P4 at the Edge

Nate Foster
Cornell University

edge, noun.

1. A place or part farthest away from the center of something
2. The point or state immediately before something momentous occurs

Photo credit: Mollie Foster

State of P4

New Features

- Continued evolution of $P4_{16}$ Language, P4Runtime, and INT specifications
- Growing set of developers actively contributing to open-source software packages

New Targets

- User-space (e.g., Orange p4c-ubpf)
- Programmable NICs (e.g., Pensando DSC)

New Applications

- Congestion control using telemetry
- DDoS mitigation on P4-enabled switches

Future Directions

Language Design

Graceful evolution to accommodate richer processing possible at edge

Architecture

Renewed enthusiasm for a “standard model” for programmable NICs

APIs

Push pipeline independence further up the stack—e.g. into switch OS

Applications

- On cusp of a “Cambrian explosion”
- Broaden scope beyond telemetry

Photo credit: Nate Foster

Technical Challenges

State

How do we manage the richer kinds of state that are available at the network edge?

Expressiveness

How do we accommodate complex transformations that go beyond P4's existing pipeline architectures?

Modularity, Portability, Predictability…

How do we do all this while retaining the essential features of the P4 language?

Guiding Principles

Community

- Open to anyone who wants to participate
- Decisions based on technical merit (not business or politics)

Strategic Goals

- Make P4 *the* de facto standard for packet processing, whether in hardware or software
- Find synergies with related efforts (e.g., ONF, eBPF, XDP, etc.)

Core Philosophy

- Declarative features with clear semantics
- Domain-specific constructs familiar to practitioners
- Predictable resource utilization and performance

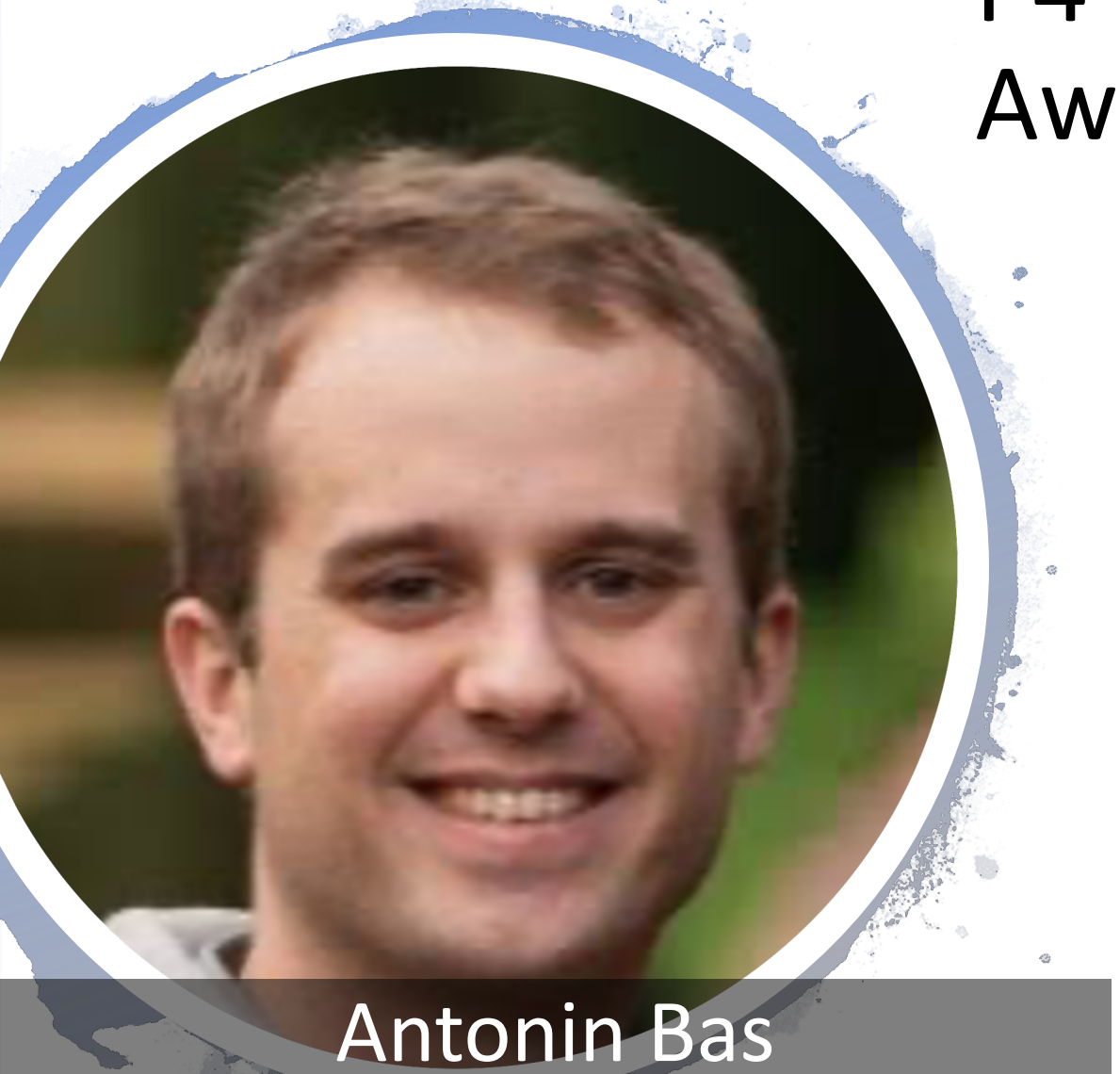
P4 Distinguished Service Award

P4 Distinguished Service Award

Mihai Budiu
VMware Research

Citation: For dedicated service to the P4 community as a designer of the $P4_{16}$ language, the primary developer of the p4c reference compiler, and co-chair of the P4 Language Design Working Group

P4 Distinguished Service Award



Antonin Bas
VMware

Citation: For dedicated service to the P4 community as a designer of P4Runtime, the primary developer of the bmv2 software switch, and co-chair of the P4 API Working Group

Working Group Updates

Language Design Working Group

v1.2.0 (October 2019)

- Strings and logging
- Richer types (int, tuples, etc.)
- Relaxed annotations

v1.2.1 (Spring 2020)

- Struct expressions
- Default initialization
- Side-effects

Software Development

- p4c-ubpf backend
- Differential testing of p4c (h/t Fabian Ruffy)

Future Plans

- Modularity
- Architecture specifications

API Working Group

v1.1.0 (March 2020)

- Overhaul master arbitration
- New RPCs for querying capabilities
- Better support for multicast

v1.2.0 (Spring 2020)

- Optional match kinds
- Structured annotations
- Language bindings (e.g., GoLang)

Software Development

- Stratum released as open-source

Future Directions

- Currently considering features for v2.0.0

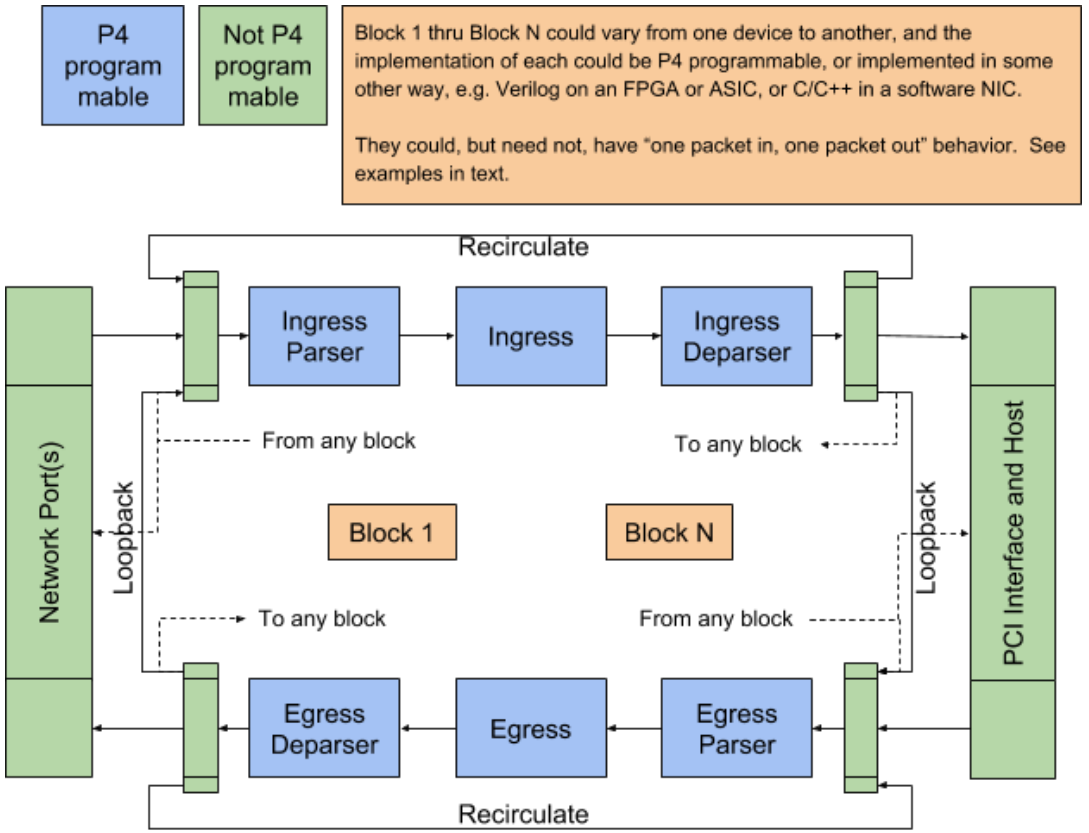
Architecture Working Group

Portable NIC Architecture (Fall 2020)

- Based on Portable Switch Architecture (PSA)
- Identifying use-cases
- Standardizing common functions and blocks
- Exploring language extensions

Software Development

- Continued work on PSA reference implementation based on p4c and bmv2



Applications Working Group

Overview

- In-band Network Telemetry (INT) has been widely adopted as a fundamental building block for building network infrastructure
- New use-cases and operation modes for INT have been identified

v2.0 (Spring 2020)

- New transport, metadata, operation modes
- Alignment with IETF IOAM
- Coalescing multiple reports in a single packet
- Domain-specific extensions provide flexibility without sacrificing efficiency

v2.x (Future)

- End host centric use cases
- INT-aware closed-loop control of transport and congestion

Community Highlights

Pensando Distributed Services Card

Features

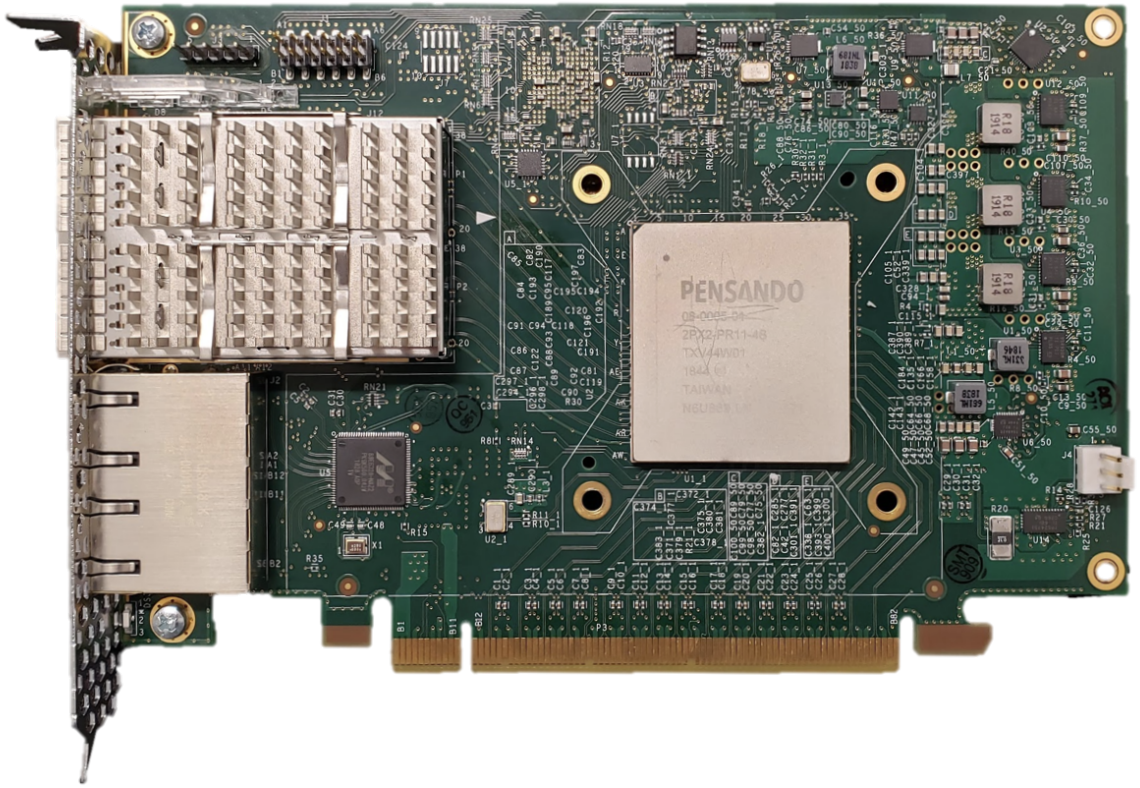
- P4-programmable pipeline
- Flexible ARM cores

Opportunities

- The *only* way to deal with widening gap between network and CPU performance
- Richer forms of processing become possible at the network edge

Questions

- How should P4 evolve to accommodate general-purpose constructs?
- How do we manage richer forms of state?
- How do we reason about performance?

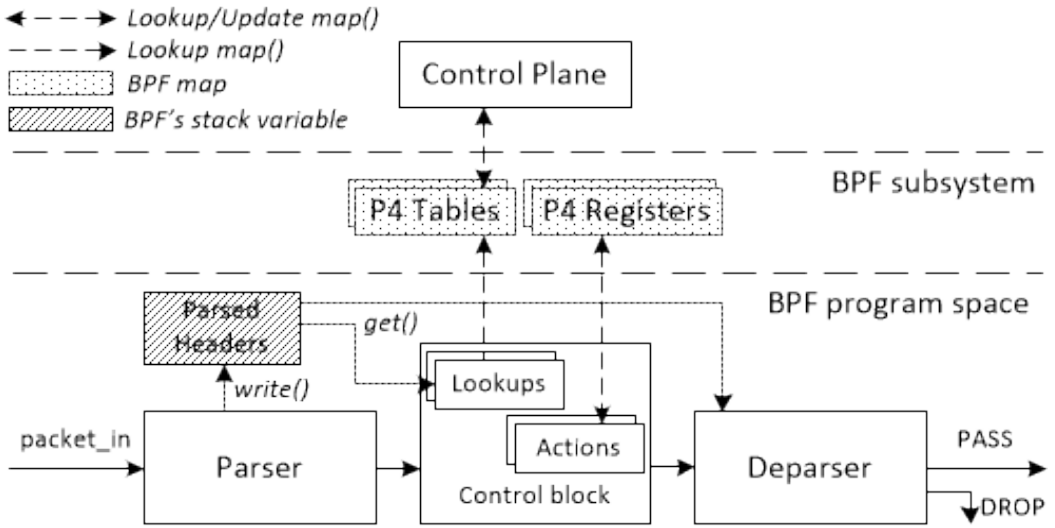


https://p4.org/p4/pensando-joins-p4.html

Orange's P4-to-uBPF Compiler

Features

- New backend for p4c
- Simple architectural model:
  - Parser
  - Match-Action Control
  - Deparser
- Enables using P4 with kernel bypass frameworks (DPDK, AF_XDP, etc.)



Opportunities

- Flexibility of eBPF
- Performance of P4
- Rapid prototyping of language extensions

Network-Assisted Congestion Feedback

Insight

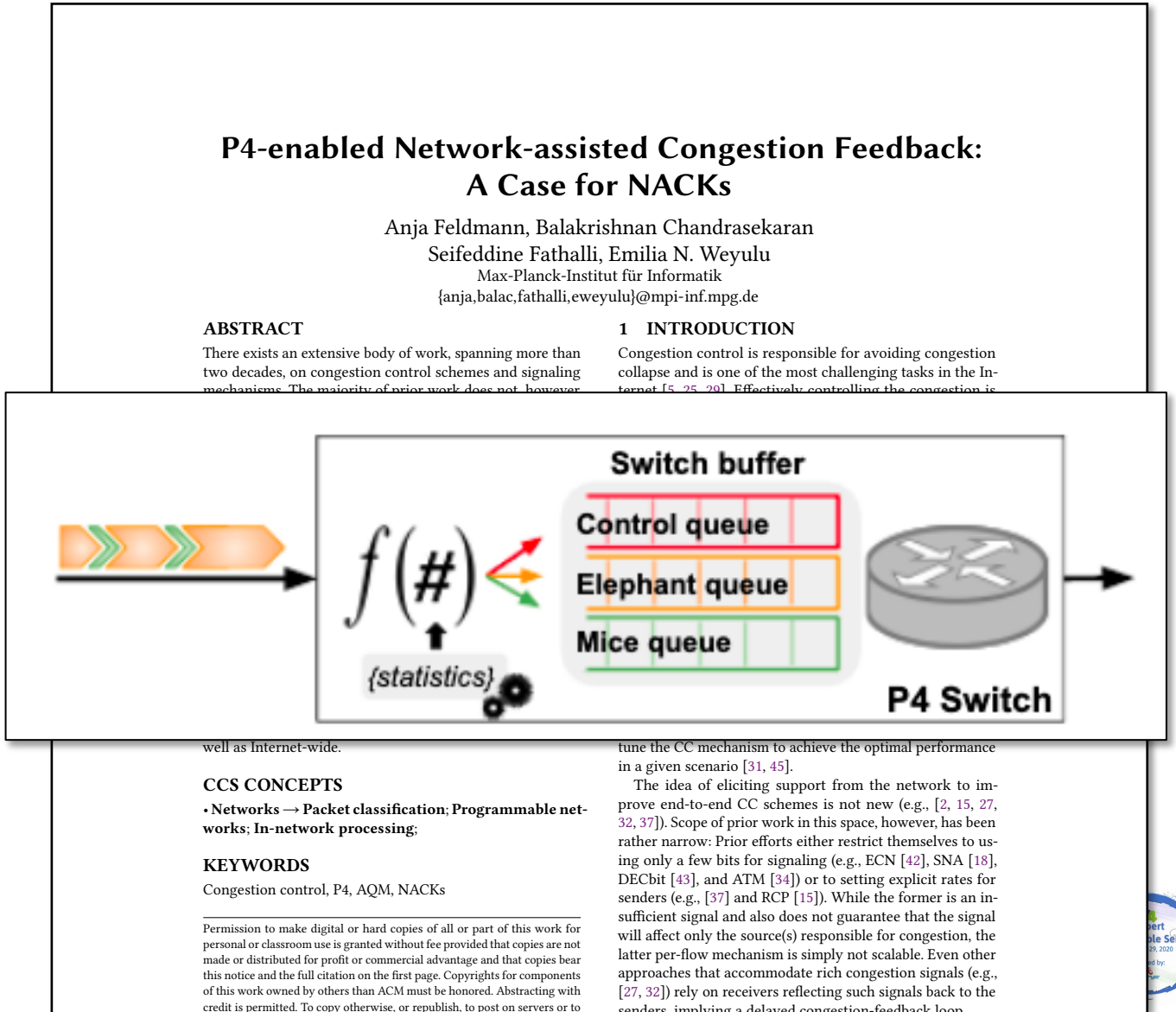
Use P4-enabled switches to give end hosts precise feedback about network congestion

Challenges

- Scalable data collection
- Elephants vs. mice flows
- Fair sharing of resources
- Avoid introducing new faults

Open Question

- How should we think about congestion control schemes with telemetry-driven approaches?



P4-enabled Network-assisted Congestion Feedback: A Case for NACKs

Anja Feldmann, Balakrishnan Chandrasekaran
Seifeddine Fathalli, Emilia N. Weyulu
Max-Planck-Institut für Informatik
{anja,balac,fathalli,eweyulu}@mpi-inf.mpg.de

ABSTRACT

There exists an extensive body of work, spanning more than two decades, on congestion control schemes and signaling

1 INTRODUCTION

Congestion control is responsible for avoiding congestion collapse and is one of the most challenging tasks in the In-

well as Internet-wide.

CCS CONCEPTS

• **Networks** → **Packet classification**; **Programmable networks**; **In-network processing**;

KEYWORDS

Congestion control, P4, AQM, NACKs

tune the CC mechanism to achieve the optimal performance in a given scenario [31, 45].

The idea of eliciting support from the network to improve end-to-end CC schemes is not new (e.g., [2, 15, 27, 32, 37]). Scope of prior work in this space, however, has been rather narrow: Prior efforts either restrict themselves to using only a few bits for signaling (e.g., ECN [42], SNA [18], DECbit [43], and ATM [34]) or to setting explicit rates for senders (e.g., [37] and RCP [15]). While the former is an insufficient signal and also does not guarantee that the signal will affect only the source(s) responsible for congestion, the latter per-flow mechanism is simply not scalable. Even other approaches that accommodate rich congestion signals (e.g., [27, 32]) rely on receivers reflecting such signals back to the

Wrapping Up…

Thank You

P4 Technical Steering Team

- Nate Foster (Cornell, chair)
- Nick McKeown (Stanford)
- Guru Parulkar (ONF)
- Jennifer Rexford (Princeton)
- Amin Vahdat (Google)

Working Group Co-Chairs

- *Language:* Mihai Budiu, Nate Foster
- *APIs:* Antonin Bas, Waqar Mohsin
- *Architecture:* Andy Fingerhut
- *Applications:* Mukesh Hira, Jeongkeun Lee
- *Education:* Robert Soulé, Noa Zilberman

General Chair: Guru Parulkar (ONF)

PC Co-Chairs

- Larry Peterson (ONF)
- Anirudh Sivaraman (NYU)

Organizers

- Sedef Ozcana (P4)
- Denise Barton (ONF)
- Rachel Everman (Intel)
- Michelle Roth (ONF)
- Timon Sloane (ONF)

Sponsors

- Google
- Intel
- Stordis

Get Involved

- **Join the P4 Project!**
  - No fee to participate
  - Lightweight legal agreement based on Apache2 License
  - Possible to become an ONF Collaborator or Member
- **Participate in Working Groups**
  - Anyone with a good idea can help shape the future of P4
  - Open governance model with code of conduct
  - Decisions made by consensus on technical merits
- **Contribute to P4 Software**
  - Compiler (p4c)
  - Software switch (bmv2)
  - Control-plane APIs (P4Runtime)
  - Tutorials & Documentation
  - Applications (INT)

Thank You!