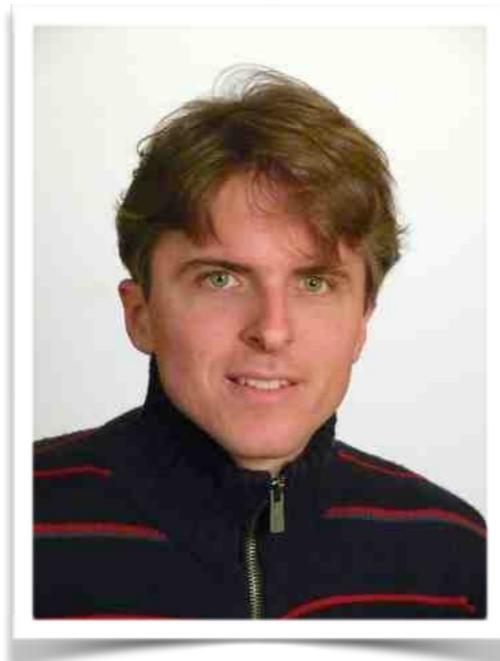


Reclaiming the Brain: Useful OpenFlow Functions in the Data Plane

Liron Schiff (Tel Aviv Uni, Israel)

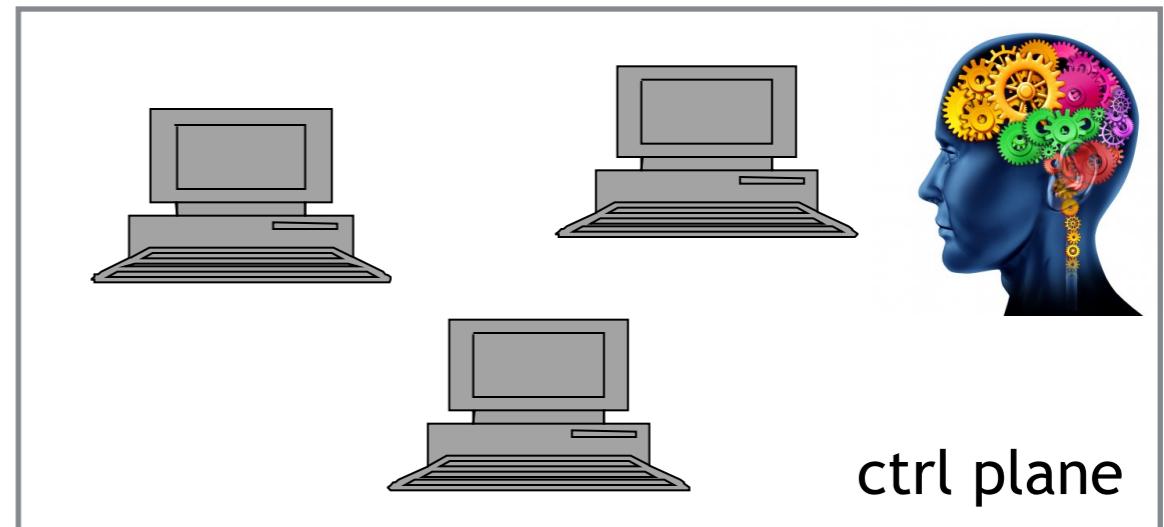
Michael Borokhovich (UT Austin, United States)

Stefan Schmid (TU Berlin & T-Labs, Germany)

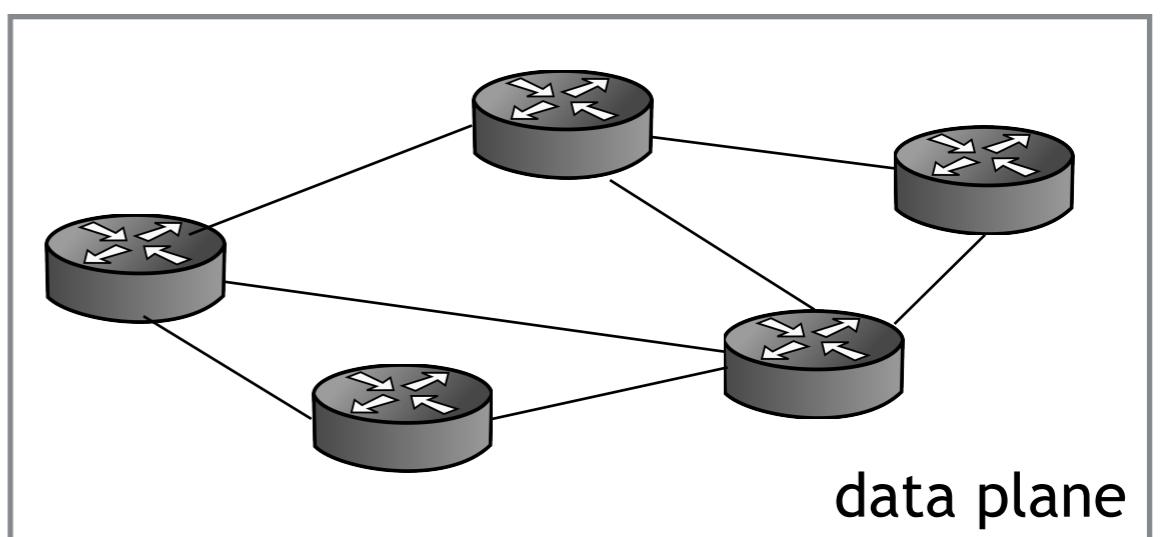


My Talk in One Slide

- * Separation of the planes enables:
 - * simplified network management and operation
 - * faster innovation



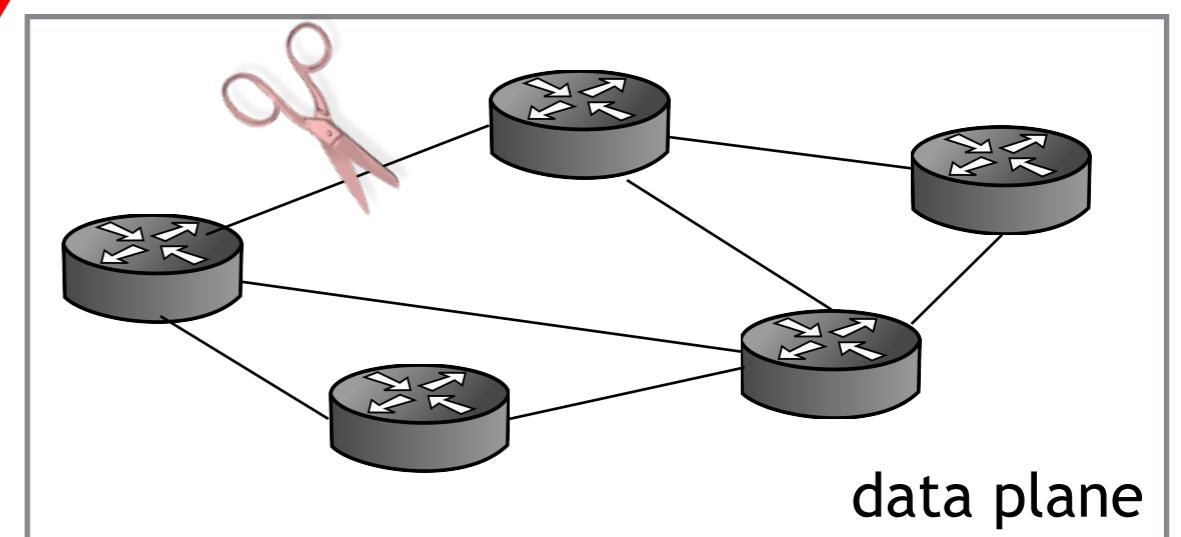
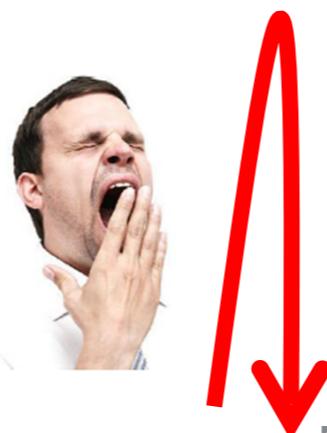
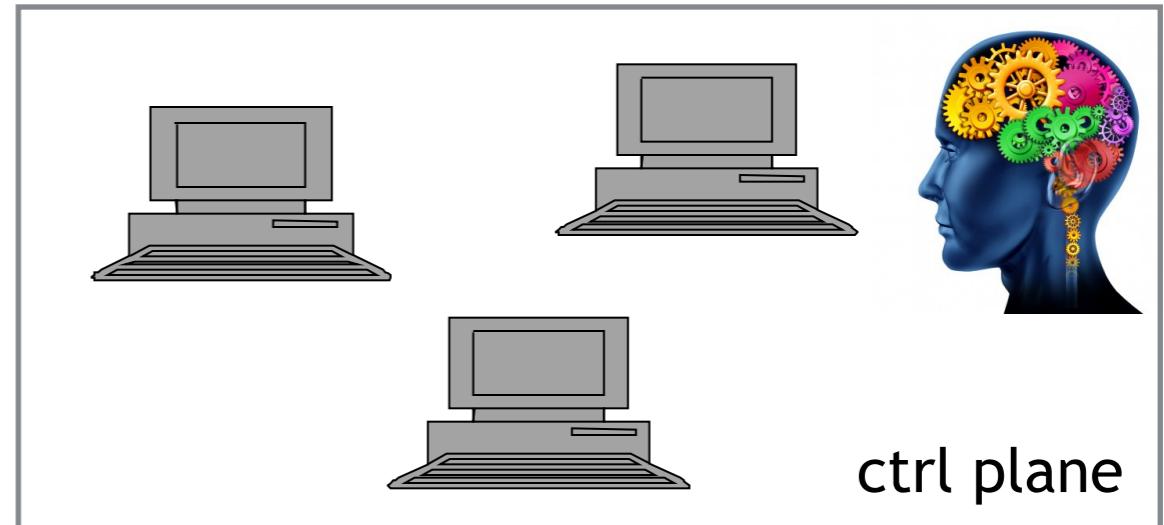
N
S



My Talk in One Slide

- * Separation of the planes enables:
 - * simplified network management and operation
 - * faster innovation

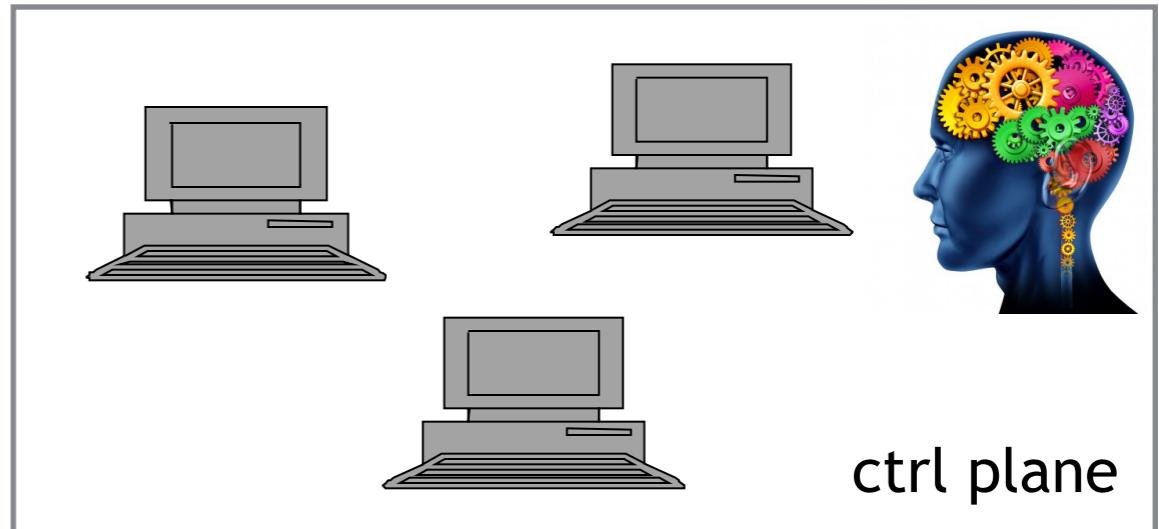
- * **However:**
 - * controller may miss certain data plane events
 - * indirection => latency



N
S

My Talk in One Slide

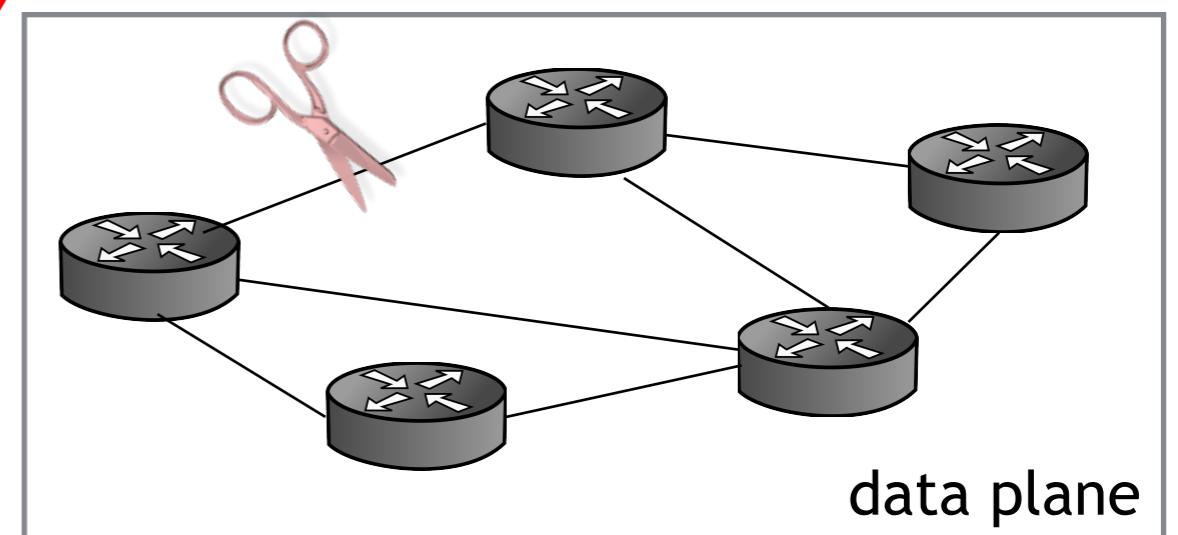
- * Separation of the planes enables:
 - * simplified network management and operation
 - * faster innovation



- * **However:**
 - * controller may miss certain data plane events
 - * indirection => latency

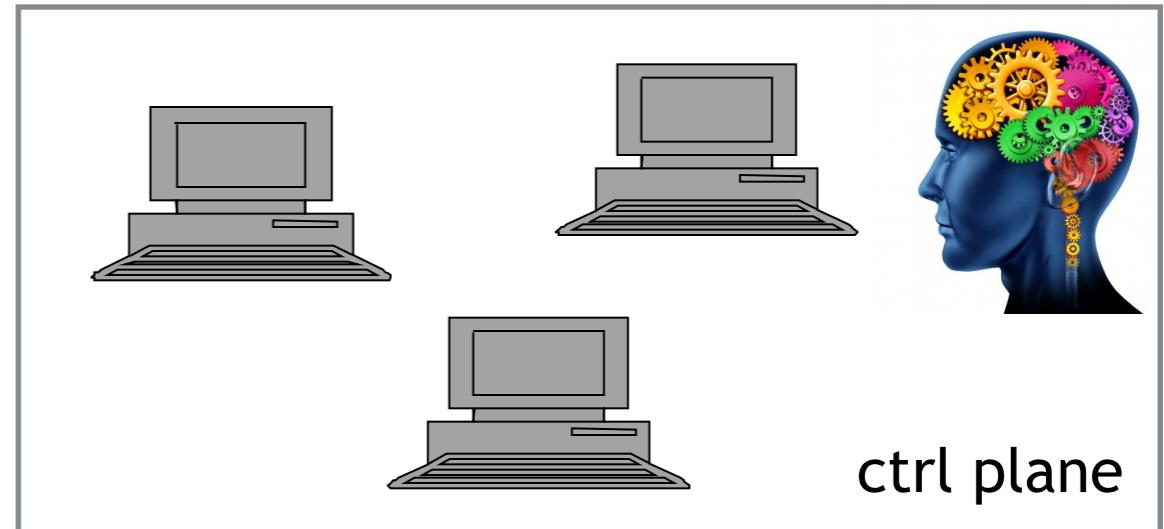


- * **What functionality should be kept in data plane? — A big question!**



My Talk in One Slide

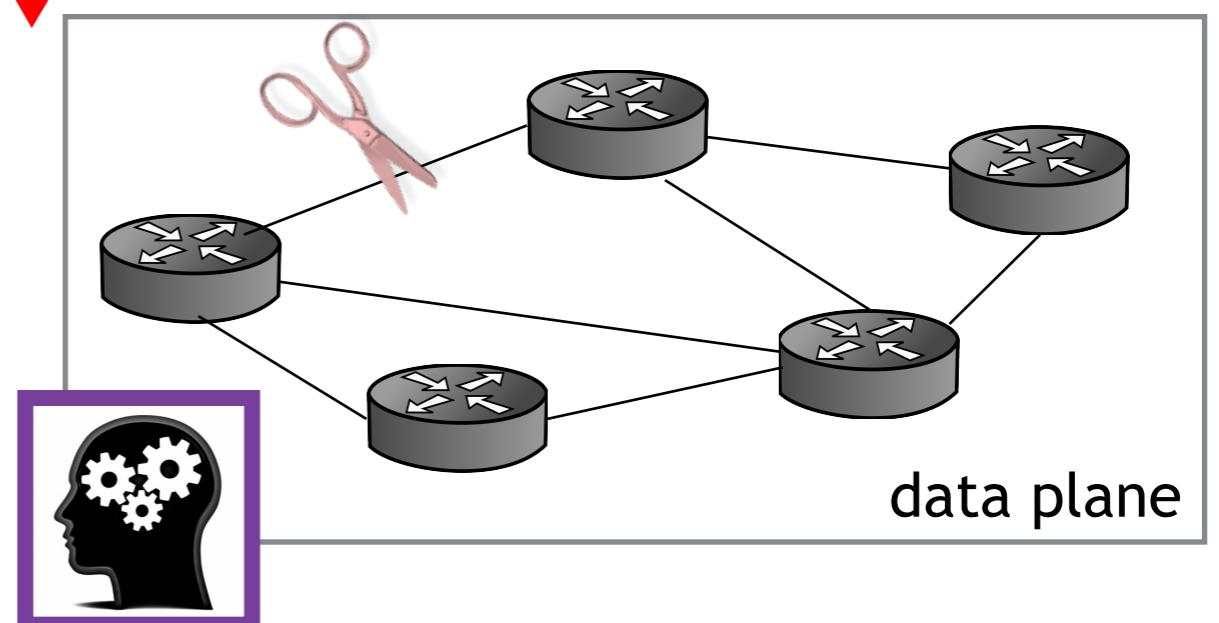
- * Separation of the planes enables:
 - * simplified network management and operation
 - * faster innovation



- * **However:**
 - * controller may miss certain data plane events
 - * indirection => latency



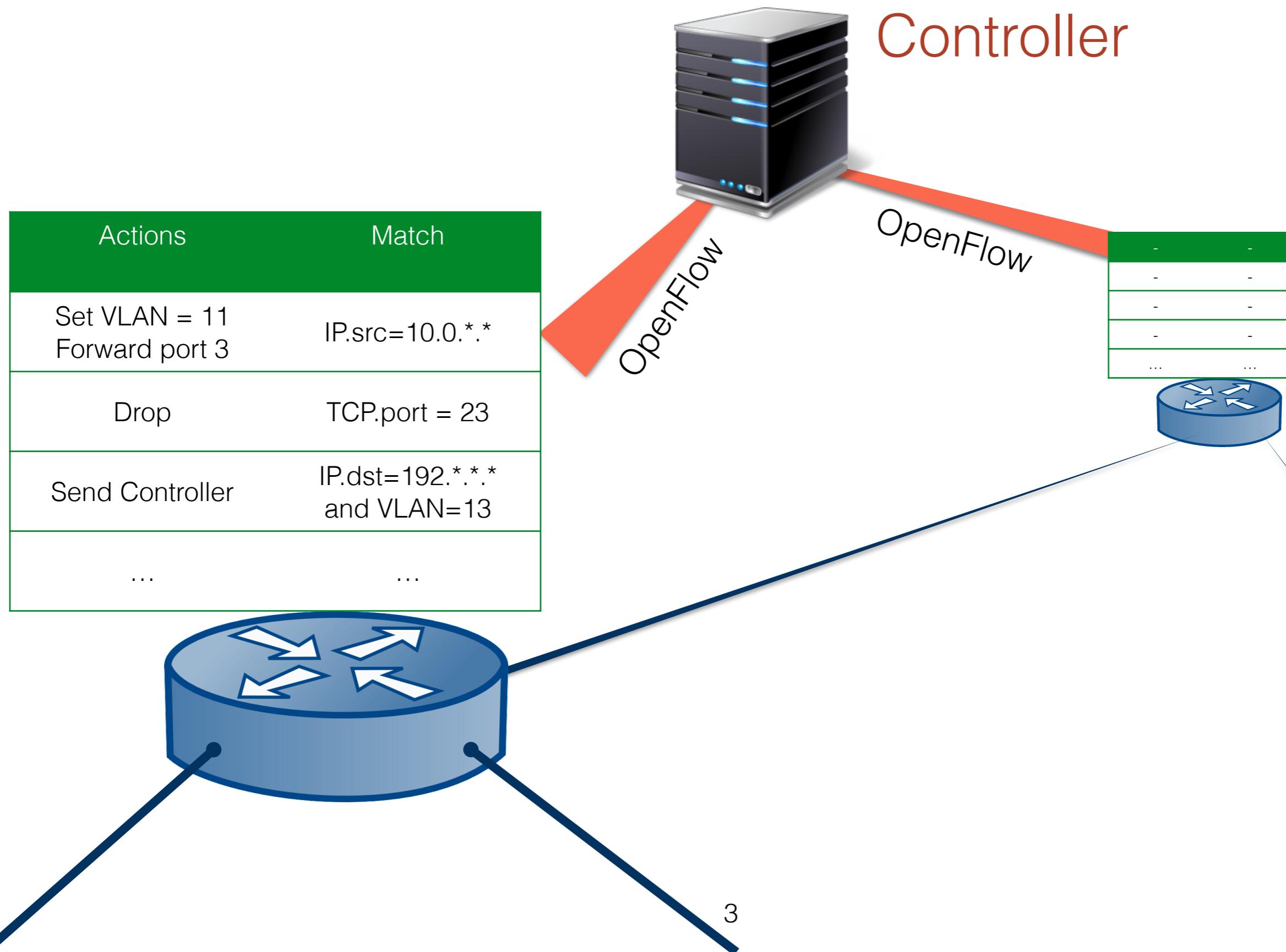
- * **What functionality should be kept in data plane? — A big question!**



In this talk: example of functions that can be kept in the data plane.

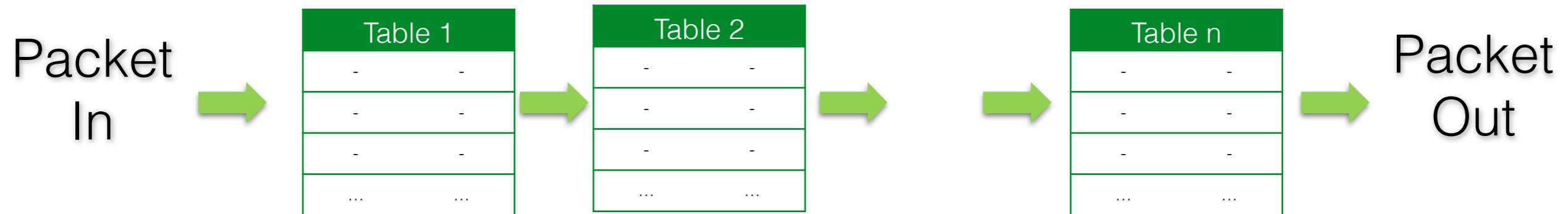
N
S

SDN and OpenFlow



OpenFlow in a Nutshell

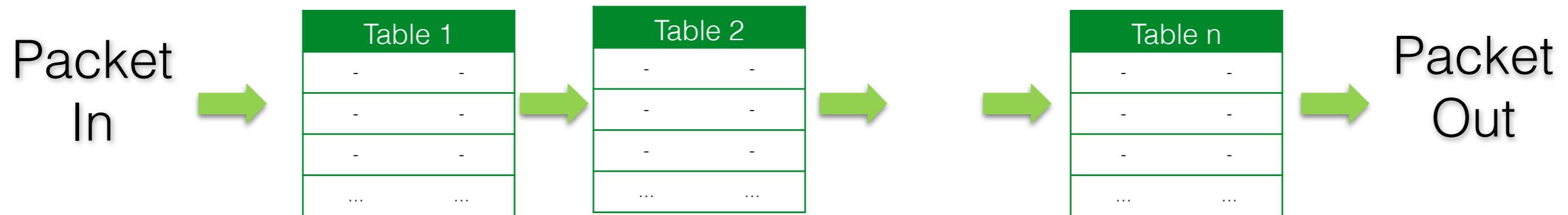
- * Switch pipeline



- * Basic Actions
 - * Set a field
 - * Append a label
 - * Forward to a port/controller/flood
 - * Goto Table x

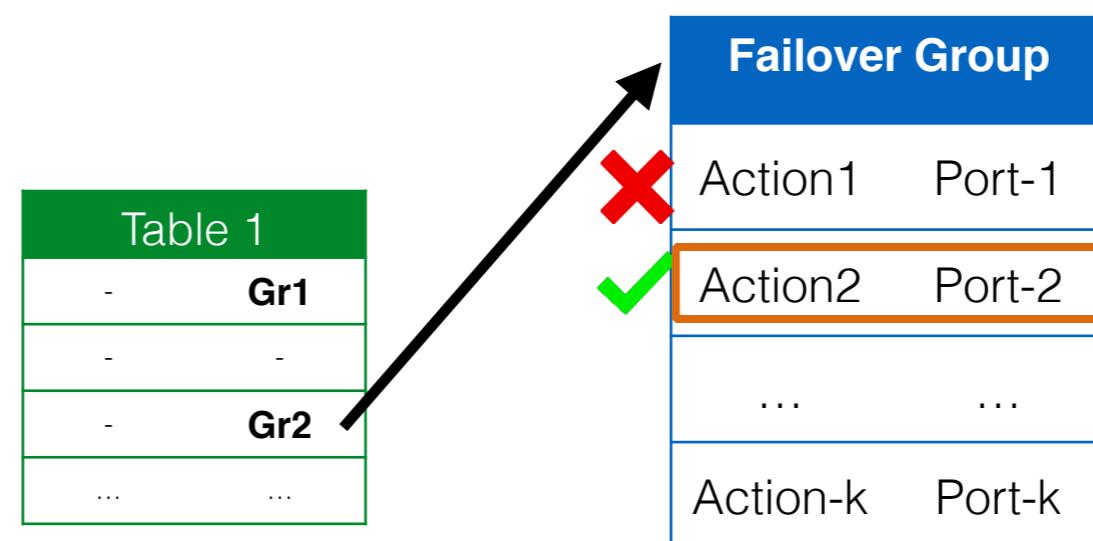
OpenFlow in a Nutshell

- * Switch pipeline



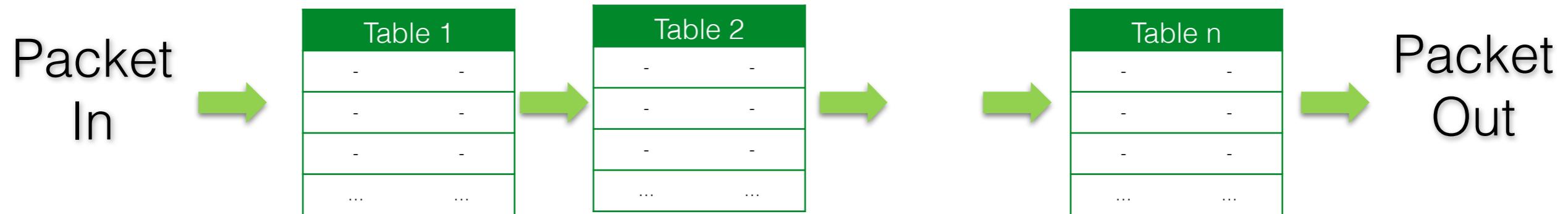
- * Basic Actions
 - * Set a field
 - * Append a label
 - * Forward to a port/controller/flood
 - * Goto Table x

- * Advanced/optional Actions
 - * Link state based
 - * Round-robin selection



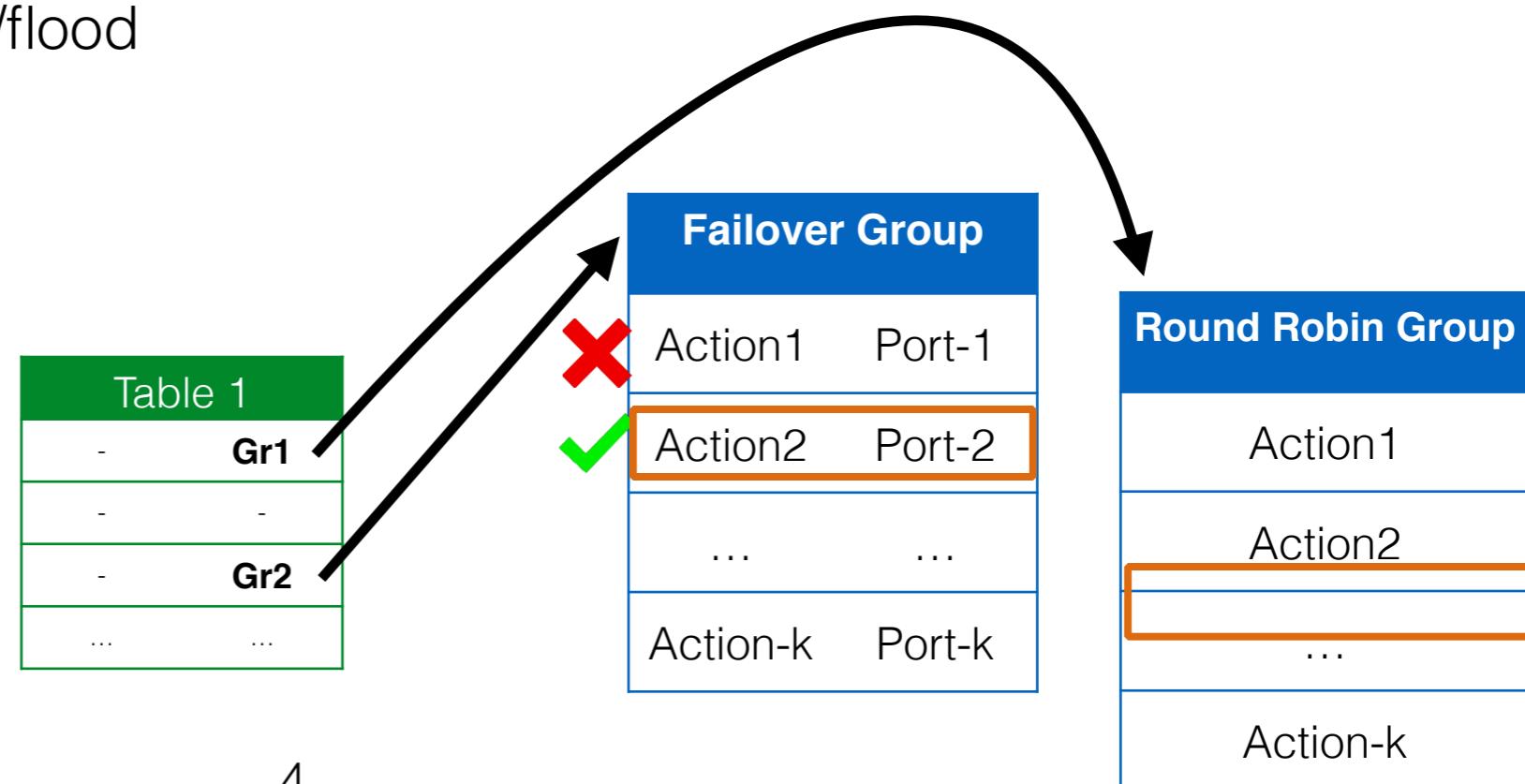
OpenFlow in a Nutshell

- * Switch pipeline



- * Basic Actions
 - * Set a field
 - * Append a label
 - * Forward to a port/controller/flood
 - * Goto Table x

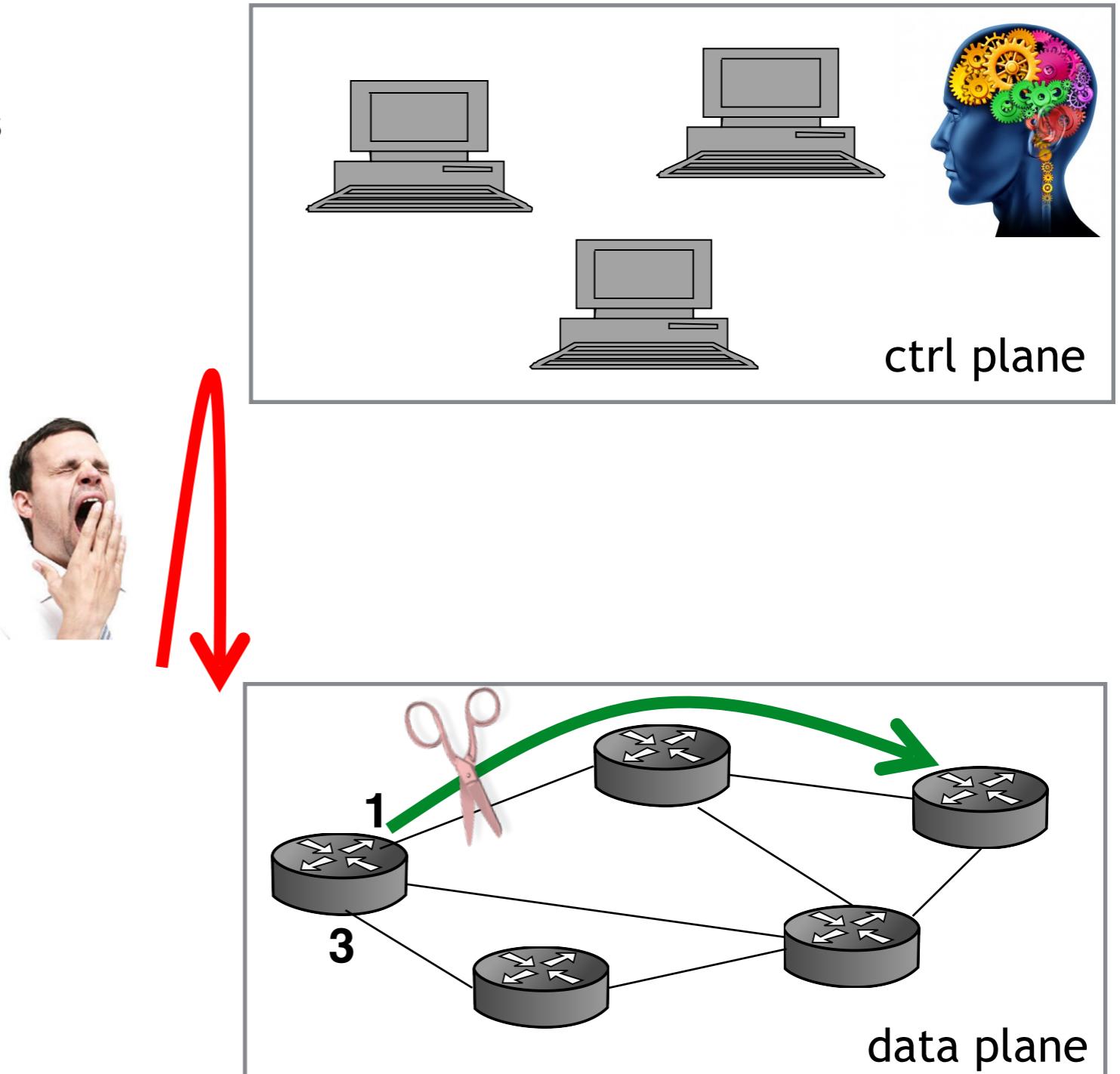
- * Advanced/optional Actions
 - * Link state based
 - * Round-robin selection



Fast Failover - Adding Brains to the South

- * Proactive reaction to link failures

Failover Group	
X	Fwd Port 1
✓	Fwd Port 3
...	...
Action-k	Port-k

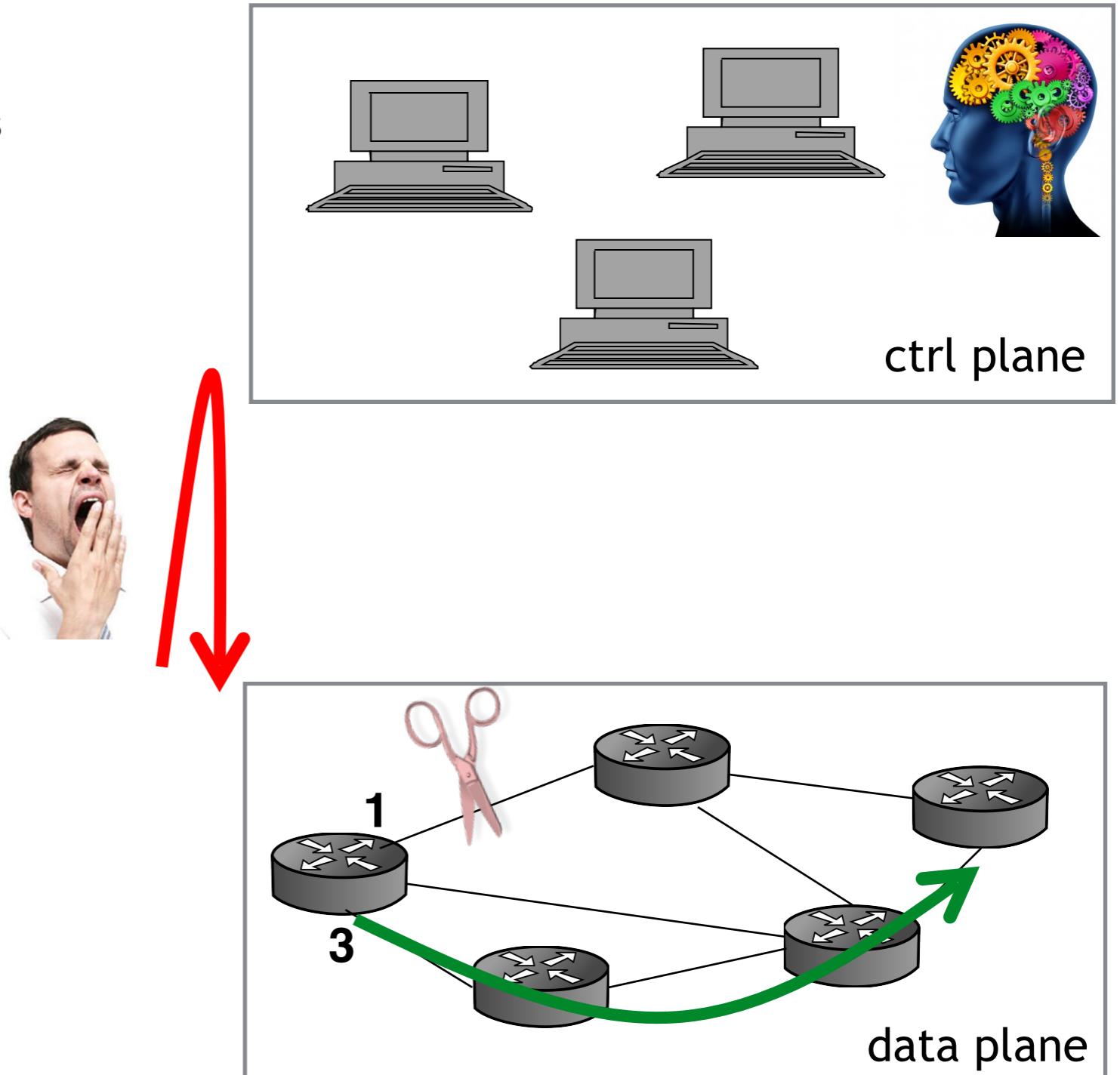


N
S

Fast Failover - Adding Brains to the South

- * Proactive reaction to link failures

Failover Group	
X	Fwd Port 1
✓	Fwd Port 3
...	...
Action-k	Port-k

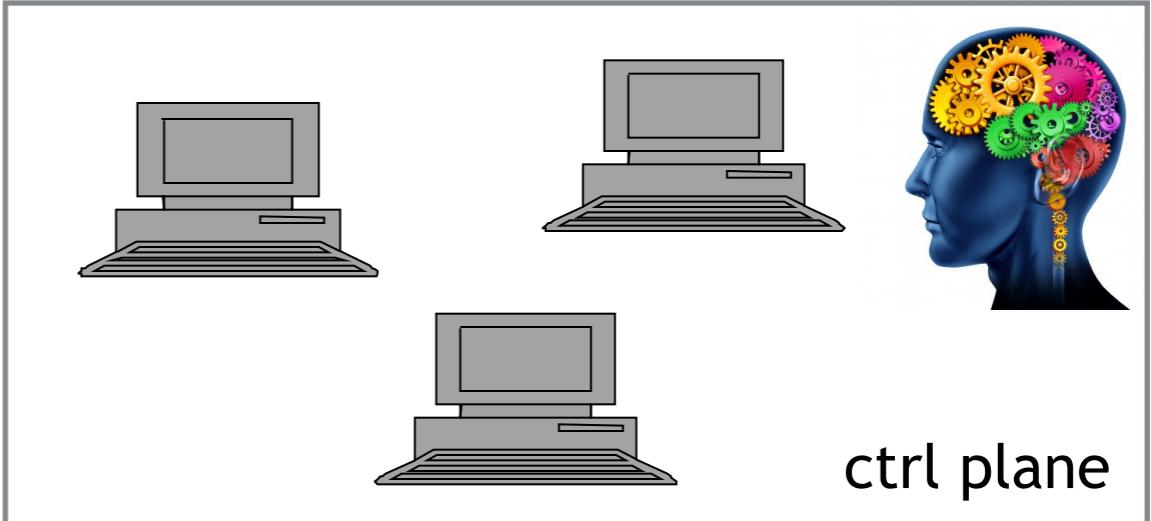


N
S

Fast Failover - Adding Brains to the South

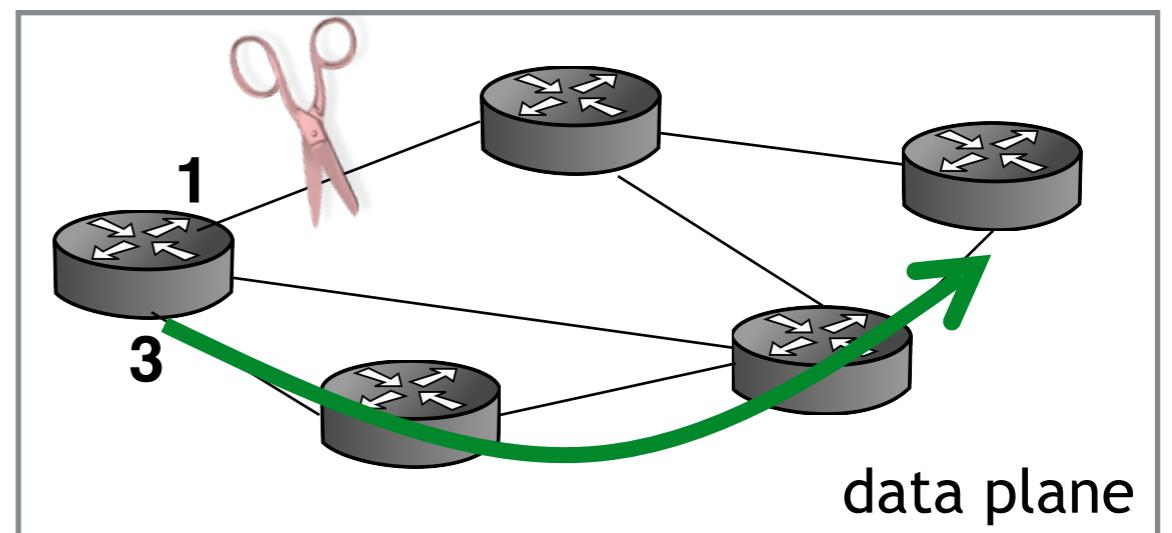
- * Proactive reaction to link failures

Failover Group	
X	Fwd Port 1
✓	Fwd Port 3
...	...
Action-k	Port-k



N
S

- * Non-trivial to use
 - * May quickly introduce loops
 - * May introduce high load
- * Much better with Tags



Fast Failover - Adding Brains to the South

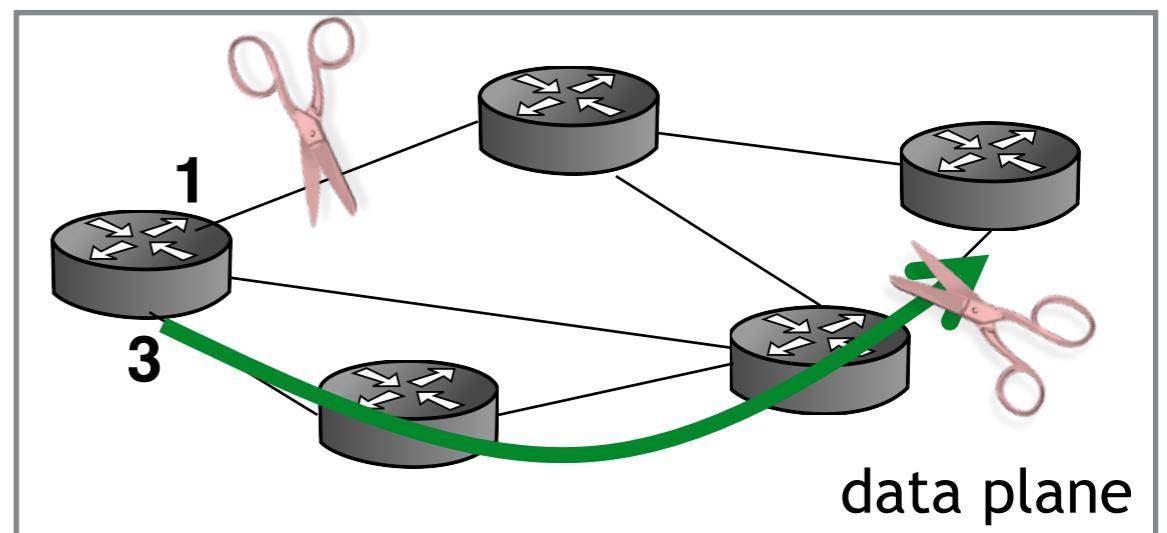
- * Proactive reaction to link failures

Failover Group	
X	Fwd Port 1
✓	Fwd Port 3
...	...
Action-k	Port-k



N
S

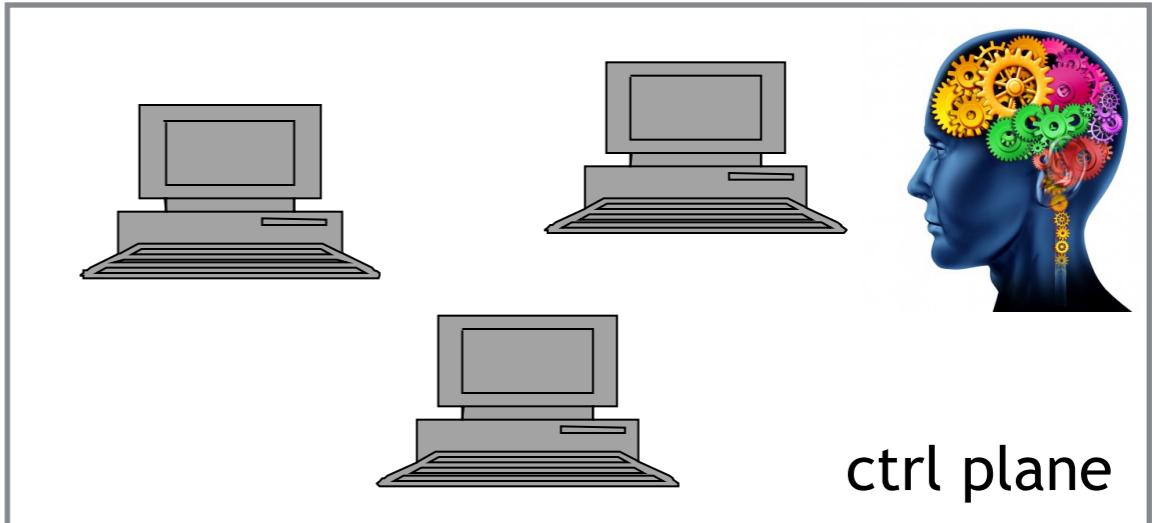
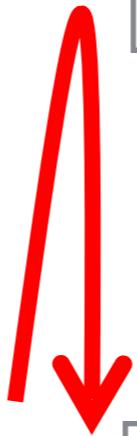
- * Non-trivial to use
 - * May quickly introduce loops
 - * May introduce high load
- * Much better with Tags



Fast Failover - Adding Brains to the South

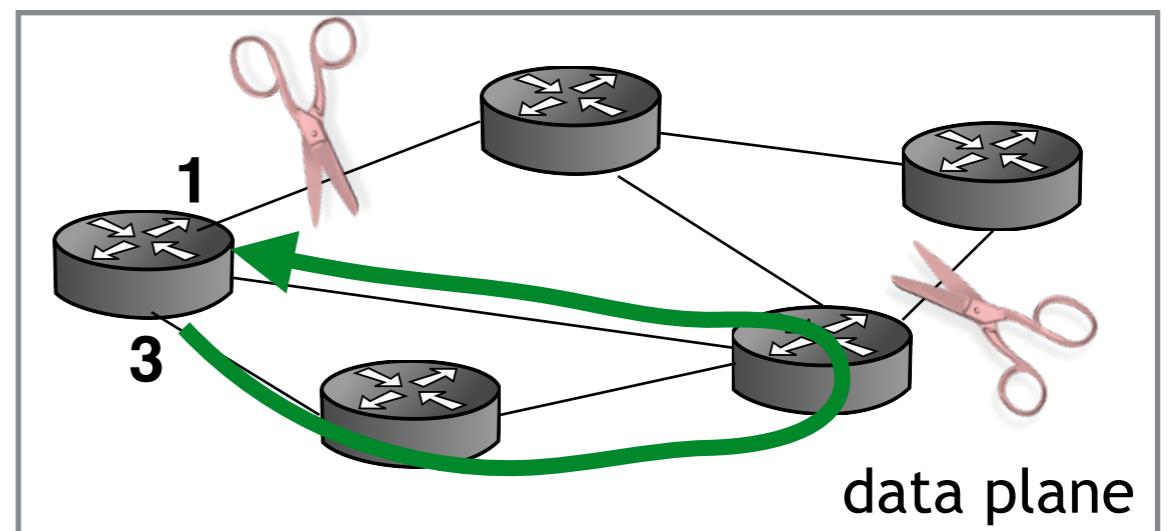
- * Proactive reaction to link failures

Failover Group	
X	Fwd Port 1
✓	Fwd Port 3
...	...
Action-k	Port-k



N
S

- * Non-trivial to use
 - * May quickly introduce loops
 - * May introduce high load
- * Much better with Tags

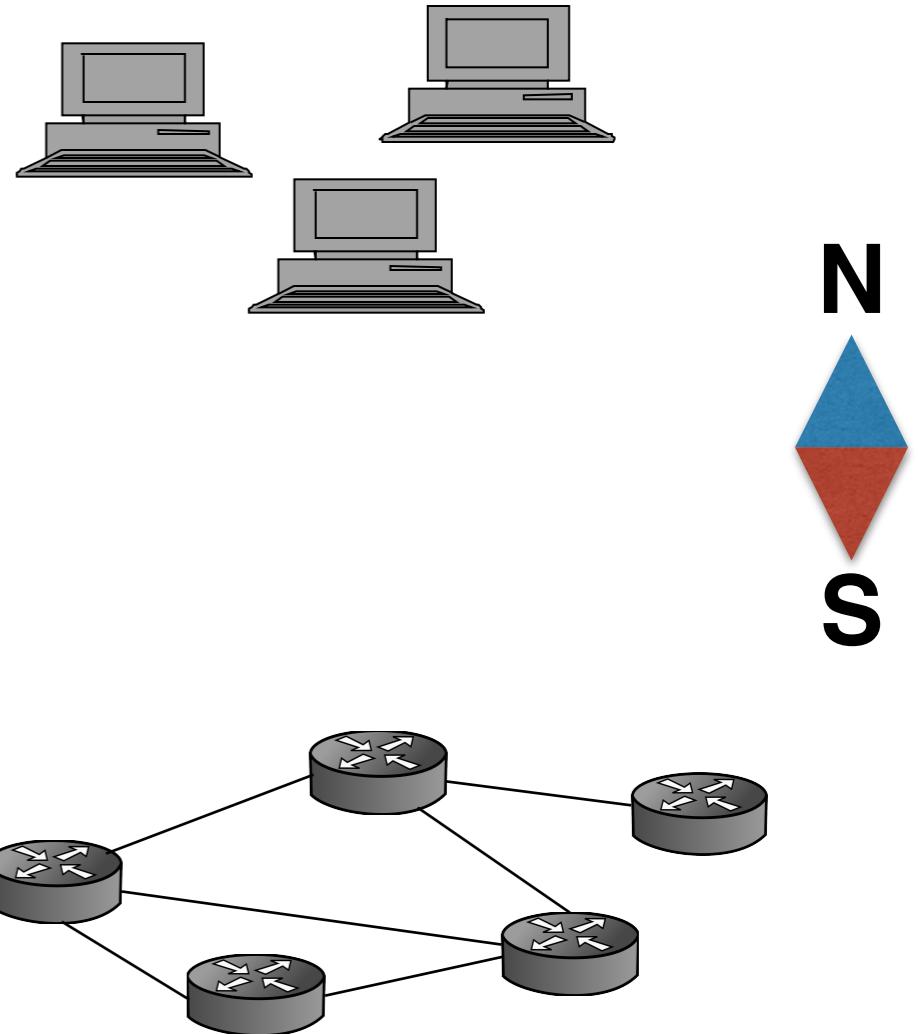


Functions in the *South*

- * Reduce interactions with the control plane
- * Make data plane more robust

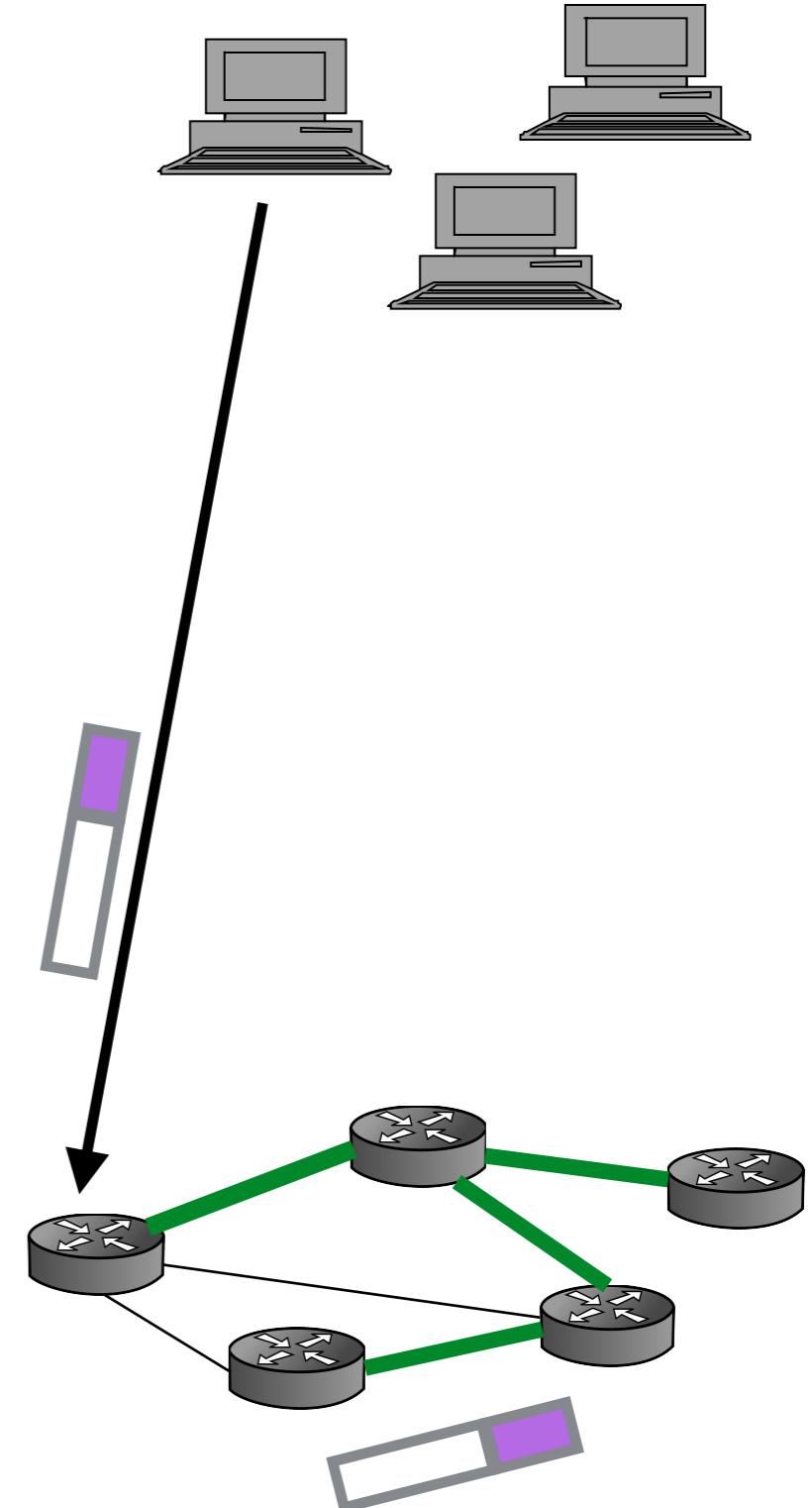
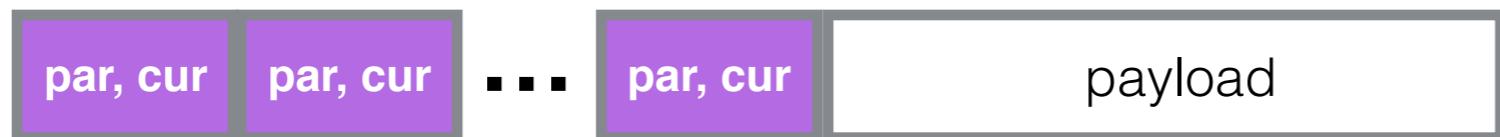
- * Monitoring functions:
 - * *Topology snapshot*
 - * *Blackhole detection*
 - * *Critical node detection*

- * Communication functions:
 - * *Anycast*



How it is possible? *SmartSouth* template.

- * SmartSouth — in-band graph DFS traversal
- * State of each node stored in the packet:
 - * parent
 - * current neighbor the node traverses
- * Implemented using a simple match-action paradigm
- * **Uses Fast Failover technique.**



How it is possible? *SmartSouth* template.

- * Pseudocode → Match&Action tables

Algorithm 1 Algorithm *SmartSouth* – Template

Input: current node: v_i , input port: in , packet global params: $pkt.start$, packet tag array: $\{pkt.v_j\}_{j \in [n]}$

Output: output port: out

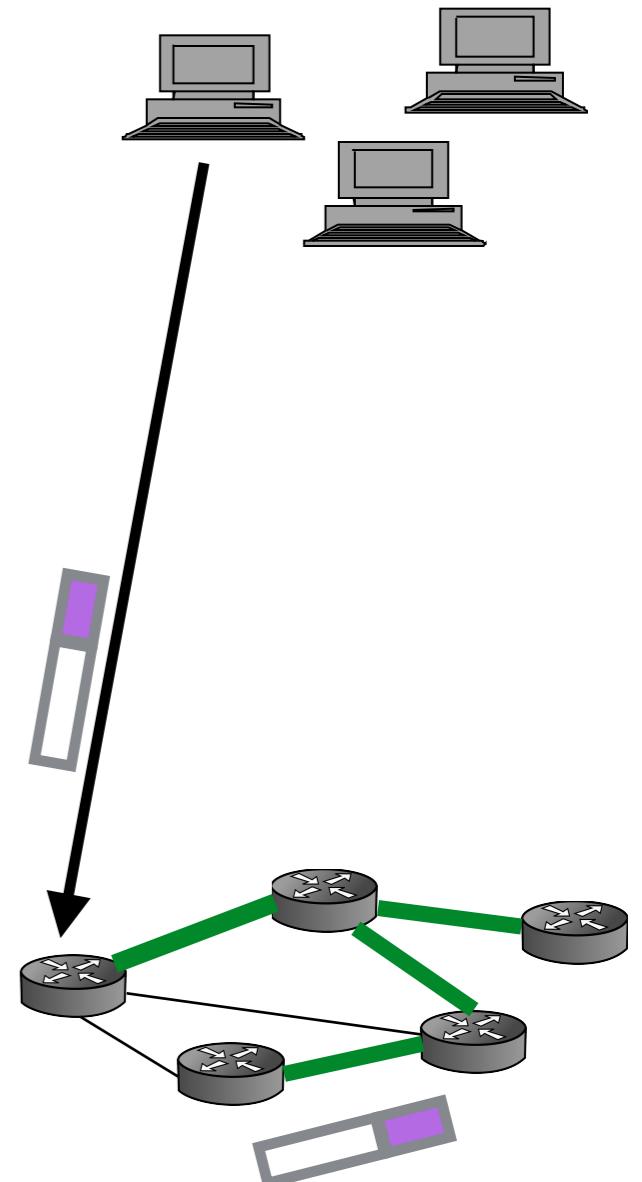
```

1: if  $pkt.start = 0$  then
2:    $pkt.start \leftarrow 1$ 
3:    $out \leftarrow 1$ 
4: else
5:   if  $pkt.v_i.cur = 0$  then
6:      $pkt.v_i.par \leftarrow in$ ;  $out \leftarrow 1$ ; First_visit()
7:   else if  $in = pkt.v_i.cur$  then
8:      $out \leftarrow pkt.v_i.cur + 1$ ; Visit_from_cur()
9:   else
10:     $out \leftarrow in$ ; Visit_not_from_cur()
11:    goto 26
12: if  $out = \Delta_i + 1$  then
13:    $out \leftarrow pkt.v_i.par$ 
14:   goto 22

15: while  $out$  failed or  $out = pkt.v_i.par$  do
16:    $out \leftarrow out + 1$ 
17:   if  $out = \Delta_i + 1$  then
18:      $out \leftarrow pkt.v_i.par$ 
19:     goto 22

20: Send_next_neighbor()
21: goto 23
22: Send_parent()
23:  $pkt.v_i.cur \leftarrow out$ 
24: if  $out = 0$  then
25:   Finish()
26: return  $out$ 

```

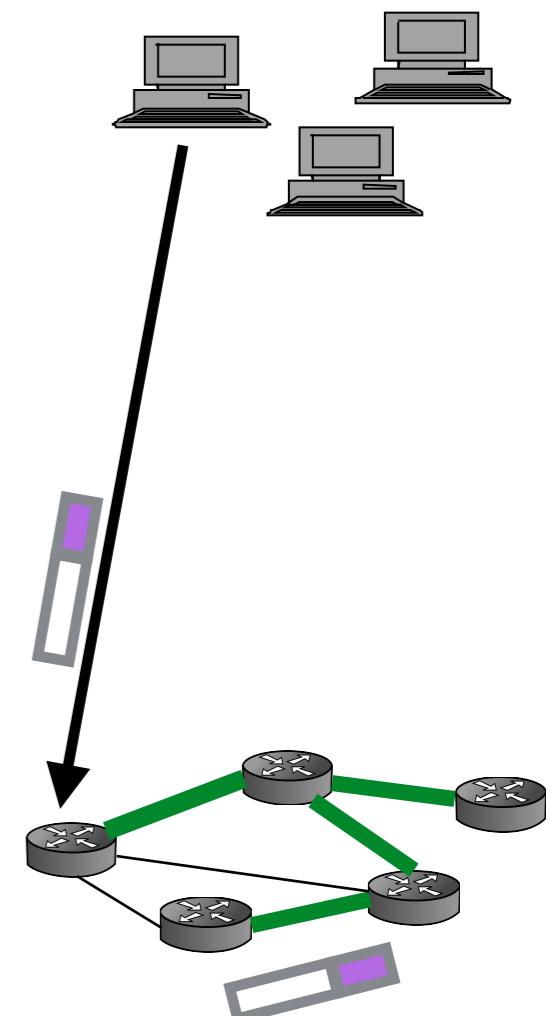


How it is possible? *SmartSouth* template.

- * Pseudocode —> Match&Action tables

```
if  $pkt.v_i.cur = 0$  then  
     $pkt.v_i.par \leftarrow in; out \leftarrow 1;$   
else if  $in = pkt.v_i.cur$  then  
     $out \leftarrow pkt.v_i.cur + 1;$ 
```

Match			Instructions
in	$pkt.v_i.cur$	$pkt.v_i.par$	
*	0	*	$pkt.v_i.par \leftarrow in$, Table 1
1	1	*	Table 2
2	2	*	Table 3
3	3	*	Table 4
...



par, cur par, cur ... par, cur payload

How it is possible? *SmartSouth* template.

- * Pseudocode → Match&Action tables

```

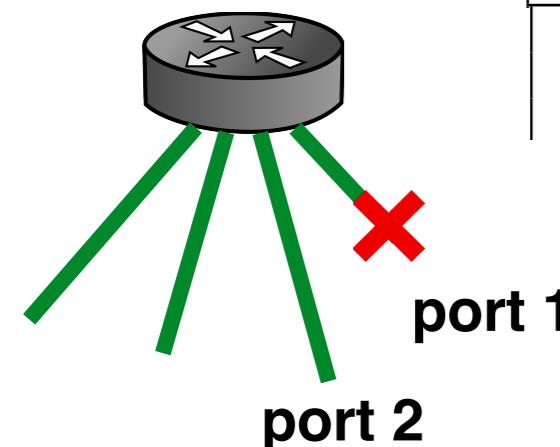
if  $pkt.v_i.cur = 0$  then
     $pkt.v_i.par \leftarrow in$ ;  $out \leftarrow 1$ ;
else if  $in = pkt.v_i.cur$  then
     $out \leftarrow pkt.v_i.cur + 1$ ;

```

```

while  $out$  failed
     $out \leftarrow out + 1$ 

```



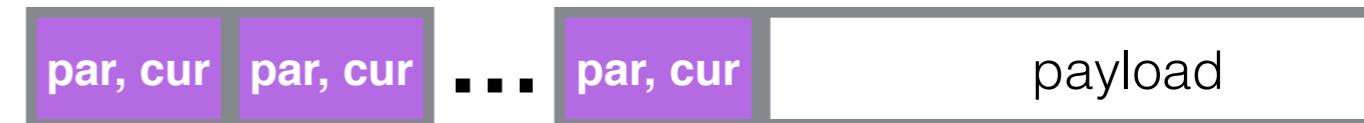
Match			Instructions
<i>in</i>	$pkt.v_i.cur$	$pkt.v_i.par$	
*	0	*	$pkt.v_i.par \leftarrow in$, Table 1
1	1	*	Table 2
2	2	*	Table 3
3	3	*	Table 4
...

Match		Instructions
<i>sb</i>		
0	Gr 1, Table 2	
1	Drop	

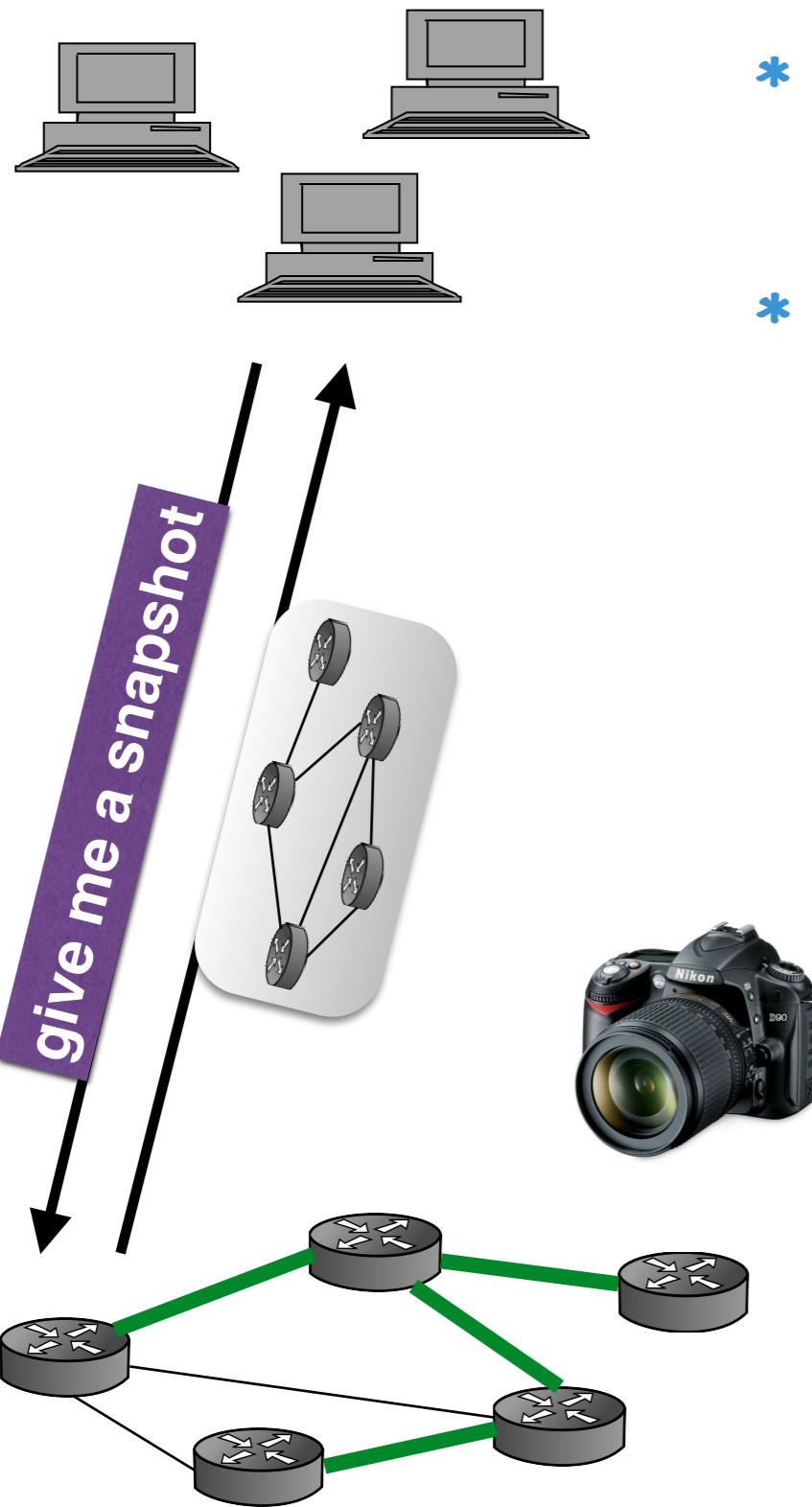
Match		Instructions
<i>sb</i>		
0	Gr 2, Table 3	
1	Drop	

...

Group	Actions
Gr 1	$\langle sb \leftarrow 1, pkt.v_i.cur \leftarrow 1, pkt.start \leftarrow 1, Fwd 1 \rangle$
Gr 2	$\langle sb \leftarrow 1, pkt.v_i.cur \leftarrow 2, pkt.start \leftarrow 1, Fwd 2 \rangle$
...	...

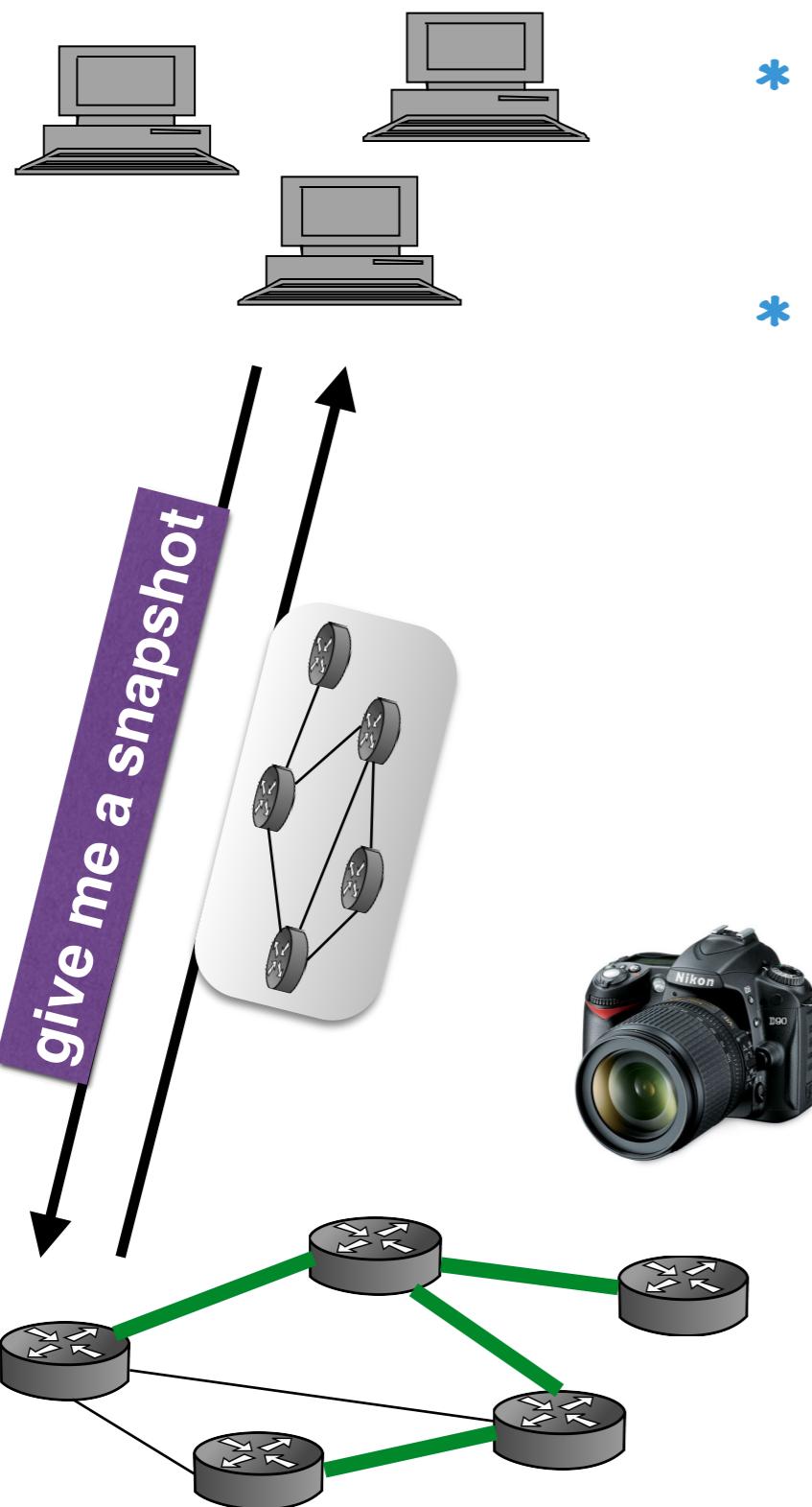


Functions in the South: *Topology Snapshot*

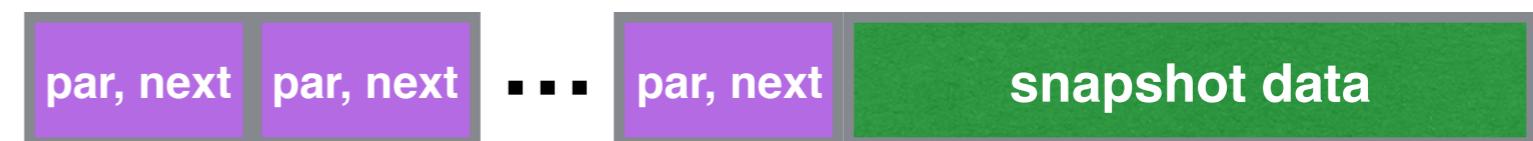


- * Fault tolerant
- * No connectivity assumption
- * Requires a single connection to controller
- * Unlike built-in “Topology service” in OpenFlow

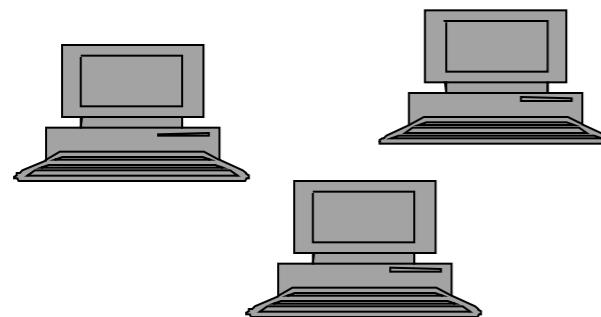
Functions in the South: *Topology Snapshot*



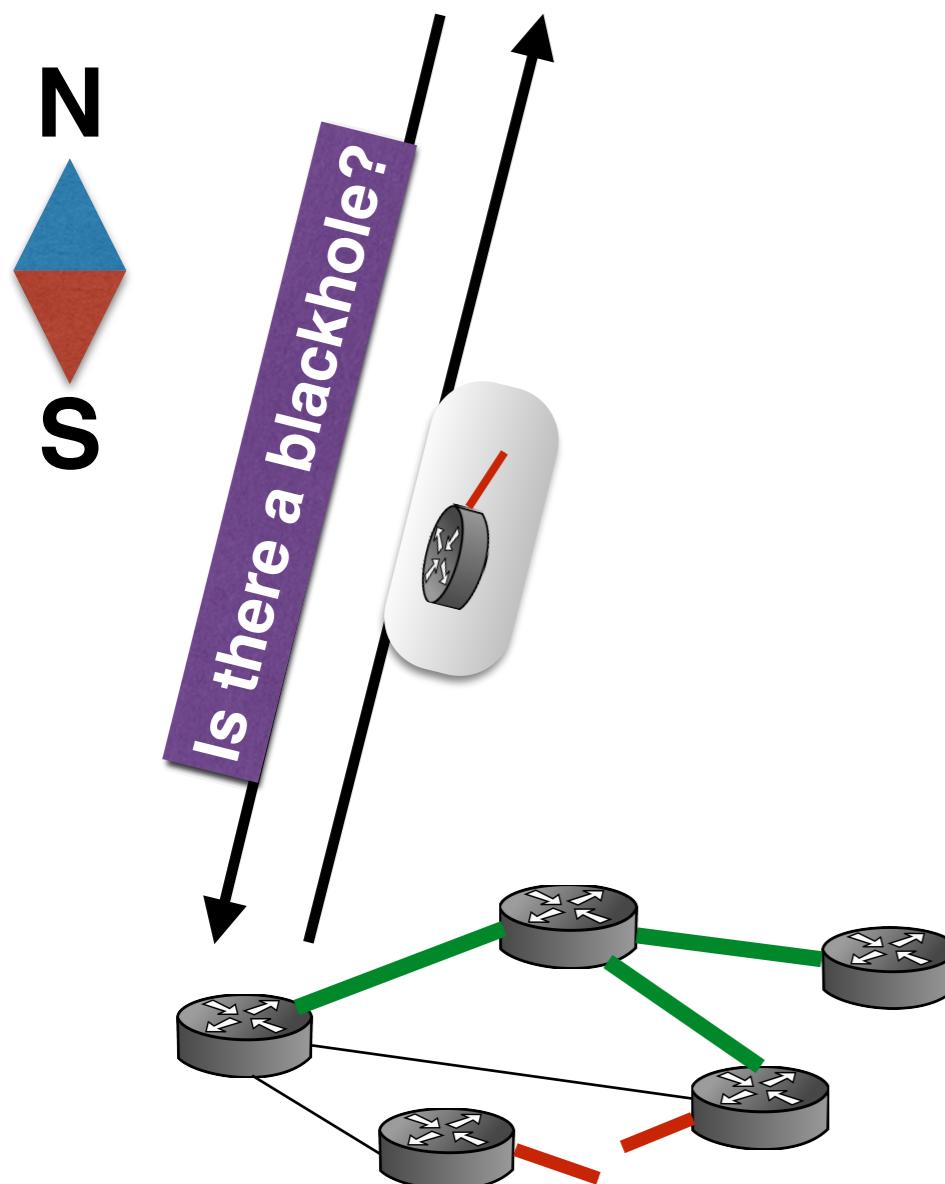
- * Fault tolerant
- * No connectivity assumption
- * Requires a single connection to controller
- * Unlike built-in “Topology service” in OpenFlow
 - * During the DFS traversal, topology information is written to the packet header



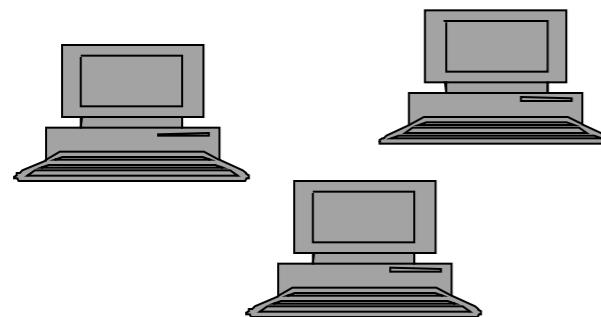
Functions in the South: *Blackhole Detection*



- * Detects connectivity loss regardless of the cause
 - * physical failure
 - * configuration errors
 - * unsupervised carrier network errors

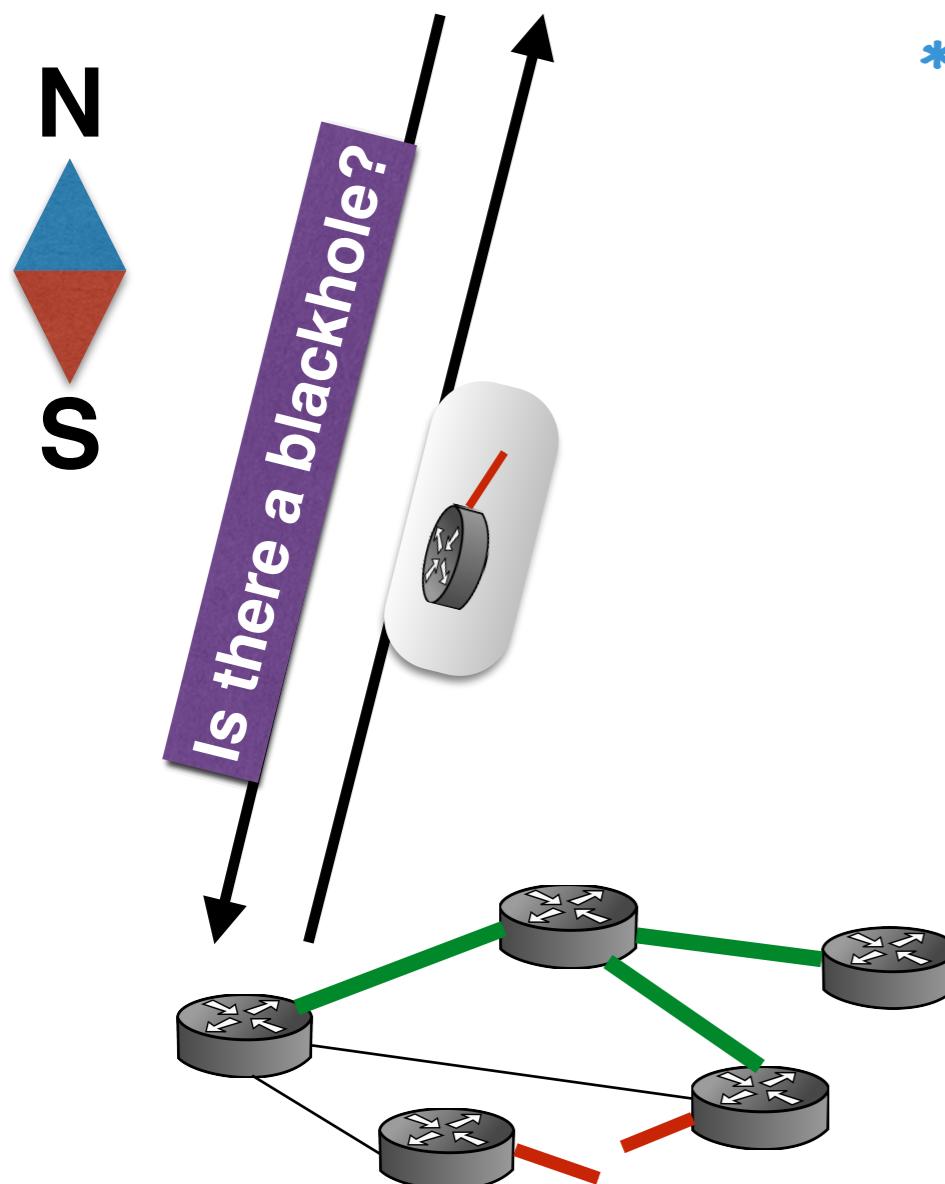


Functions in the South: *Blackhole Detection*

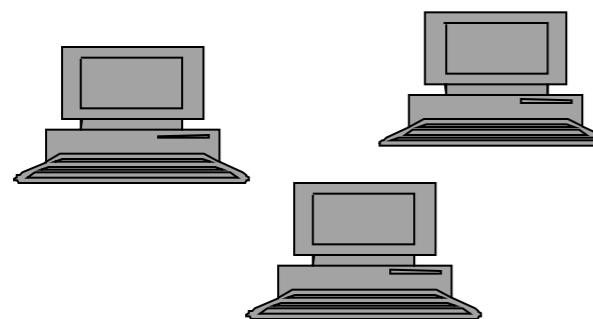


- * Detects connectivity loss regardless of the cause
 - * physical failure
 - * configuration errors
 - * unsupervised carrier network errors

- * Two possible implementations:



Functions in the South: *Blackhole Detection*

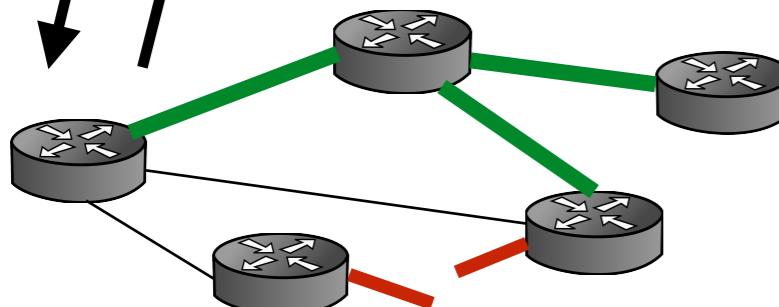
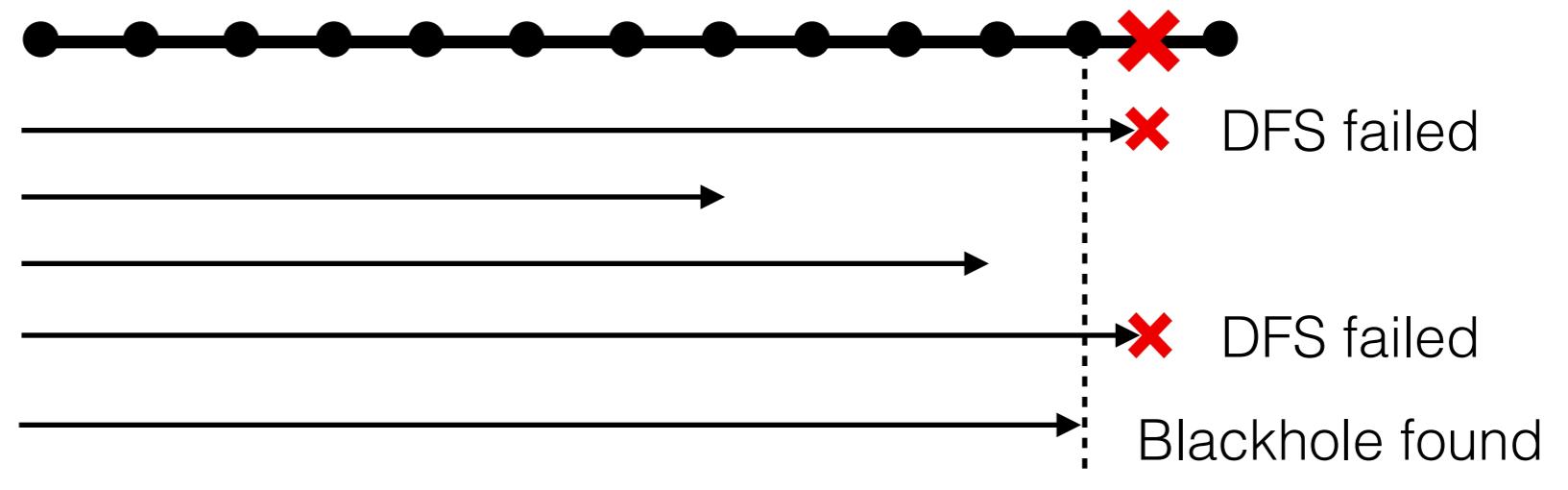


- * Detects connectivity loss regardless of the cause
 - * physical failure
 - * configuration errors
 - * unsupervised carrier network errors

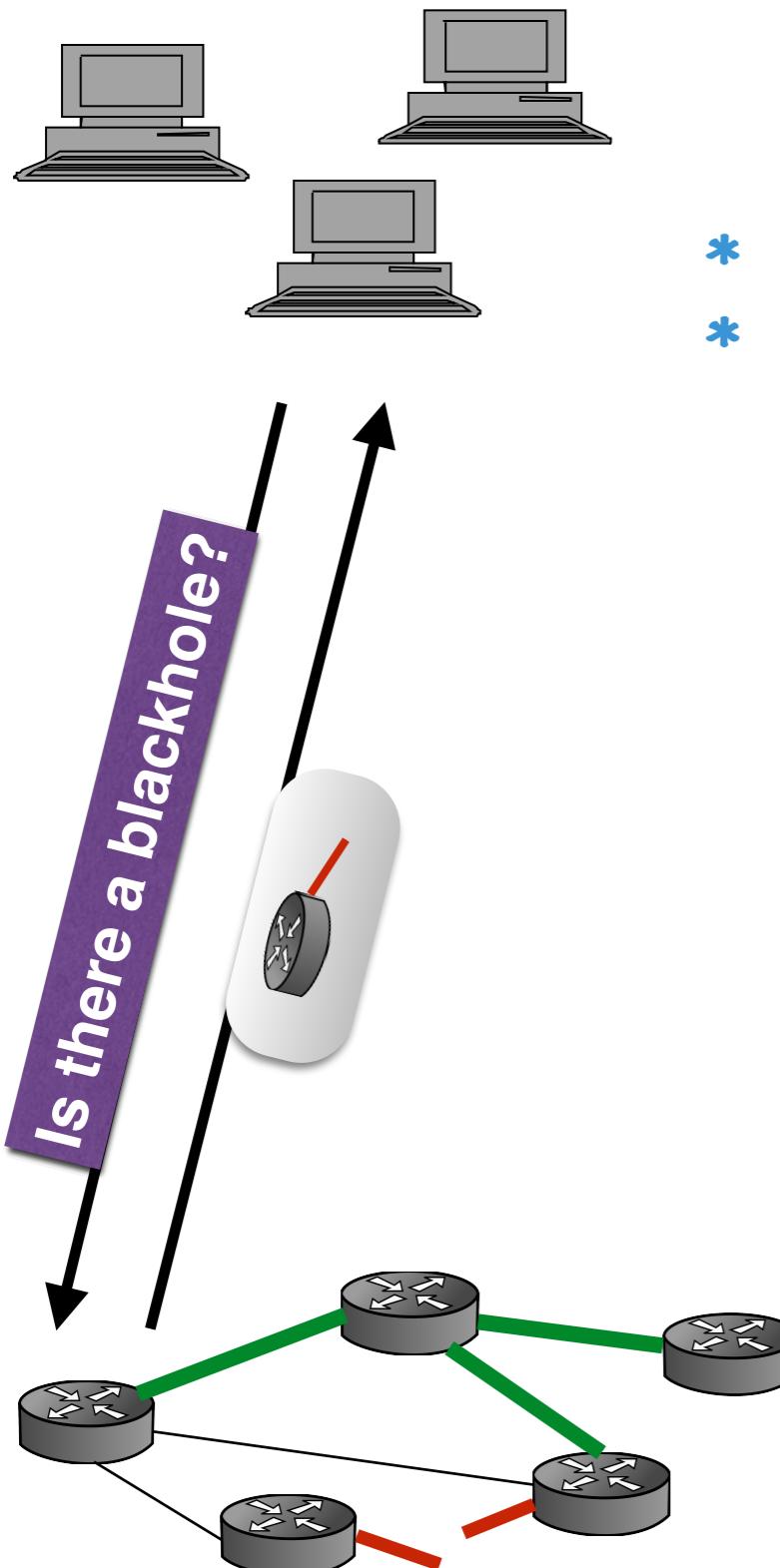
- * Two possible implementations:

- * ***DFS traversal with TTL***

- * $(\log n)$ DFS traversals (binary search)

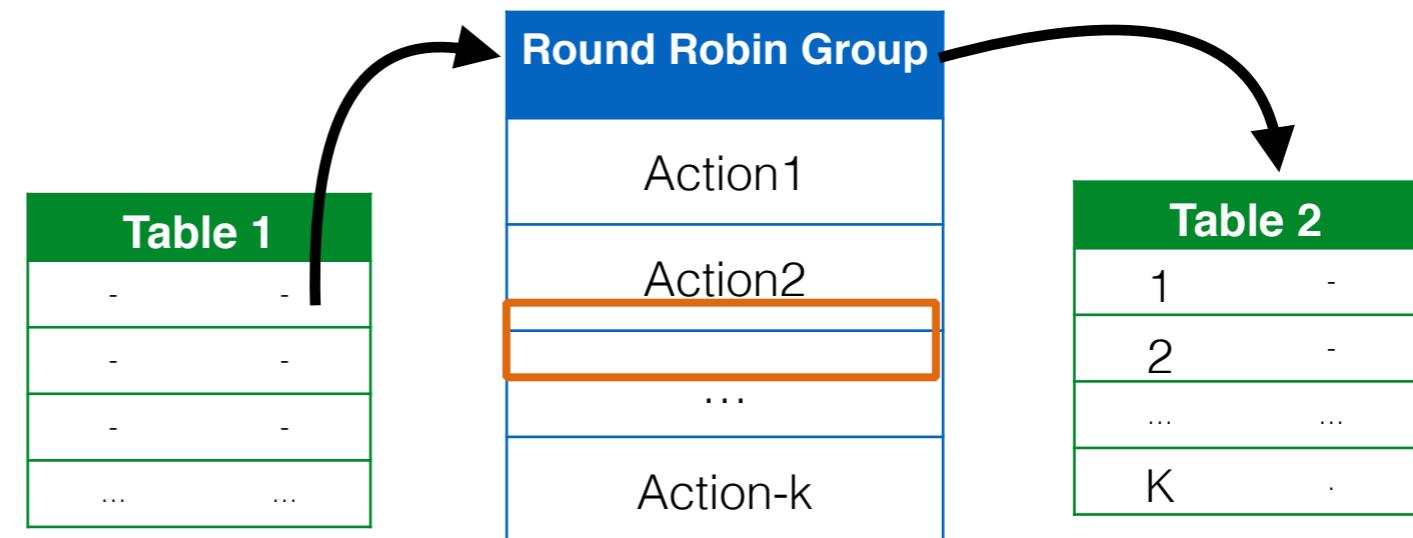


Functions in the South: *Blackhole Detection*

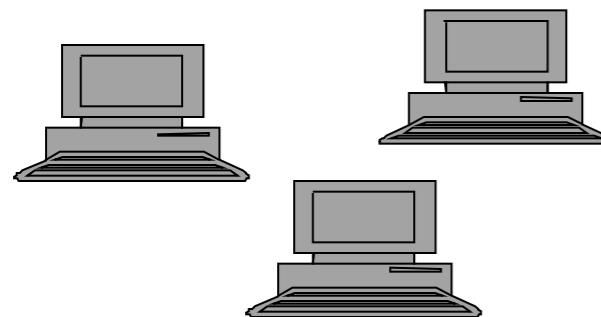


Smart “In-band” Counters

- * General counters - access only by controller
- * Our counters:
 - * access during packet processing
 - * counter value can be written to packet or metadata
 - * implemented using Round-Robin action group

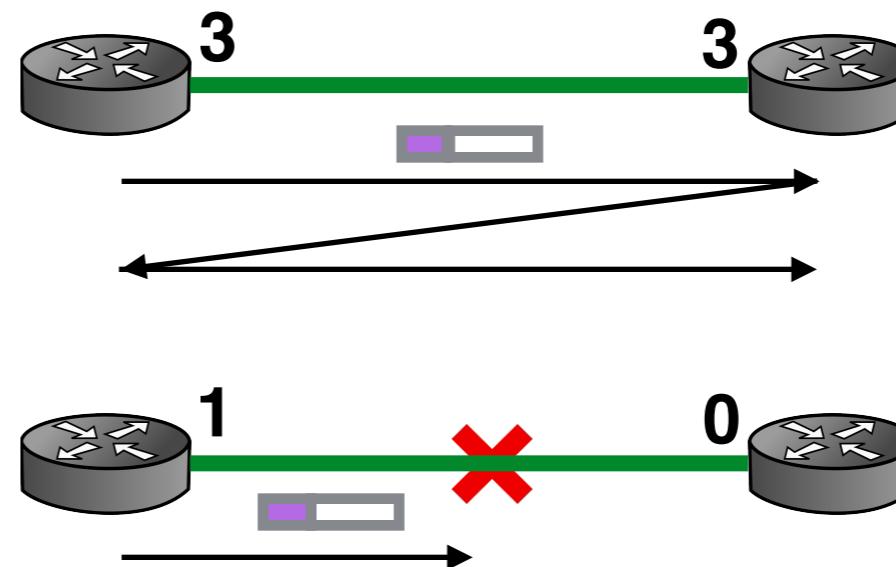
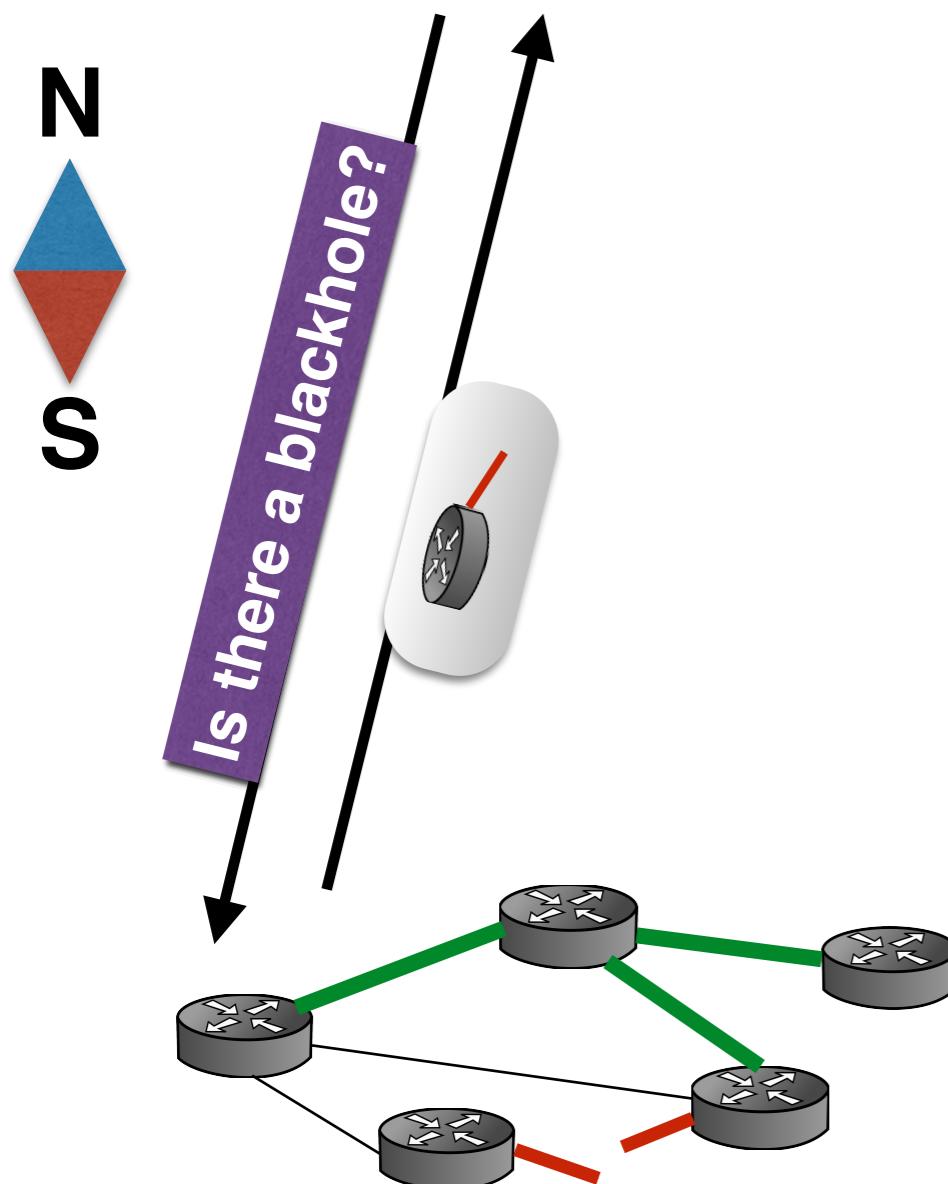


Functions in the South: *Blackhole Detection*

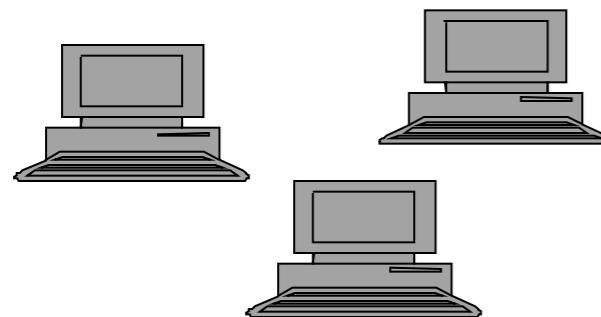


Blackhole detection with *SmartCounters*

- * Install *SmartCounter* for each port
- * Only two DFS traversals required:
- * First - back&forth on each link



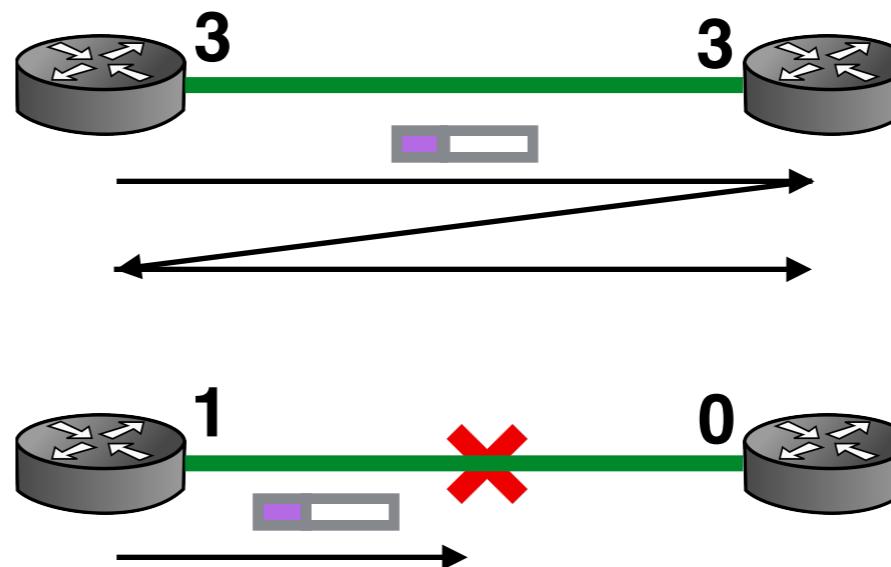
Functions in the South: *Blackhole Detection*



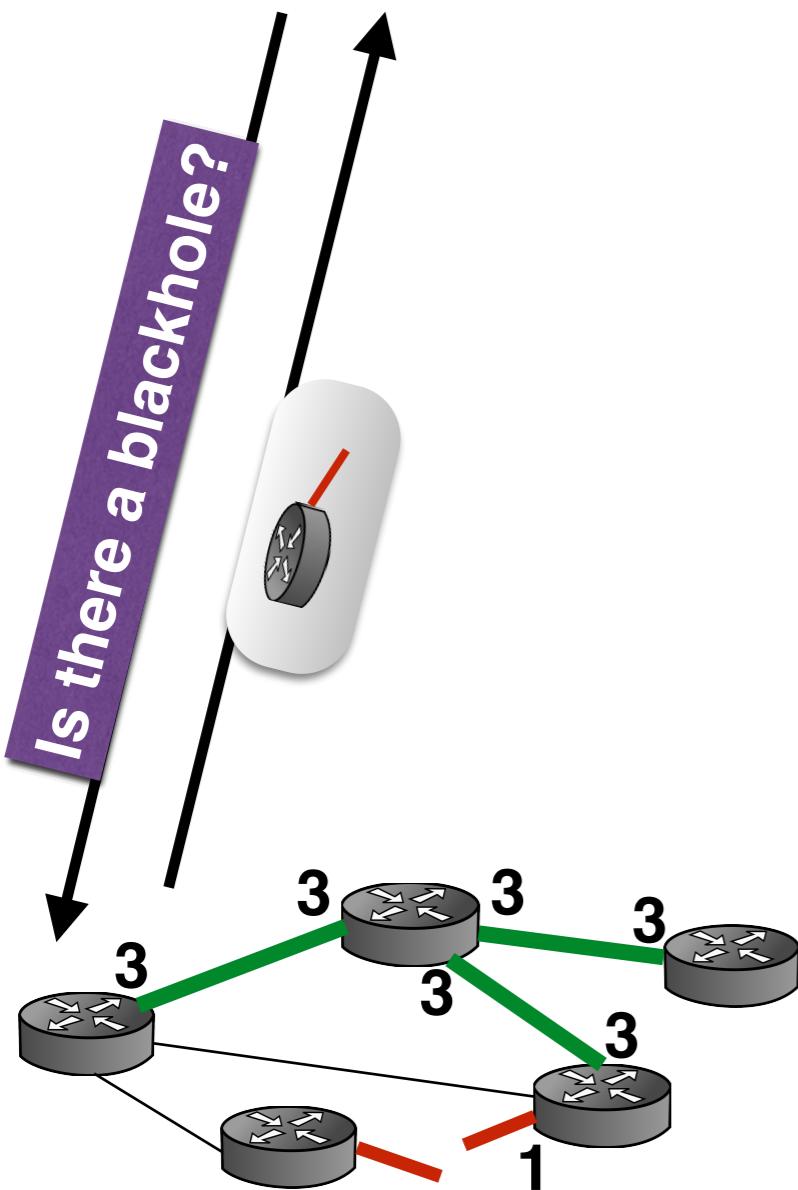
Blackhole detection with *SmartCounters*

- * Install *SmartCounter* for each port
- * Only two DFS traversals required:

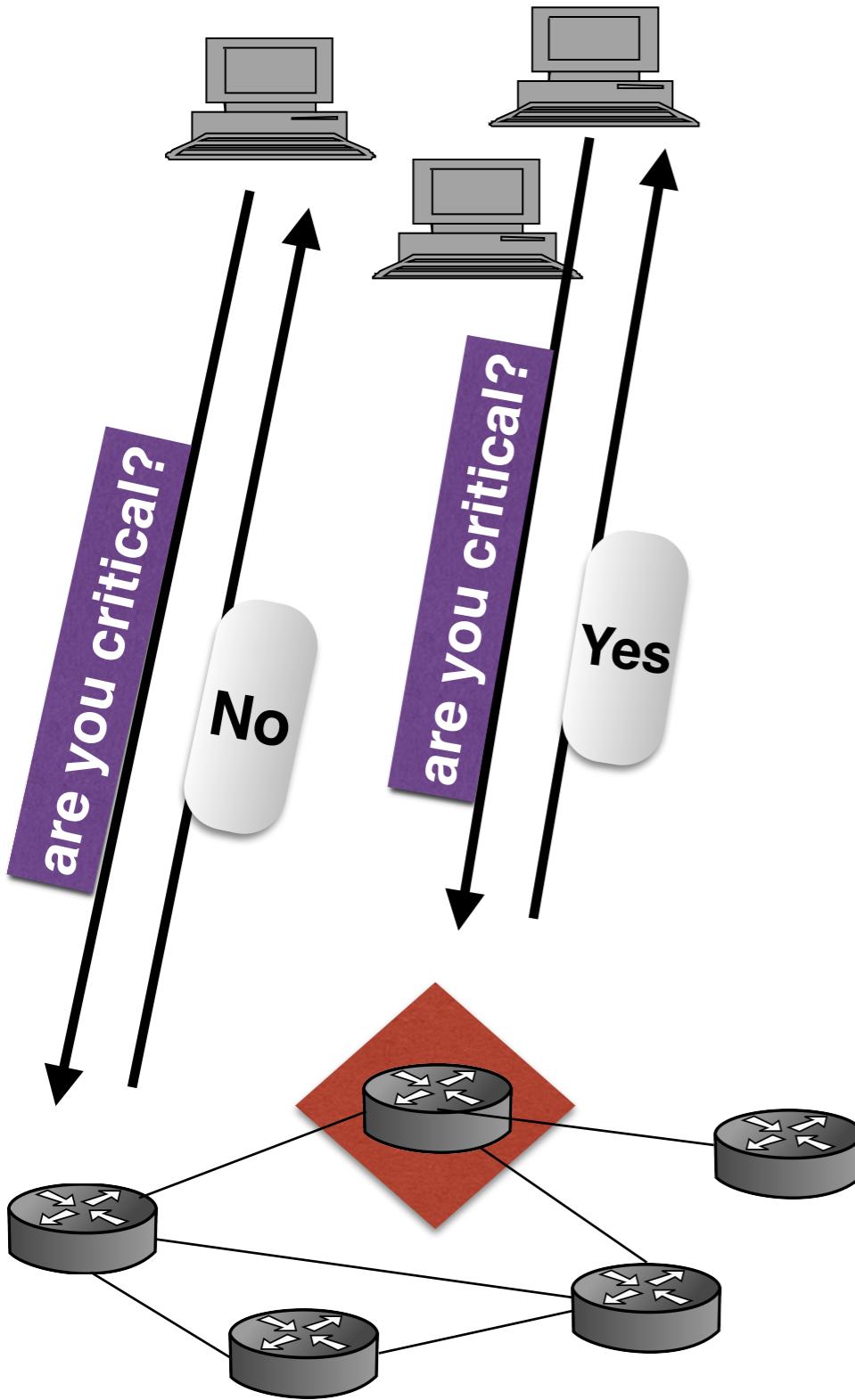
- * First - back&forth on each link



- * Second - find port with counter value 1

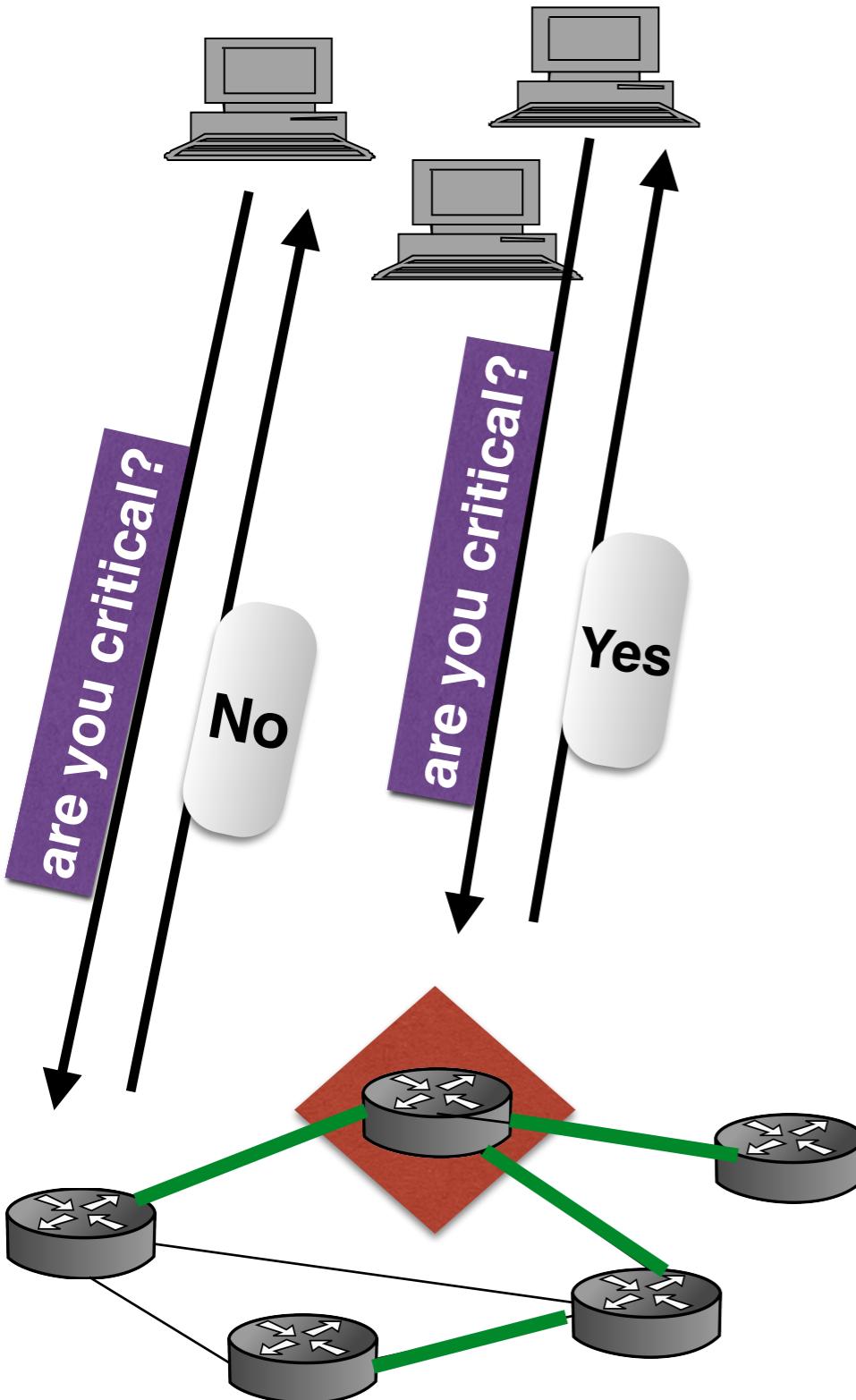


Functions in the South: *Critical Node Detection*



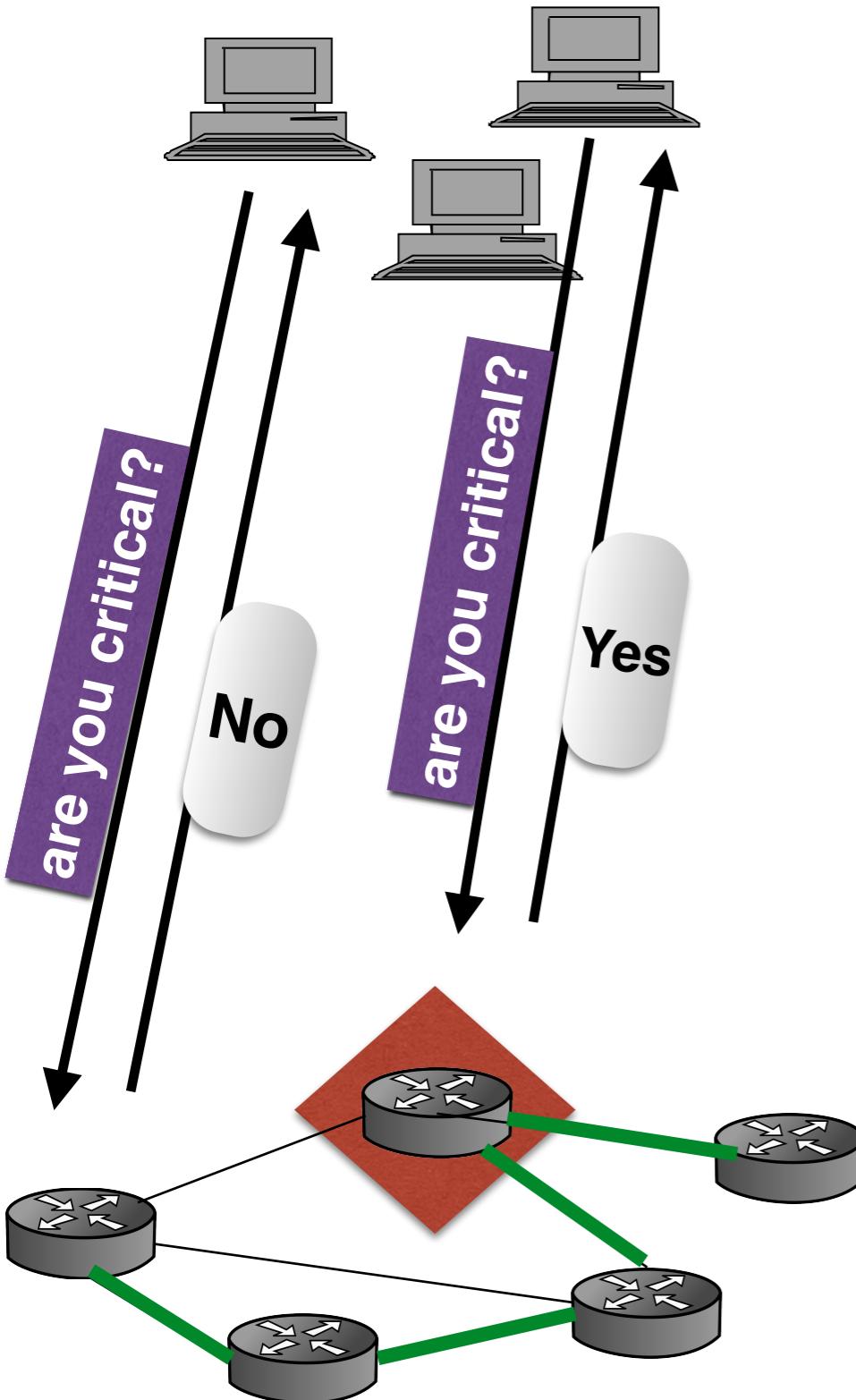
- * Checks if a node is critical for connectivity
- * Non-critical node may be removed for
 - * maintenance
 - * energy conservation
- * Cheaper than *Snapshot*

Functions in the South: *Critical Node Detection*



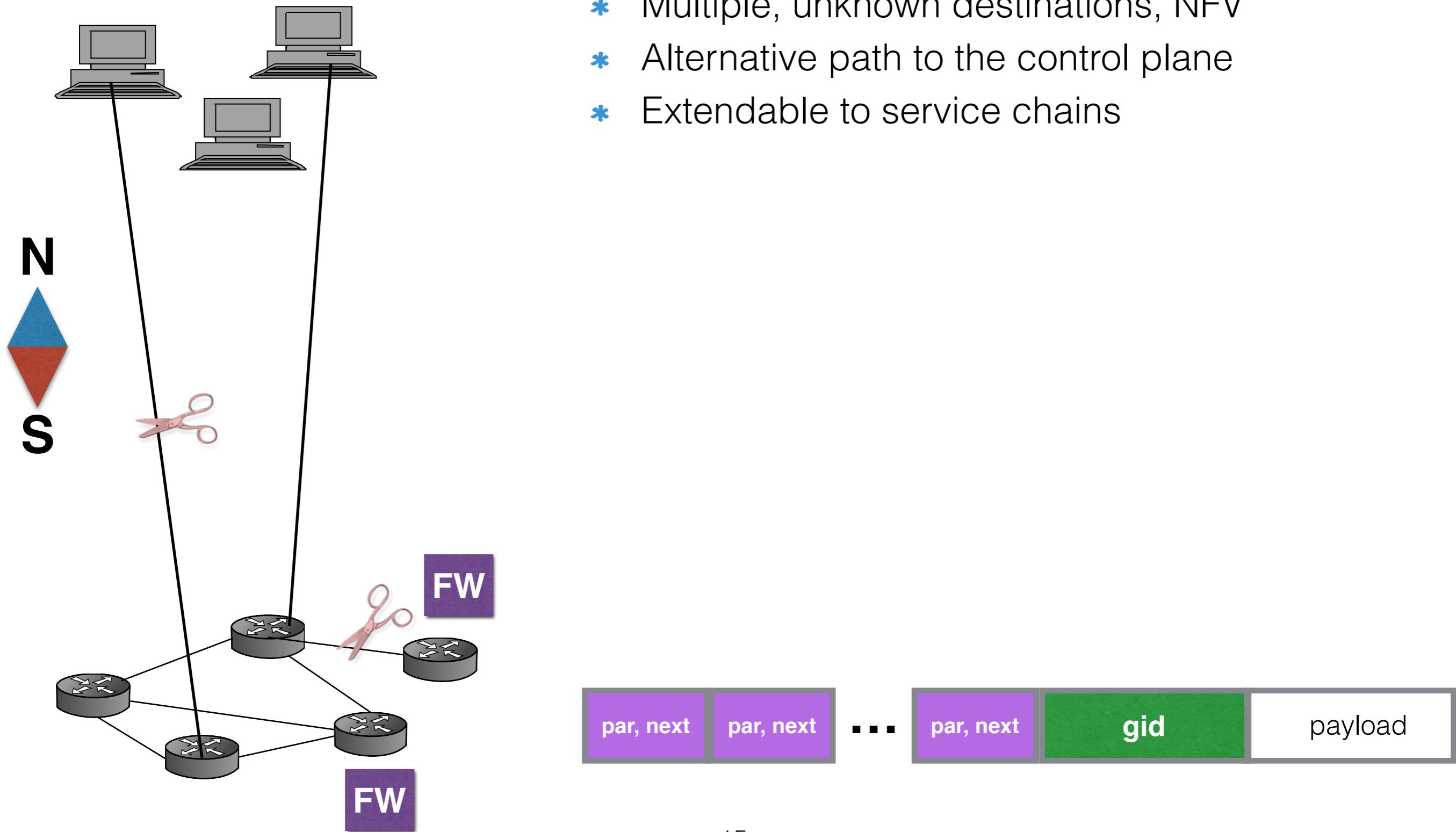
- * Checks if a node is critical for connectivity
 - * Non-critical node may be removed for
 - * maintenance
 - * energy conservation
 - * Cheaper than *Snapshot*
-
- * One DFS traversal with $\text{root} = v$
 - * **If** v non-critical: it is parent for exactly one node

Functions in the South: *Critical Node Detection*

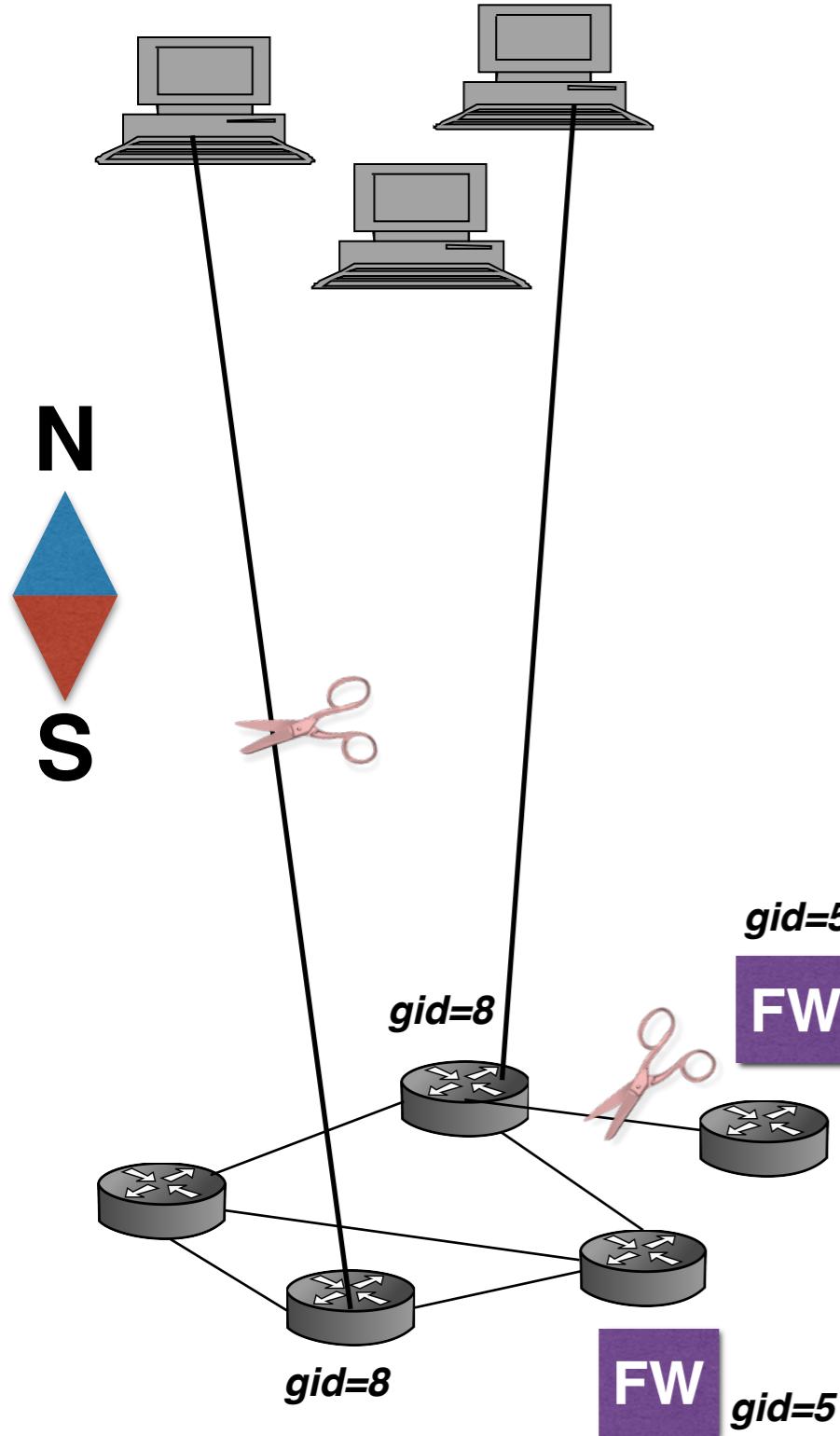


- * Checks if a node is critical for connectivity
- * Non-critical node may be removed for
 - * maintenance
 - * energy conservation
- * Cheaper than *Snapshot*
- * One DFS traversal with $\text{root} = v$
 - * **If** v non-critical: it is parent for exactly one node
 - * **Else**: it is parent for more than 1 node

Functions in the South: *Anycast*



Functions in the South: *Anycast*

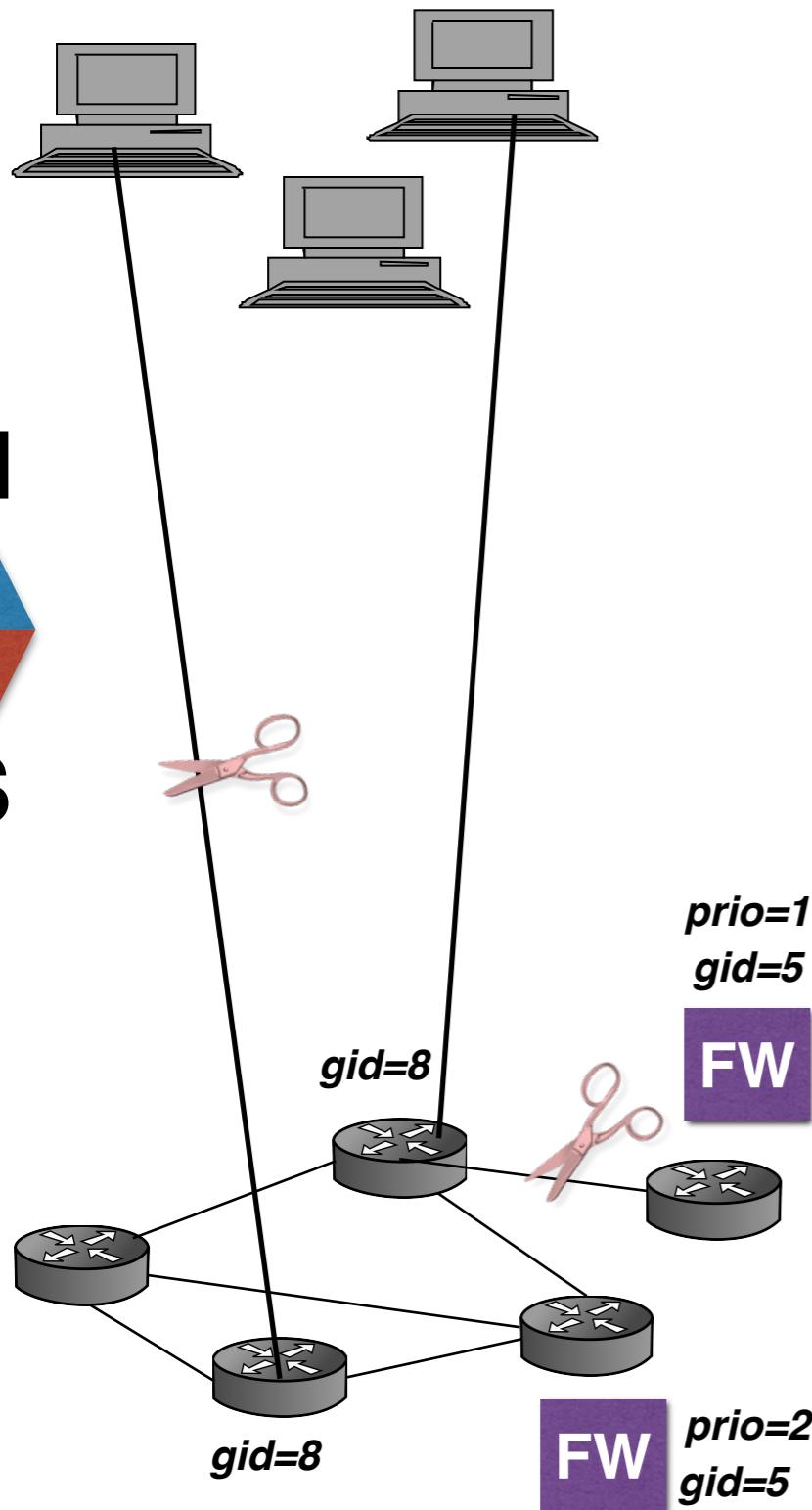


- * Multiple, unknown destinations, NFV
 - * Alternative path to the control plane
 - * Extendable to service chains
-
- * **Anycast** - one DFS traversal
 - * If ***gid*** match: forward to “self port”
 - * Else: continue DFS traversal



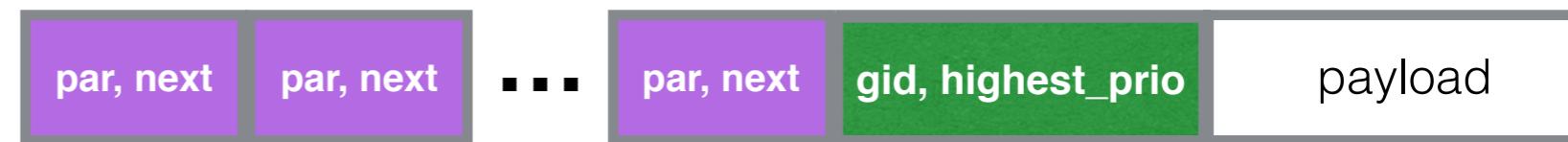
Functions in the South: *Anycast*

N
S



- * Multiple, unknown destinations, NFV
 - * Alternative path to the control plane
 - * Extendable to service chains
-
- * **Anycast** - one DFS traversal
 - * If **gid** match: forward to “self port”
 - * Else: continue DFS traversal

 - * **Priocast** - two DFS traversals
 - * First - find highest priority dest
 - * Second - deliver to the best dest

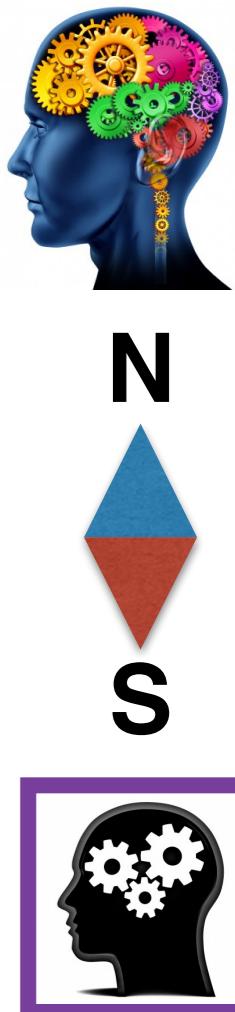


SmartSouth in practice

- * Using existing OpenFlow match fields for tagging:
 - * IPv6 addresses
 - * VLAN tags
 - * MPLS tags
 - * Using this approach we can support up to few dozens of nodes
- * More tag space in the future
 - * Future OpenFlow will probably support flexible match and set
 - * NoviFlow switches already support UDP payload access

Conclusions

- * “Dumb” data plane can implement useful and complex functionality
 - * Snapshot
 - * Blackhole detection
 - * Critical node detection
 - * Anycast
 - * Smart Counters
- * Nourish discussion on what should be implemented and where



Conclusions

- * “Dumb” data plane can implement useful and complex functionality
 - * Snapshot
 - * Blackhole detection
 - * Critical node detection
 - * Anycast
 - * Smart Counters
- * Nourish discussion on what should be implemented and where



N



S



Thank You!