisawa	役員	トヨタ自動車株式会社 Toyota Motor Corporation
岸本 貴之 Takayuki Kishimoto	役員	APRESIA Systems株式会社 APRESIA Systems, Ltd.
桑田 斉 Hitoshi Kuwata	役員	APRESIA Systems株式会社 APRESIA Systems, Ltd.
小柳 敏則 Toshinori Koyanagi	役員	インテル株式会社 Intel K. K.
清水 裕晶 Hiroaki Shimizu	役員	株式会社マクニカ　アルティマ カンパニー MACNICA, Inc. ALTIMA Company
新林 辰則 Tatsunori Shimbayashi	役員	ネットワンシステムズ株式会社 Net One Systems Co., Ltd.
鈴木 秀臣 Hideomi Suzuki	役員	株式会社マクニカ　アルティマ カンパニー MACNICA, Inc. ALTIMA Company
曽我 亨弘 Yukihiro Soga	役員	ネットワンシステムズ株式会社 Net One Systems Co., Ltd.
野津 雅洋 Masahiro Notsu	役員	ネットワンシステムズ株式会社 Net One Systems Co., Ltd.
久田 勇気 Yuki Hisata	役員	ネットワンシステムズ株式会社 Net One Systems Co., Ltd.
平部 真彬 Masaaki Hirabe	役員	株式会社マクニカ　アルティマ カンパニー MACNICA, Inc. ALTIMA Company
山崎 大輔 Daisuke Yamasaki	役員	インテル株式会社 Intel K. K.

https://p4users.org/committee-members/

https://p4users.org/

ホーム · 最新情報 · 日本 P4 ユーザ会 運営委員 · 日本 P4 ユーザ会 会則

日本 P4 ユーザ会

日本 P4 ユーザ会について

日本 P4 ユーザ会は P4 Lang (https://p4.org/) について日本語で語るグループです。

P4 関連のセミナー情報、カンファレンス情報及び技術情報を本ページで共有します。

インタラクティブなディスカッションは p4users-jp.slack.com でどうぞ。【Slack のリンク】

日本 P4 ユーザ会 2019 開催のお知らせ

第2条（目的）

1「日本 P4 ユーザ会」は、 P4 Lang (https://p4.org/) について日本語で語るグループであり、P4 言語について、オープンなディスカッションを出来るインフラを提供し、技術情報の共有、交換を活性化することで、日本の技術者、および P4 利用者に貢献することを目的としたグループである。

第3条（会の活動）

1　前条の目的達成のために、以下の活動を行なう。

(1)メーリングリストもしくは Slack による会員相互の情報交換

(2)ミーティング開催による会員相互の情報交換

(3)P4 設計技術の研究、開発

(4)P4 設計技術に関する技術文書の蓄積

(5)P4 設計技術に関する技術文書の翻訳

(6)その他本会の目的を達成するために必要な活動

第2章 会員

第4条（入会）

1 本会の参加員は、第2条の目的に賛同し、第3条の事業遂行に協力する意思を有する個人、法人、団体とする。

2 本会所定の（メーリングリストもしくは Slack） への参加により会員となる。

日本 P4 ユーザ会 2019 開催

2019年10月11日

145人＋スタッフ

日本 P4 ユーザ会 2019 開催

時間	講演タイトル / 登壇者
09:30〜14:30	受付
10:00〜10:10	**冒頭挨拶・会場説明** ネットワンシステムズ株式会社 藤田 雄介
10:10〜10:55	**P4の現状と展望・そして我々にできること** トヨタ自動車株式会社 海老澤 健太郎
10:55〜11:40	**キャリアにおける P4 ユースケースの紹介** NTTネットワークサービスシステム研究所 武井 勇樹
11:40〜12:25	**P4 テストベッドについて** 国立研究開発法人情報通信研究機構（NICT） 石井 秀治
12:25〜14:00	休憩
14:00〜14:30	**Programmable スイッチによる GTP/SRv6 の Stateless 変換の性能評価** トヨタ自動車株式会社 李 忠翰
14:30〜15:00	**インテル® FPGA PAC N3000 を P4 でプログラミングする NETCOPE P4 コンパイラの使用事例** インテル株式会社 小柳 敏則 株式会社マクニカ　アルティマ カンパニー 清水 裕晶
15:00〜15:30	**CiscoにおけるP4の活用と展望** シスコシステムズ合同会社 佐藤 哲大
15:30〜16:00	**Arista 7170紹介とデモンストレーション** アリスタネットワークスジャパン合同会社 土屋 師子生
16:30〜17:00	**Cloud-Grade Routing Stack for P4/Stratum** ジュニパーネットワークス株式会社 有村 淳矢
17:00〜17:30	**進化するデータプレンプログラマビリティ対応ハードウェアと実現されるユースケース** APRESIA Systems株式会社 桑田 斉
17:30〜18:00	**In-band Network Telemetry とその可能性** ネットワンシステムズ株式会社 新林 辰則

2019年10月11日

- コミュニティ
- ユーザー（狭義の）
- リサーチ
- テクノロジープロバイダー
- システムインテグレーター

今後の活動予定

- ハンズオン（基本的なプログラム・動作方法）
- 日本P4ユーザー会 カンファレンス（年1回？）
- テーマ別のワークショップ？
- 関西ミニカンファレンス？（企画＆運営やる人募集中！）

「実施したい！」「参加したい！」という企画を
Slack でコメントお願いします！

https://p4users.org/

ホーム · 最新情報 · 日本 P4 ユーザ会 運営委員 · 日本 P4 ユーザ会 会則

日本 P4 ユーザ会

日本 P4 ユーザ会について

日本 P4 ユーザ会は P4 Lang (https://p4.org/) について日本語で語るグループです。

P4 関連のセミナー情報、カンファレンス情報及び技術情報を本ページで共有します。

インタラクティブなディスカッションは p4users-jp.slack.com でどうぞ。【Slack のリンク】

日本 P4 ユーザ会 2019 開催のお知らせ

リンクから参加