Composing Dataplane Programs with µP4

Hardik Soni
Myriana Rifai, Praveen Kumar, Ryan Doenges, Nate Foster

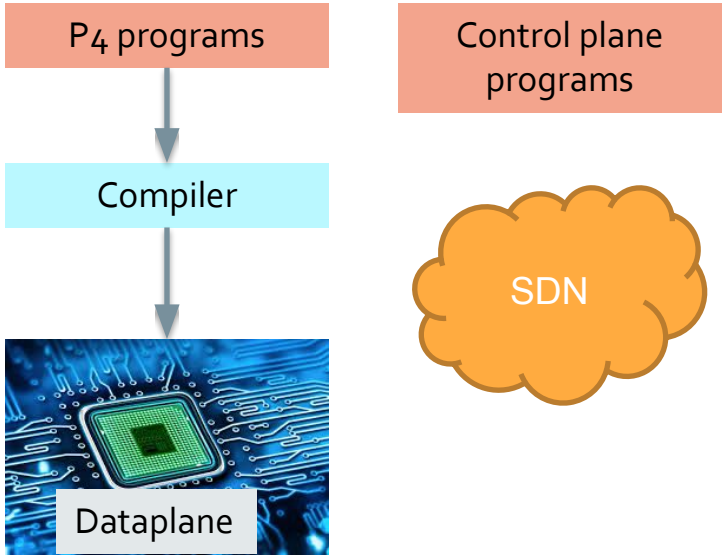
Cornell University

NOKIA Bell Labs

SIGCOMM 2020
new york city
USA

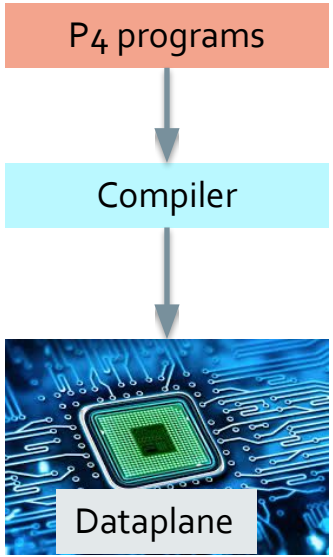
Modular Programming

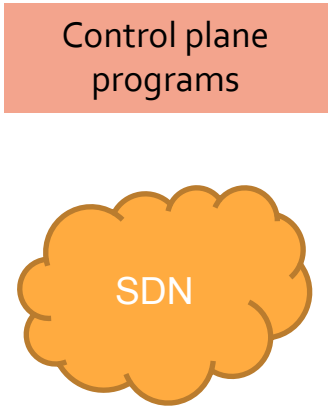
Networks increasingly look like any other software system…



Modular Programming

Networks increasingly look like any other software system…





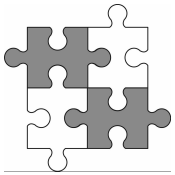
"Modularity based on abstraction is the way things are done"

2008 Turing Award Speech

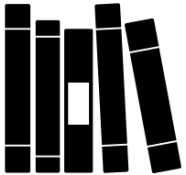
Barbara Liskov

Why is modularity important?

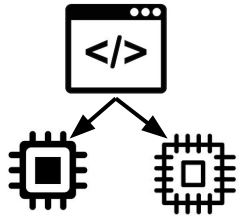
1. **Decompose** large systems into **small pieces** that can be developed **independently**



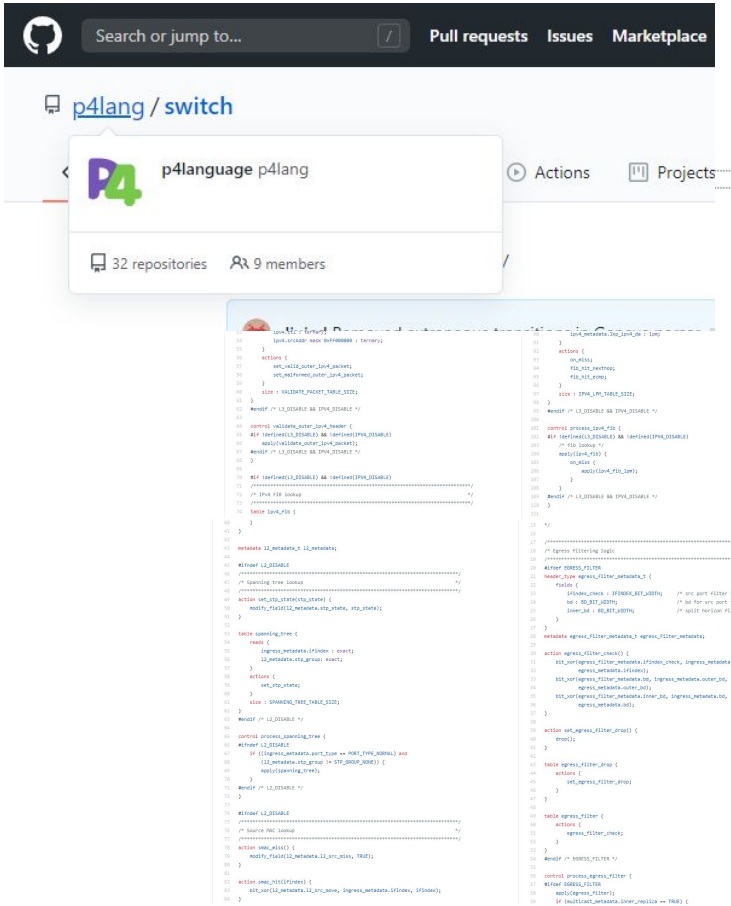
2. Develop **libraries** of common code fragments that can be **reused** in many different systems



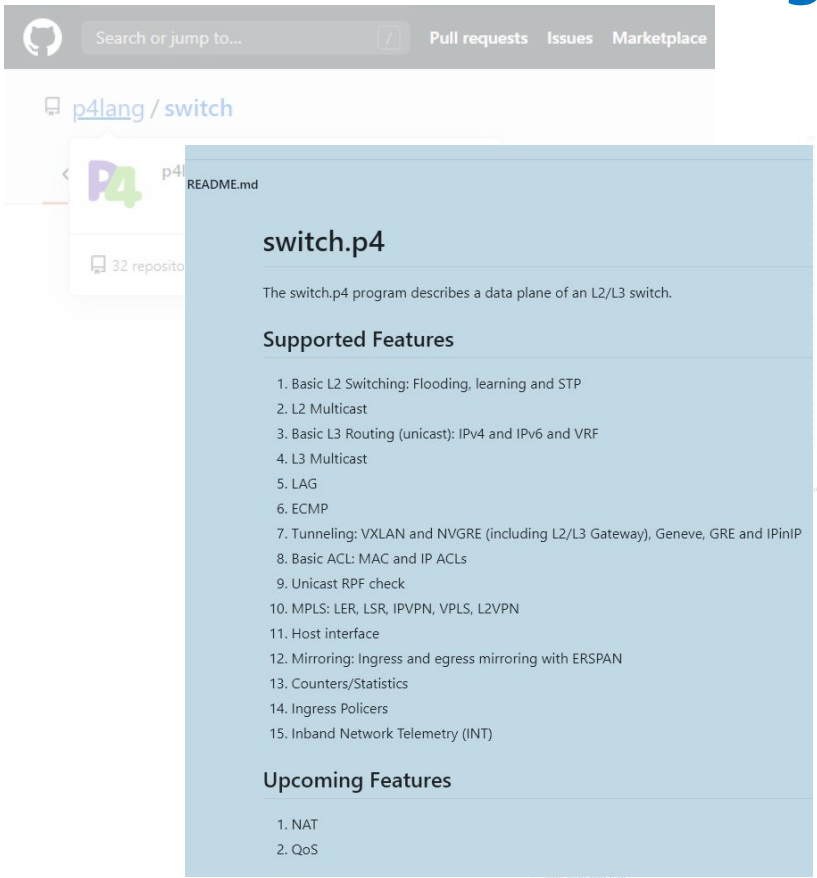
3. Write **high-level code** once, let a compiler **port** to **different** target **devices**



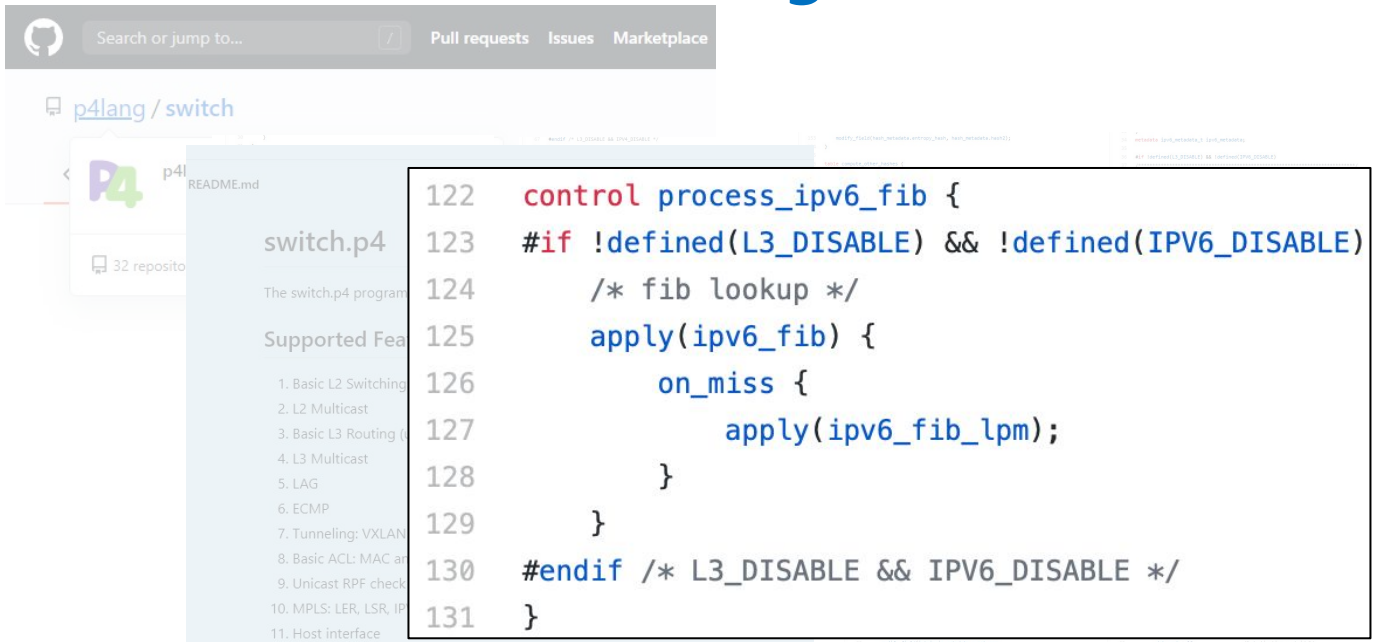
How are we building network software today?



How are we building network software today?



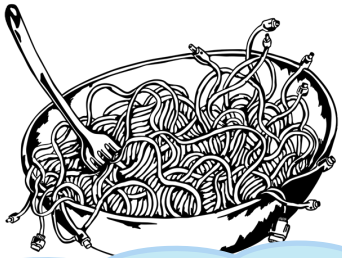
How are we building network software today?



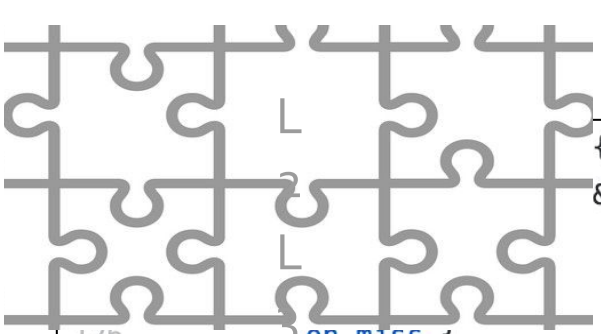

```
122  control process_ipv6_fib {
123  #if !defined(L3_DISABLE) && !defined(IPV6_DISABLE)
124      /* fib lookup */
125      apply(ipv6_fib) {
126          on_miss {
127              apply(ipv6_fib_lpm);
128          }
129      }
130  #endif /* L3_DISABLE && IPV6_DISABLE */
131  }
```

Status Quo

```
122  control process_ipv6_fib {
123  #if !defined(L3_DISABLE) && !defined(IPV6_DISABLE)
124      /* fib lookup */
125      apply(ipv6_fib) {
126          on_miss {
127              apply(ipv6_fib_lpm);
128          }
129      }
130  #endif /* L3_DISABLE && I
131  }
```



Wouldn’t it be nice if…



Challenges

P4 Programs are…

1. Monolithic
2. Written for a Heterogeneous Programming Model
3. Tightly Coupled to Target Architectures

1. P4 Programs are Monolithic

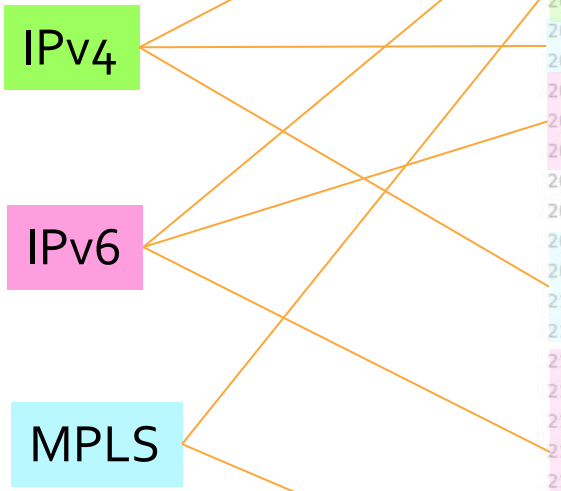
IPv4

IPv6

MPLS

Let's see how we implement routing with P4 today …

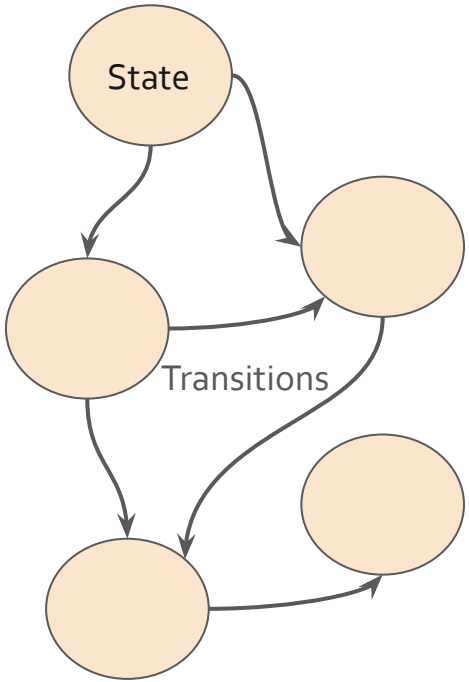
1. P4 Programs are Monolithic



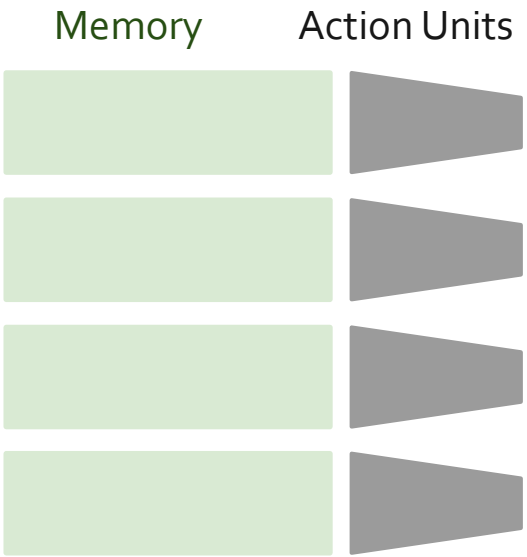

```
193  table l3_rewrite {
194      reads {
195          ipv4 : valid;
196  #ifndef IPV6_DISABLE
197          ipv6 : valid;
198  #endif /* IPV6_DISABLE */
199  #ifndef MPLS_DISABLE
200          mpls[0] : valid;
201  #endif /* MPLS_DISABLE */
202          ipv4.dstAddr mask 0xF0000000 : ternary;
203  #ifndef IPV6_DISABLE
204          ipv6.dstAddr mask 0xFF00000000000000000000000000
205  #endif /* IPV6_DISABLE */
206      }
207      actions {
208          nop;
209          ipv4_unicast_rewrite;
210  #ifndef L3_MULTICAST_DISABLE
211          ipv4_multicast_rewrite;
212  #endif /* L3_MULTICAST_DISABLE */
213  #ifndef IPV6_DISABLE
214          ipv6_unicast_rewrite;
215  #ifndef L3_MULTICAST_DISABLE
216          ipv6_multicast_rewrite;
217  #endif /* L3_MULTICAST_DISABLE */
218  #endif /* IPV6_DISABLE */
219  #ifndef MPLS_DISABLE
220          mpls_rewrite;
221  #endif /* MPLS_DISABLE */
222      }
223  }
```

Code snippet from switch.p4

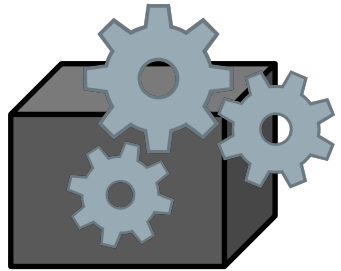
2. Heterogeneous Programming Model



State Machine based
Packet-parsing

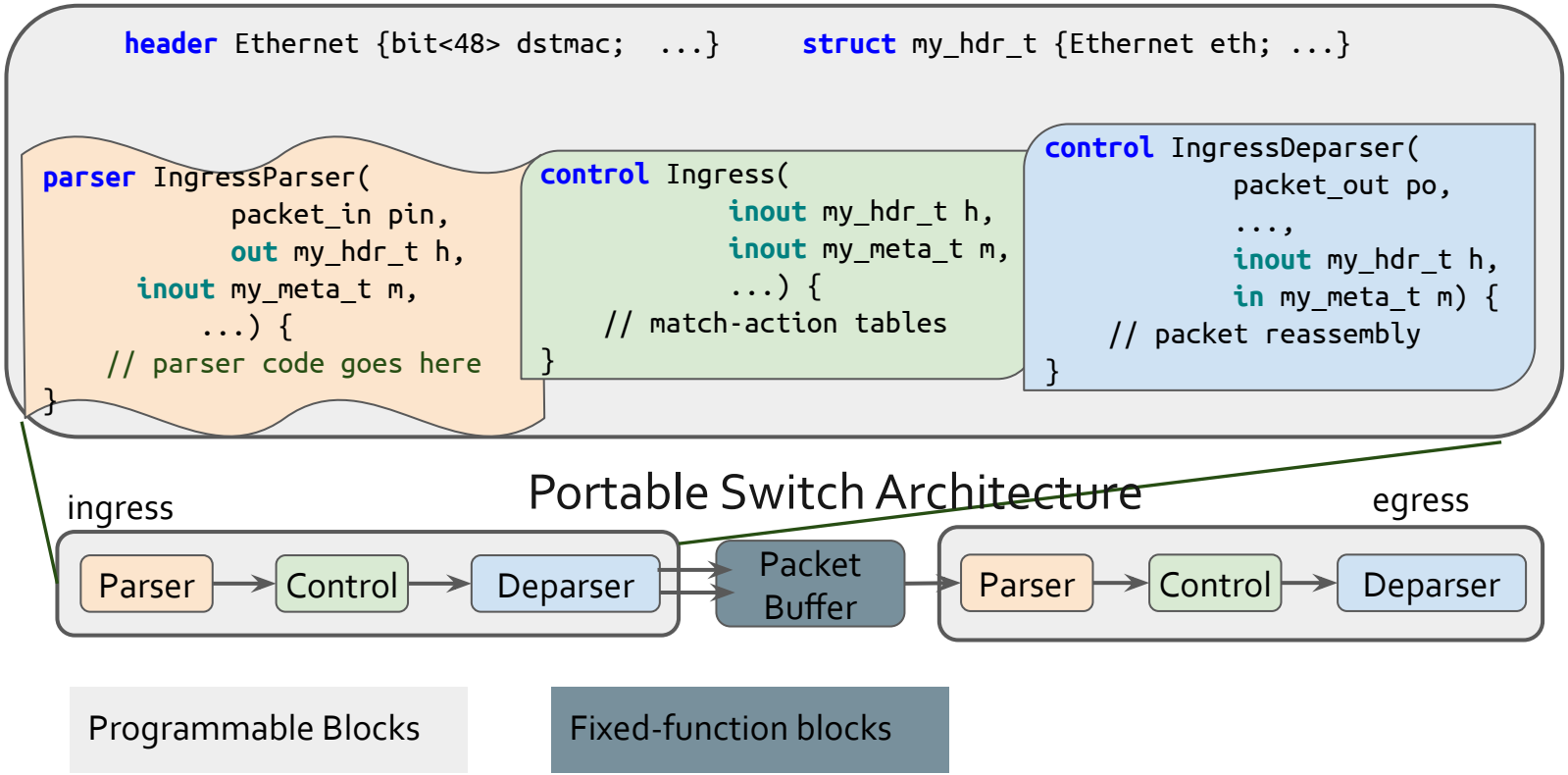


Match-Action based
Packet-processing



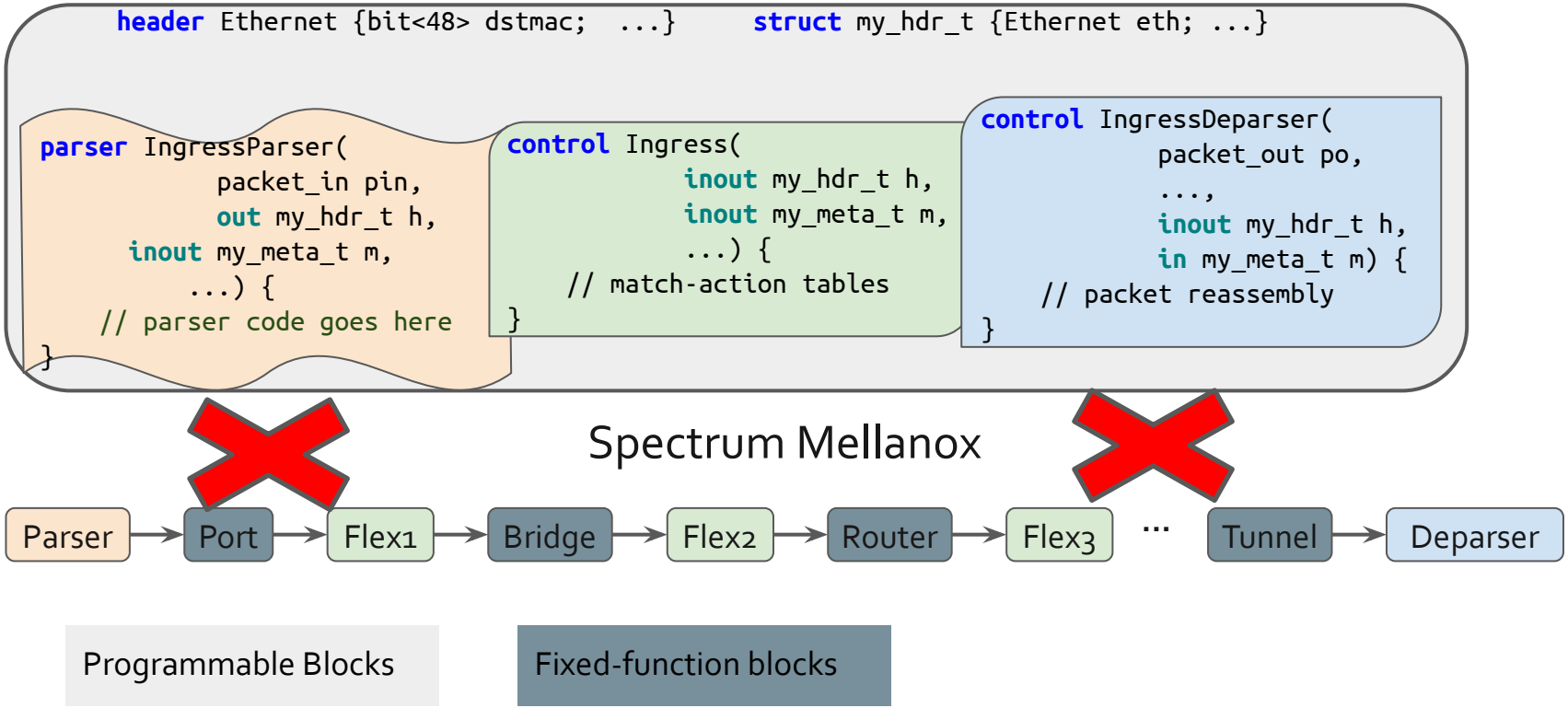
Fixed-function
Externs

3. Tight-Coupling with Architectures



3. Tight-Coupling with Architectures

Makes it difficult to port programs



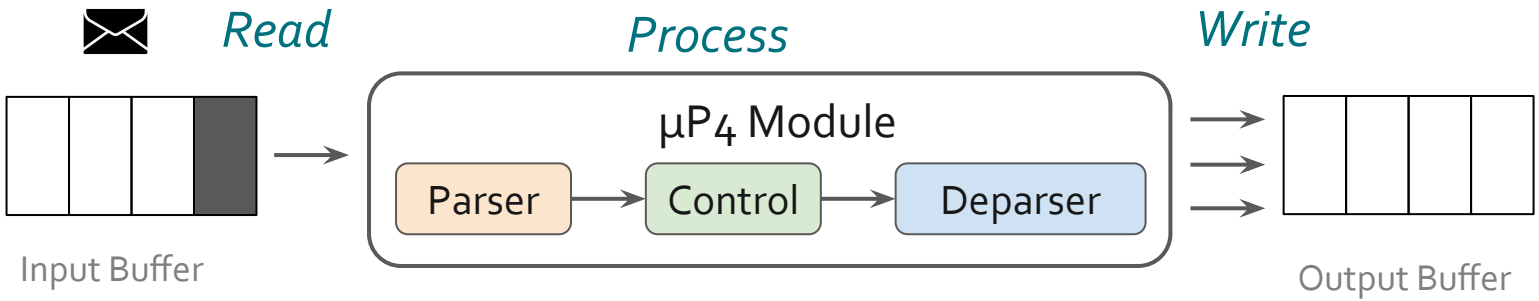
μP4

Design Insights

1. Higher-Level Abstractions for Dataplanes

2. Homogenize the Programming Model

µP4: Abstract Dataplane Model

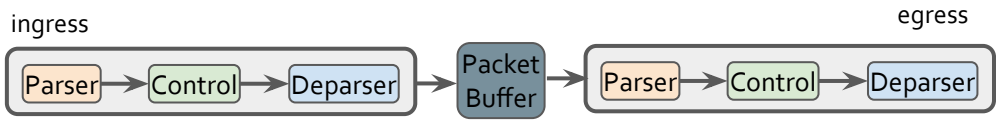


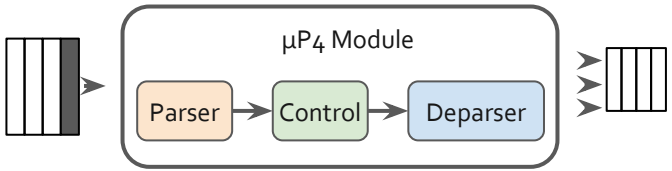
Key Ingredients:

- **Logical pipeline:** distill packet processing down to its essence, as a three-stage read-process-write function
- **Logical buffers:** Provide a common interface for composing

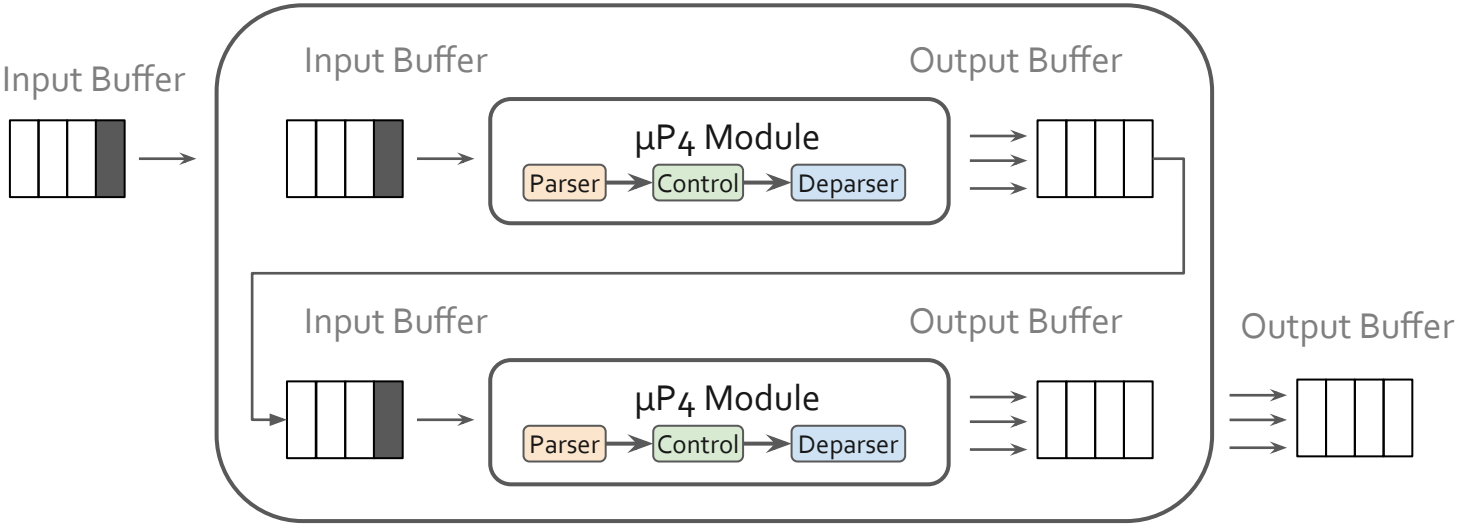
What µP4 Abstracts?

P4	µP4
Architecture pipeline	Logical micro-pipelines
Architecture metadata for packet processing	Logical Buffer
Architecture-specific fixed-functions	Logical Externs





µP4 supports Composition



Composed µP4 Module


μP4 Module
Parser
Control
Deparser

Compiling μP4

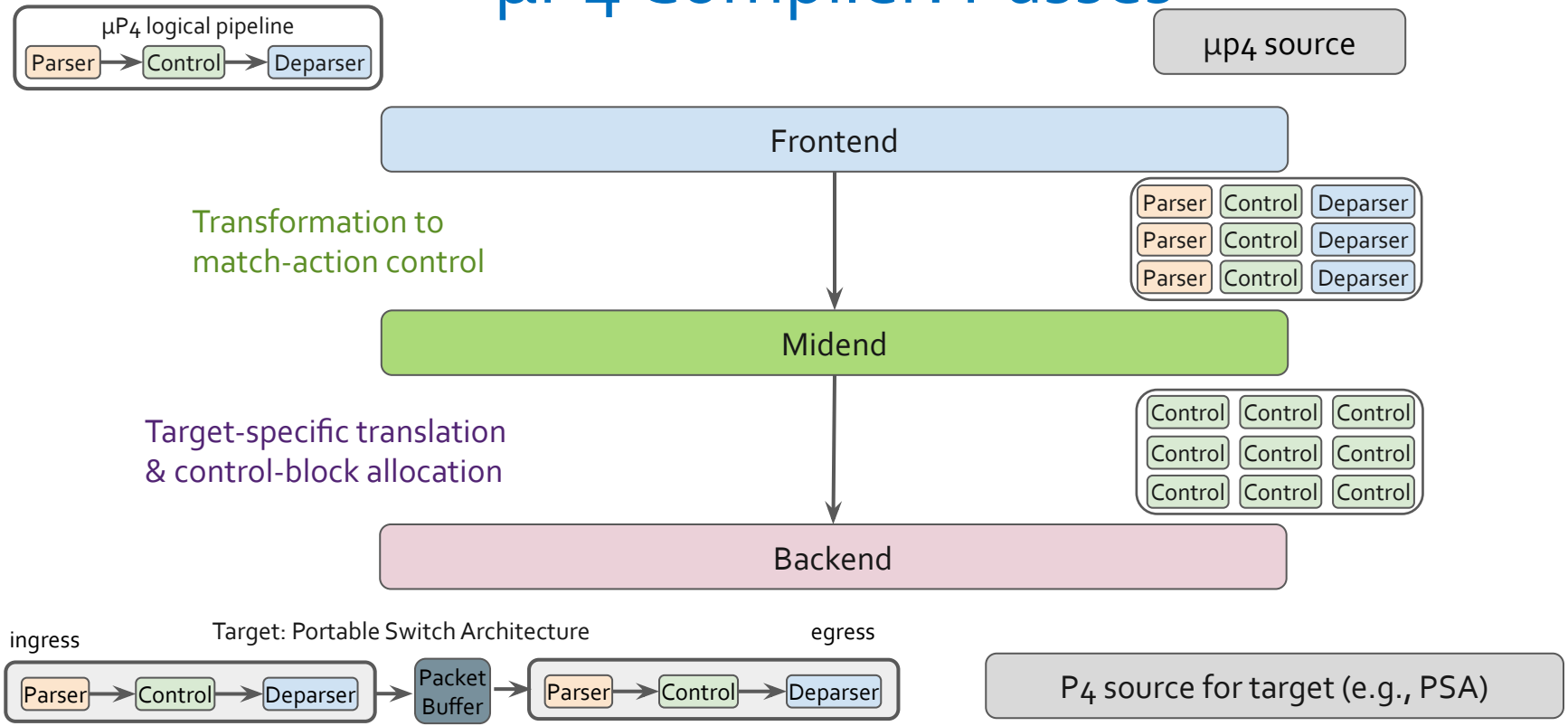
Overview

- Source-to-source P4_16 compiler
- Based on open-source p4c framework
- Implemented as 13.5 KLoC of C++
- Backends for Bmv2 and Tofino

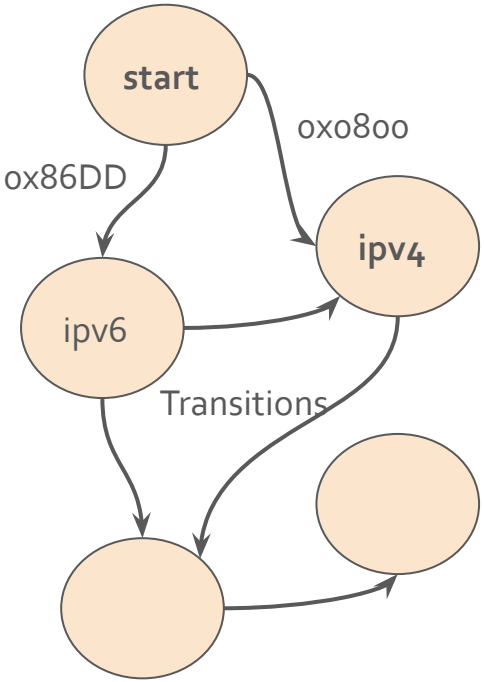
Technical Approach:

- Homogenize source program to facilitate analysis and transformation
- Rearrange code to respect target constraints
- Emit code for target's specialized packet-processing units

µP4 Compiler: Passes



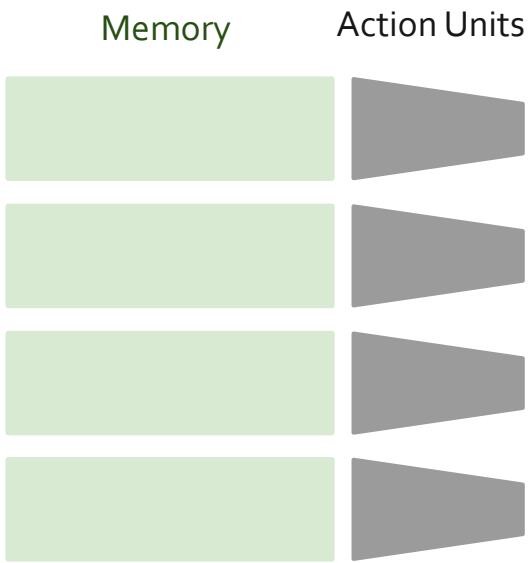
Midend: Homogenize Abstract Machines



Match: Bytes in packet

Action:

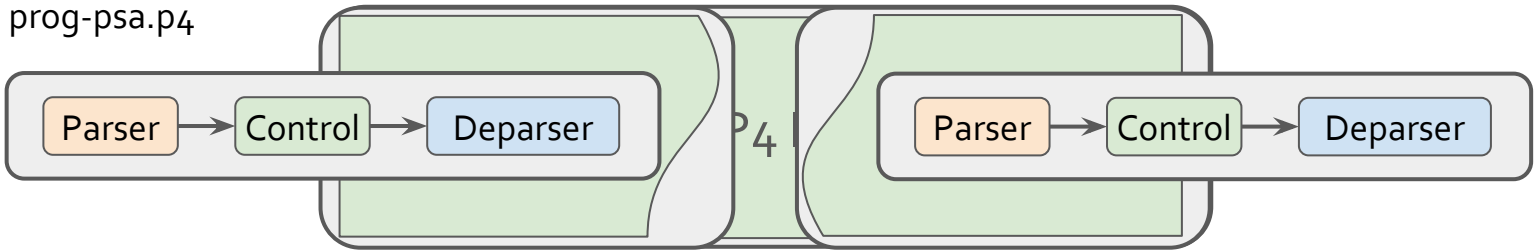
1. Extract/Copy Bytes
2. Transition to next state



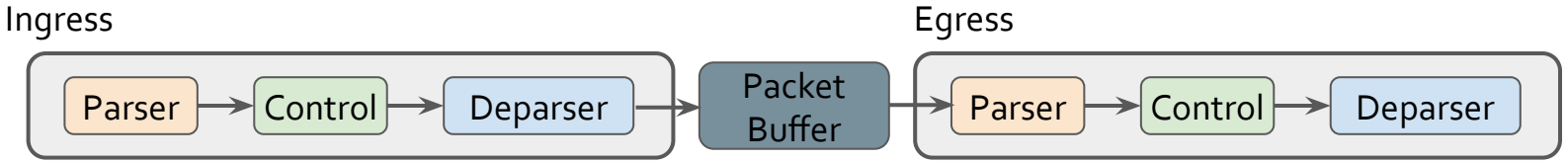
State machine based
Packet-parsing

Match-Action based
Packet-processing

Backend: Mapping to Target Architecture



Target: Portable Switch Architecture (PSA)



Evaluation

Questions

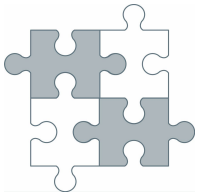
1. **Expressiveness**
   - Case study, developed library of common functions
2. **Portability**
   - Can run *same* programs on BMv2 and Tofino
3. **Efficiency**
   - Compared resource utilization against hand-written monolithic Tofino programs

Expressiveness: µP4 Modules

		LoC
µP4 Modules	ACL	120
	ETH	70
	IPv4	88
	IPv6	73
	MPLS	124
	NAT	112
	NPTv6	75
	SRv4	173
	SRv6	181

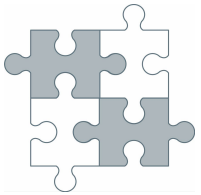
Expressiveness: Composition

	Composed Programs	P1	P2	P3	P4	P5	P6	P7
µP4 Modules	ACL						✓	
	ETH	✓	✓	✓	✓	✓	✓	✓
	IPv4		✓	✓	✓	✓	✓	✓
	IPv6	✓	✓	✓	✓		✓	✓
	MPLS		✓					
	NAT						✓	
	NPTv6						✓	
	SRv4							✓
	SRv6			✓				



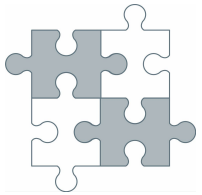
Example: Modular Router

	Composed Programs	P1	P2	P3	P4	P5	P6	P7
µP4 Modules	ACL						✓	
	ETH	✓	✓	✓	✓	✓	✓	✓
	IPv4		✓	✓	✓	✓	✓	✓
	IPv6	✓	✓	✓	✓		✓	✓
	MPLS		✓					
	NAT						✓	
	NPTv6						✓	
	SRv4							✓
	SRv6			✓				



Same program; multiple targets (BMv2 and Barefoot's Tofino)

Resource Overhead with Barefoot Tofino



	Composed Programs	P1	P2	P3	P4	P5	P6	P7
µP4 Modules	ACL						✓	
	ETH	✓	✓	✓	✓	✓	✓	✓
	IPv4		✓	✓	✓	✓	✓	✓
	IPv6	✓	✓	✓	✓		✓	✓
	MPLS		✓					
	NAT						✓	
	NPTv6						✓	
	SRv4							✓
	SRv6			✓				
#MAUs	P4 Monolithic	3	4	3	3	3	3	NA
	µP4 Composed	5	9	8	5	5	8	7

Wrapping Up…

Ongoing Work

Working to extend our current µP4 prototype to support:

- Packet replication and multicast
- Variable length headers
- Stateful packet-processing
- Control-plane APIs

https://github.com/cornell-netlab/MicroP4

Takeaways

Liskov was right: *"Modularity based on abstraction is how things are done"*

- Language Design
  - High-level abstractions for data plane programming
- Compiler
  - Composes different modules into single program
  - Generates code for underlying targets
- Experience
  - Developed a modular router

µP4 enables portable, modular, & composable data plane programming!