CSE 534
Middleboxes

Vyas Sekar
Stony Brook University

With content from: Mike Walfish
Project 3

• Who has not met yet?

• Create “group” google doc + share with vyas
  – What questions you plan to tackle
  – Whats your proposed approach
  – Milestones
  – Project log

• Report
  – Motivation: why the questions you want to answer
  – Methodology: setup, tools, extensions
  – Analysis: results, surprises, challenges, validation
  – Discussion: whats missing, alternative approaches
Assigned Reading

• CoMb
  – Sekar et al
  – Problem?
  – Main idea?

• DOA
  – Walfish et al
  – Problem?
  – Main idea?
Like/Dislike

• Like
  – Revisiting “textbook” assumptions
  – Pragmatic not philosophical
  – Nice mix of system + architecture

• Dislike
  – Trying to do too much?
  – Deployment path?
  – Management? Users?
Outline

• Background/Context

• CoMb

• DOA
Need for Network Evolution

New applications

Evolving threats

Performance, Security, Compliance

Policy constraints

New devices
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
How many middleboxes?

Typically on par with # routers and switches.
What do different middleboxes do?

- Firewall
- IDS
- IPS
- NAT
- Proxy
- WAN optimizer
- Load balancer..
Valuable, but painful

• Both practical and architectural concerns
Practical Concerns

- Hardware costs
- Management
- Extensibility
- Deployment trajectory
- Overload
- Correctness
Architectural problems

• Violates “tenets” of Internet architecture
• End to end arguments
• Opaque to users
• Bottleneck for innovation
Outline

• Background/Context

• CoMb

• DOA
Key “pain points”

“Point” solutions!

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

Specialized boxes

Narrow interfaces

Management Management Management
Key idea: Consolidation

Two levels corresponding to two sources of inefficiency

1. Consolidate Platform

2. Consolidate Management

Network-wide Controller
Consolidation reduces CapEx

Multiplexing benefit = Max_of_TotalUtilization / Sum_of_MaxUtilizations
Consolidation Enables Extensibility

Contribution of reusable modules: 30 – 80 %
Management consolidation enables flexible resource allocation

Today: All processing at logical “ingress”

Network-wide distribution reduces load imbalance
CoMb System Overview

Network-wide Controller

Logically centralized e.g., NOX, 4D

General-purpose hardware: e.g., PacketShader, RouteBricks, ServerSwitch,

Existing work: simple, homogeneous routing-like workload

Middleboxes: complex, heterogeneous, new opportunities
Goal: Balance load across network.
Leverage multiplexing, reuse, distribution

Policy Constraints
Resource Requirements
Routing, Traffic

Network-wide Controller

Processing responsibilities
Capturing Reuse with HyperApps

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

**HTTP**  
**UDP**  
**NFS**  
**IDS**  
**Proxy**  
**common**

---

**HyperApp**: find the union of apps to run

**HTTP** = **IDS** & **Proxy**

**UDP** = **IDS**

**NFS** = **Proxy**

Policy, dependency are implicit

---

Need per-packet policy, reuse dependencies!
Modeling Processing Coverage

HTTP: Run IDS $< Proxy$

What fraction of traffic of *class HTTP* from $N1 \rightarrow N3$ should *each node* process?
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
CoMb Platform

Applications

IDS

... Core1

...Proxy

Core4

Challenges:
Performance
Parallelize
Isolation

Policy Enforcer

IDS < Proxy

Policy Shim (Pshim)

Challenges:
Lightweight
Parallelize

Classification:
HTTP

NIC

Traffic

Challenges:
No contention
Fast classification
Parallelizing Application Instances

App-per-core

- 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
Contention does not seem to matter!
CoMb Platform Design

Core-local processing

Workload balancing

Core 1

Core 2

Core 3

Parallel, core-local

NIC hardware

Contention-free network I/O

M1

Hyper App1

PShim

Q1

M2

Hyper App2

PShim

Q2

PShim

Q3

M3

Hyper App3

Hyper App4

M5

Hyper App3

M1

M4

M1

M4

Q4

Q5
Benefits: Reduction in Maximum Load

Consolidation reduces maximum load by 2.5-25X
Benefits: Reduction in Provisioning Cost

Provisioning_{Today} / Provisioning_{Consolidated}

Consolidation reduces provisioning cost 1.8-2.5X
What is missing?

- Isolation
- Business models
- What’s the right software stack?
- How do you handle “modifications”
- How do you reconfigure consistently..
Outline

• Background/Context

• CoMb

• DOA
The Problem

- **Middlebox**: interposed entity doing more than IP forwarding (NAT, firewall, cache, ...)
- Not in harmony with the Internet architecture

- No unique identifiers and on-path blocking:
  - Barrier to innovation
  - Workarounds add complexity
DOA: Delegation-Oriented Architecture

Architectural extension to Internet. Core properties:

1. Restore globally unique identifiers for hosts
2. Let receivers, senders invoke (and revoke) off-path boxes: *delegation primitive*
Globally Unique Identifiers for Hosts

- Location-independent, flat, big namespace
- Hash of a public key
- These are called EIDs (e.g., 0xf12abc...)
- Carried in packets
Delegation Primitive

Let hosts invoke, revoke off-path boxes

- **Receiver-invoked**: sender resolves receiver’s EID to
  - An IP address or
  - An EID or sequence of EIDs
- DOA header has destination stack of EIDs
- **Sender-invoked**: push EID onto this stack

| IP hdr | source EID destination EID stack | transport hdr | body |
DOA in a Nutshell

- End-host replies to source by resolving $e_s$
- Authenticity, performance: discussed in the paper
A Bit More About DOA

- Incrementally deployable. Requires:
  - Changes to hosts and middleboxes
  - No changes to IP routers (design requirement)
  - Global resolution infrastructure for flat IDs

- Recall core properties:
  - Topology-independent, globally unique identifiers
  - Let end-hosts invoke and revoke middleboxes

- Recall goals: reduce harmful effects, permit new functions
Off-path Firewall

Source EID: $e_s$
IP: $i_s$

$e_{FW}$

$e_h$ $e_{FW}$ $j$

$<e_{FW}, j>$ $<e_h, e_{FW}>$

DHT

Firewall

$e_h \rightarrow (i_h, Rules)$

Sign (MAC)

End-host

Network Stack

Verify

EID: $e_{FW}$

$e_s$ $[e_{FW}, e_h]$

$j$ $i_h$

$e_s$ $e_h$

EID: $e_h$
Reincarnated NAT

- End-to-end communication
- Port fields not overloaded
  - Especially useful when NATs are cascaded
What's missing?

- Business models
- Does this work for all middleboxes?
- Verification?
- User tussles?
Takeaways

• Middleboxes are a critical piece of infrastructure
  – Security, performance, policy compliance
  – Expensive, Difficult to manage
  – Inflexible, Opaque to users

• CoMb: consolidate to reduce capex, opex

• DOA: cleanly expose functions to users

• Re-emerging topic
  – “application” services + intelligence
  – Integrate rather than patch
  – SDN? Network function virtualization?