ΕΛΛΗΝΙΚΗ ΔΗΜΟΚΡΑΤΙΑ
ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΡΗΤΗΣ

Δίκτυα Καθοριζόμενα από Λογισμικό

Ενότητα 2.2: Modular Network Programming with Pyretic

Ξενοφώντας Δημητρόπουλος
Τμήμα Επιστήμης Υπολογιστών

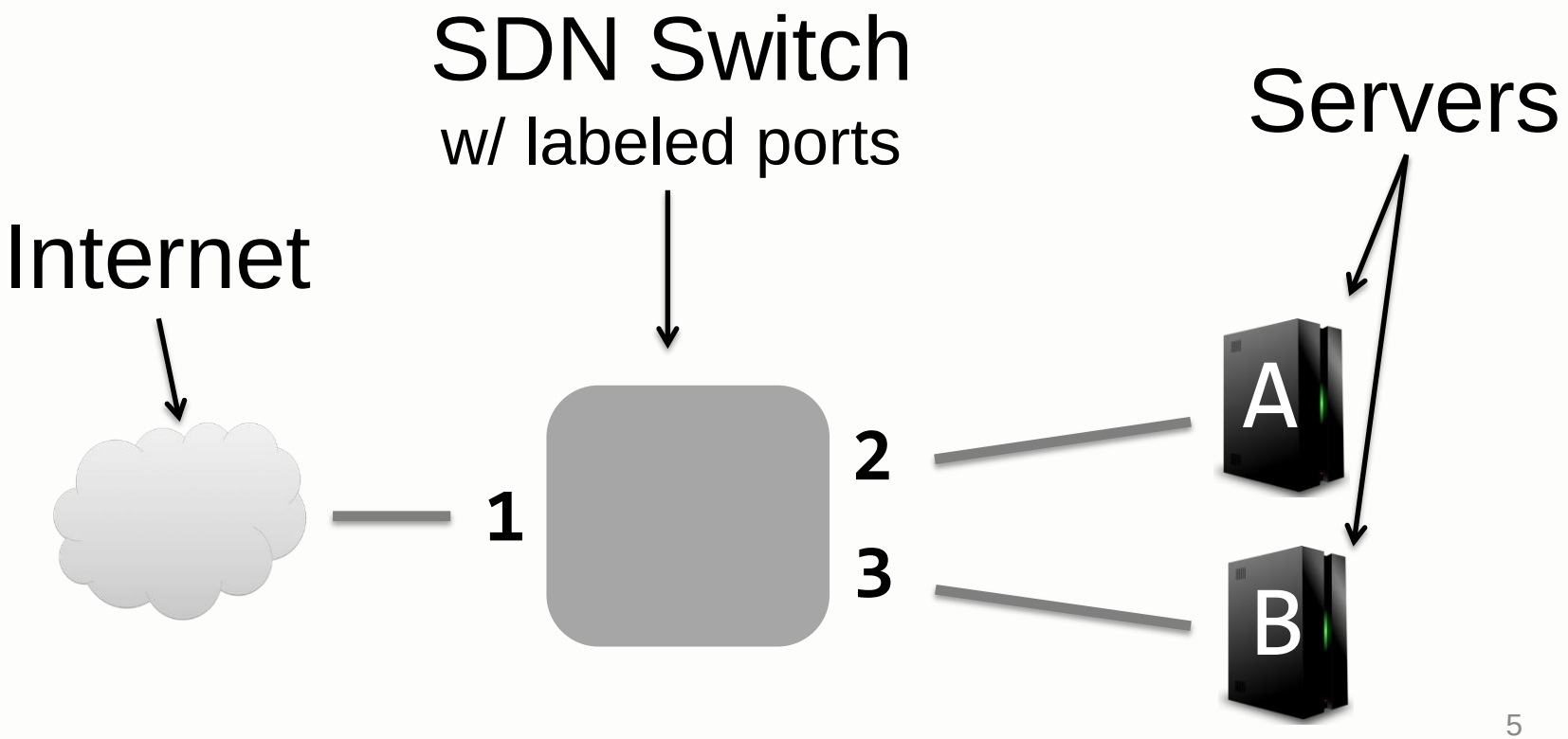
HY436: Modular Network Programming with Pyretic

Xenofontas Dimitropoulos

27/10/2014

Credits: Slides modified from Joshua Reich's (Princeton) NSDI 13 talk on "Composing Software Defined Networks"

Running Example Network



Programming in OpenFlow

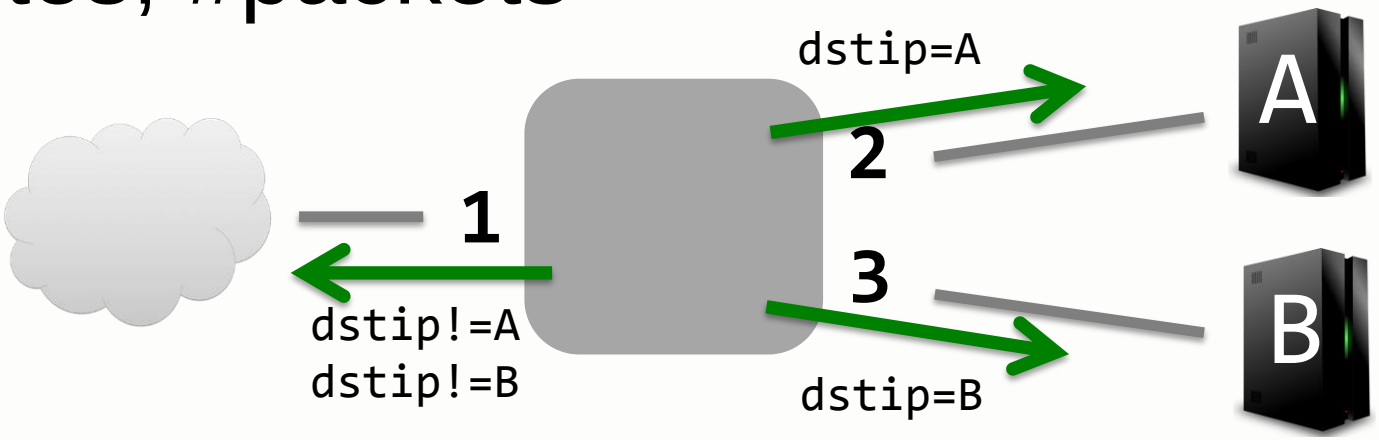
OpenFlow Priority Program

```
2:match(dstip=A)[fwd(2)]
1:match(*       )[fwd(1)]
2:match(dstip=B)[fwd(3)]
```

Pattern Action

Counters for each rule
- #bytes, #packets

Route: IP/fwd



One API, Many Uses

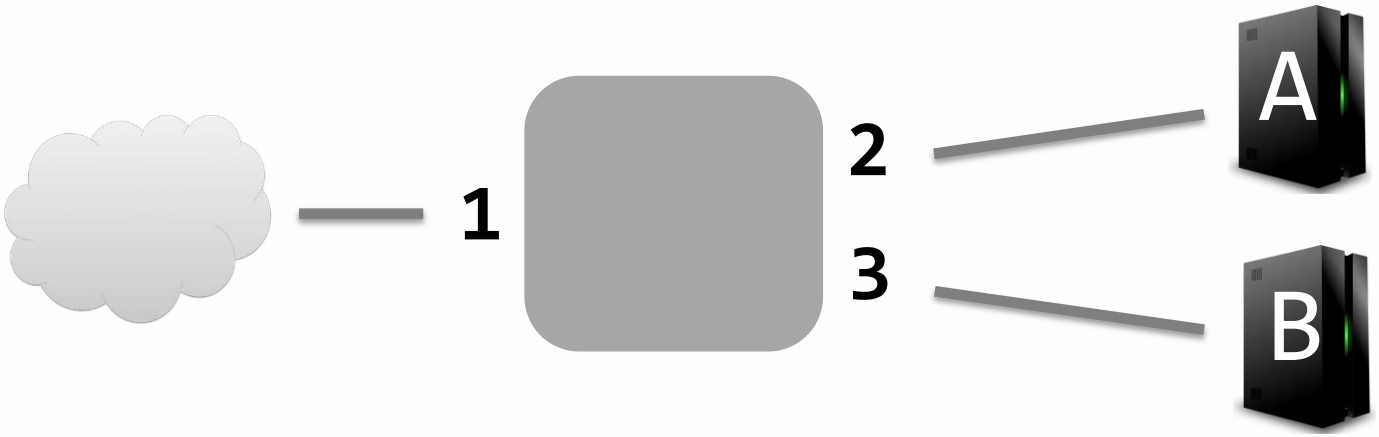
Priority Ordered

```
match(dstmac=A)[fwd(2)]
match(dstmac=B)[fwd(3)]
match(*       )[fwd(1)]
```

Pattern

Action

***Switch*: MAC/fwd**



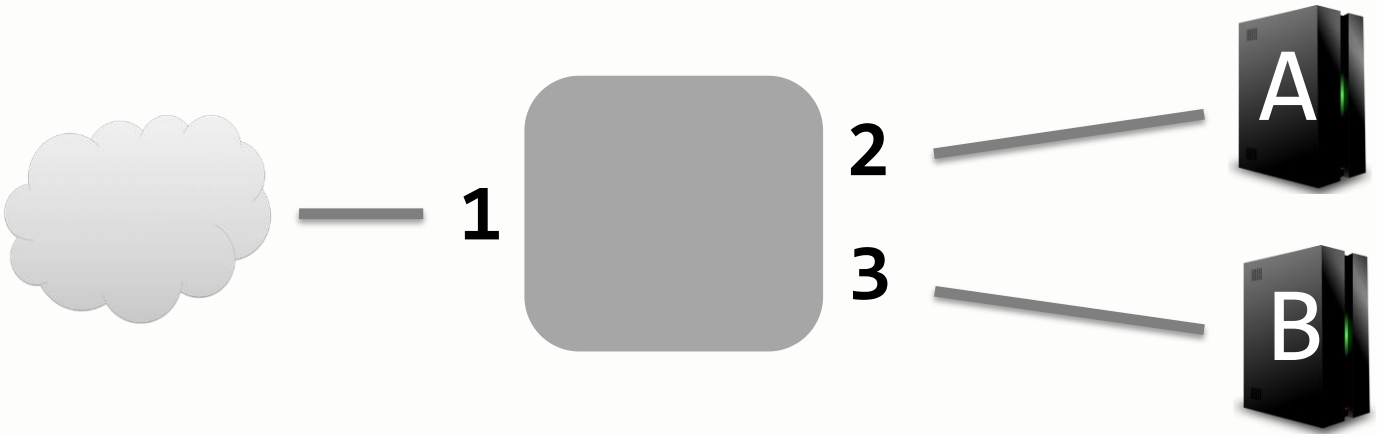
One API, Many Uses

```
match(srcip=0*,dstip=P)[mod(dstip=A)]
match(srcip=1*,dstip=P)[mod(dstip=B)]
```

Pattern

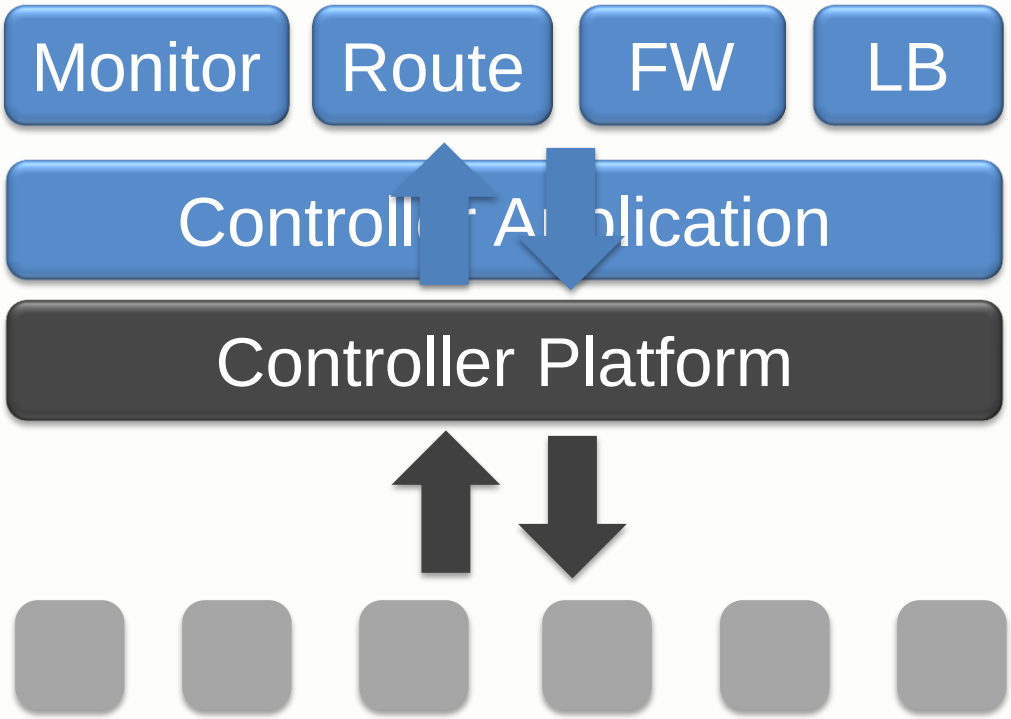
Action

***Load Balancer*: IP/mod**



But Only Half of the Story

Modular & Intuitive



Programmer Interface

?

Hardware Interface

OpenFlow (assembly)

Hardware Flow-Table Rules (Machine Language)

OpenFlow Isn't Modular

Balance then **Route**

```
match(srcip=0*,dstip=P)[mod(dstip=A)]
match(srcip=1*,dstip=P)[mod(dstip=B)]
```

```
match(dstip=A)[fwd(2)]
match(dstip=B)[fwd(3)]
match(*      )[fwd(1)]
```

Combined Rules?
(only one match)

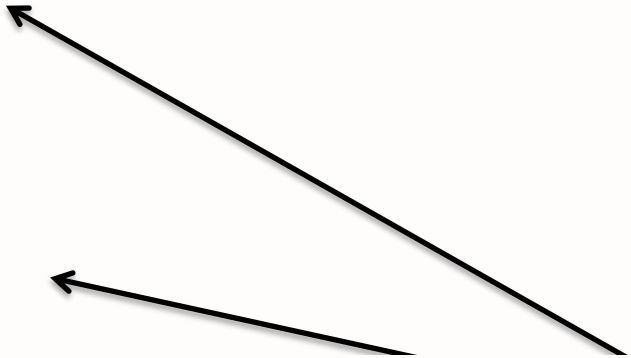
```
match(srcip=0*,dstip=A)[fwd(2)]
match(         dstip=B)[fwd(3)]
match(*                )[fwd(1)]
match(srcip=1*,dstip=A)[fwd(2)]
match(         dstip=B)[fwd(3)]
match(*                )[fwd(1)]
match(srcip=0*,dstip=P)[mod(dstip=A)]
match(srcip=1*,dstip=P)[mod(dstip=B)]
```

Balances w/o Forwards!

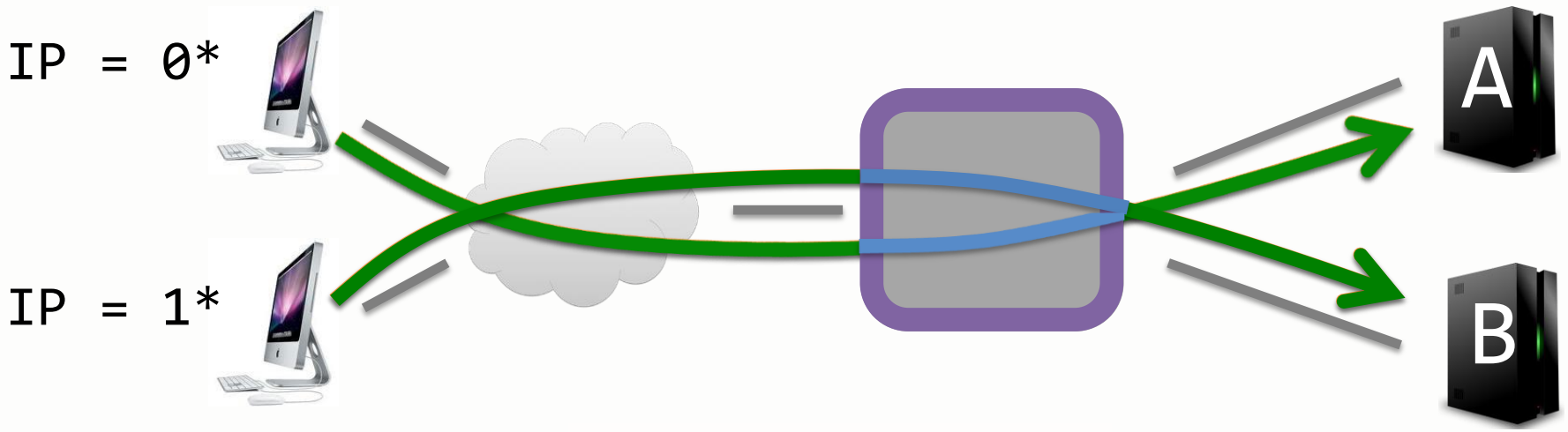
Balancing!

Pyretic (Contributions)

Abstracts	Providing	Supporting



Compositional Operators:
A Monitoring Load Balancer



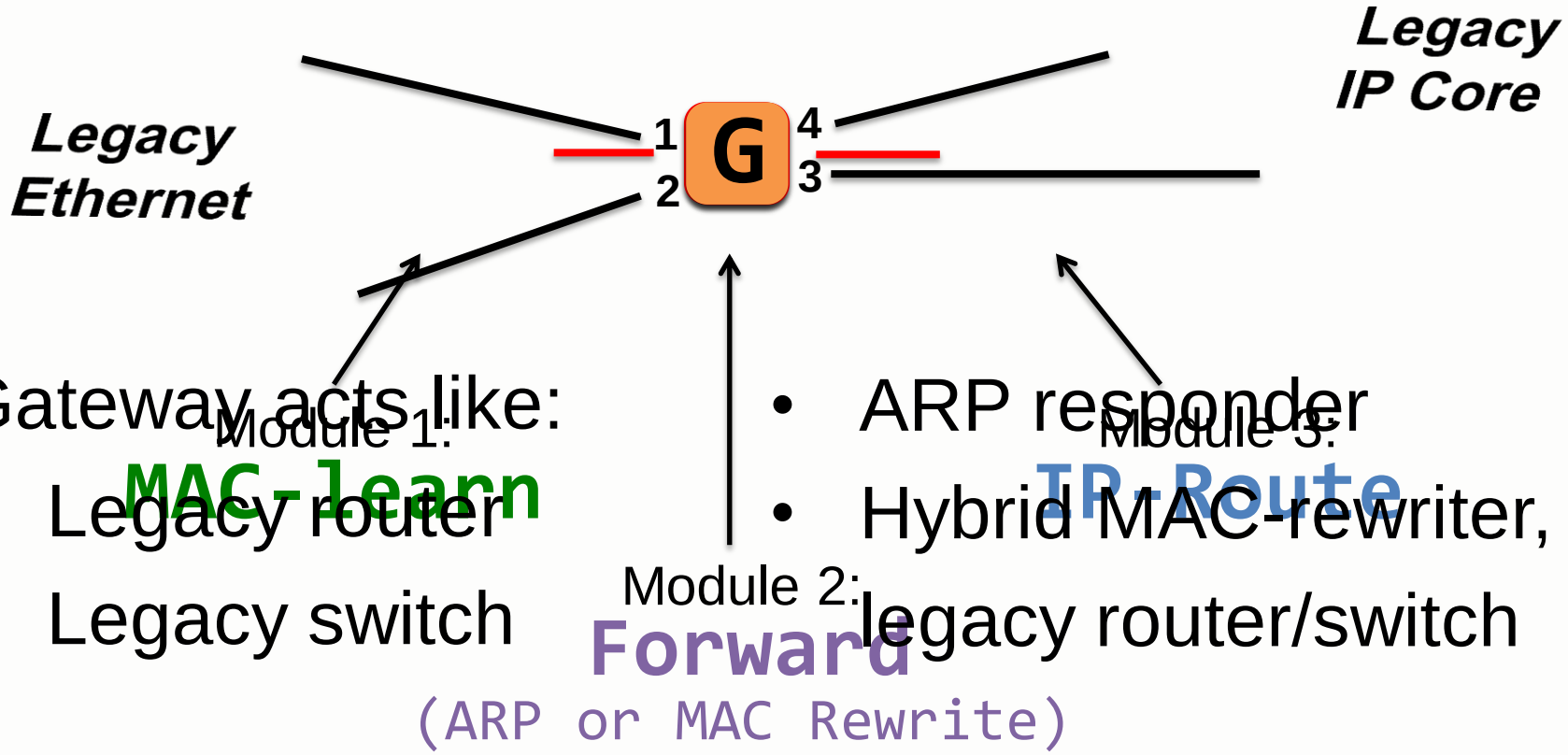
Module 1:

Balance `P` **then** `A, B`

Rewrite `dstip` `P` to
`A`, if `srcip=0*`
`B`, if `srcip=1*`

- Traffic to `P` re-addressed and fowarded to either `A` or `B`, based on source
- Counted if from source `X`

Topology Abstraction:
A Legacy Gateway Replacement



Gateway acts like:

- Legacy router
- Legacy switch
- ARP responder
- Hybrid MAC-rewriter, legacy router/switch

Pyretic's Design

- Monitoring Load Balancer
  - Encode policies as functions
  - Compositional operators
  - Queries as forwarding policies
- MAC-Learning Module
  - Dynamic Policies
- "One Big Switch" Topology Abstraction
  - Extensible packet model

`Pyretic` Drop Policy

Goal: Drop packets (i.e., `OpenFlow` drop)

Write: `drop`

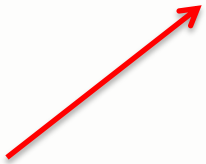
Means: `eval(drop,p) = {}`

`Pyretic` Forward Policy

Goal: Forward packets out port `a`

Write: `fwd(a)`

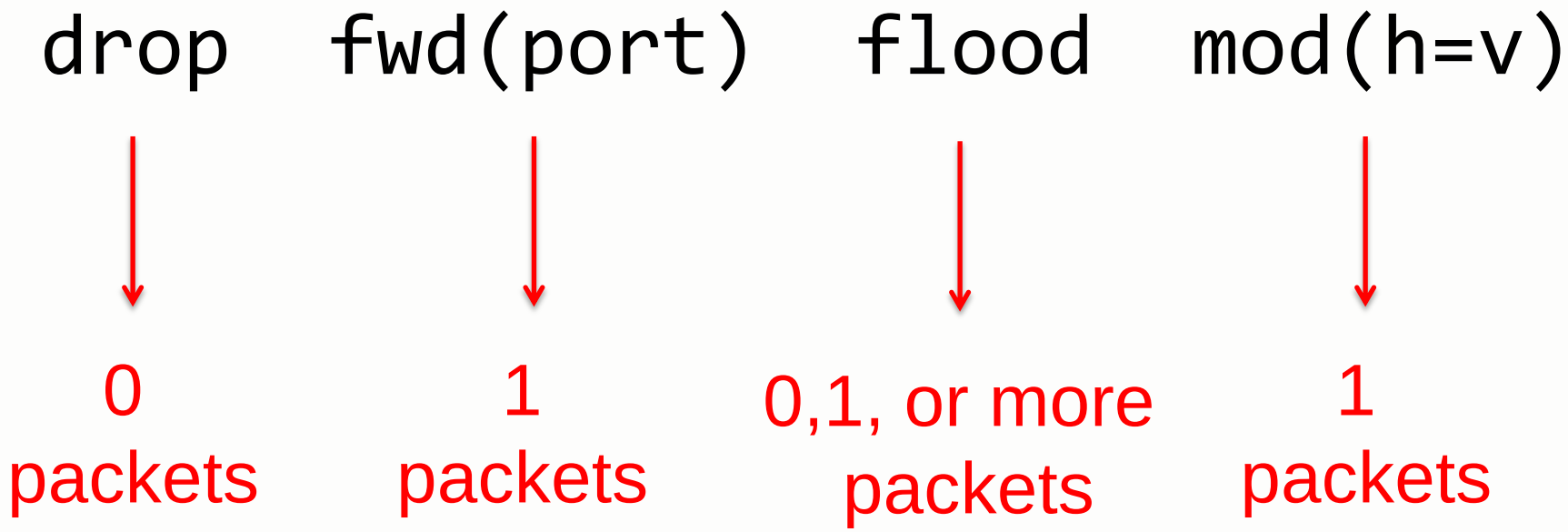
Means: `eval(fwd(a),p) = {p[outport:=a]}`



located packet w/ fields for

- `switch`
- `inport`
- `outport`

One Pyretic Policy For Each OpenFlow Action



Pyretic Policy

A function mapping a located packet to a set of located packets

```
eval(policy,packet) = {packet}
```

Puts focus on ***meaning*** instead of ***mechanics***

Enabling Compositional Operators

Parallel '+': *Do both C1 and C2 simultaneously*

```
eval(C1 + C2,p) =  eval(C1,p) U eval(C2,p)
```

Sequential '>>': *First do C1 and then do C2*

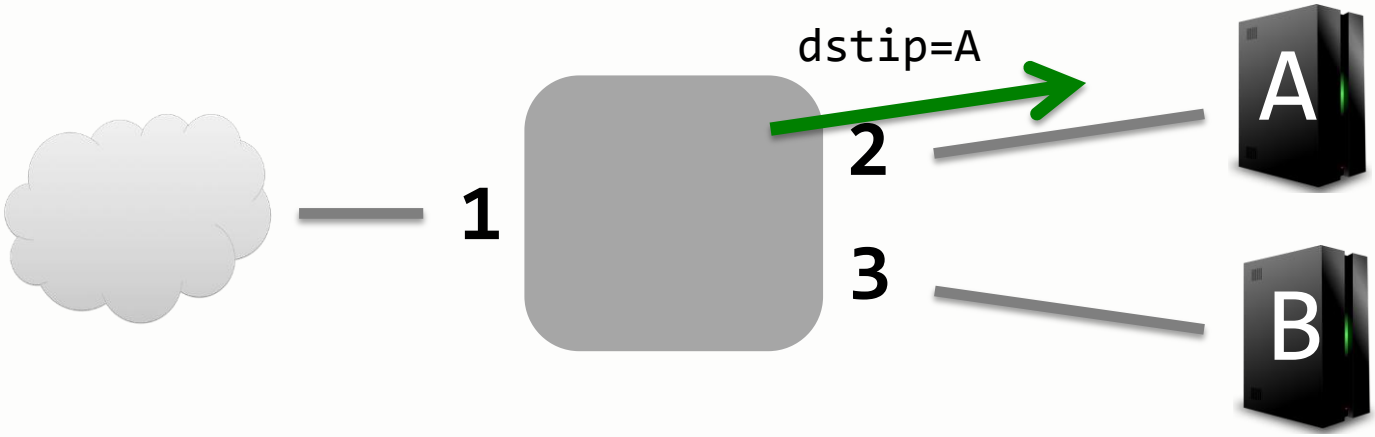
```
eval(C1 >> C2,p) =  U {eval(C2,p')
                       for p' in eval(C1,p)}
```

Sequential Composition Example

- Code to forward to A ?

```
match(dstip=A) >> fwd(2)
```

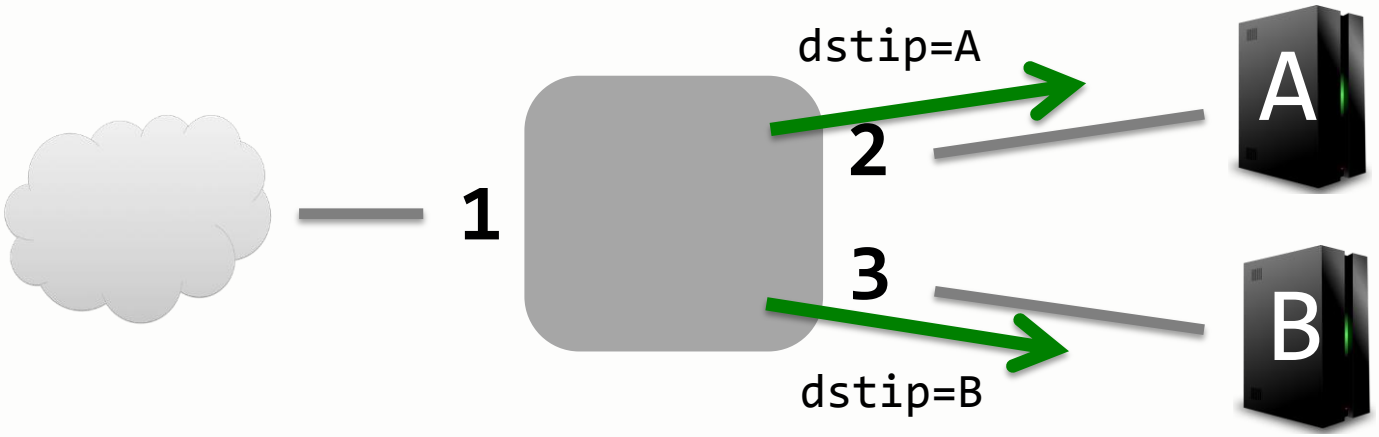
- **The old syntax has been depricated:**
  `match(dstip=A)[fwd(2)]`



Parallel Composition Example

- Code to forward to A or B?

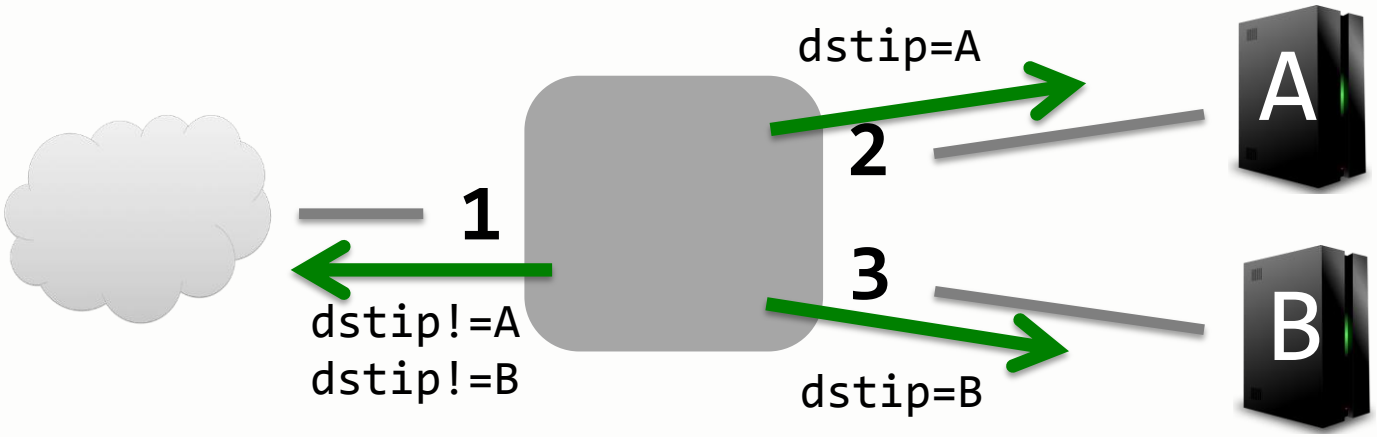
```
(match(dstip=A) >> fwd(2)) +
(match(dstip=B) >> fwd(3))
```



Pyretic program for routing?

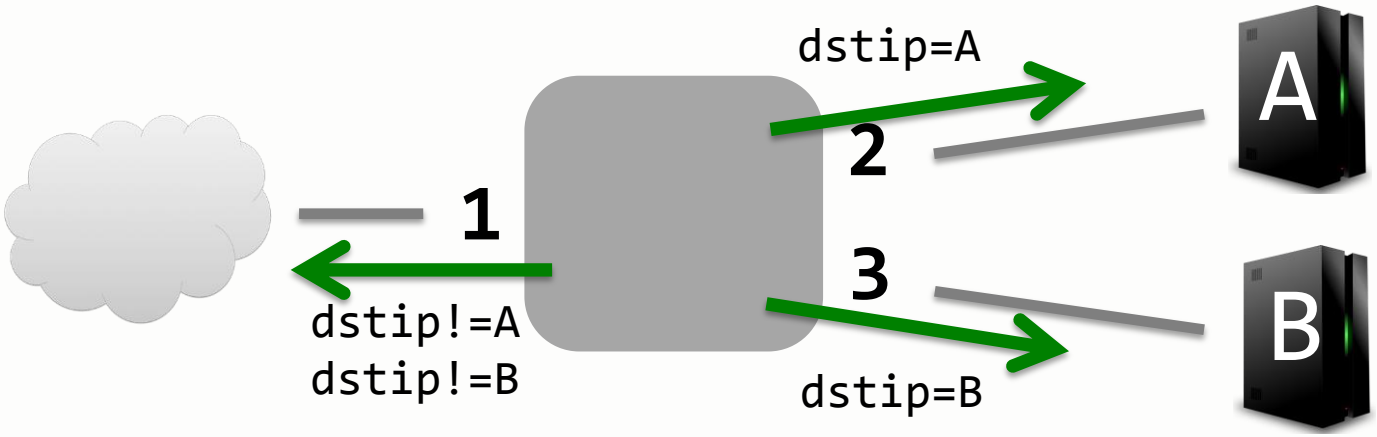
No priorities needed!

```
( match(dstip=A) >> fwd(2) ) +
( match(dstip=B) >> fwd(3) ) +
~(match(dstip=A) | match(dstip=B)) >> fwd(1)
```



if_ abbreviation

```
if_( match(dstip=A), fwd(2),
      if_( match(dstip=B), fwd(3) , fwd(1) )
   )
```



Examples

- `flood`

  ➔Broadcasts every single packet on any inport of any switch

- `match(switch=s1,inport=2,srcip='1.2.3.4') >> flood`

  ➔Only broadcasts packets on switch s1 input 2 from srcip 1.2.3.4

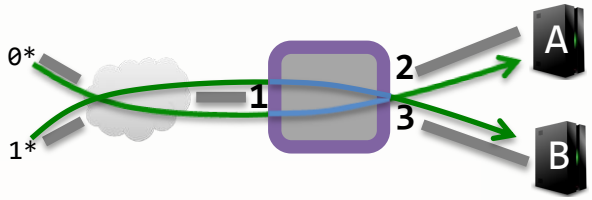
Querying as Forwarding

`packets(limit,[h])` `count_packets(every,[h])`

Abstract location corresponding to a data-structure that store packet-data and callback processing routines

```
b = count_packets(every=1)
b.register_callback(print)
match(srcip=X) >> fwd(b)
```

Monitoring Load Balancer




```
                                    eval(id,p) = {p}
route =
  ( match(dstip=A) >> fwd(2) ) +
  ( match(dstip=B) >> fwd(3) ) +
  ~(match(dstip=A) | match(dstip=B)) >> fwd(1)
```

```
b = count_packets(every=1)
b.register_callback(print)
monitor = match(srcip=X) >> fwd(b)
```

Compared to

```
install_flowmod(5,srcip=X &  dstip=P,[mod(dstip=A), fwd(2)])
install_flowmod(4,srcip=0* & dstip=P,[mod(dstip=A), fwd(2)])
install_flowmod(4,srcip=1* & dstip=P,[mod(dstip=B), fwd(3)])
install_flowmod(4,srcip=X & dstip=A ,[              fwd(2)])
install_flowmod(4,srcip=X &  dstip=B,[              fwd(3)])
install_flowmod(3,                dstip=A,[         fwd(2)])
install_flowmod(3,                dstip=B,[         fwd(3)])
install_flowmod(2,srcip=X                 ,[        fwd(1)])
install_flowmod(1,*                       ,[        fwd(3)])
```

Summary: Pyretic Policy Syntax

Check also Assignment for syntax changes

8 Actions

```
A ::= drop | fwd(port) | flood | mod(h=v) |
      id | push(h=v) | pop(h) | move(h1=h2)
```

6 Predicates

```
P ::= all_packets | no_packets | match(h=v) |
    | P & P | (P|P) | ~P
```

2 Query Buckets

```
B ::= packets(limit,[h])|count_packets(every,[h])
```

5 Policies

```
C ::= A | fwd(B) | P[C] | (C|C) | C >> C
```

New Actions Explained

- `passthrough`
  identity function (changed to `id`)
- `push(h=v)`
  pushes value v to field h
- `pop(h)`
  pops value out of field h
- `move(h1=h2)`
  pops top value from h2 and pushes it to h1

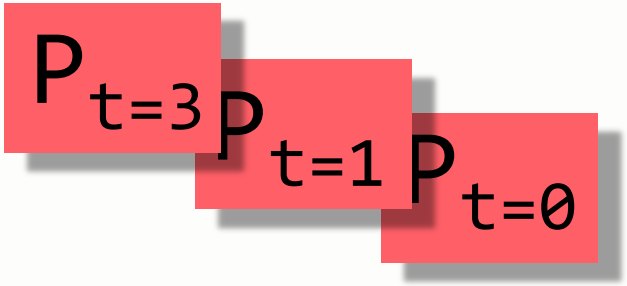
Pyretic's Design

- Monitoring Load Balancer
  - Encode Policies as Functions
  - Compositional Operators
  - Queries as Forwarding Policies
- MAC-Learning Module
  - Dynamic Policies
- "One Big Switch" Topology Abstraction
  - Extensible packet model

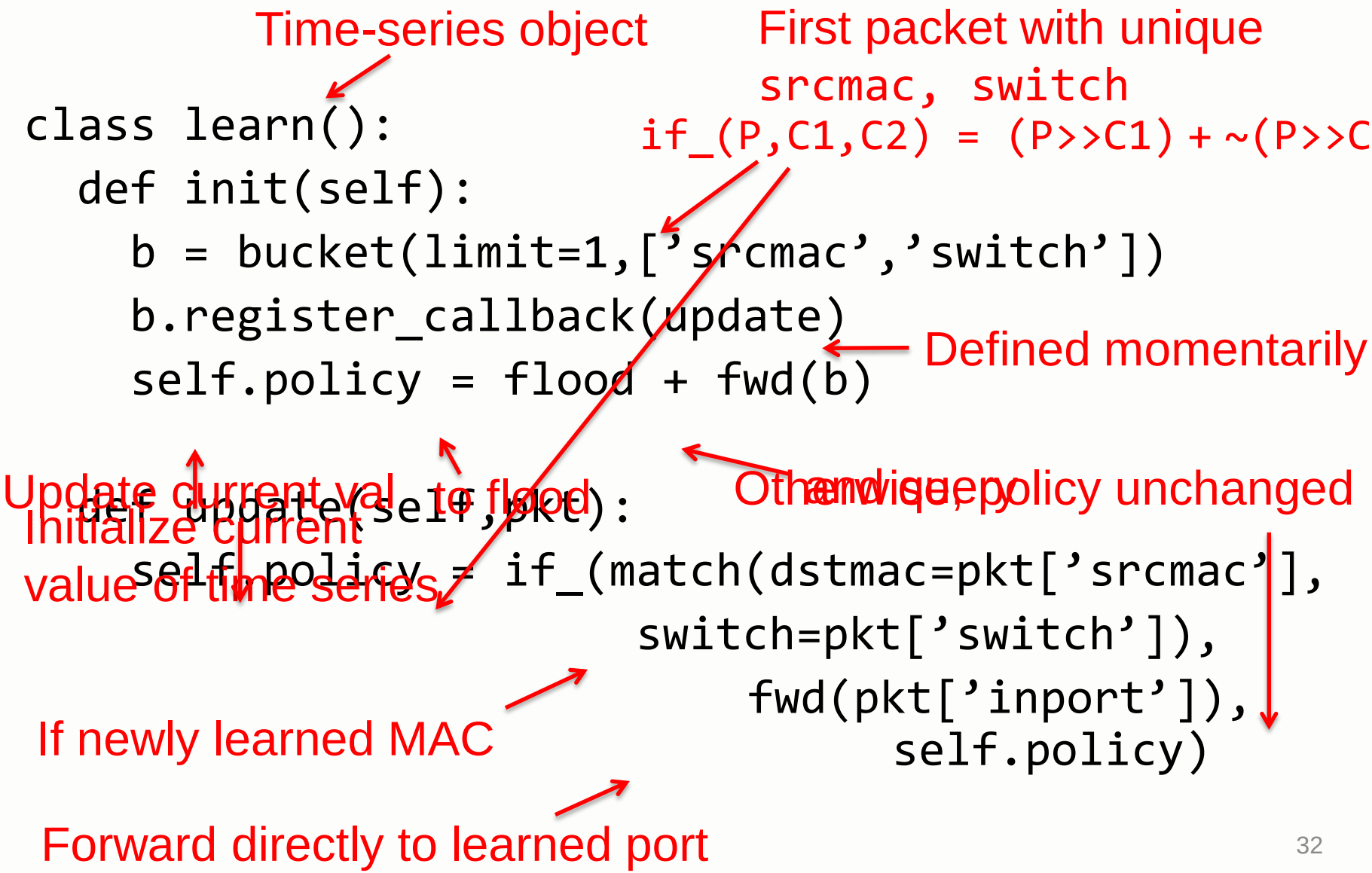
How Do We Change Policies?

Dynamic policy

a time-series of policies



MAC-Learning Module

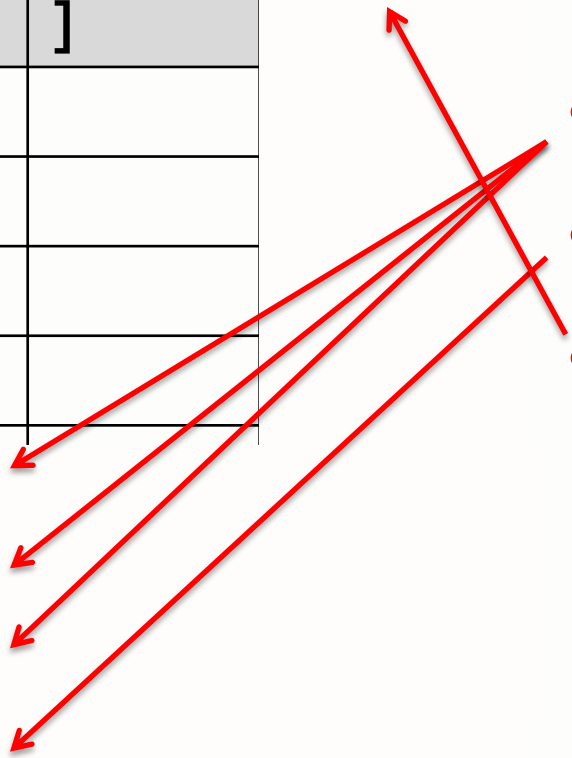


Pyretic's Design

- Monitoring Load Balancer
  - Encode Policies as Functions
  - Compositional Operators
  - Queries as Forwarding Policies
- MAC-Learning Module
  - Dynamic Policies
- "One Big Switch" Topology Abstraction
  - Extensible packet model

Extensible Pyretic Packet Model

Field	Val[0]
srcmac	
dstmac	
proto	
srcip	

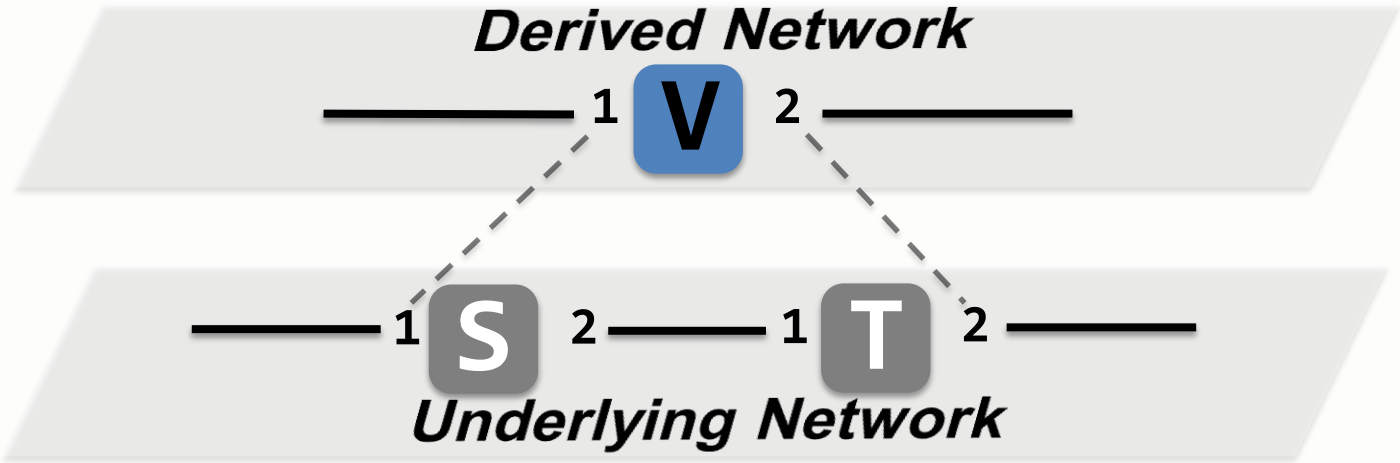


- All OpenFlow fields
- Location fields
- Virtual fields
- Stacks of values
  - push(h=v)
  - pop(h)
  - Actions and matches use (currently) top value

vswitch		

Implemented on OpenFlow by mapping extended field values to VLAN tags/MPLS labels

“One Big Switch” Topology Abstraction



- Simplest of topology abstraction examples
- Build a distributed middlebox
  by running centralized middlebox app on V!

Topology Virtualization

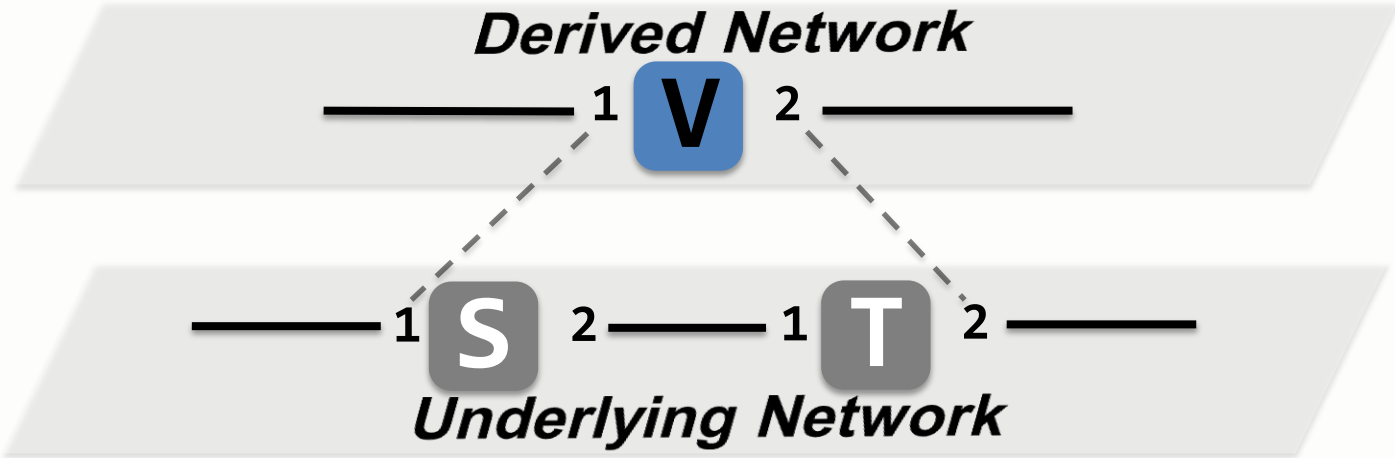
```
def virtualize(derived,transformation)
```

that "does" the `derived` policy on the abstract topology (i.e., on node V)

Returns a new policy for the underlying network (i.e., on nodes S and T)

Defines transformation policies:

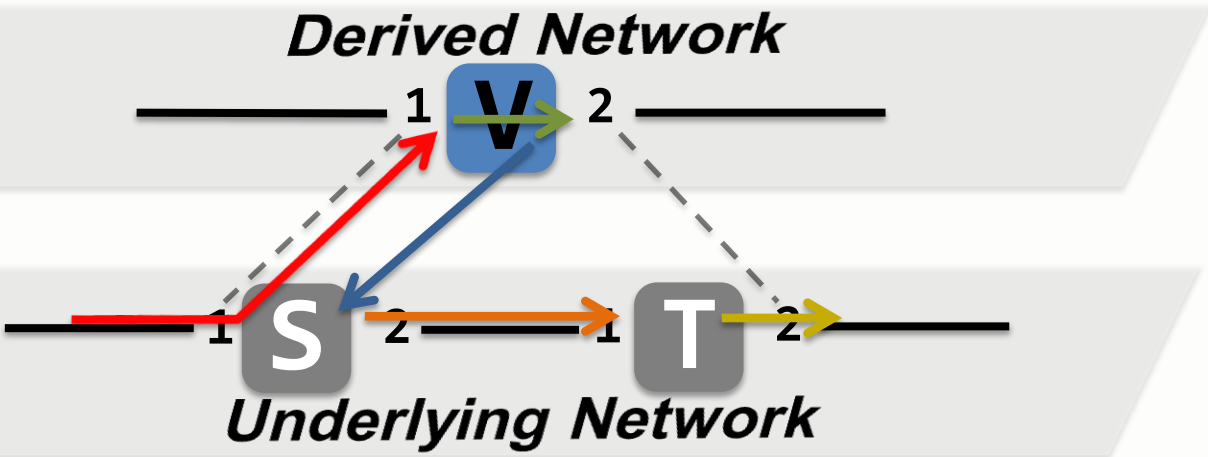
- `ingress_policy` handles packets entering abstract switch
- `fabric_policy` moves packets through abstract switch
- `egress_policy` handles packets exiting abstract switch



Simplified implementation of `virtualize(derived,transformation)`

```
return ingress_policy >> # defines part of transform
       derived        >> # app run on abstract topo
       lower_packet   >> # built-in
       fabric_policy  >> # defines part of transform
       egress_policy     # defines part of transform
```

Field	V0	V1
switch	S	V
inport	1	1
outport	2	2
vinport	1	
voutport	2	



Summary: Abstractions

Pyretic	Current APIs

Related Work:

[Frenetic, Maestro, FRESCO] / [Click]

	Pyretic	**Current APIs**
Policy	Rich Composition	**Some / Full** Composition
Network	Layered Abstract Topologies	Concrete Network
Packet	Extensible Headers	Fixed OpenFlow Headers

But only for a single software switch
not multiple hardware switches

Related Work:

[FlowVisor] / [Nicira NVP, OpenStack Quantum]

	`Pyretic`	**Current APIs**
Policy	Rich Composition	Little Composition
Network	Layered Abstract Topologies	**Disjoint Slices / Topology Hiding**
Packet	Extensible Headers	Fixed OpenFlow Headers

Both approaches support multi-tenancy,
but not topological decomposition
(or functional composition)

`Pyretic` Interpreter and Suite of Apps

Available at www.frenetic-lang.org/pyretic

- Monitoring & DPI
- Load Balancers
- Hub
- ARP
- Firewalls
- MAC learner

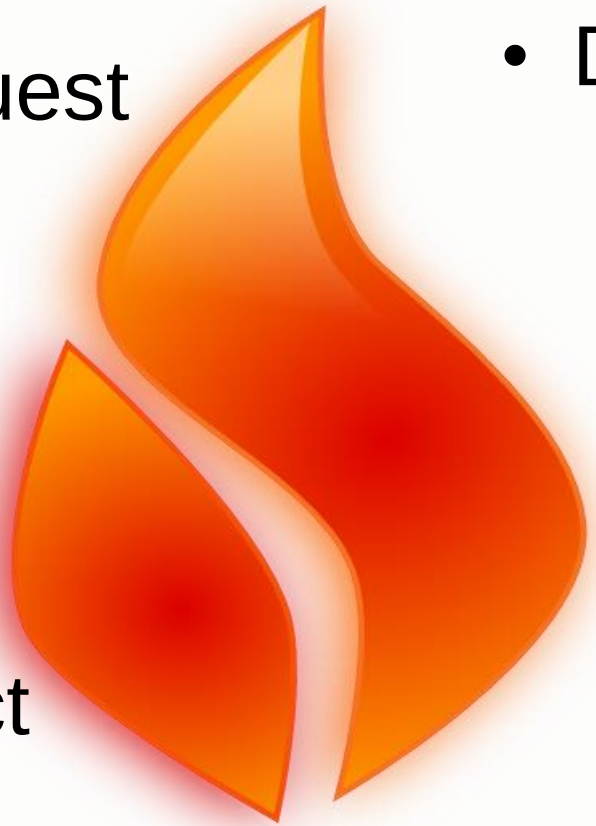
- Abstractions
  - Big switch (one-to-many)
  - Spanning tree (many-to-many)
  - Gateway (many-to-one)

And bigger applications built by combining these.

And More!

Available at www.frenetic-lang.org/pyretic

- Features Request
- Bug reporting
- Link to github
- Discuss list
- Join the project



- Dev Roadmap
  - Reactive (microflow) runtime
  - Proactive (compilation) runtime
  - Optimizations
  - Caching

Further Reading

- Pyretic description (login article): http://www.cs.princeton.edu/~jrex/papers/pyretic-login13.pdf

- Pyretic tutorial: http://youtu.be/hq7L4davQy8

- Nick Feamster's course: http://youtu.be/oa2KQ-b1SV4 http://youtu.be/UKxjitMqn88

Τέλος Ενότητας

Χρηματοδότηση

- Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στα πλαίσια του εκπαιδευτικού έργου του διδάσκοντα.
- Το έργο «**Ανοικτά Ακαδημαϊκά Μαθήματα στο Πανεπιστήμιο Κρήτης**» έχει χρηματοδοτήσει μόνο τη αναδιαμόρφωση του εκπαιδευτικού υλικού.
- Το έργο υλοποιείται στο πλαίσιο του Επιχειρησιακού Προγράμματος «Εκπαίδευση και Δια Βίου Μάθηση» και συγχρηματοδοτείται από την Ευρωπαϊκή Ένωση (Ευρωπαϊκό Κοινωνικό Ταμείο) και από εθνικούς πόρους.

Ευρωπαϊκή Ένωση
Ευρωπαϊκό Κοινωνικό Ταμείο
ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ
ΕΚΠΑΙΔΕΥΣΗ ΚΑΙ ΔΙΑ ΒΙΟΥ ΜΑΘΗΣΗ
επένδυση στην κοινωνία της γνώσης
ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ ΚΑΙ ΘΡΗΣΚΕΥΜΑΤΩΝ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ
ΕΣΠΑ 2007-2013
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ
Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης

Σημειώματα

Σημείωμα αδειοδότησης

- Το παρόν υλικό διατίθεται με τους όρους της άδειας χρήσης Creative Commons Αναφορά, Μη Εμπορική Χρήση, Όχι Παράγωγο Έργο 4.0 [1] ή μεταγενέστερη, Διεθνής Έκδοση. Εξαιρούνται τα αυτοτελή έργα τρίτων π.χ. φωτογραφίες, διαγράμματα κ.λ.π., τα οποία εμπεριέχονται σε αυτό και τα οποία αναφέρονται μαζί με τους όρους χρήσης τους στο «Σημείωμα Χρήσης Έργων Τρίτων».

[1] http://creativecommons.org/licenses/by-nc-nd/4.0/

- Ως **Μη Εμπορική** ορίζεται η χρήση:
  - που δεν περιλαμβάνει άμεσο ή έμμεσο οικονομικό όφελος από την χρήση του έργου, για το διανομέα του έργου και αδειοδόχο
  - που δεν περιλαμβάνει οικονομική συναλλαγή ως προϋπόθεση για τη χρήση ή πρόσβαση στο έργο
  - που δεν προσπορίζει στο διανομέα του έργου και αδειοδόχο έμμεσο οικονομικό όφελος (π.χ. διαφημίσεις) από την προβολή του έργου σε διαδικτυακό τόπο

- Ο δικαιούχος μπορεί να παρέχει στον αδειοδόχο ξεχωριστή άδεια να χρησιμοποιεί το έργο για εμπορική χρήση, εφόσον αυτό του ζητηθεί.

Σημείωμα Αναφοράς