Abstractions for Middleboxes

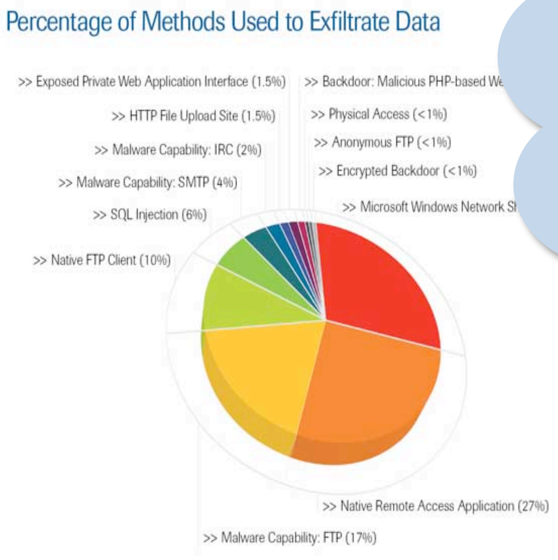
Vyas Sekar
Intel Labs
→ StonyBrook

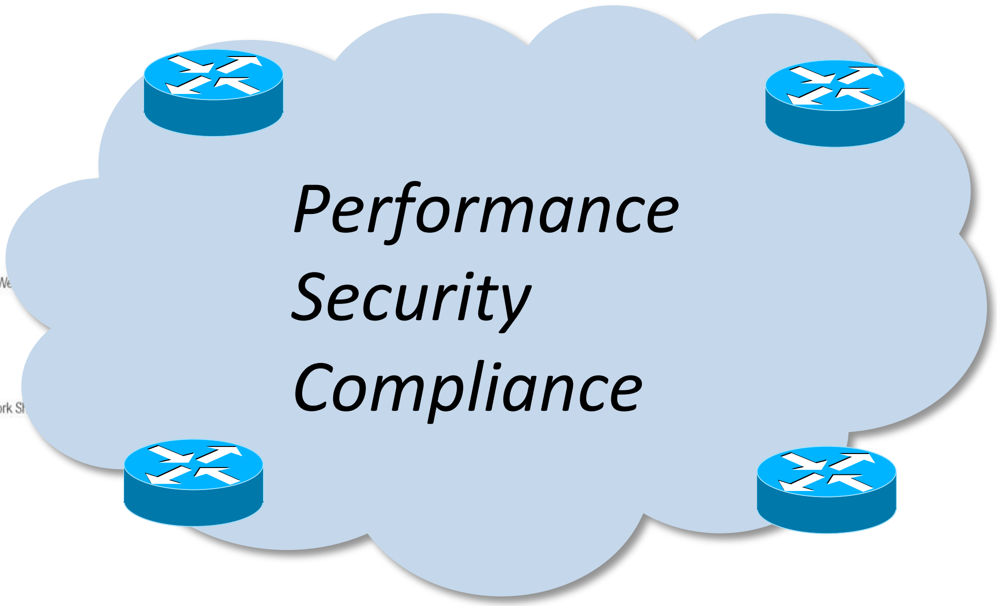
Sylvia Ratnasamy
UC Berkeley

Need for In-Network Functions

Changing applications

Evolving threats





Policy constraints

New devices

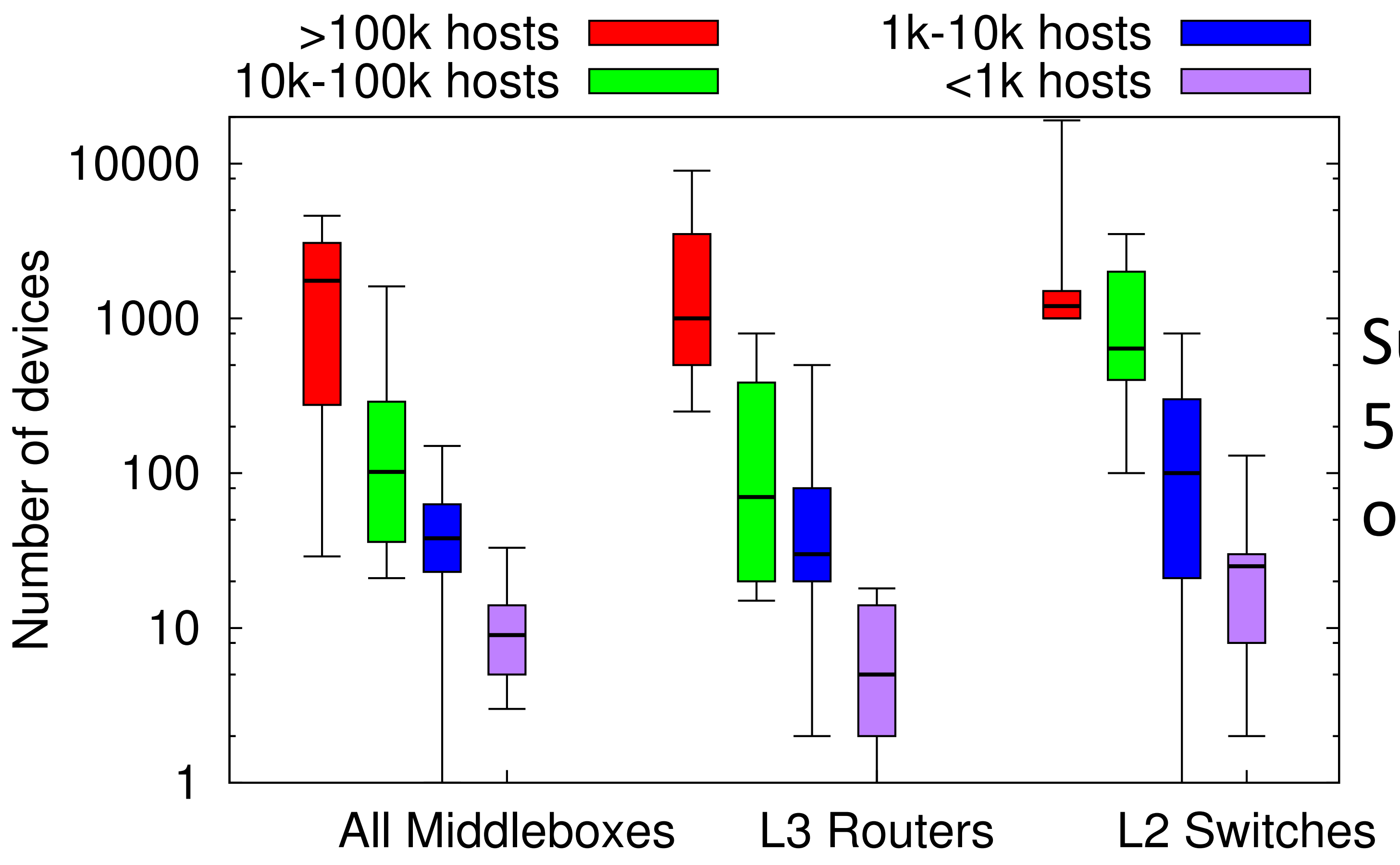
Middleboxes Galore!

Lixia Zhang: "any intermediary device performing functions other than the normal, standard functions of an IP router on the datagram path between a source host and destination host"

Type of appliance	*Number*
Firewalls	166
NIDS	127
Media gateways	110
Load balancers	67
Proxies	66
VPN gateways	45
WAN Optimizers	44
Voice gateways	11
Total Middleboxes	***636***
Total routers	***~900***

Data from a large enterprise:
>80K users across tens of sites

Middleboxes everywhere



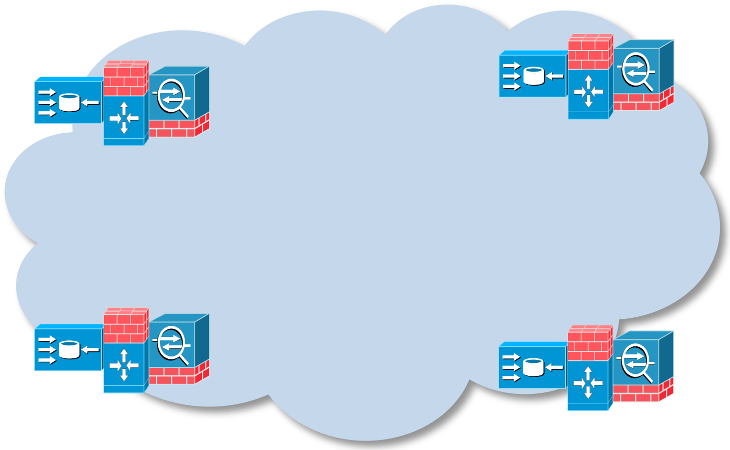
Survey across 57 network operators

Middleboxes are a critical part of the network

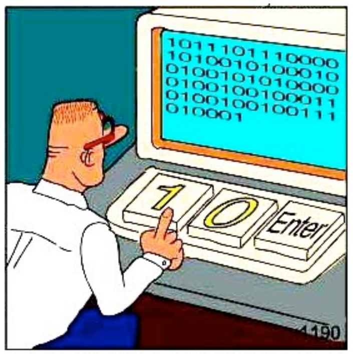
Valuable, but “pain points” for everyone

Network Operators

- Cost, Sprawl
- OpEx
- Inflexible
- Can’t monetize (ISP)



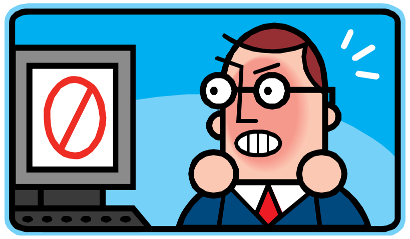
Middlebox Architects



REAL Programmers code in BINARY.

Lack high-level primitives

- Opaque black boxes
- Can’t request services



Users & Researchers

Evolution of the Middlebox Debate

Denial (they shouldn't exist)

→ Acceptance (live with/workaround them)

This is how network innovation occurs!

How can we learn from and extend this success?

What abstractions do we need?
For Operators, Users, Architects

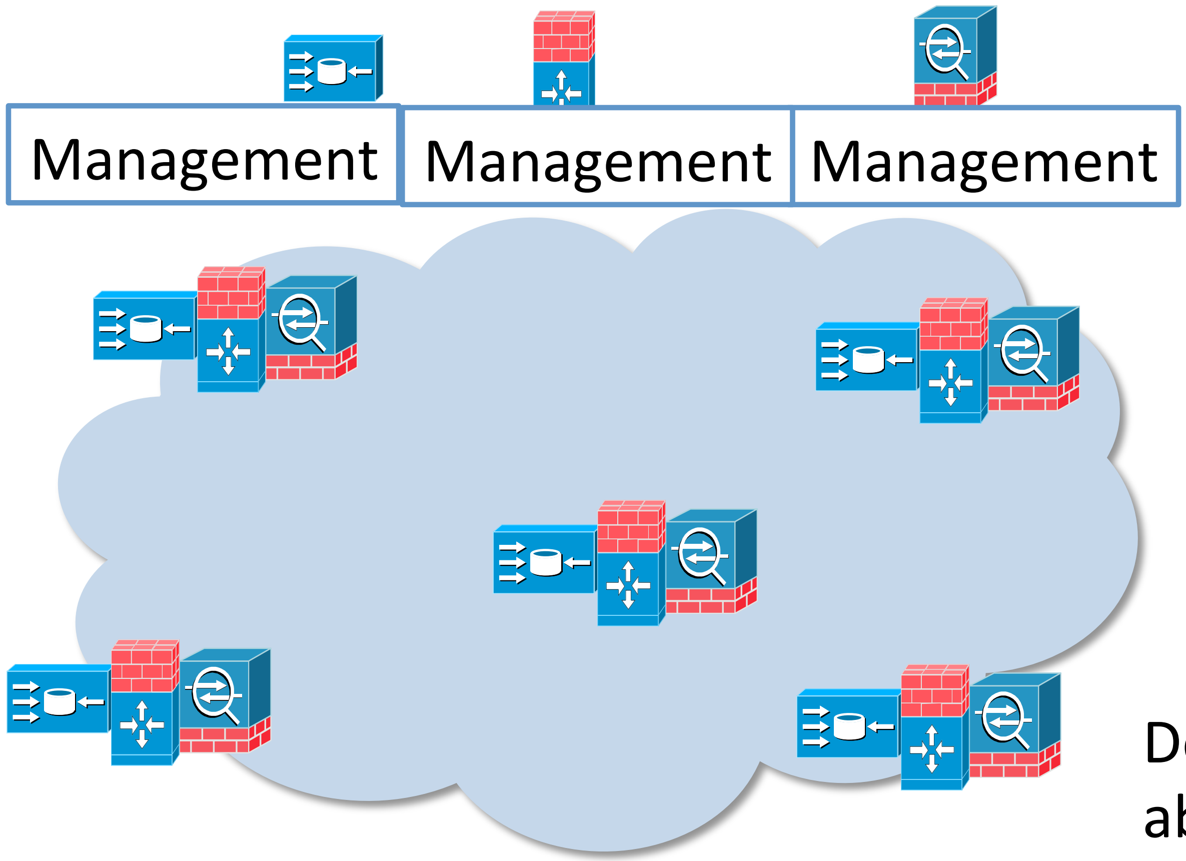
→ *Build, manage middleboxes?*

→ *Leverage the capabilities?*

Outline

- Overview
- ***Abstractions for Operators***
- Abstractions for Users
- Abstractions for Architects
- Synergies and Discussion

Operator View Today

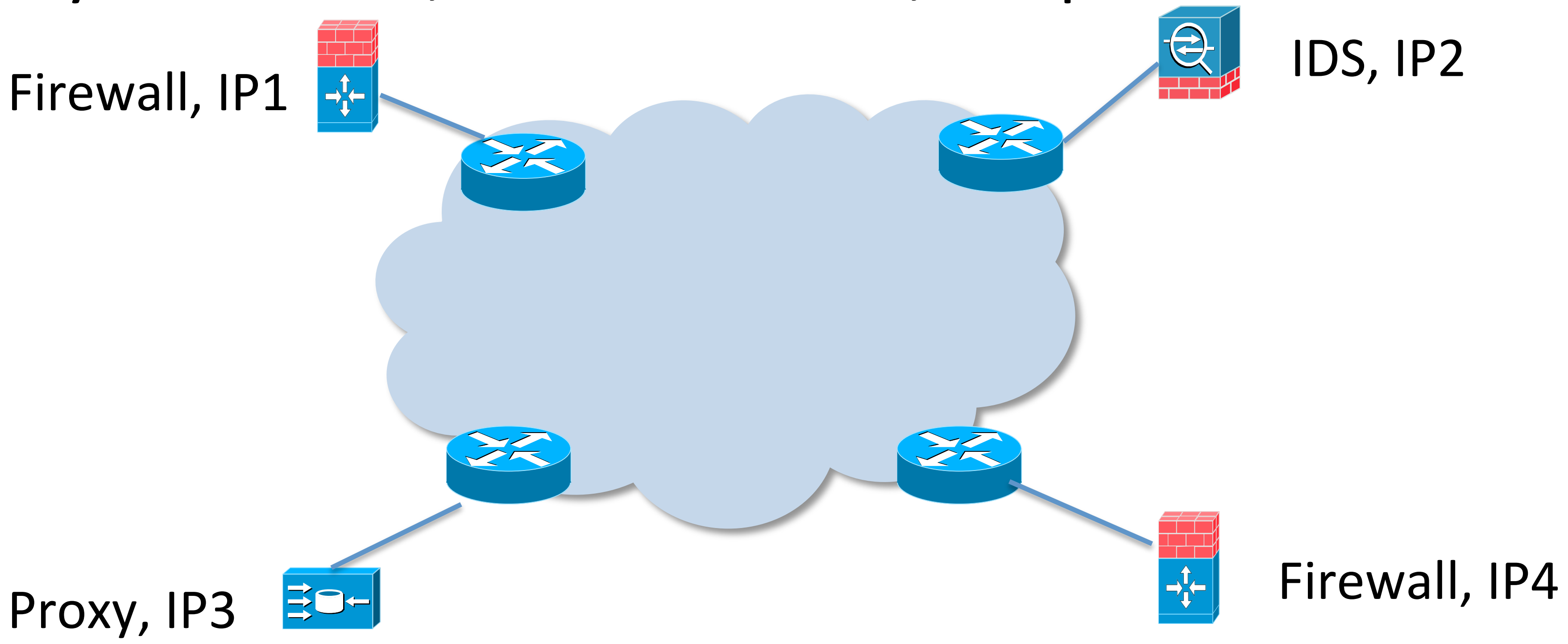


Narrow management interfaces

Device-centric abstractions

Operator View Today

Physical boxes, named with IP, coupled to locations



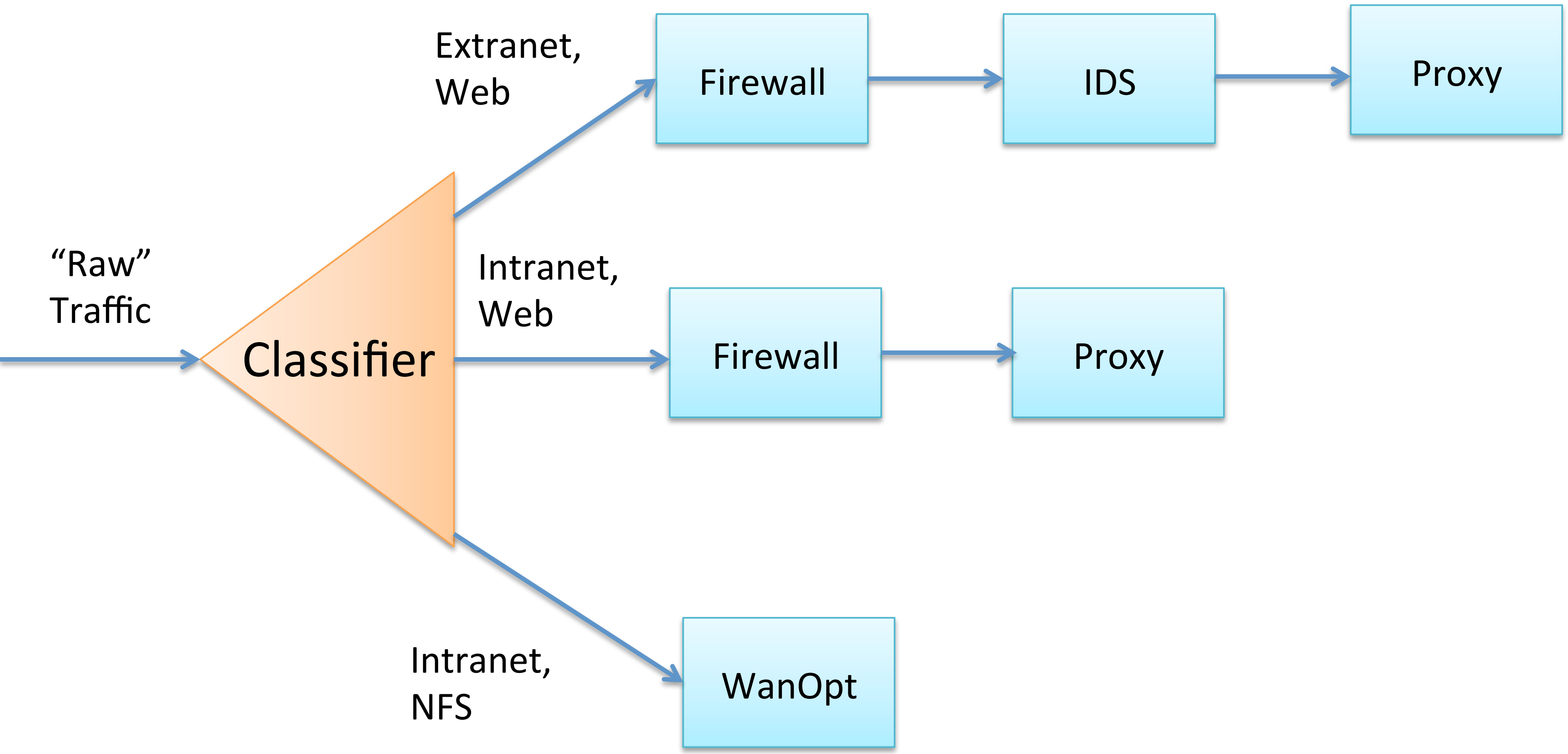
e.g., HTTP needs *Firewall* → *IDS* → *Proxy*

-- Complex, Manual, "Physical" coupling

-- Correctness is hard to verify

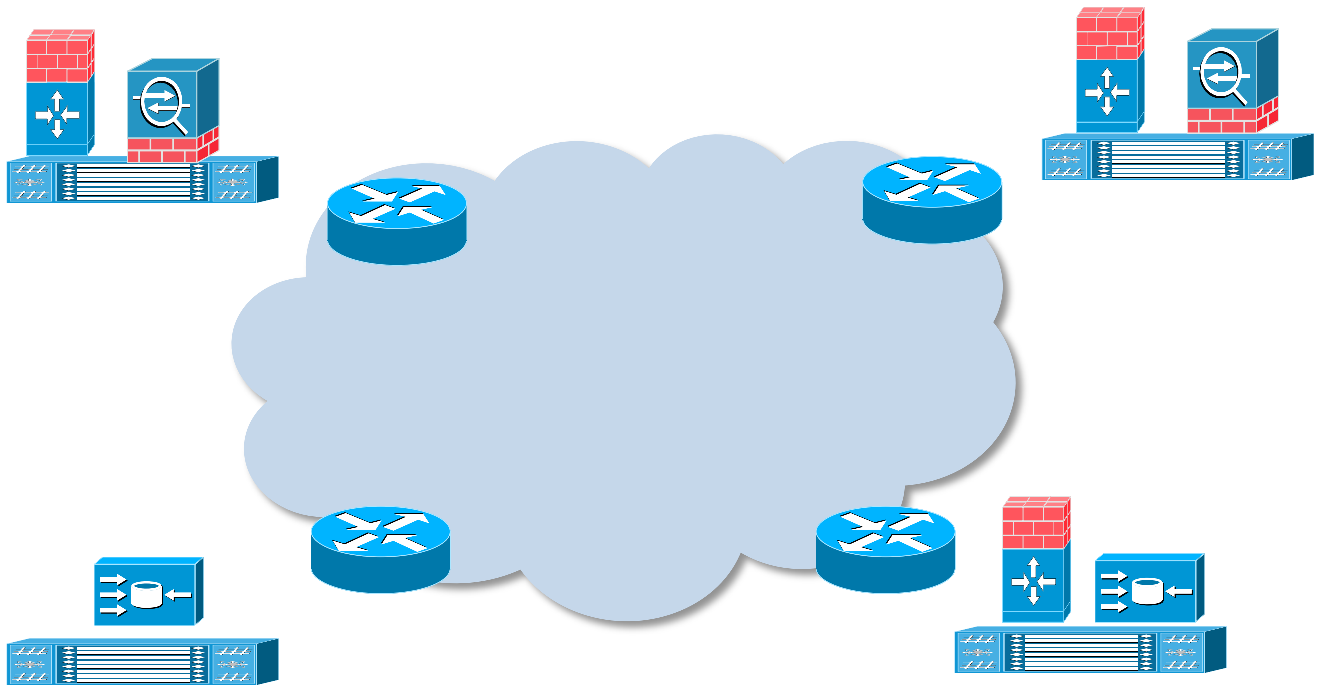
Logical view: "DataFlow" Abstraction

Specify "what" processing, not where/how



Network-level View

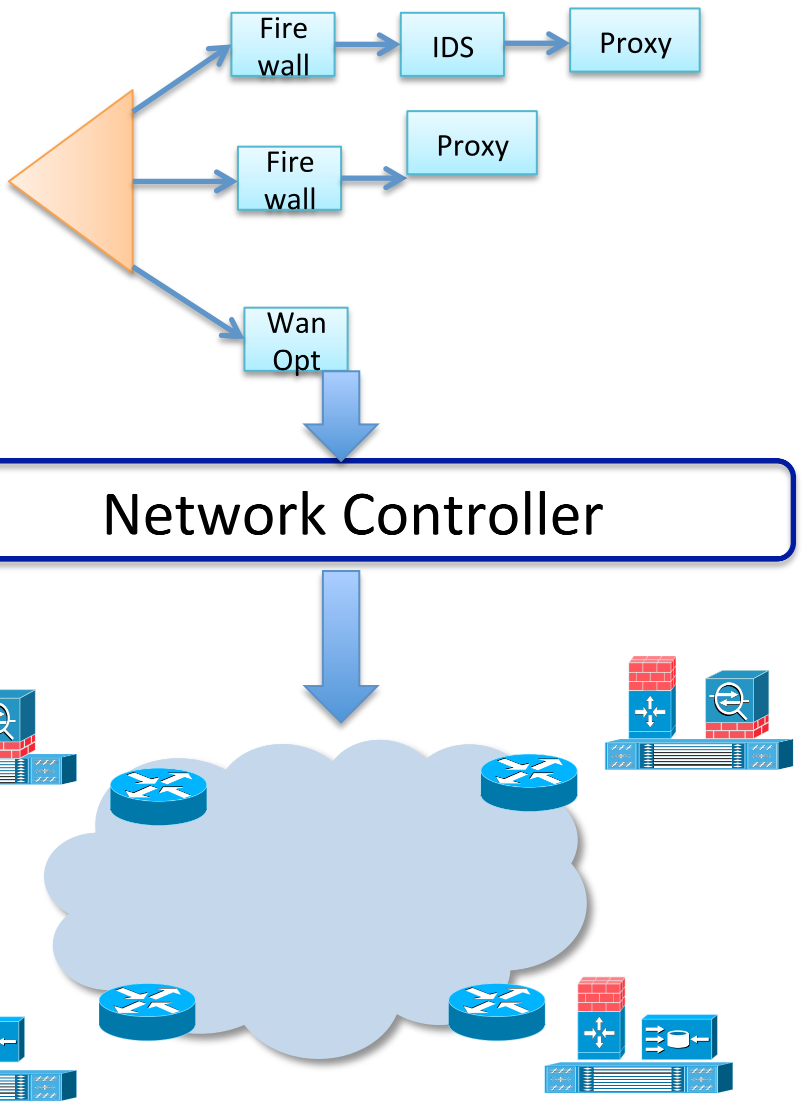
Each location offers some middlebox capability



Some boxes may offer
a subset of capabilities

Tie-in with SDN world

Logical View

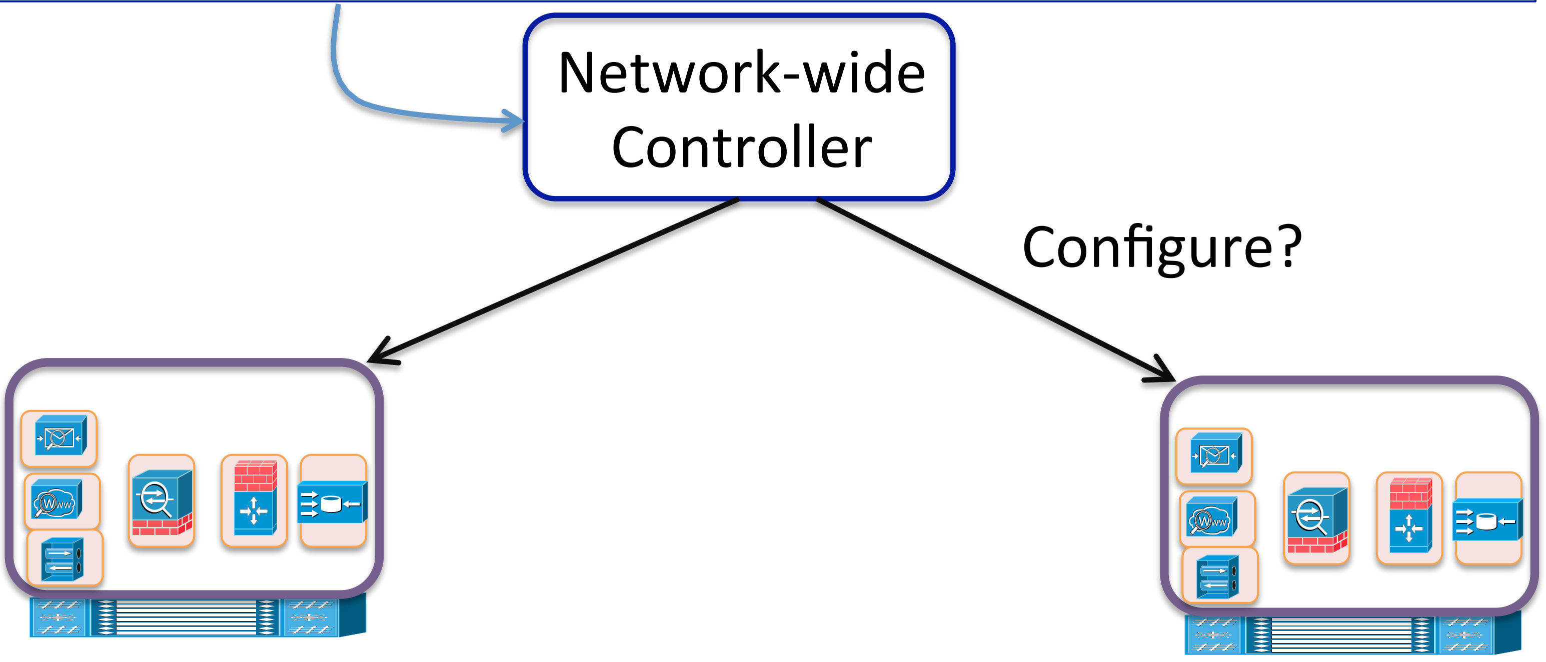


e.g., NOX, 4D, Maestro, RCP

Physical View

Control Plane for Middleboxes

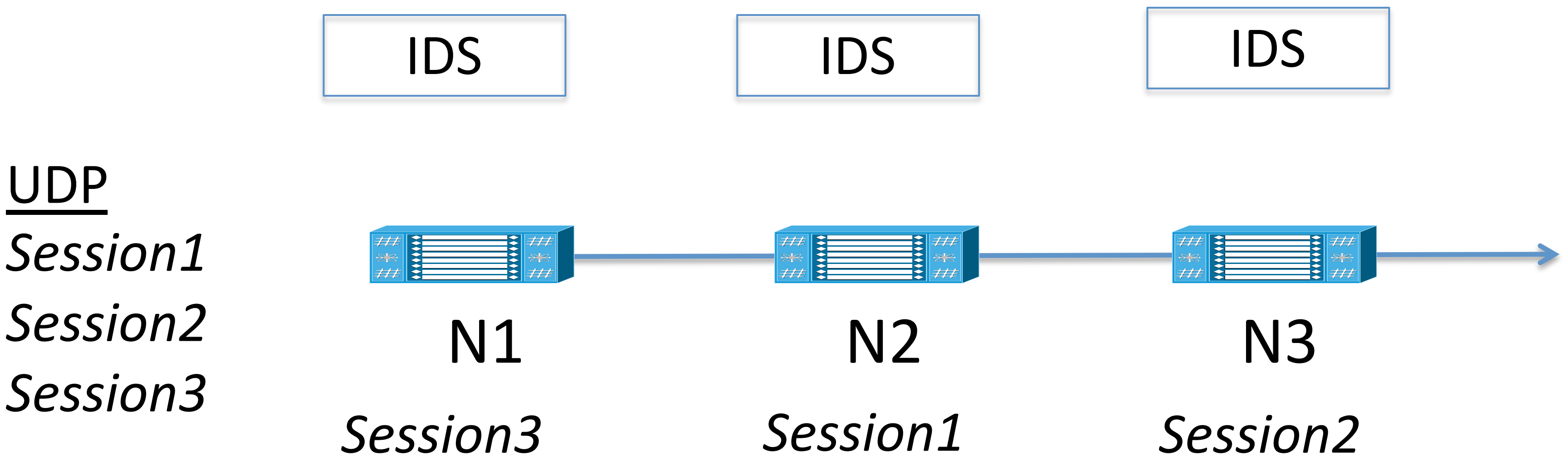
Existing work: Homogeneous routing-like workload



Middleboxes: complex, heterogeneous
→ Policy constraints, resource management
→ New challenges and opportunities

Policy: Coverage Requirement

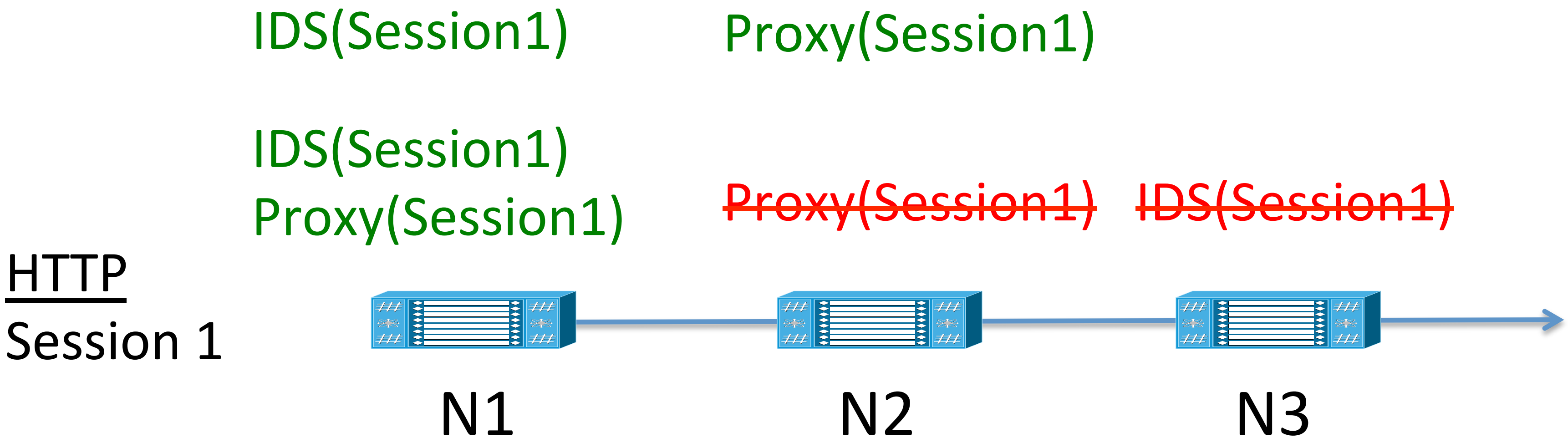
***Coverage**:* For each UDP session, some node on path runs IDS



Opportunity: Flexibility in "placement"

Policy: Ordering Constraints

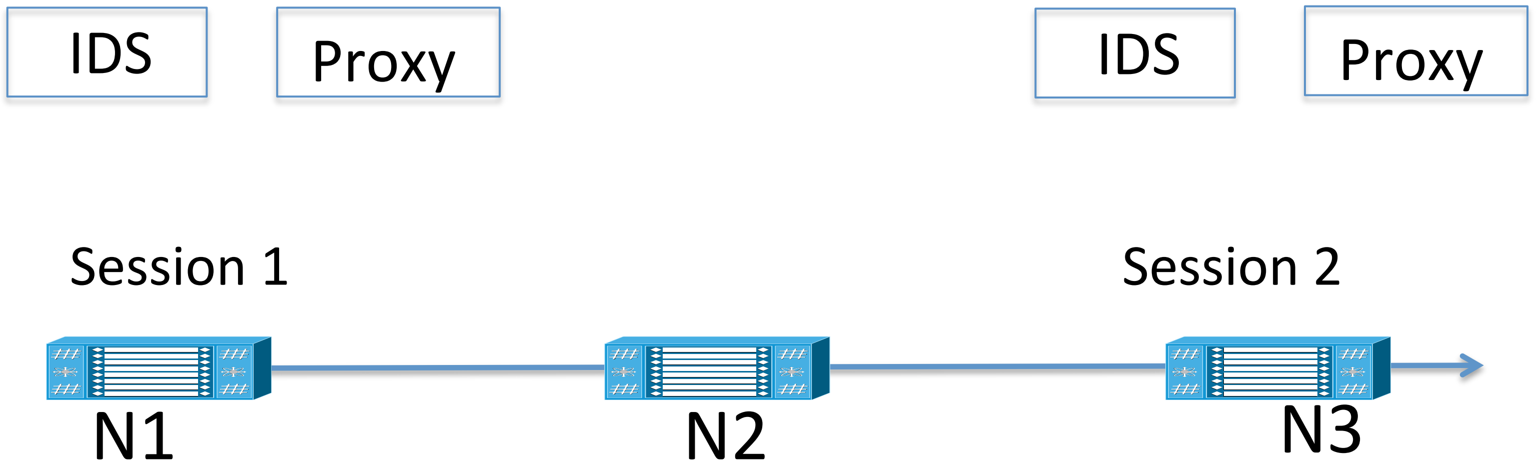
Policy Ordering:
For each HTTP session, run IDS before running proxy



Resource Requirements and Load

HTTP
IDS < Proxy
N1 → N3

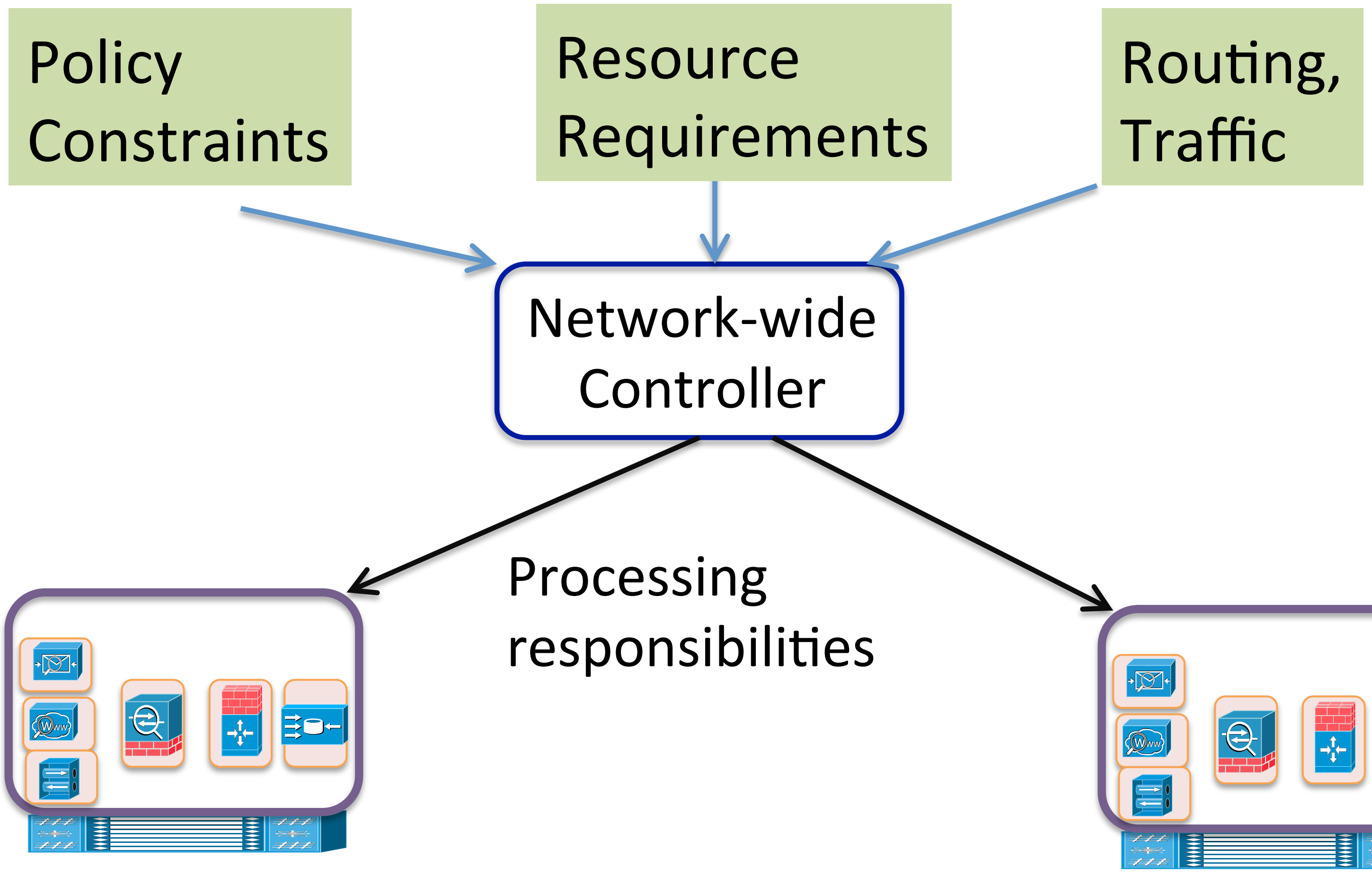
Session1
Session2



Load depends on which sessions/actions are assigned to each node

Provisioning and Load Balancing

Control Plane for Middleboxes



New components: Packet steering, Provisioning, Placement

Outline

- Overview
- Abstractions for Operators
- ***Abstractions for Users***
- Abstractions for Architects
- Synergies and Discussion

State of the world

Middleboxes are a black-box

Almost no abstraction to end users

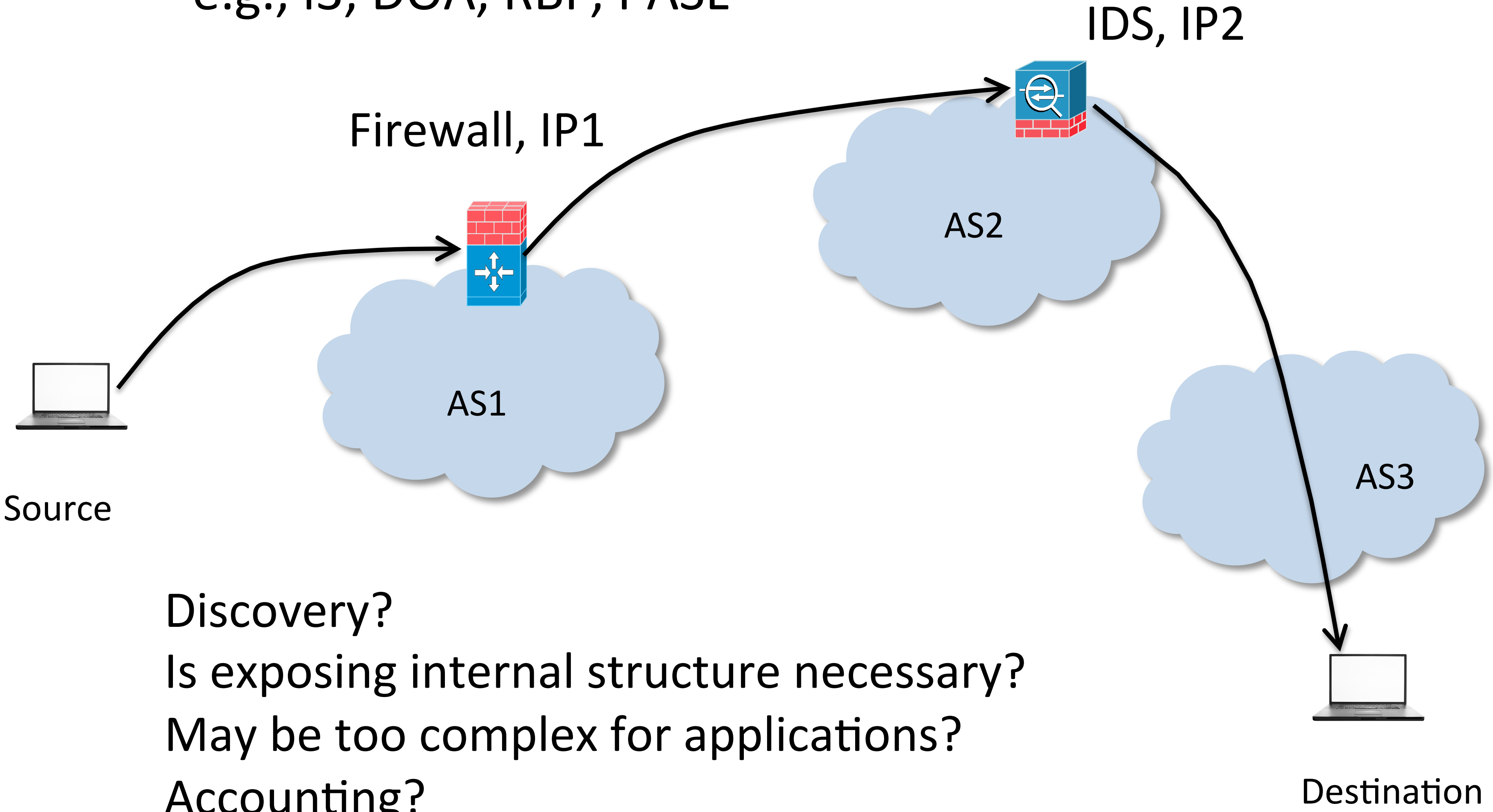
→ Can't get "on-demand" services

→ ISPs can't offer such services

Waypoint Abstraction

Explicitly route via middlebox IP(s)

e.g., i3, DOA, RBF, PASL



Discovery?

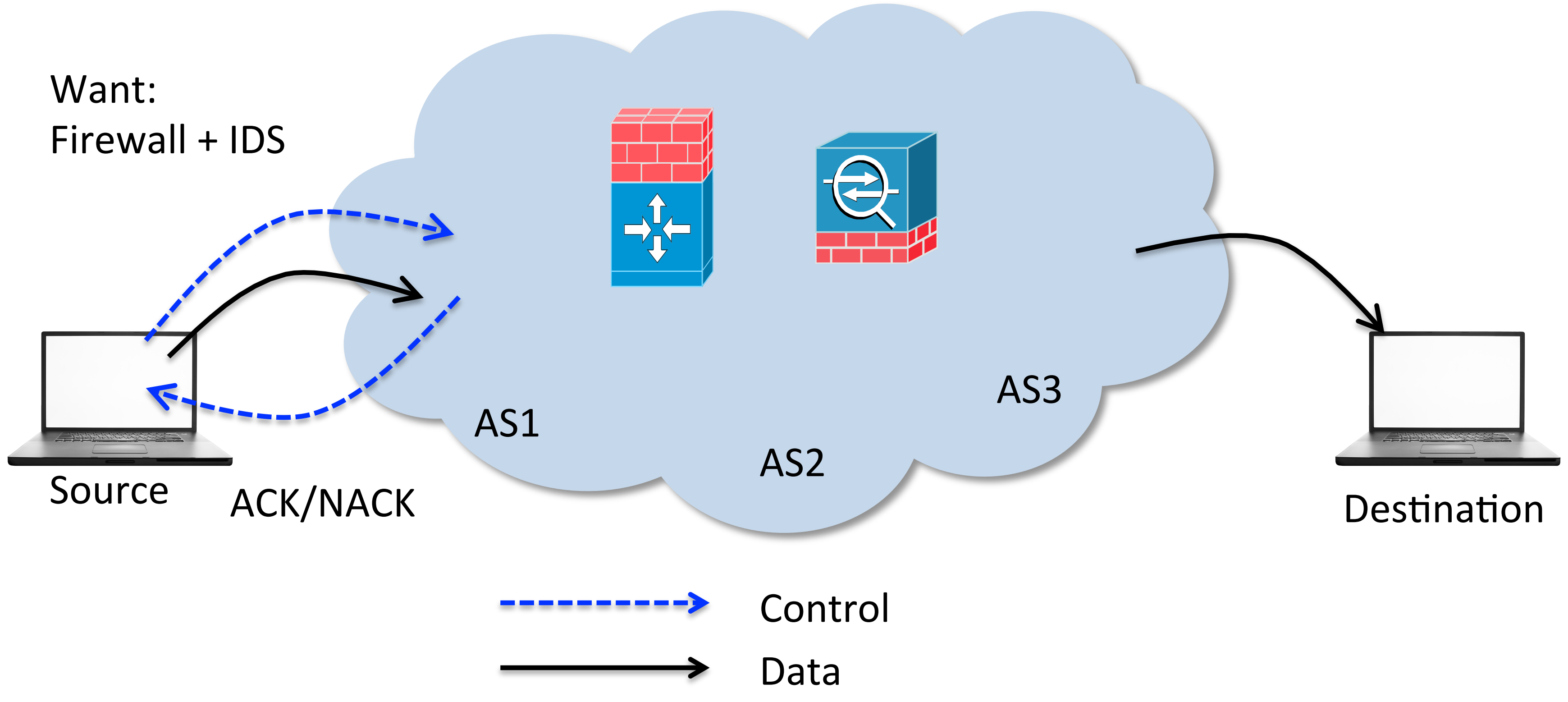
Is exposing internal structure necessary?

May be too complex for applications?

Accounting?

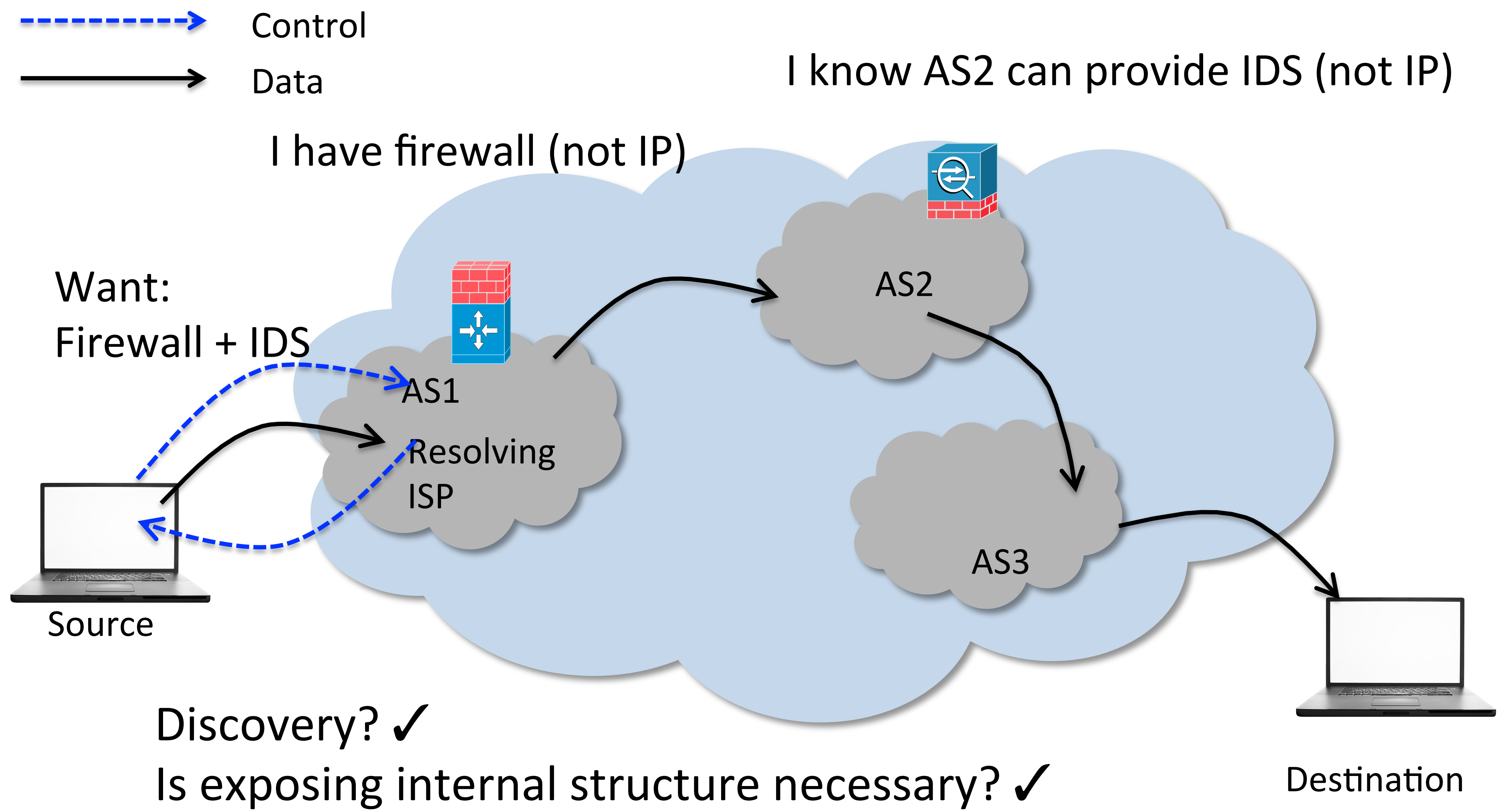
Proposal: Treat as "Service"

Single logical network providing processing service



Abstract away "Where in network" this processing occurs

Service Resolution



Discovery? ✓
Is exposing internal structure necessary? ✓
May be too complex for applications? ✓
Accounting? ✓

Tradeoffs vs. Waypoint Abstraction

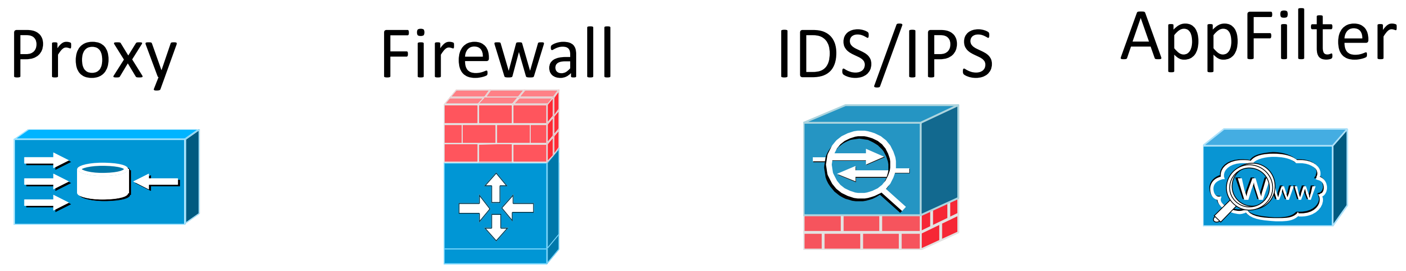
- Pros
  - Accounting is simple
    - User only pays resolving ISP akin to today's world
    - ISPs "peers" with each other
  - Control/Data decoupling
    - Data plane/Packet formats are unmodified
  - Designed for partial/incremental deployment
    - Forces apps to think of "best-effort"

- Cons
  - State in the network
    - E.g., tunnels between the middleboxes at ISPs

Outline

- Overview
- Abstractions for Operators
- Abstractions for Users
- ***Abstractions for Architects***
- Synergies and Discussion

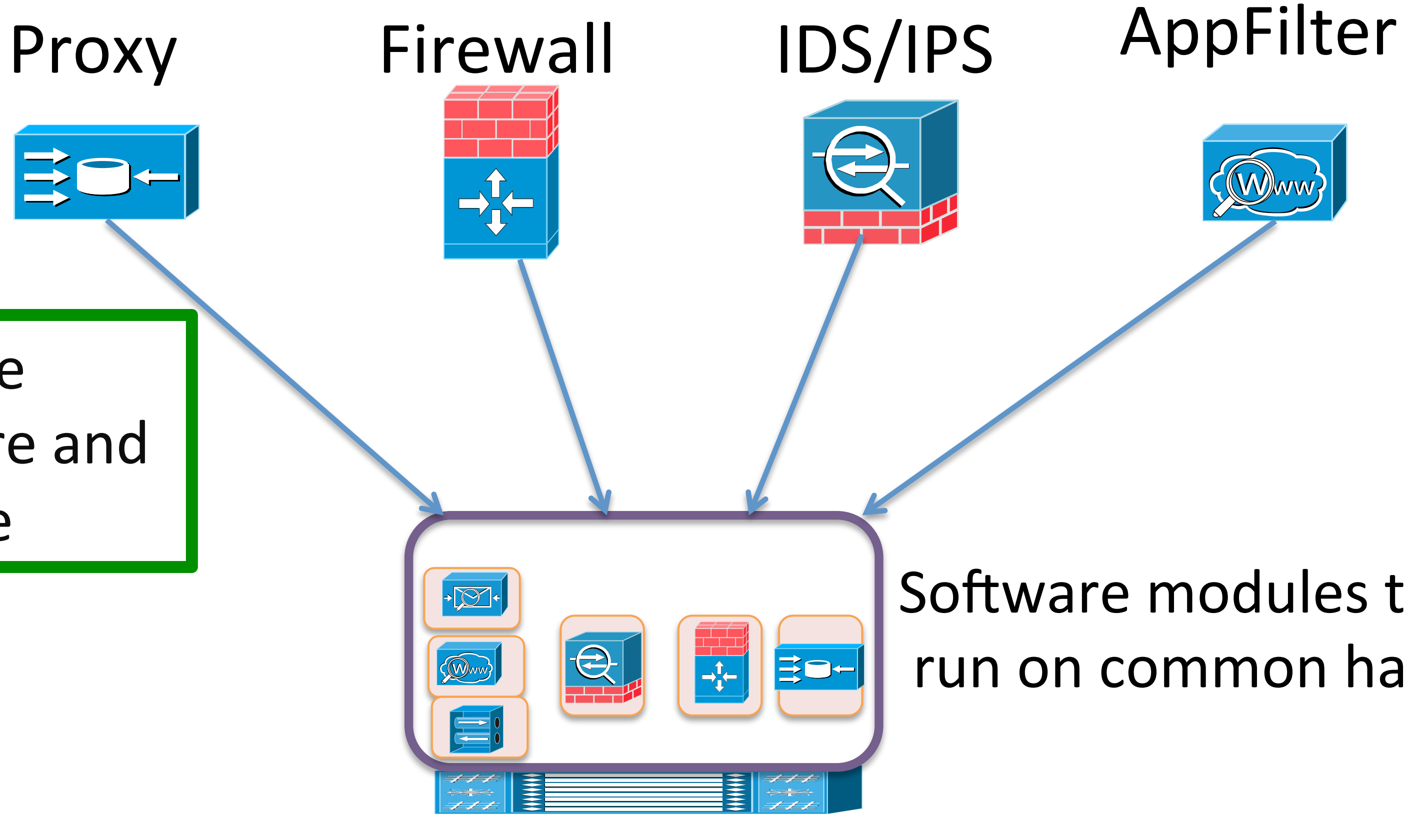
State of the world



Today: Independent, specialized boxes
Vertically integrated stacks
Custom software and/or hardware

Problem: Cost, Sprawl, Inflexible

Proposal: Treat as "Computation"

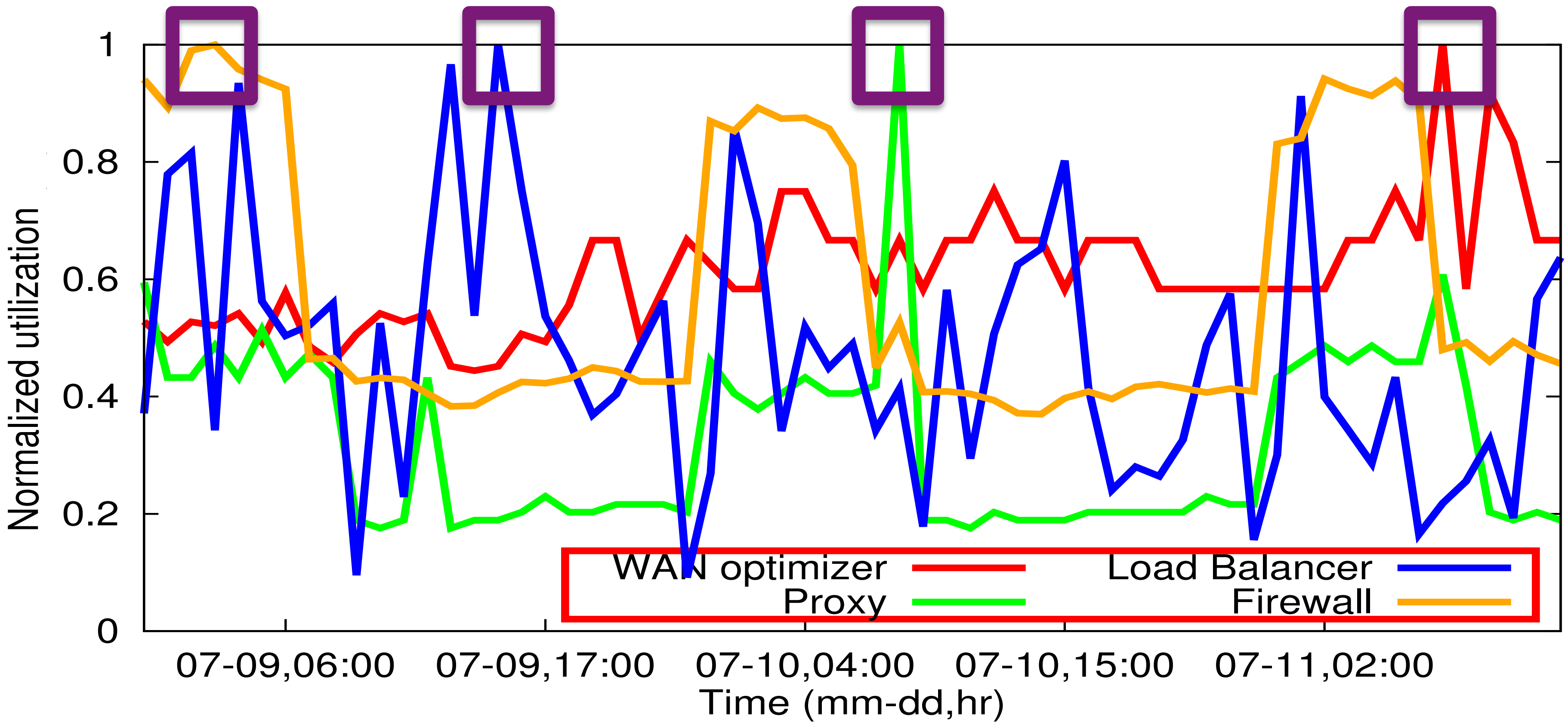


Decouple Hardware and Software

Software modules that can run on common hardware

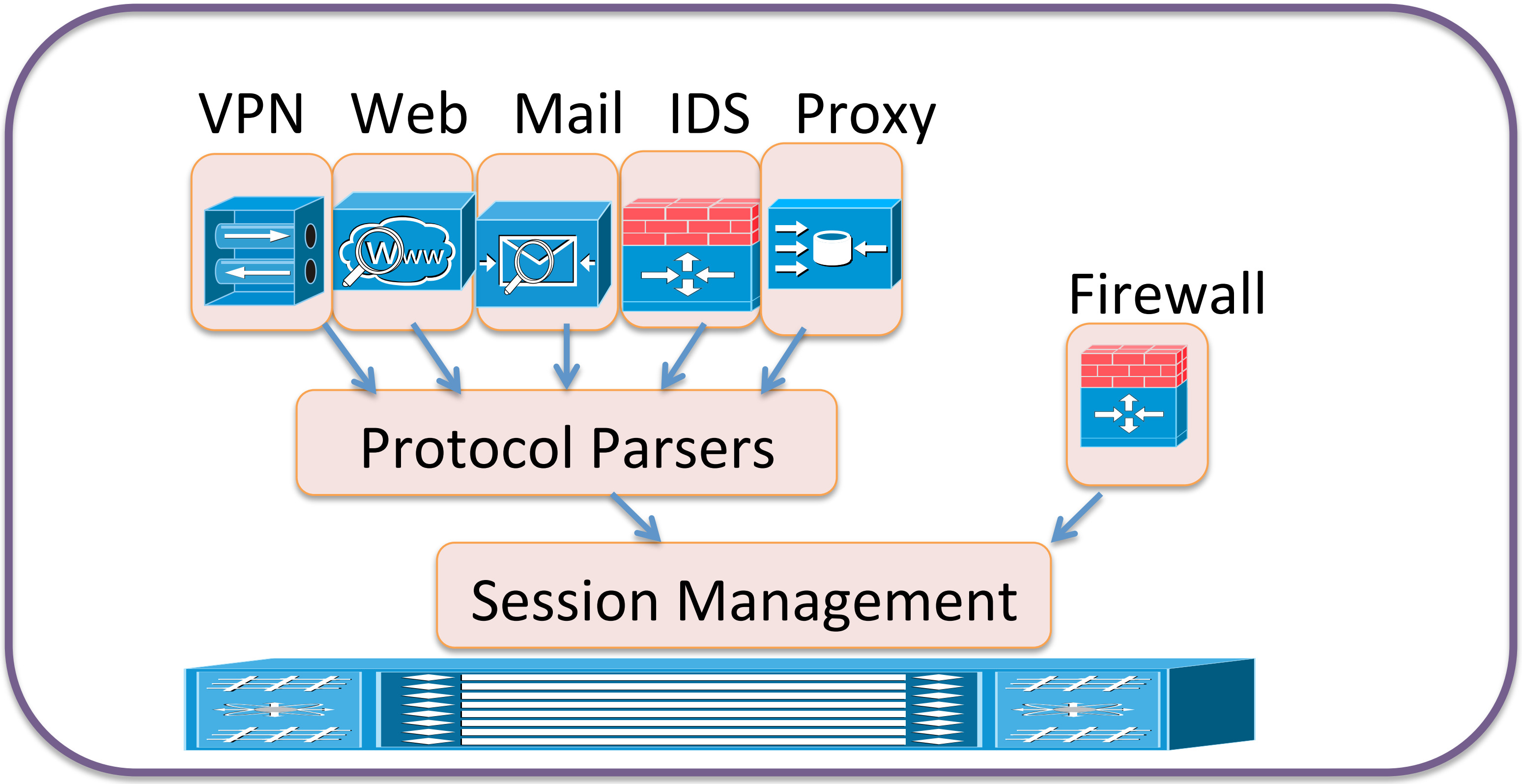
Enables Consolidation, Multiplexing, Extensibility

Reduces CapEx via Multiplexing



Multiplexing benefit = Max_of_TotalUtilization / Sum_of_MaxUtilizations

Extensible Software Stack



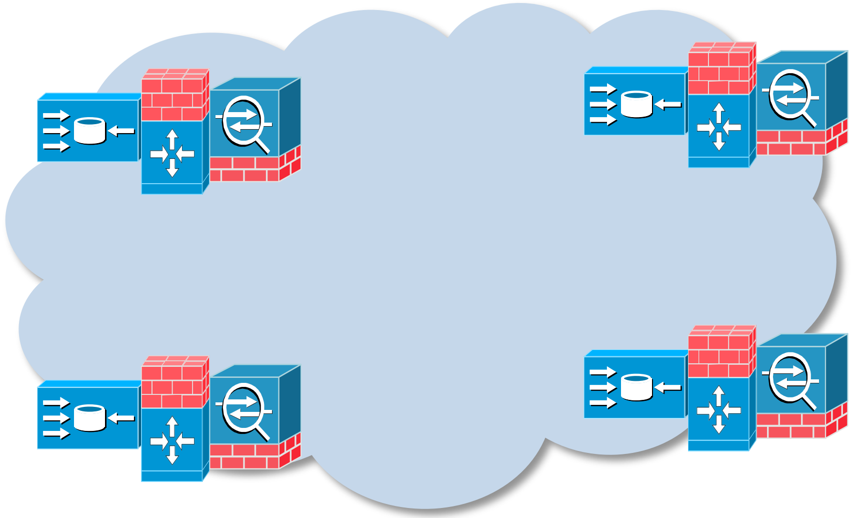
Contribution of reusable modules: 30 – 80 %

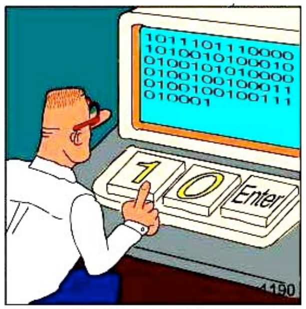
Outline

- Overview
- Abstractions for Operators
- Abstractions for Users
- Abstractions for Architects
- ***Synergies and Discussion***

Proposed abstractions

Operators:
Dataflow

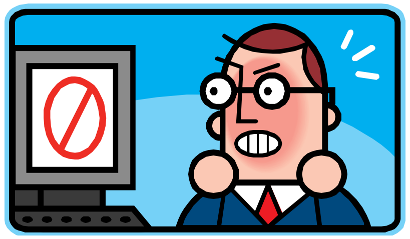




REAL Programmers code in BINARY.

Architects:
Computation

Users:
Service



Synergy between abstractions

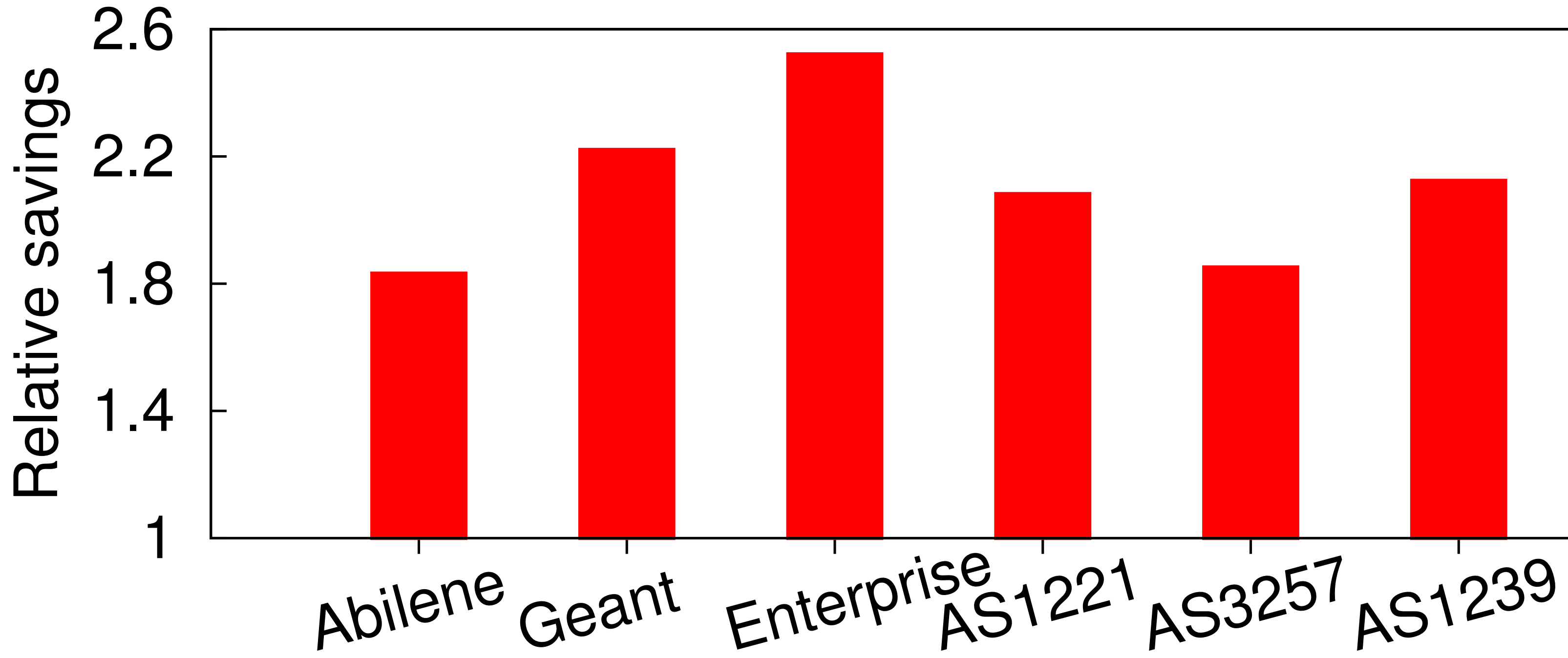
- Dataflow + Computation → Run *anywhere*
  - More flexible
- Computation + Service → *Anyone* can run this
  - Lowers barrier of entry for providers
  - New opportunities for monetization for ISPs
- Computation + Service → Economies of scale
  - Benefit of "cloud"

Discussion and Open Issues

- Operator-level:
  - Should we make middleboxes SDN-aware?
  - Does middlebox internal state need to be exposed?
- User-level:
  - Tussle between users and operators?
  - Applications vs ISP economic tension?
- Middlebox Architects:
  - Specialized hardware: Clean way to incorporate?
  - Multiplexing different vendors: Isolation vs Reuse?

Reduction in Provisioning Cost

$\text{Provisioning}_{\text{Today}} / \text{Provisioning}_{\text{Centralized}}$

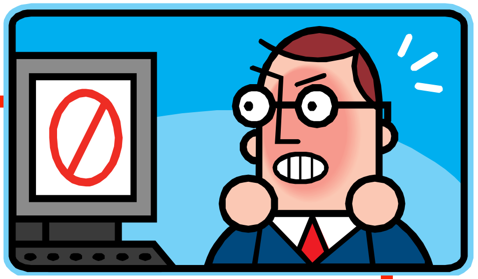


Centralized approach reduces provisioning cost 1.8-2.5X

Pain points for Operators

- High CapEx
  - Specialized solutions
  - Custom hardware
- Device Sprawl
  - Many "point" solutions
- High OpEx
  - Narrow interfaces
  - Manual tuning
- Long upgrade cycles (3--5 yrs)
- Can't effectively monetize (ISP)

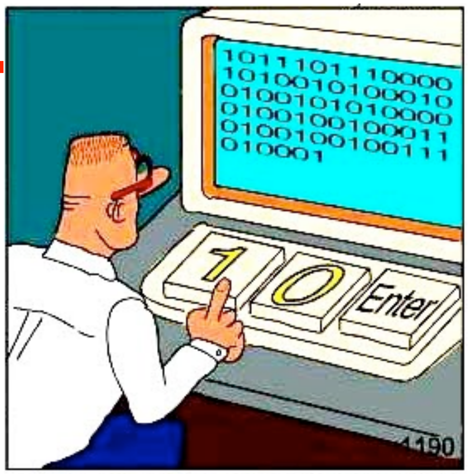
Pain points for Users and Researchers



- Opaque
  - Can't predict what processing occurs
  - "Tussle" vs. operators
- Can't request services on-demand
  - E.g., Site wants DDoS protection
  - E.g., Netflix wants transcoding
- Research/New designs:
  - Undesirable interactions
  - Can't get new ideas deployed

Pain points for Architects

- Low-level protocol details
  - E.g., fragmentation
  - E.g., session reassembly
  - E.g., HTTP corner cases
- Performance
  - Hardware-specific optimizations
- Long development cycles
  - Slows innovation



REAL Programmers code in BINARY.

Some open questions..

- Do middleboxes need to be SDN-aware?
  - Does that enable new functionality?
  - E.g., dynamically offload to other locations
- Can we handle "dynamic" dataflows?
  - E.g., invoke IDS on suspicious flows on-demand
- How much middlebox internal state does the controller need to understand?
  - E.g., does it need NAT table to setup forwarding?

Opportunities and challenges

- Opportunities
  - Service providers can monetize beyond one-hop
  - Invoke services on-demand
  - Ease some application vs. ISP tension
    - E.g., Netflix
  - Incentivizes deployment (partial/best effort)

- Challenges
  - Placement patterns
    - On-path vs. Perimeter vs. Specific location?
  - Accounting
    - Multi-lateral vs. Bilateral settlements?

Challenges

- Hardware accelerators
- Isolation among co-resident modules

What does this enable?

- Consolidation
  - Reduce device sprawl
- Multiplexing
  - Repurpose hardware resources more efficiently
- Extensibility
  - Reduce development cycles