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

Seyed Kaveh Fayazbakhsh, Carnegie Mellon University;
Luis Chiang, Deutsche Telekom Labs;
Vyas Sekar, Carnegie Mellon University;
Minlan Yu, University of Southern California;
Jeffrey C. Mogul, Google

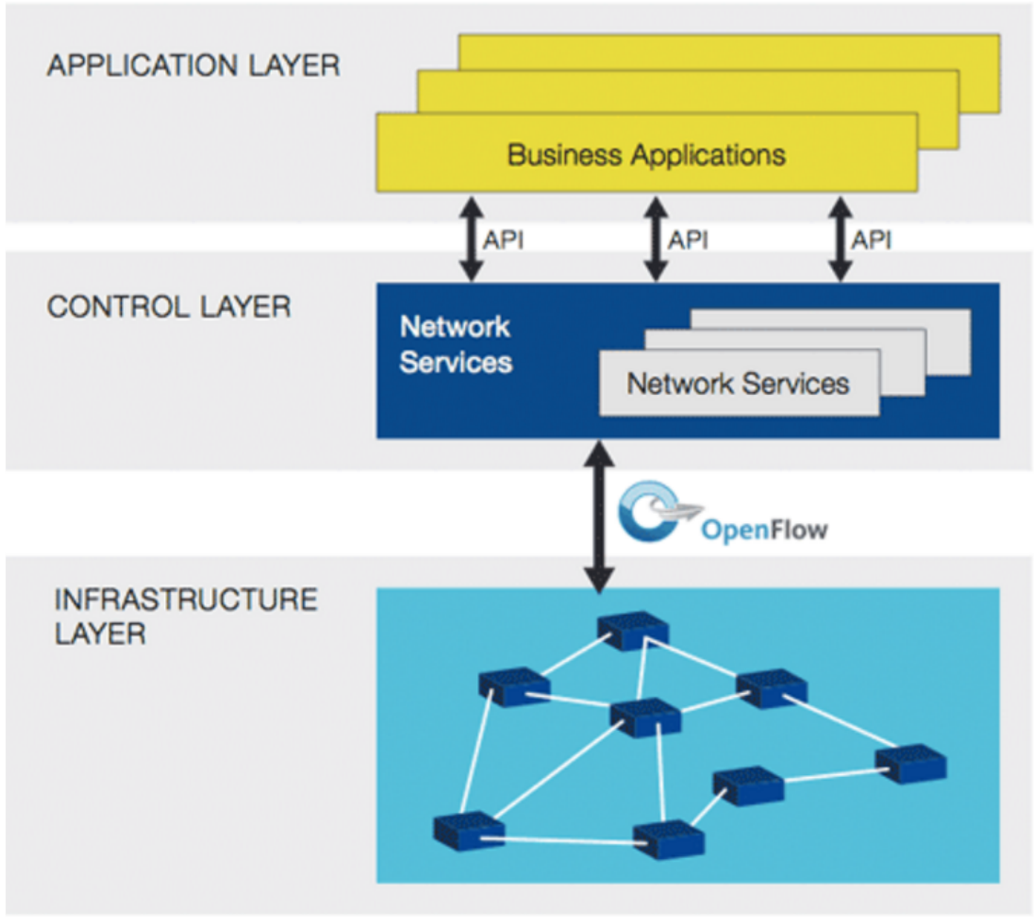
Presented by: Harsimran Pabla

Motivation

- Middle boxes complicate policy enforcement in SDN

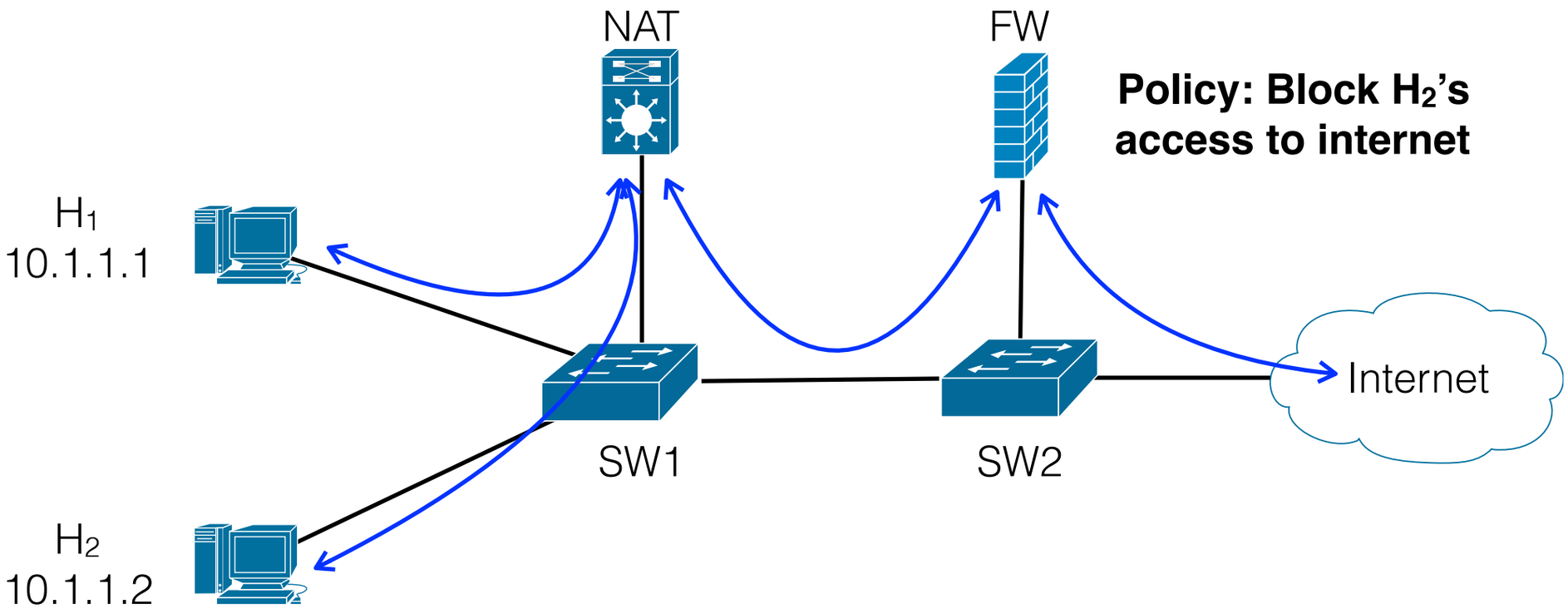
  Policy: E.g., service chaining, access control

MiddleBoxes perform dynamic and traffic-dependent packet modifications!



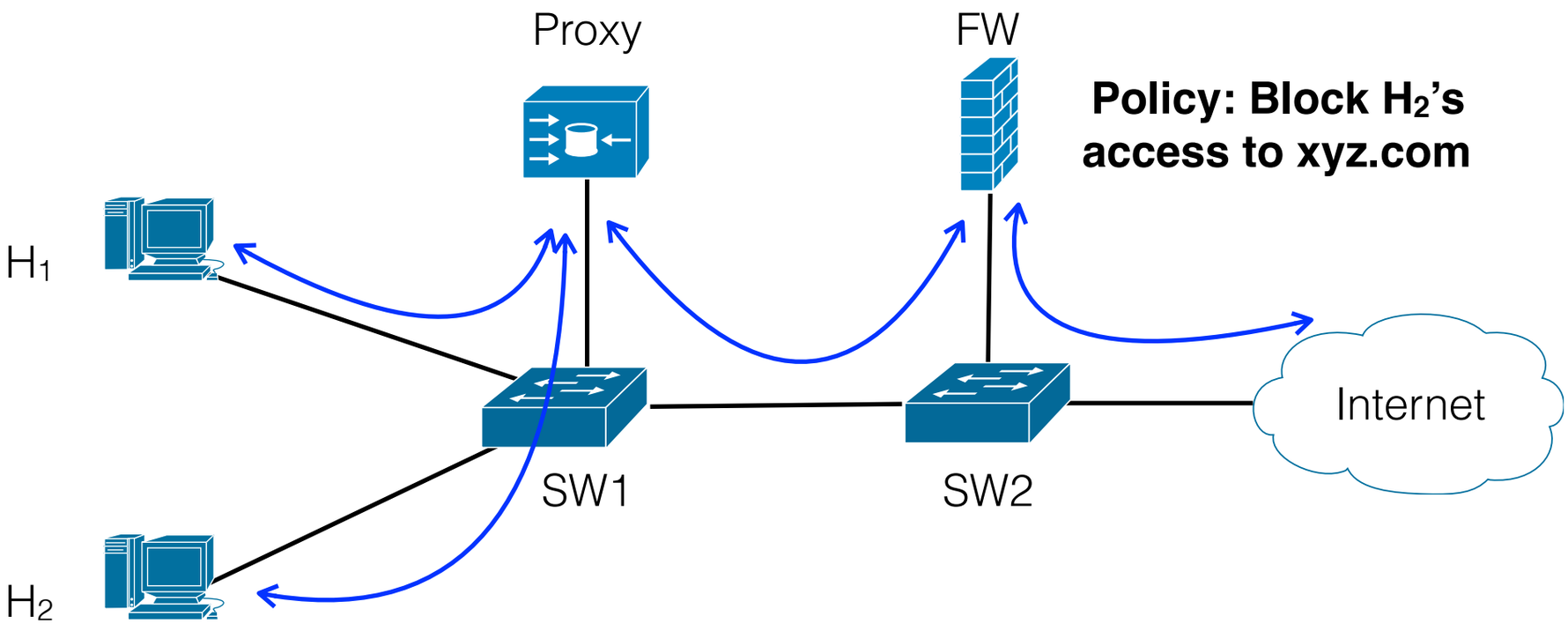
Source: https://www.opennetworking.org/sdn-resources/sdn-definition

Motivation



Modifications by NAT makes attribution difficult

Motivation



H_1 GET xyz.com
H_1 gets the remote response, it is cached in the Proxy

H2 GET xyz.com
H2 gets the cached response from the proxy, violating the policy.

Dynamic behaviour of Proxy violates the policy

Motivation

- Some Candidate solution : Placement, Tunneling, Consolidation, Correlation
  - They address symptoms not the root cause.

- Root cause of this problem, middleboxes violate following SDN principles of
  - OriginBinding
  - PathFollowPolicy

Outline

- Motivation
- **High Level Idea**
- FlowTag Design
- Evaluation
- Conclusion

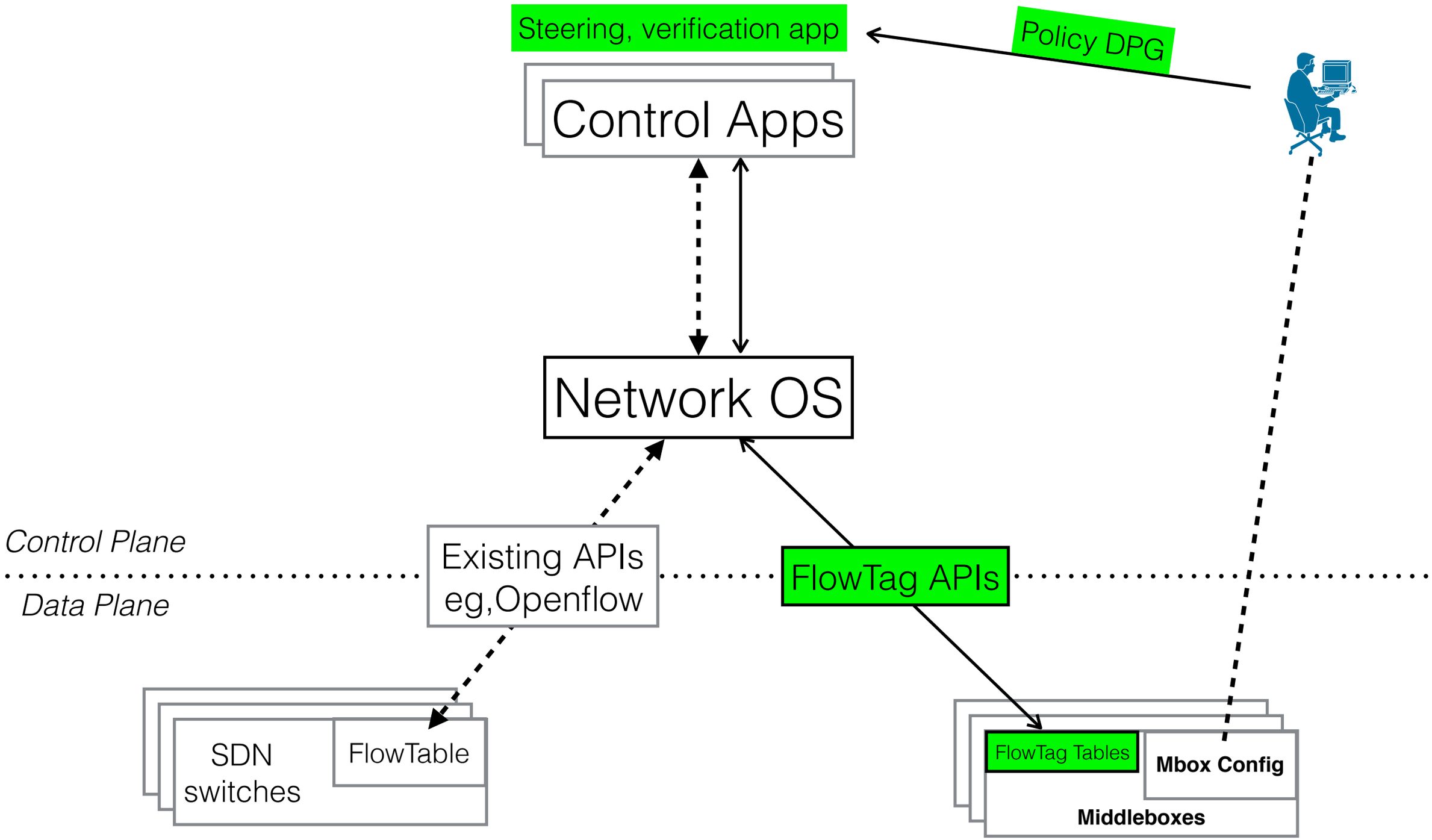
High Level Design

- Middleboxes need to restore SDN principles of OriginBinding and PathFollowPolicy.
  - Possibly only option for correctness
  - Minimal changes to middleboxes.

- Add missing contextual information as TAGS
  - For eg: NAT gives IP mapping or Proxy provide cache hit/miss info.

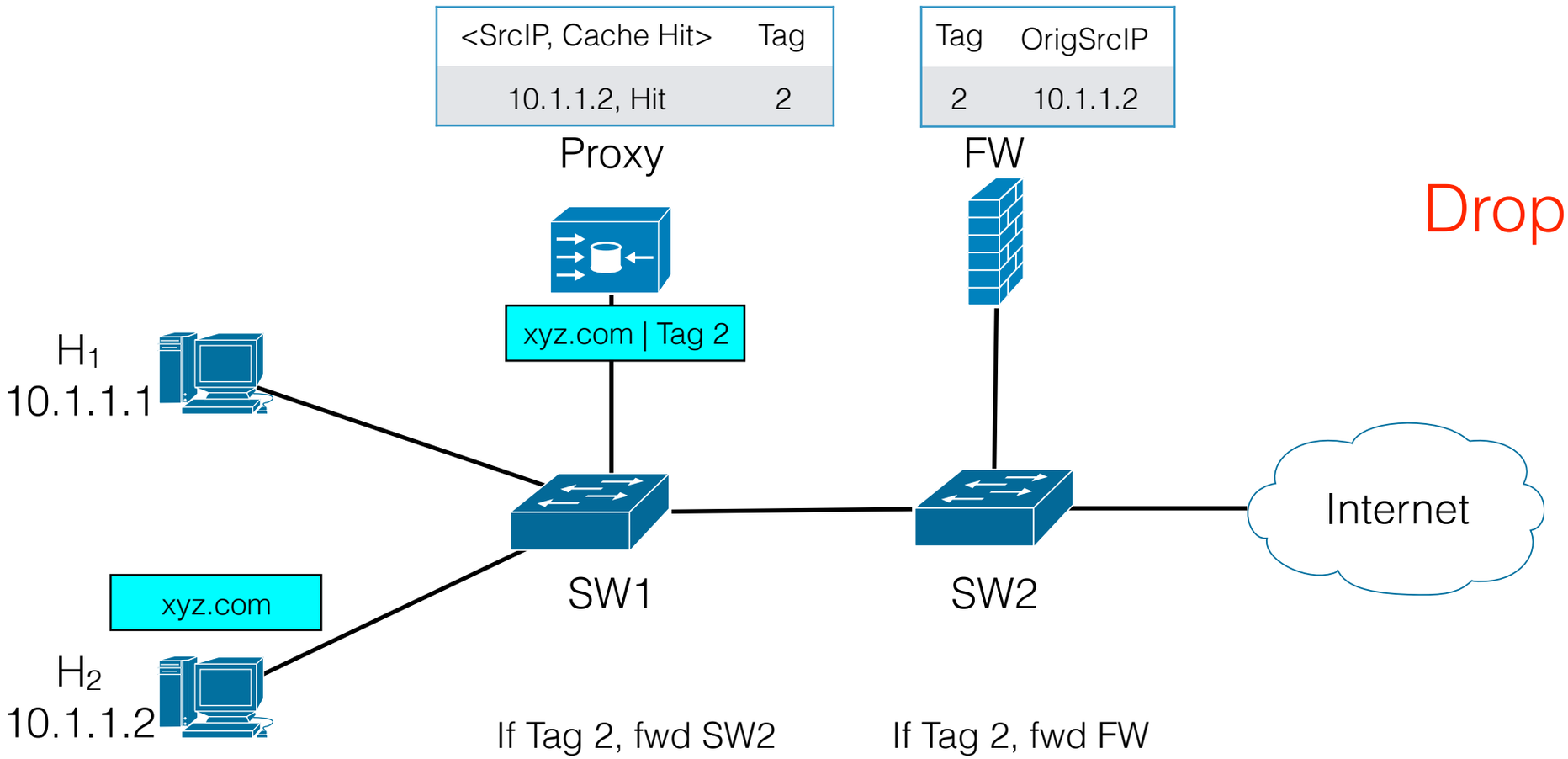
- FlowTag controller configures tagging logic.

FlowTags and SDN



FlowTags in action

Policy: Block 10.1.1.2 -> xyz.com

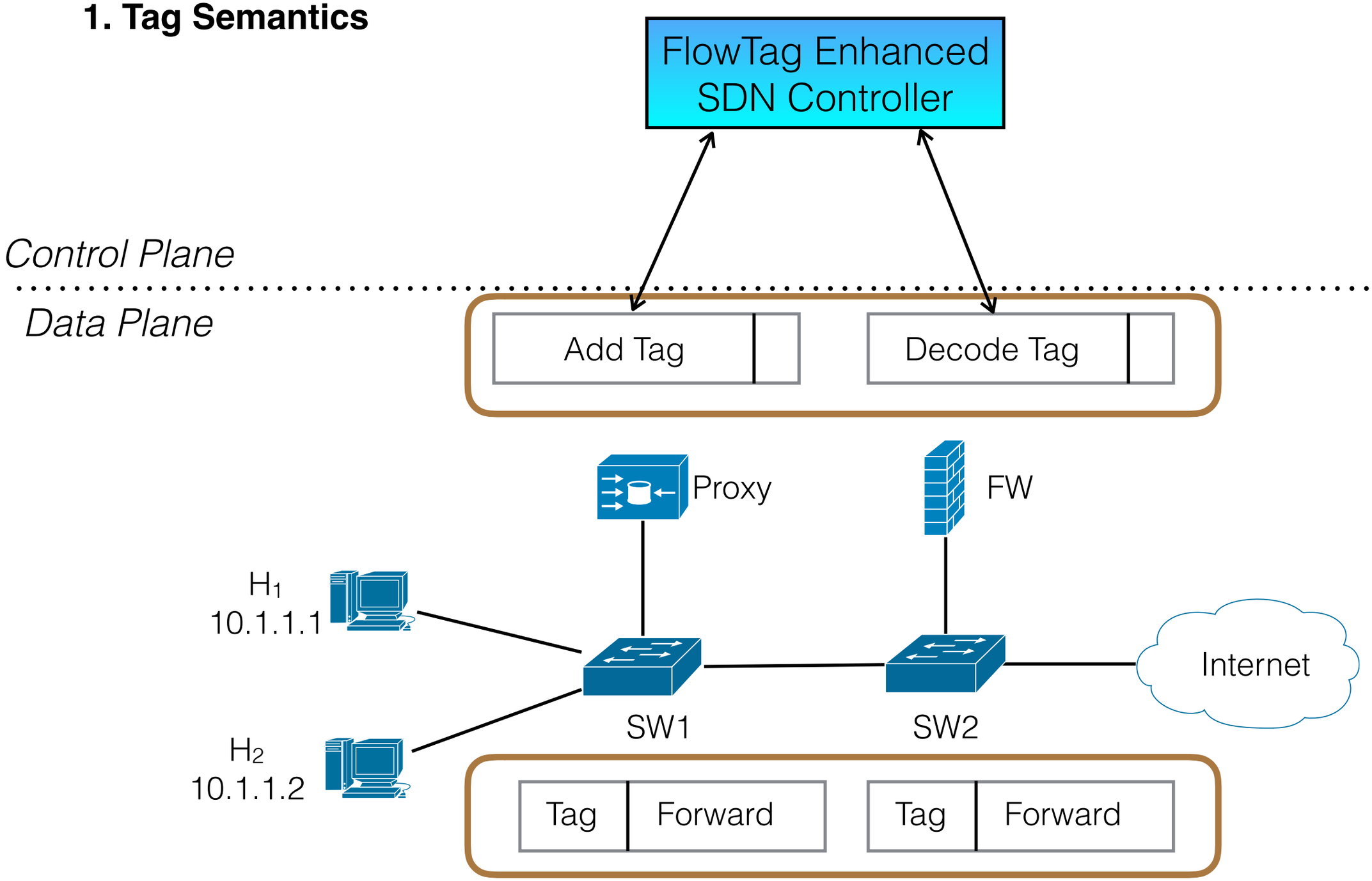


Outline

- Motivation
- High Level Idea
- **FlowTag Design**
- Evaluation
- Conclusion

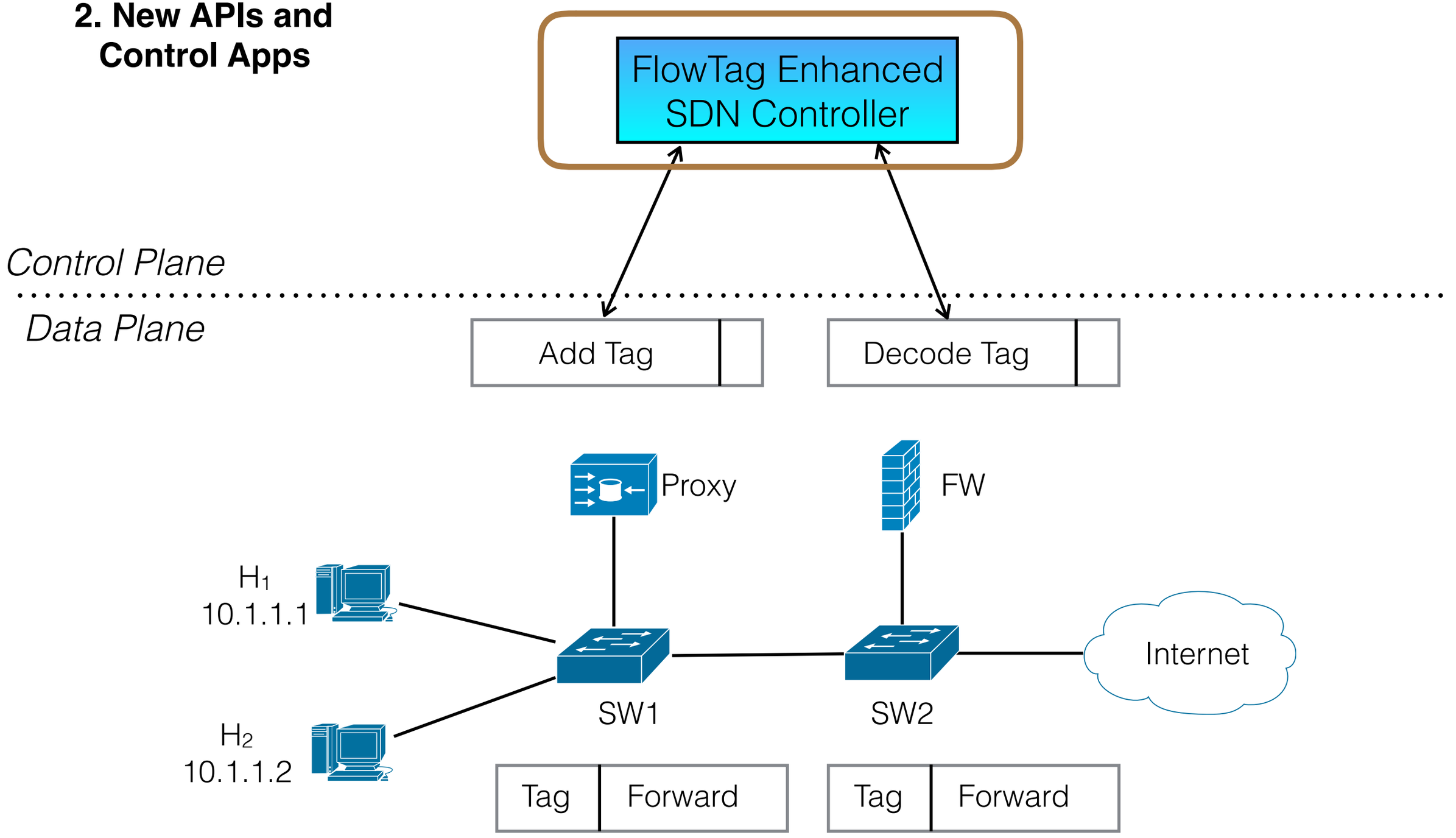
FlowTag Design Challenges

1. Tag Semantics



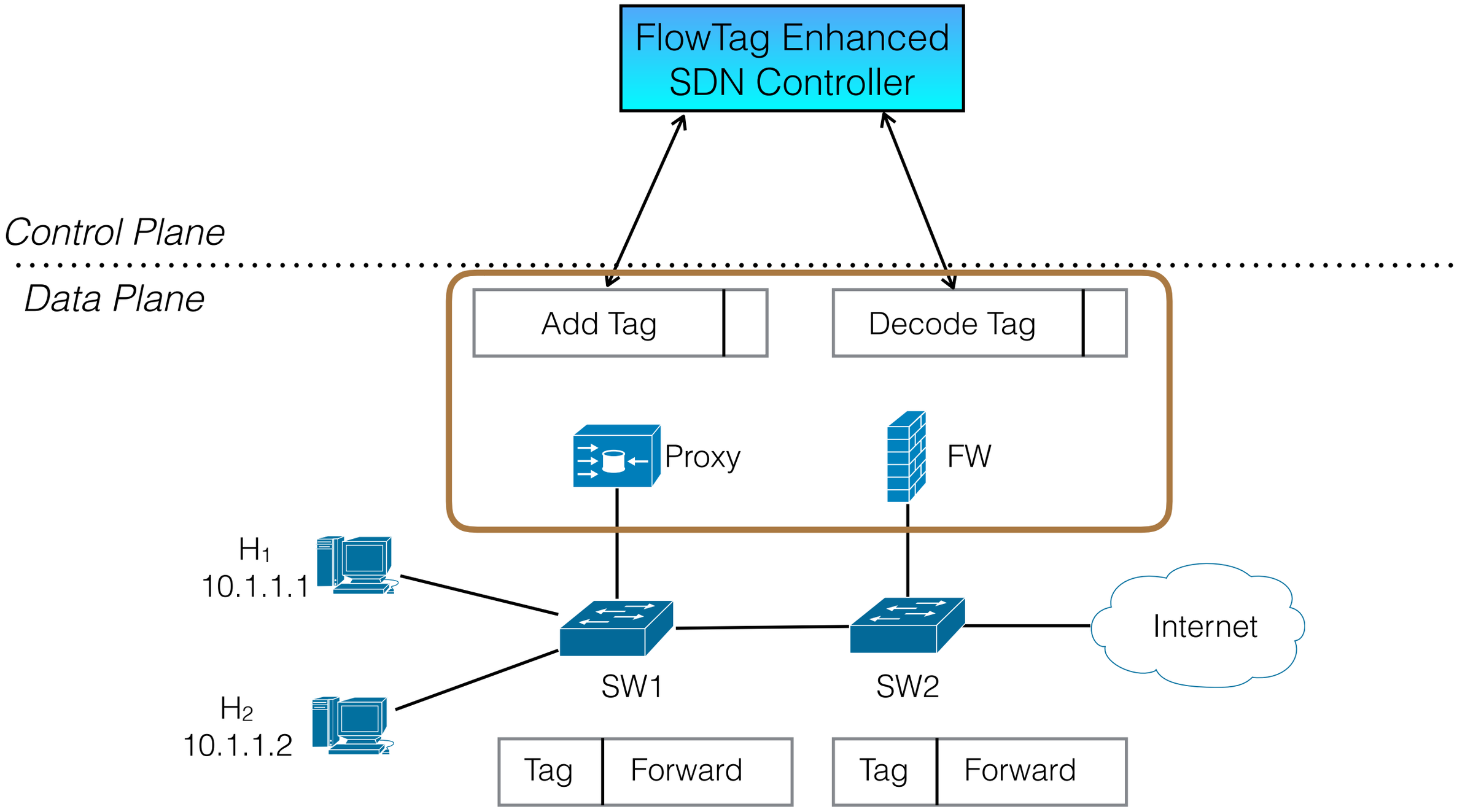
FlowTag Design Challenges

2. New APIs and Control Apps

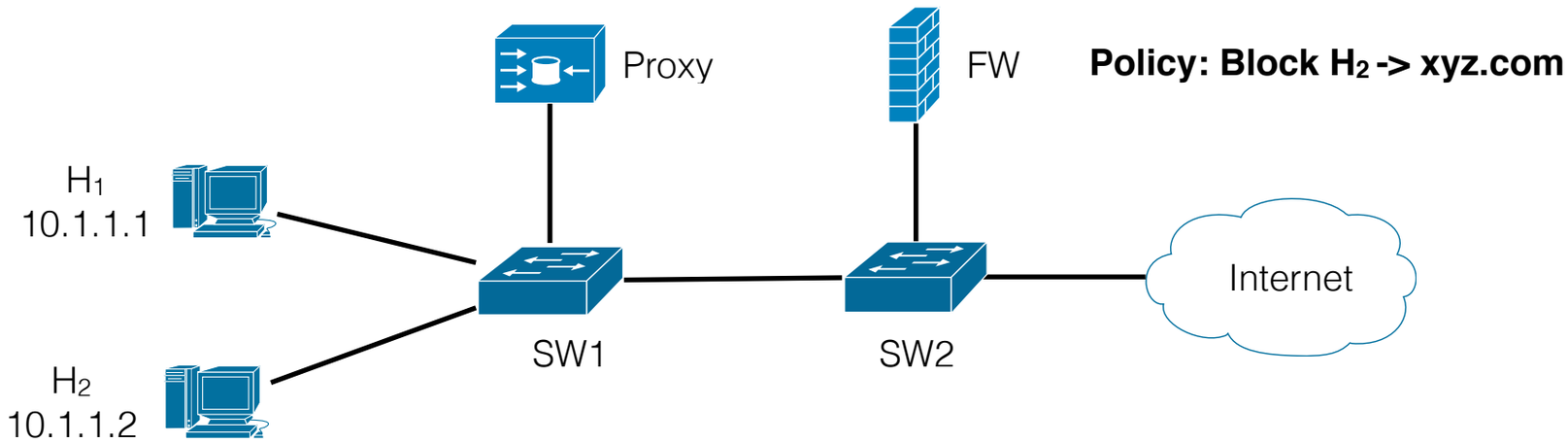


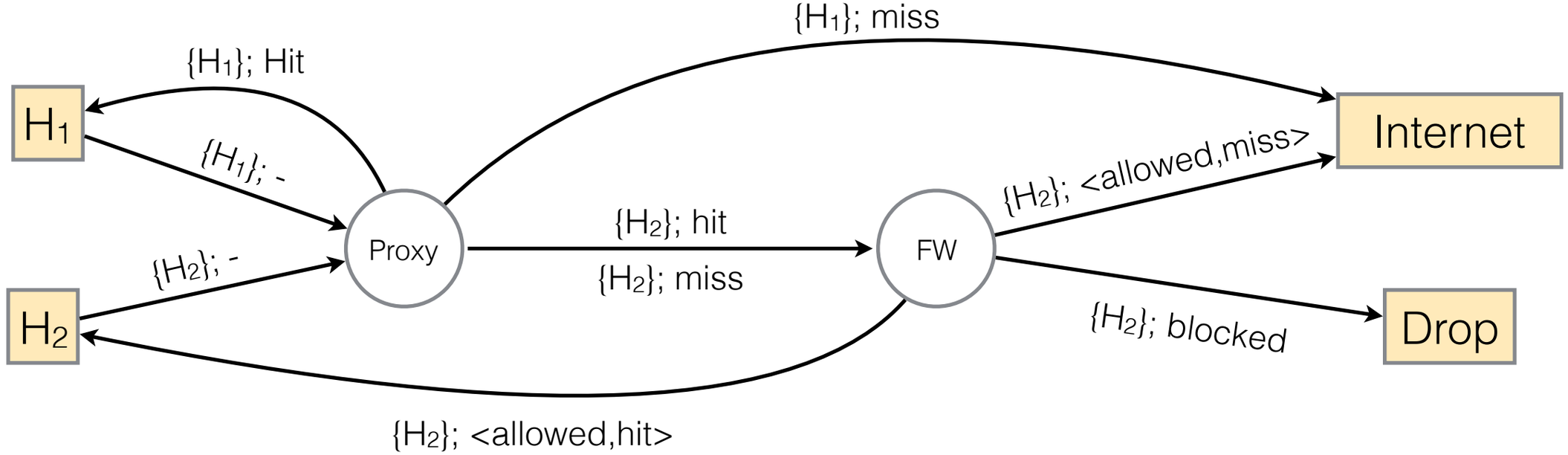
FlowTag Design Challenges

3. Middlebox Extension



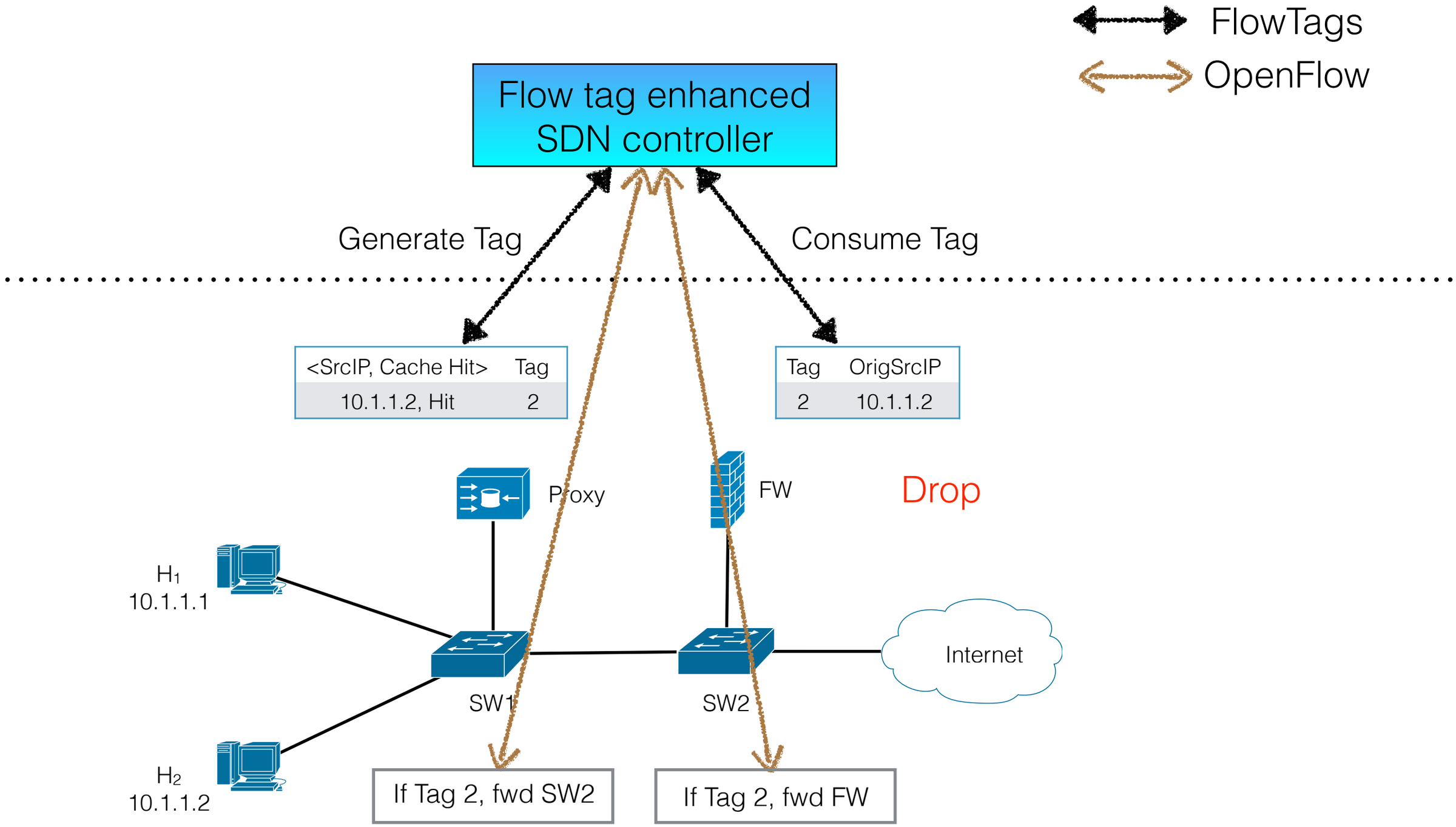
FlowTag Design : Semantics, Dynamic Policy Graph (DPG)



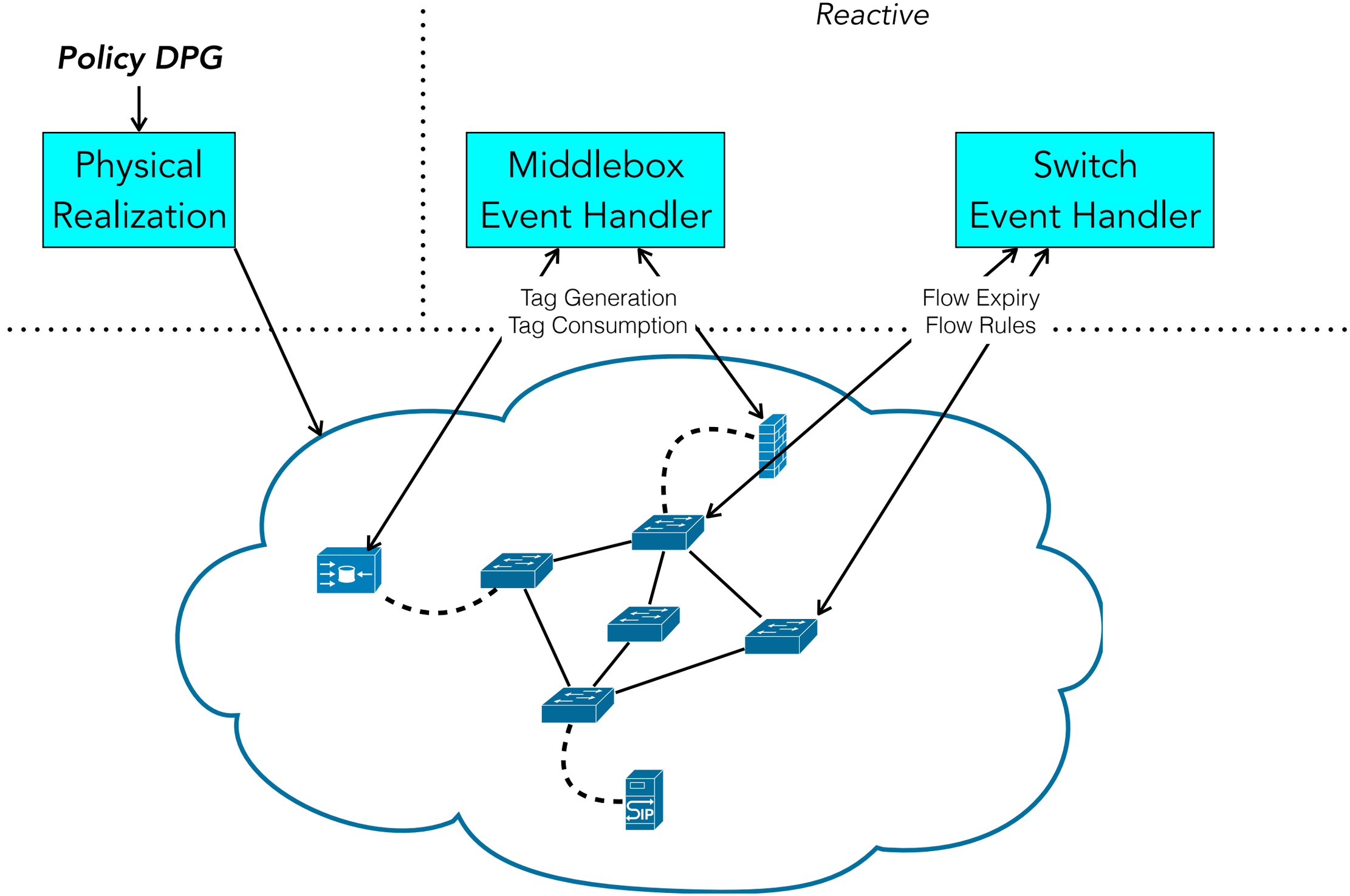


Need a tag <per flow , per edge> in DPG

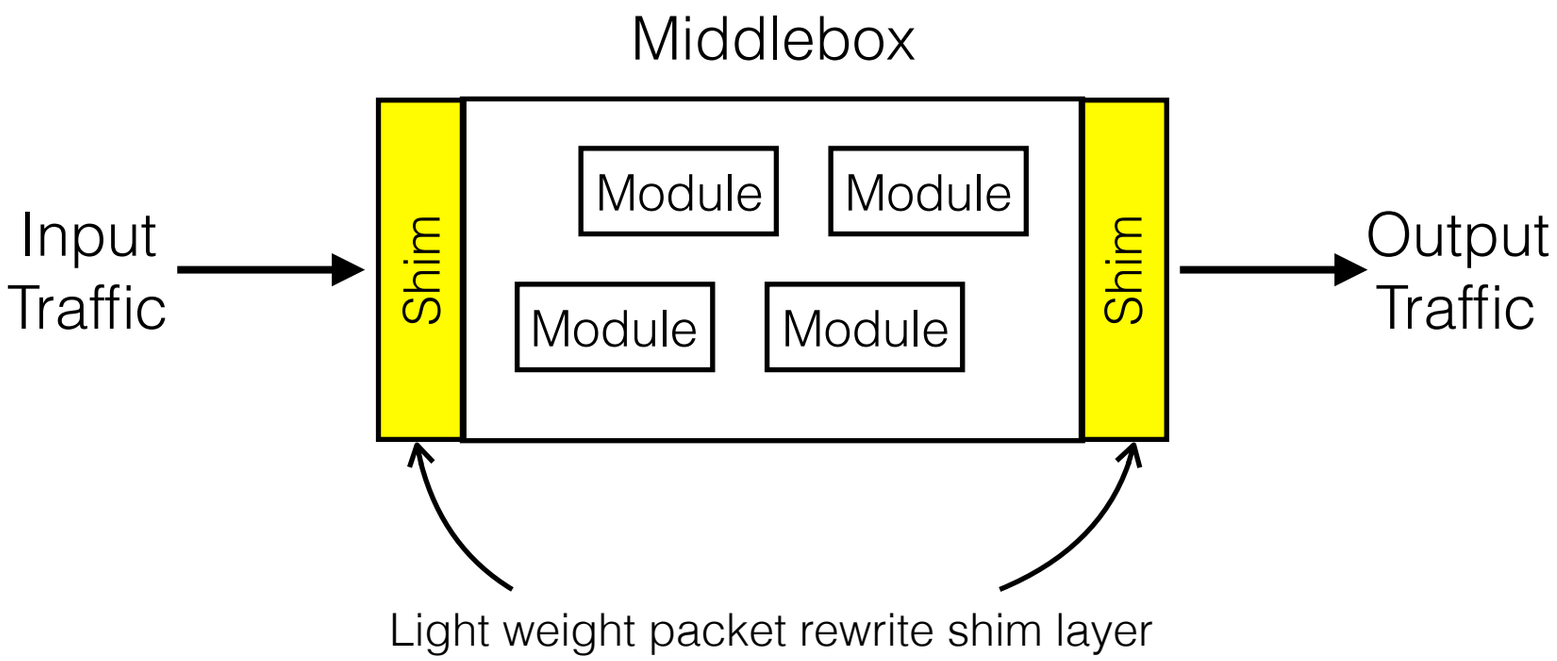
FlowTag Design : Controller and FlowTag APIs



FlowTag Design : FlowTag enhanced Controller



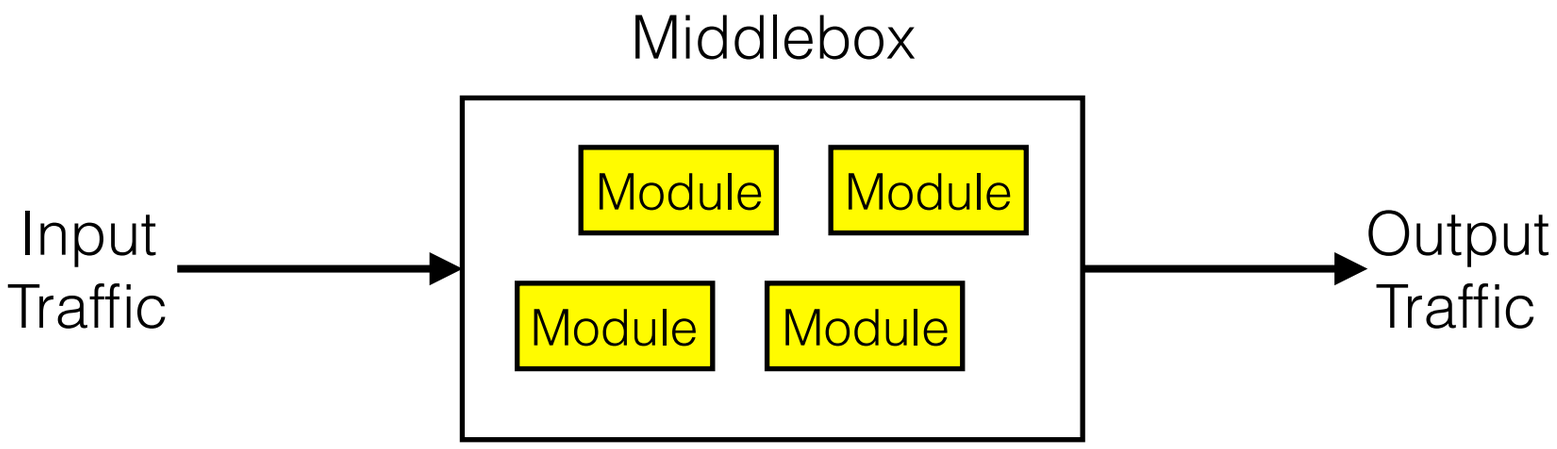
FlowTag Design : Middleboxes Extension



Pro : One shot

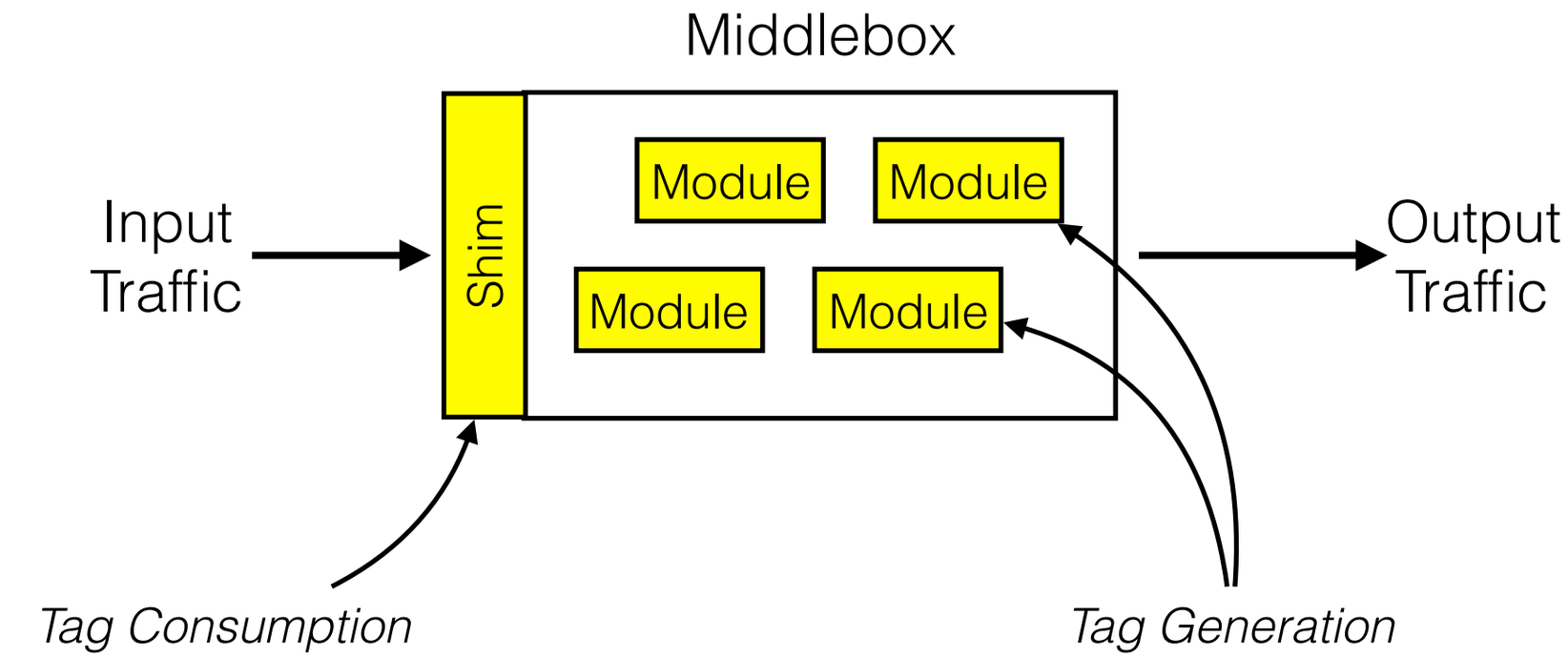
Con : Hard to get internal context

FlowTag Design : Middleboxes Extension



Pro : Can get internal context from module
Con : More changes are needed

FlowTag Design : Middleboxes Extension



Packet rewrite for Tag Consumption
Module modification for Tag Generation

Outlines

- Motivation
- High Level Idea
- FlowTag Design
- **Evaluation**
- Conclusion

Evaluation

- Key Evaluation questions :
  - Feasibility of middlebox modification
  - FlowTag Overhead
  - Number of Tag bits

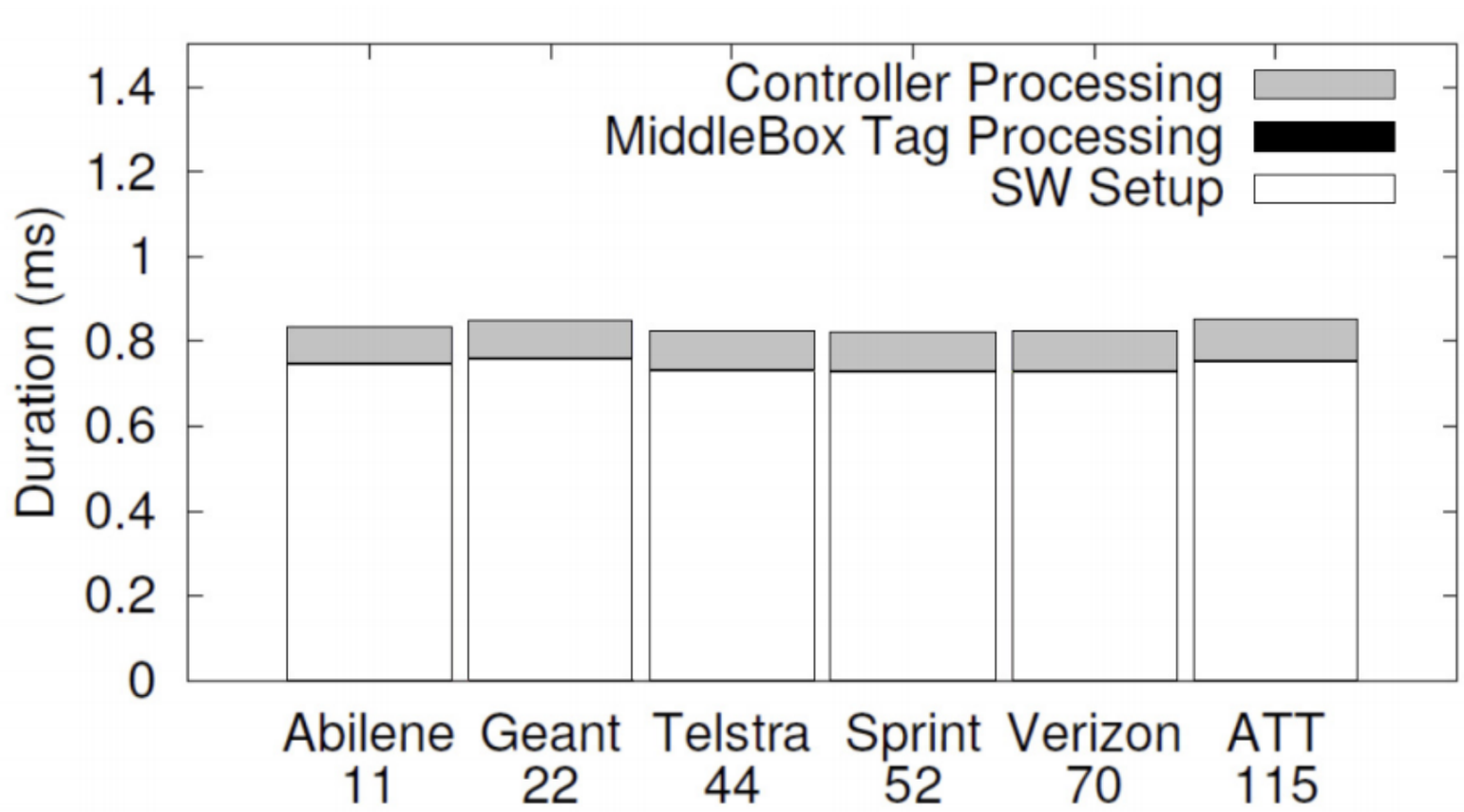
Evaluation

MiddleBox modification

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

Evaluation

MiddleBox overhead



Breakdown of flow processing time in different topologies (annotated with #nodes).

Evaluation

- 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

Conclusion

- **Middleboxes complicate policy enforcement**
  - E.g., NAT/LB rewrite headers, proxy sends cached response
- Root cause: Violation of the SDN tenets
  - OriginBinding and PathsFollowPolicy
- **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

Questions?