06

Secure Routing and Forwarding

Kai Bu
kaibu@zju.edu.cn
http://list.zju.edu.cn/kaibu/netsec2020

Routing

Forwarding

select a path for traffic in a network

Routing

Forwarding

select a path for traffic in a network

Routing

Forwarding

relay packets along a certain path

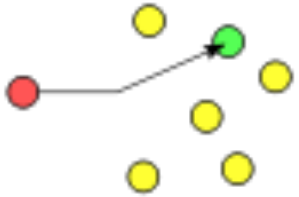
Secure Routing

How routing works?

How routing is attacked?

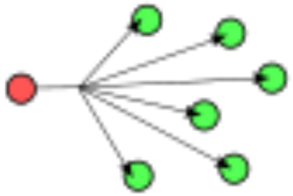
How routing is secured?

Delivery Scheme



unicast

deliver a message to a single specific node



broadcast

deliver a message to all nodes in the network

multicast

deliver a message to a group of nodes

anycast

deliver a message to any one out of a group

Delivery Scheme

unicast

deliver a message to a single specific node

broadcast

deliver a message to all nodes in the network

multicast

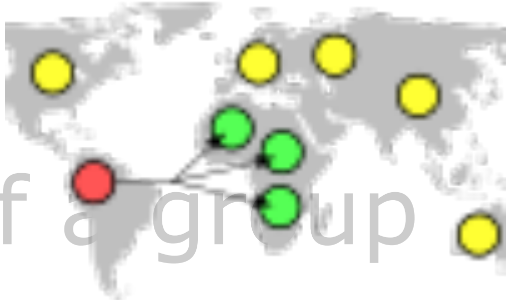
deliver a message to a group of nodes

anycast

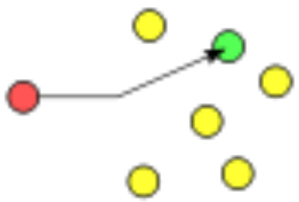
deliver a message to any one out of a group

geocast

deliver a message to a group of nodes based on geographic location



Delivery Scheme

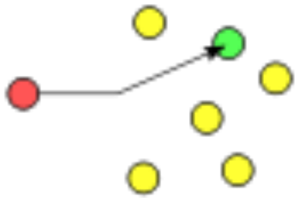


unicast

deliver a message to a single specific node

dominant form of msg delivery on Internet

Routing Scheme



unicast

deliver a message to a single specific node

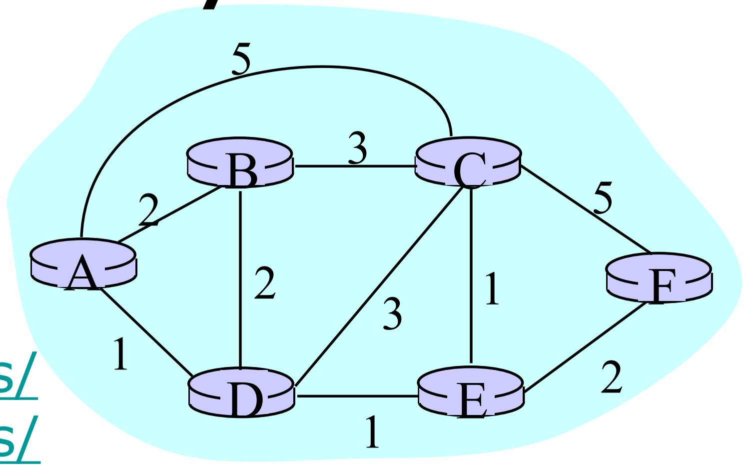
how to find a feasible path?

Routing Scheme

- Intra-domain routing
  inside an autonomous system
- Inter-domain routing
  between autonomous systems

Routing Scheme

- Intra-domain routing
  consider A-F as routers
- Inter-domain routing
  consider A-F as autonomous systems



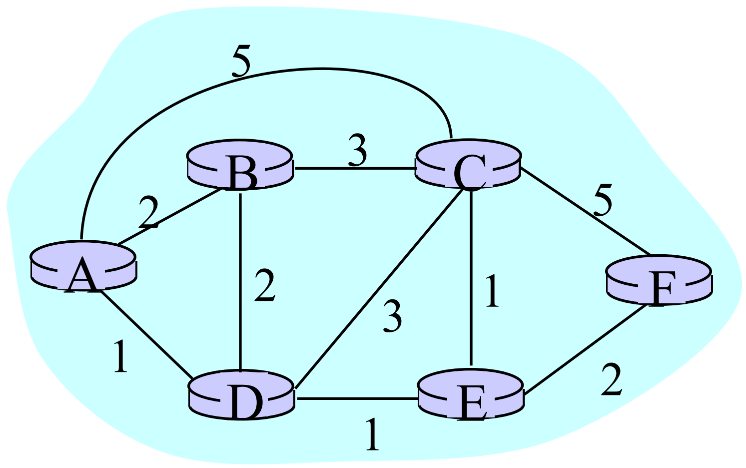
examples from
https://www.cs.umd.edu/~shankar/417-F01/Slides/chapter4a-aus/
https://www.cs.umd.edu/~shankar/417-F01/Slides/chapter4b-aus/

Route Computation

- Link-state algorithms

  each router knows complete topology & link cost information;

  independently run routing algorithm to calculate shortest path to each destination;



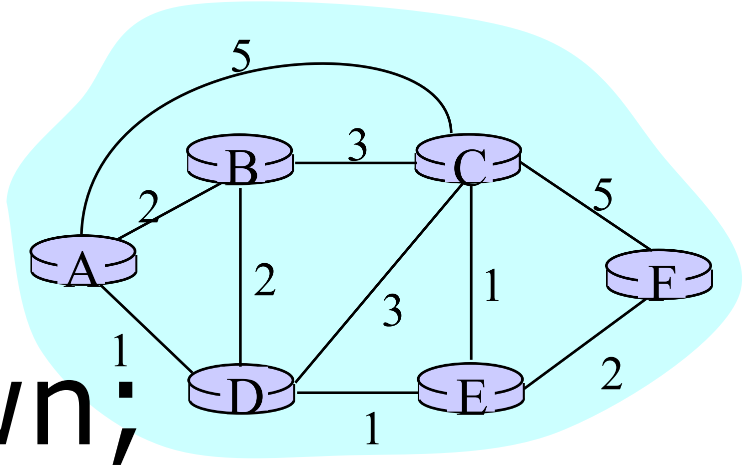
Dijkstra

c(i,j) link cost from i to j (∞ if unknown)

D(v) current value of cost of path from source to destination v;

p(v) predecessor node along path from source to v;

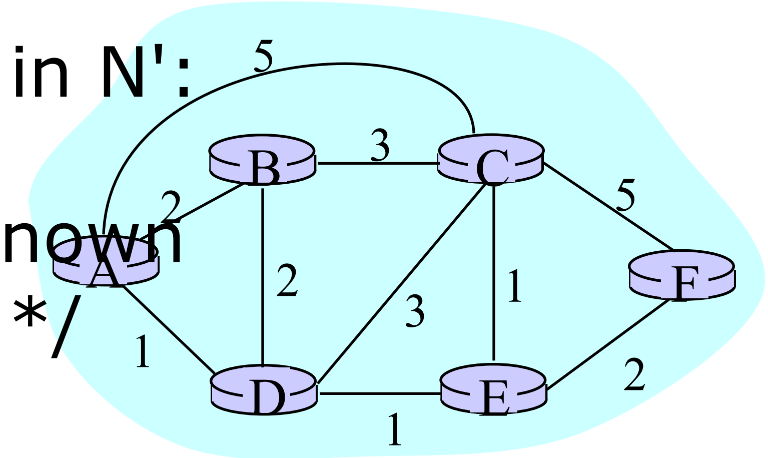
N’ set of nodes whose least cost path is already known;



Dijkstra

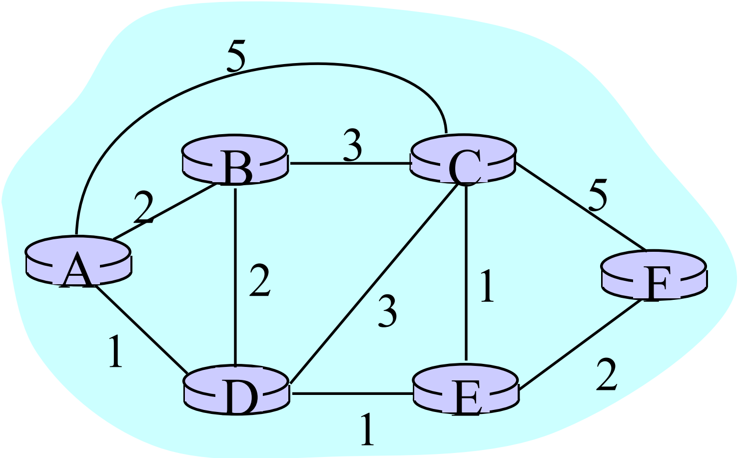
```
Initialization:
 N' = {A}
 for all nodes v
   if v adjacent to A
     then D(v) = c(A,v)
     else D(v) = ∞

 Loop
 find w not in N' such that D(w) is
   minimum
 add w to N'
 update D(v) for all v adjacent to w and not in N':
     D(v) = min(D(v), D(w) + c(w,v))
  /* new cost to v is either the old cost, or known
 shortest path cost to w plus cost from w to v */
 until all nodes in N'
```



Dijkstra

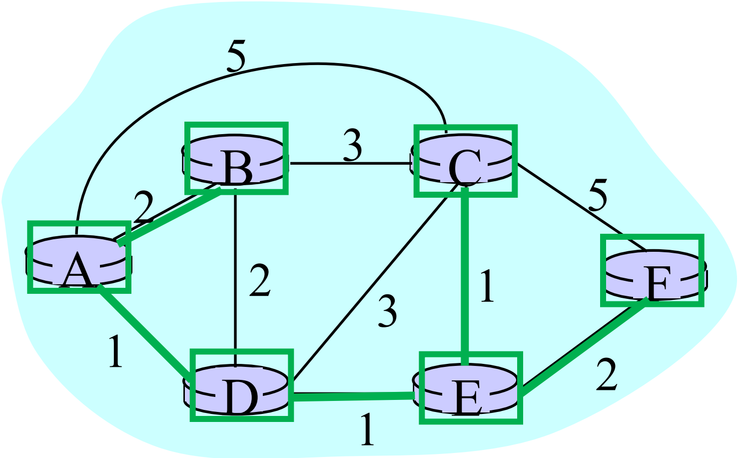
Step	start N'	D(B),p(B)	D(C),p(C)	D(D),p(D)	D(E),p(E)	D(F),p(F)
0	A	2,A	5,A	1,A	infinity	infinity
1	AD	2,A	4,D		2,D	infinity
2	ADE	2,A	3,E			4,E
3	ADEB		3,E			4,E
4	ADEBC					4,E
5	ADEBCF					



Dijkstra

Step	start N'	D(B),p(B)	D(C),p(C)	D(D),p(D)	D(E),p(E)	D(F),p(F)
0	A	2,A	5,A	1,A	infinity	infinity
1	AD	2,A	4,D		2,D	infinity
2	ADE	2,A	3,E			4,E
3	ADEB		3,E			4,E
4	ADEBC					4,E
5	ADEBCF					

resulting shortest-path tree for A:

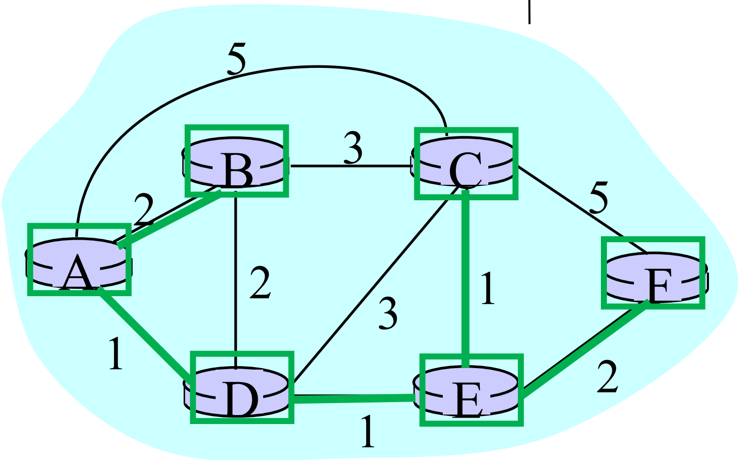


Dijkstra

Step	start N'	D(B),p(B)	D(C),p(C)	D(D),p(D)	D(E),p(E)	D(F),p(F)
0	A	2,A	5,A	1,A	infinity	infinity
1	AD	2,A	4,D		2,D	infinity
2	ADE	2,A	3,E			4,E
3	ADEB		3,E			4,E
4	ADEBC					4,E
5	ADEBCF					

resulting forwarding table at A:

destination	link
B	(A, B)
D	(A, D)
E	(A, D)
C	(A, D)
F	(A, D)



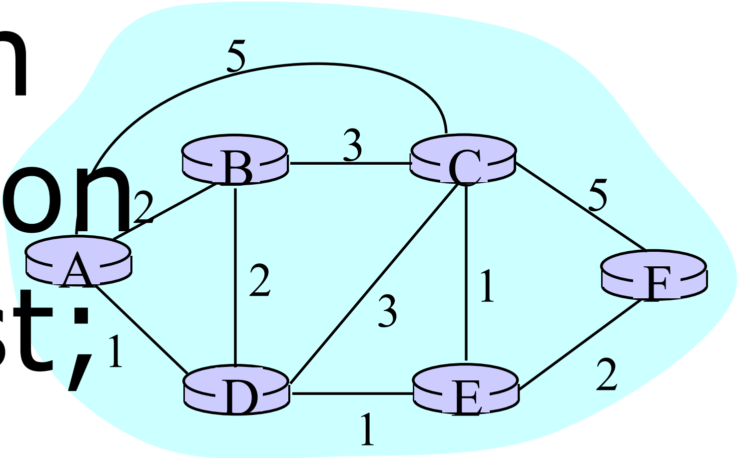
what if no global view?

Route Computation

- Distance-vector algorithms

  each router knows direct neighbors & link costs to neighbors;

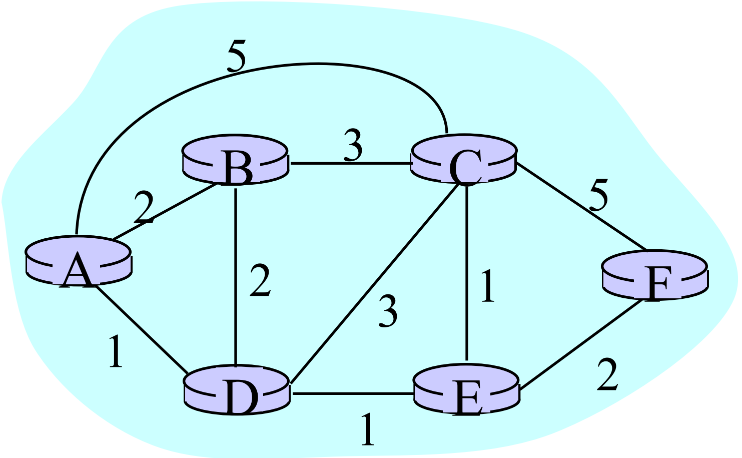
  independently calculate shortest path to each destination through an iterative process based on neighbors' distances to dest;



Bellman-Ford

$D_x(y)$ cost of least-cost path from x to y:

$D_x(y) = \min\{c(x,v) + D_v(y)\}$
for all neighbors v of x



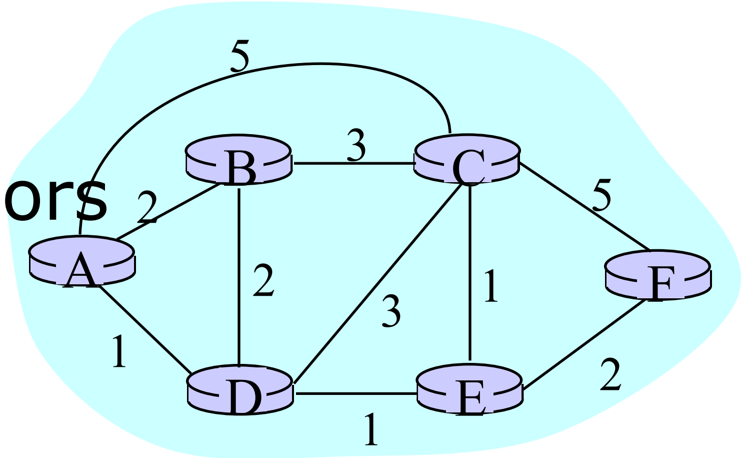
Bellman-Ford

$D_x(y)$ cost of least-cost path from x to y:

wait for (change in local link cost of msg from neighbor)

recompute estimates

if DV to any dest has changed, notify neighbors



Bellman-Ford

$D_x(y)$ cost of least-cost path from x to y:

$$D_x(y) = \min\{c(x,v) + D_v(y)\}$$

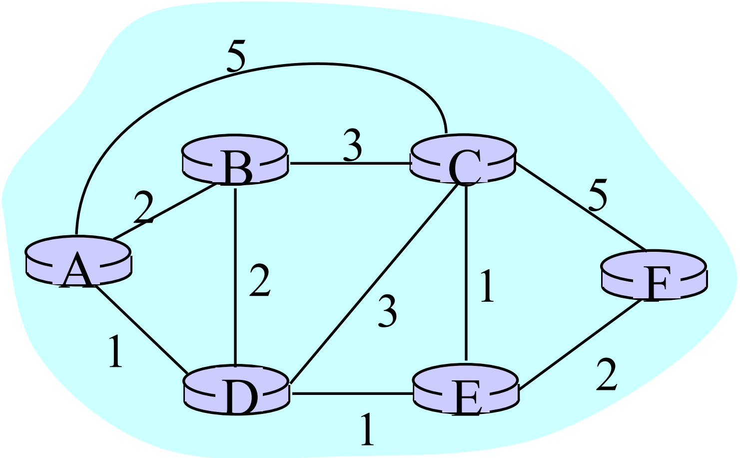
for all neighbors v of x

$$D_A(F) = \min\{c(A,B) + D_B(F),\ c(A,D) + D_D(F),\ c(A,C) + D_C(F)\}$$

$$= \min\{2 + 5,\ 1 + 3,\ 5 + 3\} = 4$$

node leading to shortest path is next hop

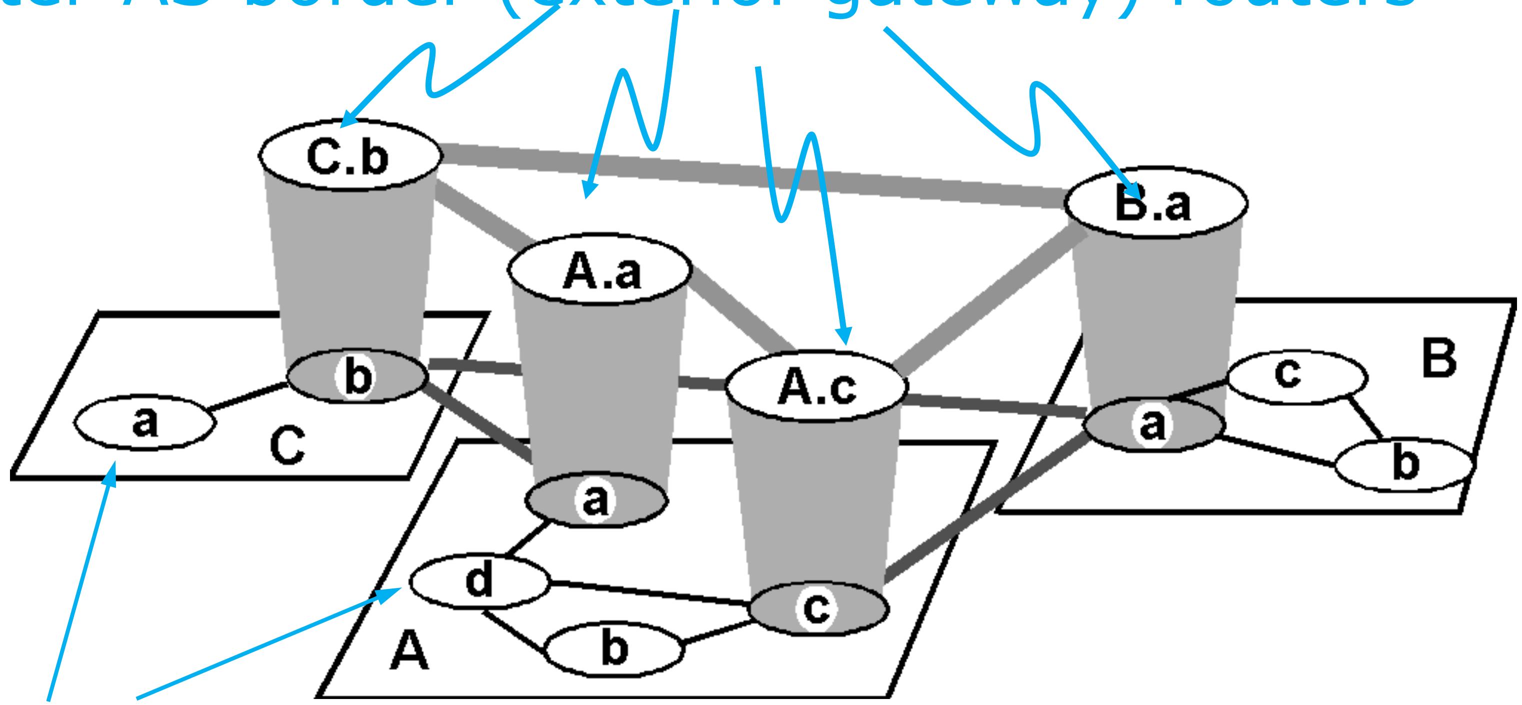
→ forwarding table



intra-domain vs inter-domain

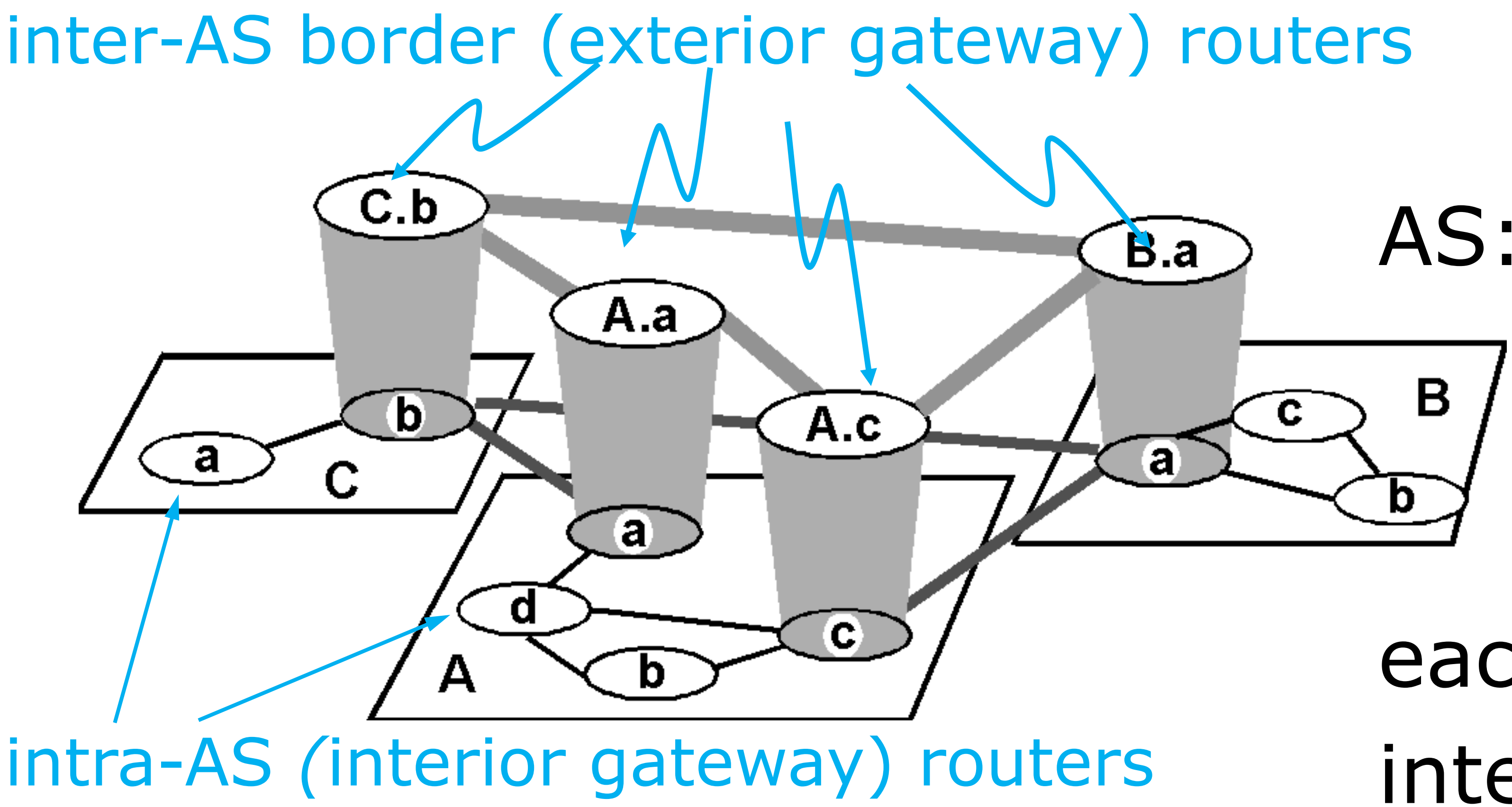
Hierarchical Routing

inter-AS border (exterior gateway) routers



intra-AS (interior gateway) routers

Hierarchical Routing



AS: autonomous system

each AS uses its own IGP internal routing protocol; border routers run BGP as well;

IGP: Interior Gateway Prot

- RIP
  routing information protocol
- OSPF
  open shortest path first

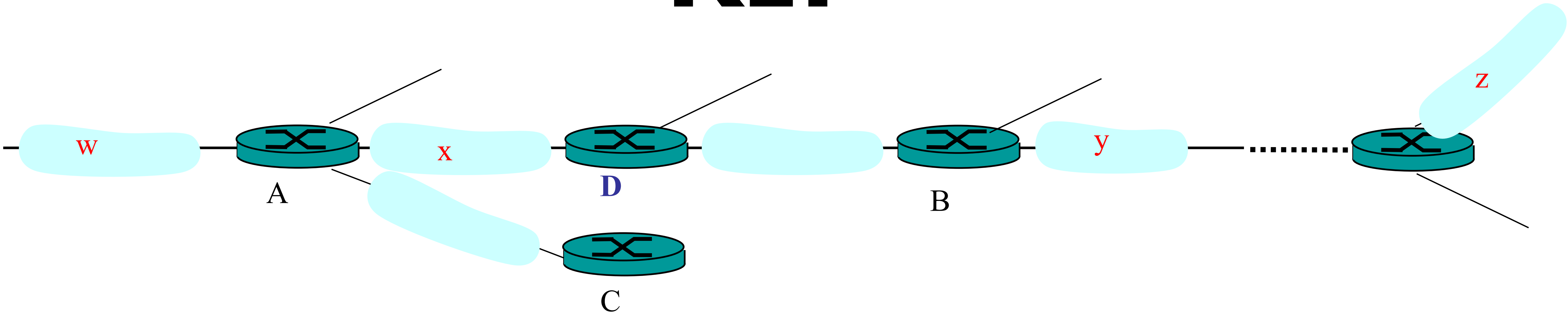
RIP

- Distance-vector algorithm

  distance metric: # of hops (max=15)
- Neighbor routers exchange routing advertisement every 30 seconds
- Failure and recovery

  if no update from neighbor N after 180s
  invalidate routes via N, notify neighbors

RIP



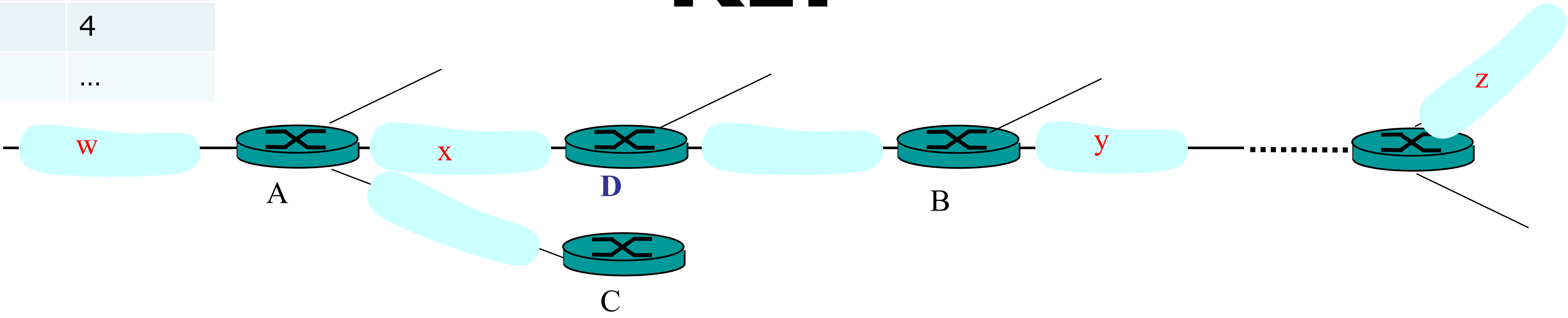
D: routing table

destination network	next router	# of hops to destination
w	A	2
y	B	2
z	B	7
x	--	1
…	…	…

dest	hops
w	1
x	1
z	4
…	…

advertisement from A to D

RIP



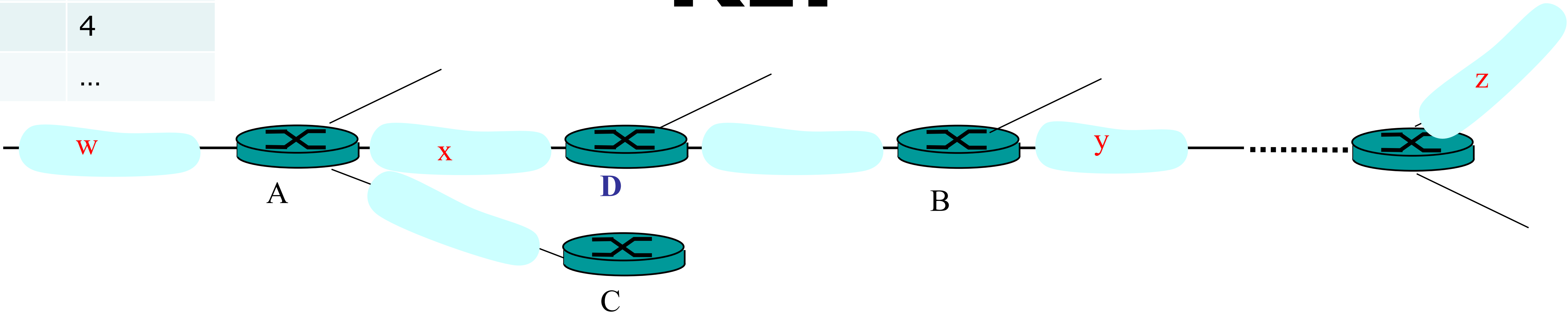
D: routing table

destination network	next router	# of hops to destination
w	A	2
y	B	2
z	B	7
x	--	1
…	…	…

dest	hops
w	1
x	1
z	4
…	…

advertisement from A to D

RIP



D: routing table

destination network	next router	# of hops to destination
w	A	2
y	B	2
z	B→A	7→5
x	--	1
…	…	…

OSPF

- Link-state algorithm

  each node knows its direct neighbors & the link distance to each(link-state);

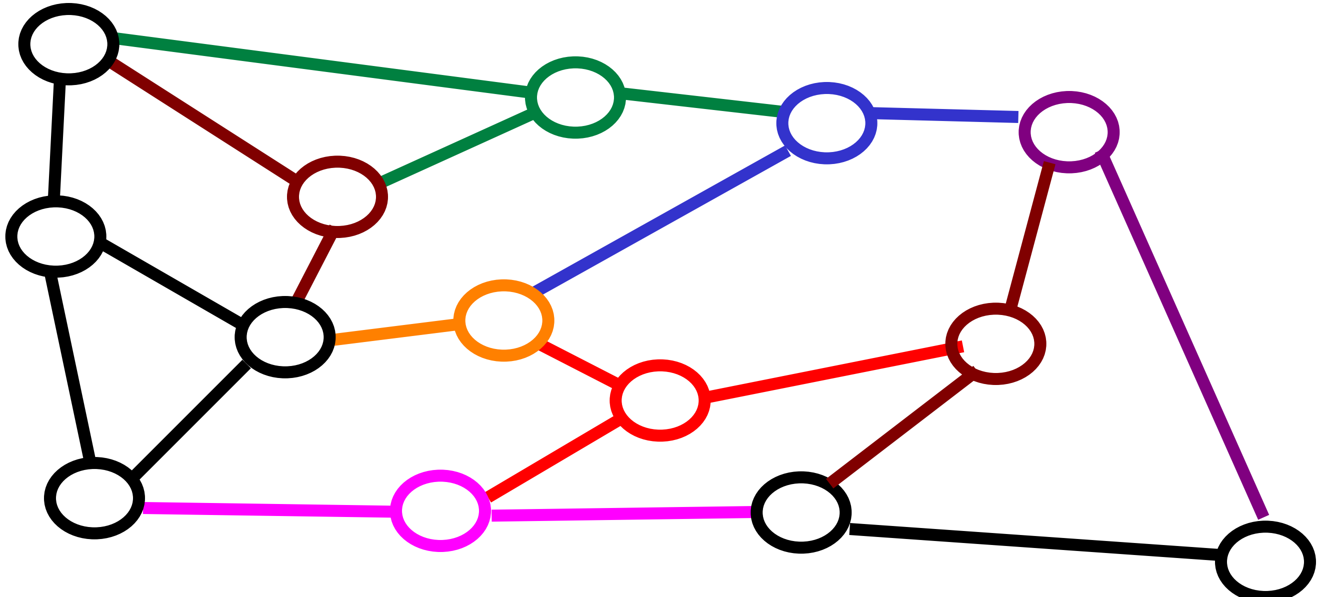
  each node periodically broadcasts its link-state to the entire network;

OSPF

- LSP (Link-State Packet)
  one entry per neighbor router:
  ID of the node that created the LSP;
  a list of direct neighbors, with link cost;
  sequence number for this LSP (SEQ);
  time-to-live (TTL) for info in this LSP;

OSPF

- Build a complete map using link states
  everyone broadcasts a piece of topology
  put all pieces together → complete map



OSPF

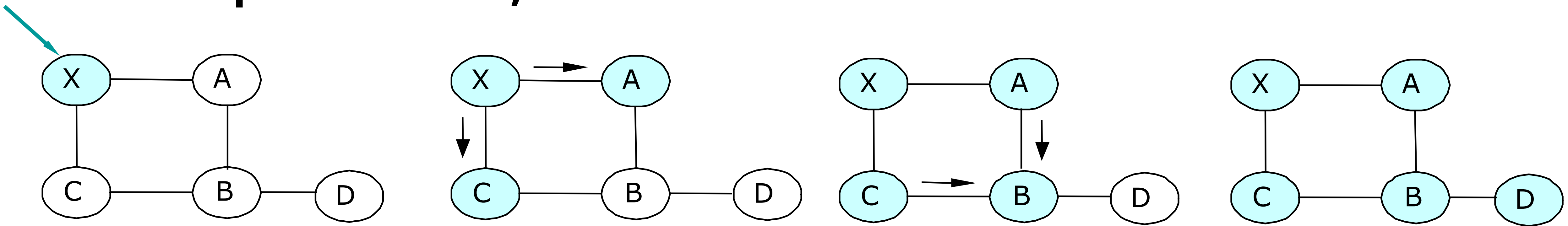
- Each node stores and forwards LSPs
- Decrement TTL of stored SLPs
- Discard info when TTL=0
- Compute routes using Dijkstra
- Generate LSPs periodically with increasing SEQ

OSPF

- Reliable flooding of LSP

  forward each received LSP to all neighbors but the one that sent it;

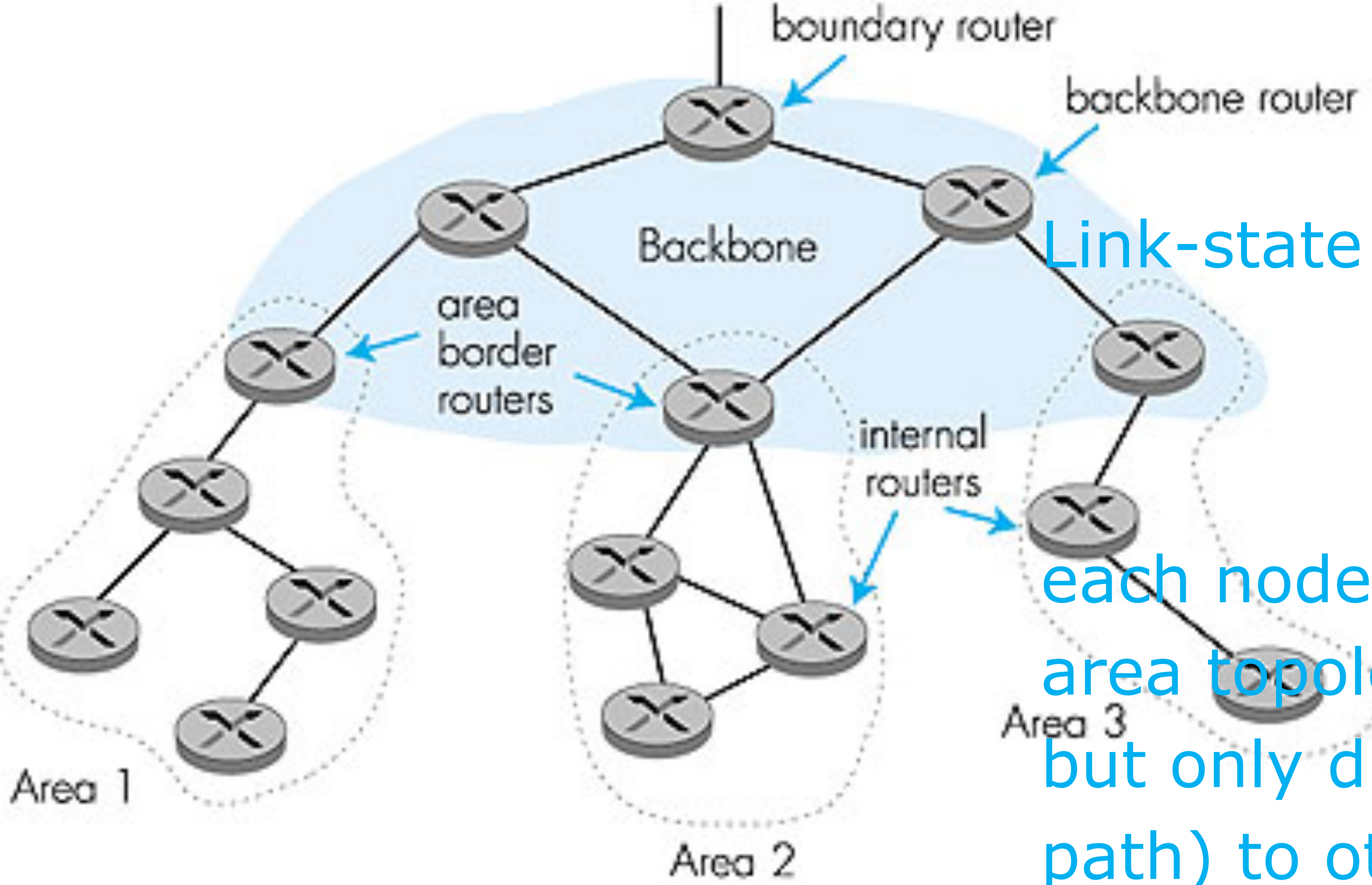
  use the source-ID and SEQ to detect duplicates;



OSPF

- All OSPF messages are authenticated
- Multiple same-cost paths are allowed
- Hierarchical OSPF is used in large dom

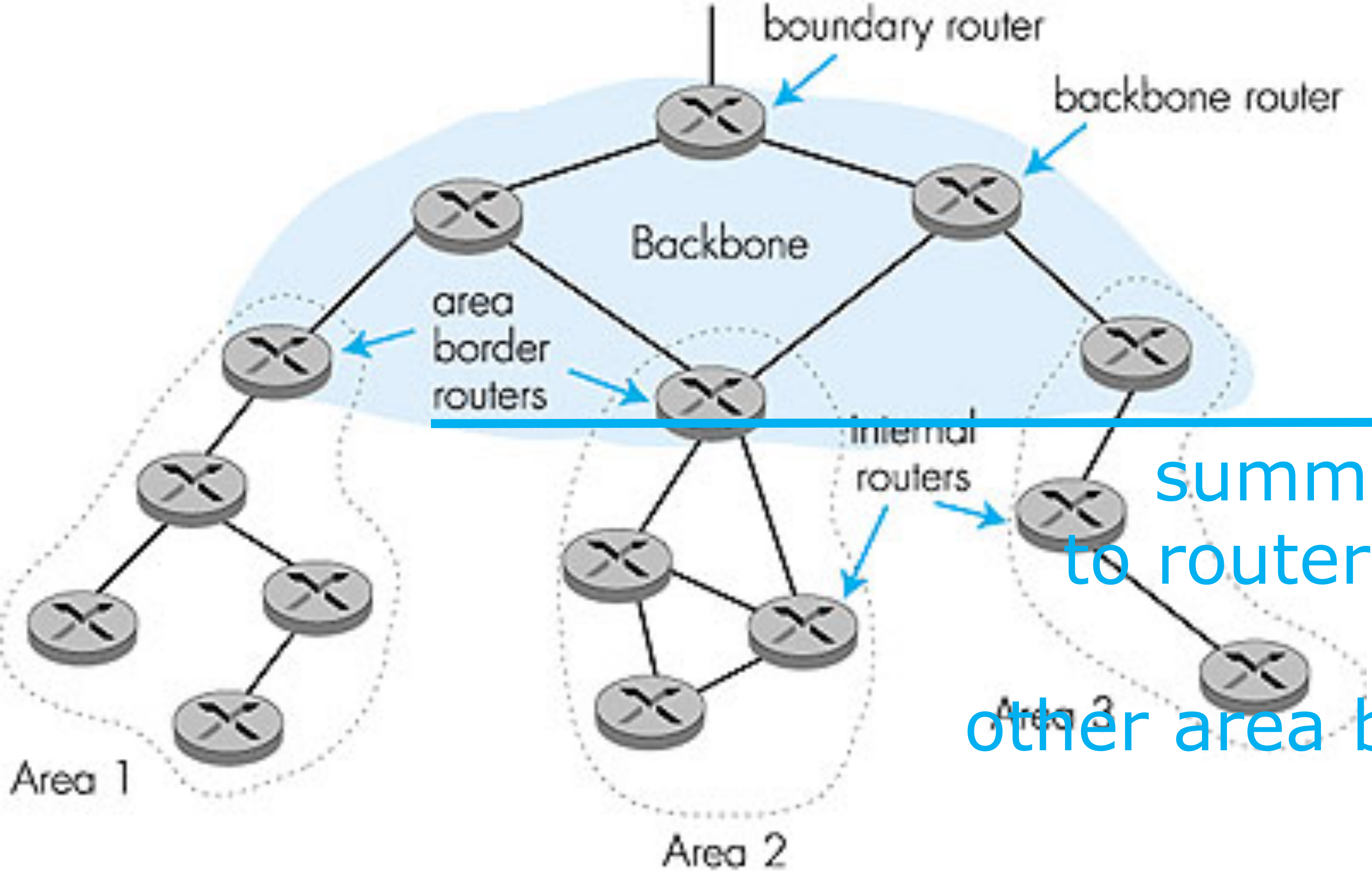
Hierarchical OSPF



Link-state ads only in area

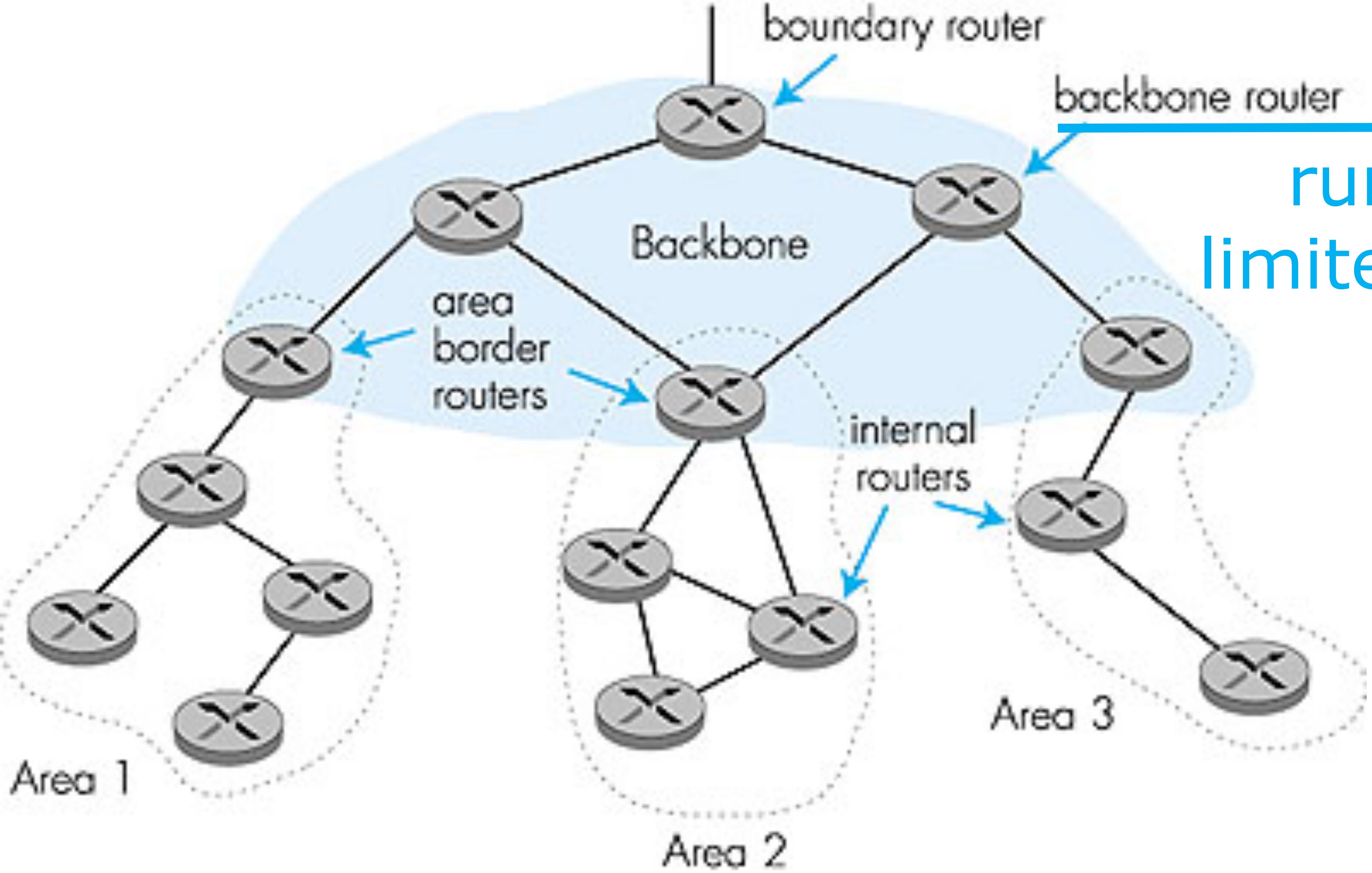
each node has detailed area topology, but only direction (shortest path) to other areas;

Hierarchical OSPF



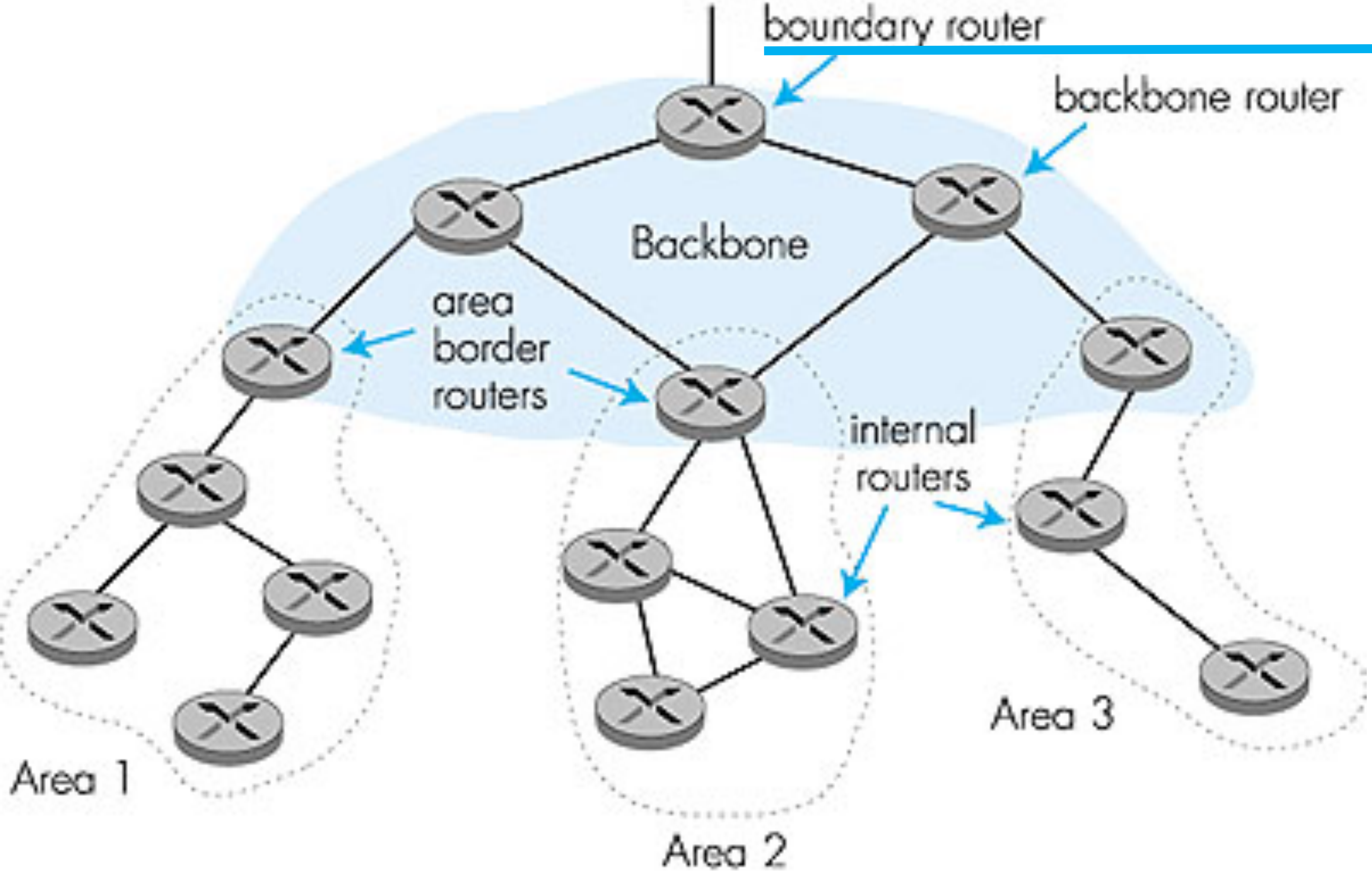
summarize distances to routers in local area; advertise to other area border routers;

Hierarchical OSPF



run OSPF routing
limited to backbone

Hierarchical OSPF

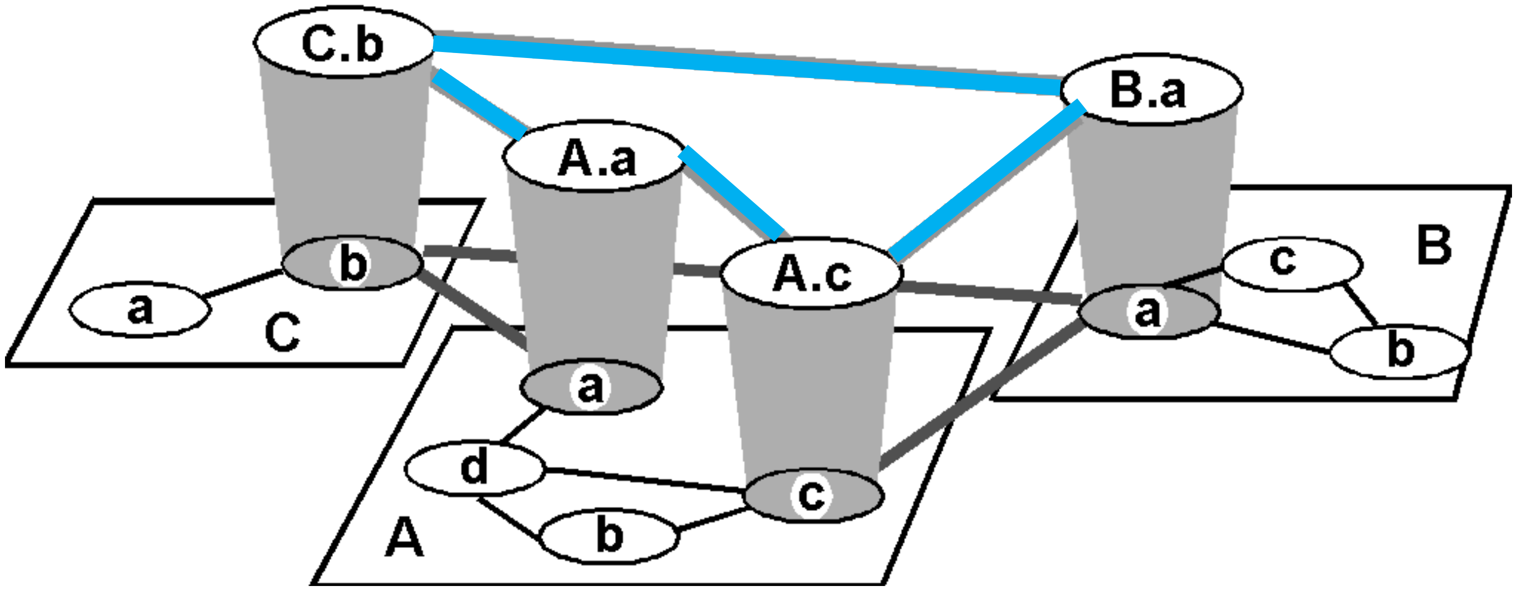


connect to other ASes

inter-domain routing

BGP: Border Gateway Protocol

BGP



- Path-vector protocol among border routers

  each border router broadcasts to neighbors entire path of AS sequence to destination:

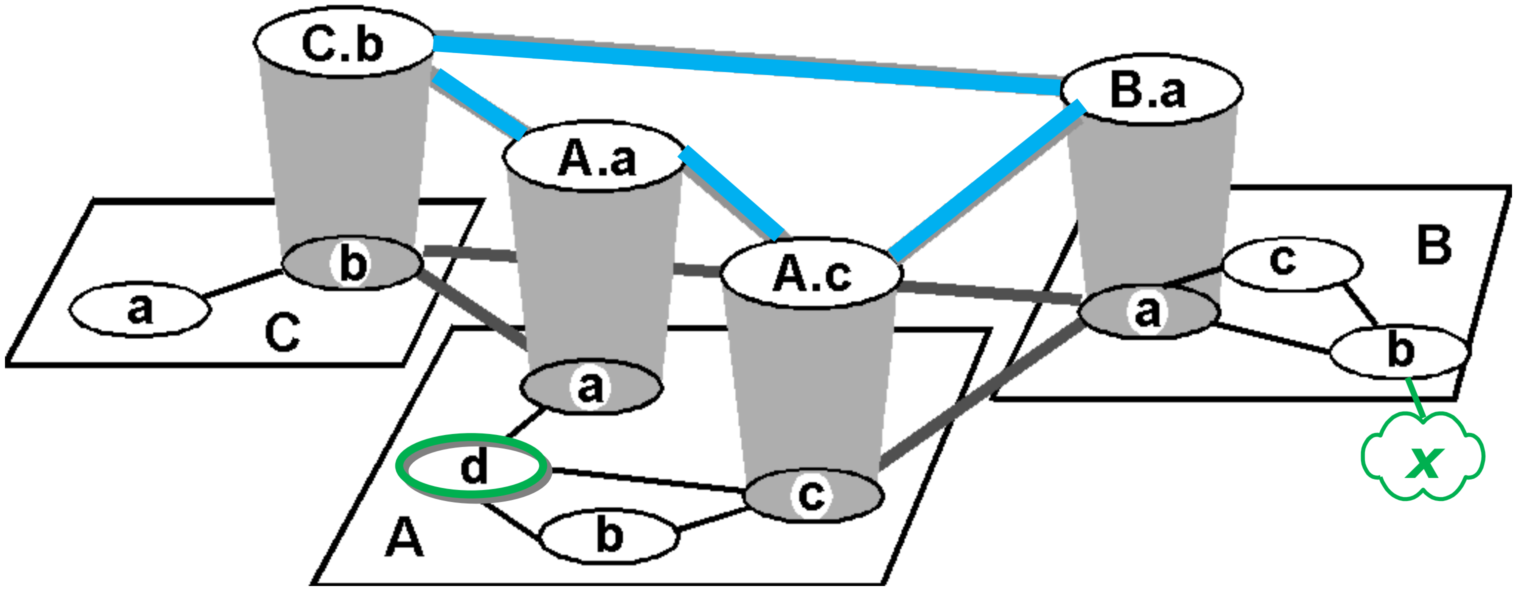
  e.g., Path(B,C) = B, A, C

BGP

For each AS:

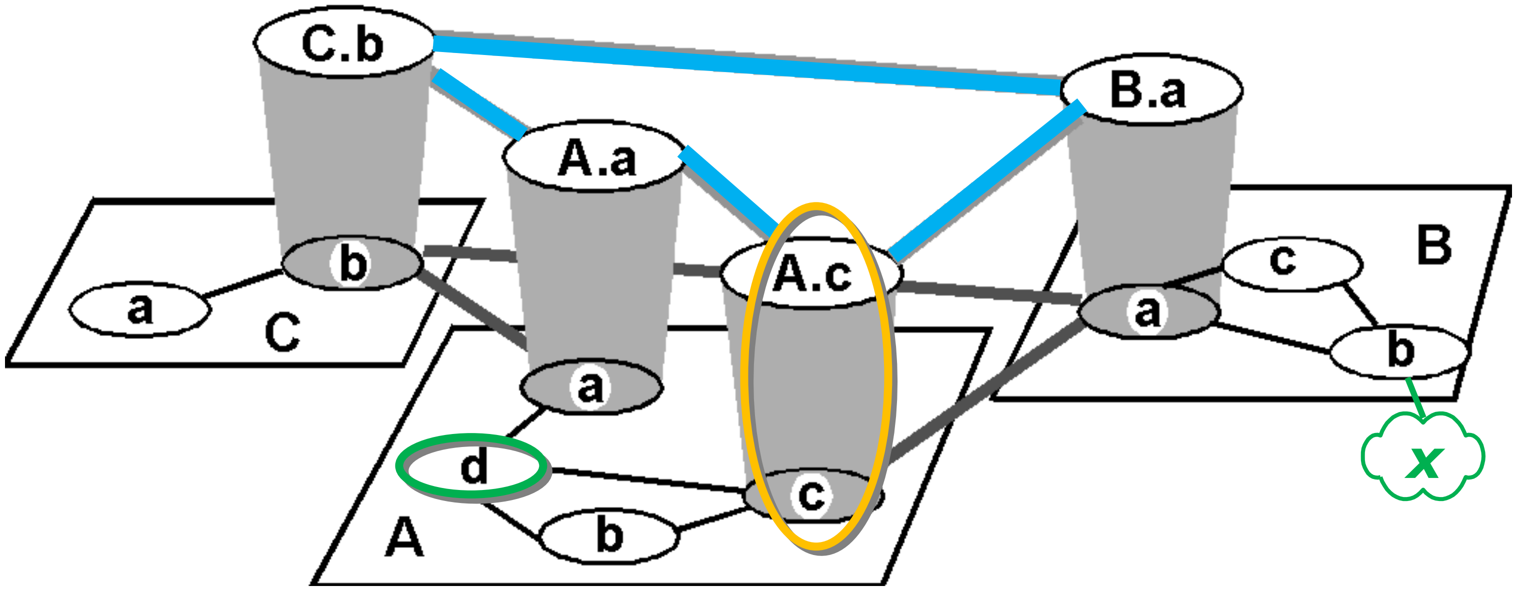
- Obtain subnet reachability information from neighbor ASes;
- Propagate the reachability information to all internal routers;
- Determine routes to subnets based on reachability information and policy

BGP



- Example: forwarding table entry for d→x

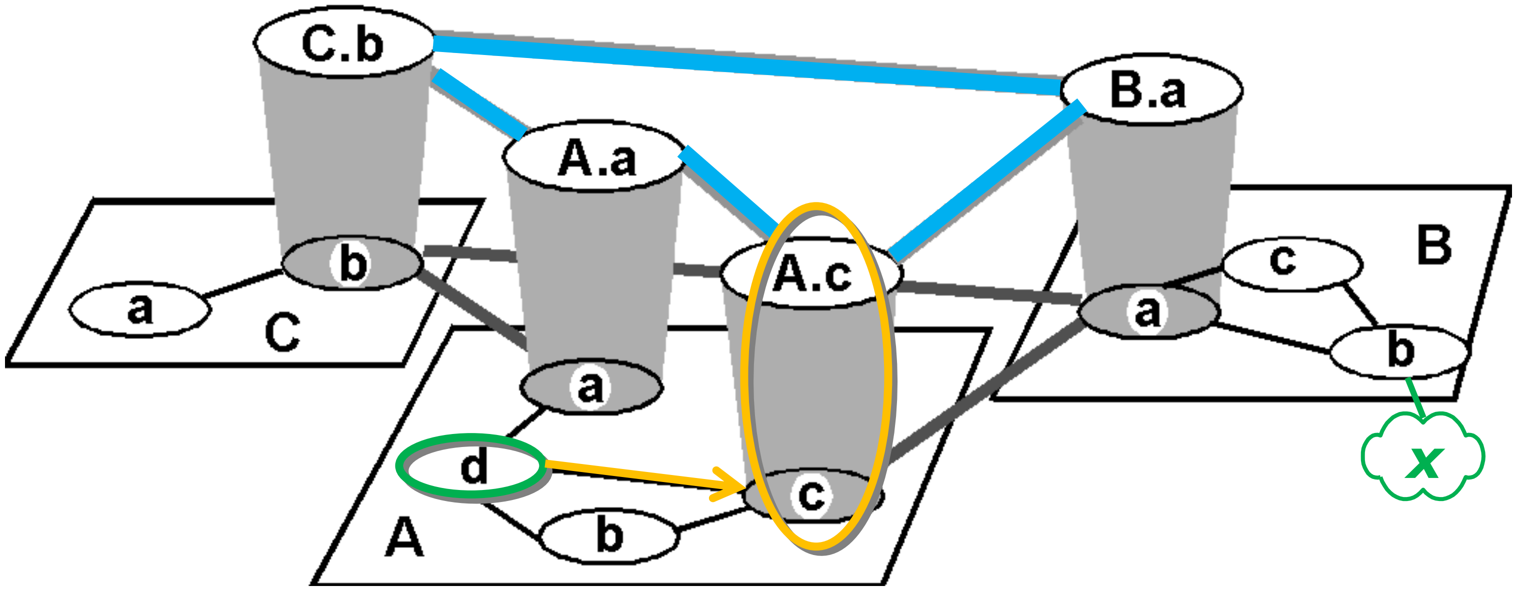
BGP



- Example: forwarding table entry for d→x

  AS A learns from BGP that subnet x is reachable from AS B via border router A.c;

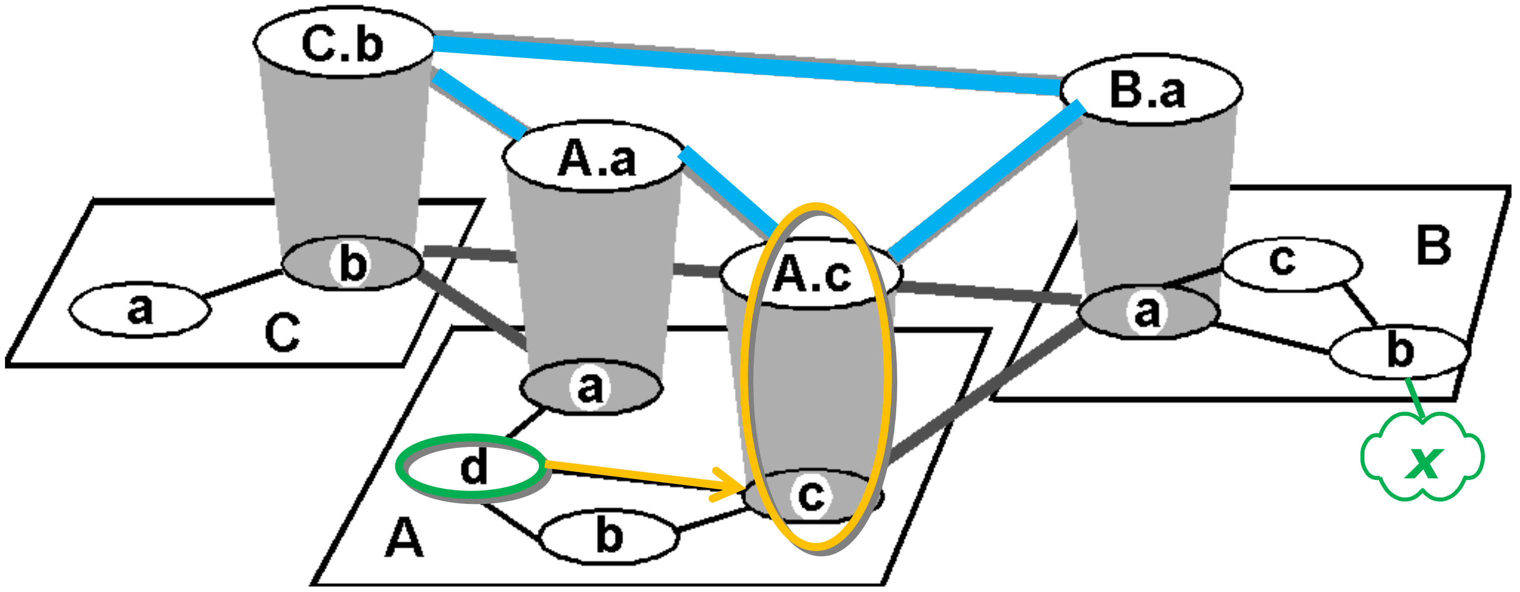
BGP



- Example: forwarding table entry for d→x

  router d determines from intra-domain routing info that its interface I is on the least cost path to c;

BGP



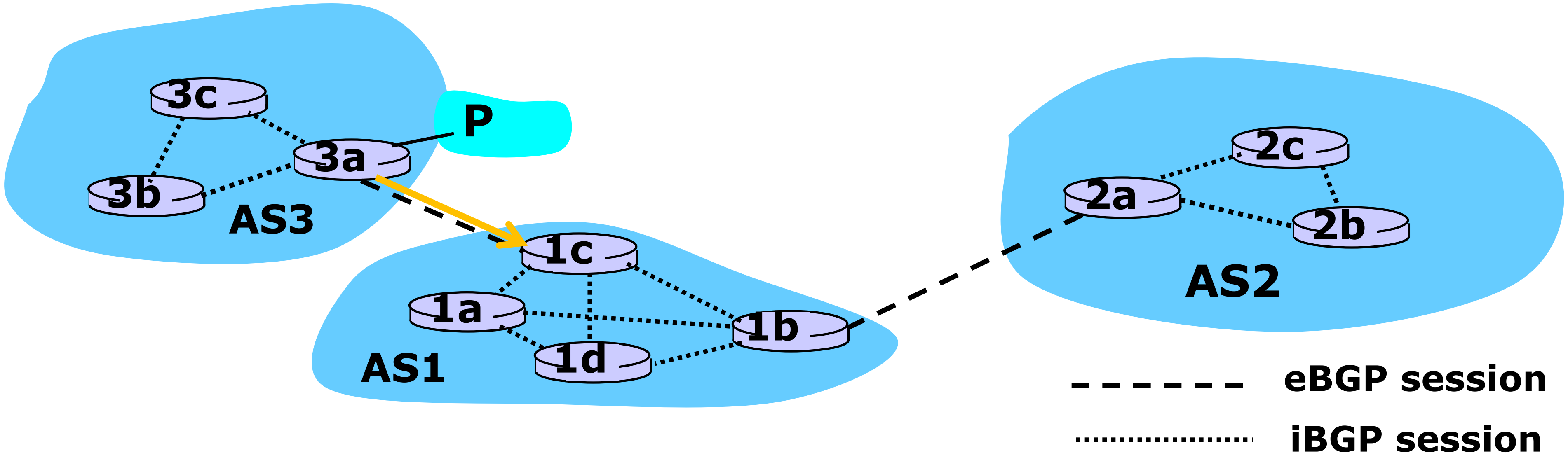
- Example: forwarding table entry for d→x

destination	next hop
x	I

BGP

Distribute reachability information:

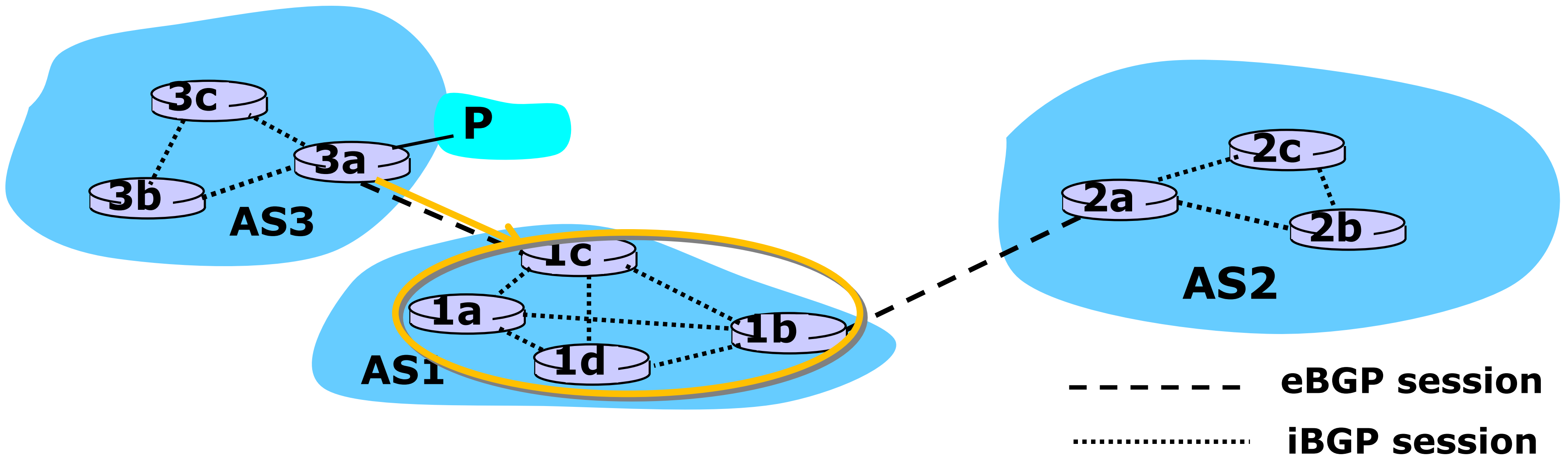
- with eBGP session 3a-to-1c,
  AS3 sends prefix reachability info to AS1



BGP

Distribute reachability information:

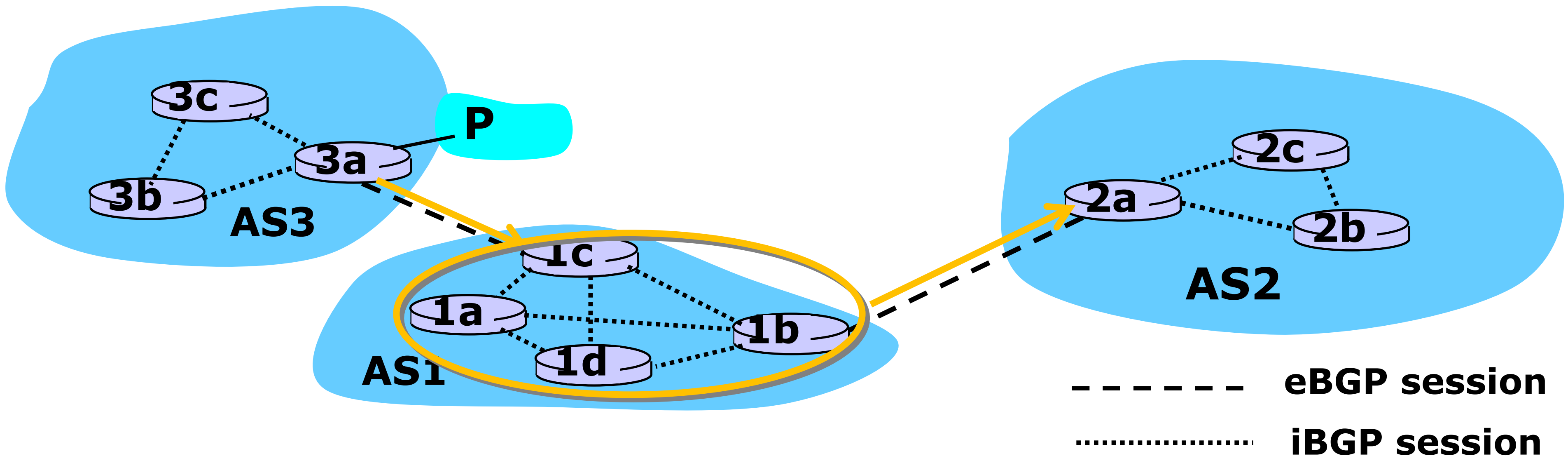
- 1c uses iBGP sessions to distribute this new prefix reachability info to all routers in AS1;



BGP

Distribute reachability information:

- 1b re-advertises the new reachability info to AS2 over the 1b-to-2a eBGP session;

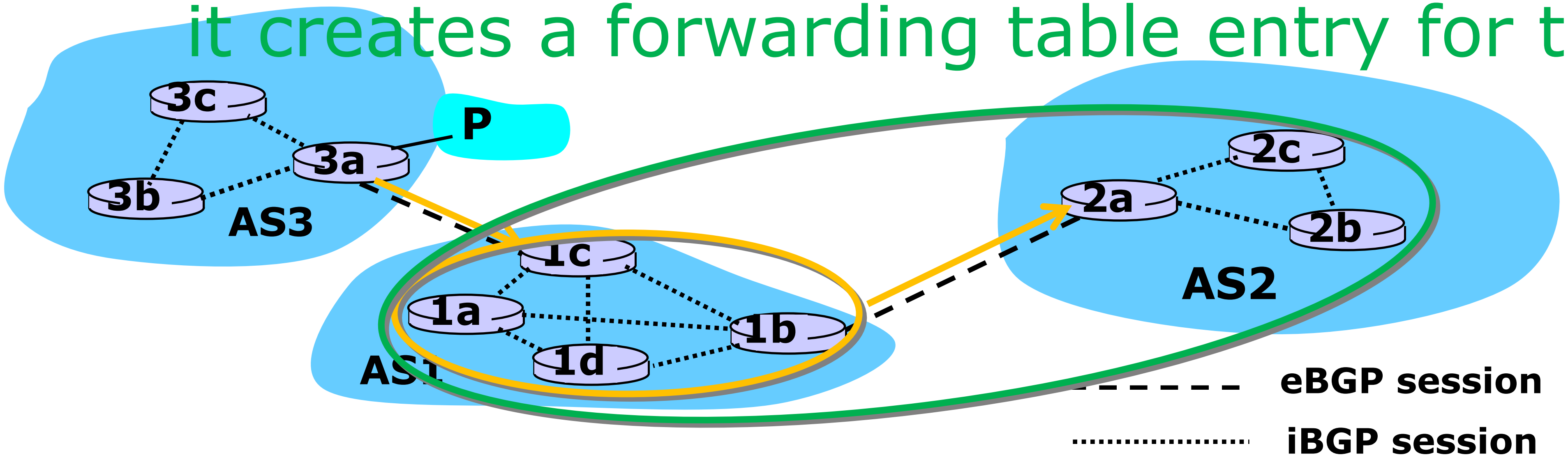


BGP

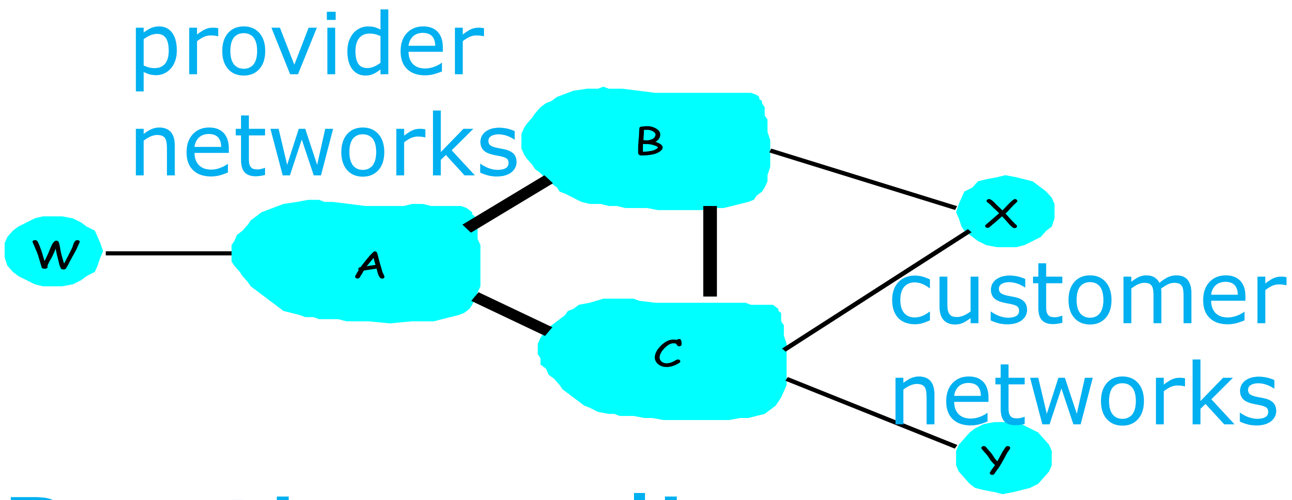
Distribute reachability information:

- 1b re-advertises the new reachability info to AS2 over the 1b-to-2a eBGP session;

when a router learns about a new prefix,
it creates a forwarding table entry for the prefix



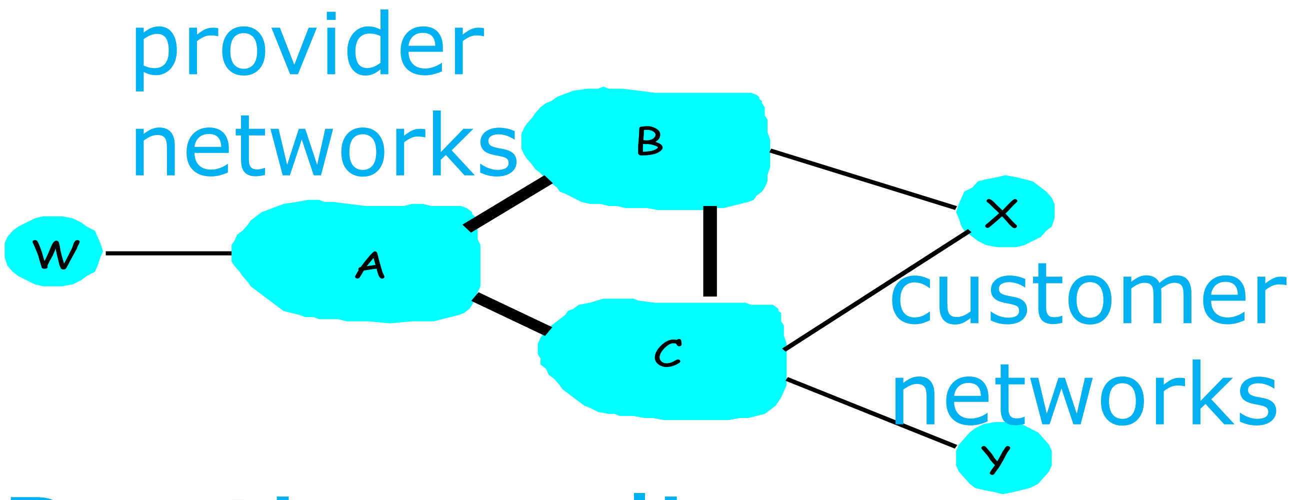
BGP



Routing policy:

- Provider networks: A, B, C
- Customer networks (of provider networks): X, Y, W

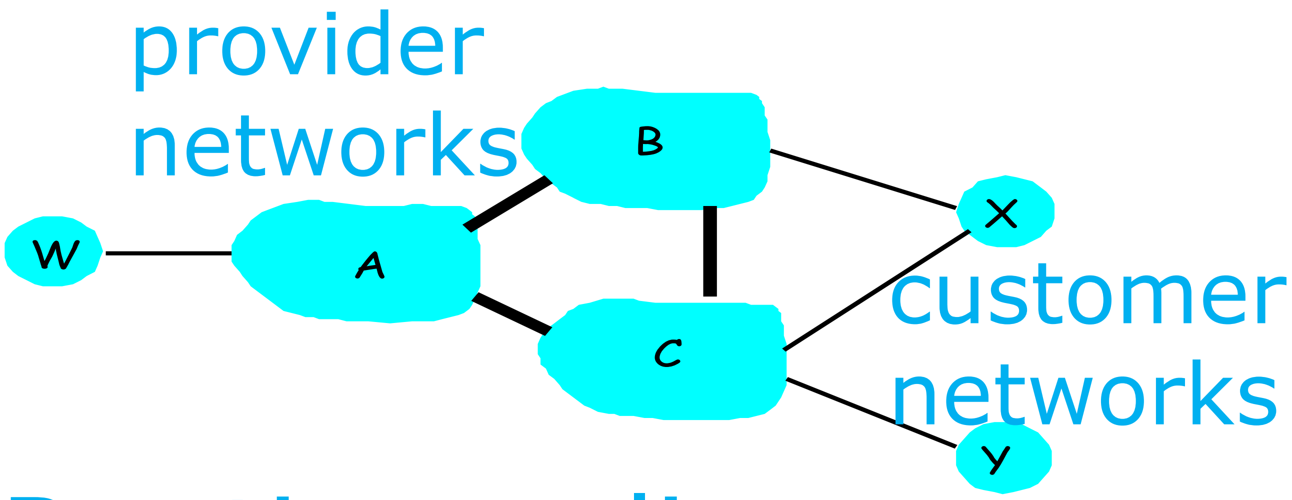
BGP



Routing policy:

- Provider networks: A, B, C
- Customer networks (of provider networks): X, Y, W
- X is dual-homed: attached to two networks

BGP

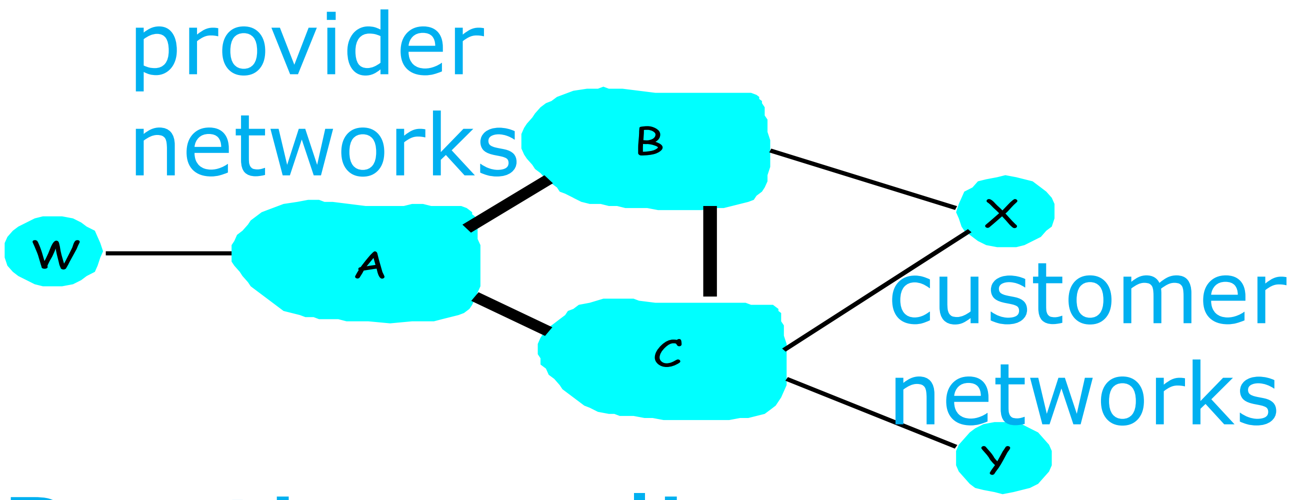


X does not want to carry traffic from B to C, so X will not advertise to B a route to C.

Routing policy:

- Provider networks: A, B, C
- Customer networks (of provider networks): X, Y, W
- X is dual-homed: attached to two networks

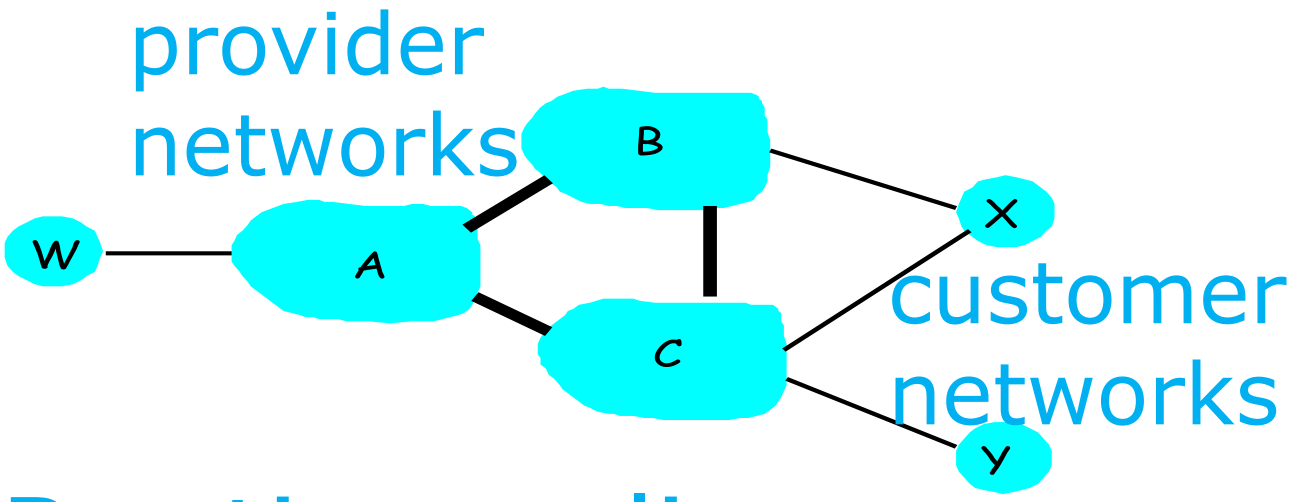
BGP



Routing policy:

- A advertises to B the path AW
- B advertises to X the path BAW

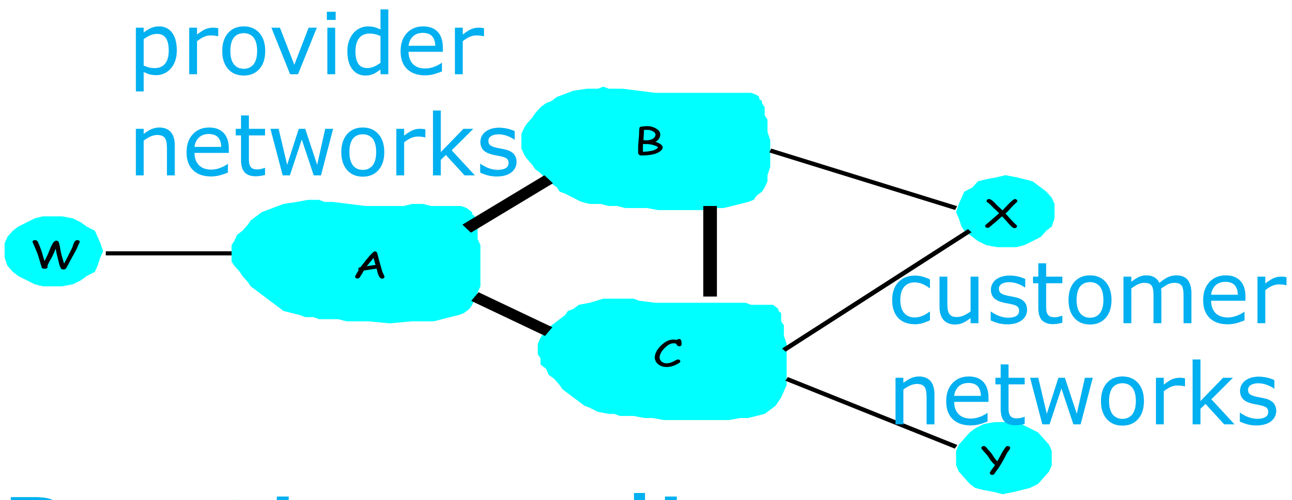
BGP



Routing policy:

- A advertises to B the path AW
- B advertises to X the path BAW
- Should B advertise to C the path BAW?

BGP



No way!
B gets no revenue for routing CBAW as neither W nor C is B’s customer.
B wants to route only to/from its customers.

Routing policy:

- A advertises to B the path AW
- B advertises to X the path BAW
- Should B advertises to C the path BAW?

routing attacks

distance-vector

link-state

BGP

routing attacks

distance-vector:

announce 0 distance to all other nodes

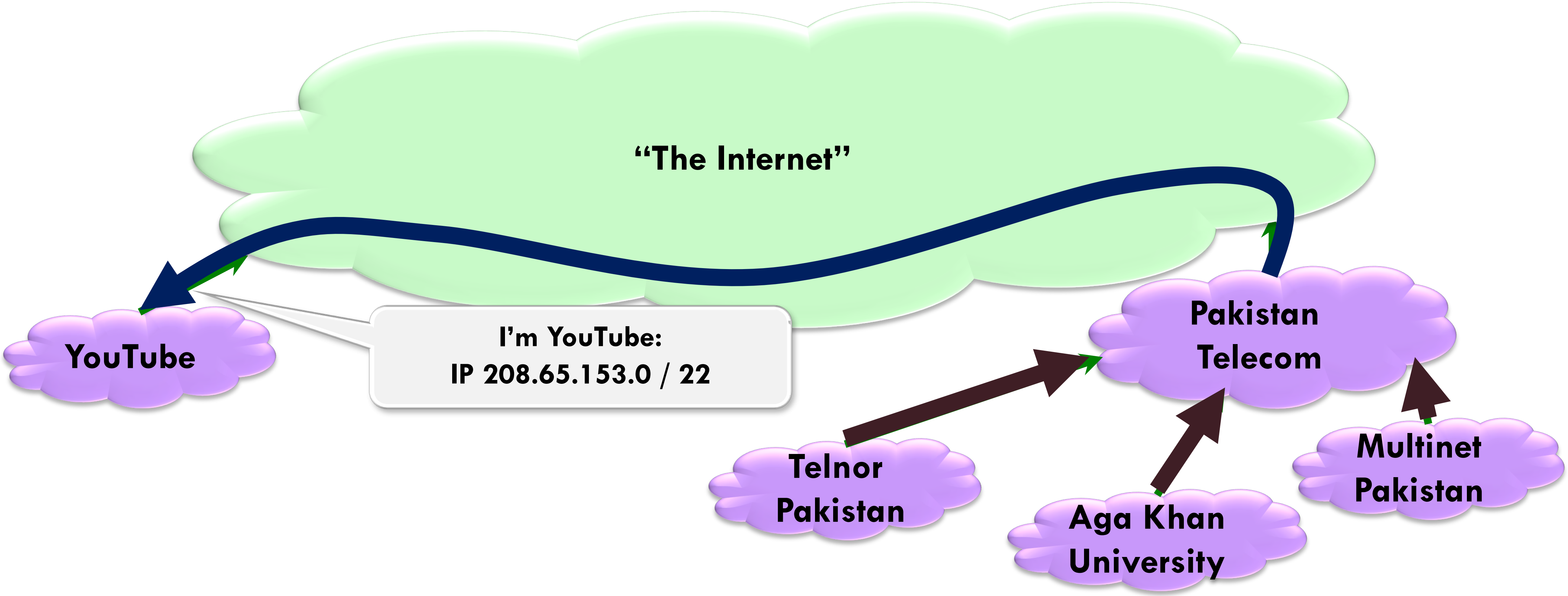
link-state:

drop links; claim direct link to any other routers

BGP:

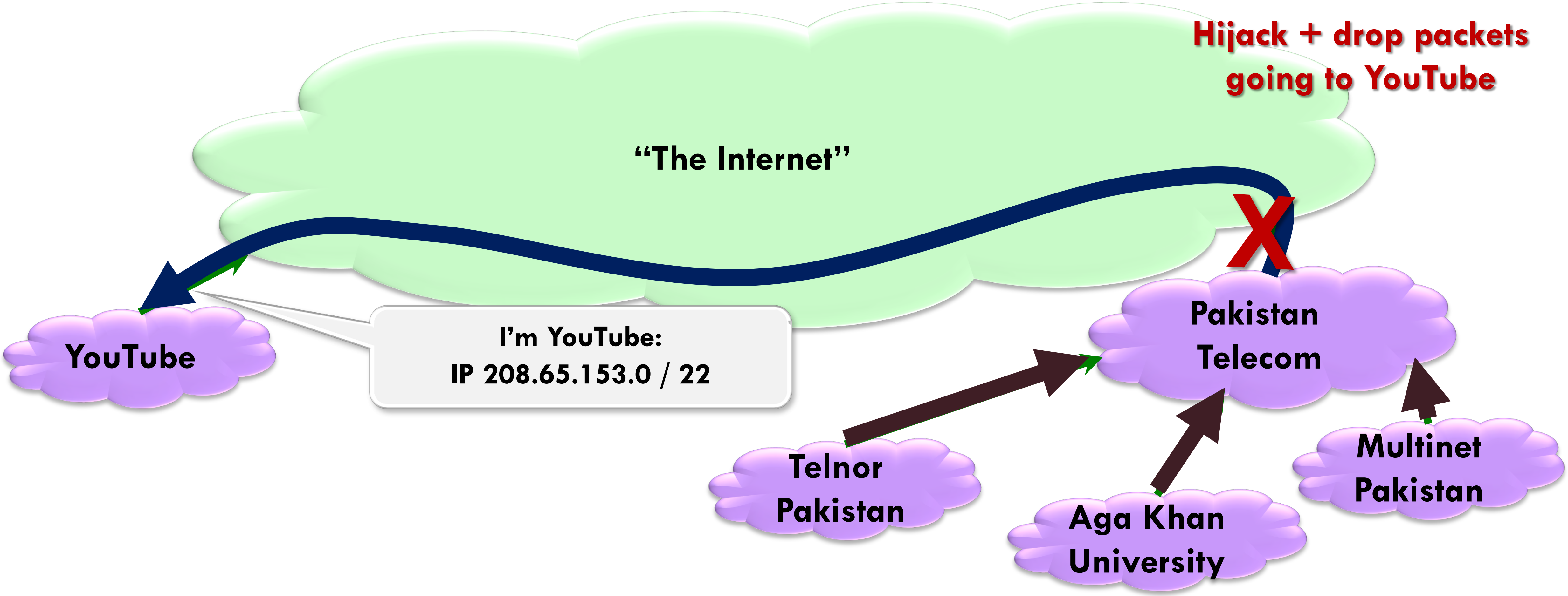
announce arbitrary prefix; alter paths

Prefix Hijacking: Case 1



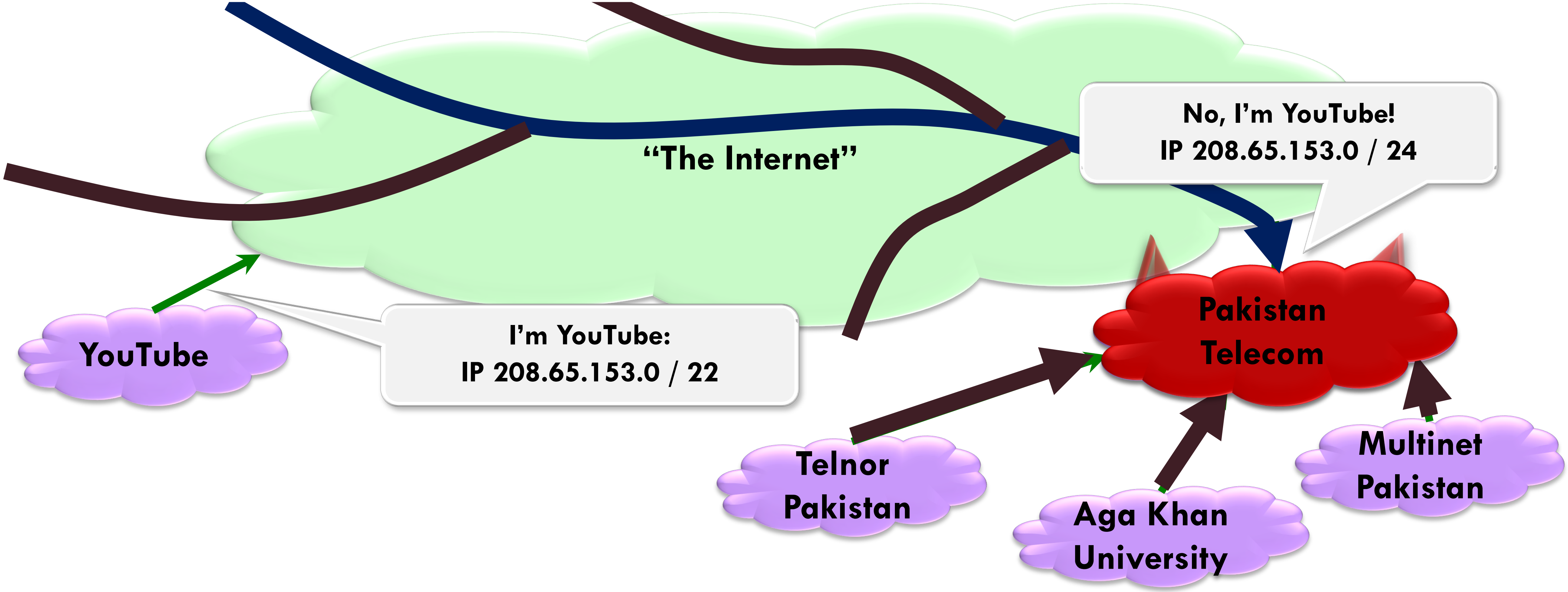
examples from https://people.cs.umass.edu/~phillipa/CSE390/RoutingSecurity.pptx

Here's what should have happened….

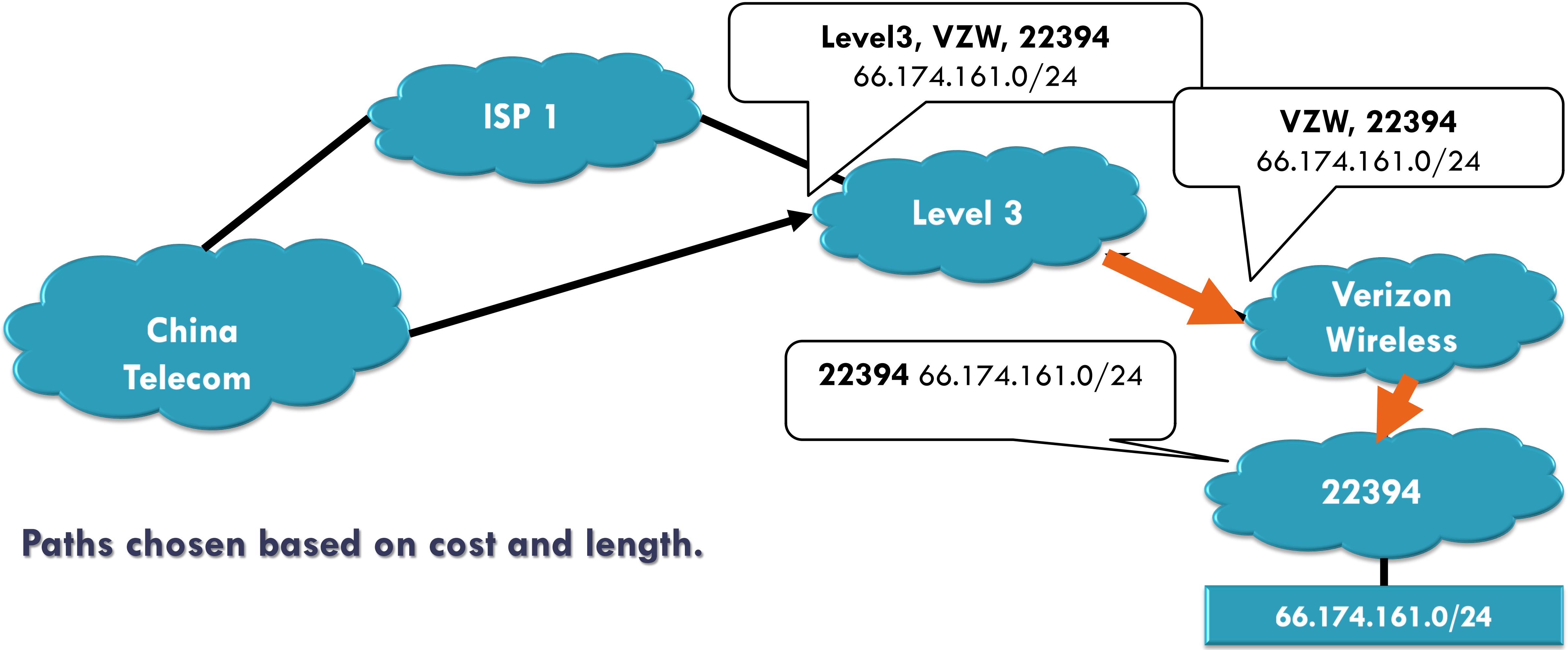


Block your own customers.

But here's what Pakistan ended up doing…

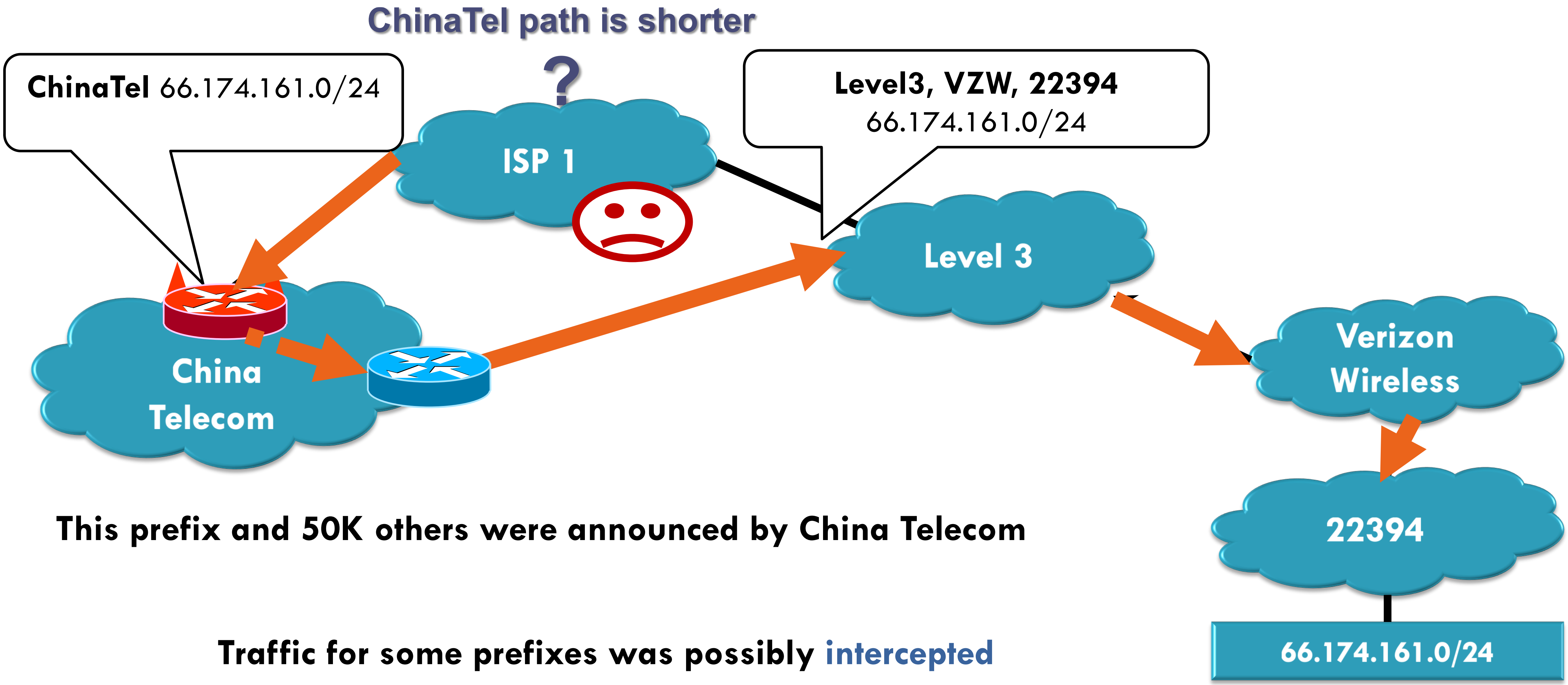


Prefix Hijacking: Case 2



Paths chosen based on cost and length.

April 2010 : China Telecom intercepts traffic



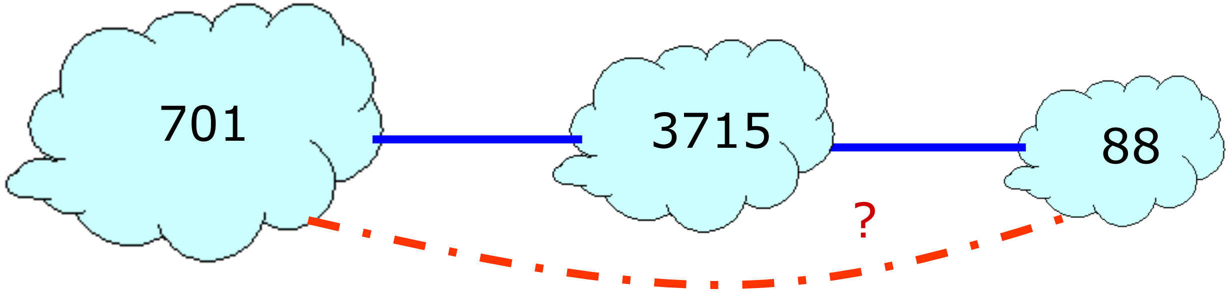
This prefix and 50K others were announced by China Telecom

Traffic for some prefixes was possibly intercepted

Path Tampering

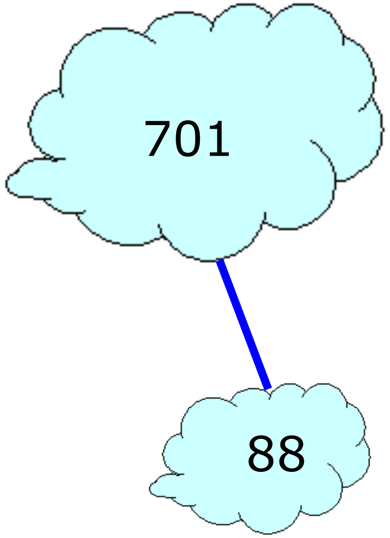
- Remove ASes from the AS path

701 ~~3715~~ 88



- Add ASes to the AS path

701 88 →
701 3715 88

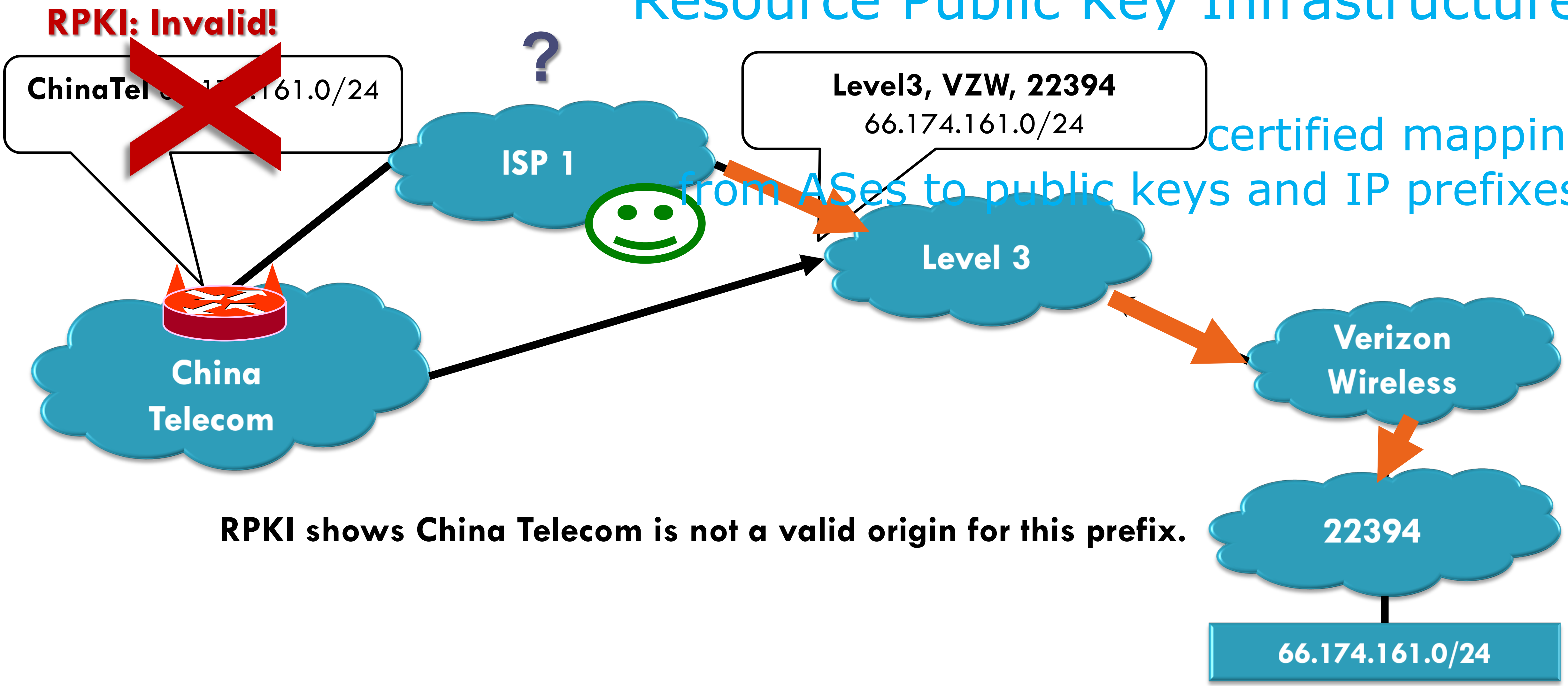


how to secure routing?

RPKI

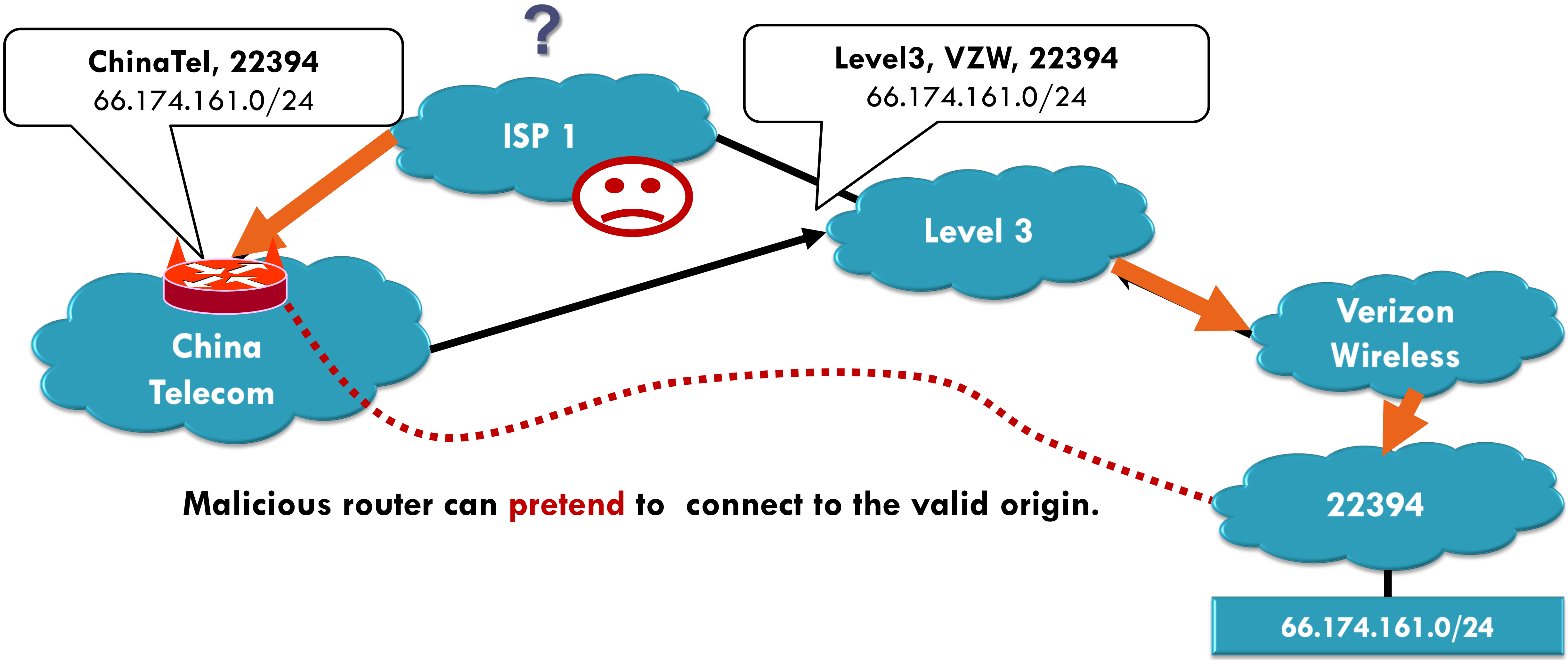
Resource Public Key Infrastructure

certified mapping from ASes to public keys and IP prefixes



RPKI shows China Telecom is not a valid origin for this prefix.

RPKI
insufficient!



Malicious router can pretend to connect to the valid origin.

S-BGP

- Each AS on the path cryptographically signs its announcement
- Guarantees that each AS on the path made the announcement in the path:

  AS path indicates the order ASes were traversed;

  No intermediate ASes were added or removed;

S-BGP

Deployment challenges:

- Complete, accurate registries
- Public key infrastructure
- Cryptographic operations
- Need to perform operations quickly
- Difficulty of incremental deployment

select a path for traffic in a network

Routing

select a path for traffic in a network

Routing

Forwarding ?

relay packets along a certain path

Forwarding Anomaly Threat

- Performance
  downgrade service quality
- Security
  bypass attacking-traffic filter

Path Validation

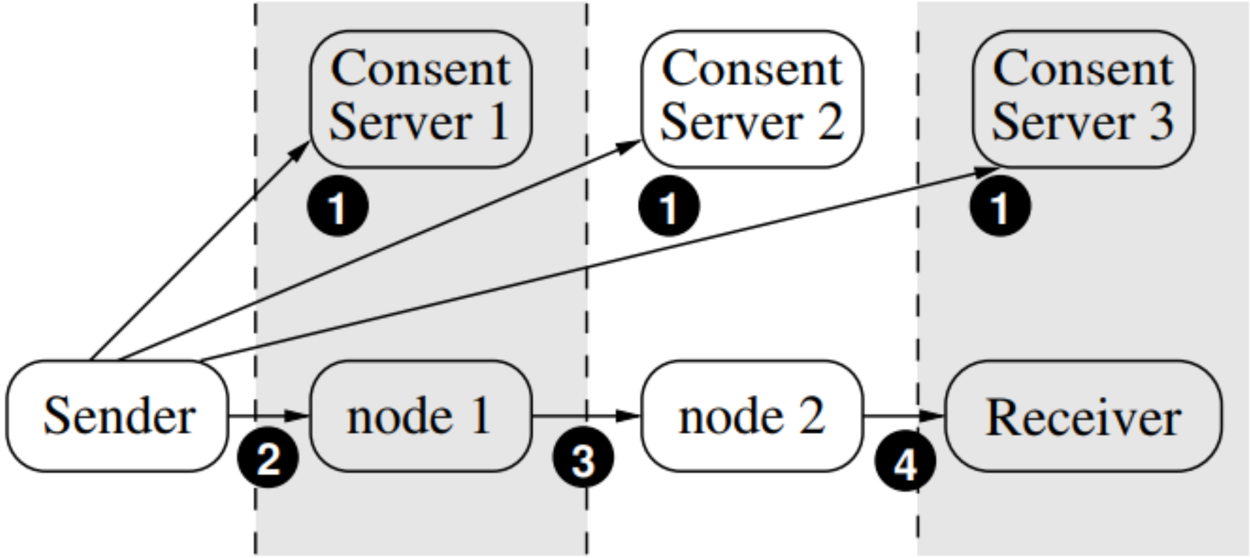
- PoC: Proof of Consent

  certify the provider's consent to carry traffic along the path

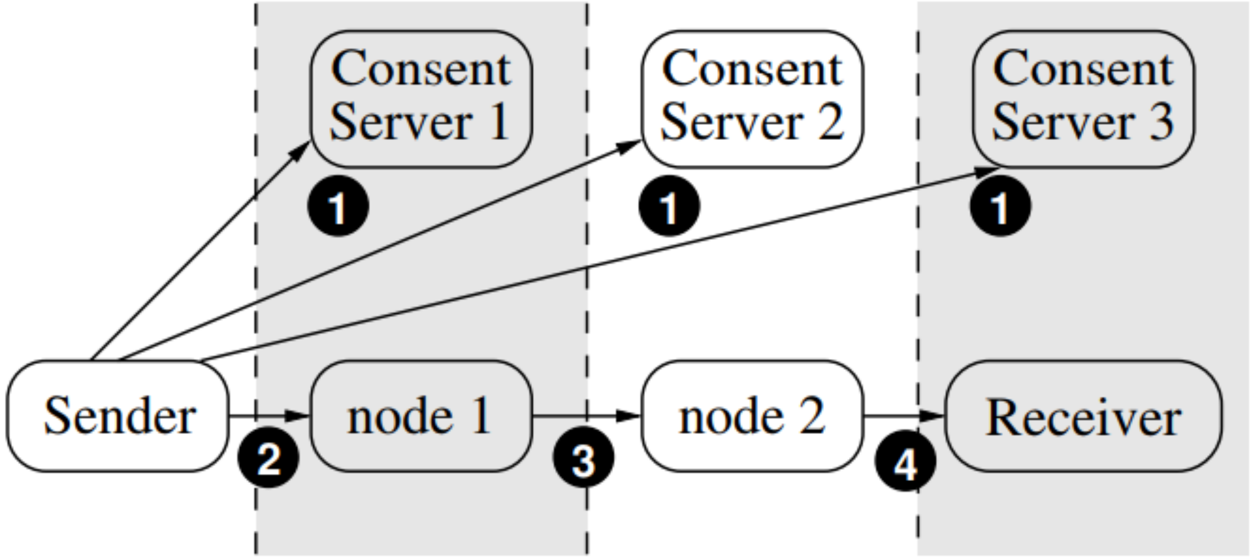
- PoP: Proof of Provenance

  allow upstream nodes to prove to downstream nodes that they carried the packet

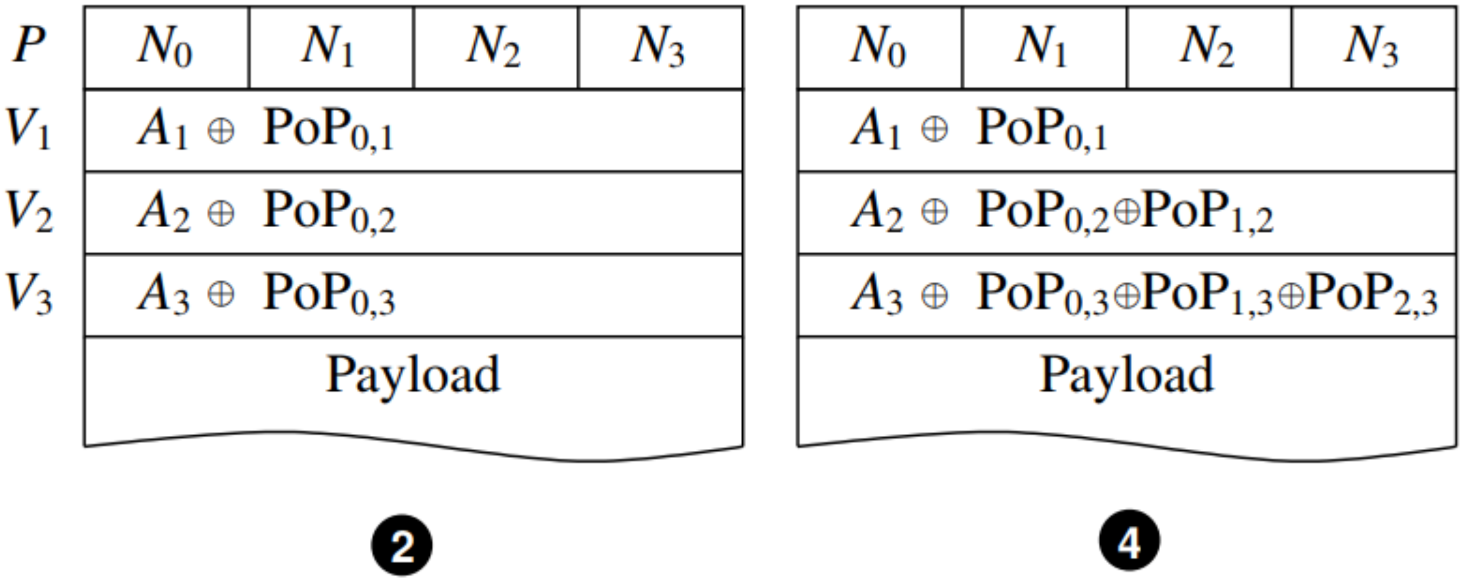
Path Validation



Path Validation



https://cs.nyu.edu/~mwalfish/papers/icing-conext11.pdf



computation-less device?

FlowCloak: Defeating Middlebox-Bypass Attacks in Software-Defined Networking

Middlebox

Middlebox: Pain Spot in modern networks

- Needs

Varieties of functions: Security & Performance

Widely deployed: A third of network devices

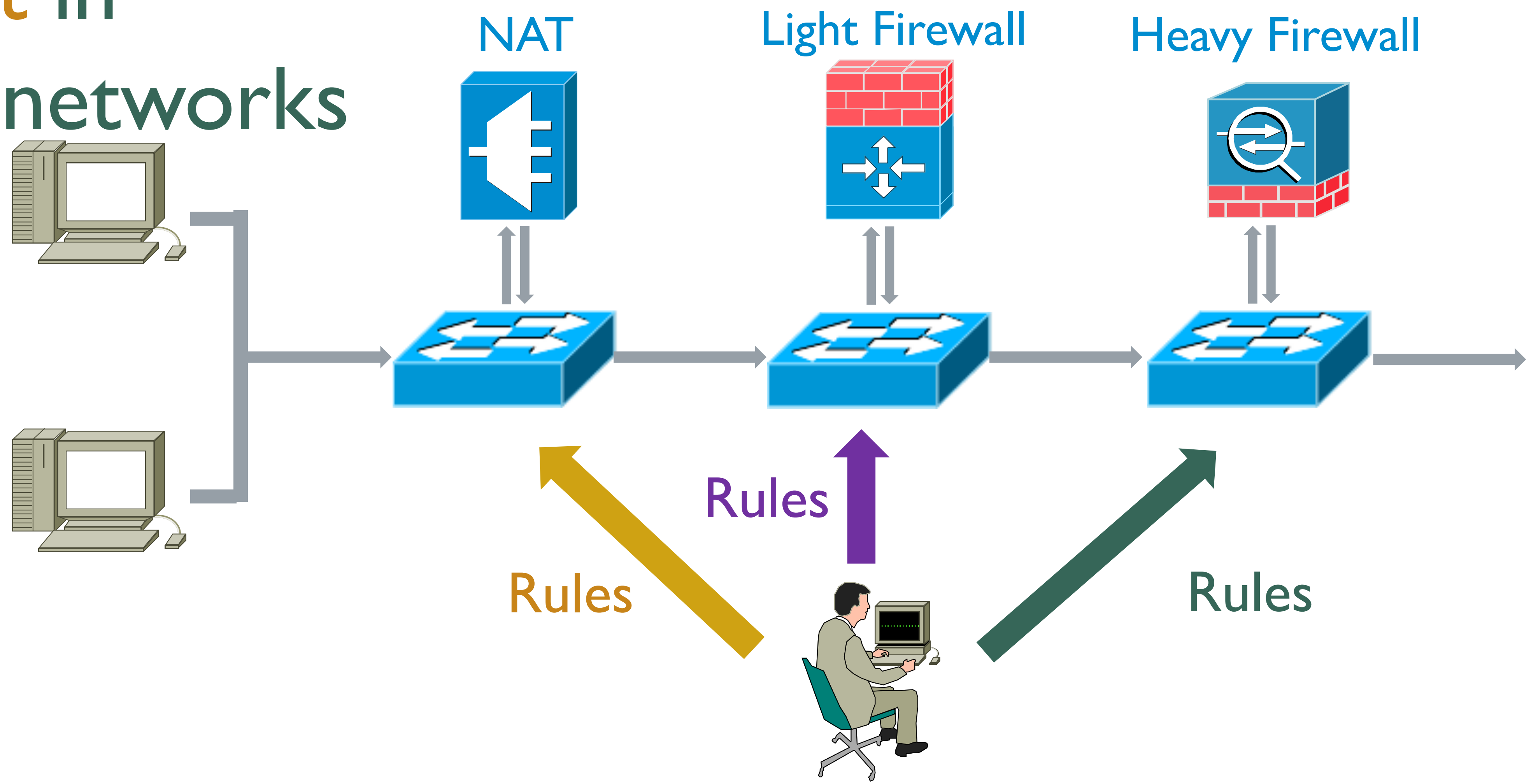
- Troubles

Deployment and configuration:
Complex & Error-prone

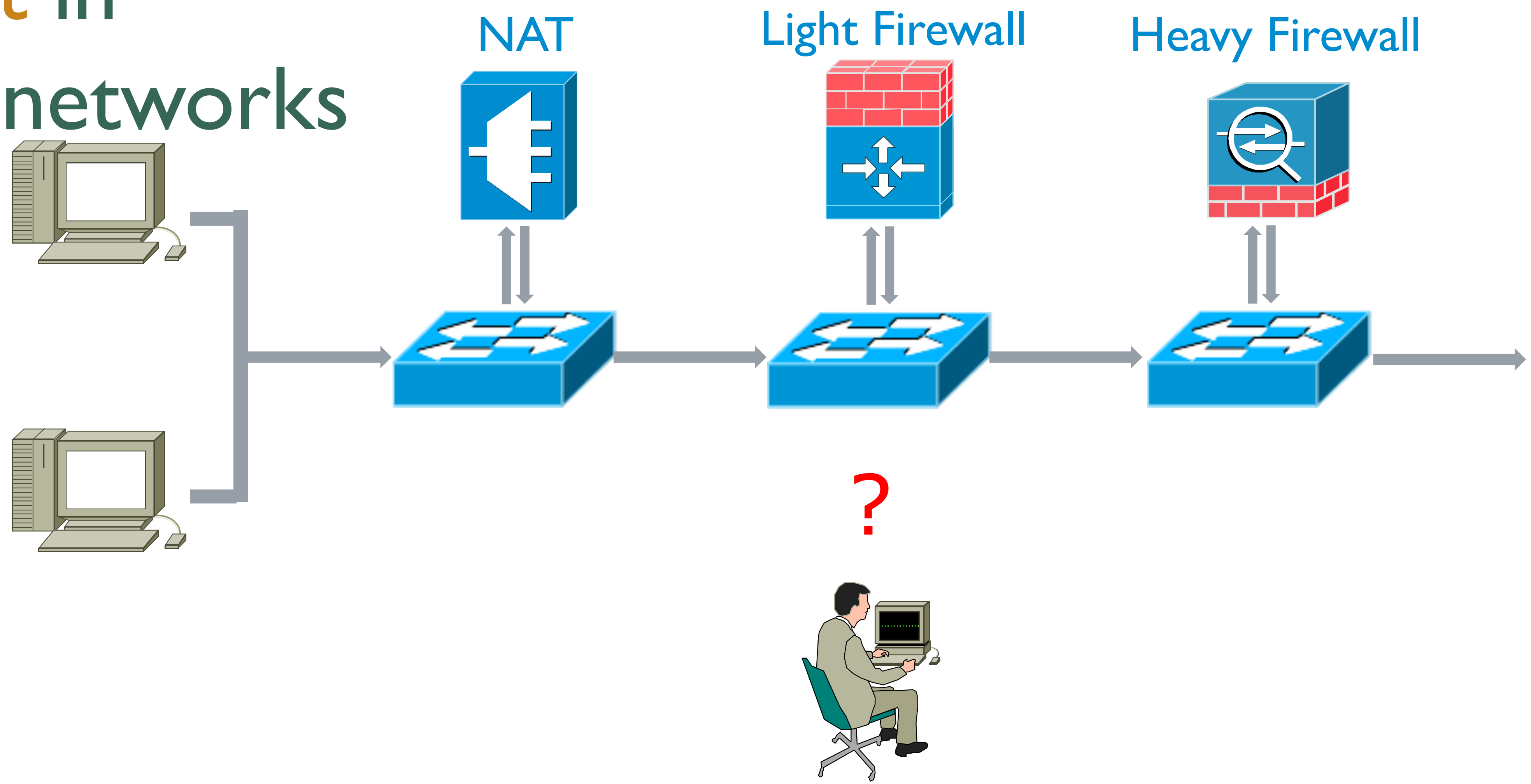
Costs: Personnel, Money, Time

Middlebox: Pain Spot in modern networks

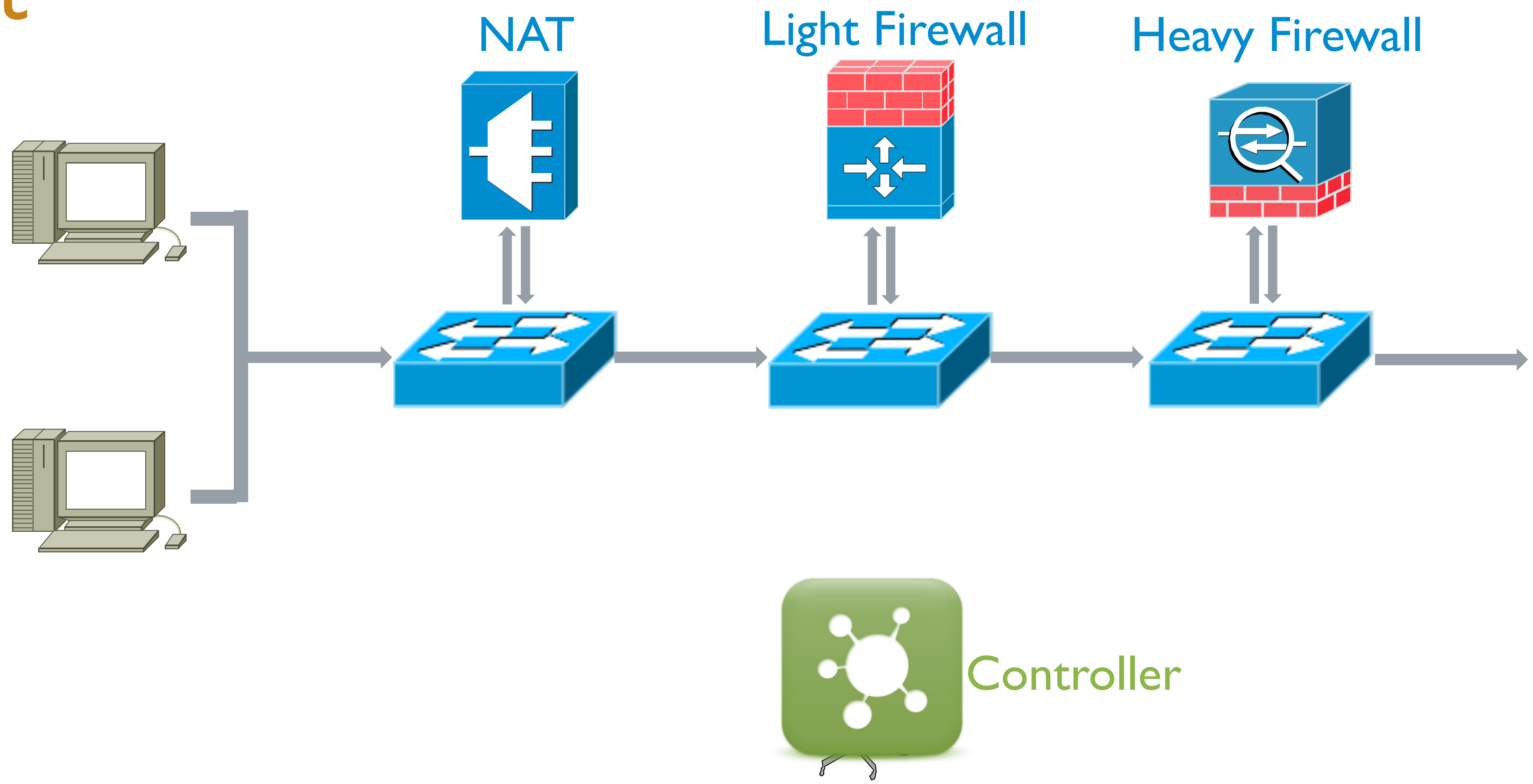
Middlebox: Pain Spot in modern networks



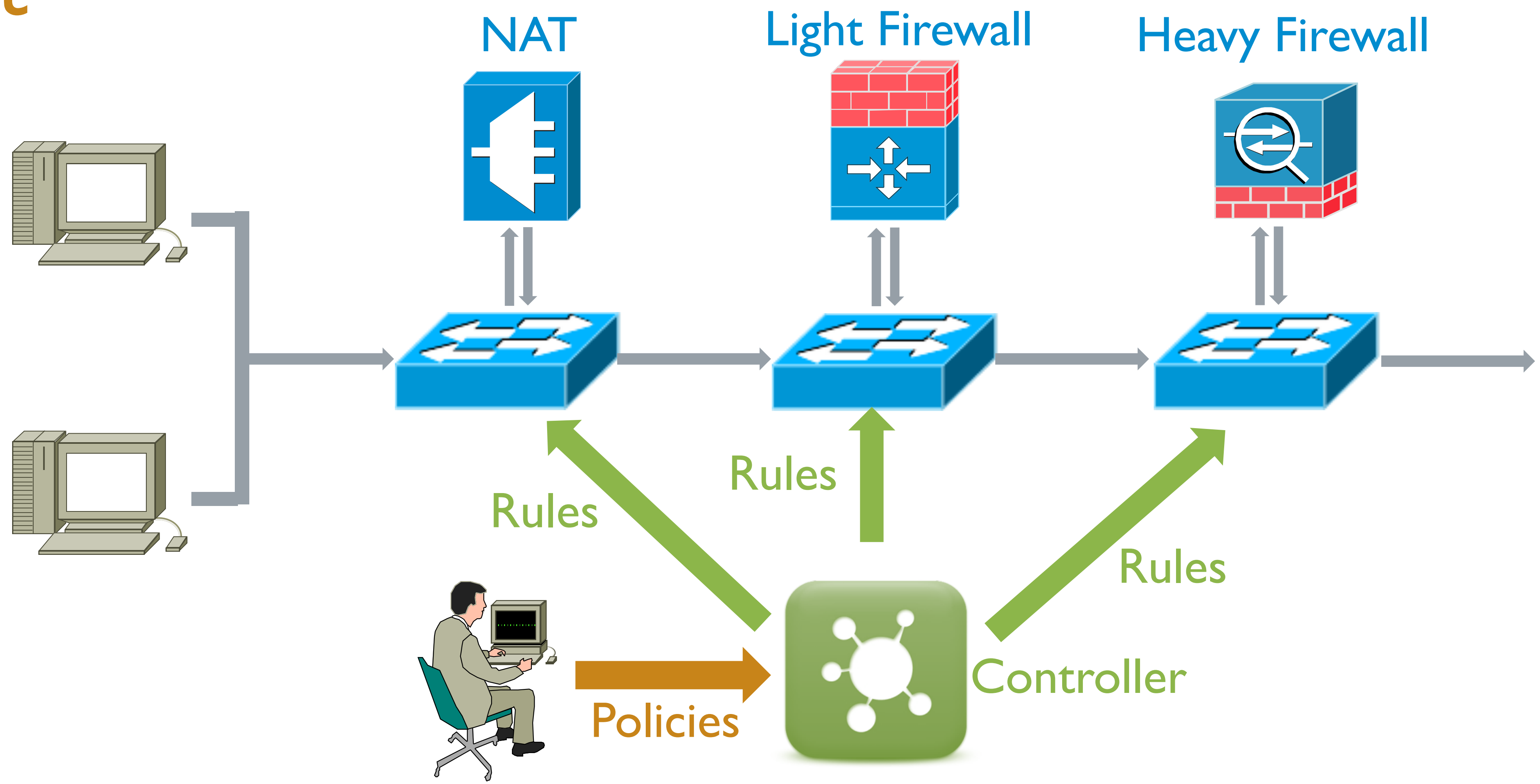
Middlebox: Pain Spot in modern networks



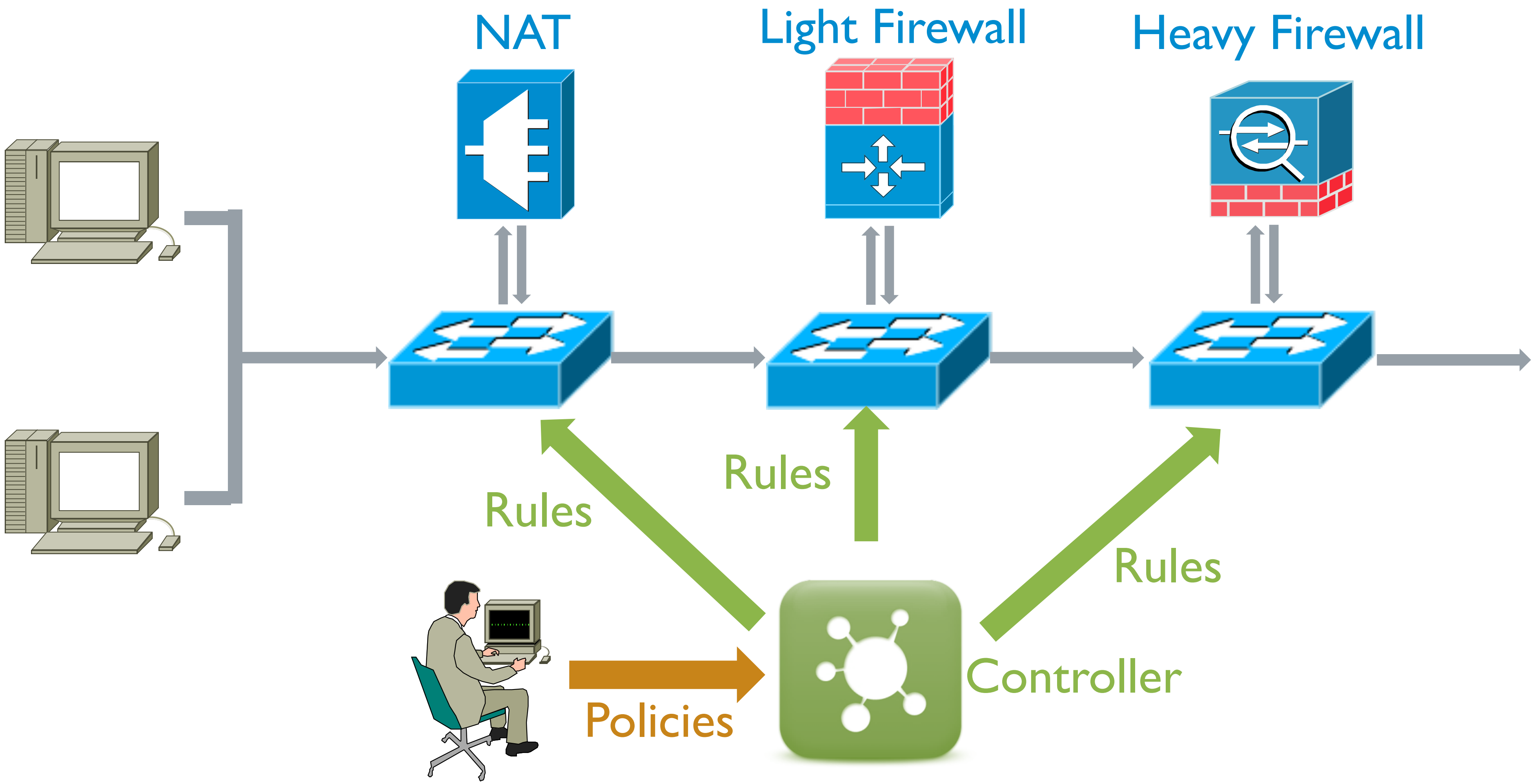
Middlebox: Pain Spot SDN



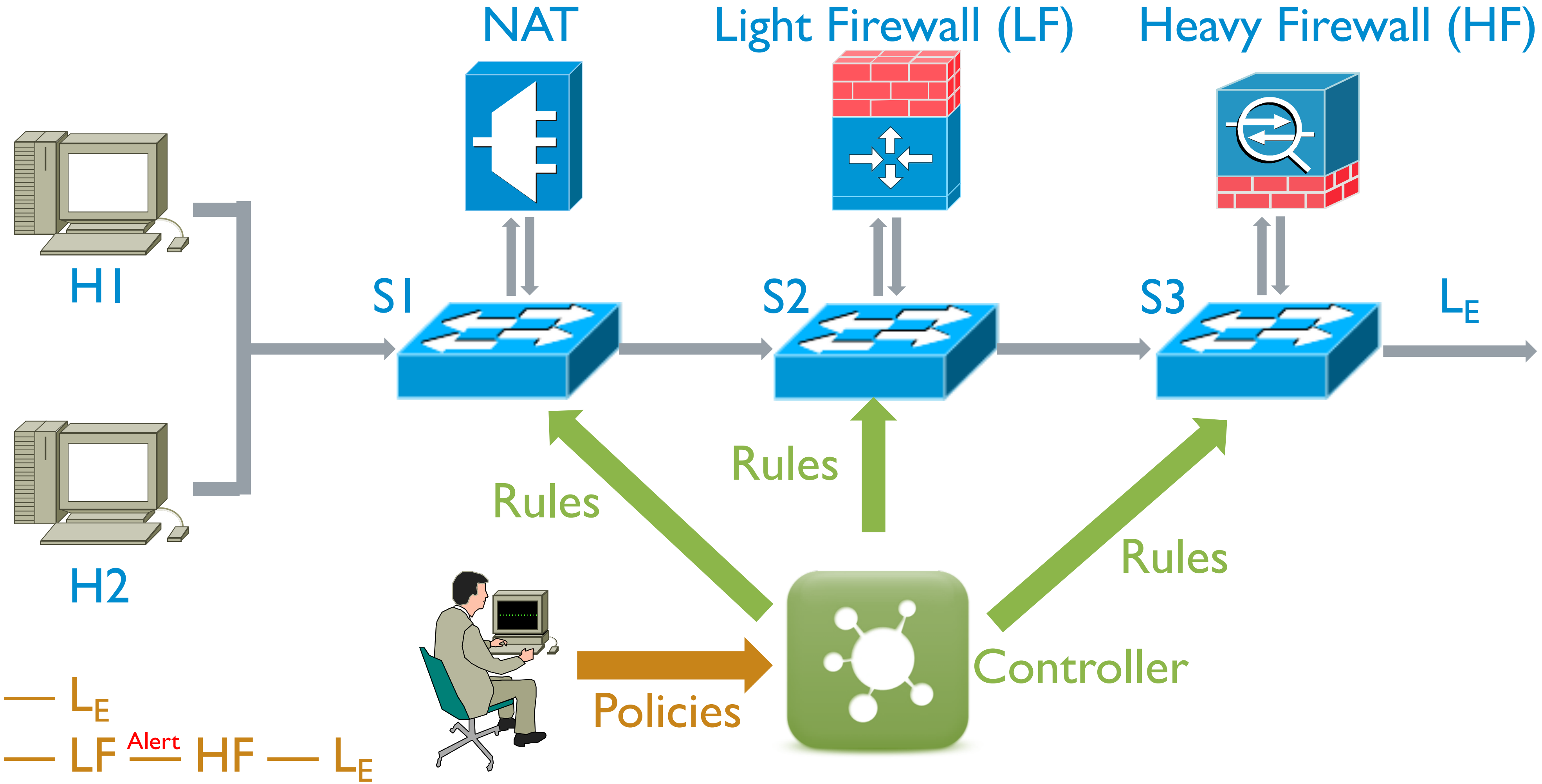
Middlebox:
Pain Spot
SDN



Middlebox meets SDN



Middlebox meets SDN



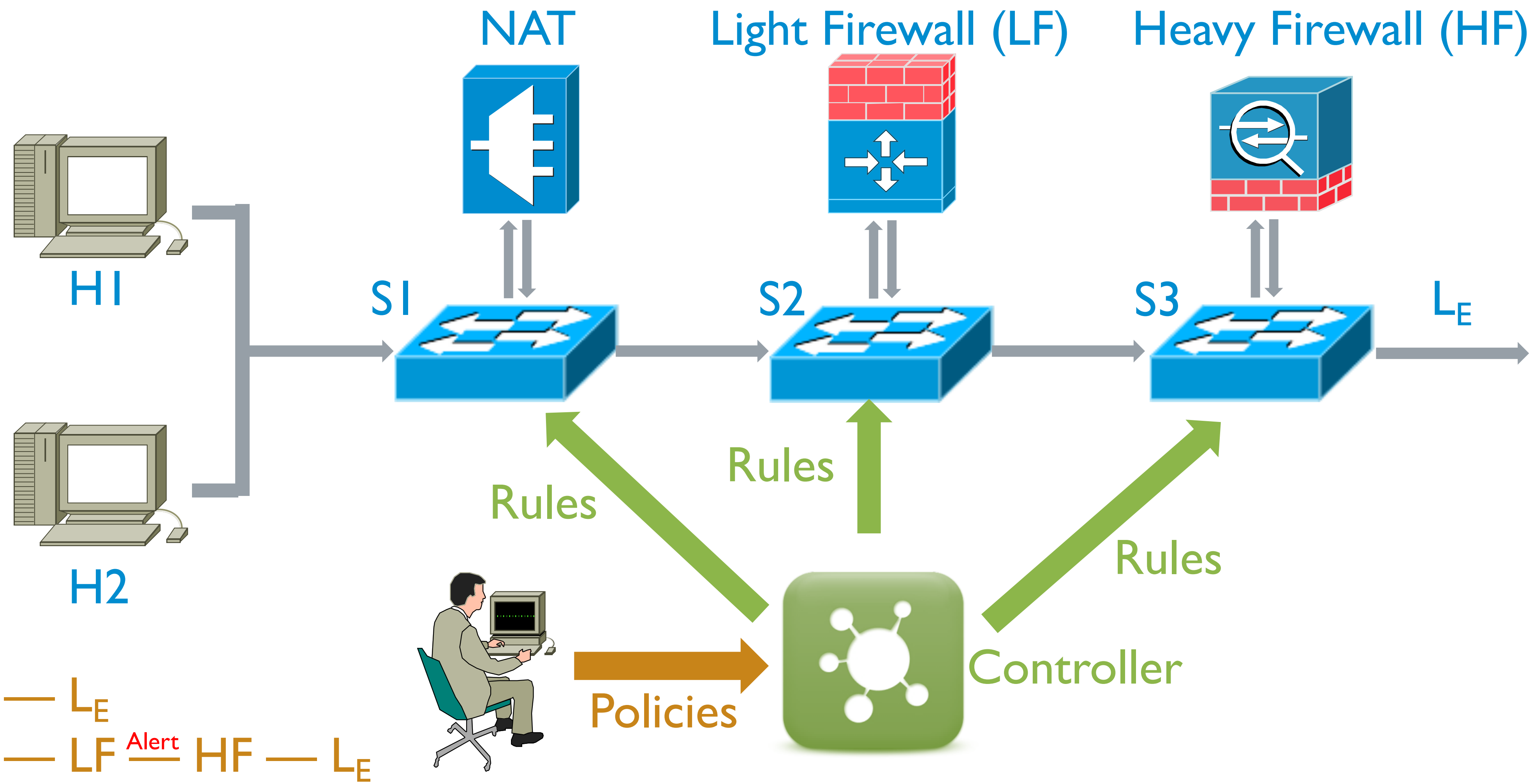
Policies:

(1) H1 — NAT — L_E

(2) H2 — NAT — LF $\overset{\text{Alert}}{—}$ HF — L_E

Forwarding Ambiguity

Middlebox meets SDN



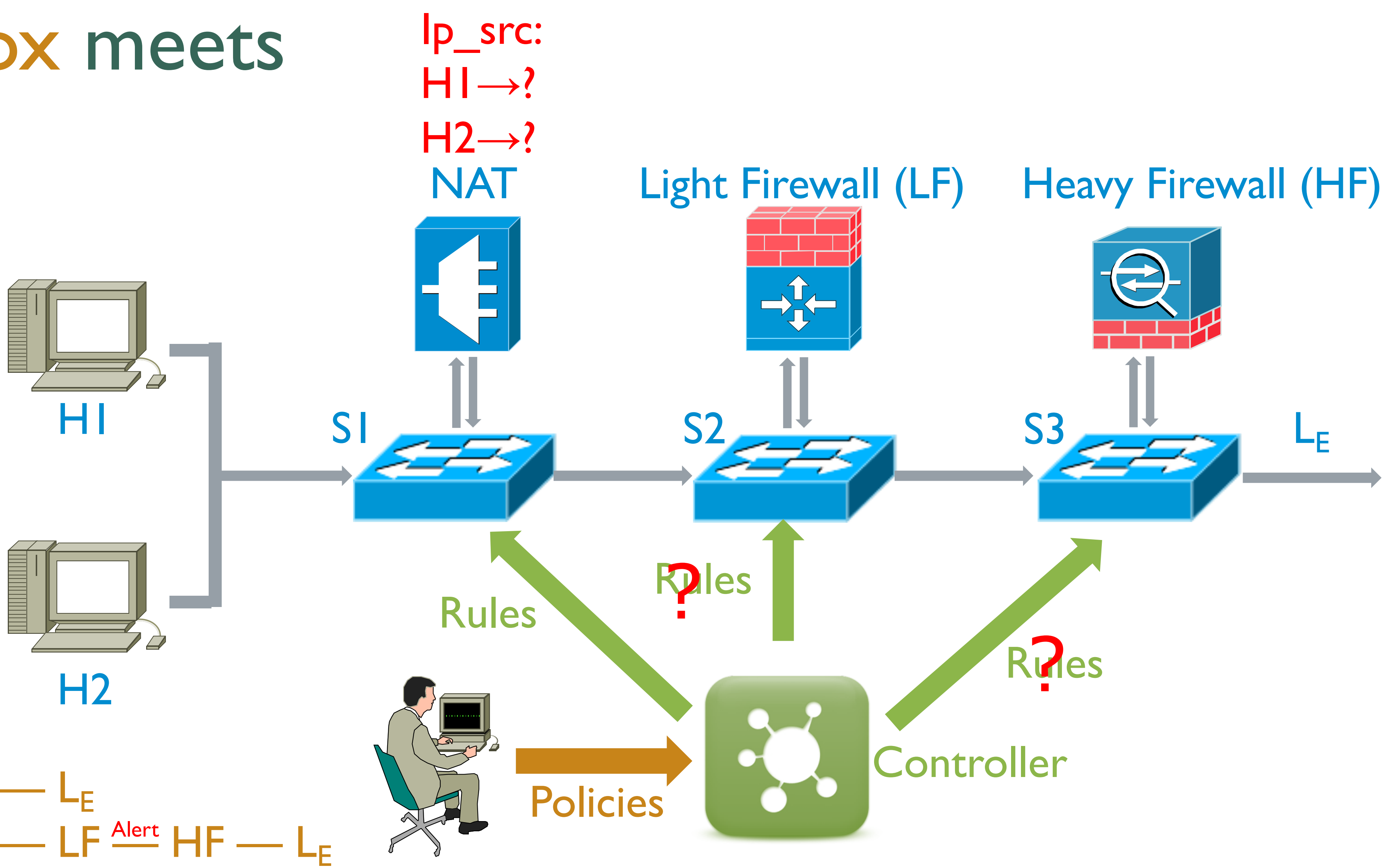
Policies:

(1) H1 — NAT — L_E

(2) H2 — NAT — LF $\overset{\text{Alert}}{—}$ HF — L_E

Forwarding Ambiguity

Middlebox meets SDN



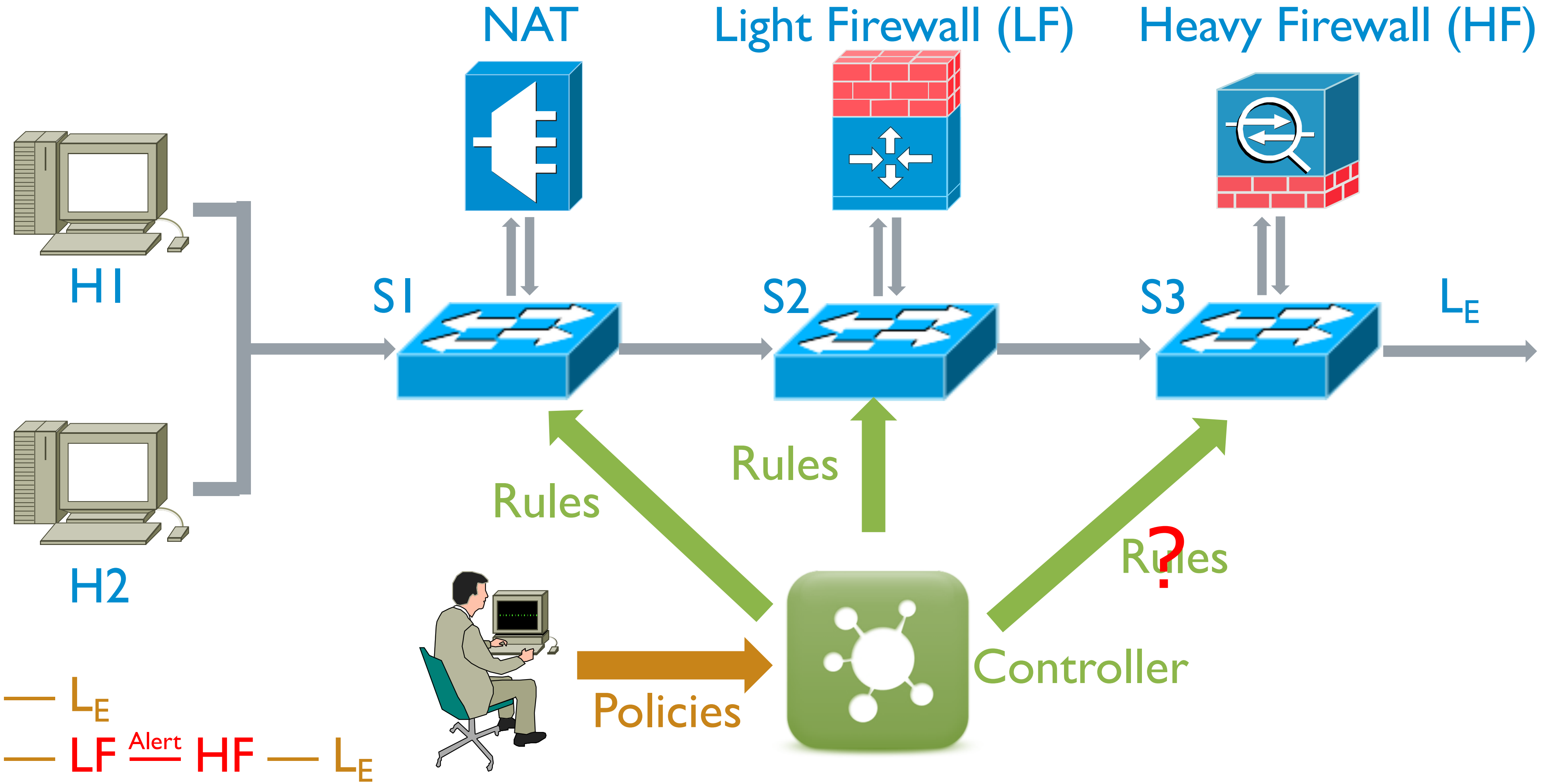
Policies:

(1) H1 — NAT — L_E

(2) H2 — NAT — LF $\overset{\text{Alert}}{—}$ HF — L_E

Forwarding Ambiguity

Middlebox meets SDN



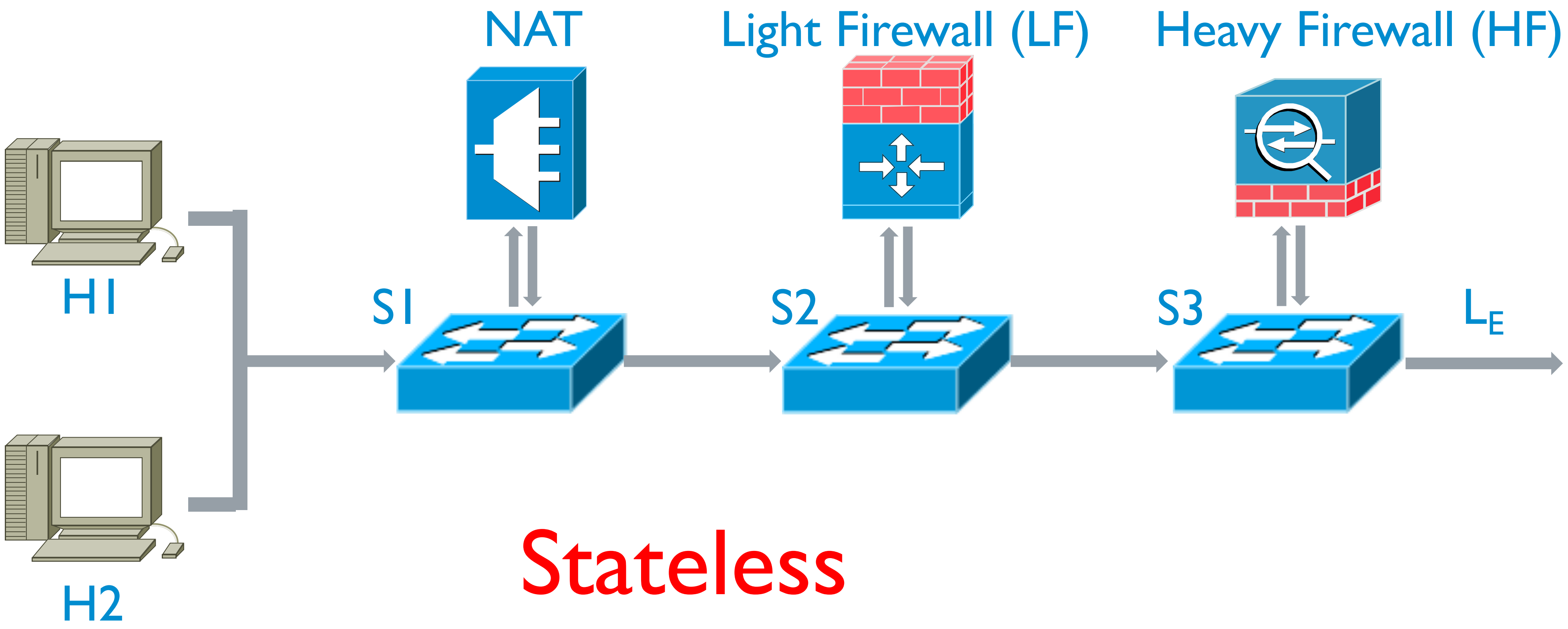
Policies:

(1) H1 — NAT — L_E

(2) H2 — NAT — LF $\xrightarrow{\text{Alert}}$ HF — L_E

Forwarding Ambiguity

Middlebox meets SDN



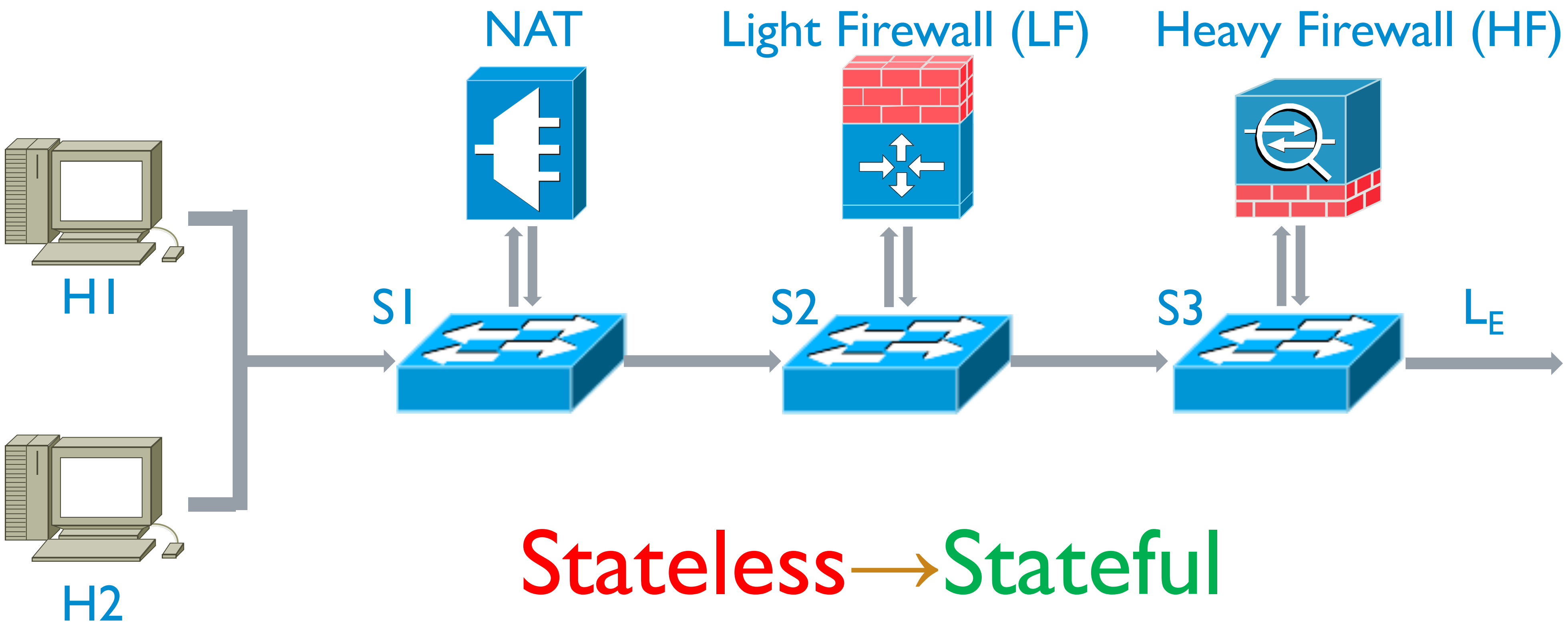
Stateless

Policies:

(1) H1 — NAT — L_E

(2) H2 — NAT — LF $\overset{\text{Alert}}{—}$ HF — L_E

Middlebox meets SDN



Stateless ⟶ Stateful

Policies:

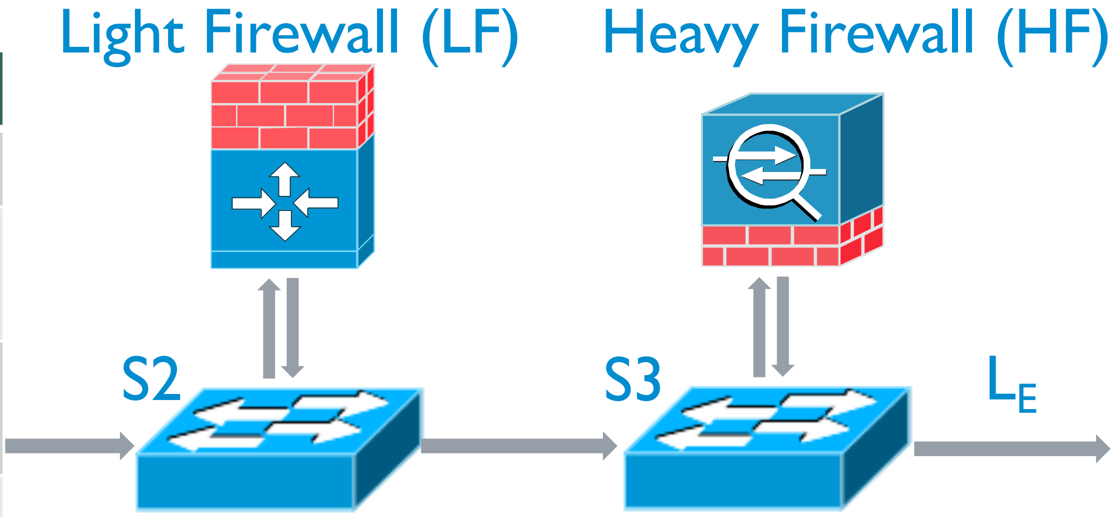
(1) H1 — NAT — L_E

(2) H2 — NAT — LF $\overset{\text{Alert}}{—}$ HF — L_E

Middlebox meets SDN

NAT

Switch	Some Crucial Rules	
	Matching	Action
S2	tag=<src:H2, NAT>, interface=S2:S1	fwd(LF)
S2	tag=<src:H1,NAT>, interface=S2:S1	fwd(S3)
S3	tag=<src:H2, LF, alert>, interface=S3:S2	fwd(HF)
S3	tag=<src:H2, LF, pass> Interface=S3:S2	fwd(L_E)



Flowtags [NSDI '14]

Stateful Tags on packer header

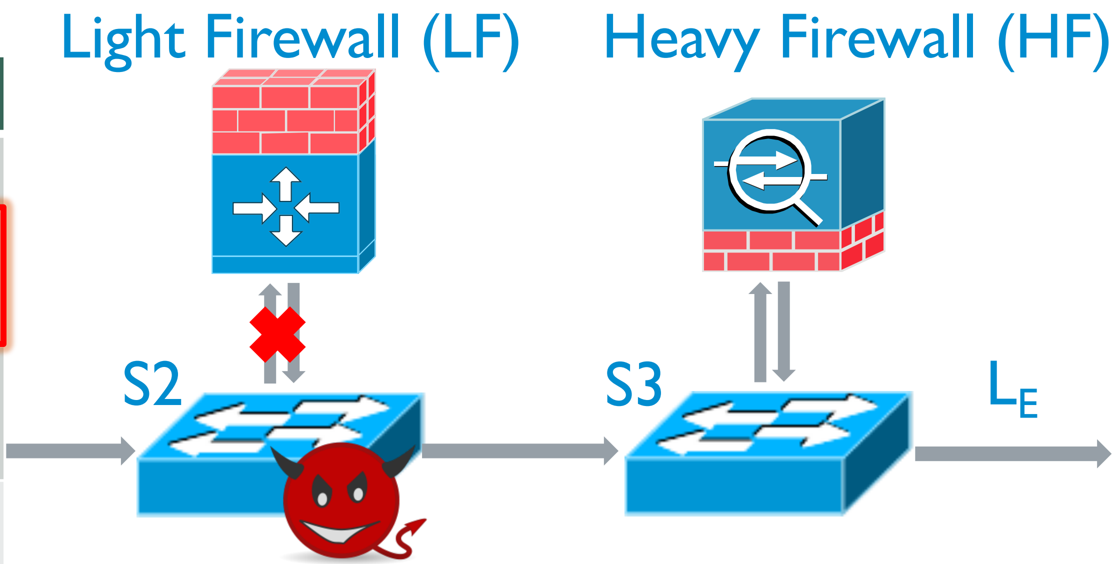
Policies:

(1) H1 — NAT — L_E

(2) H2 — NAT — LF $\overset{\text{Alert}}{—}$ HF — L_E

Middlebox-Bypass Attacks
SDN

Switch	Some Crucial Rules	
	Matching	Action
S2	tag=<src:H2, NAT>, interface=S2:S1	fwd(LF)
S2	tag=<src:H1,NAT>, interface=S2:S1	fwd(S3)
S3	tag=<src:H2, LF, alert>, interface=S3:S2	fwd(HF)
S3	tag=<src:H2, LF, pass> Interface=S3:S2	fwd(L_E)



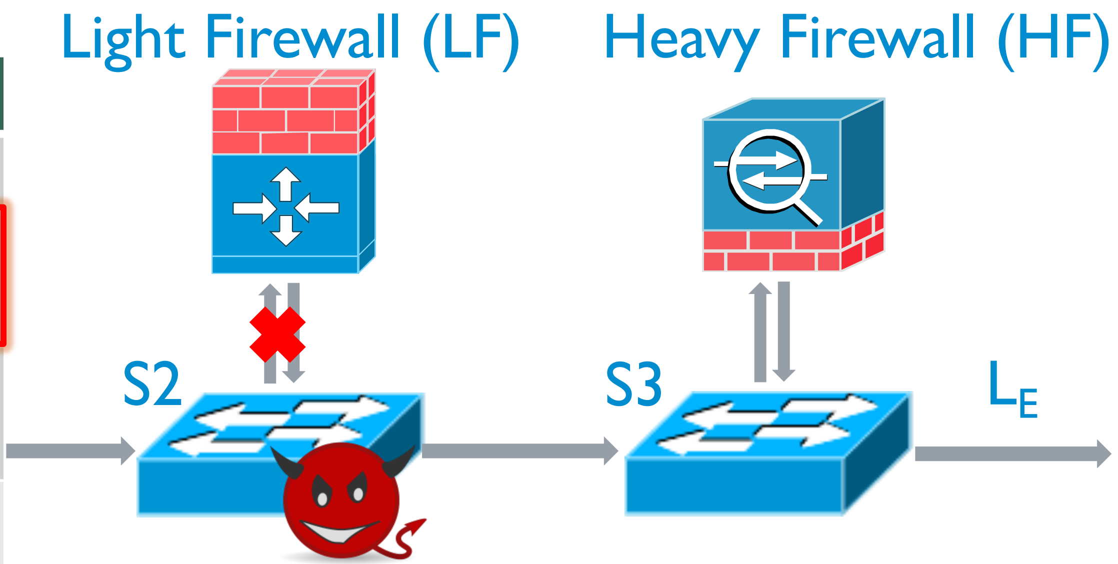
Policies:

(1) H1 — NAT — L_E

(2) H2 — NAT — LF $\overset{\text{Alert}}{—}$ HF — L_E

Middlebox-Bypass Attacks

Switch	Some Crucial Rules	
	Matching	Action
S2	tag=<src:H2, NAT>, interface=S2:S1	tag(LF, pass) fwd(HF)
S2	tag=<src:H1,NAT>, interface=S2:S1	fwd(S3)
S3	tag=<src:H2, LF, alert>, interface=S3:S2	fwd(HF)
S3	tag=<src:H2, LF, pass> Interface=S3:S2	fwd(L_E)



Policies:

(1) H1 — NAT — L_E

(2) H2 — NAT — LF $\overset{Alert}{—}$ HF — L_E

Leads to:

- Severe security breaches
- Performance degradation

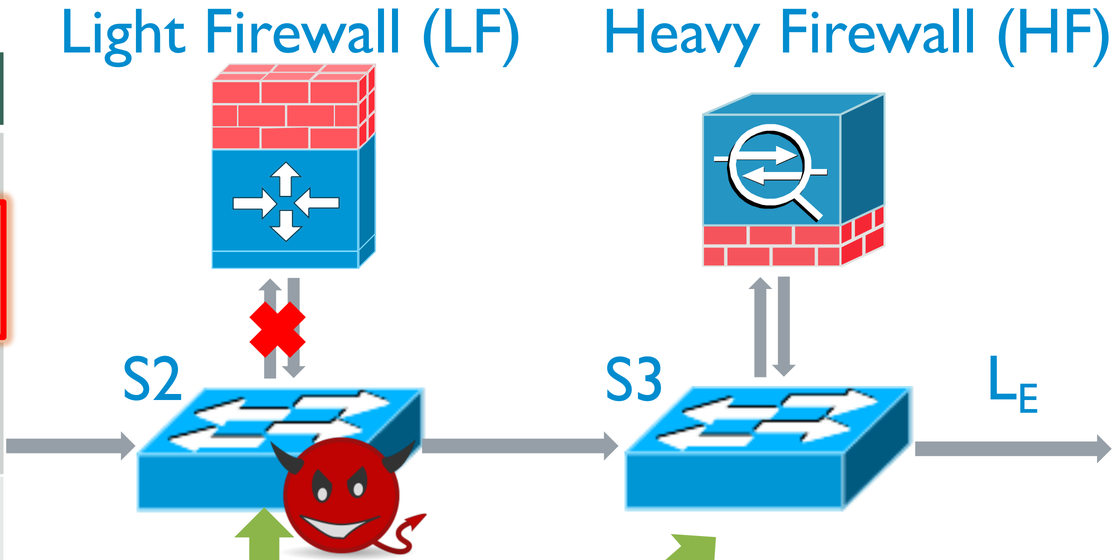
Middlebox-Bypass Attacks: More than Hypothesis

Switch	Some Crucial Rules	
	Matching	Action
S2	tag=<src:H2, NAT>, interface=S2:S1	fwd(LF)
S2	tag=<src:H1,NAT>, interface=S2:S1	fwd(S3)
S3	tag=<src:H2, LF, alert>, interface=S3:S2	fwd(HF)
S3	tag=<src:H2, LF, pass> Interface=S3:S2	fwd(L_E)

Policies:

(1) H1 — NAT — L_E

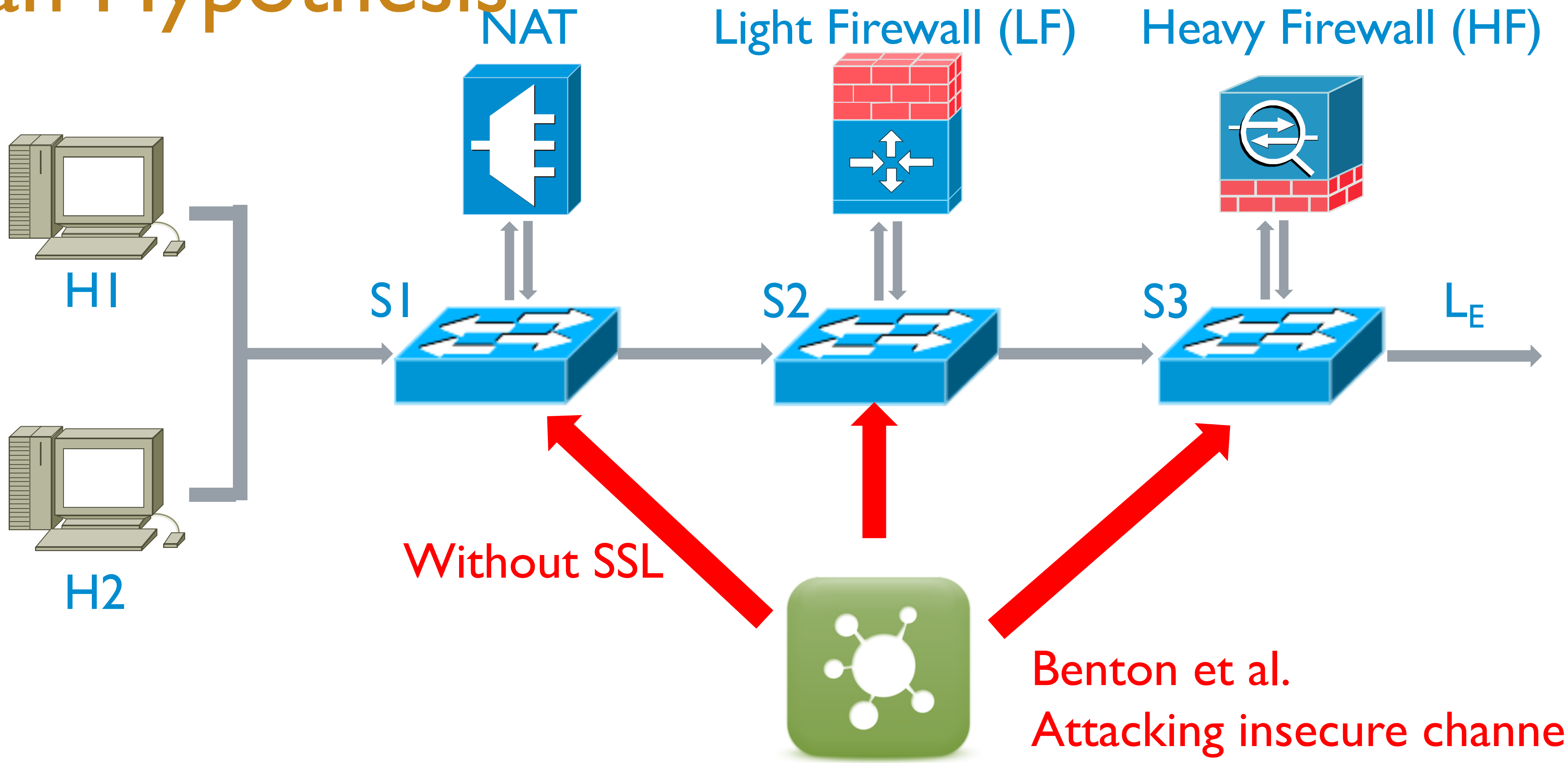
(2) H2 — NAT — LF $\overset{Alert}{—}$ HF — L_E



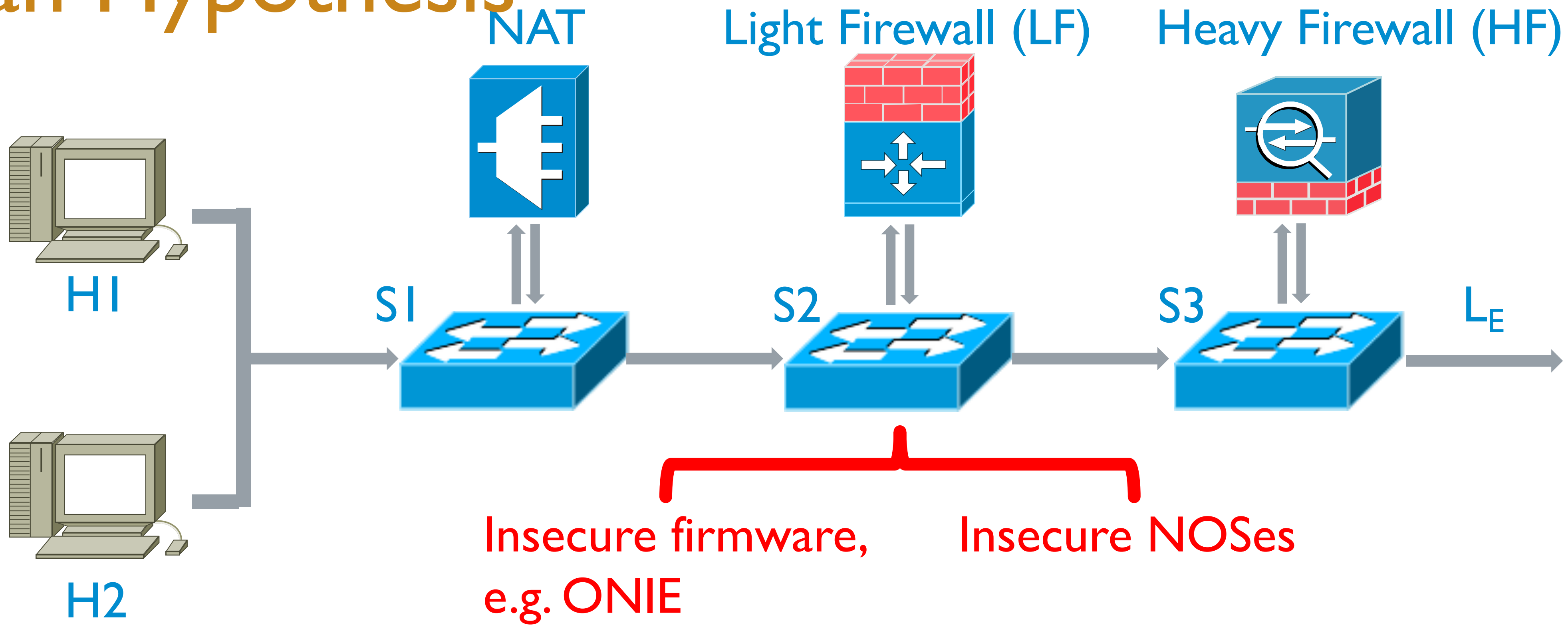
Leads to:

- Security breaches
- Performance degradation

Middlebox-Bypass Attacks: More than Hypothesis



Middlebox-Bypass Attacks: More than Hypothesis



Insecure firmware, e.g. ONIE

Insecure NOSes

Pickett @ DEFCON

Middlebox-Bypass Attacks: Existing malicious switch detection methods

- **Probe-based Methods**
  - Blinded by coward-attack
  - Waste valuable control channel bandwidth

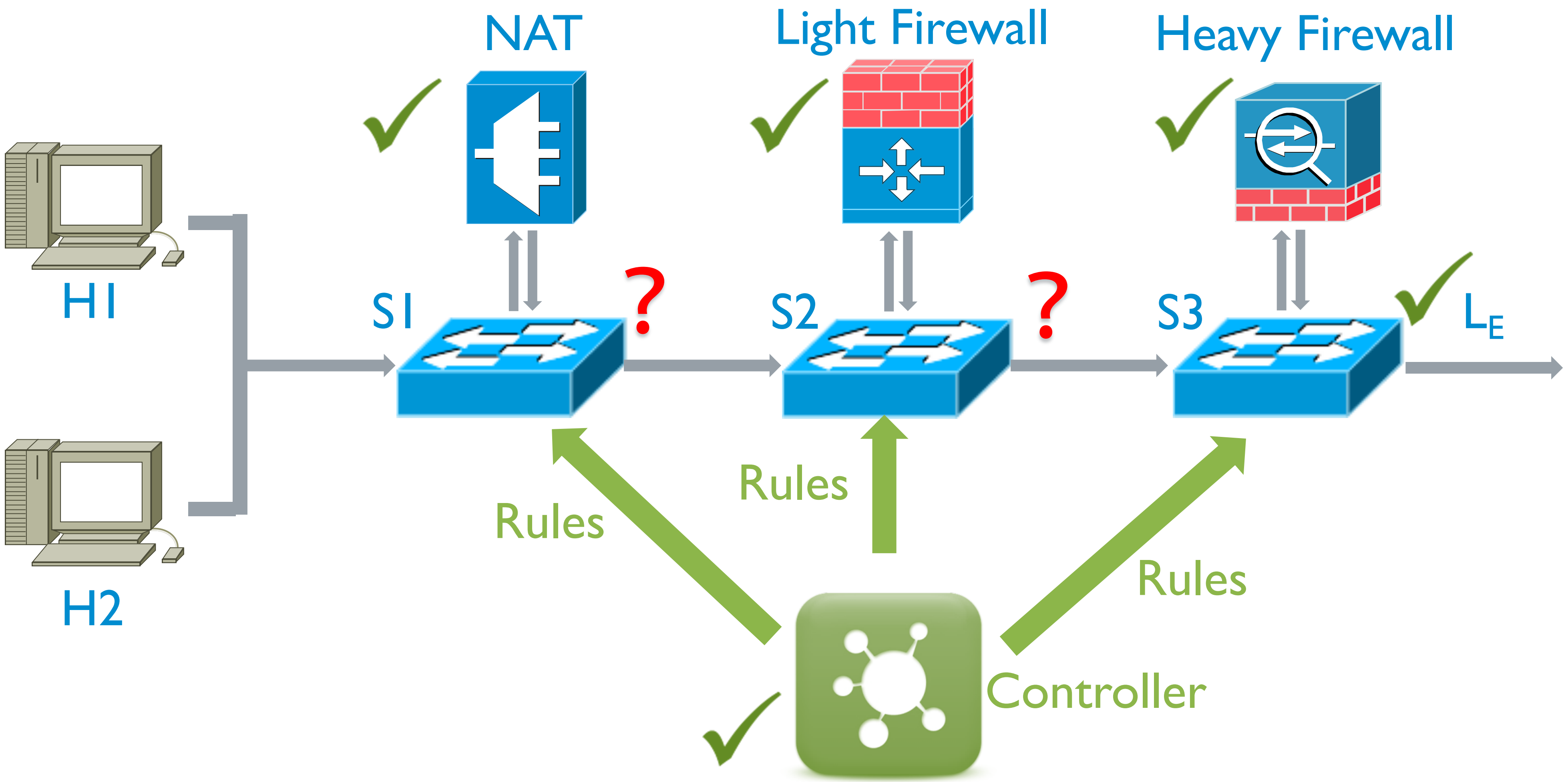
- **Statistics-based Methods**
  - False positive (negative)
  - Waste valuable control channel bandwidth

Middlebox-Bypass Attacks: ~~Existing Secure Methods~~

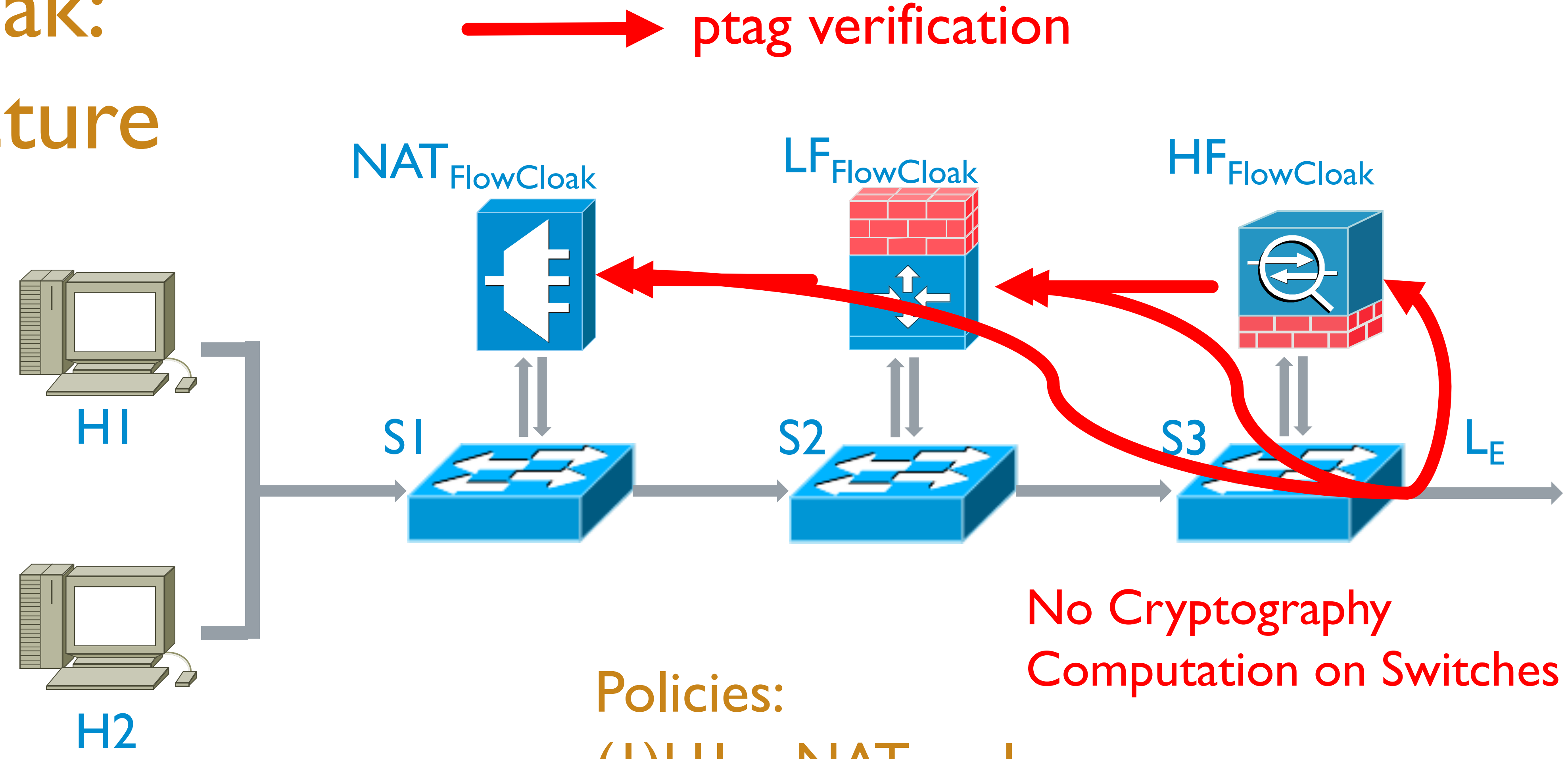
- Probe-based Methods
  - Blinded by coward-attack
  - Waste valuable control channel bandwidth
- Statistics-based Methods
  - False positive (negative)
  - Waste valuable control channel bandwidth

FlowCloak: Defeating Middlebox-Bypass Attacks in Software-Defined Networking

FlowCloak: Model



FlowCloak: Architecture



dtag	ptag

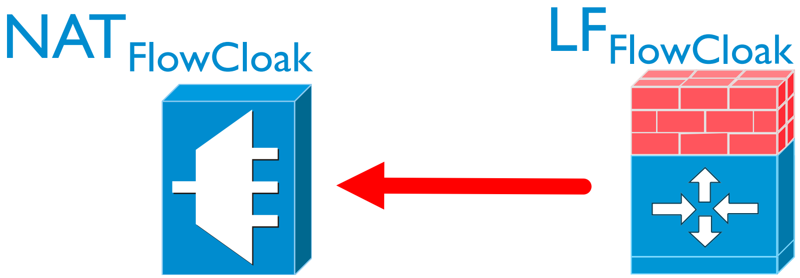
FlowCloak Tag

Policies:

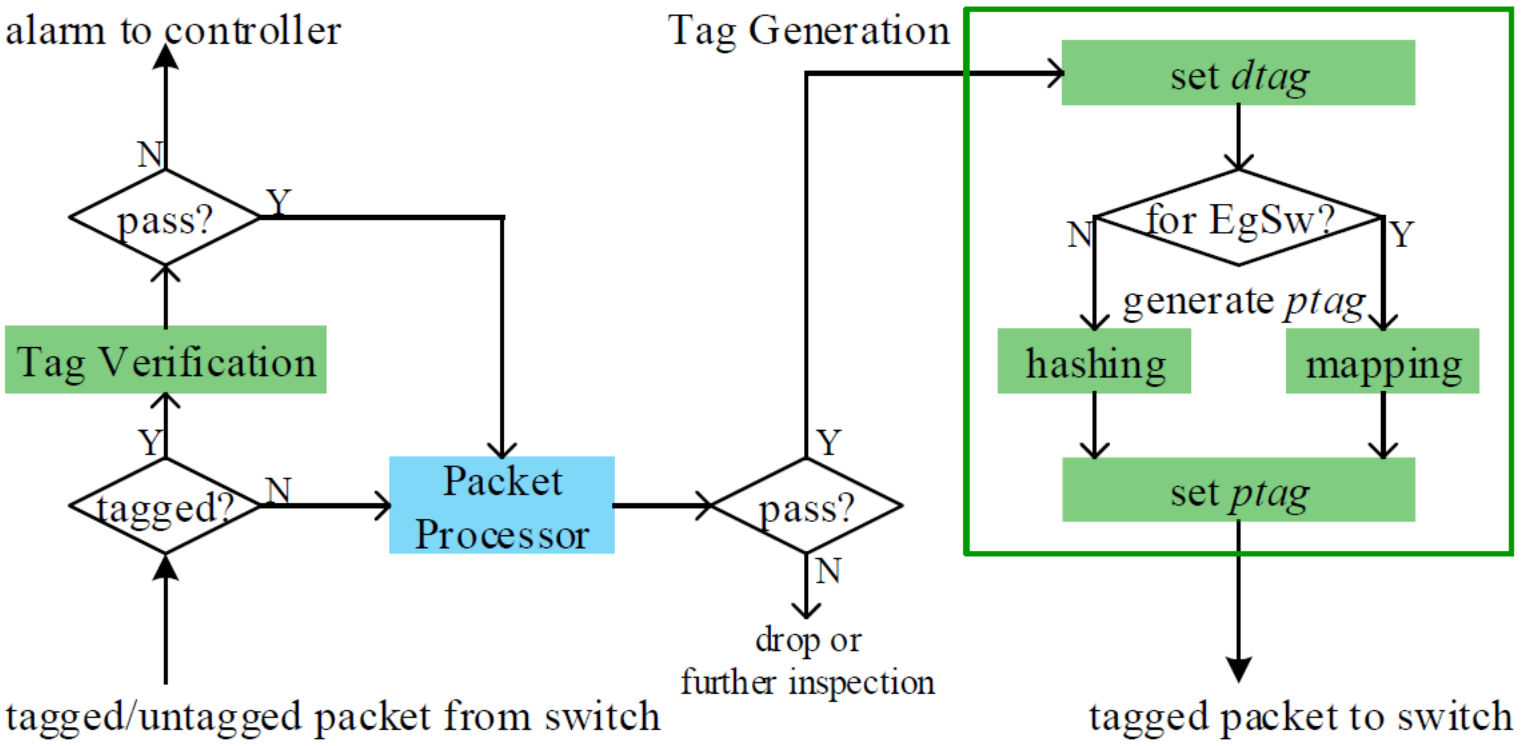
(1)H1—NAT — L_E

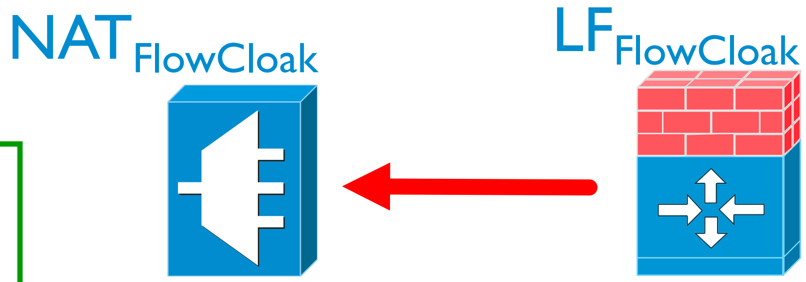
(2)H2 — NAT — LF $\xrightarrow{\text{Alert}}$ HF —L_E

FlowCloak: Architecture



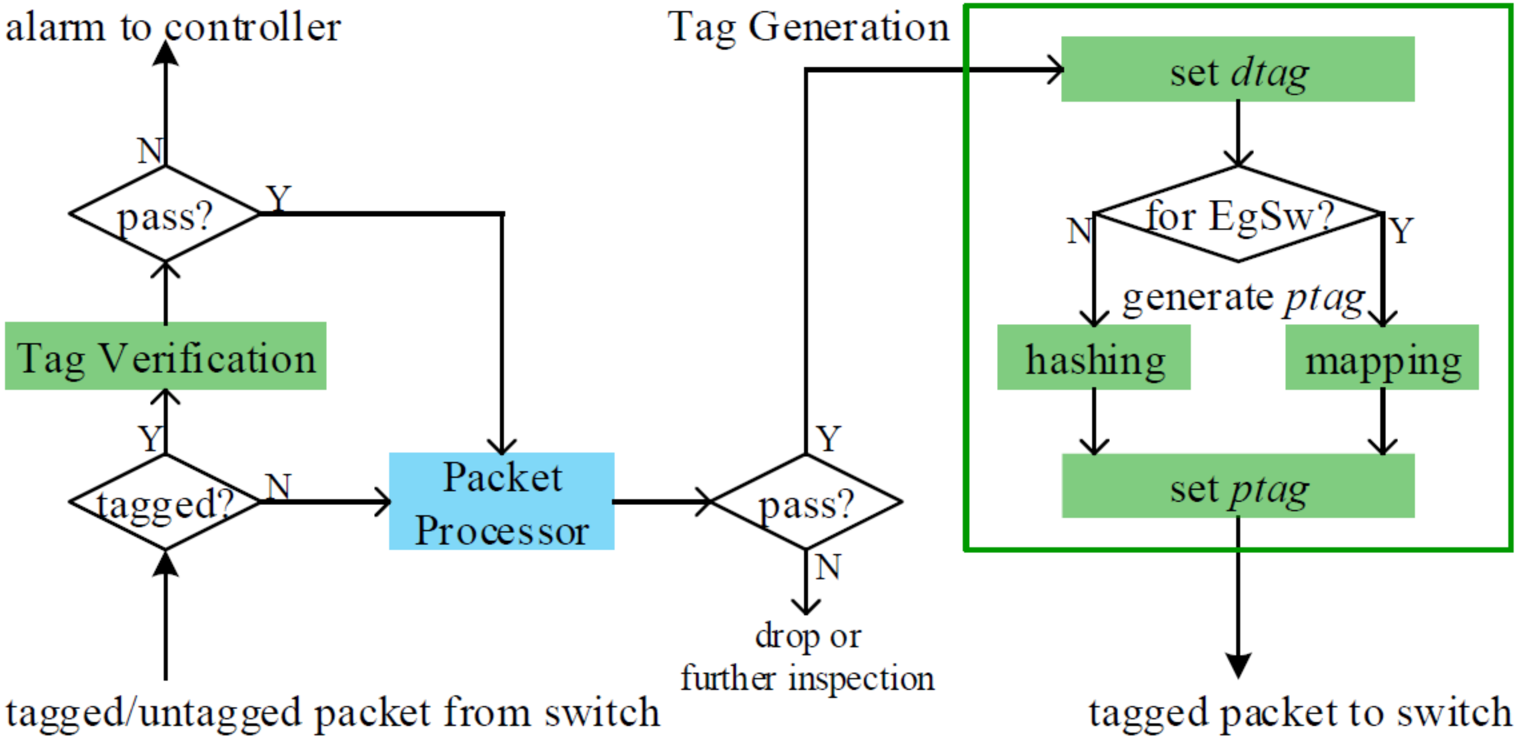
FlowCloak:
Middlebox vs. Middlebox



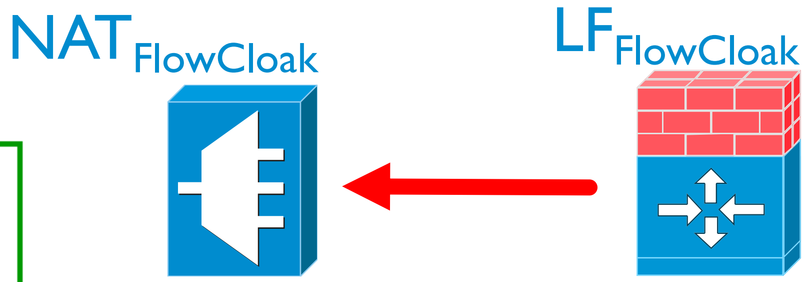


Packet Processing Logic on FC Middleboxes

FlowCloak: Middlebox vs. Middlebox



Packet Processing Logic on FC Middleboxes



TAGVERIFICATION(P)

if isexist(P. dtag, dtagmap) then

ptag’ = Hash(Sample(P. Header))

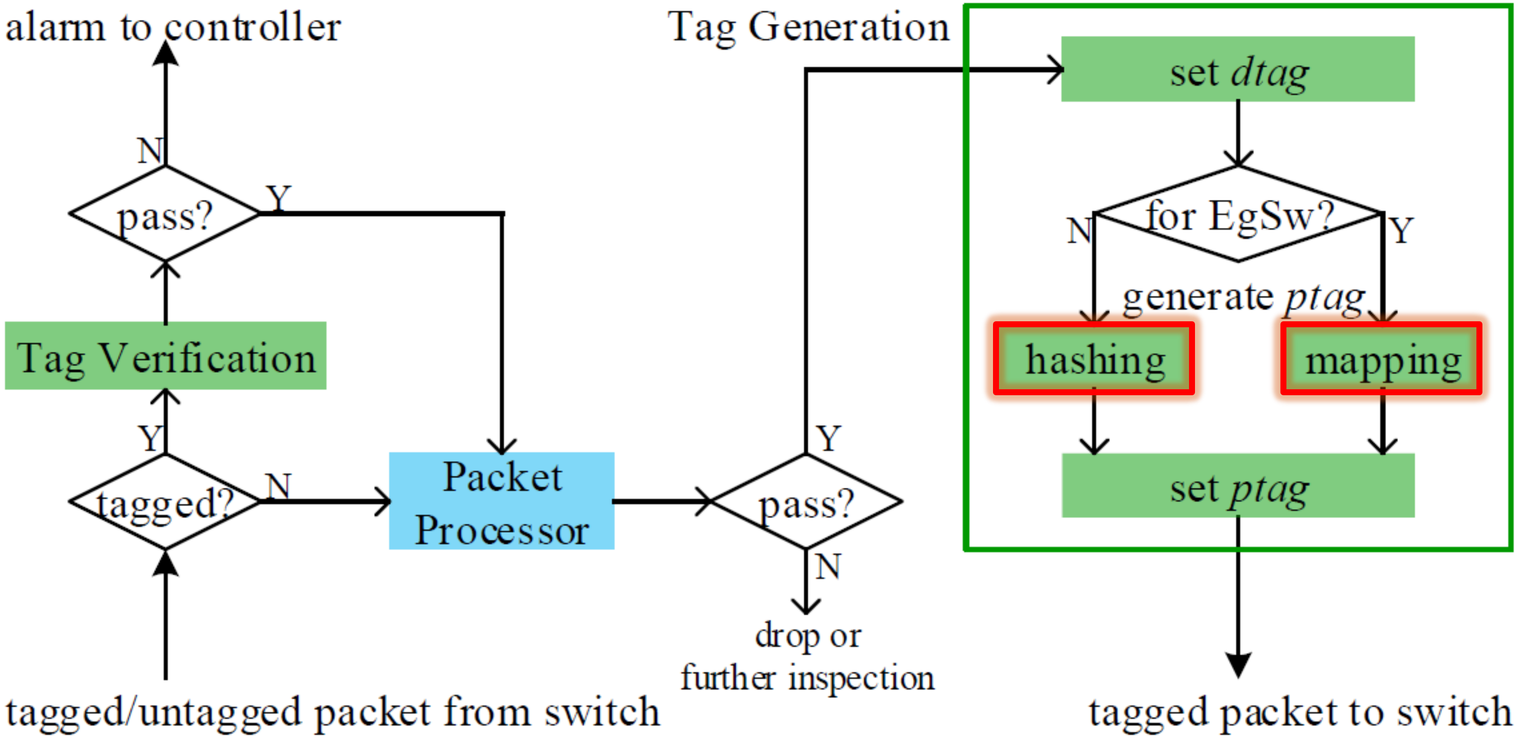
if(ptag’ == P.Header.ptag)

return TRUE

return FALSE

TAGVERIFICATION ends

FlowCloak: Middlebox vs. Middlebox



Packet Processing Logic on FC Middleboxes

TAGGENERATION(P)
  if next_dev(P) == DEV.MIDDLEBOX then
    dtag = flowtags(P, self.ID, Controller)
    writedtag(P, dtag)
    ptag = Hash(Sample(P. Header))
    writeptag(P, ptag)
  else
    ptag = Map(Sample(P. Header))
TAGGENERATION ends

FlowCloak: Middlebox vs. Switch

No cryptography computation: Simulating the hashing function using only match-forward rules

Egress Switch Rules	
Matching	Action
P.SampleDomain=0 && P.Header.ptag=1	forward
P.SampleDomain=1 && P.Header.ptag=0	forward

Hash(b)=~b:
Hash(0)=1
Hash(1)=0

FlowCloak: Middlebox vs. Switch

No cryptography computation: Simulating the hashing function using only match-forward rules

Satisfying Security means Sufficient Rules

Egress Switch Rules	
Matching	Action
P.SampleDomain=0 && P.Header.ptag=1	forward
P.SampleDomain=1 && P.Header.ptag=0	forward

Hash(b)=~b:
Hash(0)=1
Hash(1)=0

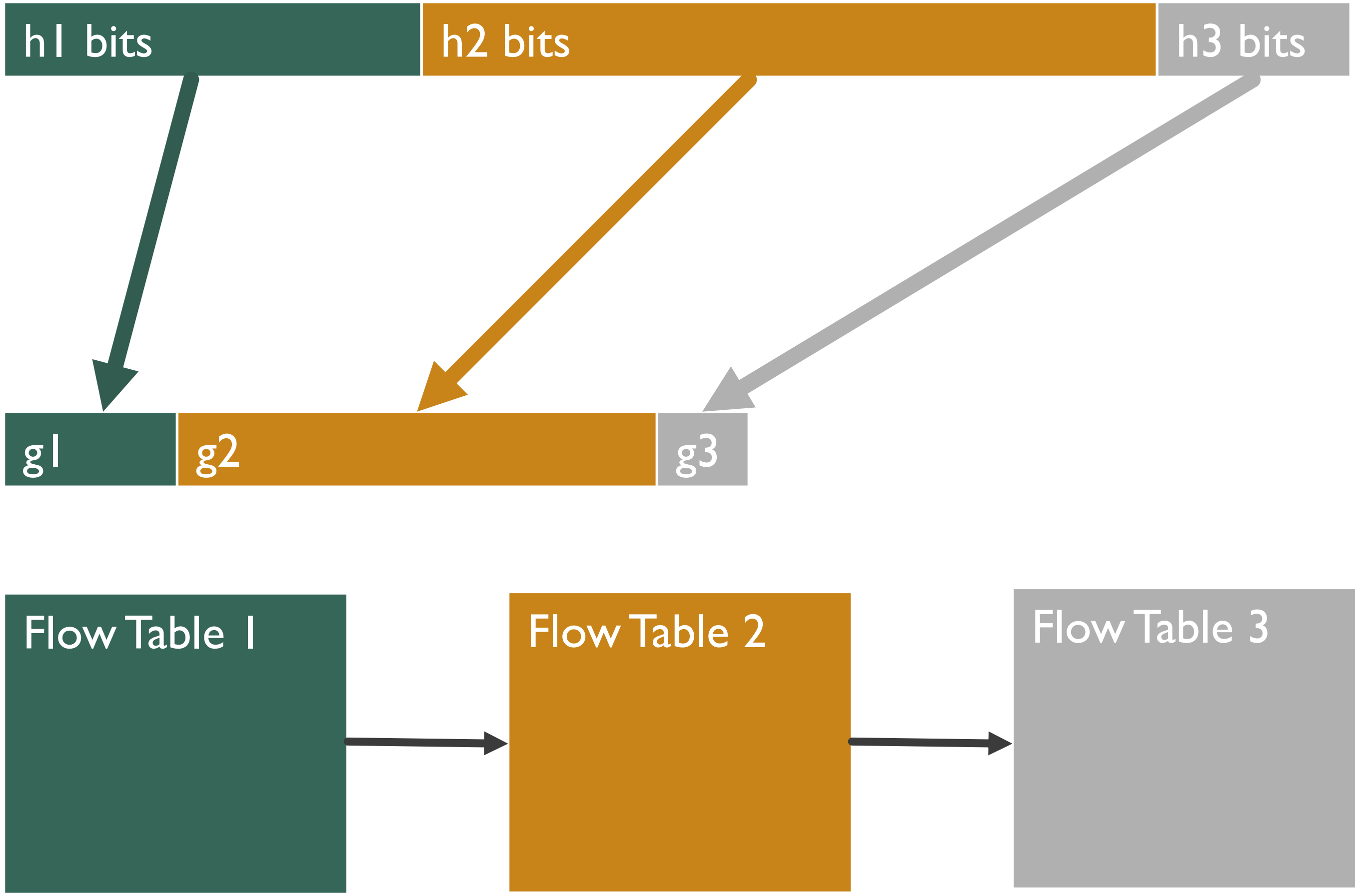
FlowCloak: Middlebox vs. Switch

Length(P.SampleDomain)=1
2 rules;
…
Length(P.SampleDomain)=n
2^n rules;
Too many rules for limited TCAM capacity

Egress Switch Rules	
Matching	Action
P.SampleDomain=0 && P.Header.ptag=1	forward
P.SampleDomain=1 && P.Header.ptag=0	forward

Hash(b)=~b:
Hash(0)=1
Hash(1)=0

FlowCloak: Middlebox vs. Switch

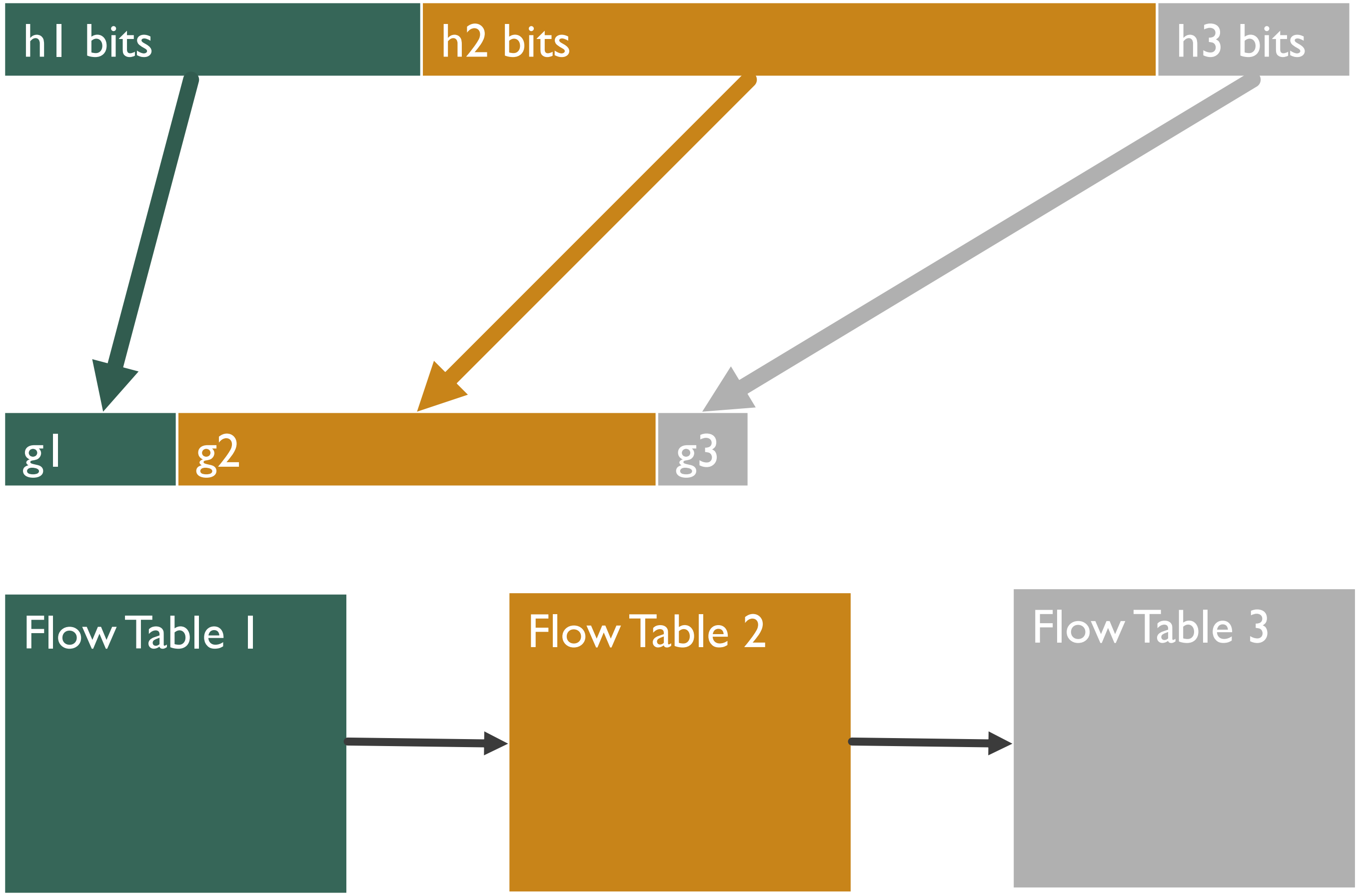


Multi-tag technology

Middlebox Side:
Multi-tag generation based on parallel generation and hashing table.

Switch Side:
Multi-tag verification using only $\sum_{i=1}^{n} 2^{hi}$ rules rather than $\prod_{i=1}^{n} 2^{hi}$ rules

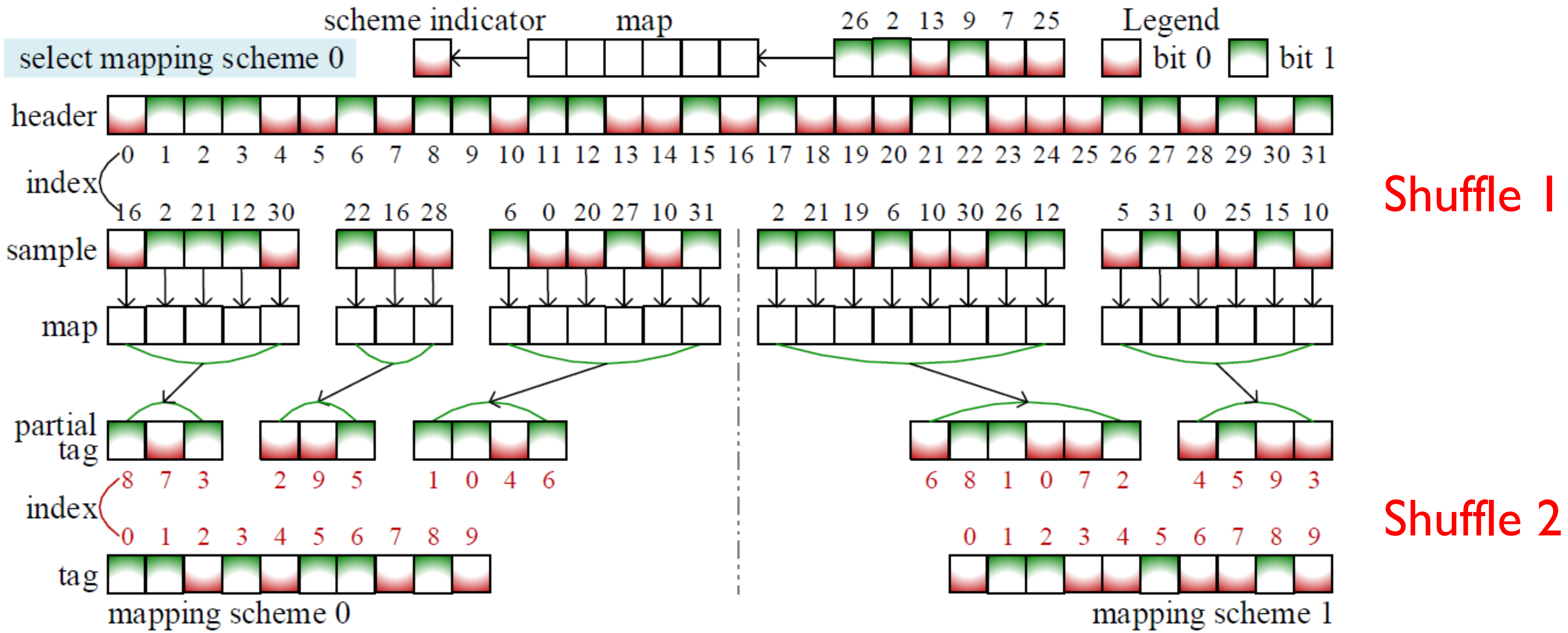
FlowCloak: Middlebox vs. Switch



Caveat:

Each tag becomes shorter →Attacking each part becomes easier?

FlowCloak: Middlebox vs. Switch



More sophisticated mapping:
multiple mapping schemes + nonconsecutive sample bits + double shuffle

Review

- Routing
- Routing Attacks
- Secure Routing
- Secure Forwarding
- Secure SDN Forwarding

?

Readings

- BGP Hijack Explained by Jorge Ribas
- Why Is It Taking So Long to Secure Internet Routing? by Sharon Goldberg
- FlowCloak: Defeating Middlebox-Bypass Attacks in Software-Defined Networking

Thank You