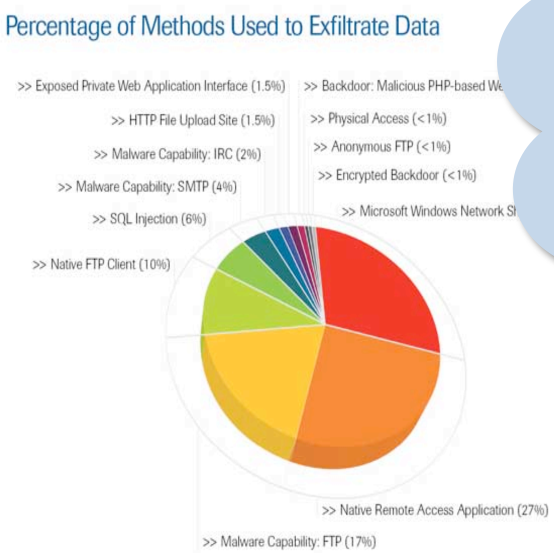
Design and Implementation of a Consolidated Middlebox Architecture

Vyas Sekar Sylvia Ratnasamy Michael Reiter Norbert Egi
Guangyu Shi

Need for Network Evolution

New applications

Evolving threats



Policy constraints

New devices

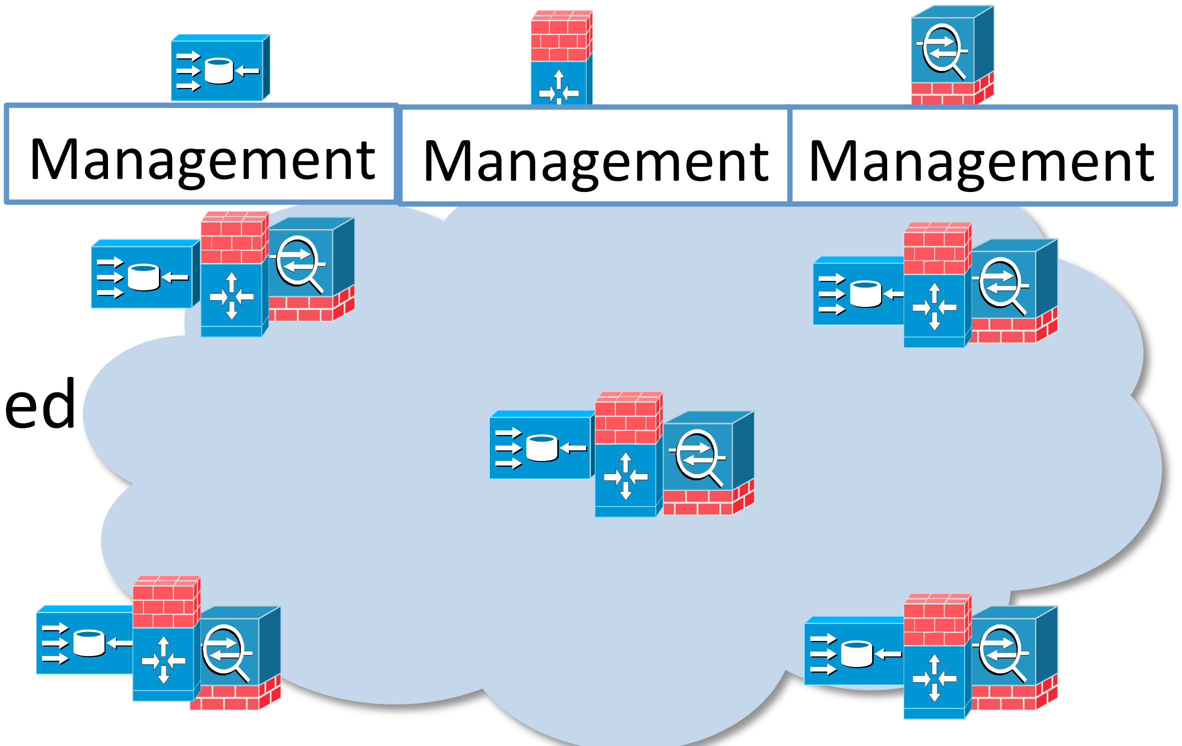
Network Evolution today: Middleboxes!

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

Just network security
$10 billion

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***

Key pain points



Narrow interfaces

Specialized boxes

Point solutions! ➔

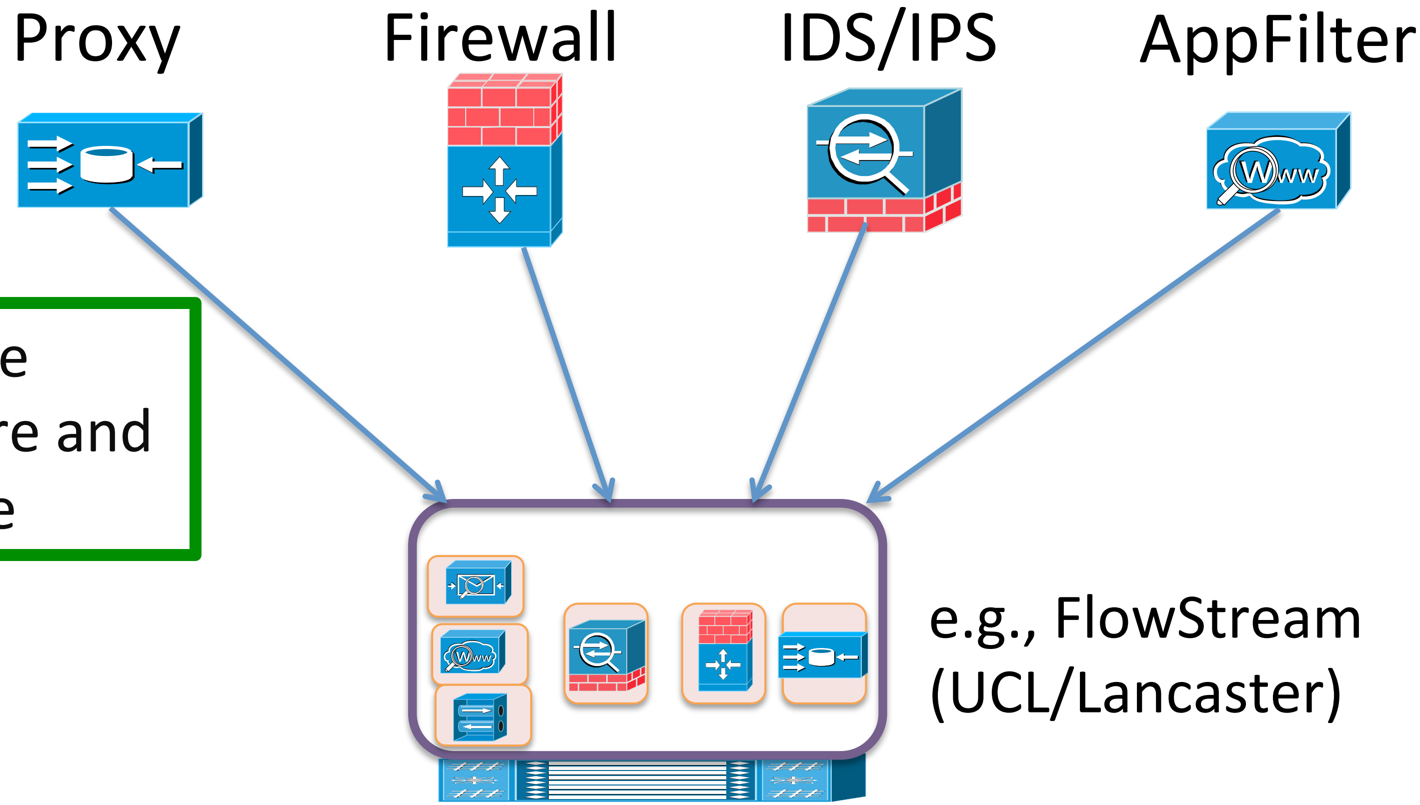
Increases capital expenses & sprawl
Increases operating expenses
Limits extensibility and flexibility

Outline

- Motivation
- *High-level idea: Consolidation*
- System design
- Implementation and Evaluation

Consolidation at Platform-Level

Today: Independent, specialized boxes

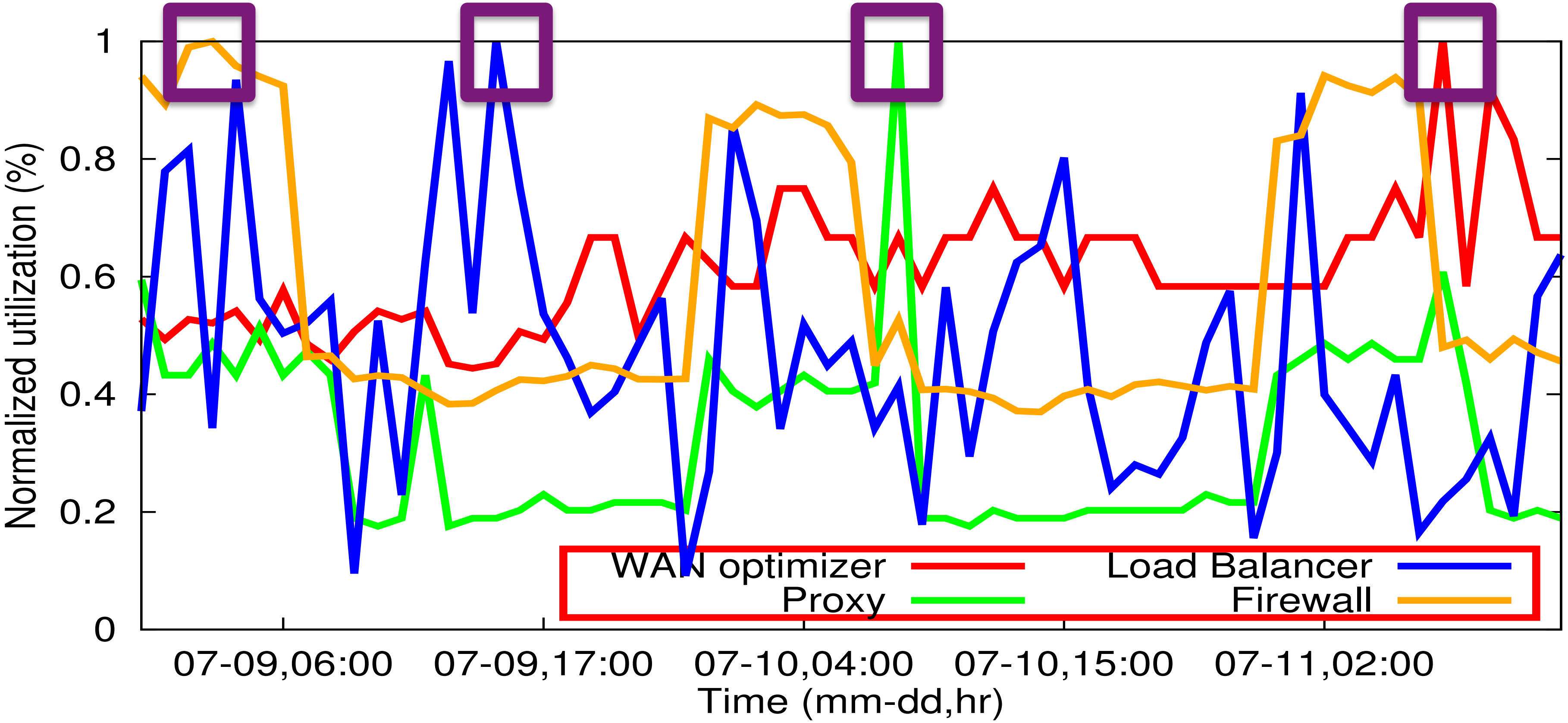


Decouple Hardware and Software

e.g., FlowStream (UCL/Lancaster)

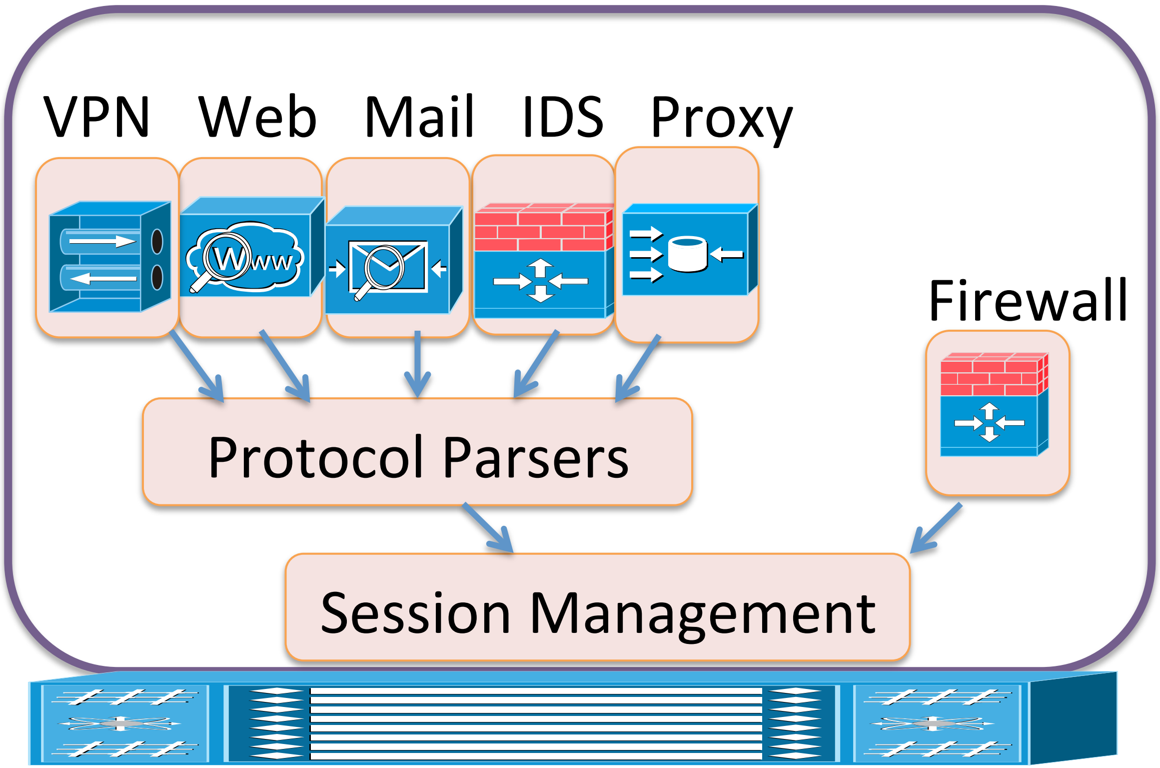
Consolidation reduces capital expenses and sprawl

Consolidation reduces CapEx



Multiplexing benefit = Max_of_TotalUtilization / Sum_of_MaxUtilizations

Consolidation Enables Extensibility



e.g., xOMB (UCSD)
Bro

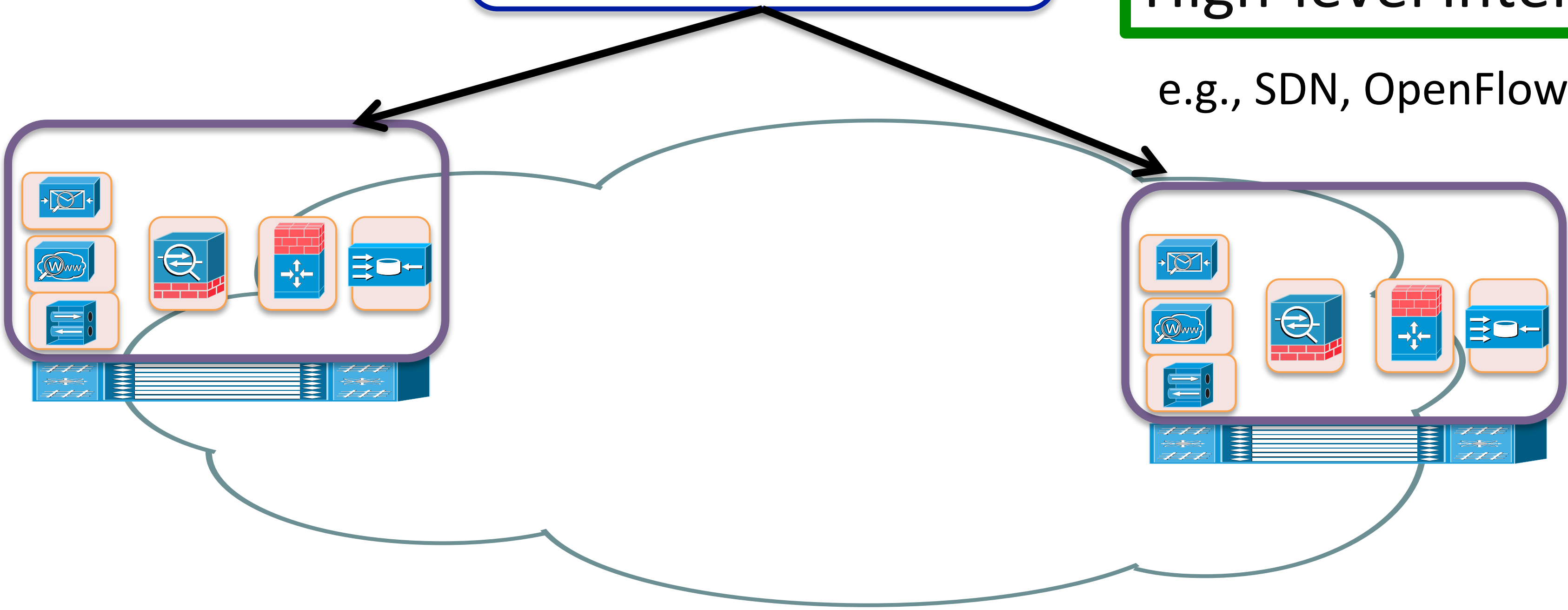
Contribution of reusable modules: 30 – 80 %

Consolidating Management

Network-Wide Management

Logically centralized
High-level interfaces

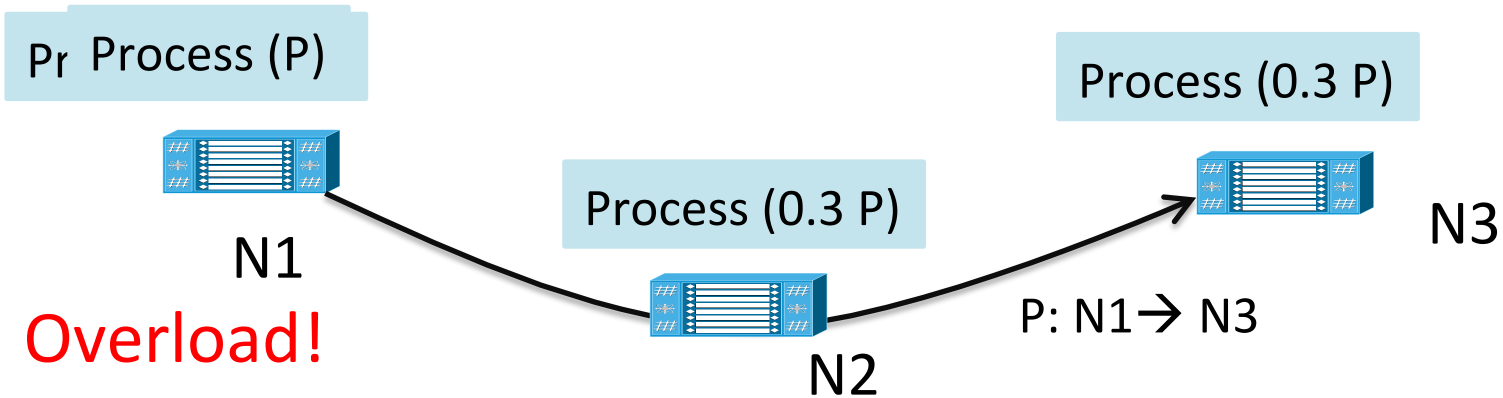
e.g., SDN, OpenFlow, 4D



Simplifies management to reduce operating expenses

Consolidation enables flexible resource management

Today: All processing at logical "ingress"

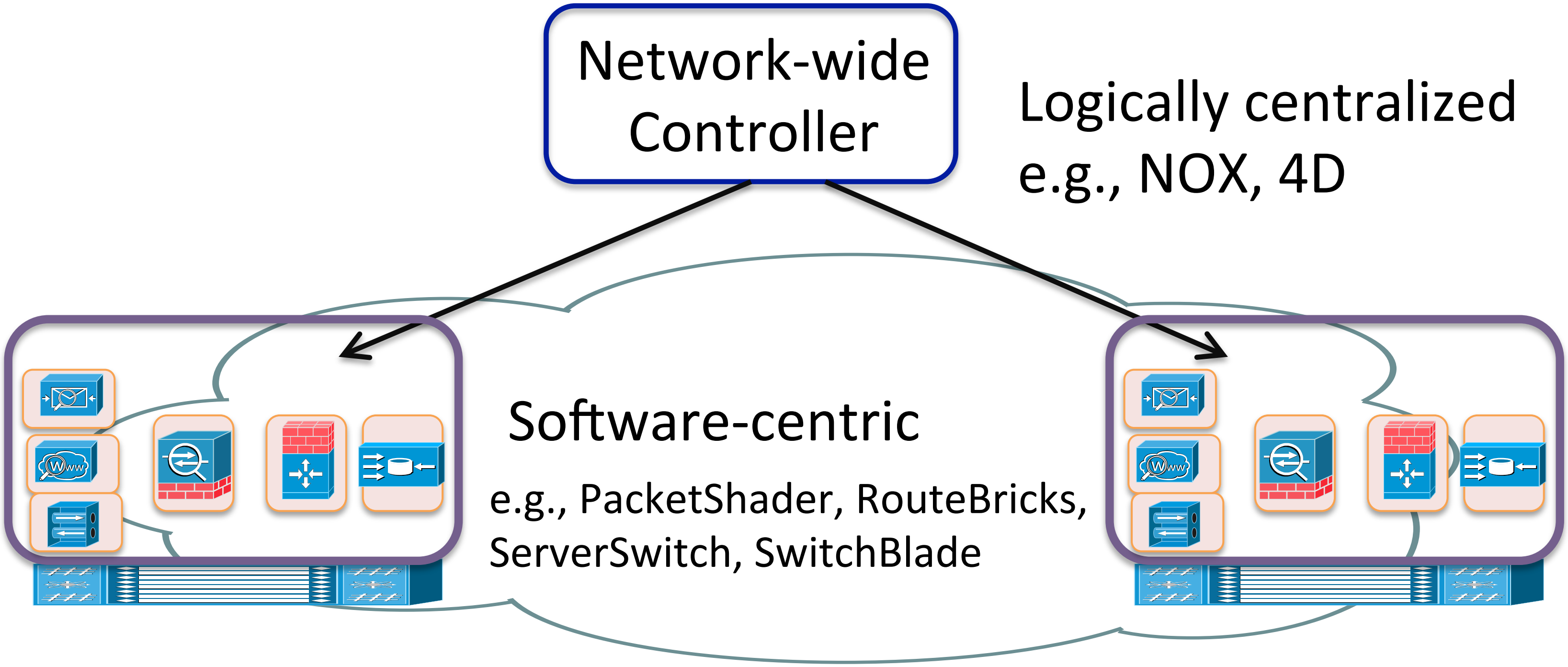


Distribution reduces load imbalance

Outline

- Motivation
- High-level idea: Consolidation
- *CoMb: System design*
- Implementation and Evaluation

CoMb System Overview



Existing work: simple, homogeneous routing-like workload

Middleboxes: complex, heterogeneous, new opportunities

CoMb Management Layer

Goal: Balance load across network

Exploit multiplexing, reuse, distribution

Policy Constraints

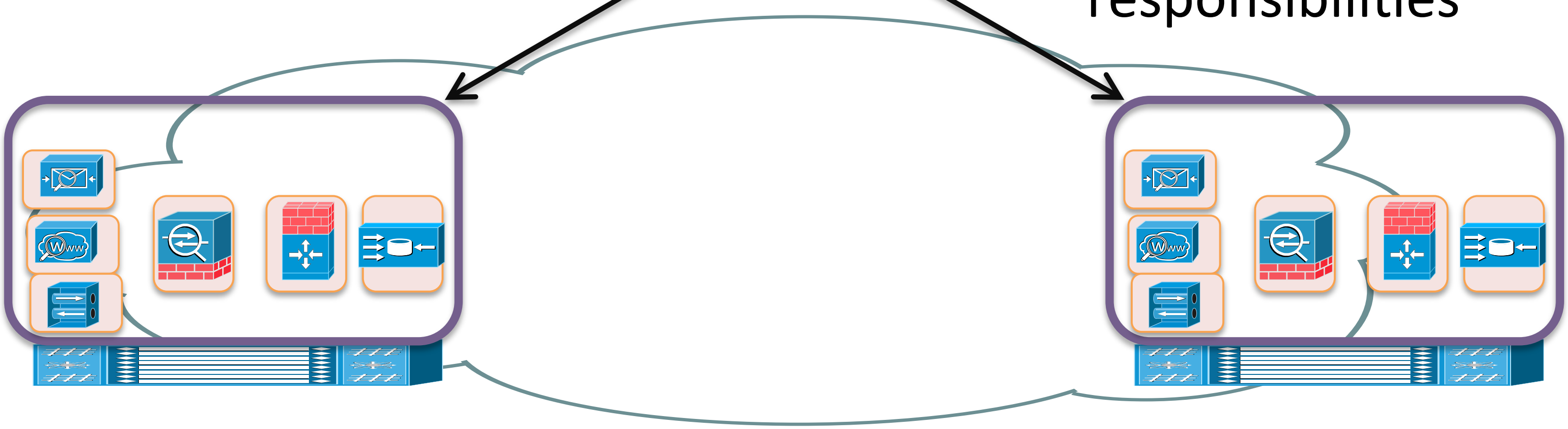
Resource Requirements

Routing, Traffic

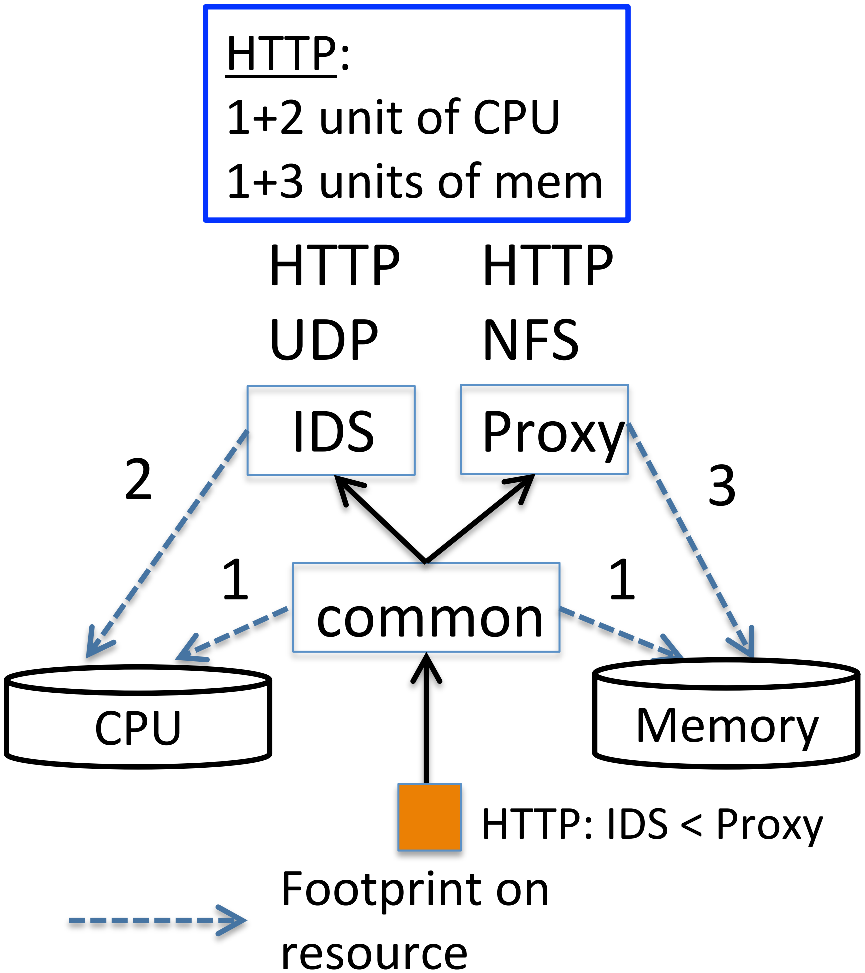
HTTP:
IDS < Proxy

Network-wide Controller

Processing responsibilities

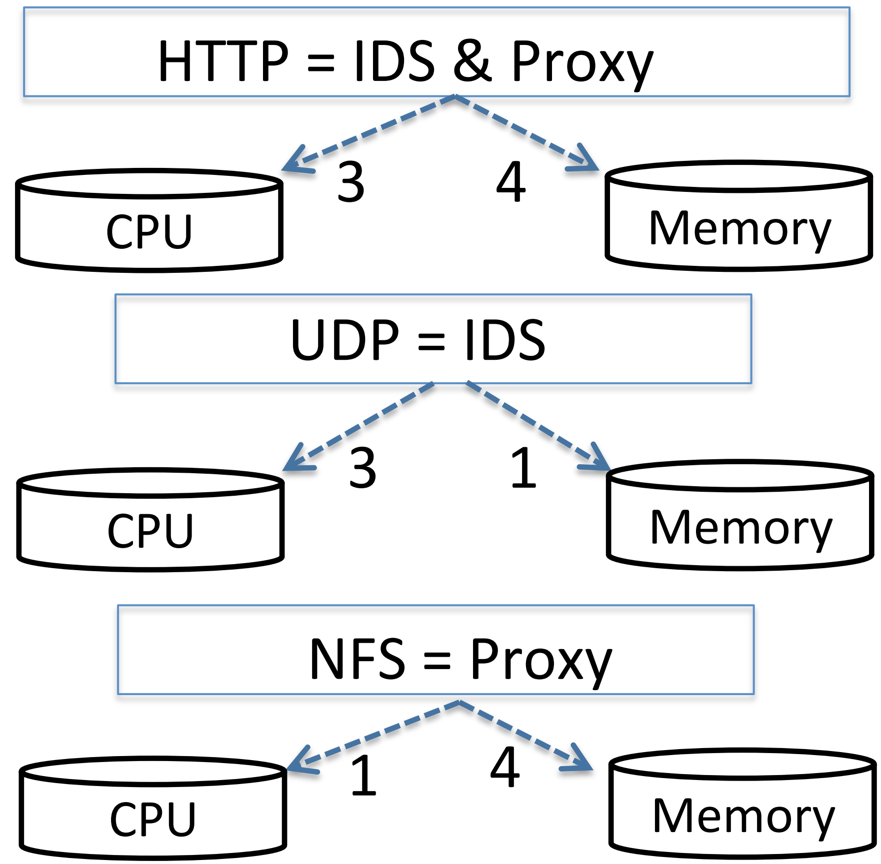


Capturing Policy and Reuse Efficiently



Need per-packet policy, reuse dependencies!

HyperApp: union of apps to run



Policy, dependency are implicit
Needs small brute-force step

Network-wide Optimization

Minimize Maximum Load, Subject to

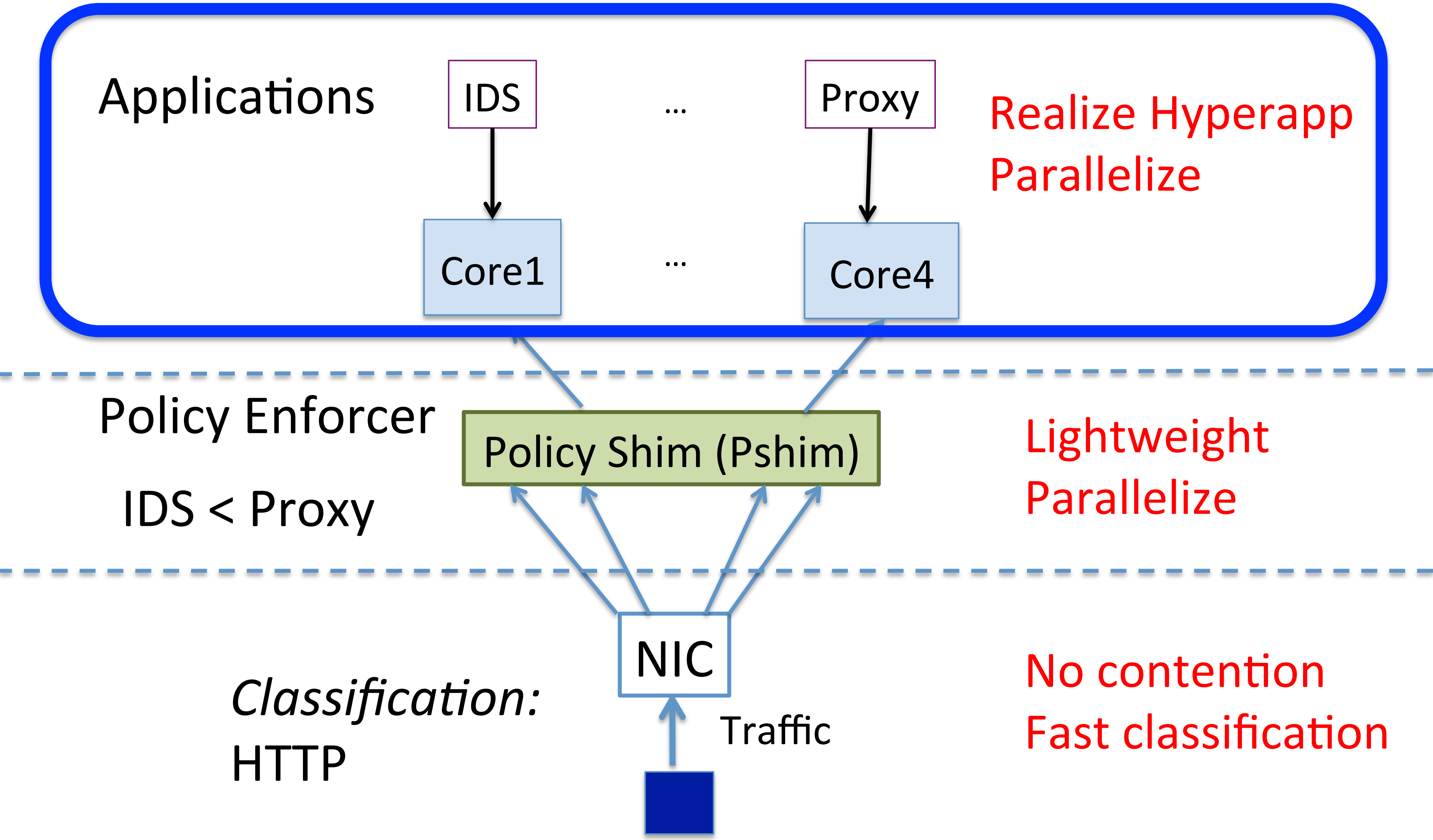
Processing coverage for each class of traffic
→ Fraction of processed traffic adds up to 1

No explicit Dependency Policy

Load on each node
→ sum over HyperApp responsibilities per-path

A simple, tractable linear program
Very close (< 0.1%) to theoretical optimal

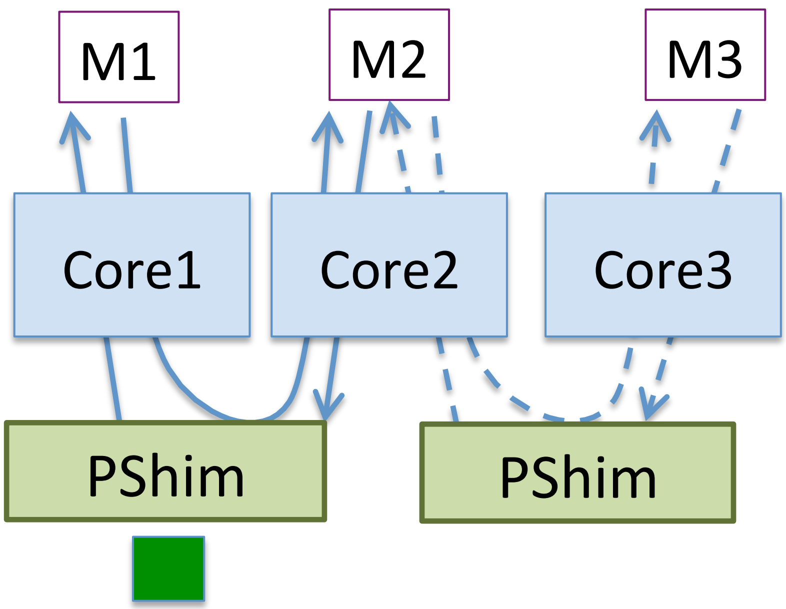
CoMb Platform



Parallelizing Application Instances

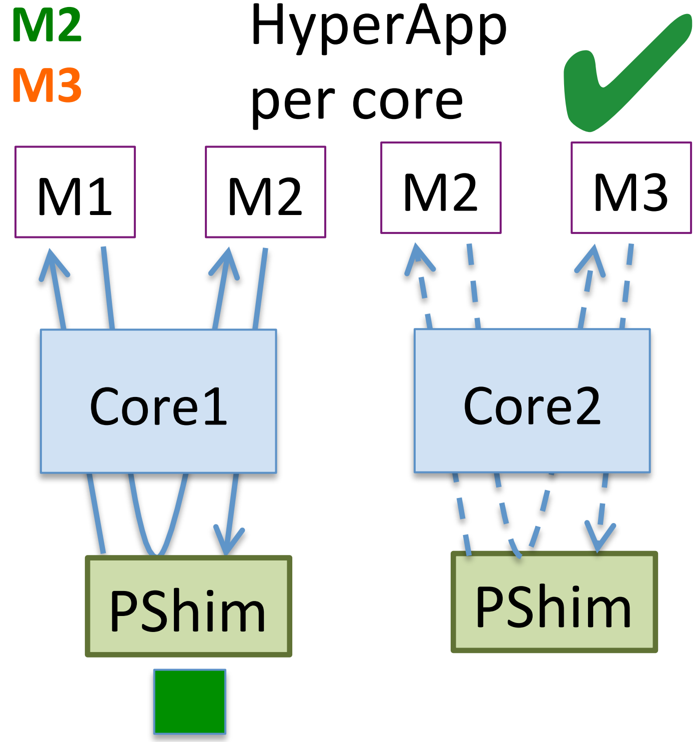
App Per core

HyperApp1: M1 < M2
HyperApp2: M2 < M3



- Inter-core communication
- More work for PShim
+ No in-core context switch

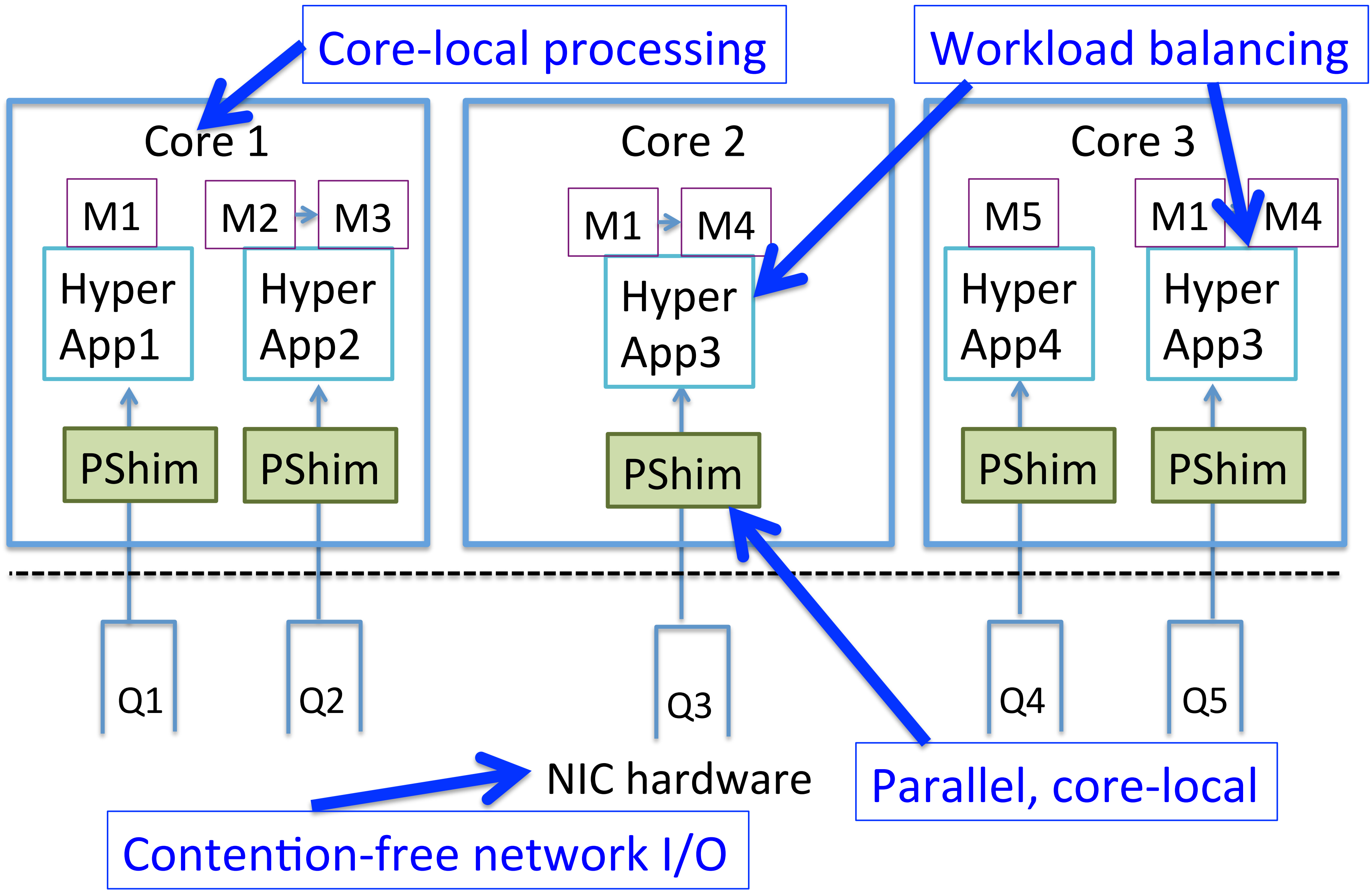
HyperApp per core



+ Keeps structures core-local
+ Better for reuse
- But incurs context-switch
- Need replicas

HyperApp-per-core is better or comparable

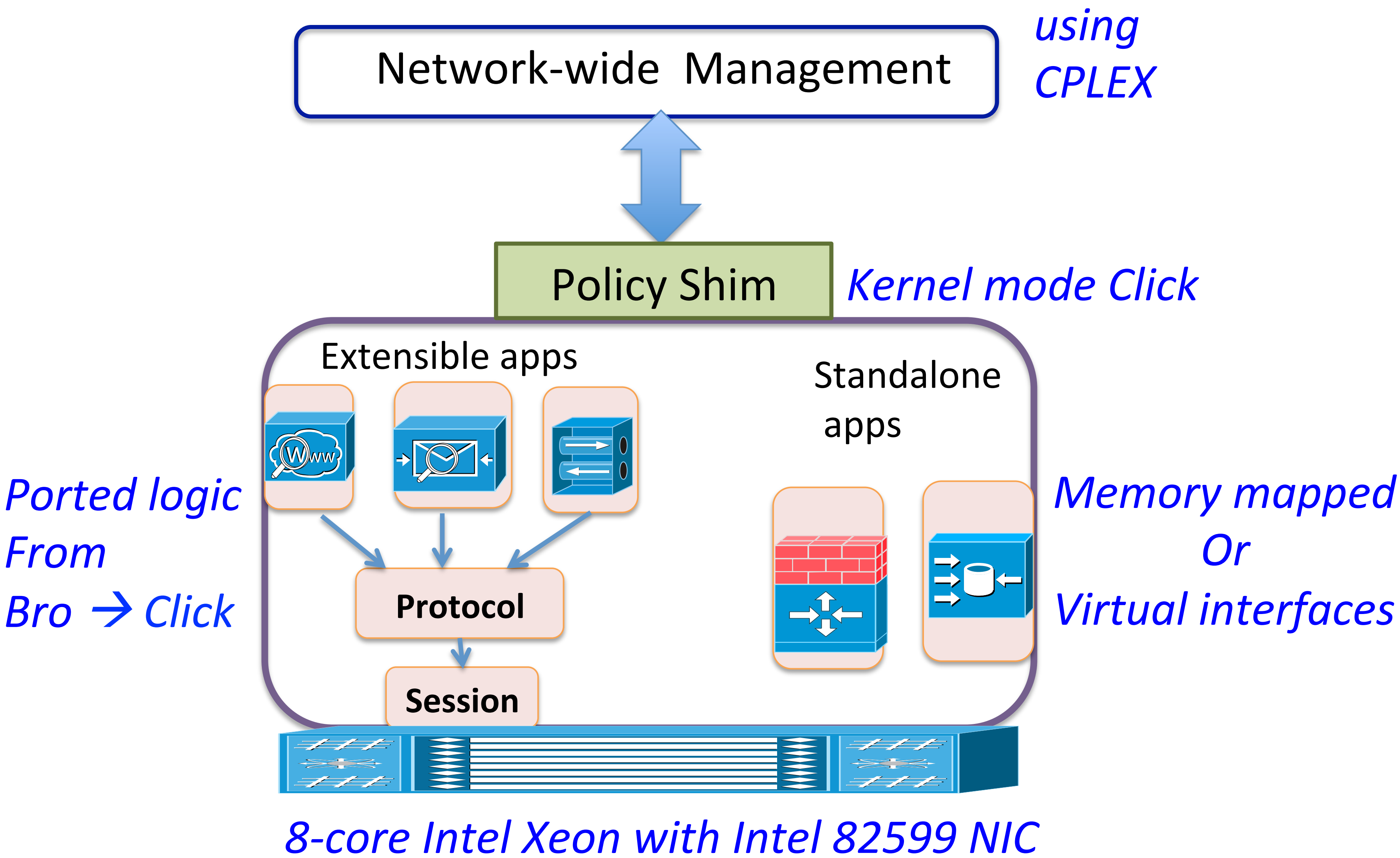
CoMb Platform Design



Outline

- Motivation
- High-level idea: Consolidation
- System design: Making Consolidation Practical
- *Implementation and Evaluation*

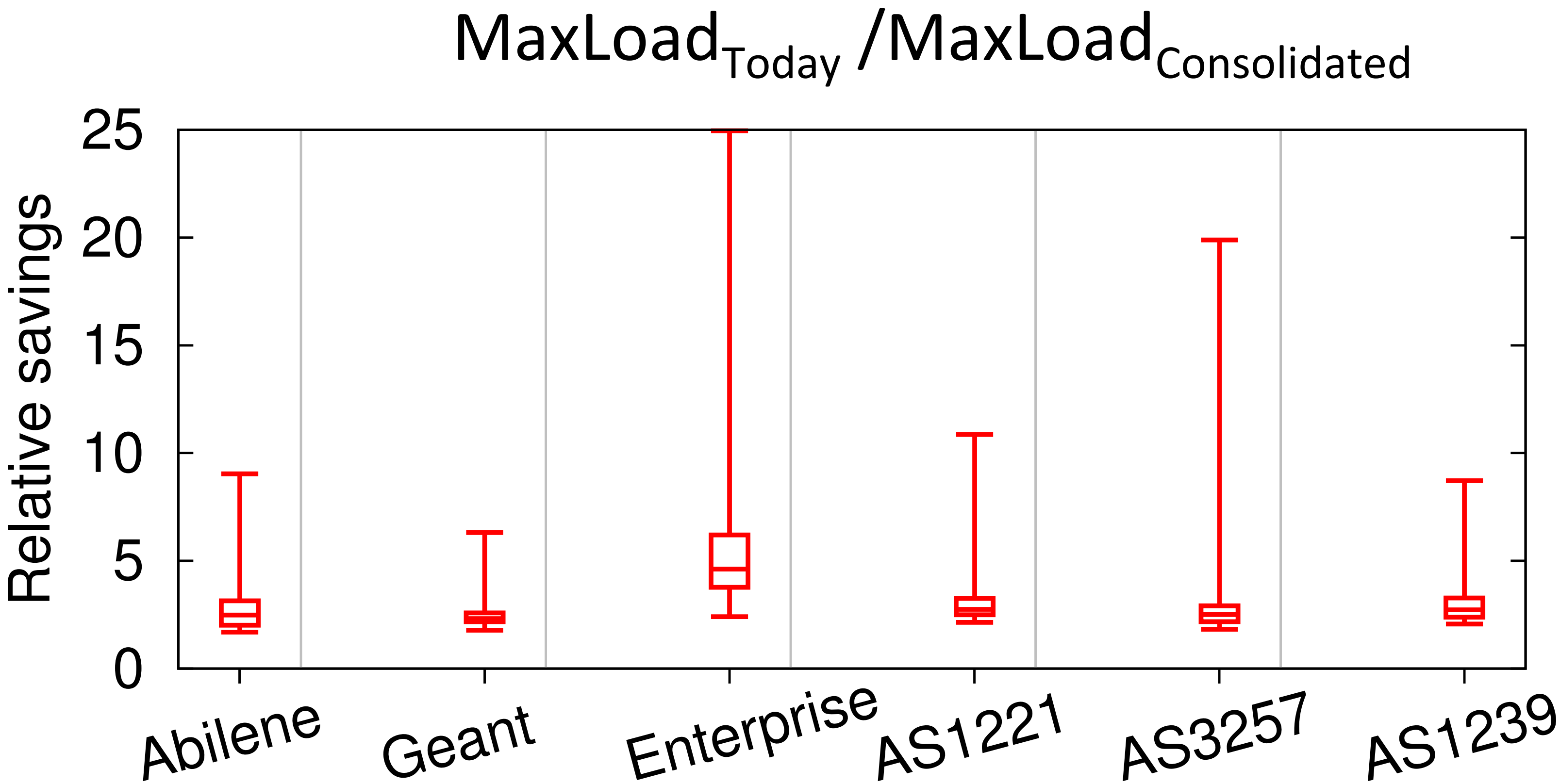
CoMb Implementation



Consolidation is Practical

- Low overhead for existing applications
- Controller takes < 1.6s for 52-node topology
- 5x better than VM-based consolidation

Benefits: Reduction in Maximum Load



Consolidation reduces maximum load by 2.5-25X

Consolidation reduces provisioning cost 1.8-2.5X

Discussion

- Isolation
  - Current: rely on process-level isolation
  - Leverage “user-space” networking
  - Get reuse-despite-isolation?
- Changes vendor business models
  - Already happening (e.g., “virtual appliances”)
  - Benefits imply someone will do it!
  - May already have extensible stacks

Conclusions

- Most network evolution today occurs via middleboxes
- Today: Narrow, point solutions
  - High CapEx, OpEx, and device sprawl
  - Inflexible, difficult to extend
- Our proposal: Consolidated architecture
  - Extensible, general-purpose
  - Reduces CapEx, OpEx, and device sprawl
- More opportunities
  - Isolation
  - APIs (H/W—Apps, Management—Apps, App Stack)