OpenBox: A Software-Defined Framework for Developing, Deploying, and Managing Network Functions

Yotam Harchol
The Hebrew University of Jerusalem

Joint work with Anat Bremler-Barr and David Hay

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](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

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
- Alert
- Output

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

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

Specify processing graph and block configuration

Events, Load information

Control Plane
Data Plane

OpenBox Protocol

OpenBox Service Instances

OpenBox Controller

OpenBox Applications

Intrusion Prevention System

Load Balancer

Firewall

Read Packets
Header Classifier
Alert
Drop
Output

Read Packets
Header Classifier
Rewrite Header
Output

Read Packets
Header Classifier
Drop
Alert
Output

Read Packets
Header Classifier
Drop
Output

Read Packets
Header Classifier
Drop
Output

Read Packets
Header Classifier
Drop
Alert
Output

OpenBox	Service	Instances

Control Plane
Data Plane

OpenBox Protocol

Specify processing graph and block configuration
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.
Naive Graph Merge

Firewall:

Read Packets -> Header Classifier -> Drop -> Alert -> Output

Intrusion Prevention System:

Read Packets -> Header Classifier -> Drop (Firewall) -> Alert (Firewall)

Header Classifier -> DPI -> Alert (IPS) -> Drop (IPS) -> Output

Concatenated Processing Graph:

Read Packets -> Header Classifier -> DPI

Header Classifier -> DPI

Performance ≈ Diameter of Graph (# of classifiers)

Total: 134μs
Graph Merge Algorithm

Merged Processing Graph:

- Read Packets: 2μs
- Header Classifier: 30μs
- Alert (Firewall): 50μs
- Alert (Firewall): 10μs
- Alert (Firewall): 2μs
- DPI: 10μs
- DPI: 50μs
- DPI: 10μs
- Alert (IPS): 10μs
- Output: 2μs

Algorithm and details are in the paper

Shorter Diameter (less classifiers)

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

E.g., an OpenFlow switch with encapsulation features (e.g., NSH, Geneve, FlowTags)
Split Processing Graph

HW Instance:

SW Instance:
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
Implementation

Java-based OpenBox Controller

<table>
<thead>
<tr>
<th>Northbound API</th>
</tr>
</thead>
<tbody>
<tr>
<td>FW</td>
</tr>
</tbody>
</table>

Control Plane

REST client/server

Graph Aggregator

Network Manager

Management API

Data Plane

Software OpenBox Service Instance

Generic wrapper for execution engines (Python)

Translation Engine

Click-based execution engine (C++)

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

github.com/OpenBoxProject
Performance Improvement

Without OpenBox

VM1 Firewall
VM2 IPS

With OpenBox

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

Standalone VM

Throughput [Mbps]
Latency [µs]

NF Pipeline

Throughput [Mbps]
Latency [µs]

-35%
+86%
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!