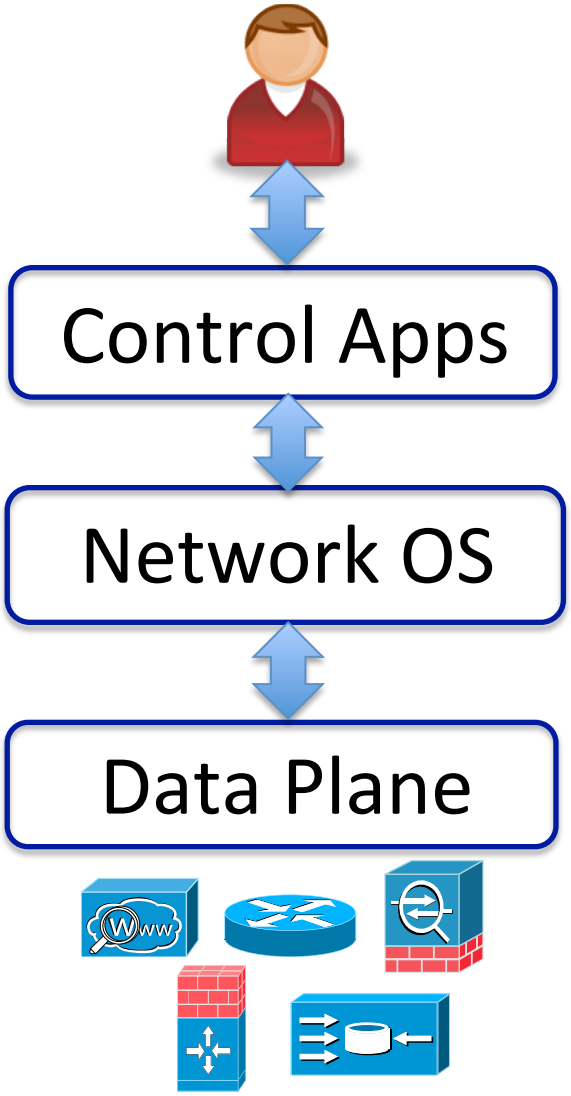
Enforcing Network-Wide Policies in the Presence of Dynamic Middlebox Actions using FlowTags

Seyed K. Fayazbakhsh[*], Luis Chiang[¶], Vyas Sekar[*], Minlan Yu[⋆], Jeffrey Mogul[◆]

[*]*CMU,* [¶]*Deutsche Telekom,* [⋆]*USC,* [◆]*Google*

Middleboxes complicate policy enforcement in SDN

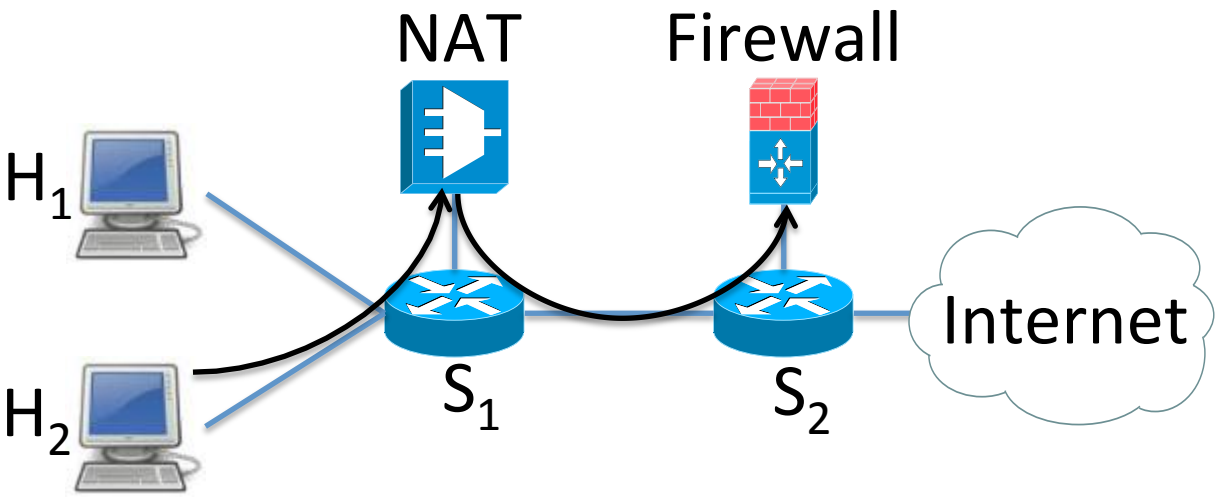
Policy:
E.g., service chaining, access control



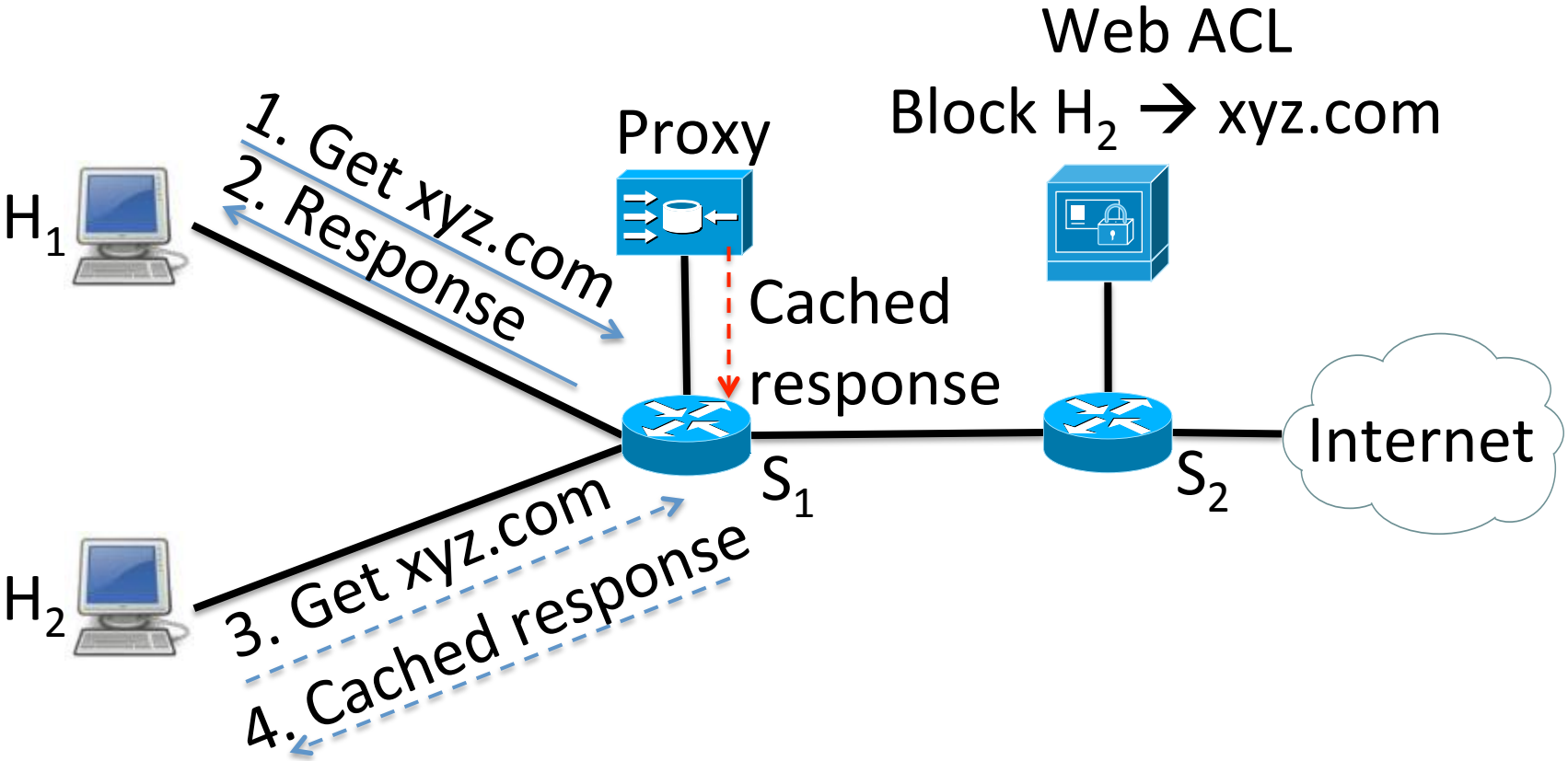
Dynamic and traffic-dependent modifications!
e.g., NATs, proxies

Modifications → Attribution is hard

Block the access of H_2 to certain websites.



Dynamic actions → Policy violations



Our work: FlowTags

Some candidate (non-)solutions:
Placement, tunneling, consolidation, correlation

Address some symptoms but not root cause
→ OriginBinding and PathsFollowPolicy violations

FlowTags provides an architectural solution:
→ Enables policy enforcement and diagnosis despite dynamic middlebox actions.

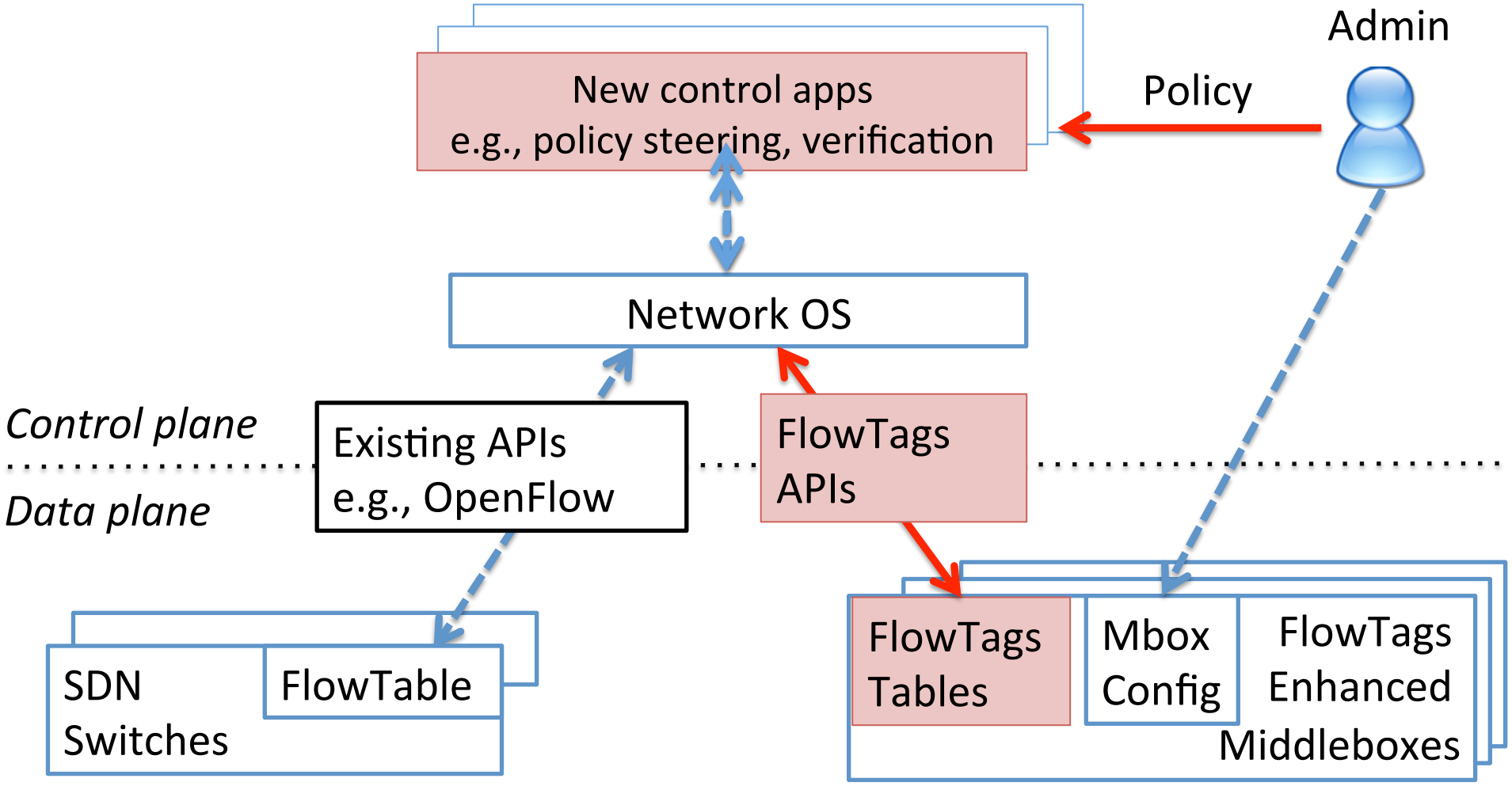
Outline

- Motivation
- High-level Idea
- FlowTags Design
- Evaluation

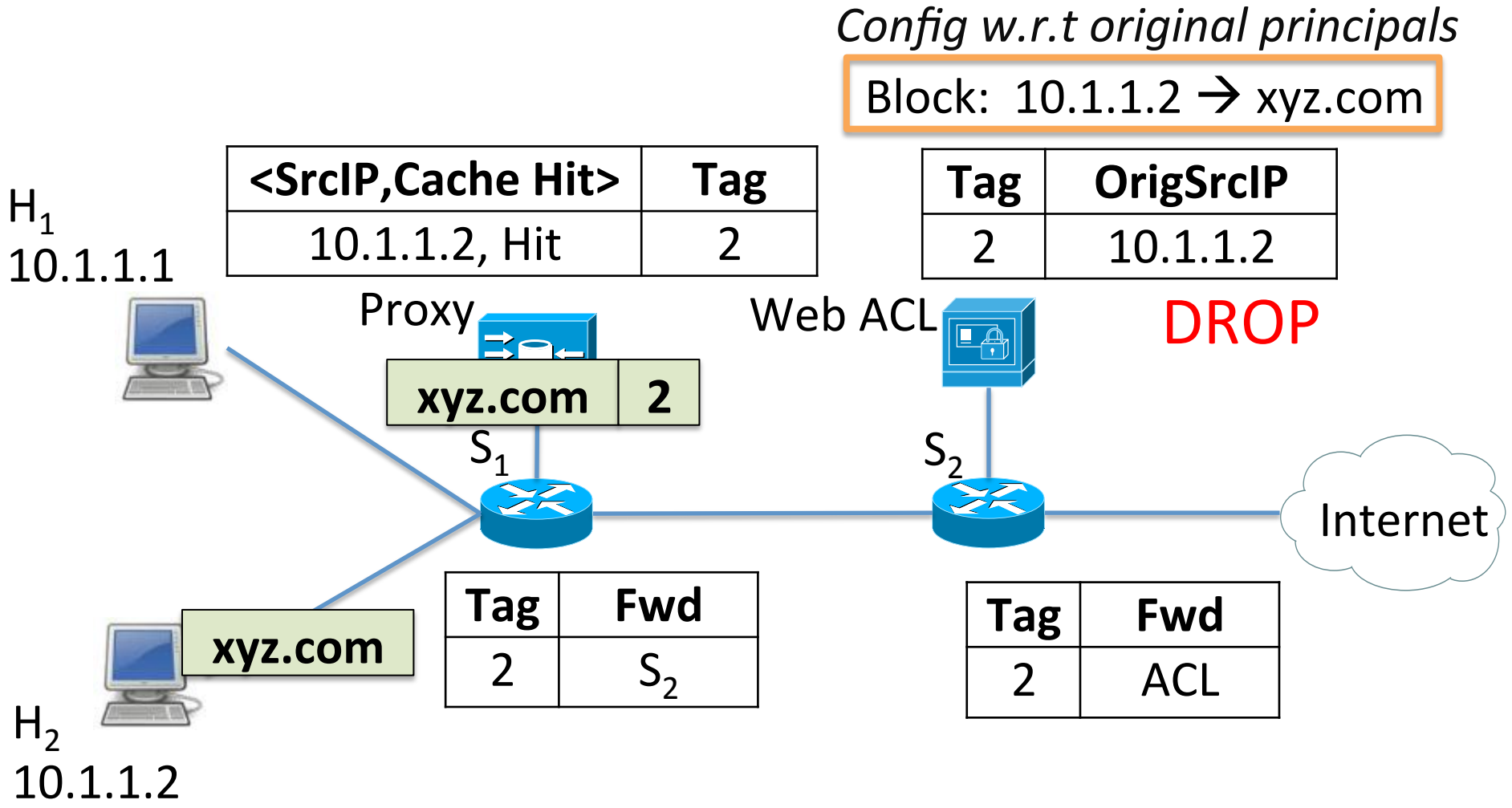
High-level idea

- Middleboxes need to restore SDN tenets
  - Possibly only option for correctness
  - Minimal changes to middleboxes
- Add missing contextual information as Tags
  - NAT gives IP mappings,
  - Proxy provides cache hit/miss info
- FlowTags controller configures tagging logic

FlowTags architecture



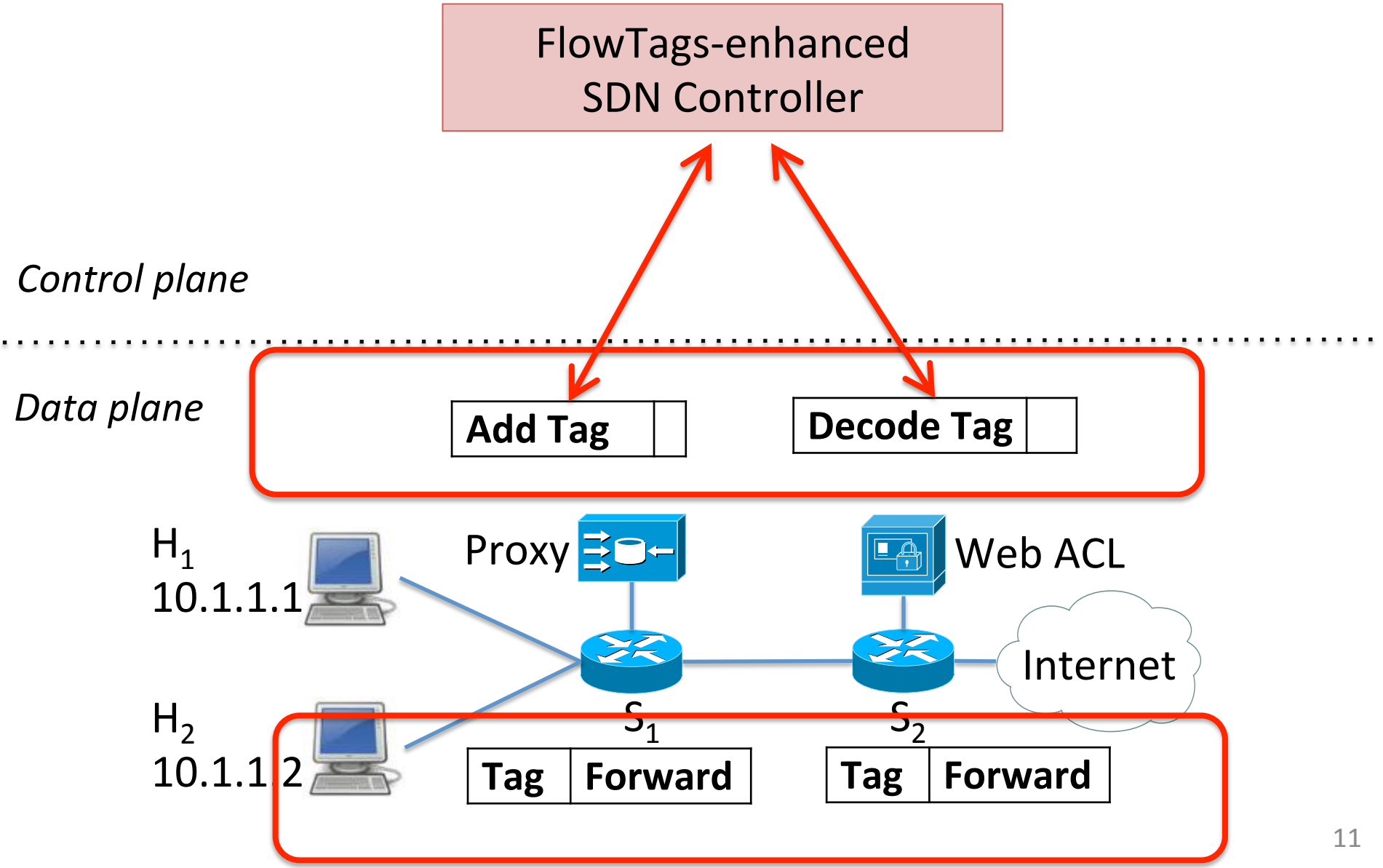
FlowTags in action



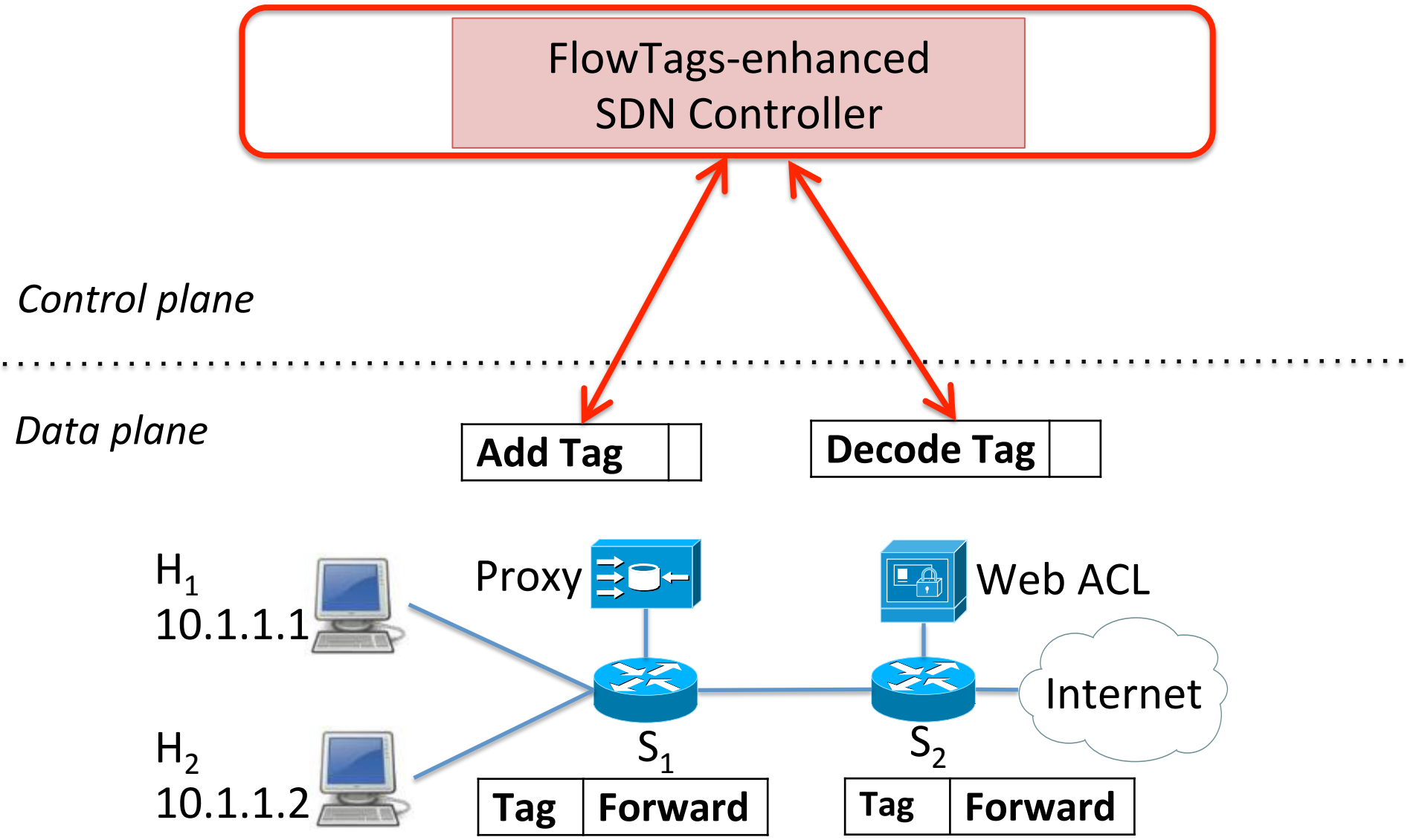
Outline

- Motivation
- High-level Idea of FlowTags
- FlowTags Design
- Evaluation

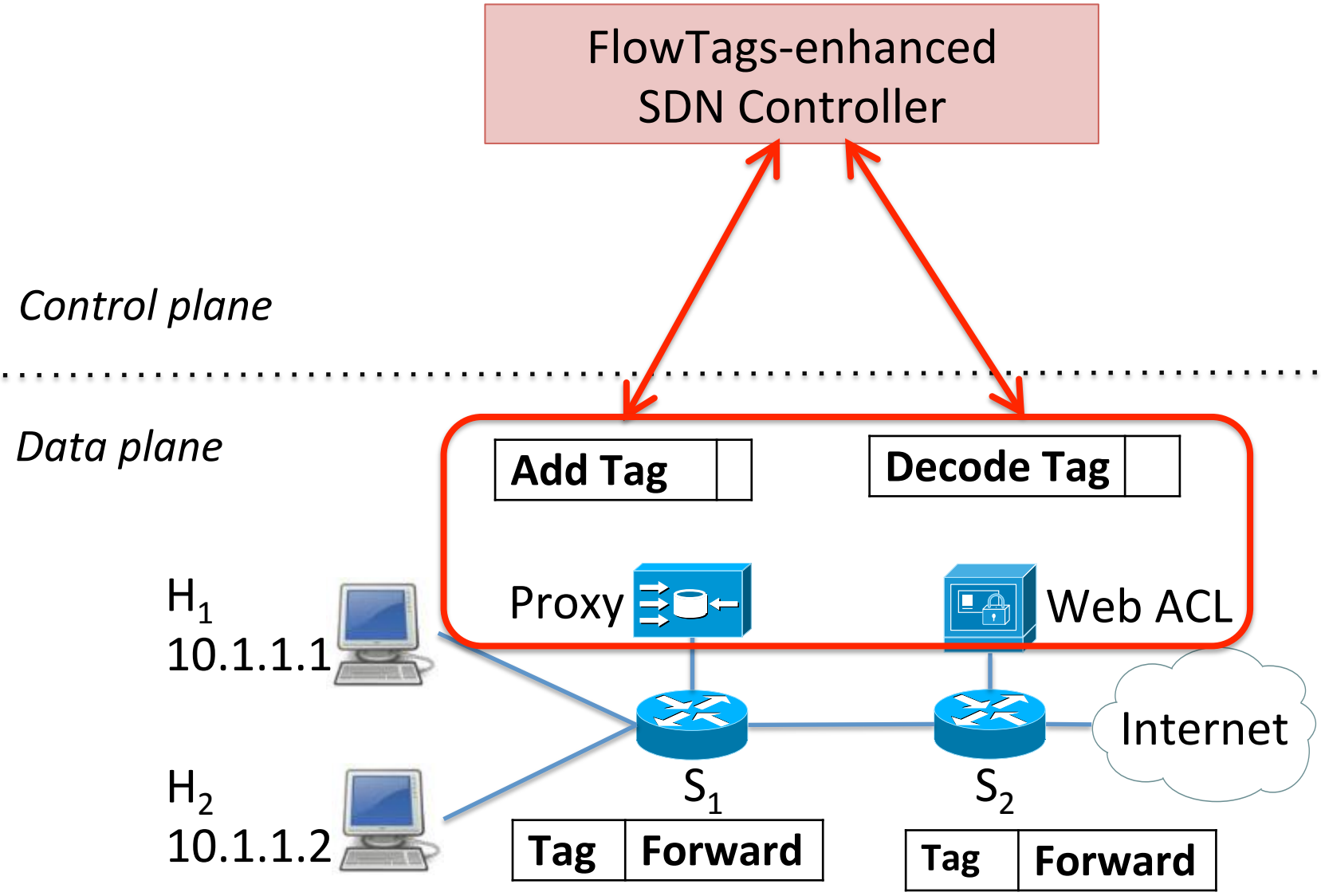
Challenge 1: Tag Semantics



Challenge 2: New APIs, control apps



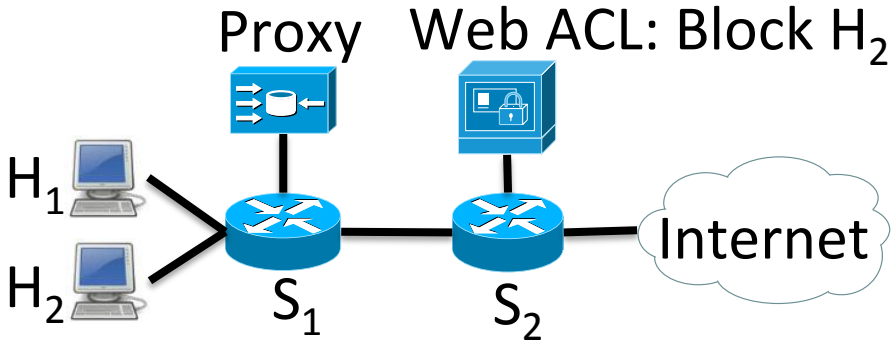
Challenge 3: Middlebox Extensions

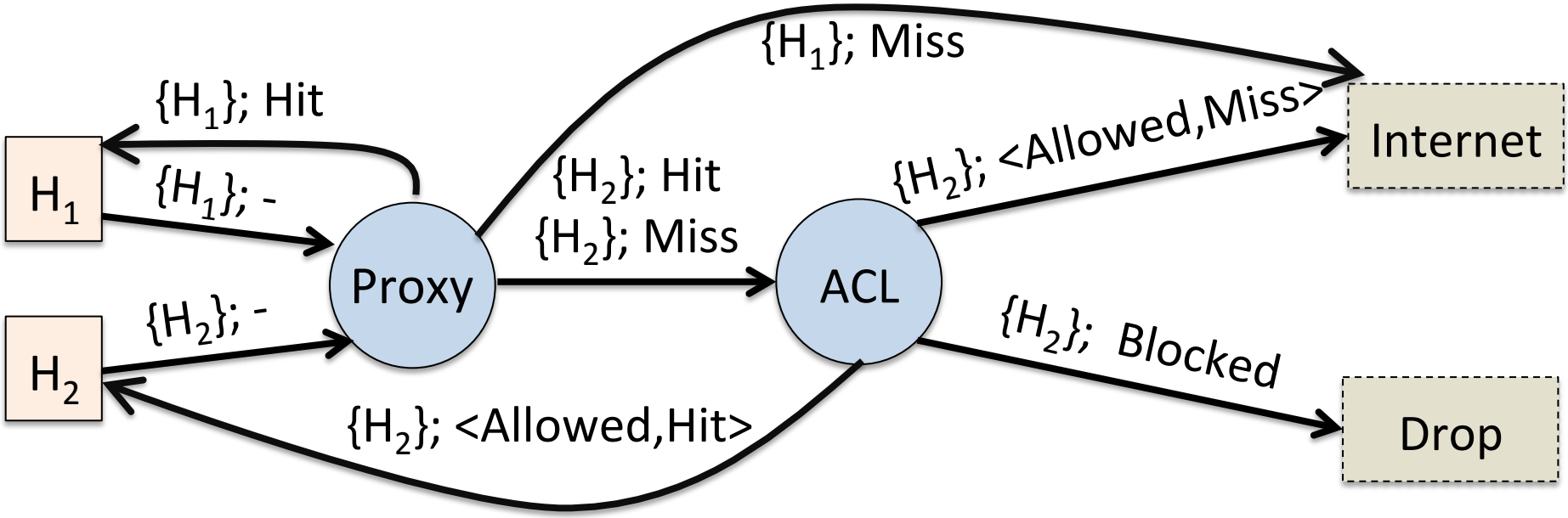


Outline

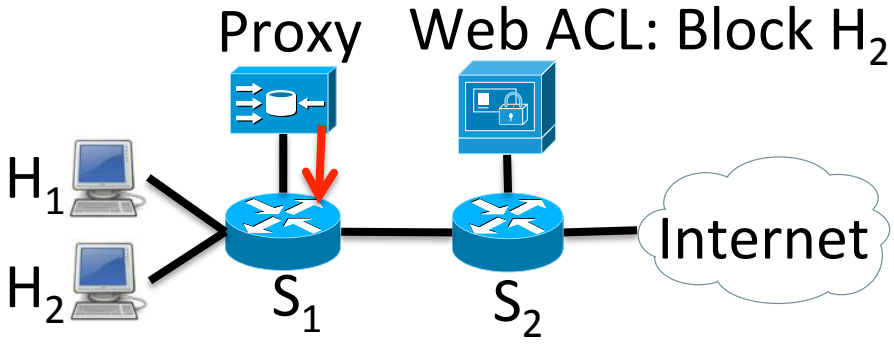
- Motivation
- High-level Idea of FlowTags
- FlowTags Design
  - Tag semantics
  - Controller and APIs
  - Middlebox modification
- Evaluation

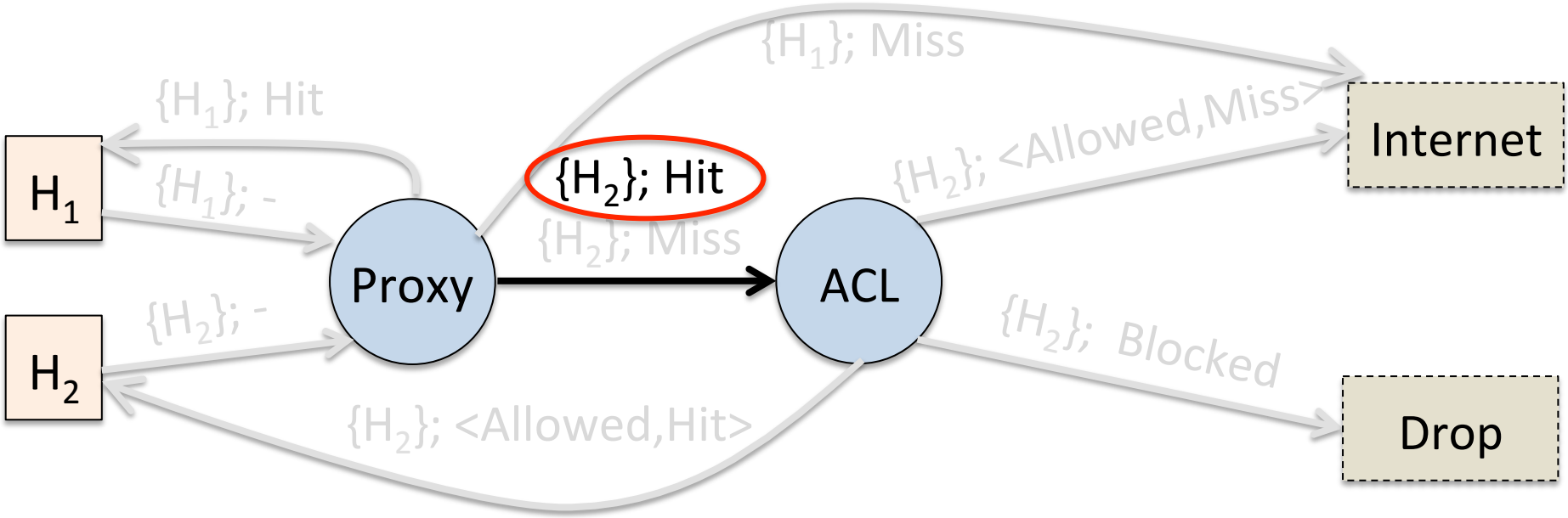
Semantics: Dynamic Policy Graph (DPG)





Semantics: Dynamic Policy Graph (DPG)



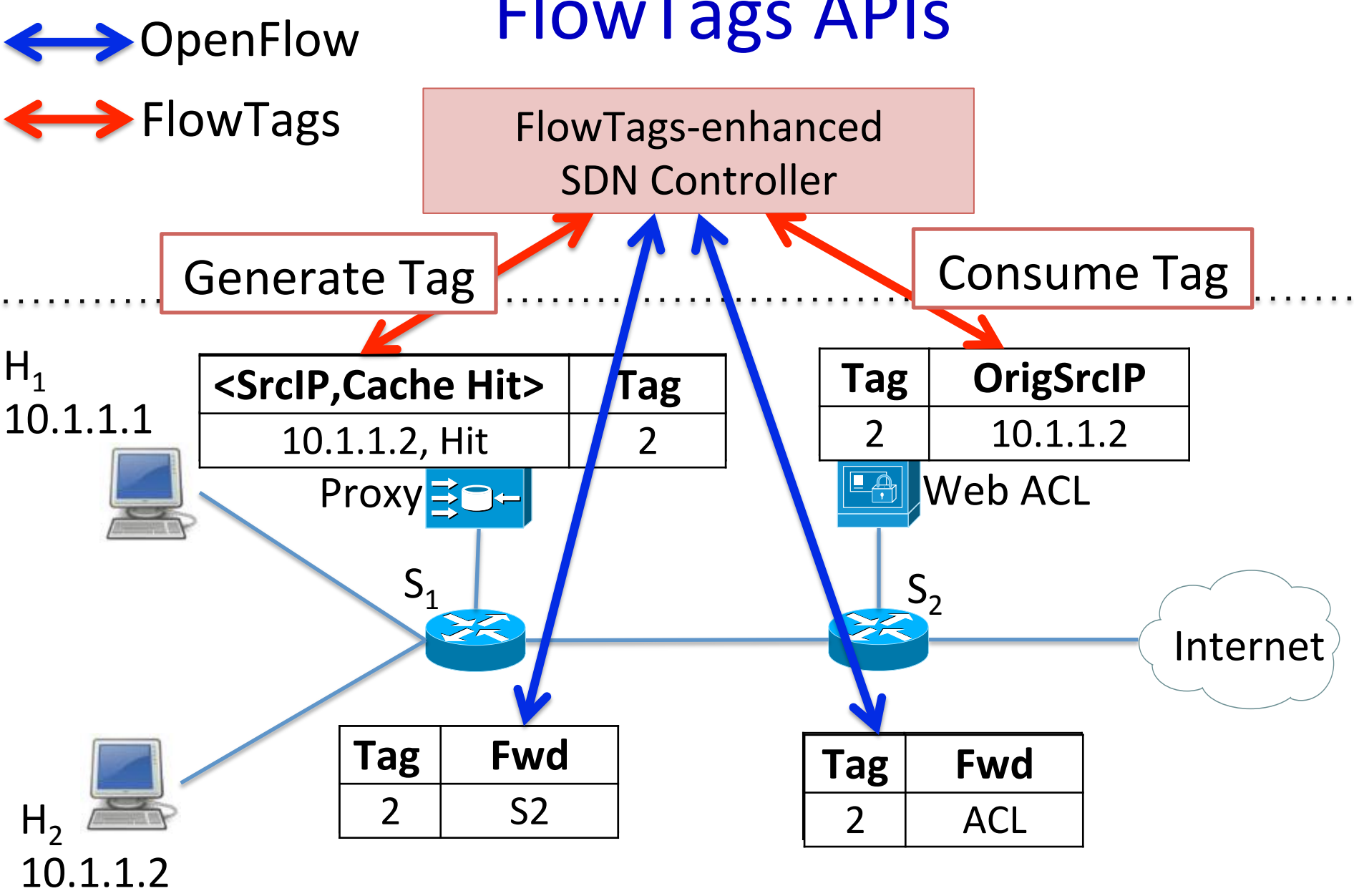


Intuitively, need a Tag <per flow, per-edge> in DPG

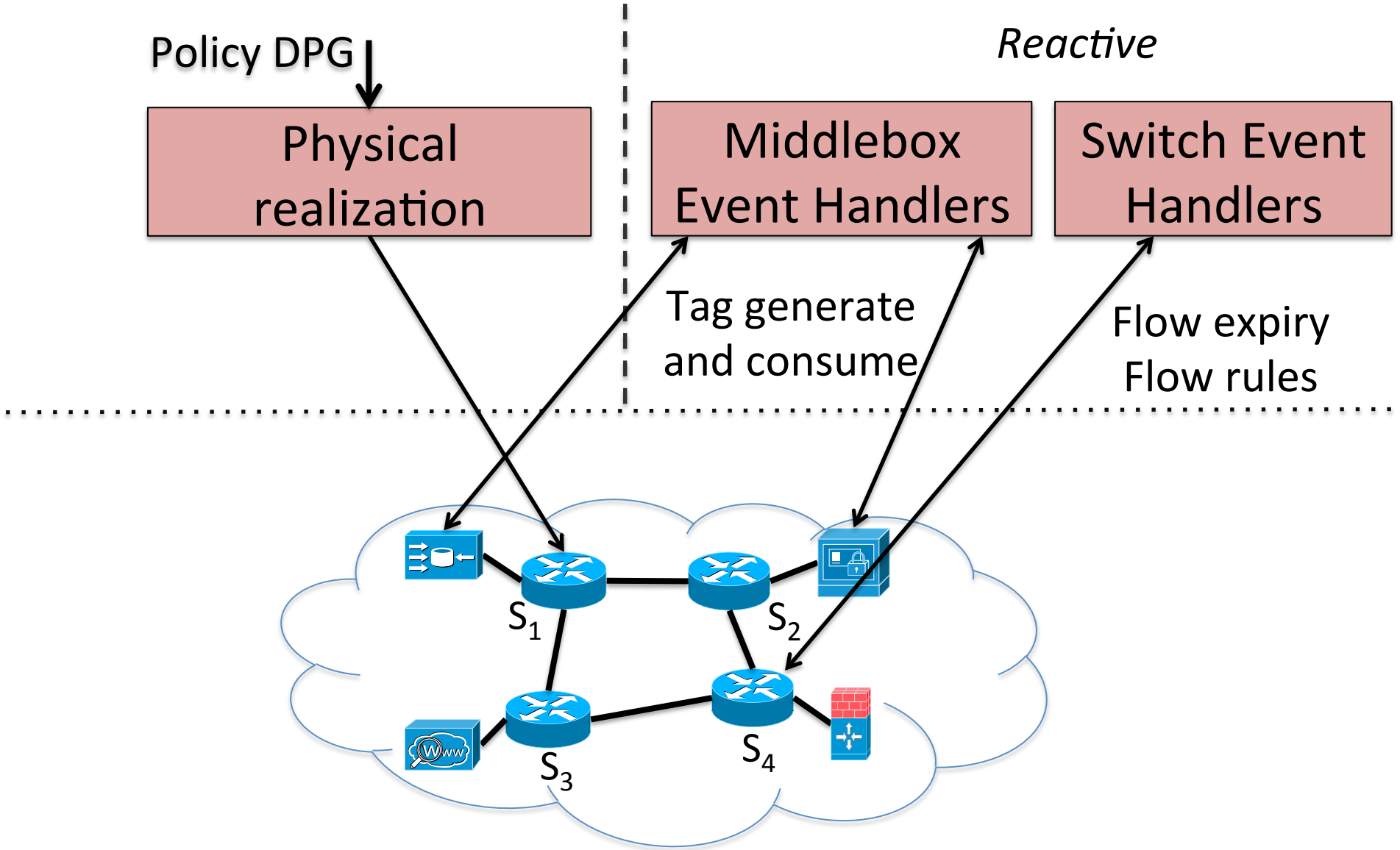
Outline

- Motivation
- High-level Idea of FlowTags
- FlowTags Design
  - Tag semantics
  - Controller and APIs
  - Middlebox modification
- Evaluation

FlowTags APIs



FlowTags-enhanced controller

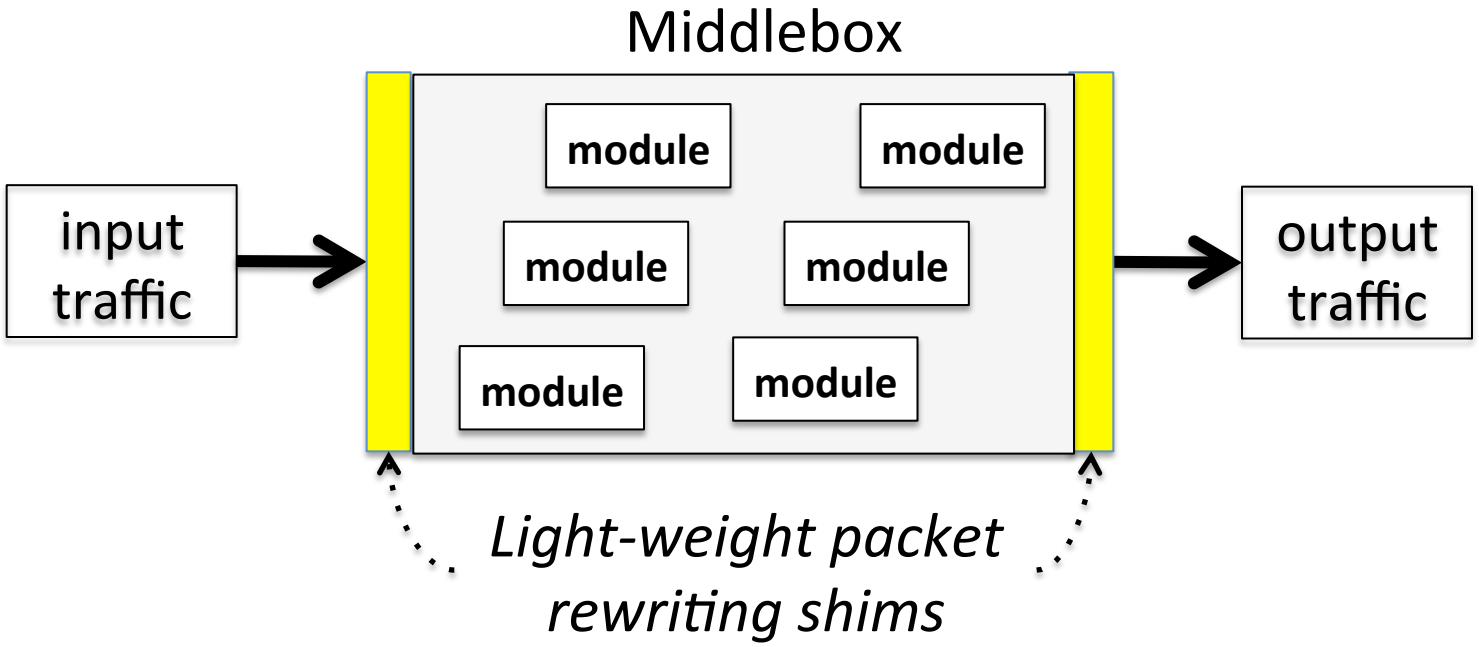


Outline

- Motivation
- High-level Idea of FlowTags
- FlowTags Design
  - Tag semantics
  - Controller and APIs
  - Middlebox modification
- Evaluation

Middlebox extension strategies to add FlowTags support

Strategy 1: Packet Rewriting

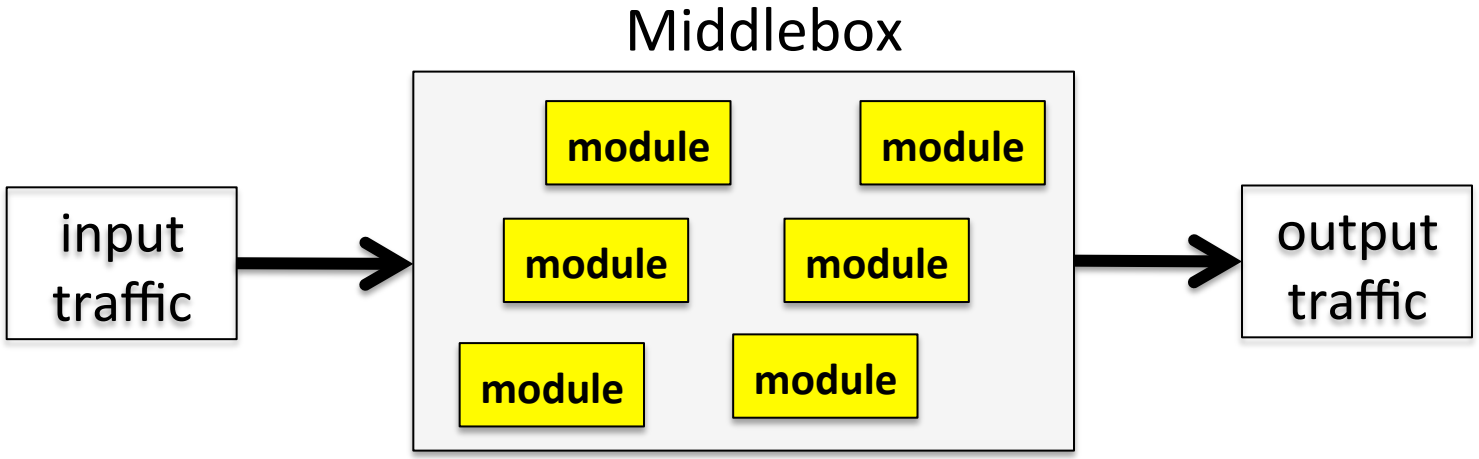


Pro: One shot

Con: Hard to get internal context

Middlebox extension strategies to add FlowTags support

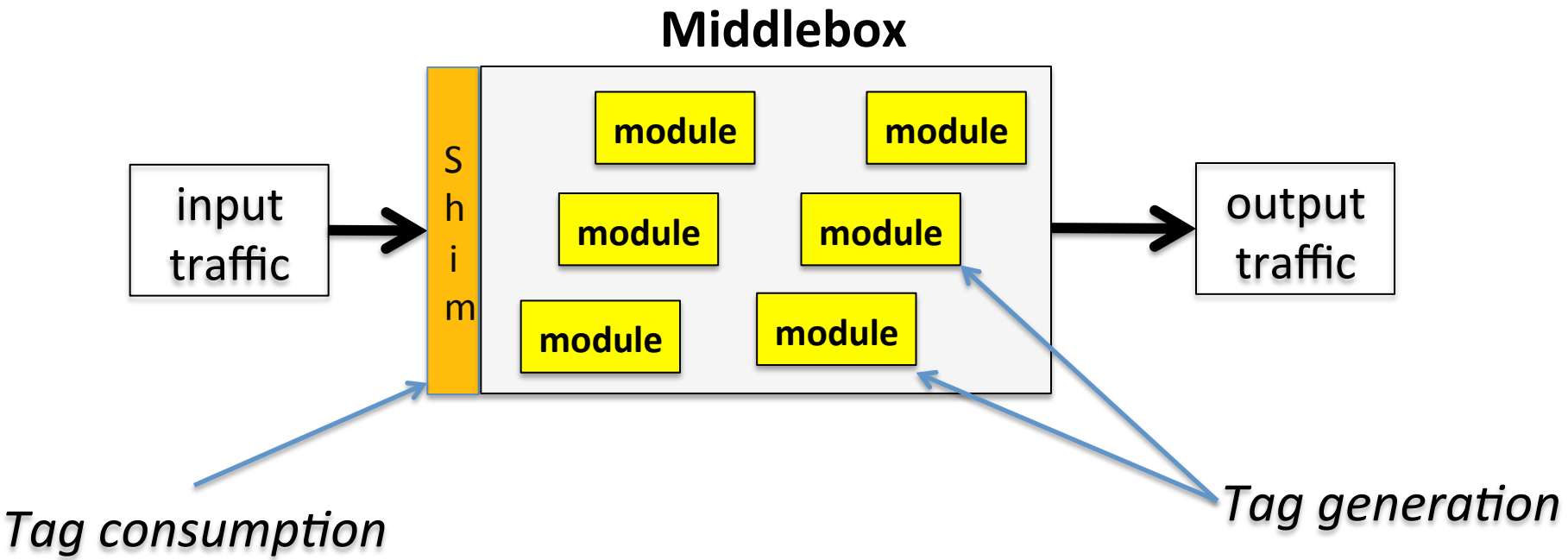
Strategy 2: Module Modification



Pro: More change is needed

Con: Suited for getting internal context

Middlebox extension strategies to add FlowTags support



Our Strategy:

Packet rewriting for Tag consumption

Module modification for Tag generation

Outline

- Motivation
- High-level Idea of FlowTags
- FlowTags Design
- Evaluation

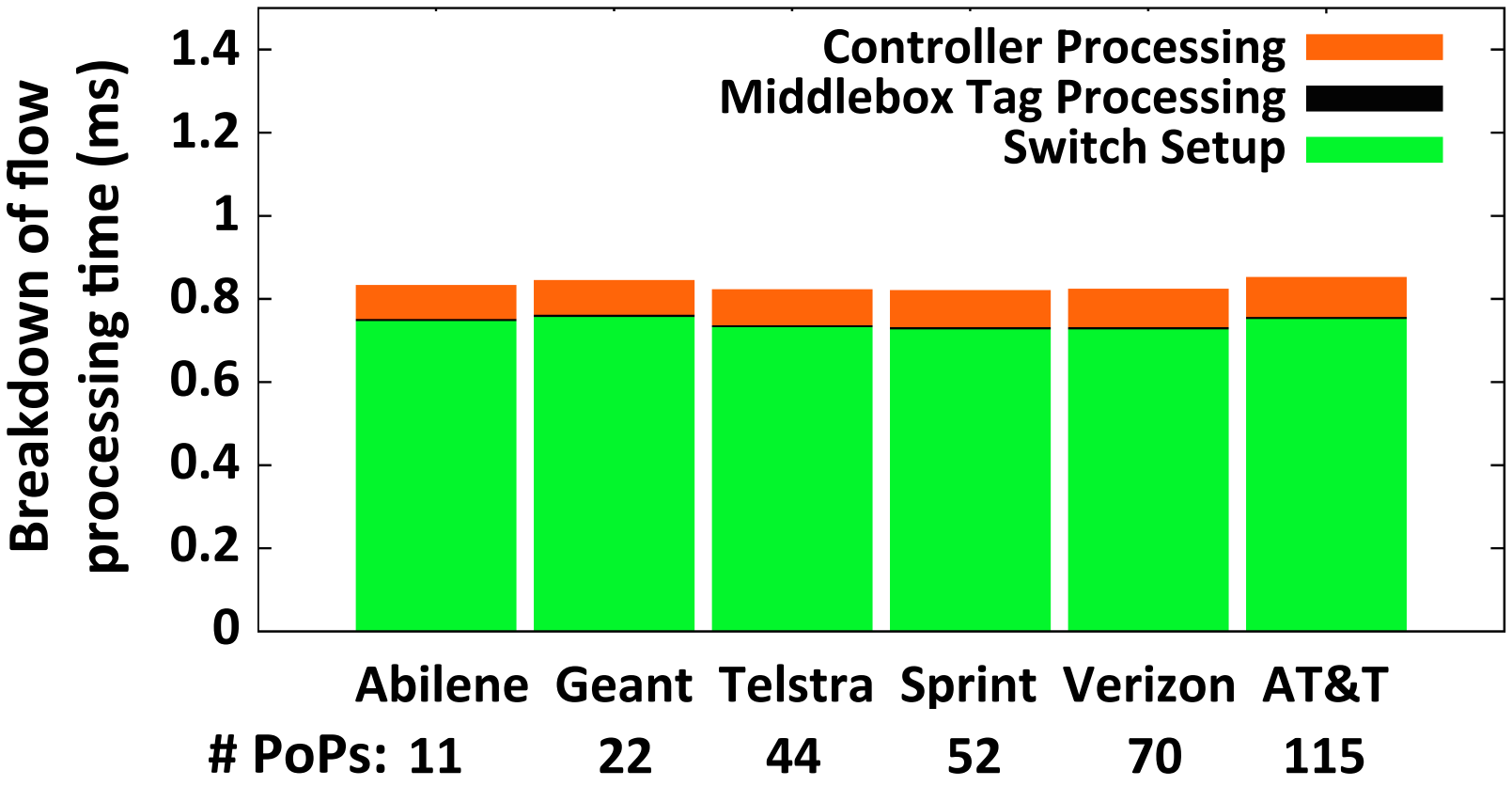
Key evaluation questions

- Feasibility of middlebox modification
- FlowTags overhead
- Number of Tag bits
- New capabilities

FlowTags needs minimal middlebox modifications

Middlebox	Total LOC	Modified LOC
Squid	216,000	75
Snort	336,000	45
Balance	2,000	60
iptables	42,000	55
PRADS	15,000	25

FlowTags adds low overhead



Summary of other results

- Adds < 1% overhead to middlebox processing
- Tags can be encoded in ~ 15 bits
  - E.g., IP-ID, IPv6 FlowLabel, EncapHeaders (NVP)
- Can enable new capabilities
  - Extended header space analysis
  - Diagnosing network bottlenecks

Conclusions

- Middleboxes complicate enforcement
  - E.g., NAT/LB rewrite headers, proxy sends cached response
- Root cause: Violation of the SDN tenets
  - Origin Binding and Paths-Follow-Policy
- FlowTags extends SDN with new middlebox APIs
  - Restores tenets using new DPG abstraction
  - No changes to switches and switch APIs
- FlowTags is practical
  - Minimal middlebox changes, low overhead
  - An enabler for verification, testing, and diagnosis