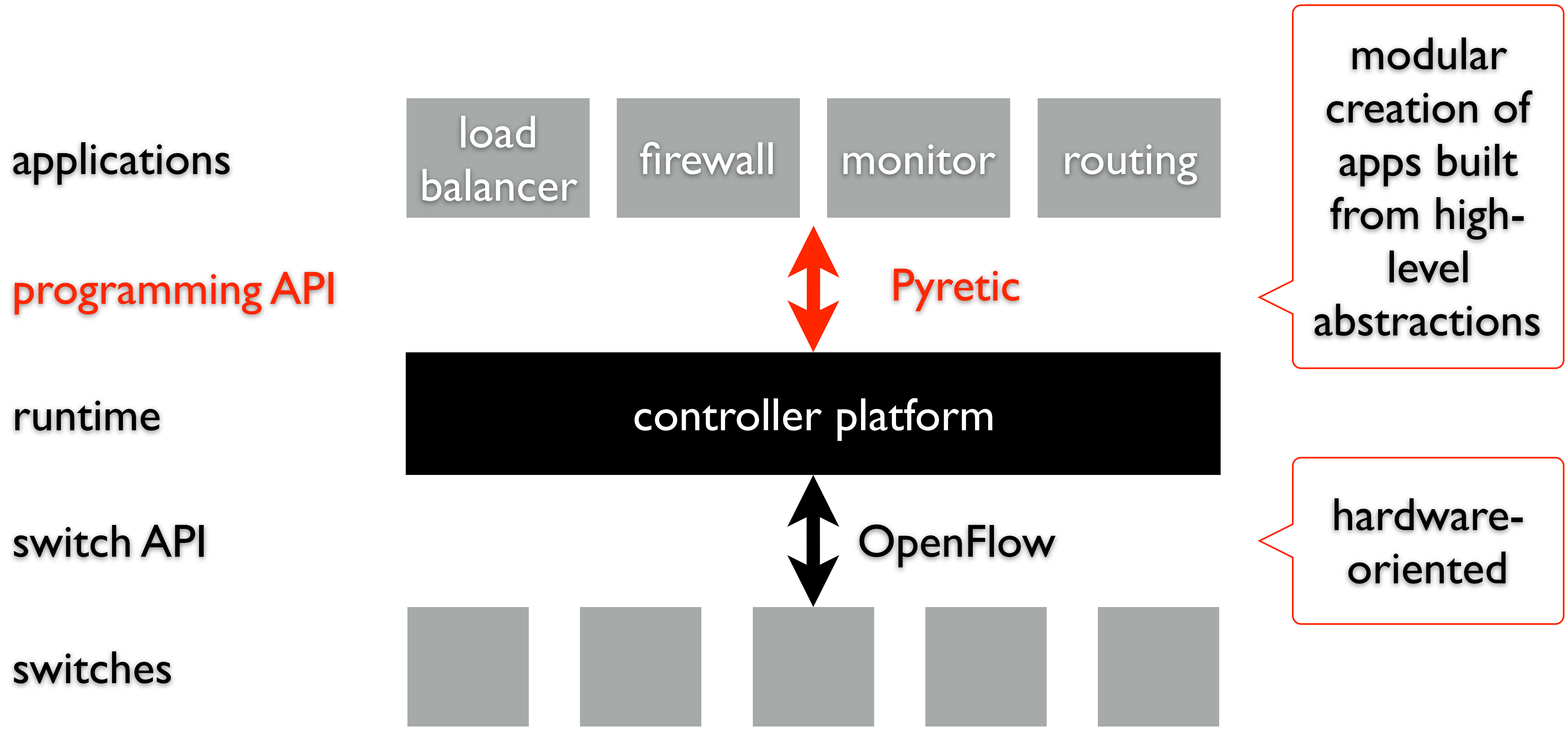
lecture 11: composing controllers

5590: software defined networking

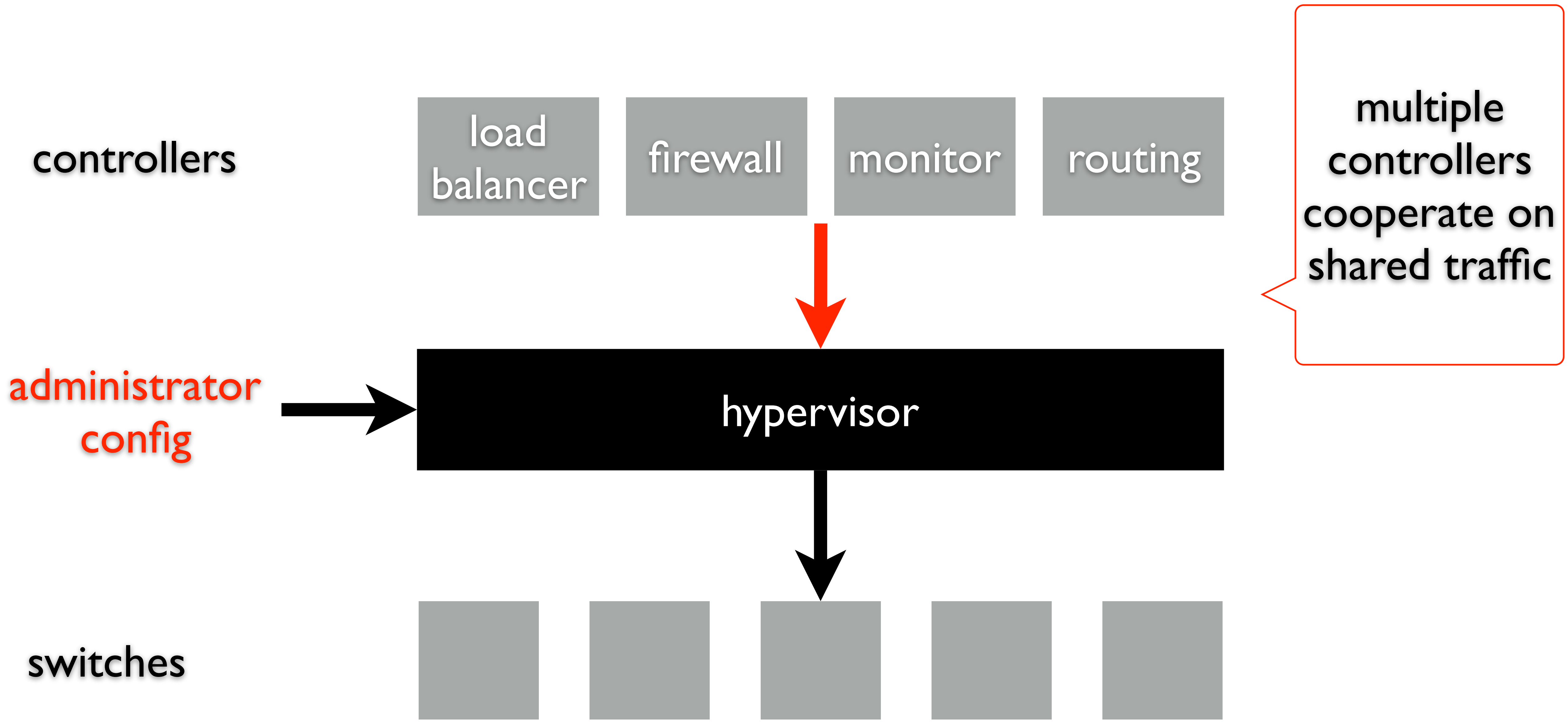
anduo wang, Temple University

TTLMAN 401B, R 17:30-20:00

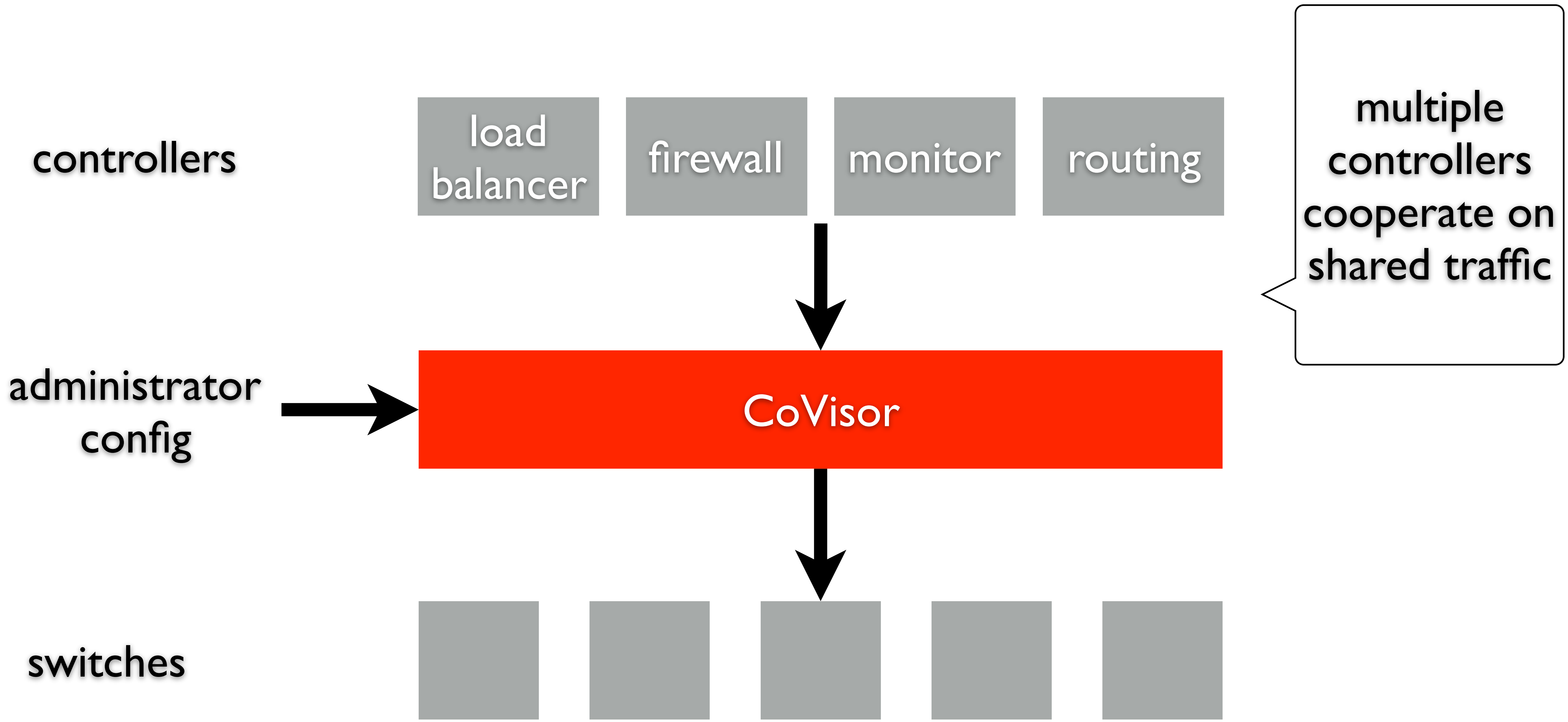
Pyretic: composing policies



composing controllers



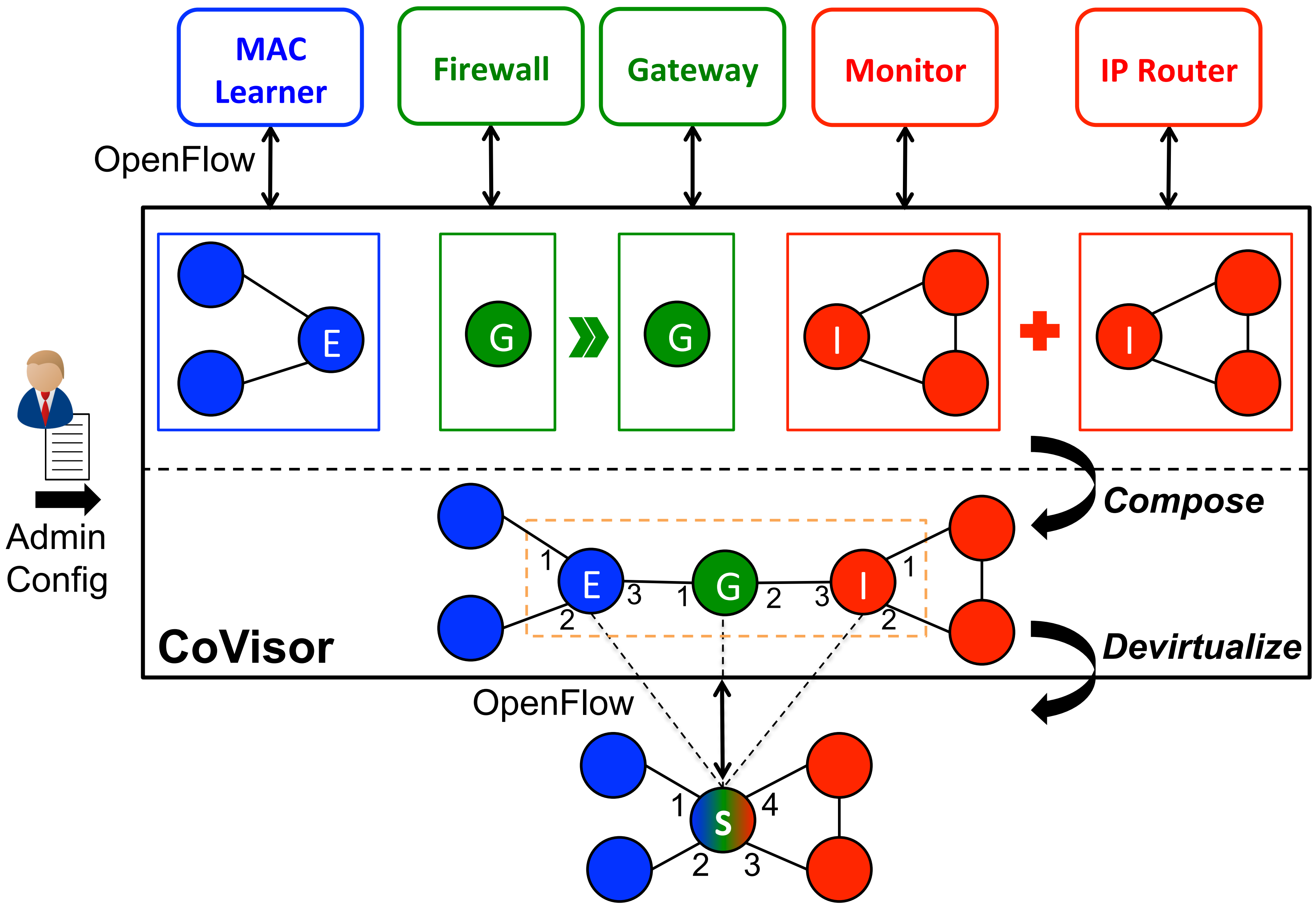
CoVisor



challenges and technical contribution

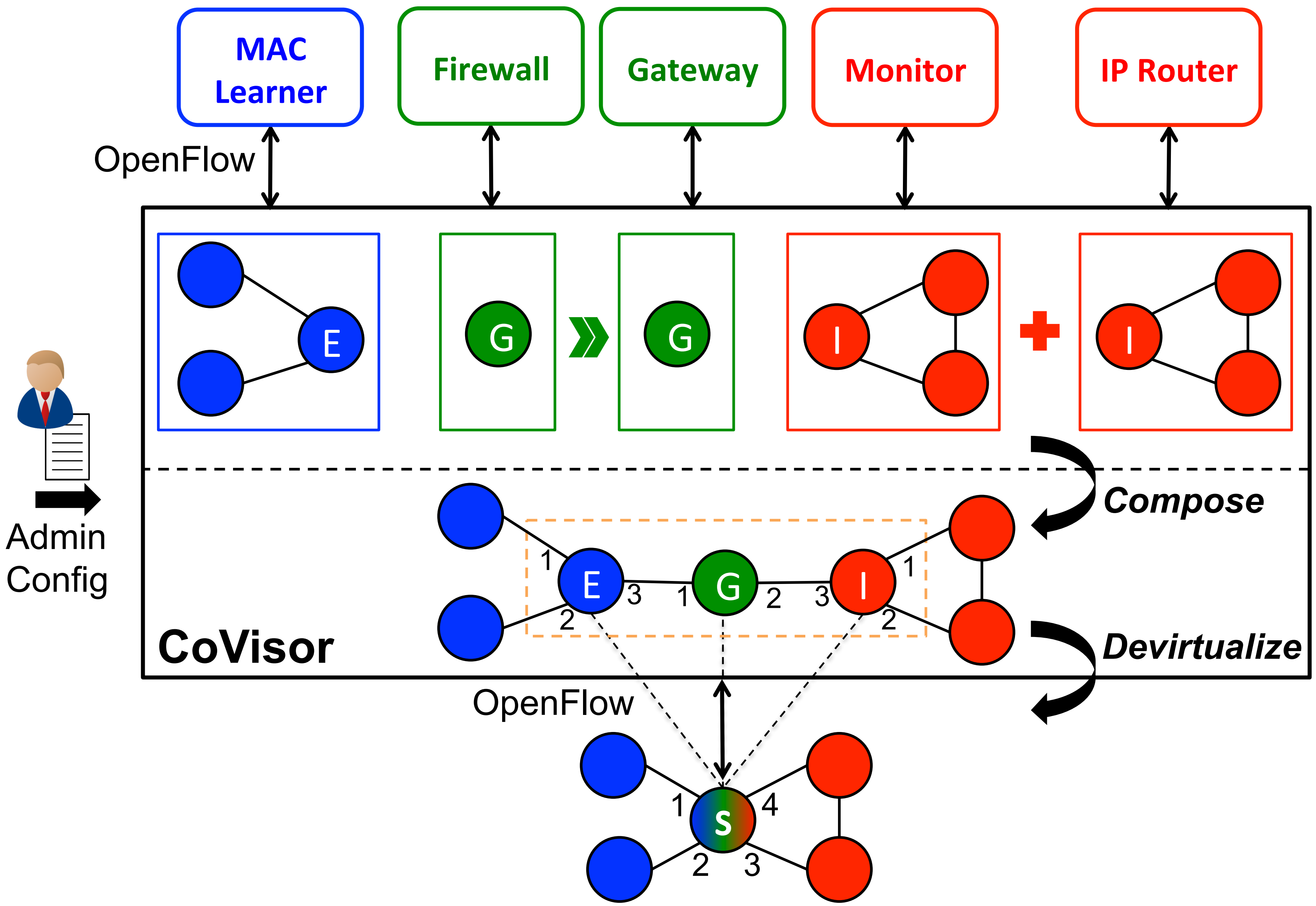
- efficient algorithms

CoVisor



MAC Learner
Firewall
Gateway
Monitor
IP Router
OpenFlow
E
G
G
I
I
Compose
Devirtualize
Admin Config
CoVisor
E
G
I
1
2
3
1
2
3
1
2
OpenFlow
S
1
2
3
4

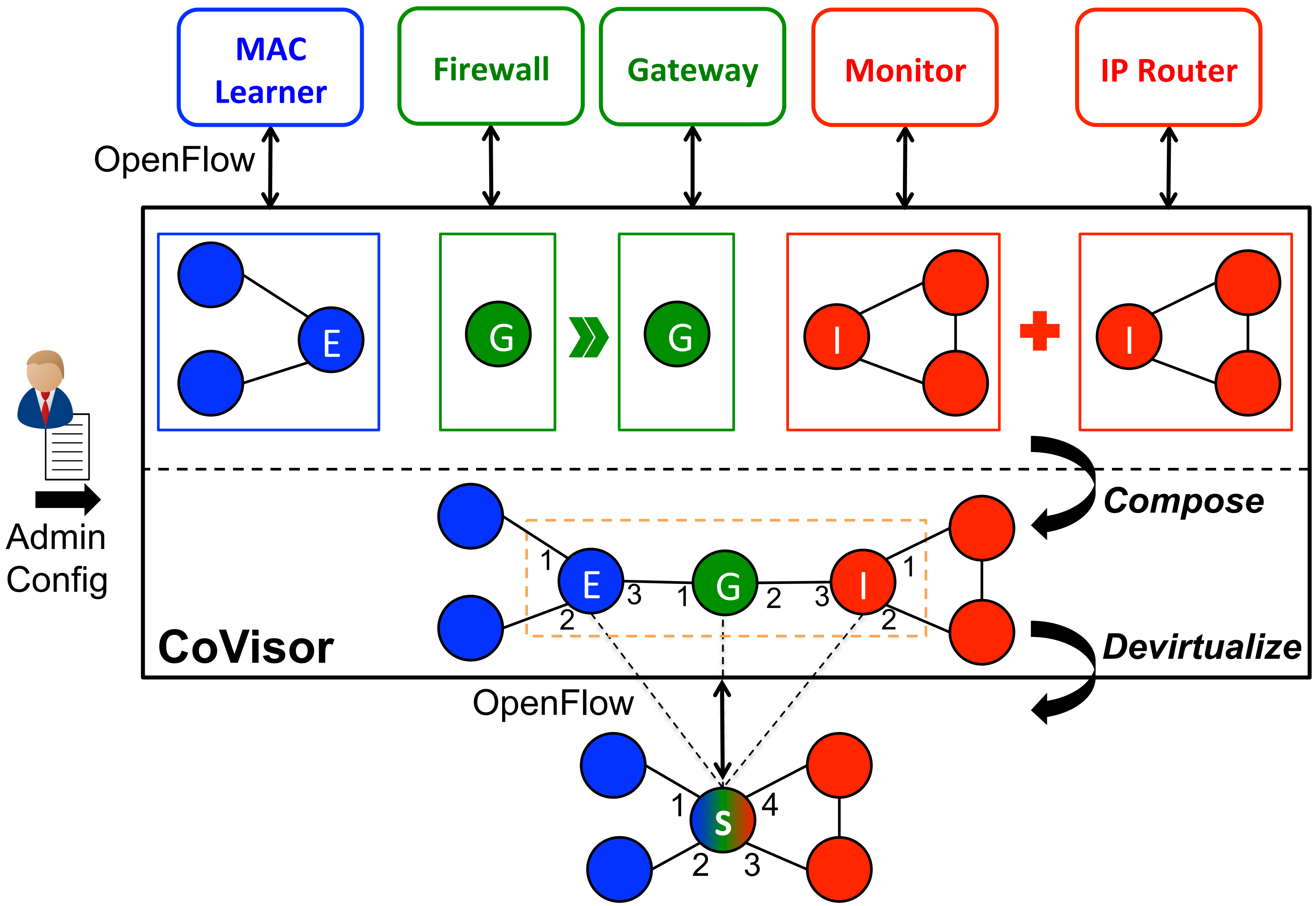
CoVisor



assemble multiple controllers

- parallel, sequential, override

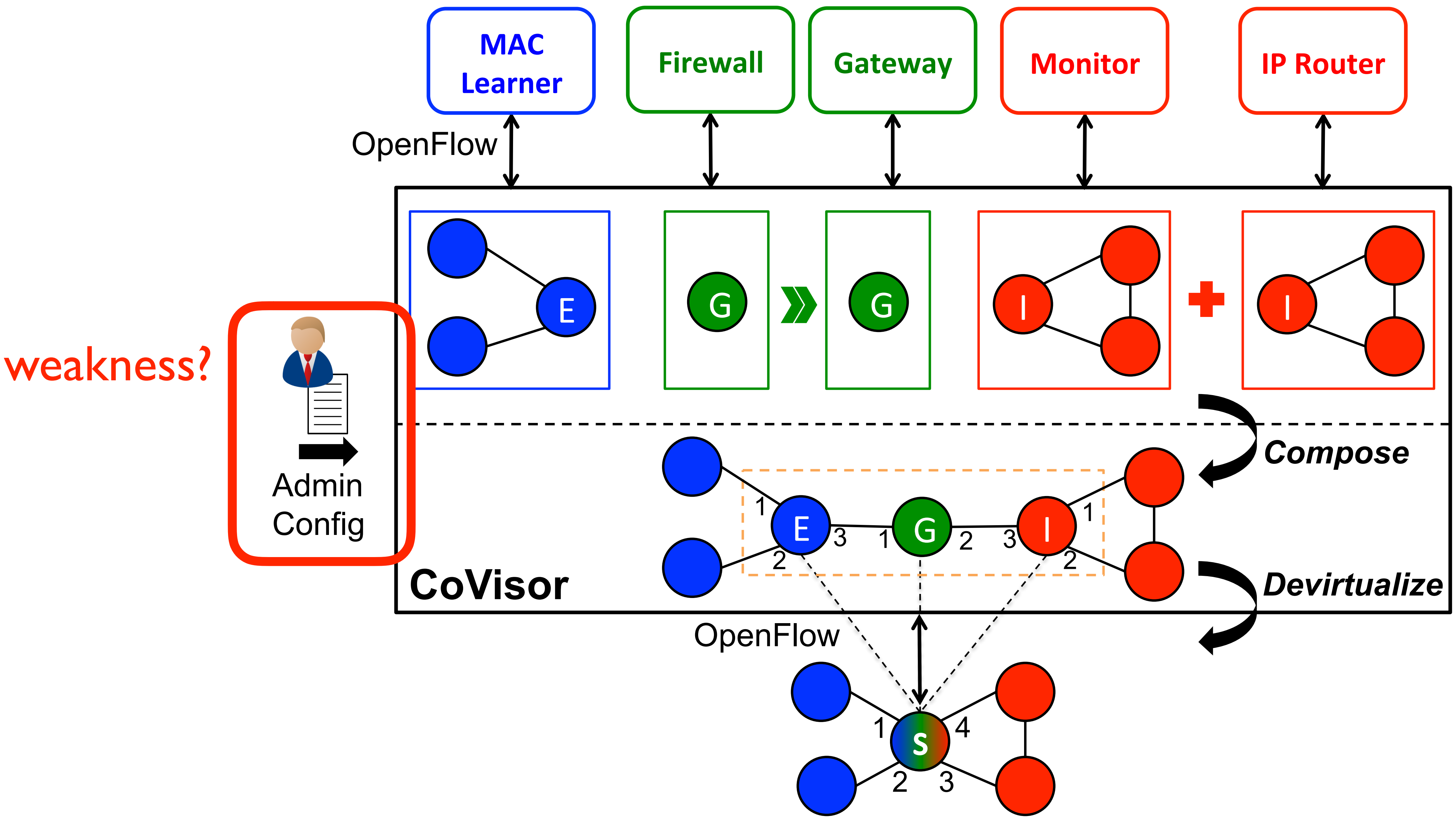
CoVisor



abstract topology

- customer virtual topology to each controller

CoVisor



protection

- fine-grained control over how a controller can operate

administrator role

administrator role

configure CoVisor to compose policies

administrator role

configure CoVisor to compose policies

- *manual* spec: $T_1 + T_2$, $T_1 > T_2$, $T_1 \triangleright T_2$

administrator role

configure CoVisor to compose policies

- *manual*
- *proactive* incremental compilation, optimization

administrator role

configure CoVisor to compose policies

- *manual*
- *proactive*

virtualize the network, sets packet-processing constraints

administrator role

configure CoVisor to compose policies

- *manual*
- *proactive*

virtualize the network, sets packet-processing constraints

- virtual topo: many-to-one, one-to-many (physical-to-virtual)

administrator role

configure CoVisor to compose policies

- *manual*
- *proactive*

virtualize the network, sets packet-processing constraints

- virtual topo: many-to-one, one-to-many (physical-to-virtual)
- packet handling: match, action

the "efficiency" challenge

to host tens of controllers

- each installs tens of thousands rules
- constantly updated rules

naive approach — prohibitively expensive

- time to recompile new policy
- time to install new rules on swithes

efficient CoVisor algorithms

incrementally composing controller policies

- priorities form a convenient algebra, obviating recompiling from scratch

devirtualization

- one(physical)-to-many(virtual)

optimizing composition

- smart data structure accelerate compilation

incremental composition

stream of member policy updates → CoVisor → updates to the composed policy

frequently and fast

update (OF rule) r = (p m; a)

- a → r.action
- m → r.priority
- p → r.match, r.mSet

policy composition revisit

$comp_+(R_1, R_2)$

- for every (r_1, r_2) in $(R_1 \times R_2)$
- generate new r if r_1.mSet intersects with r_2.mSet
  - r.match = intersection of r_1.mSet and r_2.mSet
  - r.action = union of r_1.action and r_2.action

policy composition revisit

$comp_{»}(R_1, R_2)$

- for every (r_1, r_2) in $(R_1 \times R_2)$
- generate new r if packets produced by r_1.action intersects with r_2.mSet
  - r.match = ?
  - r.action = ?

policy composition revisit

$comp_{\triangleright}(R_1, R_2)$

- stacking R1 on top of R2 with higher priority

role of priority

ideally (goal)

- single rule addition in a member policy will NOT
  - recomputing entire composed policy
  - cleaning the physical switch's flow tables

i.e., reduce update overhead

- computation
  - # of rule pairs *comp* needs to iterate
- rule update
  - # of flowmods to update a switch

strawman priority assignment

Monitoring M_R

$(1;\ srcip = 1.0.0.0/24;\ count)$
$(0;\ *;\ drop)$

Routing Q_R

$(1;\ dstip = 2.0.0.1;\ fwd(1))$
$(1;\ dstip = 2.0.0.2;\ fwd(2))$
$(0;\ *;\ drop)$

+

(**1**; **dstip=2.0.0.3**; **fwd(3)**)

Parallel composition: $comp_{+}(M_R, Q_R)$

(**7**; **srcip=1.0.0.0/24,dstip=2.0.0.1**; **fwd(1),count**)
(**6**; **srcip=1.0.0.0/24,dstip=2.0.0.2**; **fwd(2),count**)
(**5**; **srcip=1.0.0.0/24,dstip=2.0.0.3**; **fwd(3),count**)
(**4**; **srcip=1.0.0.0/24**; **count**)
(**3**; **dstip=2.0.0.1**; **fwd(1)**)
(**2**; **dstip=2.0.0.2**; **fwd(2)**)
(**1**; **dstip=2.0.0.3**; **fwd(3)**)
$(0;\ *;\ drop)$

position of the rule indicates relative priority

strawman priority assignment

Monitoring M_R
$(1;\ srcip = 1.0.0.0/24;\ count)$
$(0;\ *;\ drop)$

Routing Q_R
$(1;\ dstip = 2.0.0.1;\ fwd(1))$
$(1;\ dstip = 2.0.0.2;\ fwd(2))$
$(0;\ *;\ drop)$

\+

(**1**; **dstip=2.0.0.3**; **fwd(3)**)

Parallel composition: $comp_{+}(M_R, Q_R)$
(**7**; **srcip=1.0.0.0/24,dstip=2.0.0.1**; **fwd(1),count**)
(**6**; **srcip=1.0.0.0/24,dstip=2.0.0.2**; **fwd(2),count**)
(**5**; **srcip=1.0.0.0/24,dstip=2.0.0.3**; **fwd(3),count**)
(**4**; **srcip=1.0.0.0/24**; **count**)
(**3**; **dstip=2.0.0.1**; **fwd(1)**)
(**2**; **dstip=2.0.0.2**; **fwd(2)**)
(**1**; **dstip=2.0.0.3**; **fwd(3)**)
$(0;\ *;\ drop)$

rules in bold count toward rule update overhead

strawman priority assignment

Monitoring M_R

$(1;\ srcip = 1.0.0.0/24;\ count)$
$(0;\ *;\ drop)$

Routing Q_R

$(1;\ dstip = 2.0.0.1;\ fwd(1))$
$(1;\ dstip = 2.0.0.2;\ fwd(2))$
$(0;\ *;\ drop)$

\+

(**1**; **dstip=2.0.0.3**; **fwd(3)**)

Parallel composition: $comp_{+}(M_R, Q_R)$

(**7**; **srcip=1.0.0.0/24,dstip=2.0.0.1**; **fwd(1),count**)
(**6**; **srcip=1.0.0.0/24,dstip=2.0.0.2**; **fwd(2),count**)
(**5**; **srcip=1.0.0.0/24,dstip=2.0.0.3**; **fwd(3),count**)
(**4**; **srcip=1.0.0.0/24**; **count**)
(**3**; **dstip=2.0.0.1**; **fwd(1)**)
(**2**; **dstip=2.0.0.2**; **fwd(2)**)
(**1**; **dstip=2.0.0.3**; **fwd(3)**)
$(0;\ *;\ drop)$

rules in bold count toward rule update overhead

smartly set priority

- to make updates incremental

incremental update and priority algebra

r is computed from r_1 and r_2

- r.priority ⟵ r_1.priority, r_2.priority
- incremental update without modifying existing priorities

incremental update and priority algebra

r is computed from r_1 and r_2

- $r.priority \longleftarrow r_1.priority, r_2.priority$

$comp_+$

- $r.priority = r_1.priority + r_2.priority$

$comp_{<<}$

- $r.priority = r_1.priority \times MAX_2 + r_2.priority$

incremental update and priority algebra

r is computed ($comp_{\rhd}$) from R_1 and R_2

- r.priority = r.priority + MAX_2 if r in R1
- r.priority = r.priority if r in R2

algebra properties

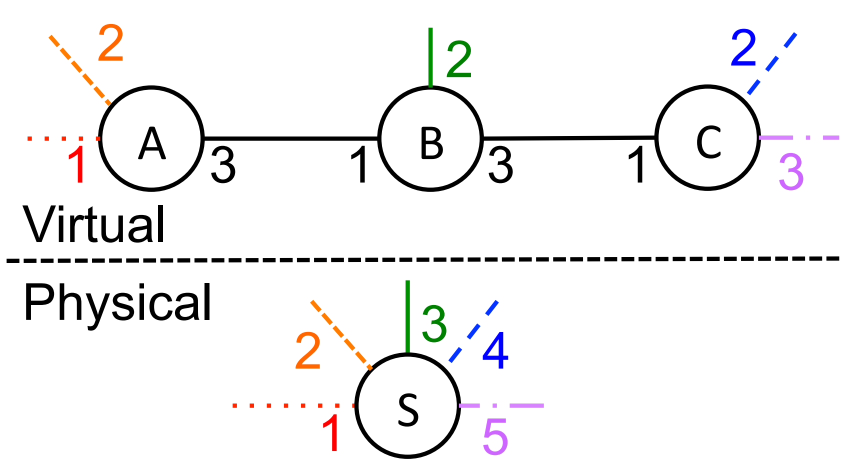
identify and prove properties

- the assignment schema ensures newly generated priority
  - leaves existing priority unchanged
  - together, the new and existing priorities are compliant with the straw man scheme

devirtualization

topology transformation for one-to-many

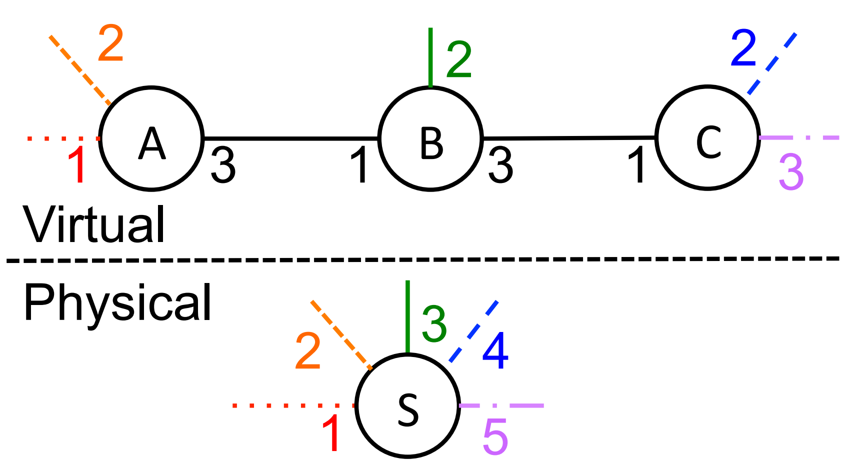
- generate symbolic path (from the virtual ingress to egress)
- on each virtual path, sequentially compose virtual policies to into a single (physical) rule

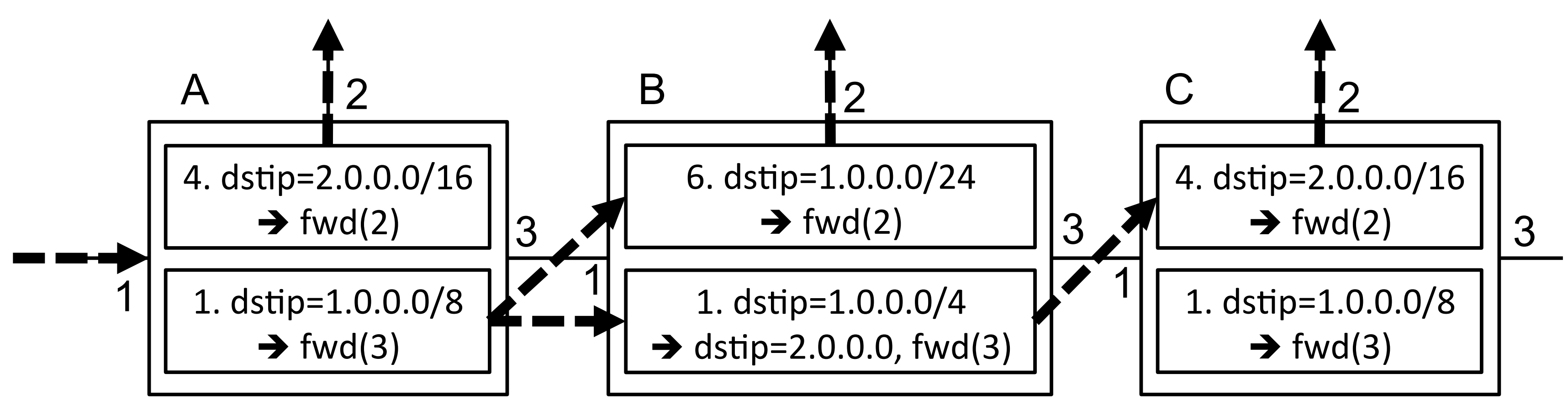


devirtualization

on the virtual topology, find symbolic paths

- inject wildcard packet * at ingress
- at each hop
  - evaluate the virtual policy, resulting in new packets
- until all packets reach egress

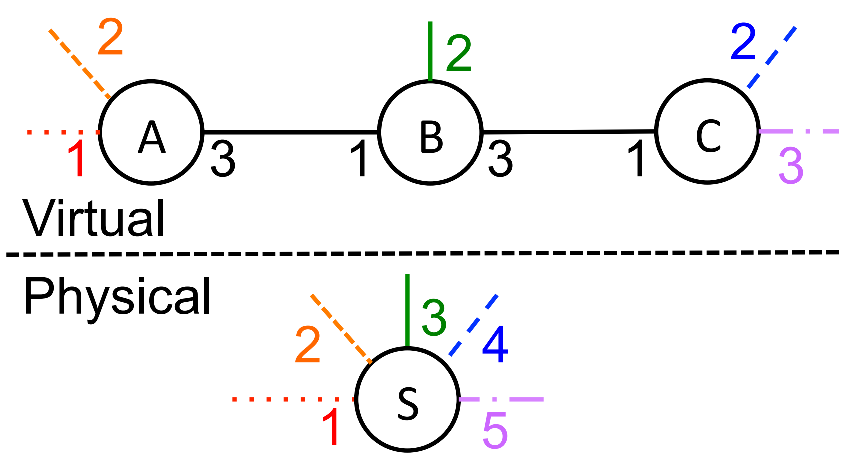


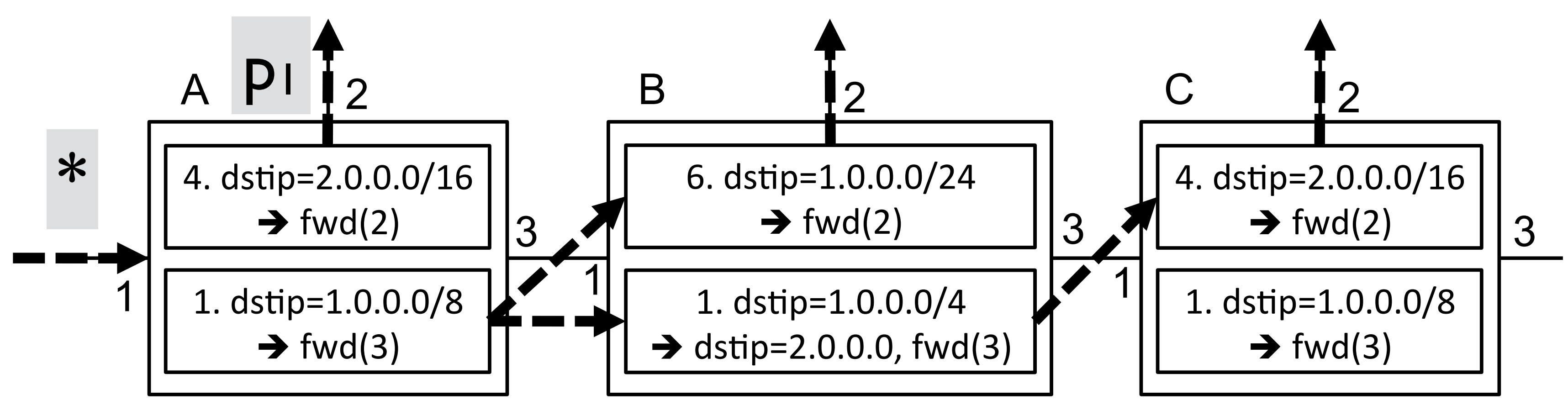


devirtualization

on the virtual topology, find symbolic paths

* → A(R_1) → p_1



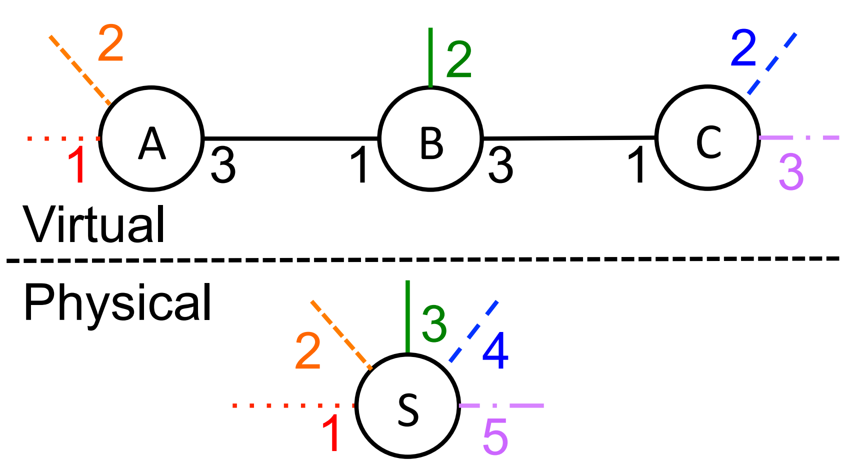


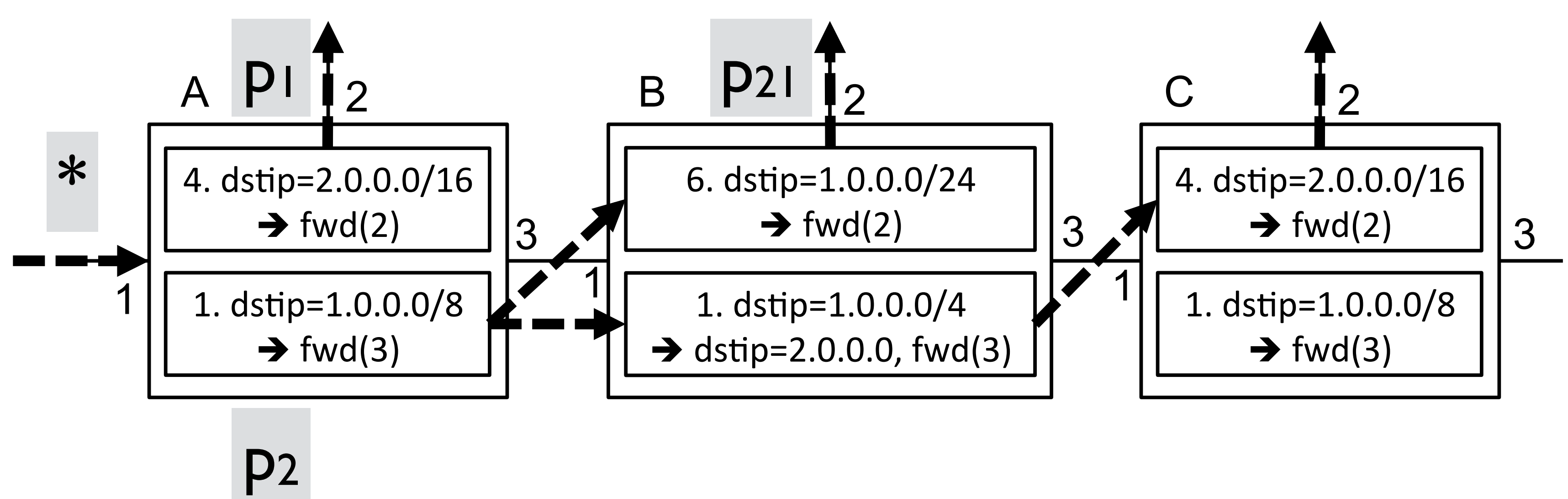
devirtualization

on the virtual topology, find symbolic paths

$* \longrightarrow A(R_1) \longrightarrow p_1$

$* \longrightarrow A(R_2) \longrightarrow p_2 \longrightarrow B(R_1) \longrightarrow p_{21}$





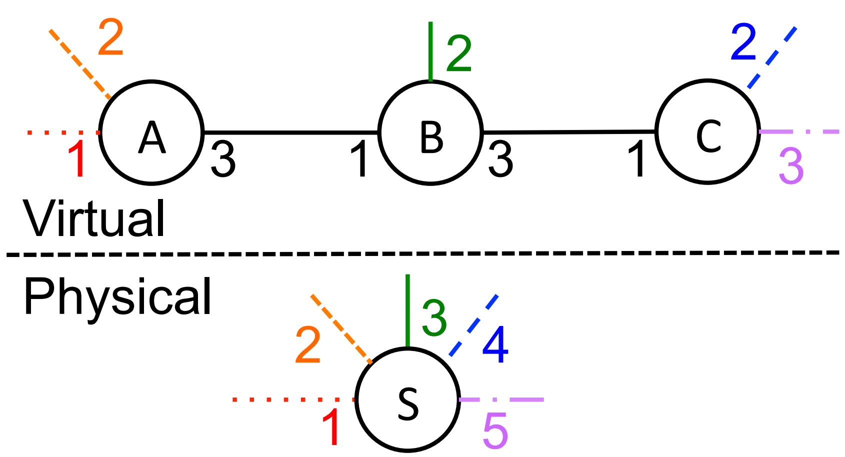
devirtualization

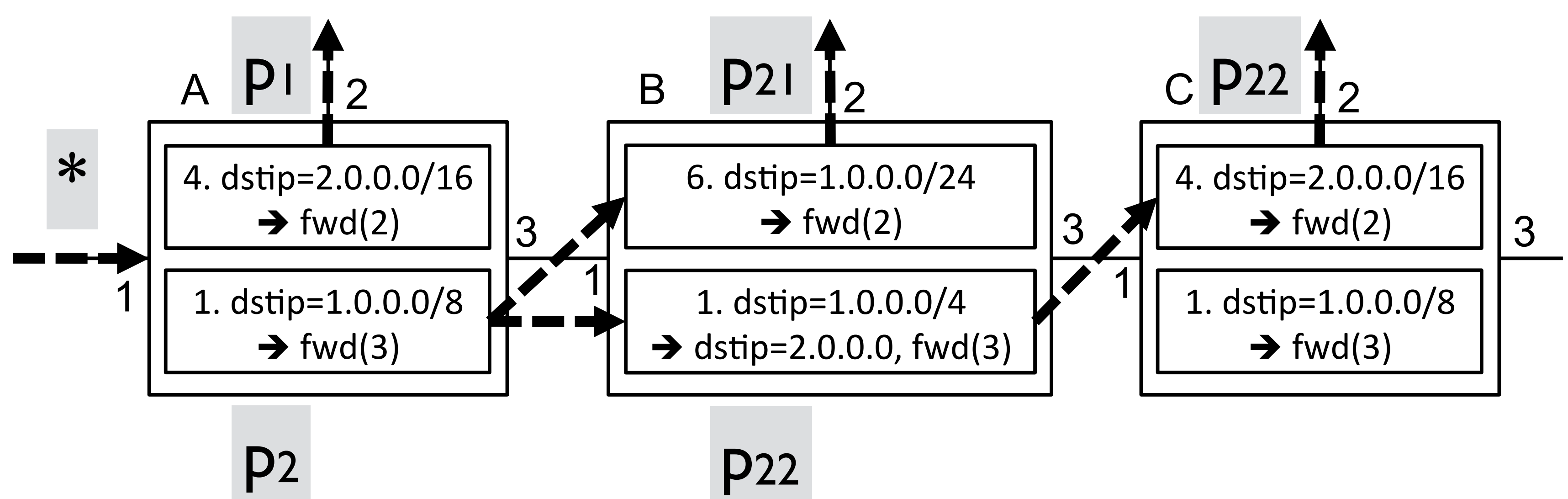
on the virtual topology, find symbolic paths

* → A(R_1) → p_1

* → A(R_2) → p_2 → B(R_1) → p_{21}

* → A(R_2) → p_2 → B(R_2) → p_{22} → C(R_1) → p_{22}





devirtualization

sequentially compose policies on each path

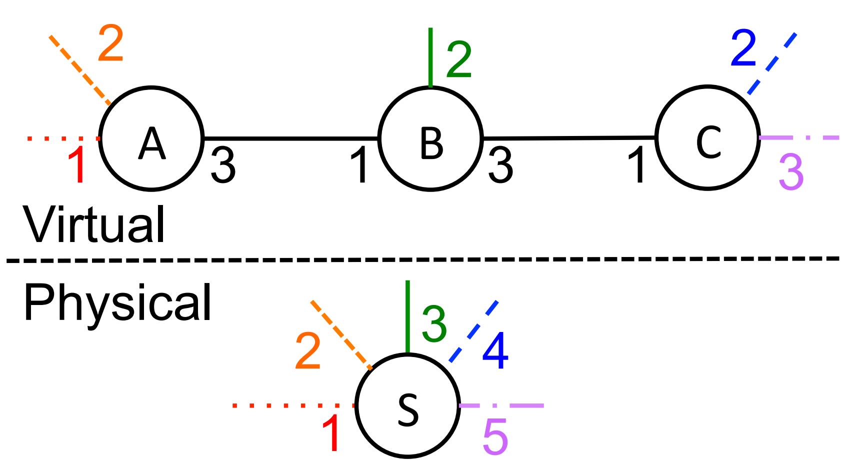
priority

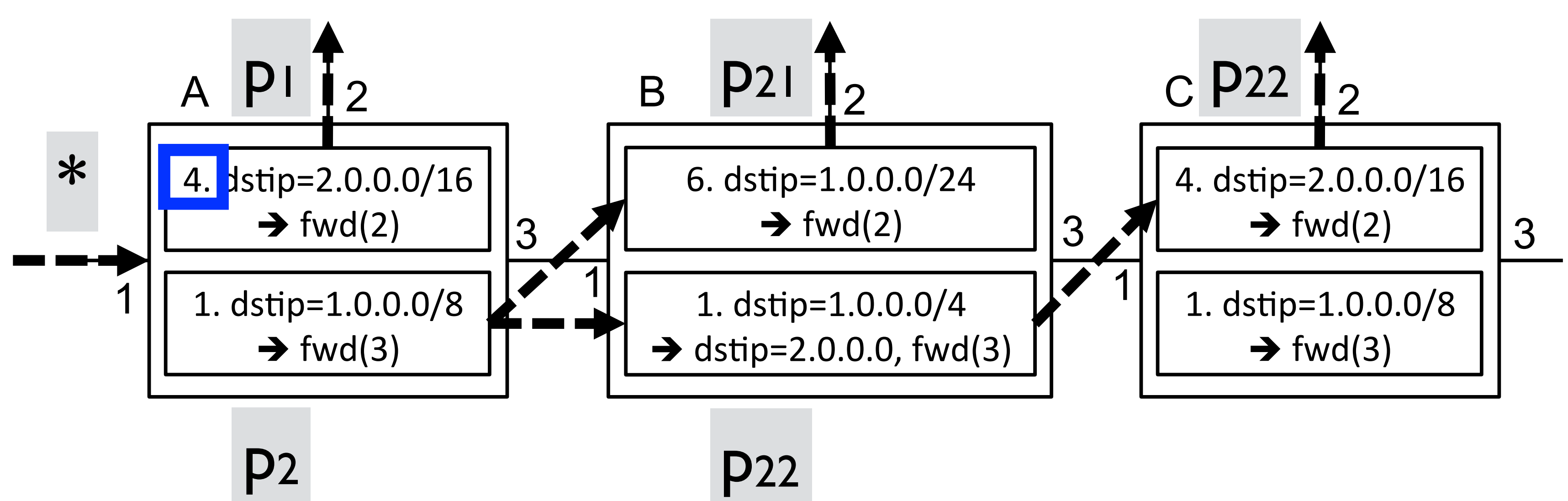
4

$* \rightarrow A(R_1) \rightarrow p_1$

$* \rightarrow A(R_2) \rightarrow p_2 \rightarrow B(R_1) \rightarrow p_{21}$

$* \rightarrow A(R_2) \rightarrow p_2 \rightarrow B(R_2) \rightarrow p_{22} \rightarrow C(R_1) \rightarrow p_{22}$

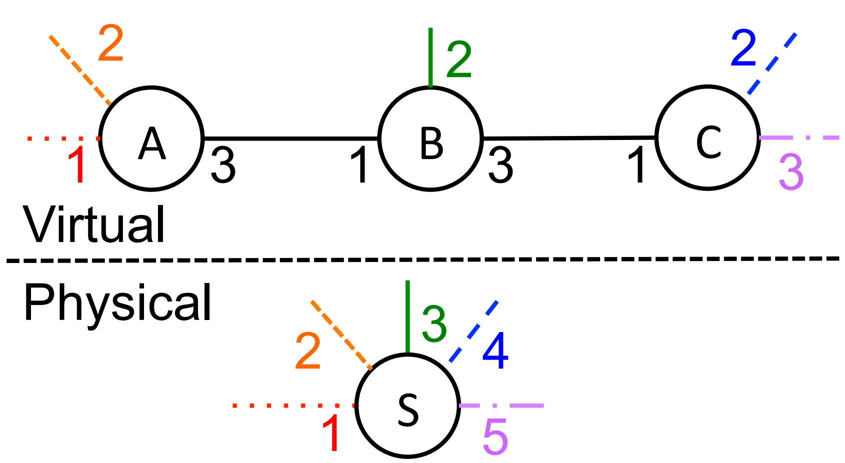


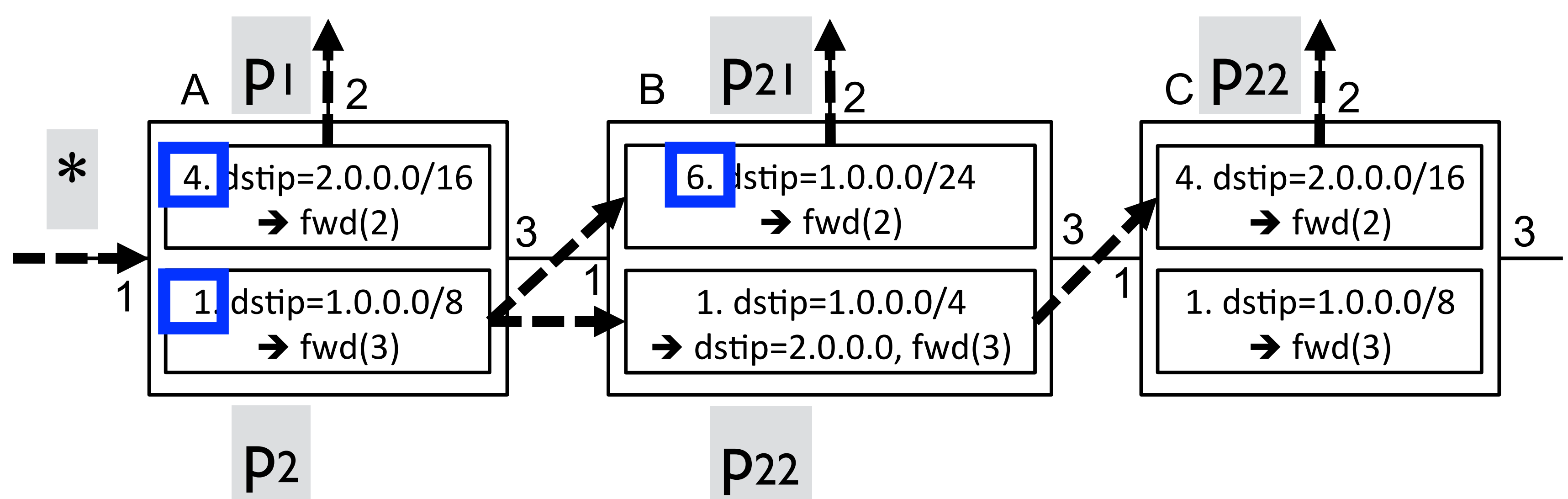


devirtualization

sequentially compose policies on each path

priority	
4	$* \rightarrow A(R_1) \rightarrow p_1$
$1 \circ 6$ (=14)	$* \rightarrow A(R_2) \rightarrow p_2 \rightarrow B(R_1) \rightarrow p_{21}$
	$* \rightarrow A(R_2) \rightarrow p_2 \rightarrow B(R_2) \rightarrow p_{22} \rightarrow C(R_1) \rightarrow p_{22}$

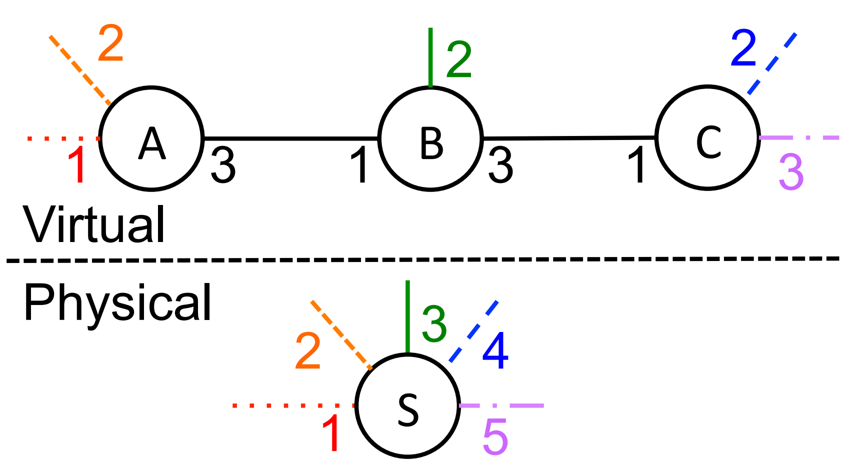


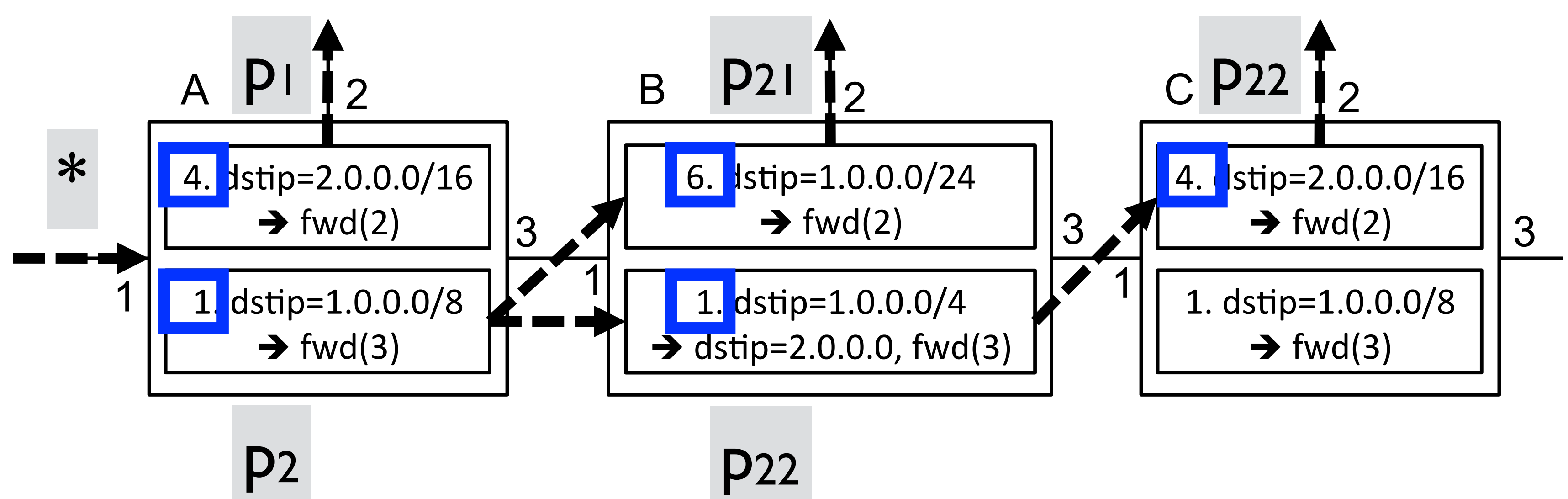


devirtualization

sequentially compose policies on each path

priority		
4		* → $A(R_1)$ → p_1
$1 \circ 6$	(=14)	* → $A(R_2)$ → p_2 → $B(R_1)$ → p_{21}
$1 \circ 1 \circ 4$	(=76)	* → $A(R_2)$ → p_2 → $B(R_2)$ → p_{22} → $C(R_1)$ → p_{22}



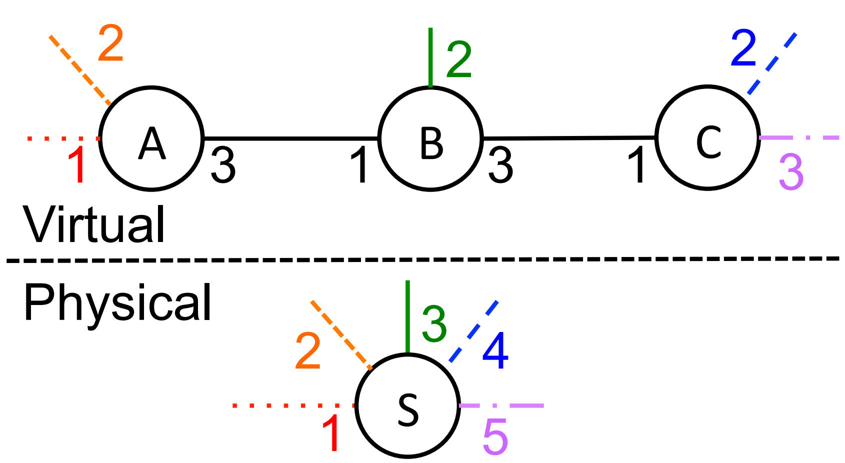


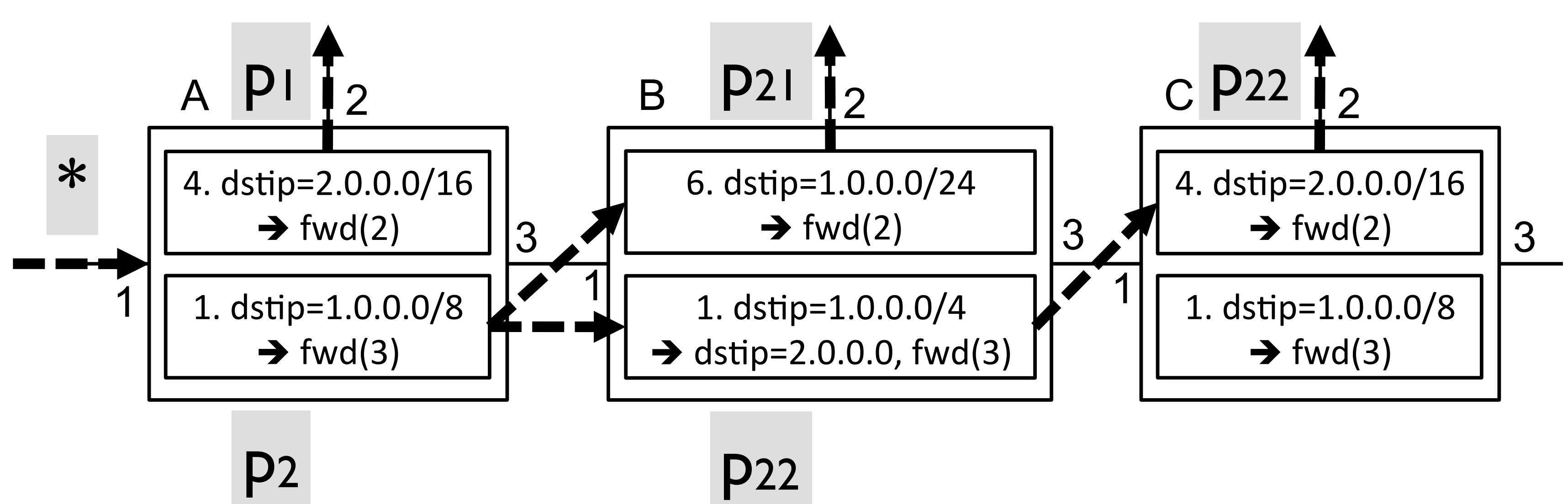
devirtualization

sequentially compose policies on each path

priority

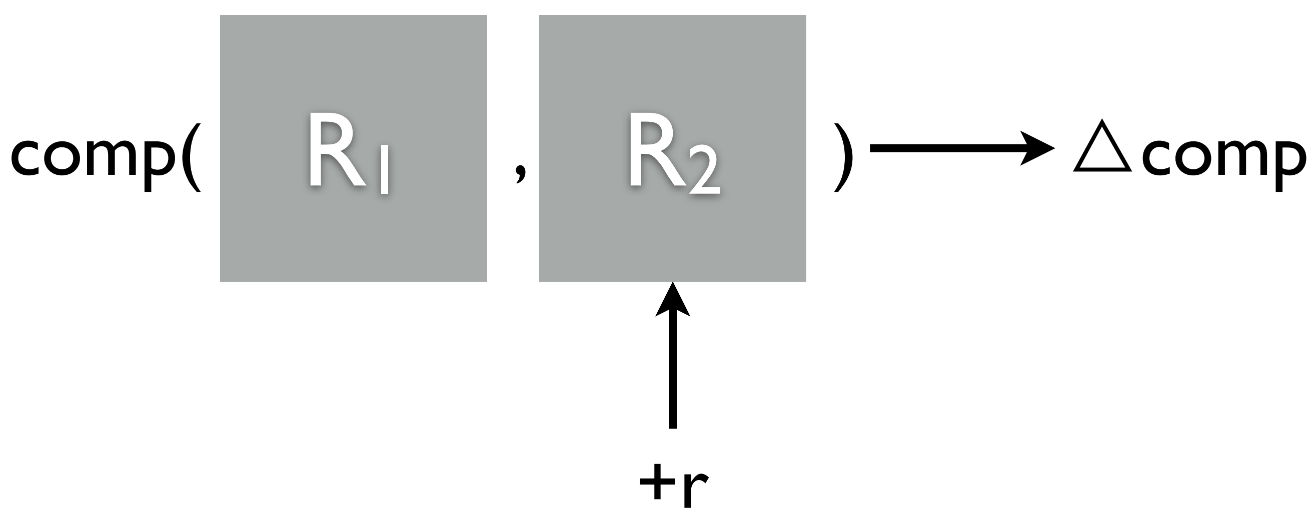
4 ○ 0 ○ 0	(=256)	* → $A(R_1)$ → p_1
1 ○ 6 ○ 0	(=112)	* → $A(R_2)$ → p_2 → $B(R_1)$ → p_{21}
1 ○ 1 ○ 4	(=76)	* → $A(R_2)$ → p_2 → $B(R_2)$ → p_{22} → $C(R_1)$ → p_{22}





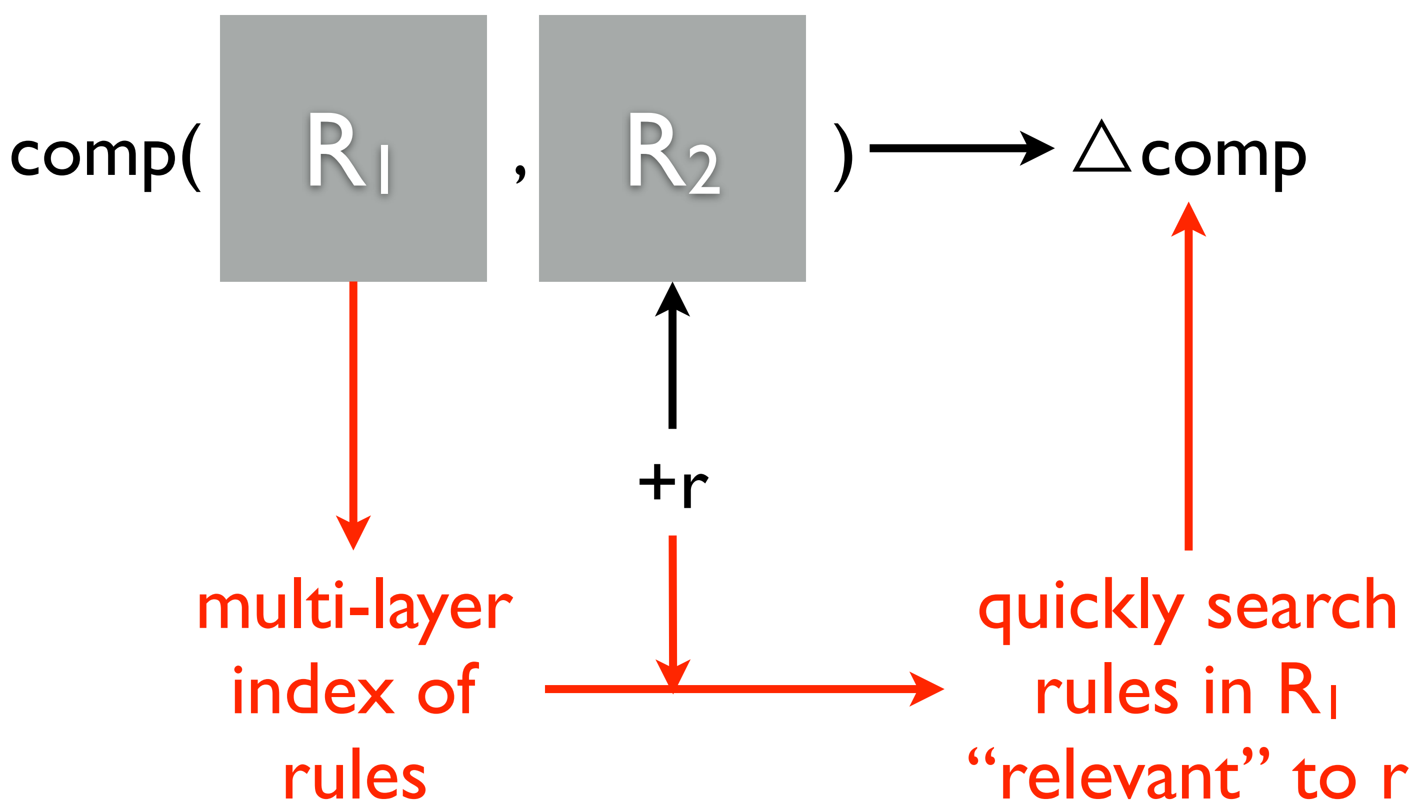
optimization: indexing rules

accelerate complication with smart data structure



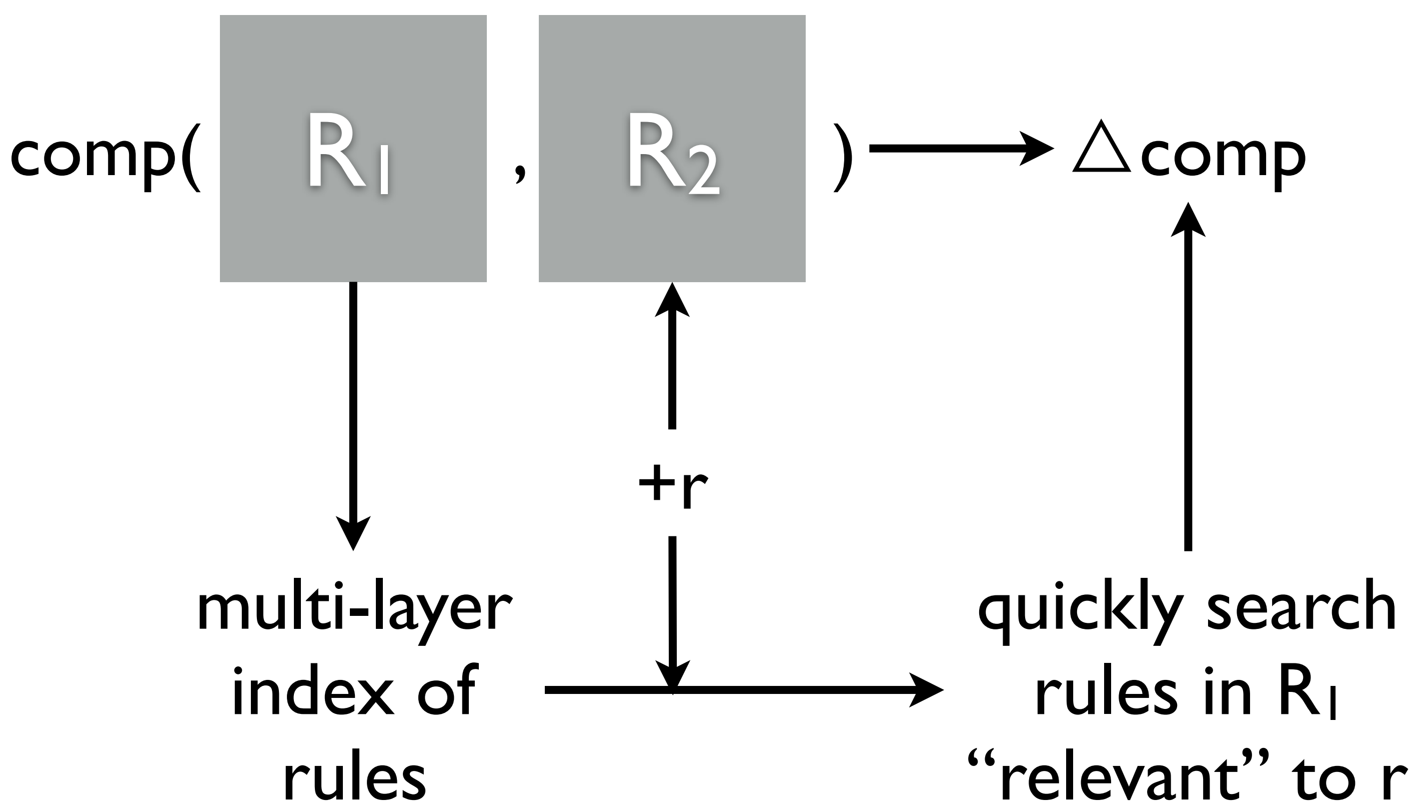
optimization: indexing rules

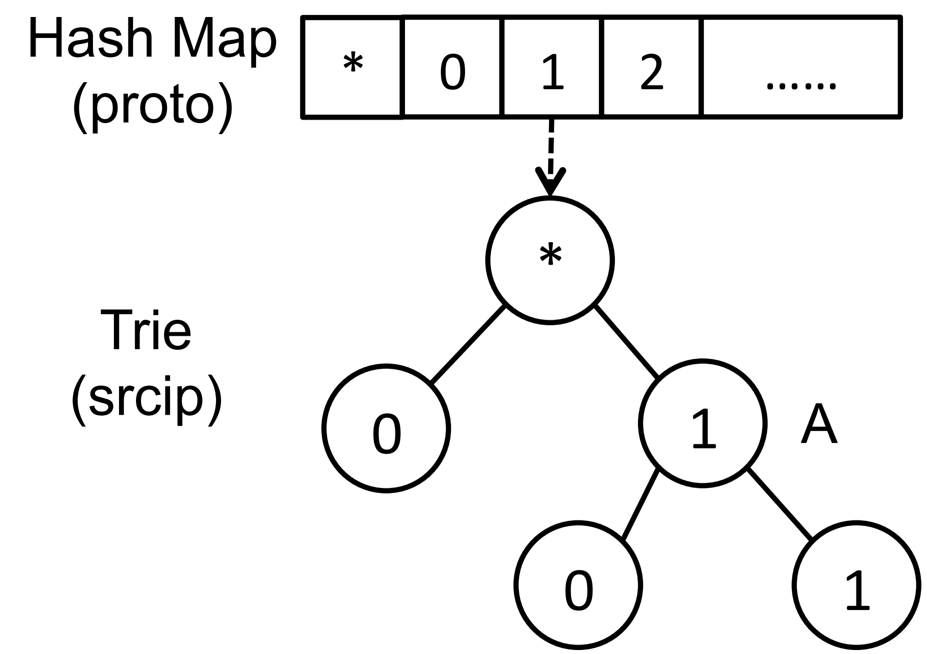
accelerate complication with smart data structure



optimization: indexing rules

accelerate complication with smart data structure



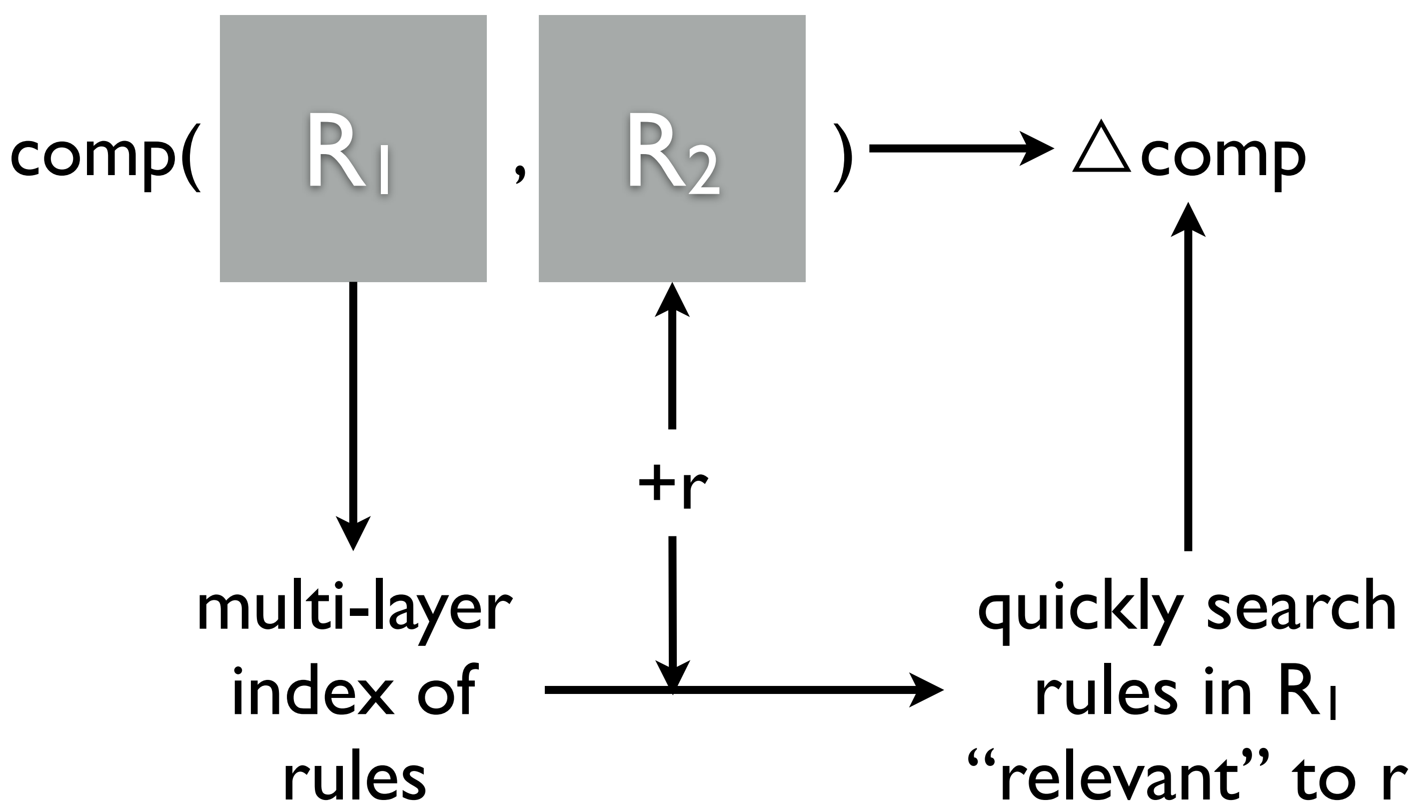


example multi-layer index

- hash table for exact match
- trie for prefix match
- list for arbitrary wildcard-match

optimization: indexing rules

reduce index size by policy correlation



only index the "correlated" info

R_1.index = R_2.index
= R_1.fields ∩ R_2.fields