Design and Implementation of a Consolidated Middlebox Architecture

Vyas Sekar   Sylvia Ratnasamy   Michael Reiter   Norbert Egi
Guangyu Shi

[intel] [University of California] [The University of North Carolina at Chapel Hill] [Huawei]
Need for Network Evolution

New applications

Evolving threats

Performance, Security, Compliance

Policy constraints

New devices

Percentage of Methods Used to Exfiltrate Data

- Exposed Private Web Application Interface (1.5%)
- HTTP File Upload Site (1.5%)
- Malware Capability: IRC (2.5%)
- Malware Capability: SMTP (4.6%)
- SQL Injection (8.1%)
- Native FTP Client (10.0%)
- Native Remote Access Application (27.1%)
- Malware Capability: FTP (17%)
Network Evolution today: Middleboxes!

<table>
<thead>
<tr>
<th>Type of appliance</th>
<th>Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Firewalls</td>
<td>166</td>
</tr>
<tr>
<td>NIDS</td>
<td>127</td>
</tr>
<tr>
<td>Media gateways</td>
<td>110</td>
</tr>
<tr>
<td>Load balancers</td>
<td>67</td>
</tr>
<tr>
<td>Proxies</td>
<td>66</td>
</tr>
<tr>
<td>VPN gateways</td>
<td>45</td>
</tr>
<tr>
<td>WAN Optimizers</td>
<td>44</td>
</tr>
<tr>
<td>Voice gateways</td>
<td>11</td>
</tr>
<tr>
<td><strong>Total Middleboxes</strong></td>
<td><strong>636</strong></td>
</tr>
<tr>
<td><strong>Total routers</strong></td>
<td><strong>~900</strong></td>
</tr>
</tbody>
</table>

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

Just network security $10 billion
Key pain points

- Specialized boxes
- Narrow interfaces
- 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

Proxy  Firewall  IDS/IPS  AppFilter

Decouple Hardware and Software

Consolidation reduces capital expenses and sprawl

e.g., FlowStream (UCL/Lancaster)
Consolidation reduces CapEx

Multiplexing benefit = \frac{\text{Max_of_TotalUtilization}}{\text{Sum_of_MaxUtilizations}}
Consolidation Enables Extensibility

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”

Overload!

Distribution reduces load imbalance
Outline

• Motivation

• High-level idea: Consolidation

• CoMb: System design

• Implementation and Evaluation
CoMb System Overview

Network-wide Controller

Logically centralized e.g., NOX, 4D

Software-centric

e.g., PacketShader, RouteBricks, ServerSwitch, SwitchBlade

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

**HTTP**: 1+2 unit of CPU, 1+3 units of mem

**HyperApp**: union of apps to run

- **HTTP = IDS & Proxy**
  - CPU: 3
  - Memory: 4

- **UDP = IDS**
  - CPU: 3
  - Memory: 1

- **NFS = Proxy**
  - CPU: 1
  - Memory: 4

Need per-packet policy, reuse dependencies!

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

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

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

No explicit Dependency Policy
CoMb Platform

Applications → IDS → Core1 → Policy Shim (Pshim) → Policy Enforcer → NIC → Traffic → Proxy → Core4

- Realize Hyperapp
- Parallelize

Policy Enforcer:
- IDS < Proxy

Classification: HTTP
- Lightweight
- Parallelize
- No contention
- Fast classification
Parallelizing Application Instances

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

HyperApp1: M1 < M2
HyperApp2: M2 < M3

HyperApp-per-core is better or comparable
CoMb Platform Design

Core-local processing

Core 1
- M1
- Hyper App1
- PShim
- Q1

M2
- Hyper App2
- PShim
- Q2

M3
- Hyper App3
- PShim
- Q3

Core 2
- M1
- Hyper App3
- PShim

M4
- Q3

Core 3
- M5
- Hyper App4
- PShim

M1
- Hyper App3
- PShim

M4
- Q5

Workload balancing

NIC hardware

Parallel, core-local

Contention-free network I/O
Outline

• Motivation

• High-level idea: Consolidation

• System design: Making Consolidation Practical

• Implementation and Evaluation
CoMb Implementation

Network-wide Management

Policy Shim

Extensible apps

Standalone apps

Protocol

Session

using CPLEX

Kernel mode Click

Ported logic
From
Bro \rightarrow Click

Memory mapped
Or
Virtual interfaces

8-core Intel Xeon with Intel 82599 NIC
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)