# SIMPLE-fying Middlebox Policy Enforcement Using SDN

Zafar Ayyub Qazi
Cheng-Chun Tu
Luis Chiang
Vyas Sekar

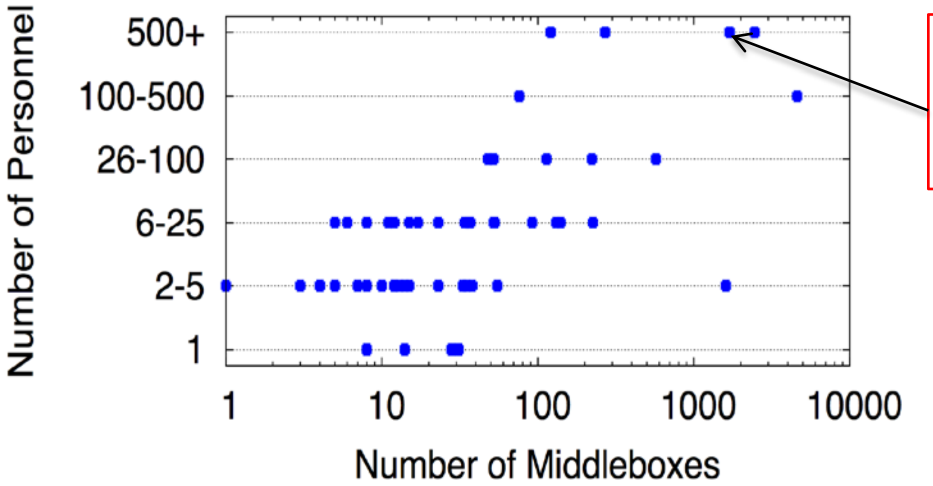
Rui Miao
Minlan Yu

Stony Brook University

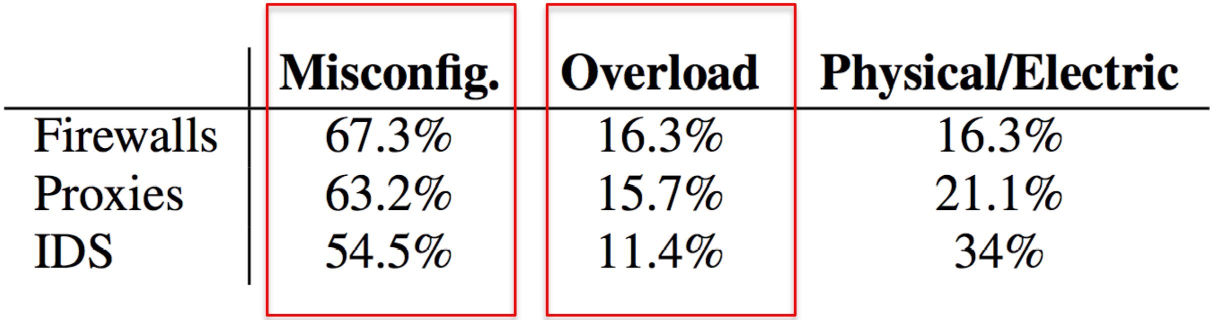
USC University of Southern California

# Middleboxes management is hard!

Survey across 57 network operators *(J. Sherry et al. SIGCOMM 2012)*



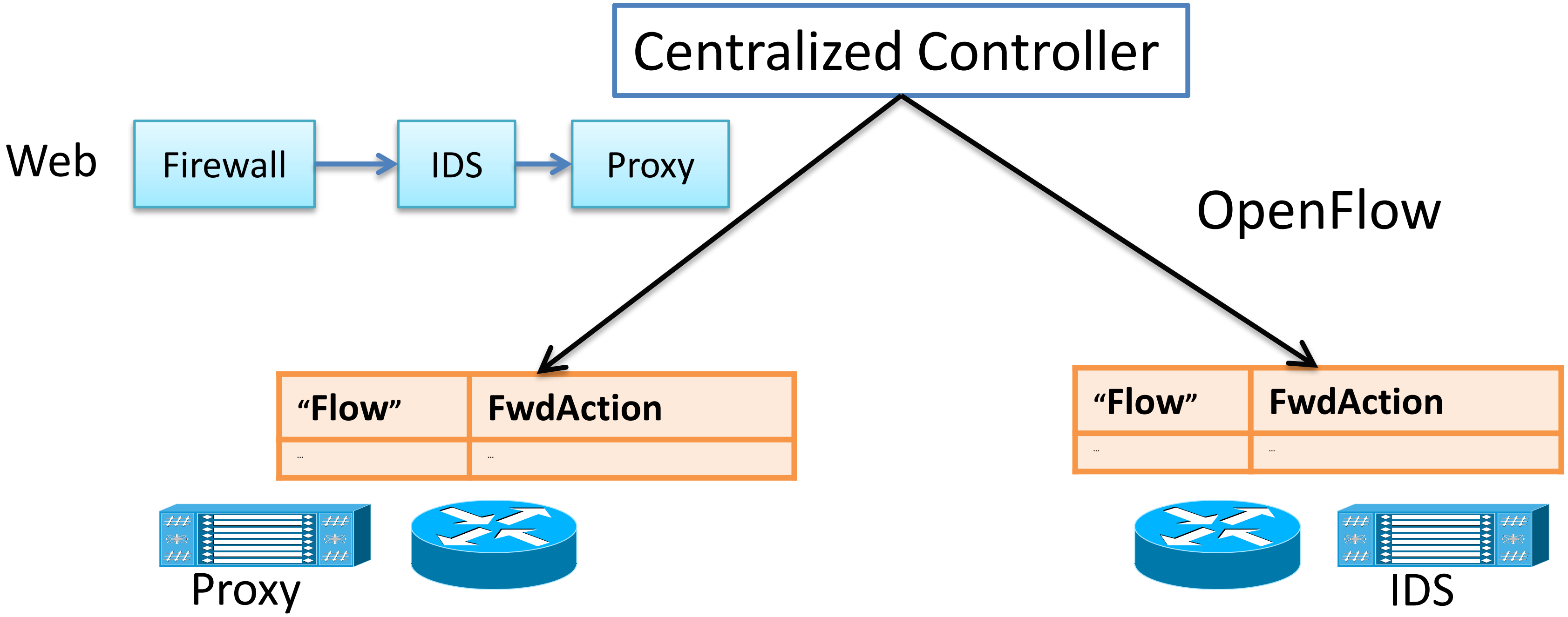
e.g., a network with ~2000 middleboxes required 500+ operators



| | Misconfig. | Overload | Physical/Electric |
|---|---|---|---|
| Firewalls | 67.3% | 16.3% | 16.3% |
| Proxies | 63.2% | 15.7% | 21.1% |
| IDS | 54.5% | 11.4% | 34% |

Critical for security, performance, compliance
But expensive, complex and difficult to manage

# Can SDN simplify middlebox management?



Centralized Controller

Web

Firewall → IDS → Proxy

OpenFlow

| “Flow” | FwdAction |
|---|---|
| … | … |

| “Flow” | FwdAction |
|---|---|
| … | … |

Proxy

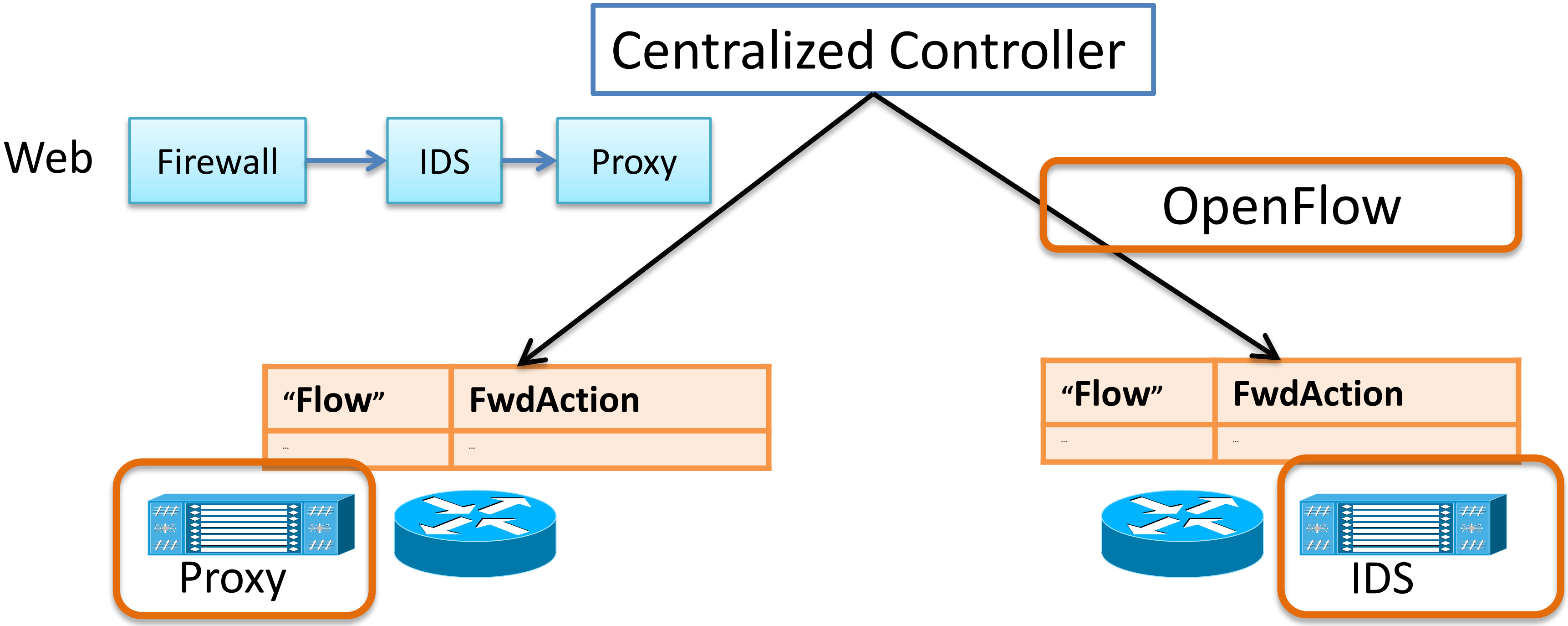
IDS

**Scope**: Enforce middlebox-specific steering policies

**Necessity + Opportunity:**

Incorporate functions markets views as important

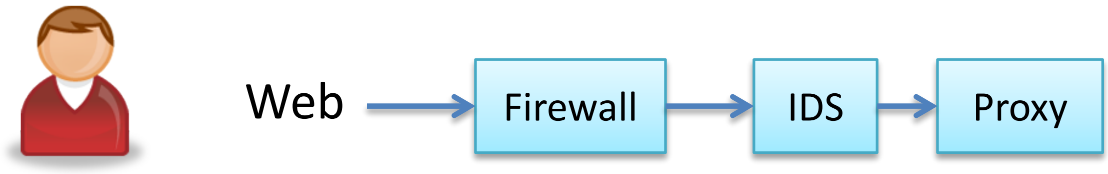
# What makes this problem challenging?



Middleboxes introduce new dimensions beyond L2/L3 tasks.

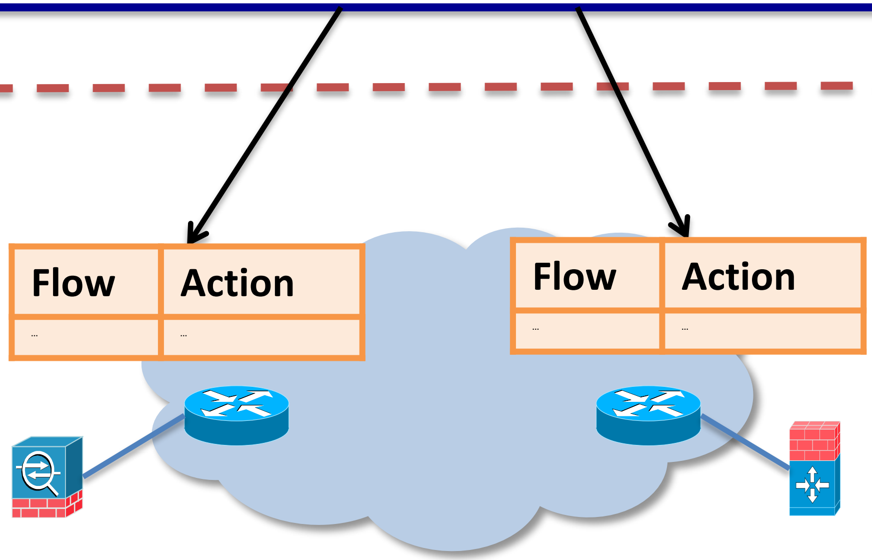
Achieve this with ***unmodified*** middleboxes and ***existing*** SDN APIs

# Our Work: SIMPLE



Web → Firewall → IDS → Proxy

Policy enforcement layer for middlebox-specific "traffic steering"



| Flow | Action |
|---|---|
| … | … |

| Flow | Action |
|---|---|
| … | … |

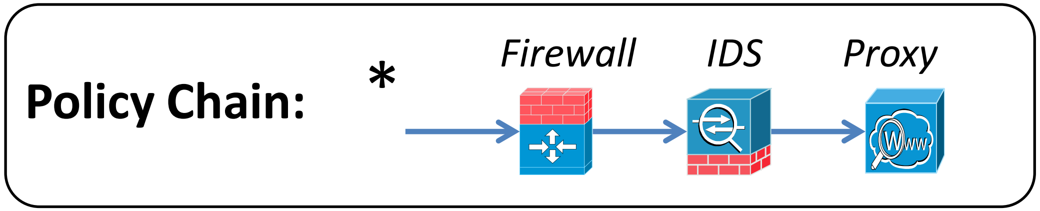
*Legacy Middleboxes*

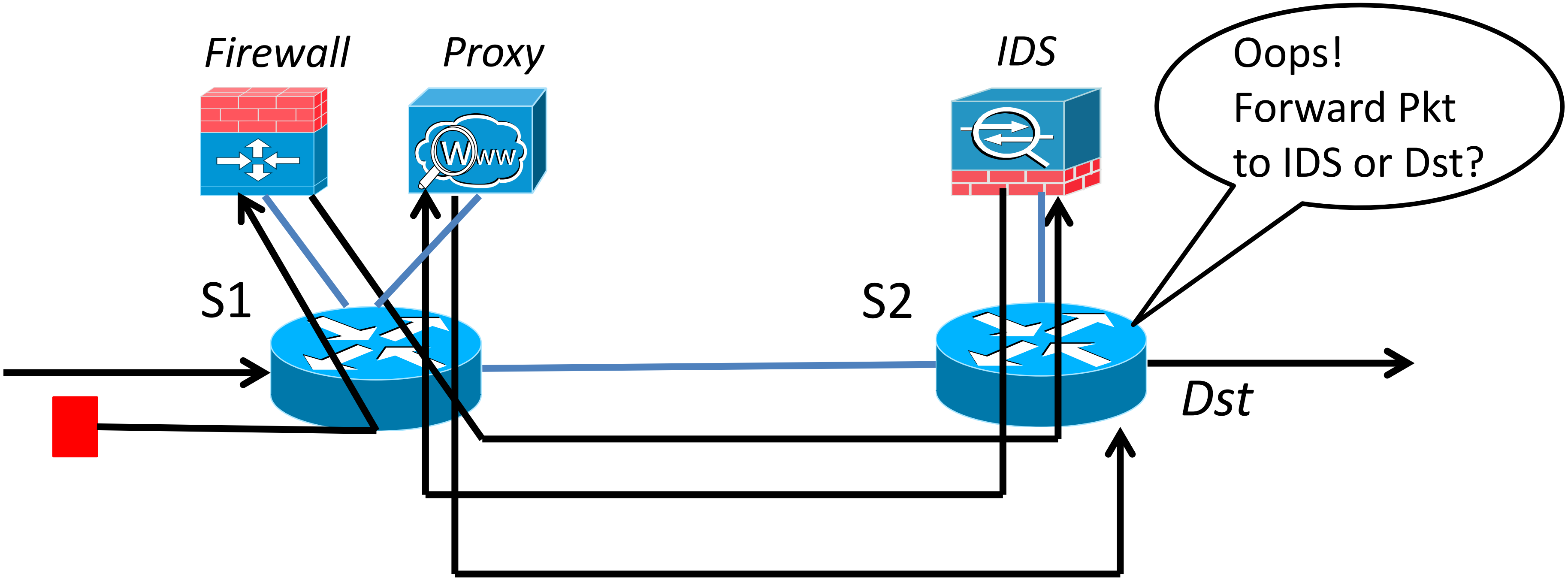
*OpenFlow capable*

# Outline

- Motivation
- ***Challenges***
- SIMPLE Design
- Evaluation
- Conclusions

# Challenge: Policy Composition

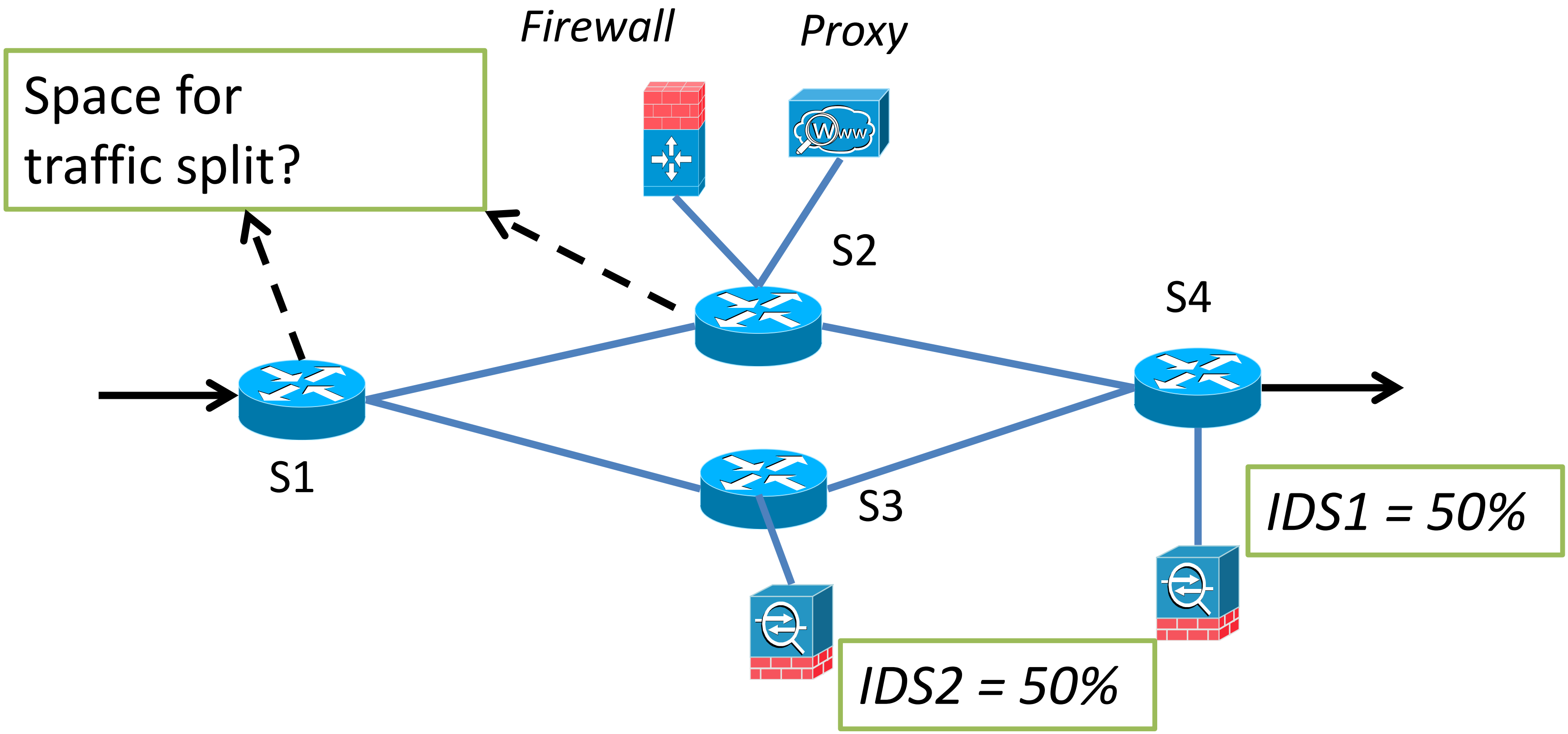




"Loops"

Traditional flow rules may not suffice!

# Challenge: Resource Constraints



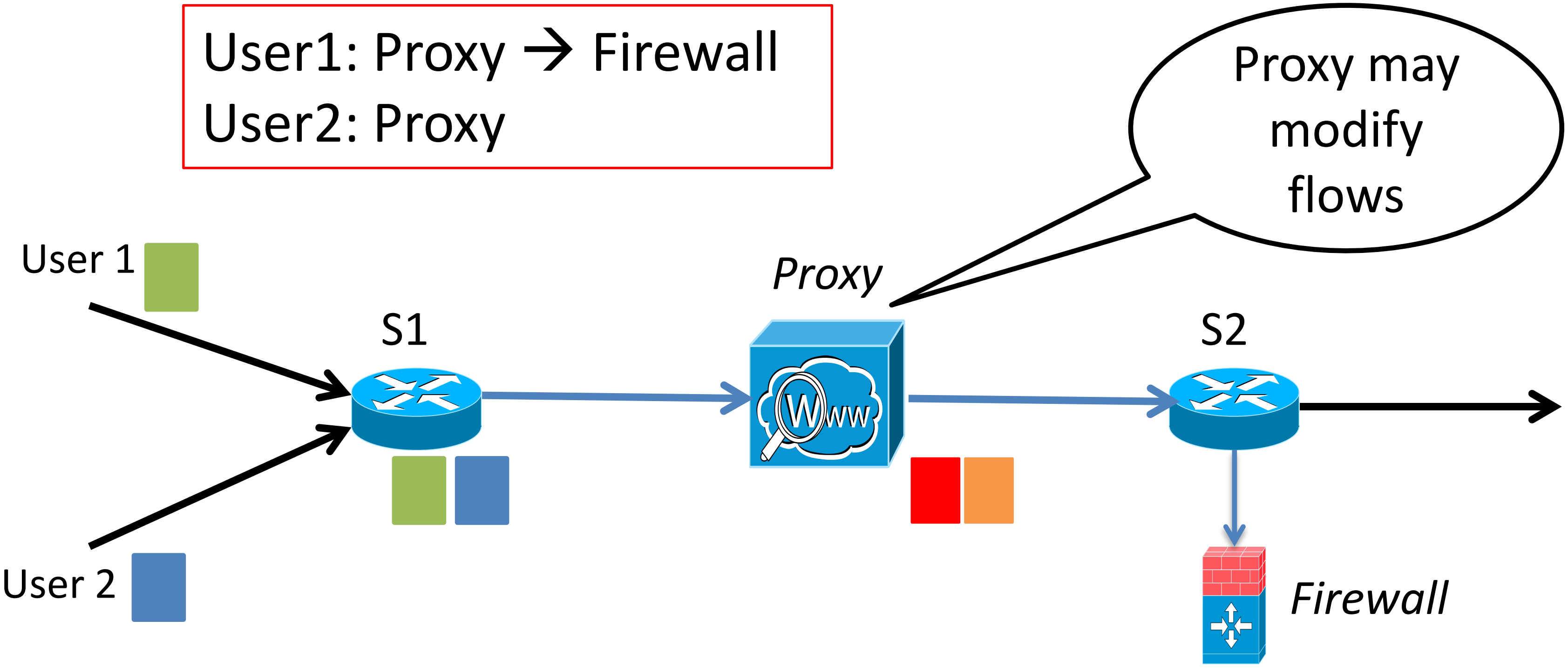
Space for traffic split?

*IDS1 = 50%*

*IDS2 = 50%*

Can we set up “feasible” forwarding rules?

# Challenge: Dynamic Modifications



User1: Proxy → Firewall
User2: Proxy

Proxy may modify flows

Are forwarding rules at S2 correct?

# New dimensions beyond Layer 2-3 tasks

1) Policy Composition → Potential loops

2) Resource Constraints → Switch + Middlebox

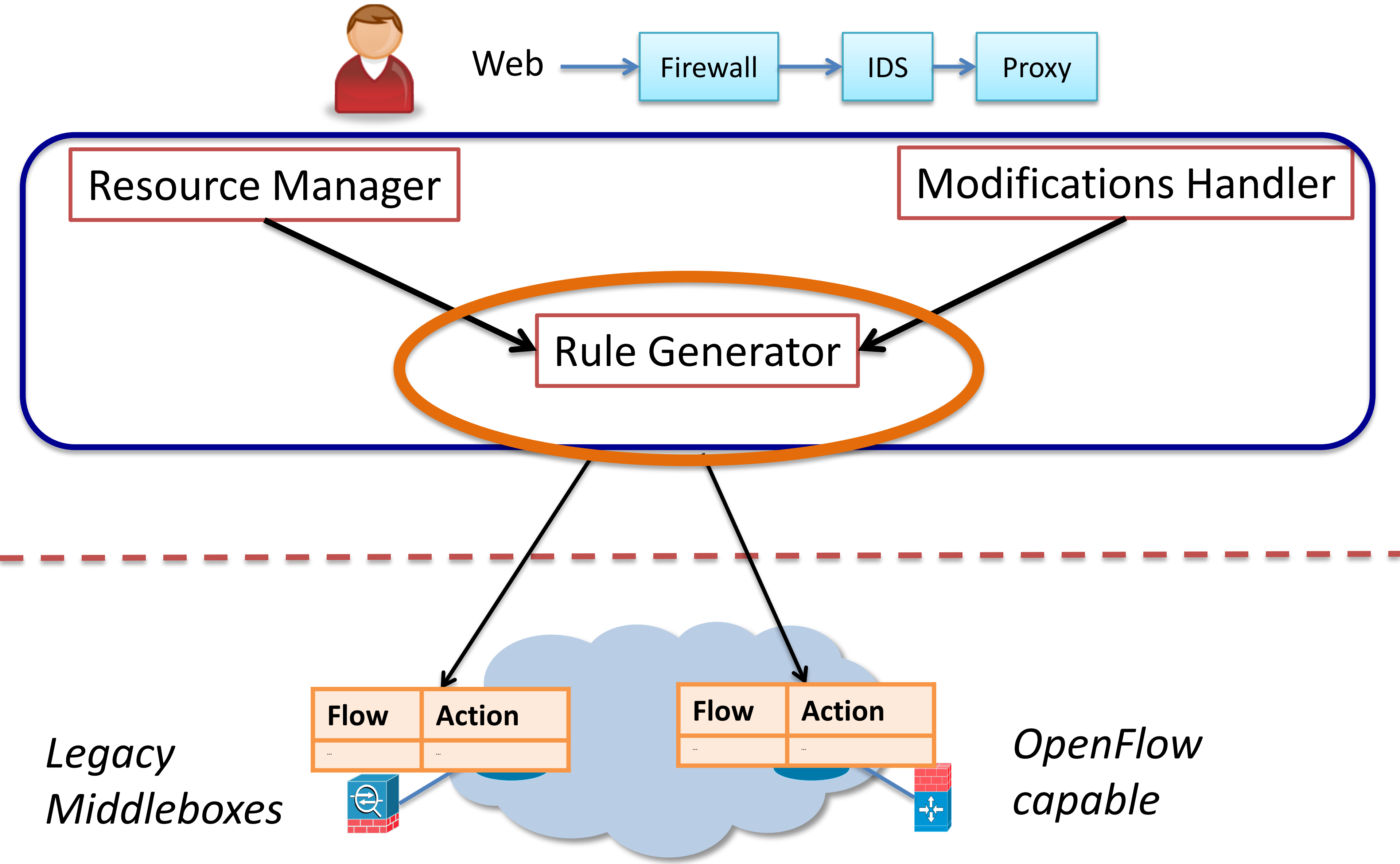
3) Dynamic Modifications → Correctness?

Can we address these with ***unmodified*** middleboxes and ***existing*** SDN APIs?

# Outline

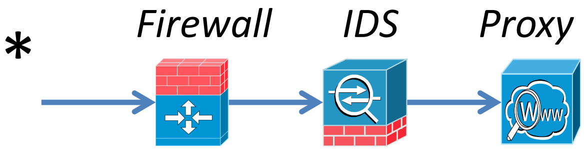
- Motivation + Context for the Work
- Challenges
- ***SIMPLE Design***
- Evaluation
- Conclusion

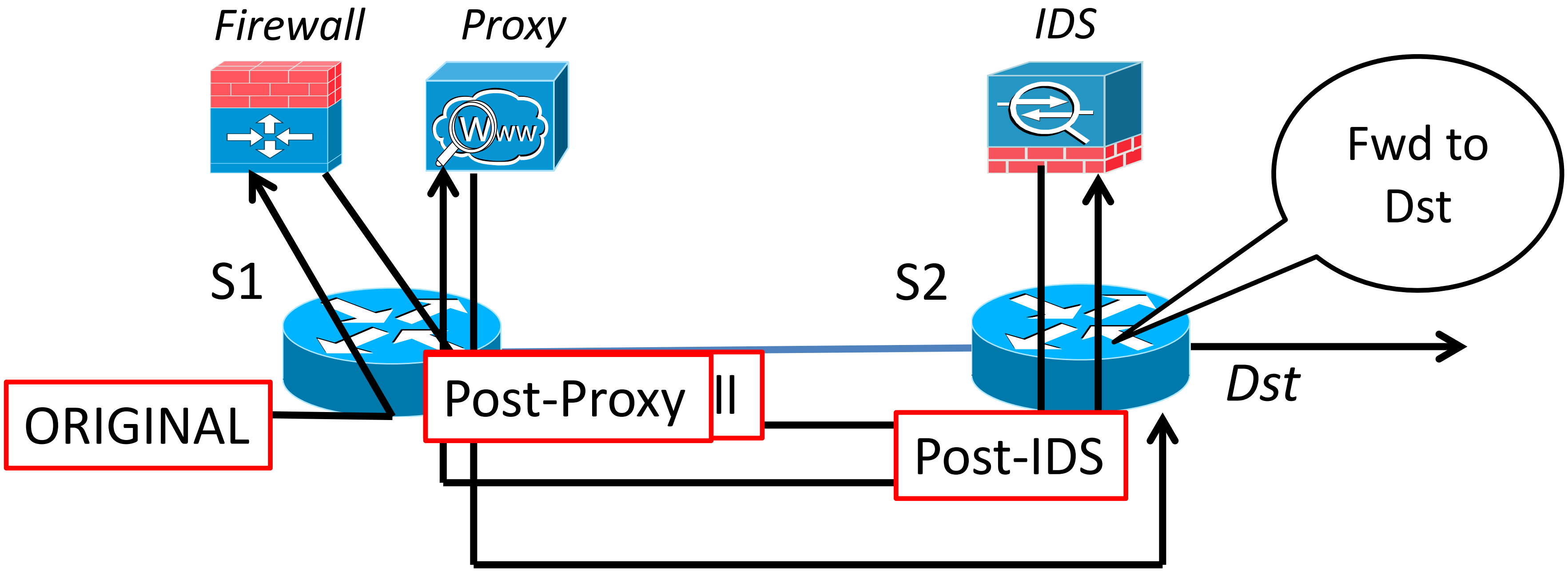
# SIMPLE System Overview



# Composition → Tag Processing State

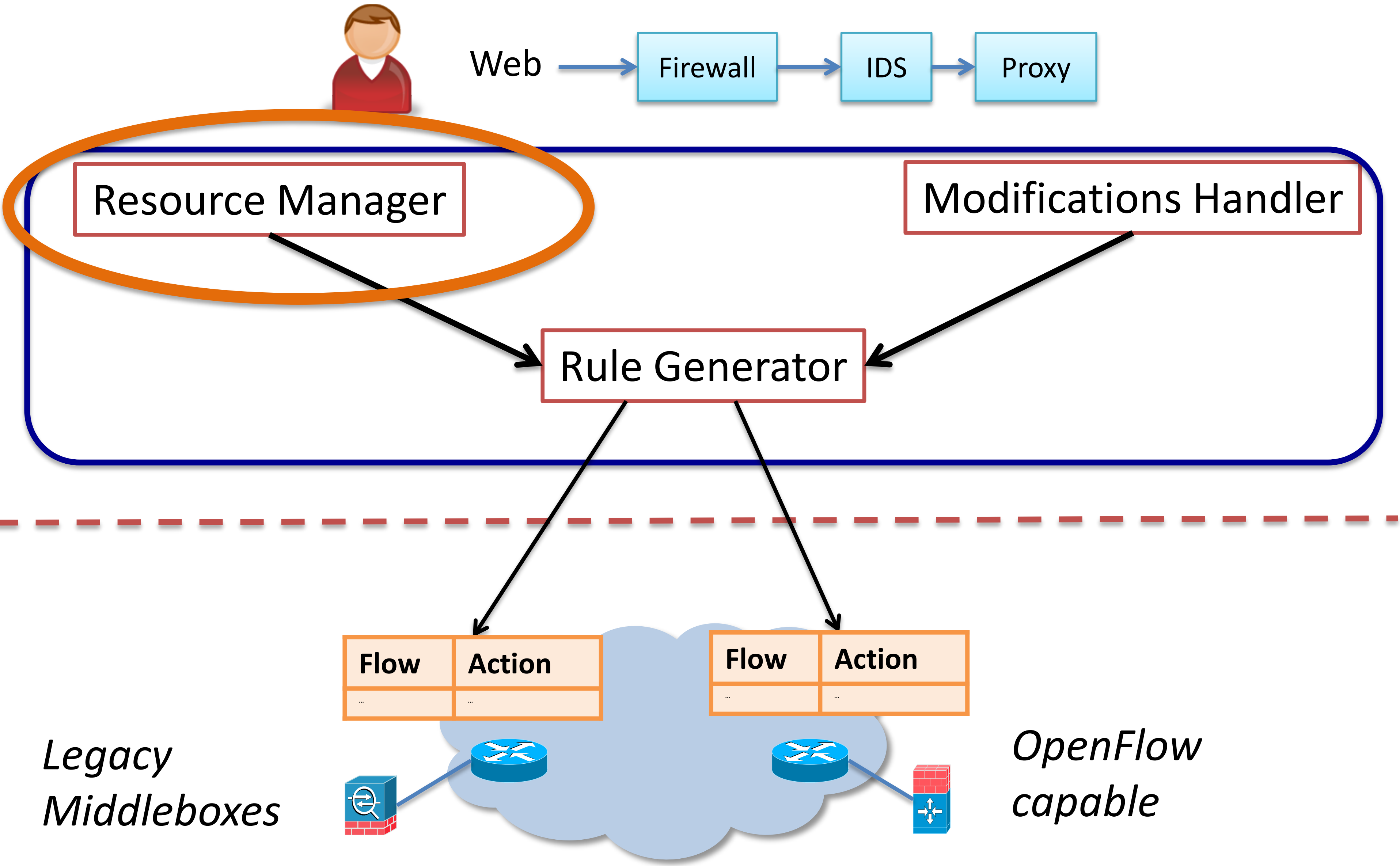
**Policy Chain:**



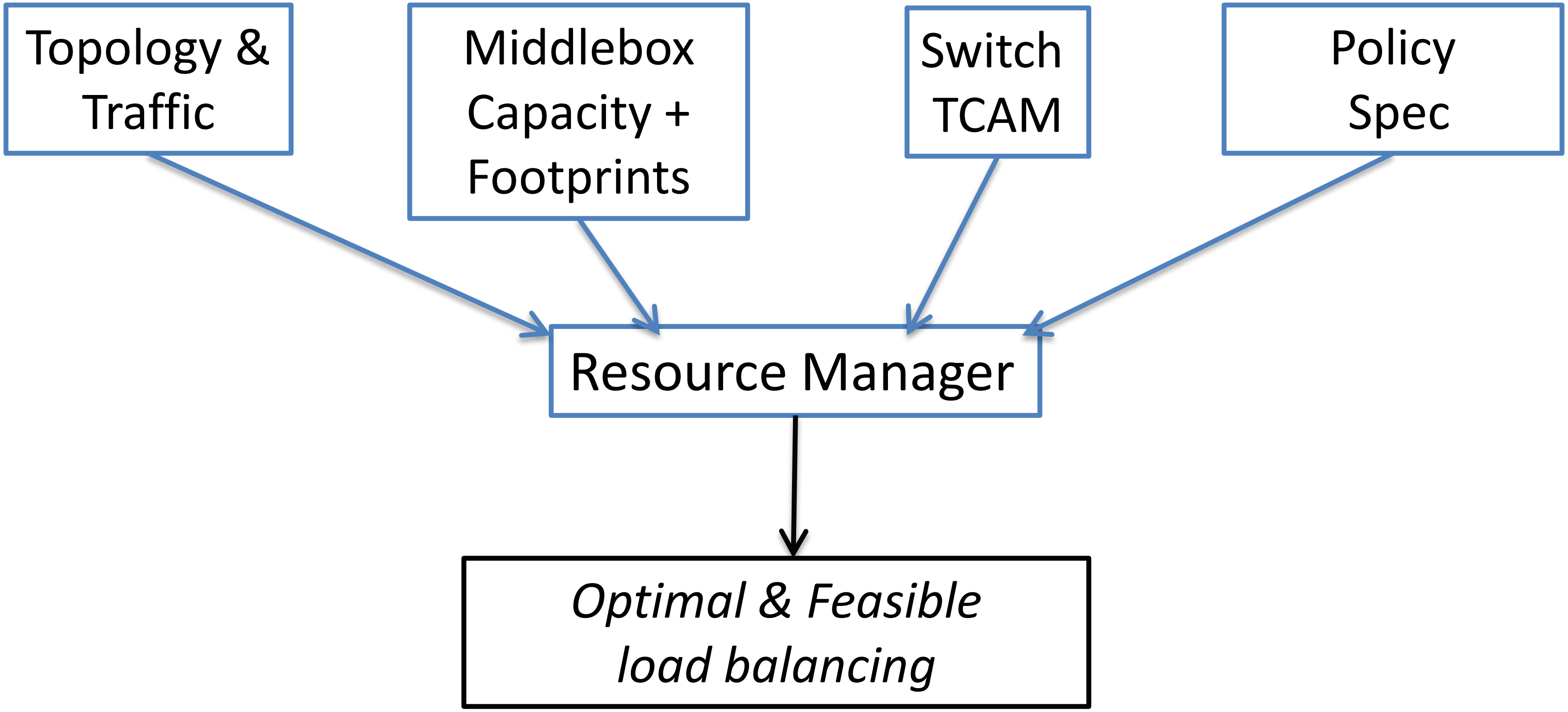


Insight: Distinguish different instances of the same packet

# SIMPLE System Overview



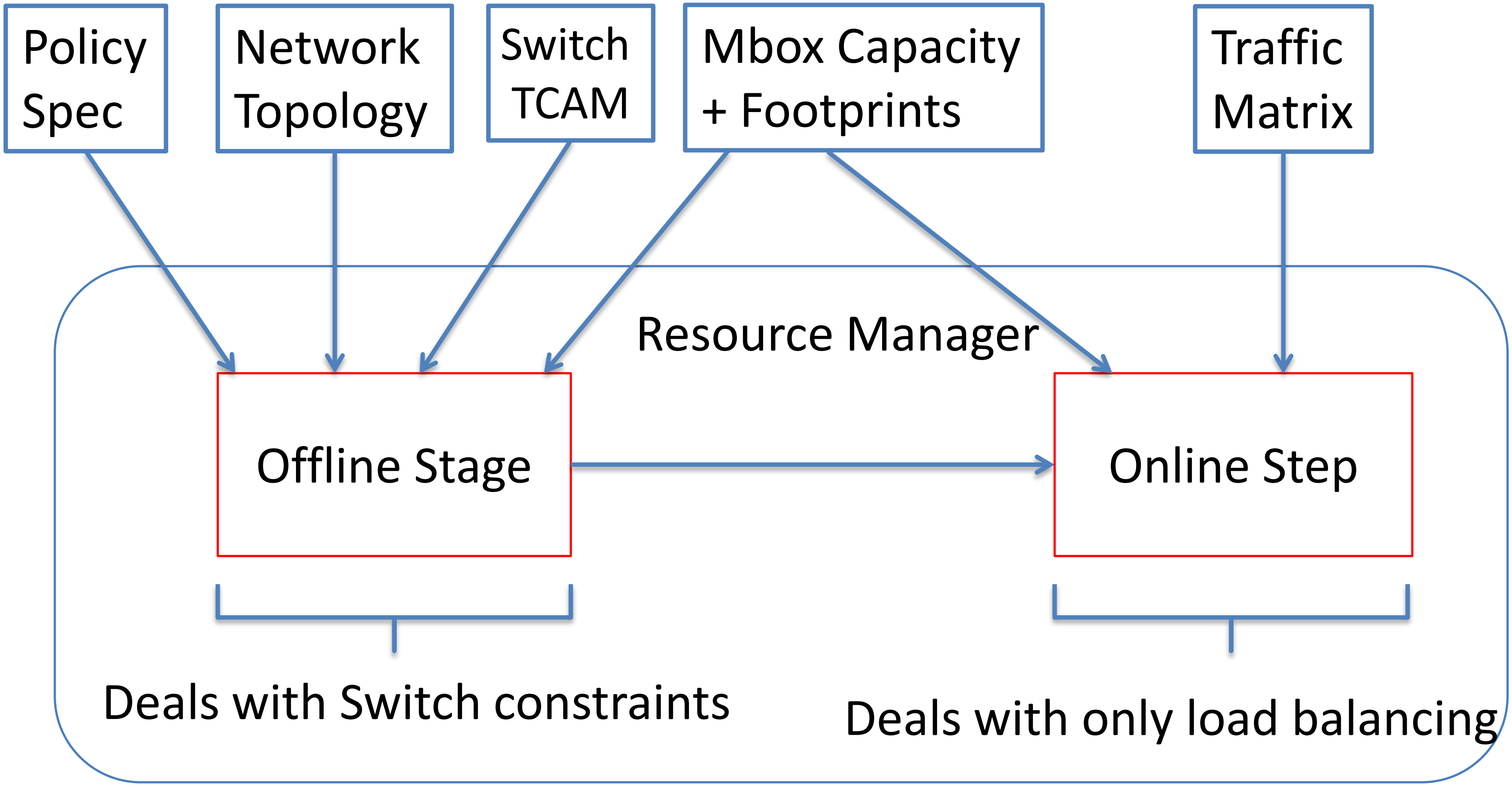
# Resource Constraints→ Joint Optimization


Topology & Traffic
Middlebox Capacity + Footprints
Switch TCAM
Policy Spec
Resource Manager
*Optimal & Feasible load balancing*

Theoretically hard!

Not obvious if some configuration is feasible!

# Offline + Online Decomposition



Policy Spec

Network Topology

Switch TCAM

Mbox Capacity + Footprints

Traffic Matrix

Resource Manager

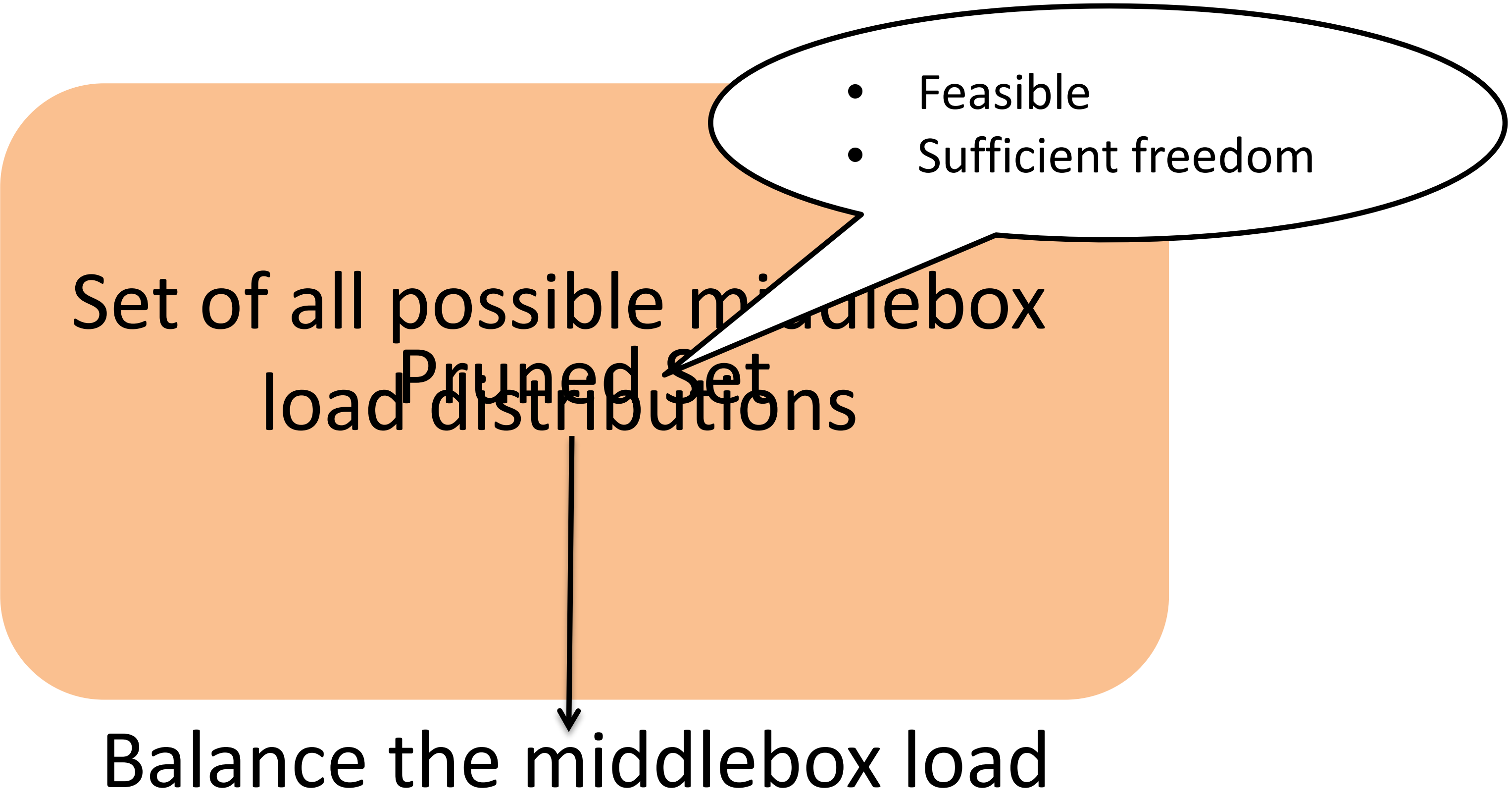
Offline Stage

Online Step

Deals with Switch constraints

Deals with only load balancing

# Offline Stage: ILP based pruning



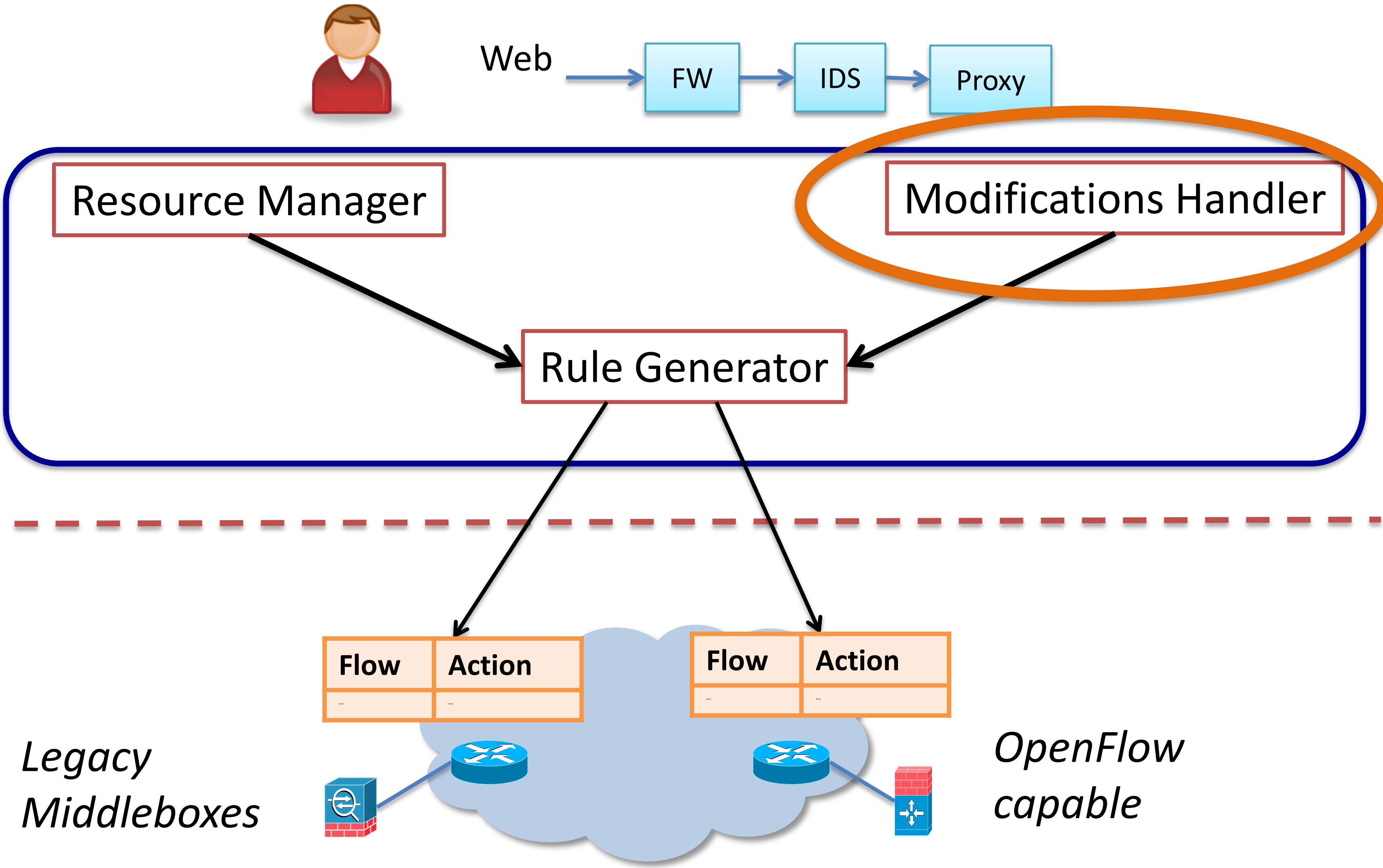
Set of all possible middlebox load distributions

Pruned Set

- Feasible
- Sufficient freedom

Balance the middlebox load

# SIMPLE System Overview



Web

FW

IDS

Proxy

Resource Manager

Modifications Handler

Rule Generator

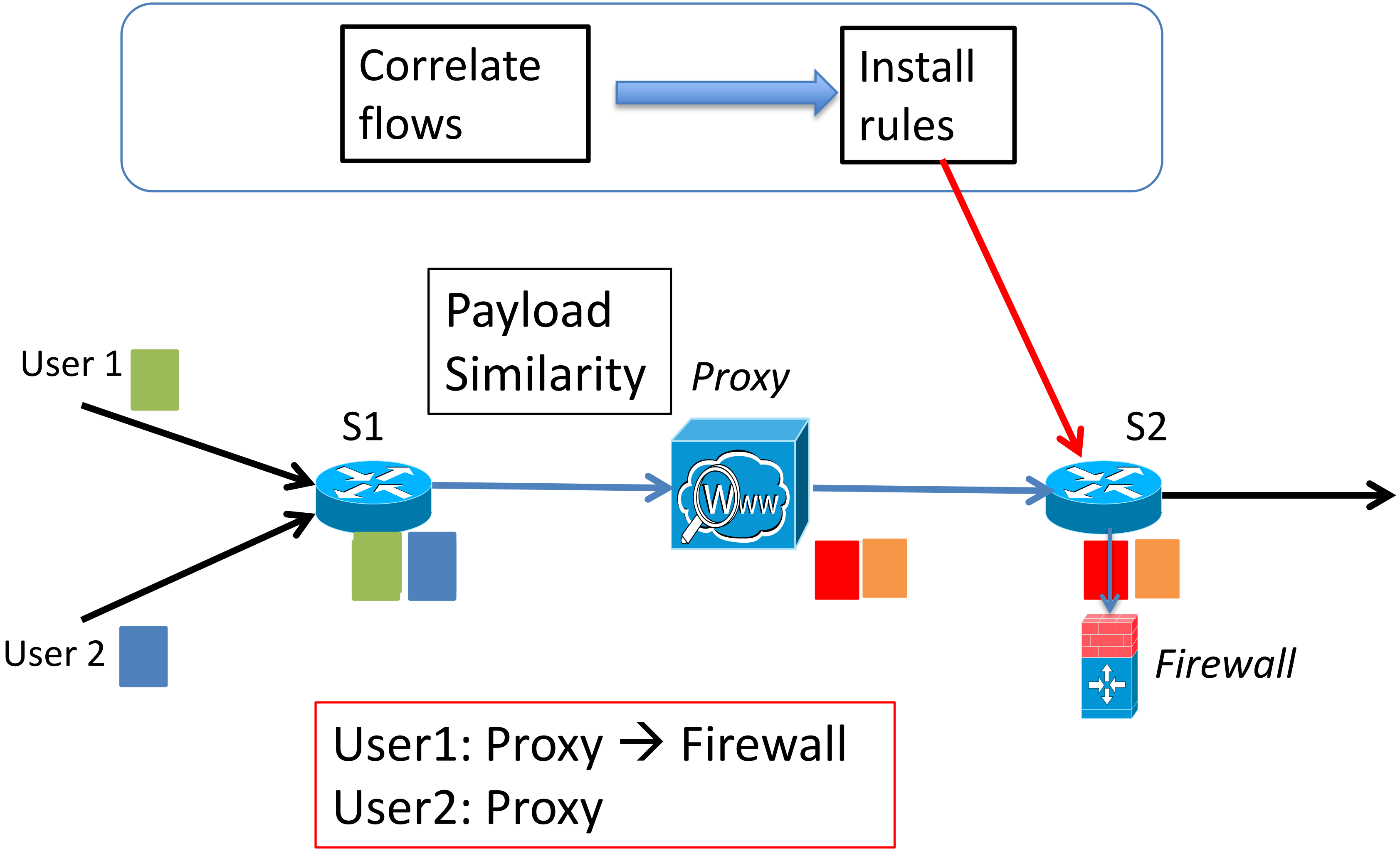
| Flow | Action |
|---|---|
| … | … |

| Flow | Action |
|---|---|
| … | … |

*Legacy Middleboxes*

*OpenFlow capable*

# Modifications → Infer flow correlations

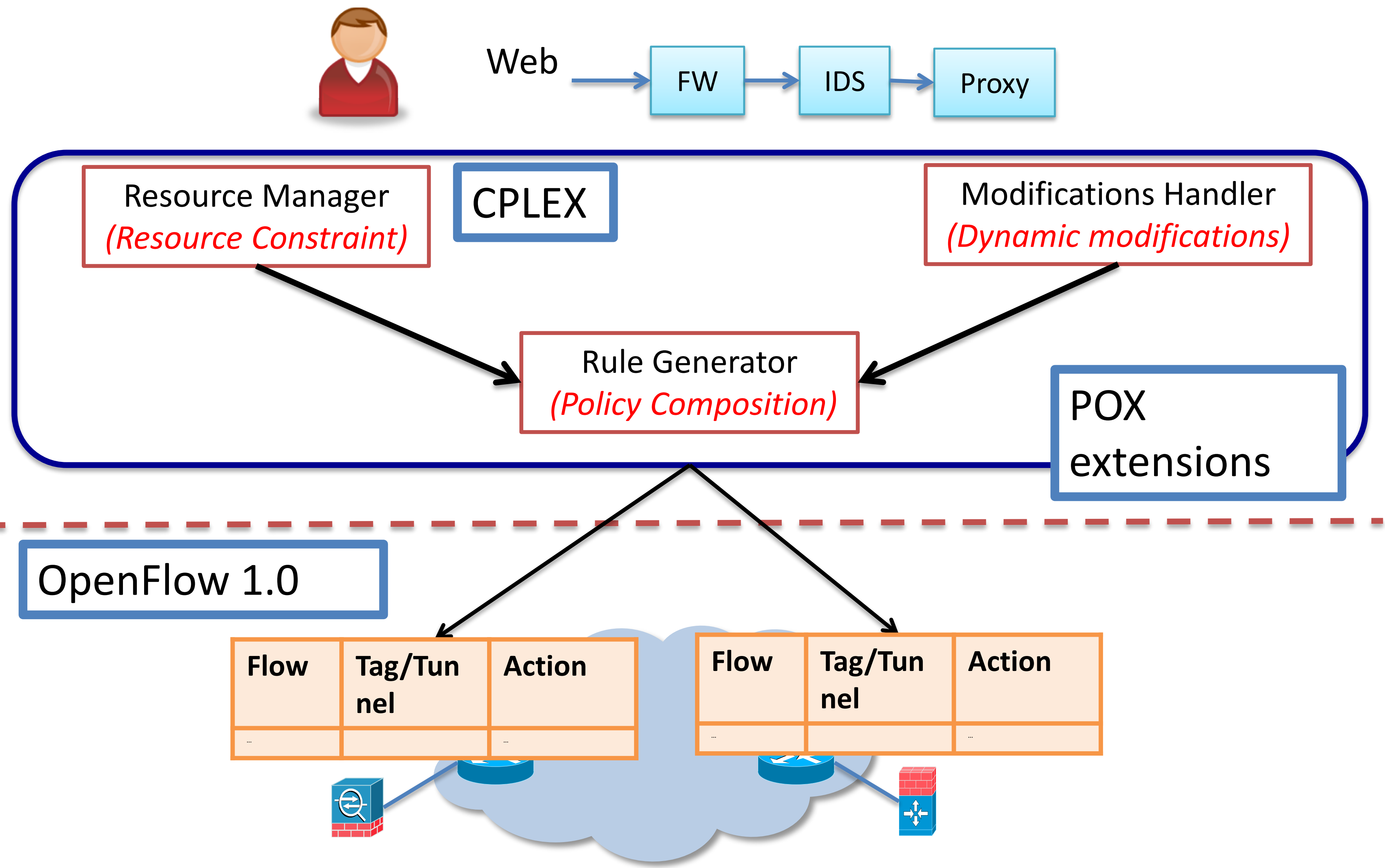

Correlate flows

Install rules

Payload Similarity

User1: Proxy → Firewall
User2: Proxy

# SIMPLE Implementation



Web → FW → IDS → Proxy

Resource Manager *(Resource Constraint)*

CPLEX

Modifications Handler *(Dynamic modifications)*

Rule Generator *(Policy Composition)*

POX extensions

OpenFlow 1.0

| Flow | Tag/Tunnel | Action |
|---|---|---|
| … | | … |

| Flow | Tag/Tunnel | Action |
|---|---|---|
| … | | … |

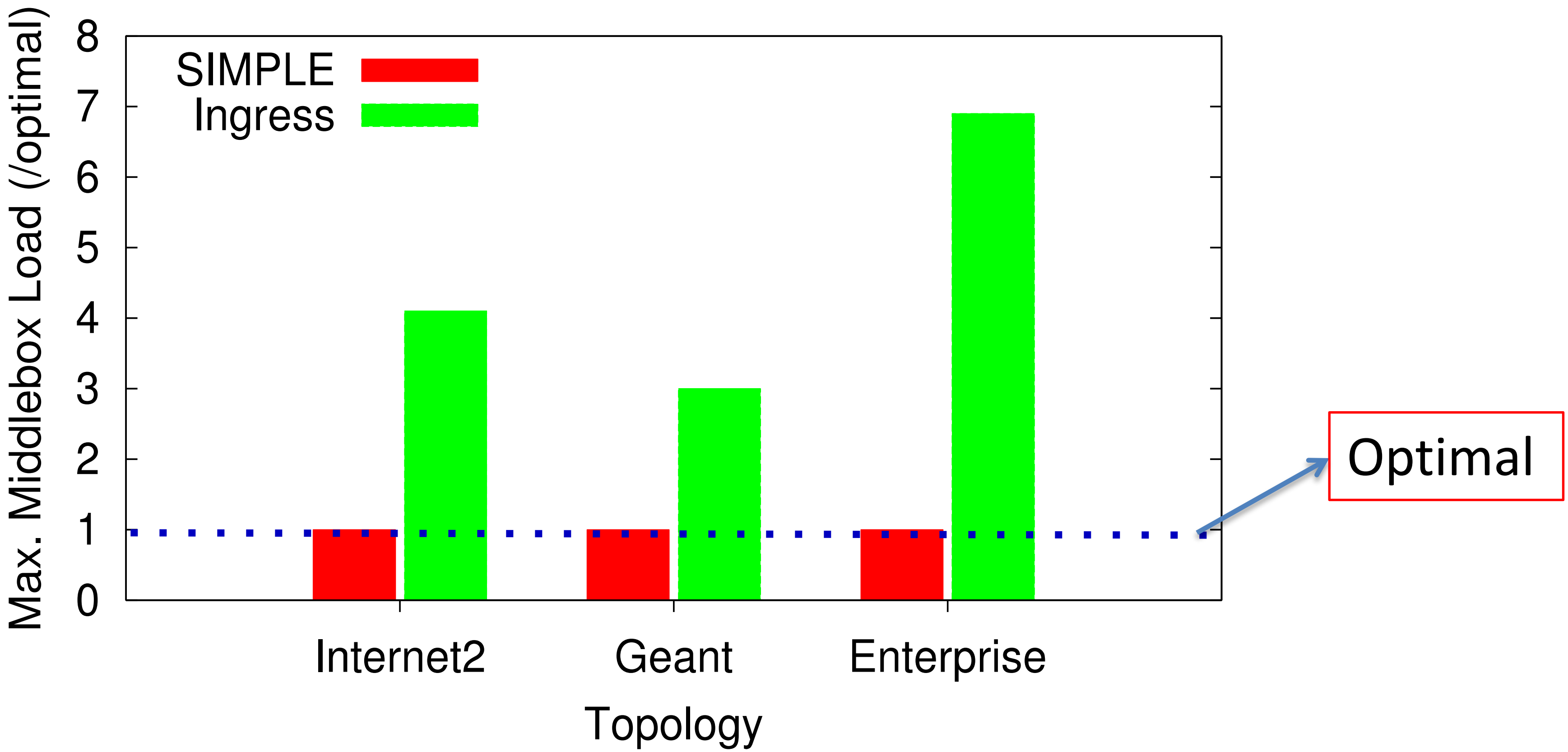
# Outline

- Motivation + Context for the Work
- Challenges
- SIMPLE Design
- ***Evaluation***
- Conclusion

# Evaluation and Methodology

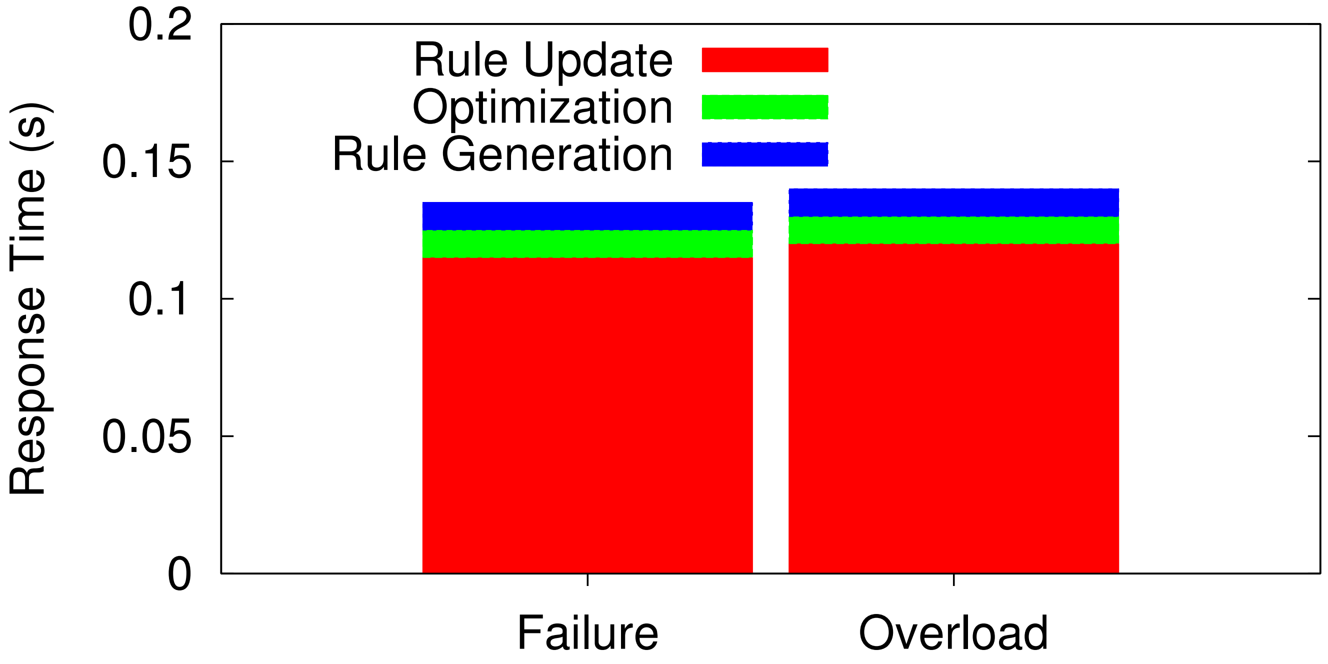
- What benefits SIMPLE offers? load balancing?
- How scalable is the SIMPLE optimizer?
- How close is the SIMPLE optimizer to the optimal?
- How accurate is the dynamic inference?
- Methodology
  - Small-scale real test bed experiments (Emulab)
  - Evaluation over Mininet (with up to 60 nodes)
  - Large-scale trace driven simulations (for convergence times)

# Benefits: Load balancing



4-7X better load balancing and near optimal

# Overhead: Reconfiguration Time



33 node topology including 11 switches

Around 125 ms to reconfigure, most time spent in pushing rules

# Other Key Results

- LP solving takes 1s for a 252 node topology
  - 4-5 orders of magnitude faster than strawman
- 95 % accuracy in inferring flow correlations
- Scalability of pruning: 1800s → 110s

# Conclusions

- Middleboxes: Necessity and opportunity for SDN
- Goal: Simplify middlebox-specific policy enforcement
- Challenges: Composition, resource constraints, modifications
- SIMPLE: policy enforcement layer
  - Does not modify middleboxes
  - No changes to SDN APIs
  - No visibility required into the internal of middleboxes
- Scalable and offers 4-7X improvement in load balancing