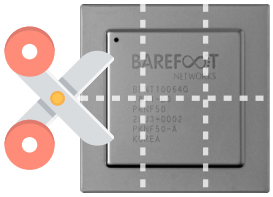


Slicing Match-Action Pipeline Resources for Multitenancy on Programmable Switches

Johannes Krude, Felix Frei, Pedram Ahmadiyeh,
René Glebke, Mirko Stoffers, Klaus Wehrle

RWTH Aachen University

krude@comsys.rwth-aachen.de

DFG Collaborative Research Centre 1053 – MAKI
Multi Mechanism Adaptation for the Future Internet

NetSoft, 2025-06-26

COM SYS | RWTH AACHEN UNIVERSITY

Slicing Match-Action Pipeline Resources for Multitenancy

- **Programmable Switch, e.g., Tofino → guaranteed and high throughput**
- **But, often only a subset of resources (SRAM, TCAM, action slots,…) needed**
  → Multiple tenant and programs on a shared switch

Slicing Match-Action Pipeline Resources for Multitenancy

- **Programmable Switch, e.g., Tofino → guaranteed and high throughput**
- **But, often only a subset of resources (SRAM, TCAM, action slots,…) needed**
  - → Multiple tenant and programs on a shared switch

Related Work

- **Modifiy pipeline for multitenancy [Wang et.al. 2022, MTPSA, P4VBox]**
  - ► Not applicable to existing hardware
- **Compile merged P4 program [P4Weaver, Wang et.al. 2020, P4Visor, P4Brick]**
  - ► No guaranteed pipeline resources

Slicing Match-Action Pipeline Resources for Multitenancy

- **Programmable Switch, e.g., Tofino → guaranteed and high throughput**
- **But, often only a subset of resources (SRAM, TCAM, action slots,…) needed**
  - → Multiple tenant and programs on a shared switch

Related Work

- **Modifiy pipeline for multitenancy [Wang et.al. 2022, MTPSA, P4VBox]**
  - ► Not applicable to existing hardware
- **Compile merged P4 program [P4Weaver, Wang et.al. 2020, P4Visor, P4Brick]**
  - ► No guaranteed pipeline resources

Proposal: Resource guarantees on existing pipeline

- **Divide pipeline resources into slices**
- **Compile tenant program for slice**

Slicing Match-Action Pipeline Resources for Multitenancy

- **Programmable Switch, e.g., Tofino → guaranteed and high throughput**
- **But, often only a subset of resources (SRAM, TCAM, action slots,…) needed**
  - → Multiple tenant and programs on a shared switch

Related Work

- **Modifiy pipeline for multitenancy [Wang et.al. 2022, MTPSA, P4VBox]**
  - ► Not applicable to existing hardware
- **Compile merged P4 program [P4Weaver, Wang et.al. 2020, P4Visor, P4Brick]**
  - ► No guaranteed pipeline resources

Proposal: Resource guarantees on existing pipeline

- **Divide pipeline resources into slices**
- **Compile tenant program for slice**

Implemented for Tofino

- **Eval on real P4 programs**
- **Sourcecode on GitHub**

IXP Use Case: Virtualized Edge Routers

IXPs (e.g., DeCIX) need a lot of space and energy

Currently: Each customer a physical router box

Instead: Upload P4 router to shared switch

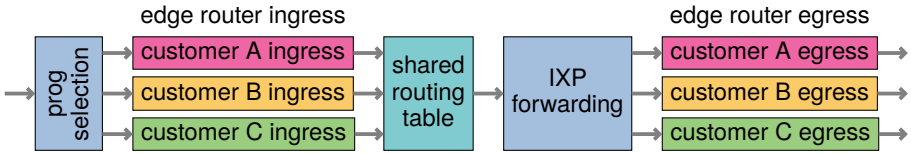
IXP Use Case: Virtualized Edge Routers

IXPs (e.g., DeCIX) need a lot of space and energy

Currently: Each customer a physical router box

Instead: Upload P4 router to shared switch

IXP virtualized edge routers



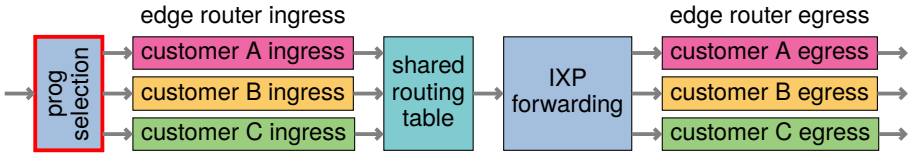
IXP Use Case: Virtualized Edge Routers

IXPs (e.g., DeCIX) need a lot of space and energy

Currently: Each customer a physical router box

Instead: Upload P4 router to shared switch

IXP virtualized edge routers



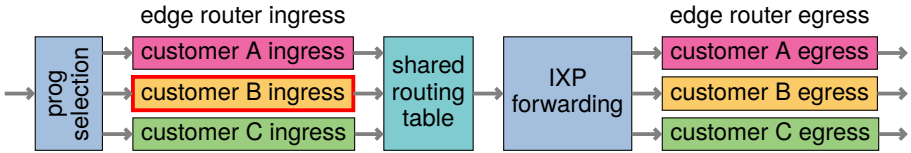
IXP Use Case: Virtualized Edge Routers

IXPs (e.g., DeCIX) need a lot of space and energy

Currently: Each customer a physical router box

Instead: Upload P4 router to shared switch

IXP virtualized edge routers



- At most one router per ingress/egress

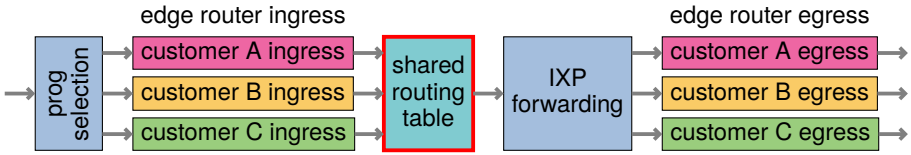
IXP Use Case: Virtualized Edge Routers

IXPs (e.g., DeCIX) need a lot of space and energy

Currently: Each customer a physical router box

Instead: Upload P4 router to shared switch

IXP virtualized edge routers



- At most one router per ingress/egress
- Sharing common routines and data

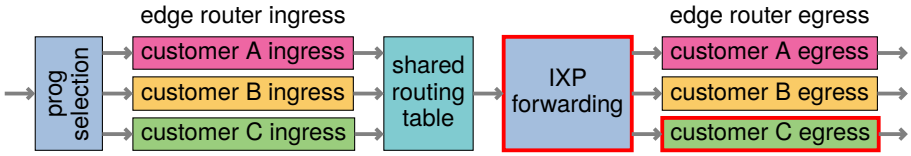
IXP Use Case: Virtualized Edge Routers

IXPs (e.g., DeCIX) need a lot of space and energy

Currently: Each customer a physical router box

Instead: Upload P4 router to shared switch

IXP virtualized edge routers



- At most one router per ingress/egress
- Sharing common routines and data

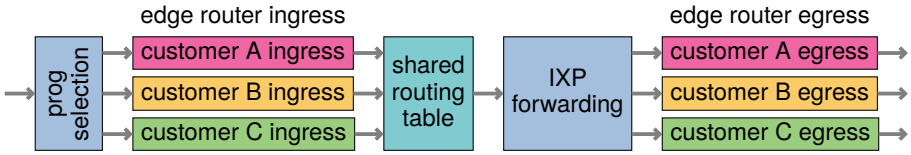
IXP Use Case: Virtualized Edge Routers

IXPs (e.g., DeCIX) need a lot of space and energy

Currently: Each customer a physical router box

Instead: Upload P4 router to shared switch

IXP virtualized edge routers

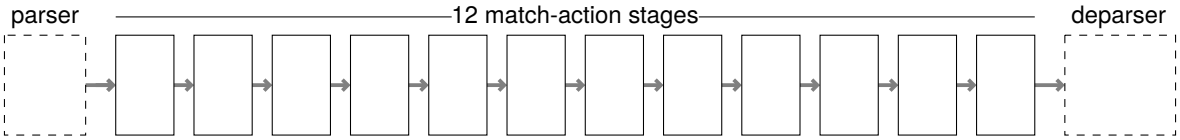


- At most one router per ingress/egress
- Sharing common routines and data

Requirements for switch sharing

- **Safely composing programs** [P4Weaver, Wang et.al. 2020, P4Visor, P4Brick]
- **(Our Contribution): Guaranteed switch pipeline resources for each customer**

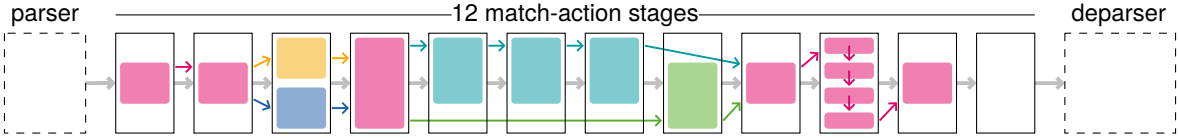
Pipeline Resources



The Match-Action Pipeline

- Parser, m-a stages, deparser
- Separate resources at each stage (SRAM, TCAM, action slots, …)
- Each stage divided into logical tables

Pipeline Resources

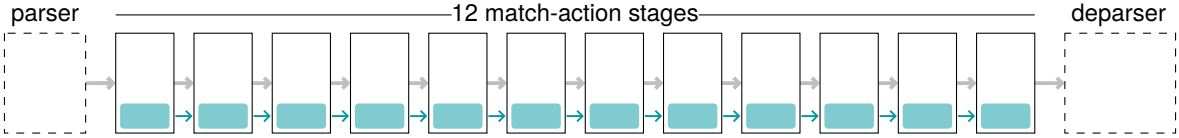


The Match-Action Pipeline

- Parser, m-a stages, deparser
- Separate resources at each stage (SRAM, TCAM, action slots, …)
- Each stage divided into logical tables

- **Compiler: fits P4 tables into stages (NP-hard)**
- **Related Work: compile merged P4 program**
  - If it does not fit, which tenant should be blamed?

Pipeline Resources

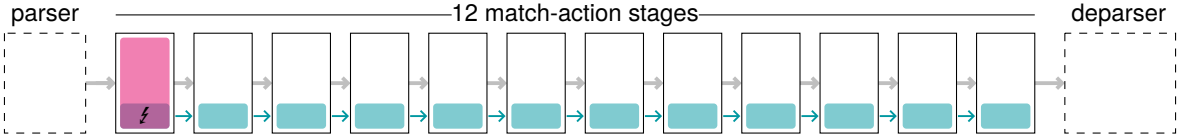


The Match-Action Pipeline

- Parser, m-a stages, deparser
- Separate resources at each stage (SRAM, TCAM, action slots, …)
- Each stage divided into logical tables

- **Compiler: fits P4 tables into stages (NP-hard)**
- **Related Work: compile merged P4 program**
  - If it does not fit, which tenant should be blamed?
- **Example:**
  - Long Program needs all stages

Pipeline Resources

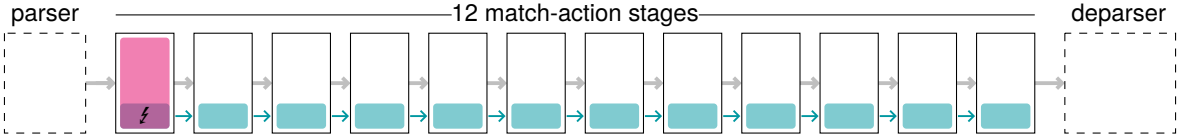


The Match-Action Pipeline

- Parser, m-a stages, deparser
- Separate resources at each stage (SRAM, TCAM, action slots, …)
- Each stage divided into logical tables

- **Compiler: fits P4 tables into stages (NP-hard)**
- **Related Work: compile merged P4 program**
  - If it does not fit, which tenant should be blamed?
- **Example:**
  - Long Program needs all stages
  - Wide Program needs full stage

Pipeline Resources



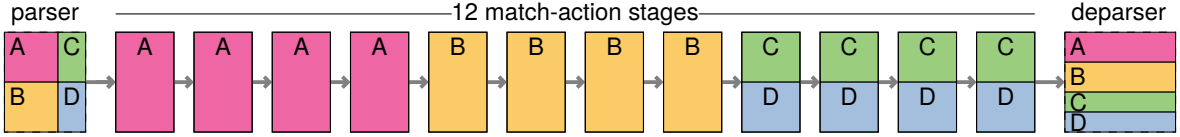
The Match-Action Pipeline

- Parser, m-a stages, deparser
- Separate resources at each stage (SRAM, TCAM, action slots, …)
- Each stage divided into logical tables

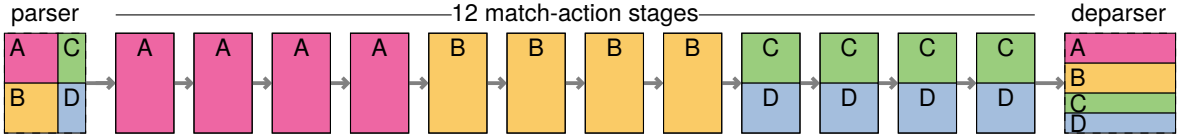
- **Compiler: fits P4 tables into stages (NP-hard)**
- **Related Work: compile merged P4 program**
  - If it does not fit, which tenant should be blamed?
- **Example:**
  - Long Program needs all stages
  - Wide Program needs full stage

Tenants should know in advance the acceptable resource usage and program shape

Slicing Match-Action Pipeline Resources



Slicing Match-Action Pipeline Resources



Switch Provider

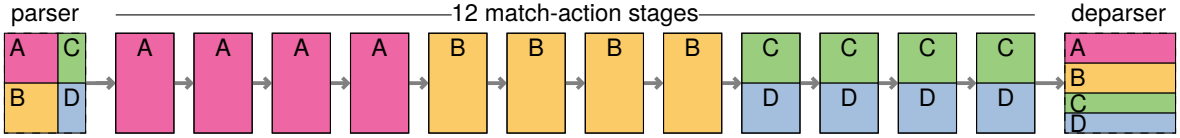
divides pipeline into slices

negotiate pipeline slice

Tenant

creates or adapts program

Slicing Match-Action Pipeline Resources



Switch Provider

divides pipeline into slices

negotiate pipeline slice

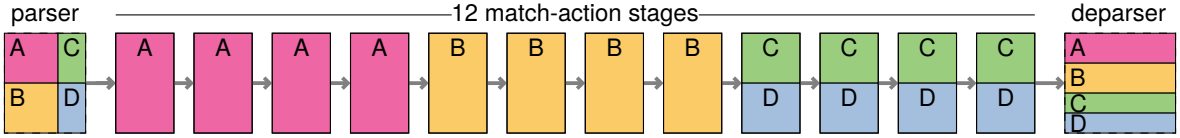
submit compiled program

Tenant

creates or adapts program

compiles program for slice

Slicing Match-Action Pipeline Resources



Switch Provider

divides pipeline into slices

checks if program is within slice

negotiate pipeline slice

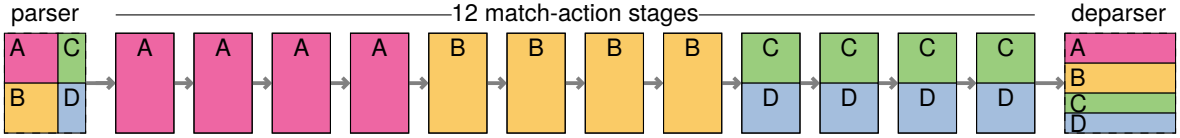
submit compiled program

Tenant

creates or adapts program

compiles program for slice

Slicing Match-Action Pipeline Resources



Switch Provider

divides pipeline into slices

↓

checks if program is within slice

↓

composed with other programs

negotiate pipeline slice ↔

← submit compiled program

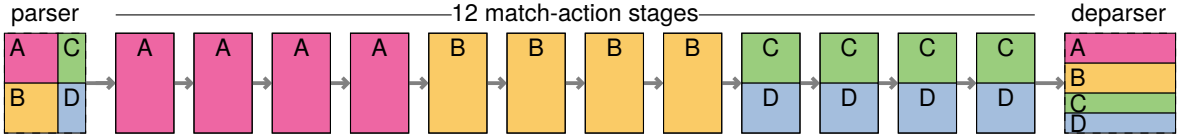
Tenant

creates or adapts program

↓

compiles program for slice

Slicing Match-Action Pipeline Resources



Switch Provider

divides pipeline into slices

checks if program is within slice

composed with other programs

negotiate pipeline slice

submit compiled program

Tenant

creates or adapts program

compiles program for slice

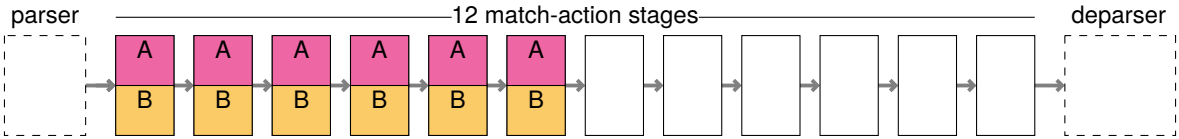
Slice checking guarantees all other tenants the availability of resources

Program and Slice Shapes

Slices are only useful if the tenant program can be shaped to fit into the slice

Same slice for each tenant → Little flexibility

Individual slice shapes → Unallocated leftover resources

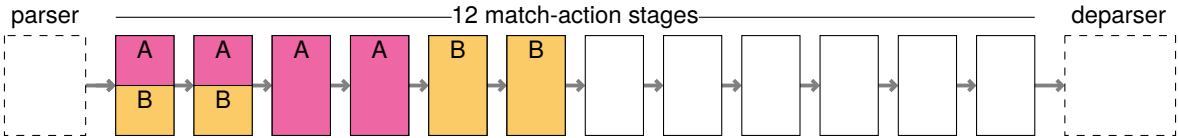


Program and Slice Shapes

Slices are only useful if the tenant program can be shaped to fit into the slice

Same slice for each tenant → Little flexibility

Individual slice shapes → Unallocated leftover resources

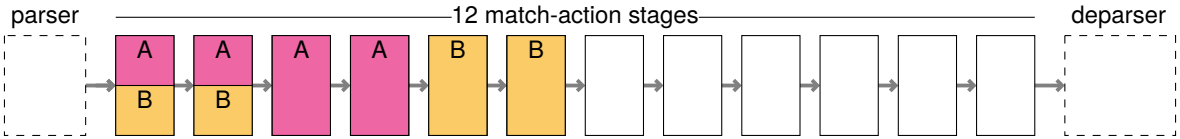


Program and Slice Shapes

Slices are only useful if the tenant program can be shaped to fit into the slice

Same slice for each tenant → Little flexibility

Individual slice shapes → Unallocated leftover resources



Reordering undivided stages

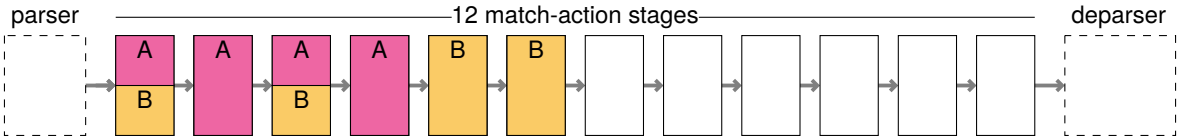
- **A tenant may use undivided stages at any place in his program**
- **But, only undivided**

Program and Slice Shapes

Slices are only useful if the tenant program can be shaped to fit into the slice

Same slice for each tenant → Little flexibility

Individual slice shapes → Unallocated leftover resources



Reordering undivided stages

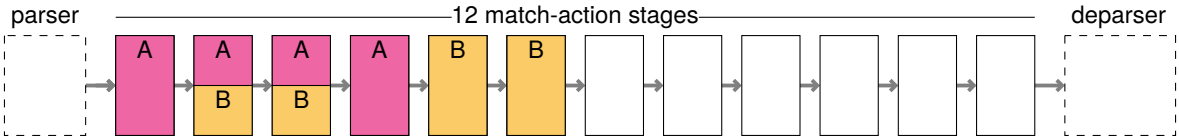
- **A tenant may use undivided stages at any place in his program**
- **But, only undivided**

Program and Slice Shapes

Slices are only useful if the tenant program can be shaped to fit into the slice

Same slice for each tenant → Little flexibility

Individual slice shapes → Unallocated leftover resources



Reordering undivided stages

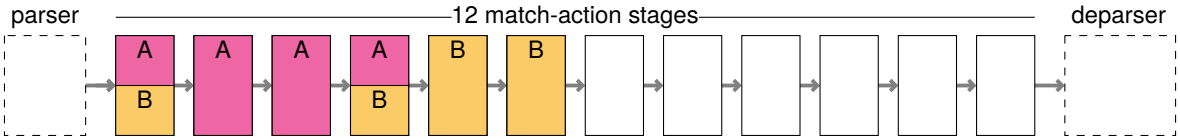
- **A tenant may use undivided stages at any place in his program**
- **But, only undivided**

Program and Slice Shapes

Slices are only useful if the tenant program can be shaped to fit into the slice

Same slice for each tenant → Little flexibility

Individual slice shapes → Unallocated leftover resources



Reordering undivided stages

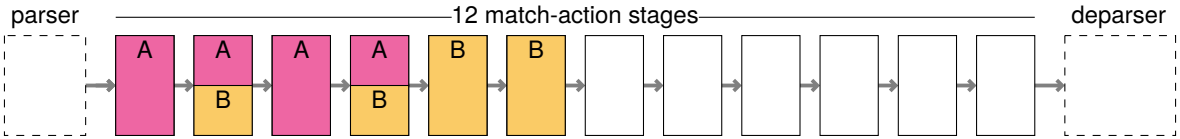
- **A tenant may use undivided stages at any place in his program**
- **But, only undivided**

Program and Slice Shapes

Slices are only useful if the tenant program can be shaped to fit into the slice

Same slice for each tenant → Little flexibility

Individual slice shapes → Unallocated leftover resources



Reordering undivided stages

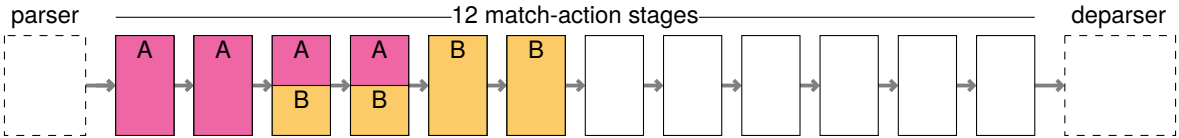
- **A tenant may use undivided stages at any place in his program**
- **But, only undivided**

Program and Slice Shapes

Slices are only useful if the tenant program can be shaped to fit into the slice

Same slice for each tenant → Little flexibility

Individual slice shapes → Unallocated leftover resources



Reordering undivided stages

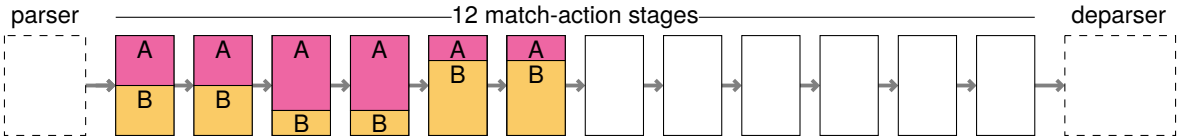
- **A tenant may use undivided stages at any place in his program**
- **But, only undivided**

Program and Slice Shapes

Slices are only useful if the tenant program can be shaped to fit into the slice

Same slice for each tenant → Little flexibility

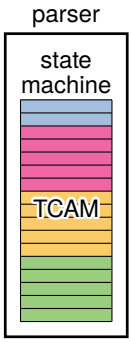
Individual slice shapes → Unallocated leftover resources



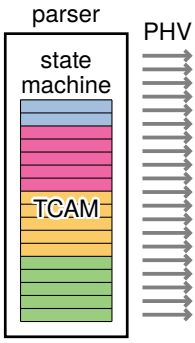
Reordering undivided stages

- **A tenant may use undivided stages at any place in his program**
- **But, only undivided**

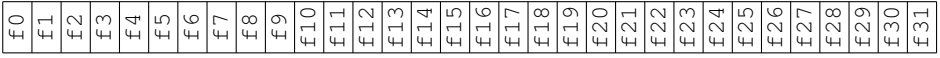
Slicing Individual Resources



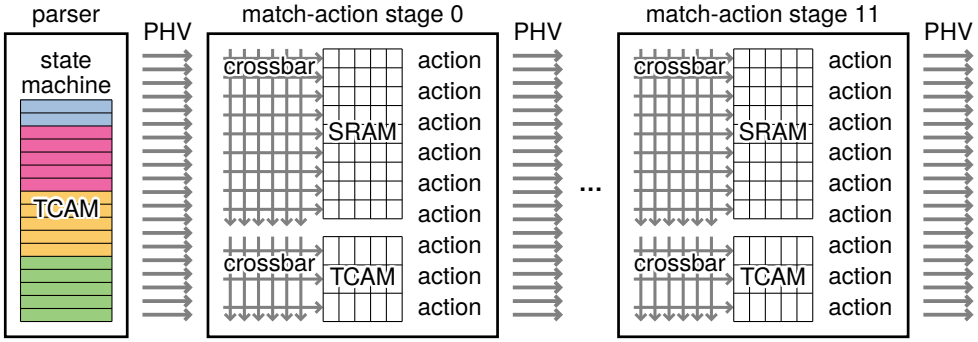
Slicing Individual Resources



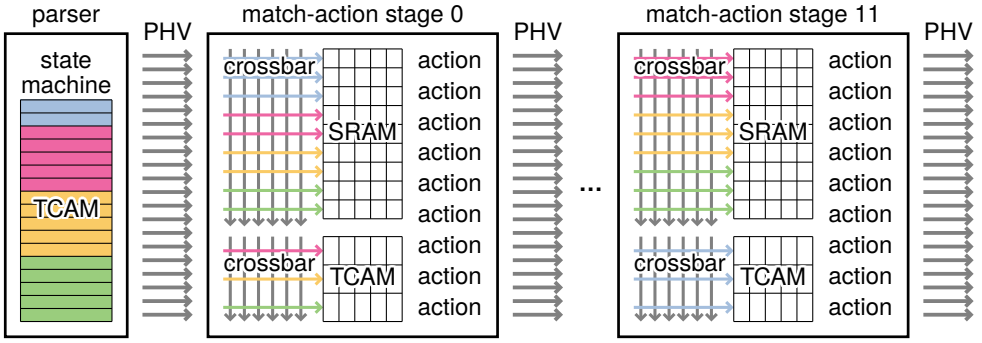
packet header vector (PHV)



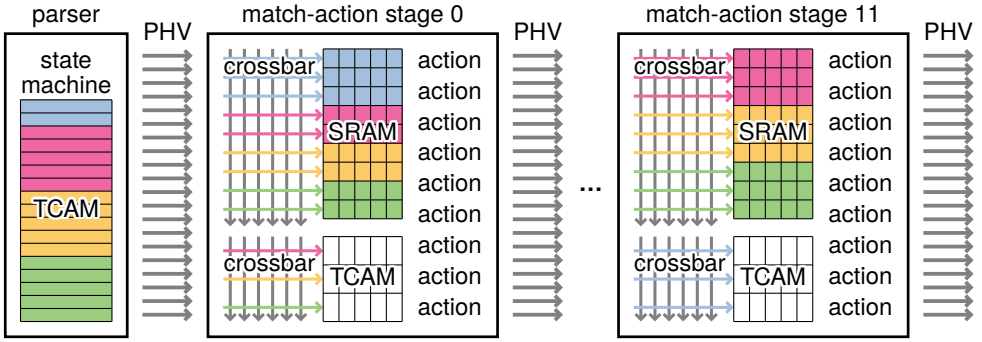
Slicing Individual Resources



Slicing Individual Resources



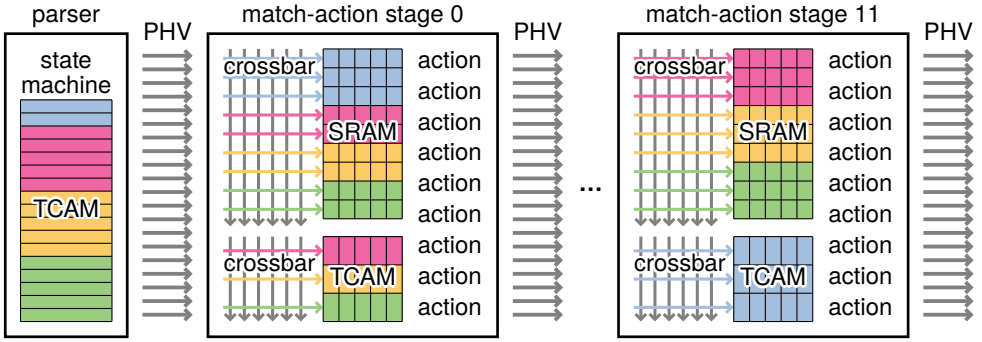
Slicing Individual Resources



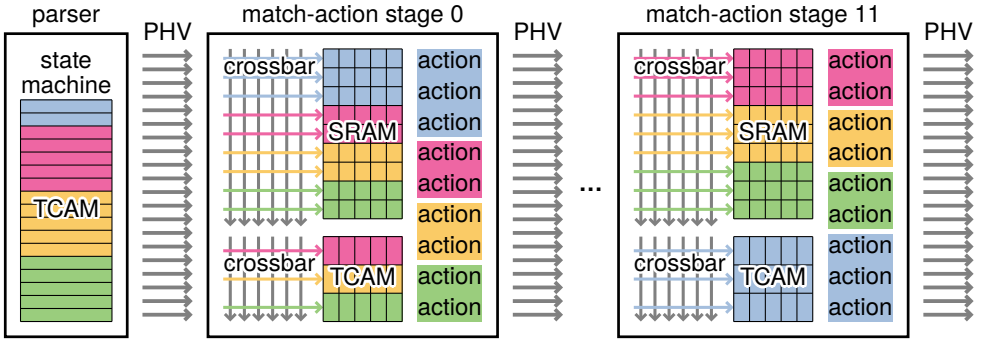
packet header vector (PHV)

f0	f1	f2	f3	f4	f5	f6	f7	f8	f9	f10	f11	f12	f13	f14	f15	f16	f17	f18	f19	f20	f21	f22	f23	f24	f25	f26	f27	f28	f29	f30	f31

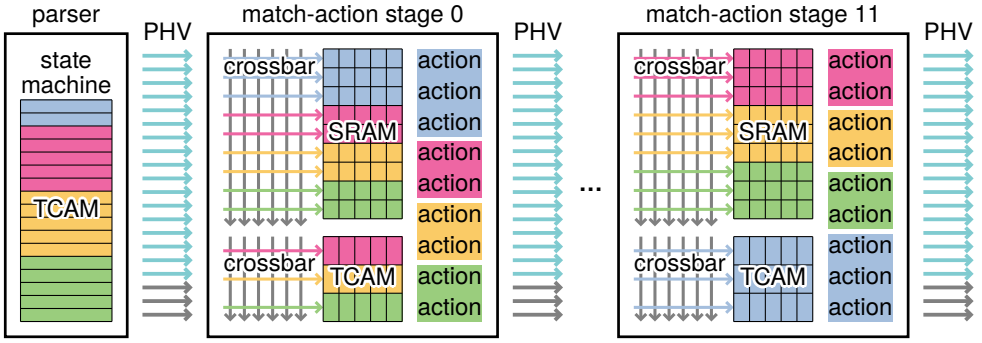
Slicing Individual Resources



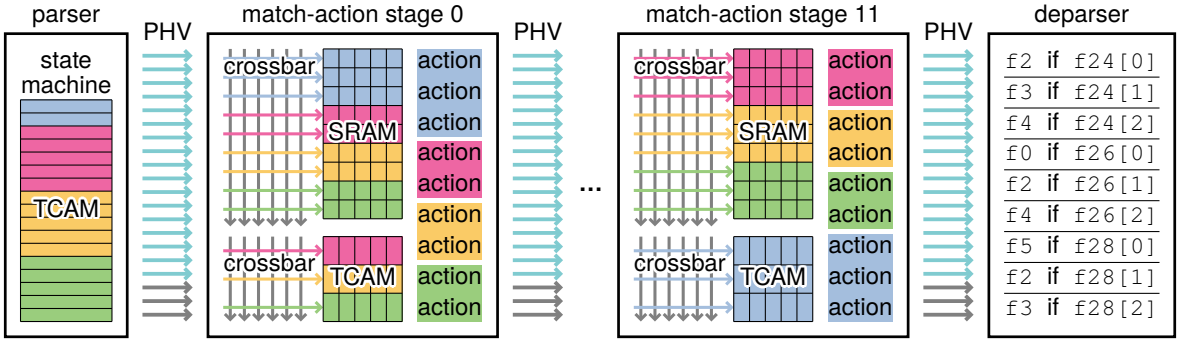
Slicing Individual Resources



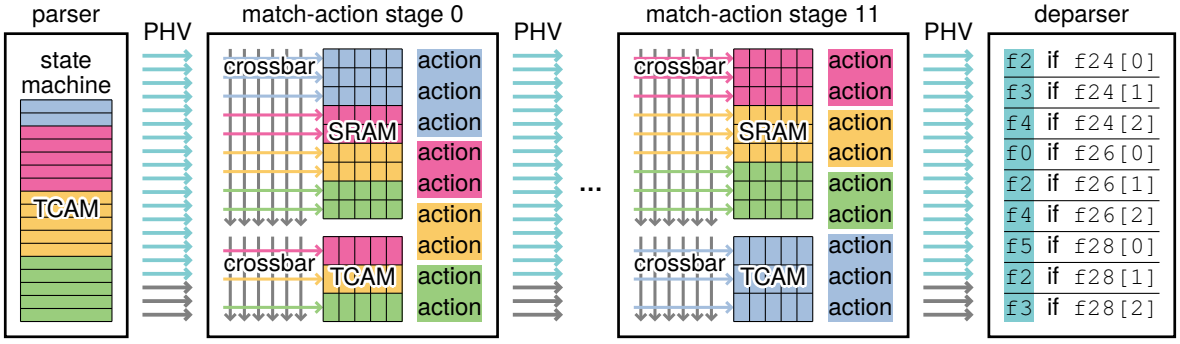
Slicing Individual Resources



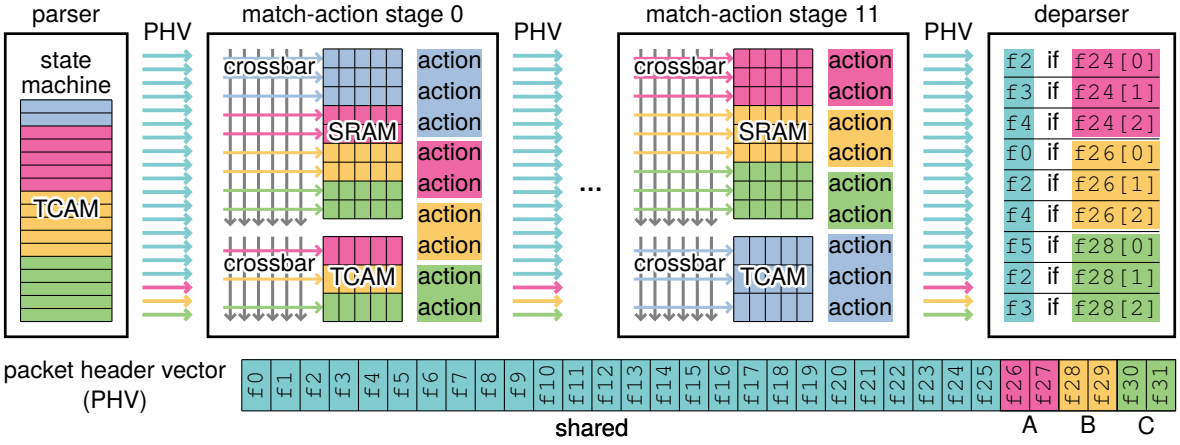
Slicing Individual Resources



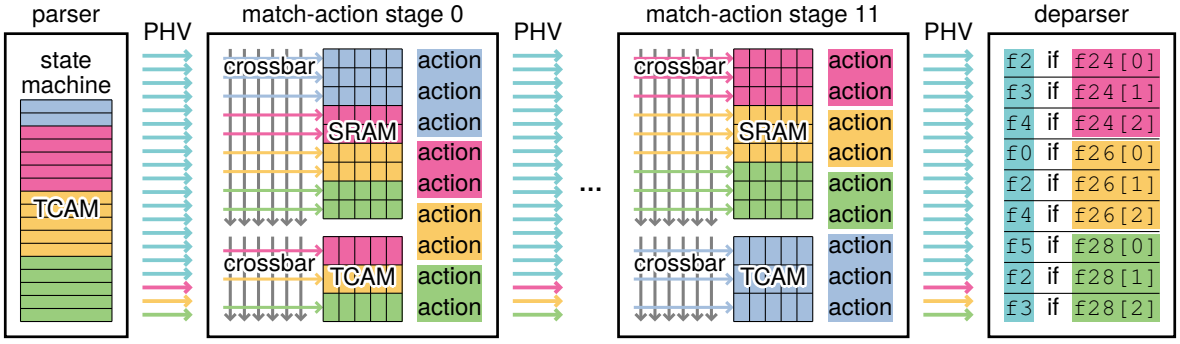
Slicing Individual Resources



Slicing Individual Resources



Slicing Individual Resources



Most of the packet header vector can be shared, except for the validity bits

Slicing Tofino

- **Each pipeline has 16x 100 GbE port → target up to 16 slices per pipeline**

Parser
- Up to 256 slices

Deparser
- Up to 192 slices

Packet header vector
- Up to 48 slices, each with 16+16 validity bits and 128 shared fields

Slicing Tofino

- **Each pipeline has 16x 100 GbE port $\rightarrow$ target up to 16 slices per pipeline**

Parser
- Up to 256 slices

Deparser
- Up to 192 slices

Packet header vector
- Up to 48 slices, each with 16+16 validity bits and 128 shared fields

Match Action Stages
- Crossbars only dividable into three identical slices

Stage Division	$1/1$
	$1/2$
	$1/3$

Slicing Tofino

- **Each pipeline has 16x 100 GbE port $\rightarrow$ target up to 16 slices per pipeline**

Parser
- Up to 256 slices

Deparser
- Up to 192 slices

Packet header vector
- Up to 48 slices, each with 16+16 validity bits and 128 shared fields

Match Action Stages
- Crossbars only dividable into three identical slices

		Stages per Slice					
		12	6	4	3	2	1
Stage Division	$1/1$						
	$1/2$						
	$1/3$						

Slicing Tofino

- **Each pipeline has 16x 100 GbE port → target up to 16 slices per pipeline**

Parser

- Up to 256 slices

Deparser

- Up to 192 slices

Packet header vector

- Up to 48 slices, each with 16+16 validity bits and 128 shared fields

Match Action Stages

- Crossbars only dividable into three identical slices

		Stages per Slice						
		12	6	4	3	2	1	
Stage Division	$^1/_1$	1	2	3	4	6	12	*Number of Slices*
	$^1/_2$	2	4	6	8	12	24	
	$^1/_3$	3	6	9	12	18	36	

- Up to 36 identical slices

Slicing Tofino

- **Each pipeline has 16x 100 GbE port → target up to 16 slices per pipeline**

Parser
- Up to 256 slices

Deparser
- Up to 192 slices

Packet header vector
- Up to 48 slices, each with 16+16 validity bits and 128 shared fields

Match Action Stages
- Crossbars only dividable into three identical slices

		Stages per Slice						
		12	6	4	3	2	1	
Stage Division	$1/1$	1	2	3	4	6	12	*Number of Slices*
	$1/2$	2	4	6	8	12	24	
	$1/3$	3	6	9	12	18	36	

- Up to 36 identical slices

Implemented slice checking and program composition on Barefoot Assembly (.bfa) files

Compiling P4 for a Slice

Implemented 3 variants

1. **Modified bf-p4c**
   - No PHV constraints
2. **SMT table placement**
   - Reordering stages
3. **Add additional tables**
   - Unmodified compiler

Compiling P4 for a Slice

Implemented 3 variants

1. **Modified bf-p4c**
   - No PHV constraints
2. **SMT table placement**
   - Reordering stages
3. **Add additional tables**
   - Unmodified compiler

Compiled 19 P4 programs from various sources

Program	Smallest Slice
RTT	$12 \cdot 1/1$
firewall	$6 \cdot 1/1$, $12 \cdot 1/2$
chord	$4 \cdot 1/1$, $12 \cdot 1/3$
accelerator	$4 \cdot 1/2$
source_routing	$4 \cdot 1/3$
flowlet	$2 \cdot 1/2$
qos	$2 \cdot 1/3$
ecn	$2 \cdot 1/3$
MPLS	$1 \cdot 1/2$

Program	Smallest Slice
IXP_routing	$4 \cdot 1/1$
aes	$12 \cdot 1/3$
conquest_b.	$2 \cdot 1/1$, $4 \cdot 1/2$
speedtest	$2 \cdot 1/1$, $4 \cdot 1/2$, $6 \cdot 1/3$
a._tunnel	$2 \cdot 1/2$
firewall_s.	$1 \cdot 1/1$, $2 \cdot 1/2$, $3 \cdot 1/3$
calc	$1 \cdot 1/2$
single_table	$1 \cdot 1/3$
VXLAN	$3 \cdot 1/2$
multicast	$1 \cdot 1/2$

Compiling P4 for a Slice

Implemented 3 variants

1. Modified bf-p4c
- No PHV constraints

2. SMT table placement
- Reordering stages

3. Add additional tables
- Unmodified compiler

Compiled 19 P4 programs from various sources

Program	Prog. Instances **Slicing**
RTT	1
firewall	2
chord	3
accelerator	6
source_routing	9
flowlet	12
qos	18
ecn	18
MPLS	24

Program	Prog. Instances **Slicing**
IXP_routing	3
aes	3
conquest_b.	6
speedtest	6
a._tunnel	12
firewall_s.	12
calc	24
single_table	36
VXLAN	**8**
multicast	**24**

Compiling P4 for a Slice

Implemented 3 variants

1. Modified bf-p4c
- No PHV constraints

2. SMT table placement
- Reordering stages

3. Add additional tables
- Unmodified compiler

X. Merge P4 programs
- Similar to related work
- Compile merged P4

Compiled 19 P4 programs from various sources

Program	Prog. Instances **Slicing**	Merge
RTT	1	
firewall	2	
chord	3	
accelerator	6	
source_routing	9	
flowlet	12	
qos	18	
ecn	18	
MPLS	24	

Program	Prog. Instances **Slicing**
IXP_routing	3
aes	3
conquest_b.	6
speedtest	6
a._tunnel	12
firewall_s.	12
calc	24
single_table	36
VXLAN	**8**
multicast	**24**

Compiling P4 for a Slice

Implemented 3 variants

1. Modified bf-p4c
- No PHV constraints

2. SMT table placement
- Reordering stages

3. Add additional tables
- Unmodified compiler

X. Merge P4 programs
- Similar to related work
- Compile merged P4

Compiled 19 P4 programs from various sources

Program	Prog. Instances **Slicing**		Merge
RTT	1	<	2
firewall	2	<	4
chord	3	<	7
accelerator	6	<	9
source_routing	9		
flowlet	12		
qos	18		
ecn	18		
MPLS	24		

Program	Prog. Instances **Slicing**
IXP_routing	3
aes	3
conquest_b.	6
speedtest	6
a._tunnel	12
firewall_s.	12
calc	24
single_table	36
VXLAN	**8**
multicast	**24**

Compiling P4 for a Slice

Implemented 3 variants

1. Modified bf-p4c
- No PHV constraints

2. SMT table placement
- Reordering stages

3. Add additional tables
- Unmodified compiler

X. Merge P4 programs
- Similar to related work
- Compile merged P4

Compiled 19 P4 programs from various sources

Program	Prog. Instances **Slicing**		Merge
RTT	1	$<$	2
firewall	2	$<$	4
chord	3	$<$	7
accelerator	6	$<$	9
source_routing	9	$\approx$	10
flowlet	12	$\approx$	13
qos	18	$\approx$	21
ecn	18	$\approx$	21
MPLS	24	$\approx$	32

Program	Prog. Instances **Slicing**	Merge
IXP_routing	3	
aes	3	
conquest_b.	6	
speedtest	6	
a._tunnel	12	
firewall_s.	12	
calc	24	
single_table	36	
VXLAN	**8**	
multicast	**24**	

Compiling P4 for a Slice

Implemented 3 variants

1. Modified bf-p4c
- No PHV constraints

2. SMT table placement
- Reordering stages

3. Add additional tables
- Unmodified compiler

X. Merge P4 programs
- Similar to related work
- Compile merged P4

Compiled 19 P4 programs from various sources

Program	Prog. Instances Slicing		Merge
RTT	1	$<$	2
firewall	2	$<$	4
chord	3	$<$	7
accelerator	6	$<$	9
source_routing	9	$\approx$	10
flowlet	12	$\approx$	13
qos	18	$\approx$	21
ecn	18	$\approx$	21
MPLS	24	$\approx$	32

Program	Prog. Instances Slicing		Merge
IXP_routing	3	$=$	3
aes	3	$=$	3
conquest_b.	6	$=$	6
speedtest	6	$=$	6
a._tunnel	12	$=$	12
firewall_s.	12	$=$	12
calc	24	$=$	24
single_table	36	$=$	36
VXLAN	**8**		
multicast	**24**		

Compiling P4 for a Slice

Implemented 3 variants

1. Modified bf-p4c
- No PHV constraints

2. SMT table placement
- Reordering stages

3. Add additional tables
- Unmodified compiler

X. Merge P4 programs
- Similar to related work
- Compile merged P4

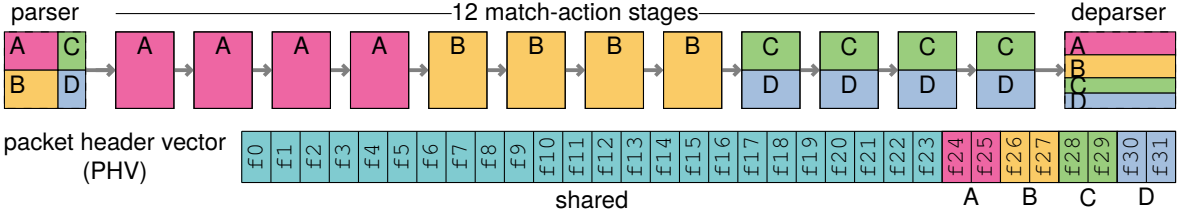
Compiled 19 P4 programs from various sources

Program	Prog. Instances Slicing		Merge
RTT	1	$<$	2
firewall	2	$<$	4
chord	3	$<$	7
accelerator	6	$<$	9
source_routing	9	$\approx$	10
flowlet	12	$\approx$	13
qos	18	$\approx$	21
ecn	18	$\approx$	21
MPLS	24	$\approx$	32

Program	Prog. Instances Slicing		Merge
IXP_routing	3	$=$	3
aes	3	$=$	3
conquest_b.	6	$=$	6
speedtest	6	$=$	6
a._tunnel	12	$=$	12
firewall_s.	12	$=$	12
calc	24	$=$	24
single_table	36	$=$	36
VXLAN	**8**	$>$	6
multicast	**24**	$>$	15

Conclusion

- **Resource guarantees for multitenany on match-action pipelines**
  - ▶ Provider divides pipeline into slices
  - ▶ Provider checks program for slice
  - ▶ Tenant compiles program for slice
- **Our packet header vector sharing approach can increase switch utilization**



Full implementation, all example programs, and scripted eval on GitHub