Making Network Functions Software-Defined

Yotam Harchol
VMware Research / The Hebrew University of Jerusalem

Joint work with Anat Bremler-Barr and David Hay

Appeared in ACM SIGCOMM 2016

This research was supported by the European Research Council ERC Grant agreement no 259085, the Israeli Centers of Research Excellence (I-CORE) program (Center No. 4/11), and the Neptune Consortium.
Network Functions (Middleboxes)

• Monolithic **closed** black-boxes
  - High *cost*
  - Limited *provisioning* and *scalability*

Network Function Virtualization (NFV):
- ✔ Reduce *cost* (by moving to software)
- ✔ Improve *provisioning* and *scalability* (by virtualizing software NFs)
Network Functions (Middleboxes)

- High cost
- Limited *provisioning* and *scalability*
- Limited and separate *management*
  - Different vendors
  - No standards
  - Separate control plane
Network Functions (Middleboxes)

- Actually, many of these black-boxes are very modular

- High cost
- Limited provisioning and scalability
- Limited and separate management
- Limited *functionality* and limited *innovation* (High entry barriers)
- Similar complex processing steps, *no re-use*
OpenBox: A new software-defined framework for network functions
- Decouples network function control from their data plane
- Unifies data plane of multiple network functions

Benefits:
- Easier, unified control
- Better performance
- Scalability
- Flexible deployment
- Inter-tenant isolation
- Innovation

www.openboxproject.org
github.com/OpenBoxProject
Software Defined Networking

- High **cost** of *middleboxes switches*
- Limited **provisioning** and **scalability** of *middleboxes switches*
- Limited **management** of *middleboxes switches*
- Limited **functionality** and limited **innovation**
- Complex processing steps - *distributed algorithms*

40%-60% of the appliances in large-scale networks are middleboxes!
[Sherry & Ratnasamy, ‘12]
The OpenBox Framework

Network Functions: OpenBox Applications

Logically-Centralized OpenBox Controller

Northbound API

OpenBox Protocol

OpenBox Service Instances

Control Plane
Data Plane

Additionally:

✔ Isolation between NFs / multiple tenants
✔ Support for hardware accelerators
✔ Dynamically extend the protocol
Observation:

Most network functions do very similar processing steps

But there is no re-use...

The design the OpenBox framework is based on this observation
Network Function Decomposition

Firewall:
- Read Packets
- Header Classifier
- Drop
- Output
- Alert

Load Balancer:
- Read Packets
- Header Classifier
- Output
- Rewrite Header

Intrusion Prevention System:
- Read Packets
- Header Classifier
- DPI
- DPI
- DPI
- Drop
- Alert
- Output
Northbound API

Specify processing graph and block configuration

Events, Load information

OpenBox Protocol

Control Plane
Data Plane

OpenBox Service Instances

OpenBox Controller

OpenBox Applications

Intrusion Prevention System

Load Balancer

Firewall

Read Packets  Header Classifier  Drop

Read Packets  Header Classifier  Alert

Read Packets  Header Classifier  Rewrite Header

Read Packets  Header Classifier  Output

Read Packets  Header Classifier  Drop

Read Packets  Header Classifier  DPI

Read Packets  Header Classifier  DPI

Read Packets  Header Classifier  DPI

Read Packets  Header Classifier  Output
Multiple tenants run multiple applications for multiple policies in the same network.

Isolation between applications and tenants enforced by NB API.

Network-wide view, Automatic scaling, provisioning, placement, and steering.
Naïve Graph Merge

Firewall:

- Read Packs
- Header Classifier
- Drop
- Alert
- Output

Intrusion Prevention System:

- Read Packs
- Header Classifier
- Drop (Firewall)
- Alert
- DPI
- Output

Concatenated Processing Graph:

- Read Packs
- Header Classifier
- DPI
- Alert
- Output

Performance $\approx$ Diameter of Graph (# of classifiers)

Total: 134μs
Graph Merge Algorithm

Merged Processing Graph:

Algorithm and details are in the paper

Read Packets  
2μs

Header Classifier  
30μs

Alert (Firewall)
10μs

DPI  
50μs

Alert (Firewall)
50μs

Alert (Firewall)
10μs

Alert (Firewall)
30μs

DPI  
10μs

Alert (IPS)

Drop

Output  
2μs

Total: 104μs (22% improvement)
OpenBox Data Plane Processing

- Provides data plane services to realize the logic of network functions
- Controlled by the logically-centralized OpenBox controller
Distributed Data Plane

- OpenBox Service Instance
  - Hardware (TCAM)
  - E.g., an OpenFlow switch with encapsulation features (e.g., NSH, Geneve, FlowTags)

- OpenBox Service Instance
  - Software
  - Header Classifier
  - Metadata
  - Alert
  - DPI
  - Rewrite Header
Split Processing Graph

HW Instance:

Read Packets → Header Classifier → Write Metadata → Encapsulate Metadata → Output → Drop

SW Instance:

Read Packets → Decapsulate Metadata → Read Metadata → DPI → DPI → DPI → Alert → Output → Drop
Distributed Data Plane
Extensible Data Plane

Option 1: New hardware implementation
Supports encapsulation

Option 2: Software module injection
Custom software module (signed)
On the fly
No need to recompile
No need to redeploy
Scalable & Reliable Data Plane

Scalability

Provisioning

Reliability

OpenBox Controller

Hypervisor

OBI
OpenBox Protocol: Connection Setup

Controller

Service Instance

Hello

SetParametersRequest

SetParametersResponse

AddCustomModuleRequest

AddCustomModuleResponse

BarrierRequest

SetProcessingGraphRequest

SetProcessingGraphResponse

BarrierRequest
5.9. RegexClassifier
Classify a packet using a regex match on its content. The Classifier has N outputs, each associated with the corresponding pattern. If a packet doesn’t match any rule it will be discarded.

5.9.1. Configuration

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pattern</td>
<td>true</td>
<td>array(string)</td>
<td>A list of pattern to match against. Each pattern must be legal regex (with no backtracking).</td>
</tr>
<tr>
<td>payload_only</td>
<td>false</td>
<td>bool</td>
<td>If true, the match will be only on the payload part of the packet. The payload is deteremined by the set of network layers. Default is false.</td>
</tr>
</tbody>
</table>

5.9.2. Read Handles

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>count_i</td>
<td>number</td>
<td>Returns the number of packets matched pattern_i</td>
</tr>
<tr>
<td>byte_count_i</td>
<td>number</td>
<td>Returns the number of bytes matched pattern_i</td>
</tr>
<tr>
<td>rate_i</td>
<td>number</td>
<td>Returns the matching rate for pattern_i, measured by exponential weighted moving average, in packets per second.</td>
</tr>
<tr>
<td>byte_rate_i</td>
<td>number</td>
<td>Returns the matching rate for pattern_i, measured by exponential weighted moving average, in bytes per second.</td>
</tr>
<tr>
<td>payload_only</td>
<td>bool</td>
<td>Read the ‘payload_only’ value.</td>
</tr>
</tbody>
</table>

5.9.3. Write Handles

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>payload_only</td>
<td>bool</td>
<td>Set the ‘payload_only’ value.</td>
</tr>
</tbody>
</table>
OpenBox Protocol: Block Hierarchy

Abstract Processing Block

- HeaderClassifier
- TCAMClassifier
- TrieClassifier

Hello
...
Supported implementations:
HeaderClassifier:
[TCAMClassifier, TrieClassifier]

SetProcessingGraphRequest
...
Use TCAMClassifier in graph
Implementation

github.com/OpenBoxProject

Java-based OpenBox Controller

<table>
<thead>
<tr>
<th>FW</th>
<th>IPS</th>
<th>Load Balancer</th>
<th>...</th>
</tr>
</thead>
</table>

Northbound API

- REST client/server
- Graph Aggregator
- Network Manager
- Management API

7500 LoCs (Java)

Control Plane

Data Plane

Software OpenBox Service Instance

Generic wrapper for execution engines (Python)

Translation Engine

<table>
<thead>
<tr>
<th>Click-based execution engine (C++)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2400 LoCs for plugin (C++)</td>
</tr>
</tbody>
</table>

(Plug here other execution engines. E.g., ClickNP)

5500 LoCs (Python)
Implementation

github.com/OpenBoxProject

Java-based OpenBox Controller

<table>
<thead>
<tr>
<th>FW</th>
<th>IPS</th>
<th>Load Balancer</th>
<th>...</th>
</tr>
</thead>
</table>

Northbound API

- REST client/server
- Graph Aggregator
- Network Manager
- Management API

Control Plane

Data Plane

Generic wrapper for execution engines (Python)

Translation Engine

Work in progress:
Hybrid HW-SW execution engine using Click and an OpenFlow switch
OpenBox with OpenFlow Hardware

Hardware switch provides:
- Header classification
- Alerting / logging (through OFC)
- Output / drop blocks

Hybrid OBI splits processing graph:
- Prefix with only the above blocks → To switch
- Suffix → To SW OBI (using Click)

(Work in progress)
Performance Improvement

**Without OpenBox**

- VM1: Firewall
- VM2: IPS

**With OpenBox**

- VM1: OBI1: FW+IPS
- VM2: OBI2: FW+IPS

**Standalone VM**

<table>
<thead>
<tr>
<th>Firewall</th>
<th>IPS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Throughput [Mbps]</td>
<td>Throughput [Mbps]</td>
</tr>
<tr>
<td>900</td>
<td>900</td>
</tr>
<tr>
<td>800</td>
<td>800</td>
</tr>
<tr>
<td>700</td>
<td>700</td>
</tr>
<tr>
<td>600</td>
<td>600</td>
</tr>
<tr>
<td>500</td>
<td>500</td>
</tr>
<tr>
<td>400</td>
<td>400</td>
</tr>
<tr>
<td>300</td>
<td>300</td>
</tr>
<tr>
<td>200</td>
<td>200</td>
</tr>
<tr>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

**NF Pipeline**

- Throughput [Mbps]
- Latency [µs]

Without OpenBox: -35%

With OpenBox: +86%
Dynamic Load Balancing

**Without OpenBox**

VM1
Firewall1
VM2
Firewall2

**With OpenBox**

VM1
OBI1
VM2
OBI2

![Graph showing Dynamic Load Balancing Throughput Region and Static Load Balancing Throughput Region](image)
Dynamic Load Balancing

**Without OpenBox**

VM1
Firewall

HTTP

VM2
IPS

**With OpenBox**

VM1
OBI1

VM2
OBI2

---

Graph showing the max throughput of Firewall and IPS with dynamic and static load balancing.
Graph Merge Algorithm ➔ Improved Performance

Overall Throughput

<table>
<thead>
<tr>
<th></th>
<th>Throughput [Mbps]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Naïve Merge</td>
<td>700</td>
</tr>
<tr>
<td>Our Algorithm</td>
<td>900 (+20%)</td>
</tr>
</tbody>
</table>

WAN ➔ Gateway Firewall ➔ Web Cache ➔ Dept. Firewall ➔ Load Balancer ➔ Servers
Control Channel

Controller

Service Instance

RTT: 20ms

KeepAlive

HTTP 200 OK

GlobalStatisticsRequest

HTTP 200 OK

GlobalStatisticsResponse

SetProcessingGraphRequest

HTTP 200 OK

SetProcessingGraphResponse

RTT: 25ms

RTT: 1285ms

Hard-coded 1000 ms delay in Click code (can be reduced)
Related Work

• Orthogonal to OpenBox:
  – NF traffic steering (e.g., SIMPLE [SIGCOMM ’14])
  – NF orchestration (e.g., Stratos, OpenMano, OpenStack)
  – Runtime platforms (e.g., xOMB [ANCS ‘12], ClickNP [SIGCOMM ‘16])

• Similar Motivation:
  – CoMb [NSDI ‘12] – focuses on resource sharing and placement
  – Slick [SOSR ’15] – focuses on the placement of data plane units

• Only OpenBox provides:
  – Core processing decomposition and reuse
  – Standardization and full decoupling of NF control and data planes
Conclusions

• Network functions are currently a real challenge in large scale networks
• OpenBox decouples the data plane processing from network function control logic and:
  – Reduces costs
  – Enhances performance
  – Improves scalability
  – Increases reliability
  – Provides inter-tenant isolation
  – Allows easier innovation
Questions?

THANK YOU!

Play with OpenBox on a Mininet VM: 
github.com/OpenBoxProject/openbox-mininet