Middleboxes

Lecture 21, Computer Networks (198:552)
The Internet

• Textbook view: A “smooth” pipe from source to destination
The Internet

Reality: Lots of “glue”

Middleboxes!
What are middleboxes?

- **Specialized applications**
  - Security, application optimization, network management

- **Specialized *appliances***
  - Hardware boxes with custom management interfaces

- **Thousands to millions of $$ in equipment**
  - ... and more to operate, upgrade, optimize their performance
Significant deployments in enterprises!

Figure 1: Box plot of middlebox deployments for small (fewer than 1k hosts), medium (1k-10k hosts), large (10k-100k hosts), and very large (more than 100k hosts) enterprise networks. Y-axis is in log scale.

Source: Aplomb, Justine Sherry et al., SIGCOMM ‘12
Some key concerns for operators

• Upgradability: hardware vs. software

• Expertise to operate: configure, optimize, …

• Monitoring: Need visibility & diagnostics

• Managing load: what if network traffic suddenly increases?

• Could really use an elastic cloud + SDN-like approach 😊
Example: Need for load management

Source: Aplomb, Justine Sherry et al., SIGCOMM '12
Example: Need for better ways to use

- Most common causes of middlebox failure

<table>
<thead>
<tr>
<th></th>
<th>Misconfig.</th>
<th>Overload</th>
<th>Physical/Electric</th>
</tr>
</thead>
<tbody>
<tr>
<td>Firewalls</td>
<td>67.3%</td>
<td>16.3%</td>
<td>16.3%</td>
</tr>
<tr>
<td>Proxies</td>
<td>63.2%</td>
<td>15.7%</td>
<td>21.1%</td>
</tr>
<tr>
<td>IDS</td>
<td>54.5%</td>
<td>11.4%</td>
<td>34%</td>
</tr>
</tbody>
</table>
Network Function Virtualization (NFV)

• Encapsulate specialized app as software **network functions**

• Run **network functions inside VMs** in a cluster
Benefits of NFV

• Easier to upgrade
  • Regular software maintenance!

• Easier to manage
  • Use server+VM management tools

• Easier to develop
  • Write software instead of develop hardware

• Reduce costs by consolidating VMs
  • And expand elastically as needed
NetBricks: Taking the V out of NFV

Aurojit Panda et al., OSDI ’16

(ACK: Material by Aurojit Panda)
NFV Requirements

- **High Packet Rates**: Must keep up with line rate which is >10MPPS
NFV Requirements

- **High Packet Rates**: Must keep up with line rate which is >10MPPS
- **Low Latency**: Used for applications like VoIP and video conferencing
NFV Requirements

- **High Packet Rates**: Must keep up with line rate which is >10MPPS
- **Low Latency**: Used for applications like VoIP and video conferencing
- **NF Chaining**: Packets go through sequence of NFs
Challenges for NFV
Challenges for NFV

- Running NFs
- Isolation and Performance
Challenges for NFV

- Running NFs
- Isolation and Performance

- Building NFs
- High-Level Programming and Performance
Running NFs
Isolation

- **Memory Isolation**: Each NF’s memory cannot be accessed by other NFs.
Isolation

- **Memory Isolation**: Each NF’s memory cannot be accessed by other NFs.

- **Packet Isolation**: When chained, each NF processes packets in isolation.
Isolation

- **Memory Isolation**: Each NF’s memory cannot be accessed by other NFs.

- **Packet Isolation**: When chained, each NF processes packets in isolation.

- **Performance Isolation**: One NF does not affect another’s performance.
Isolation

• **Memory Isolation**: Each NF’s memory cannot be accessed by other NFs.

• **Packet Isolation**: When chained, each NF processes packets in isolation.

• **Performance Isolation**: One NF does not affect another’s performance.
Current Solution

- vSwitch
- NIC
- VM/Container
- VM/Container
- VM/Container

Memory Isolation
Packet Isolation
Performance
Current Solution

- vSwitch
- NIC
- VM/Container
- NIC
- NIC
- VM/Container
- VM/Container
- VM/Container

Memory Isolation
Packet Isolation
Performance
Current Solution

Memory Isolation
Packet Isolation
Performance
Current Solution

- vSwitch
- NIC
- VM/Container
- NIC
- VM/Container
- VM/Container

✔ Memory Isolation
Packet Isolation
Performance
Current Solution

- vSwitch
- NIC
- VM/Container
- NIC
- VM/Container
- VM/Container

- ✔ Memory Isolation
- Packet Isolation
- Performance
Current Solution

- Memory Isolation✔
- Packet Isolation
- Performance

Diagram:
- vSwitch
- NIC
- VM/Container
- NIC

Understanding the networking setup:
- vSwitch connects to NIC, providing a shared network interface for VM/Containers.
- Memory Isolation ensures secure and isolated memory access for each VM/Container.
- Packet Isolation prevents packets from traversing beyond the intended VM/Container, enhancing security.
Current Solution

- Memory Isolation
- Packet Isolation
- Performance

Diagram:
- vSwitch
- NIC
- Copy
- VM/Container
- VM/Container
- VM/Container
Current Solution

- NIC
- vSwitch
- VM/Container
- Copy
- NIC

✔ Memory Isolation
Packet Isolation
Performance
Current Solution

NIC

Memory Isolation
Packet Isolation
Performance
Current Solution

- Memory Isolation ✔
- Packet Isolation ✔
- Performance ✗
Isolation Costs Performance

No Isolation

Processing Rate (Mpps)
Isolation Costs Performance

![Graph showing processing rate (Mpps) comparison between No Isolation and OVS VM.

- No Isolation: Processing rate significantly higher than OVS VM.
- OVS VM: Processing rate much lower compared to No Isolation.]
Isolation Costs Performance

![Bar chart showing processing rate (Mpps) for different isolation methods. The chart compares the performance of No Isolation, NetBricks, OVS VM, BESS VM, and BESS Container. The chart indicates that NetBricks and No Isolation have the highest processing rates, while BESS VM and BESS Container have lower rates.](attachment:bar_chart.png)
NetBricks Runtime Architecture

Single Process Space

ZCSI Scheduler

DPDK Poll for I/O

NICs
NetBricks Runtime Architecture

ZCSI Scheduler

Function Call

Single Process Space

DPDK Poll for I/O

NICs
NetBricks Runtime Architecture

ZCSI Scheduler

Single Process Space

NF A

NF B

NF C

NF D

NF X

NF Y

NF Z

DPDK Poll for I/O

NICs
NetBricks Runtime Architecture

ZCSI Scheduler

NF A
NF B
NF C
NF D
NF X
NF Y
NF Z

Single Process Space

Run to Completion Scheduling

ZCSI Scheduler

DPDK Poll for I/O

Single Process Space

Run to Completion Scheduling

ZCSI Scheduler

DPDK Poll for I/O

NICs
What about Isolation?
ZCSI: Zero Copy Soft Isolation

- VMs and containers impose cost on packets crossing isolation boundaries.
ZCSI: Zero Copy Soft Isolation

- VMs and containers impose cost on packets crossing isolation boundaries.
- **Insight**: Use type checking (compile time) and runtime checks for isolation.
- Isolation costs largely paid at compile time (small runtime costs).
Our Approach

• Disallow pointer arithmetic in NF code: use safe subset of languages.
Our Approach

• Disallow pointer arithmetic in NF code: use safe subset of languages.

• Type checks + array bounds checking provide **memory isolation**.
Our Approach

• Disallow pointer arithmetic in NF code: use safe subset of languages.

• Type checks + array bounds checking provide memory isolation.

• Build on unique types for packet isolation.
Our Approach

• Disallow pointer arithmetic in NF code: use safe subset of languages.

• Type checks + array bounds checking provide **memory isolation**.

• Build on unique types for **packet isolation**.

  • Unique types ensure references destroyed after certain calls.
Our Approach

• Disallow pointer arithmetic in NF code: use safe subset of languages.

• Type checks + array bounds checking provide memory isolation.

• Build on unique types for packet isolation.

  • Unique types ensure references destroyed after certain calls.

  • Ensure only one NF has a reference to a packet.
Our Approach

• Disallow pointer arithmetic in NF code: use safe subset of languages.

• Type checks + array bounds checking provide memory isolation.

• Build on unique types for packet isolation.
  • Unique types ensure references destroyed after certain calls.
  • Ensure only one NF has a reference to a packet.
  • Enables zero copy packet I/O.
Our Approach

• Disallow pointer arithmetic in NF code: use safe subset of languages.

• Type checks + array bounds checking provide **memory isolation**.

• Build on unique types for **packet isolation**.
  
  • Unique types ensure references destroyed after certain calls.
  
  • Ensure only one NF has a reference to a packet.

• Enables zero copy packet I/O.

• All of these features implemented on top of **Rust**.
Software Isolation

• Provides **memory** and **packet isolation**.
Software Isolation

• Provides memory and packet isolation.

• Improved consolidation: multiple NFs can share a core.
Software Isolation

• Provides **memory** and **packet isolation**.

• Improved **consolidation**: multiple NFs can share a core.

  • Function call to NF (~ few cycles) vs context switch (~1µs).
Software Isolation

- Provides **memory** and **packet isolation**.
- Improved **consolidation**: multiple NFs can share a core.
  - Function call to NF (~ few cycles) vs context switch (~1µs).
- Reduce **memory** and **cache pressure**.
Software Isolation

• Provides memory and packet isolation.

• Improved consolidation: multiple NFs can share a core.
  • Function call to NF (~ few cycles) vs context switch (~1µs).

• Reduce memory and cache pressure.
  • Zero copy I/O => do not need to copy packets around.
Challenges for NFV

- Running NFs
- Isolation and Performance

- Building NFs
- High-Level Programming and Performance
How to write NFs?

• **Current**: NF writers concerned about meeting performance targets
How to write NFs?

- **Current**: NF writers concerned about meeting performance targets
  - Low level abstractions (I/O, cache aware data structures) and low level code.
How to write NFs?

• **Current:** NF writers concerned about meeting performance targets
  
  • Low level abstractions (I/O, cache aware data structures) and low level code.
  
  • Spend lots of time optimizing how abstractions are used to get performance.
How to write NFs?

- **Current**: NF writers concerned about meeting performance targets
  - Low level abstractions (I/O, cache aware data structures) and low level code.
  - Spend lots of time optimizing how abstractions are used to get performance.
- **Observation**: NFs exhibit common patterns: abstract and optimize these.
How to write NFs?

- **Current**: NF writers concerned about meeting performance targets
  - Low level abstractions (I/O, cache aware data structures) and low level code.
  - Spend lots of time optimizing how abstractions are used to get performance.

- **Observation**: NFs exhibit common patterns: abstract and optimize these.

- What happened in other areas
How to write NFs?

- **Current**: NF writers concerned about meeting performance targets
  - Low level abstractions (I/O, cache aware data structures) and low level code.
  - Spend lots of time optimizing how abstractions are used to get performance.
- **Observation**: NFs exhibit common patterns: abstract and optimize these.
- What happened in other areas
  - MPI to Map Reduce, etc.
Abstractions

Packet Processing
- Parse/Deparse
- Transform
- Filter

Byte Stream
- Window
- Packetize

Control Flow
- Group By
- Shuffle
- Merge

State
- Bounded
- Consistency
Abstractions

Packet Processing
- Parse/Deparse
- Transform
- Filter

Byte Stream
- Window
- Packetize

Control Flow
- Group By
- Shuffle
- Merge

State
- Bounded Consistency

Header
UDF
UDF
UDF
UDF
UDF
Shuffle Abstraction

Spread packets across cores for scaling
Shuffle Abstraction

Spread packets across cores for scaling

Might even use hardware for this.
Example NF: Maglev

- **Maglev**: Load balancer from Google (NSDI’16).
- Main contribution: a novel consistent hashing algorithm.
  - Most of the work in common optimization: batching, scaling cross core.
- NetBricks implementation: **105 lines, 2 hours of time**.
- Comparable performance to optimized code
Managing NFs

E2 (SOSP'15)
Stratos
FTMB (SIGCOMM '15)
FlowTags (NSDI '14)

Building and Running NFs

No Isolation
CoMB (NSDI'12)
xOMB (ANCS'12)

VM Isolation
ClickOS (NSDI'14)
NetVM (IEEE TNSM)
HyperSwitch (ATC'13)
mSwitch (SOSR'15)
Conclusion

• Software isolation is necessary for high performance NFV.
• Type checking + bound checking + unique types.
• Performance is not anathema to high-level programming
• Abstract operators + UDF simplify development.

Code available at http://netbricks.io/
Backup
Both Memory Isolation and I/O Induce Overheads

![Graph showing processing rate (Mpps) for different isolation methods. The graph compares No Isolation, 0-Copy Container, and BESS Container. The No Isolation method has the highest processing rate, followed by the 0-Copy Container and then the BESS Container.]