CompSci 514: Computer Networks
Lecture 16: Network Function Virtualization

Xiaowei Yang
Adapted from Prof. Srinivas Seshan’s lecture slides
Overview

• NFV Motivation

• NFV case study

• More NFV challenges and solutions

• Optional homework
Network as we knew it vs. Reality

Traditional view: “Dumb” network

Reality: Lots of in-network processing

Appliances or Middleboxes:
IDS, Firewall, Proxies, Load balancers….
Need for Network Evolution

- New applications
- Evolving threats
- New devices
- Policy constraints

Performance, Security, Compliance
## Middleboxes Galore!

### Data from a large enterprise

<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>
</tbody>
</table>

### Total routers ~900

### Survey across 57 network operators

- **>100k hosts**
- **10k-100k hosts**
- **1k-10k hosts**
- **<1k hosts**

APLOMB (SIGCOMM’13)
Specialized boxes

“Point” solutions!

Key “pain points”

- Management
- Management
- Management

Narrow interfaces

- Increases capital expenses & sprawl
- Increases operating expenses
- Limits extensibility and flexibility
Middlebox management is hard!

Critical for security, performance, compliance
But expensive, complex and difficult to manage

<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>
Vision

• Why can't networking get the same benefits as IT and cloud world?

• Commodity hardware?

• Virtualization?

• Consolidation
Network Functions Virtualisation

Introduction

Network operators' networks are populated with a large and increasing variety of proprietary hardware appliances. To launch a new network service often requires yet another variety and finding the space and power to accommodate these boxes is becoming increasingly difficult; compounded by the increasing costs of energy, capital investment challenges and the rarity of skills necessary to design, integrate and operate increasingly complex hardware-based appliances. Moreover, hardware-based appliances rapidly reach end of life, requiring much of the procure-design-integrate-deploy cycle to be repeated with little or no revenue benefit. Worse, hardware lifecycles are becoming shorter as technology and services innovation accelerates, inhibiting the rollout of new revenue earning network services and constraining innovation in an increasingly network-centric connected world.

Definition

Network Functions Virtualisation aims to transform the way that network operators architect networks by evolving standard IT virtualisation technology to consolidate many network equipment types onto industry standard high volume servers, switches and storage, which could be located in Datacentres, Network Nodes and in the end user premises, as illustrated in Figure 1. It involves the implementation of network functions in software that can run on a range of industry standard server hardware, and that can be moved to, or instantiated in, various locations in the network as required, without the need for installation of new equipment.

Figure 1: Vision for Network Functions Virtualisation

Relationship with Software Defined Networks (SDN)

As shown in Figure 2, Network Functions Virtualisation is highly complementary to Software Defined Networking (SDN), but not dependent on it (or vice-versa). Network Functions Virtualisation can be

Classical Network Appliance Approach

- Fragmented non-commodity hardware.
- Physical install per appliance per site.
- Hardware development large barrier to entry for new vendors, constraining innovation & competition.

Independent Software Vendors

Orchestrated, automatic & remote install.

Standard High Volume Servers

Standard High Volume Storage

Standard High Volume Ethernet Switches

Network Virtualisation Approach
Key idea: Consolidation

Two levels corresponding to two sources of inefficiency

1. Consolidate Platform

2. Consolidate Management

Network-wide Controller
What are the grand challenges?

• High performance virtual appliances?
• Isolation/coexistence
• Management solutions
• Fault tolerance
• Vendor independence/multi-vendor
What’s missing?

• What functions yield most benefits?

• Can it really replace hardware acceleration?

• Is virtualization necessary?

• What novel services can be developed?

• How much benefit is “enough” to motivate adoption?
Consolidation at Platform-Level

Today: Independent, specialized boxes

Proxy          Firewall          IDS/IPS          AppFilter

Decouple Hardware and Software

Commodity hardware: e.g., PacketShader, RouteBricks, ServerSwitch, SwitchBlade

Consolidation reduces capital expenses and sprawl
Consolidation reduces CapEx

Normalized utilization (%)

Time (mm-dd,hr)

WAN optimizer
Proxy
Load Balancer
Firewall

Multiplexing benefit = \frac{\text{Max of TotalUtilization}}{\text{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
Can NFV/SDN help middlebox management?

Centralized Controller

OpenFlow

Web

Firewall → IDS → Proxy

Proxy

IDS

Necessity and an Opportunity:
Incorporate functions markets views as important
SDN vs NFV

• Complementary

• SDN is all about “control” plane

• NFV can happen w/o SDN

• Natural allies
  – SDN enables orchestration, routing
  – NFV can be the “substrate” over which SDN runs
CoMb System Overview

Network-wide Controller

Logically centralized e.g., NOX, 4D

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

Middleboxes: complex, heterogeneous, new opportunities
CoMb Management Layer

Goal: Balance load across network. Leverage multiplexing, reuse, distribution

Policy Constraints
Resource Requirements
Routing, Traffic

Network-wide Controller
Processing responsibilities
Capturing Reuse with HyperApps

HyperApp: find the union of apps to run

HTTP: IDS & Proxy

HTTP

UDP

NFS

IDS

Proxy

common

CPU

Memory

Footprint on resource

Need per-packet policy dependencies!

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

UDP: 1 unit of CPU
1+3 units of mem

NFS: 1+3 units of mem

Policy dependency are implicit
Modeling Processing Coverage

HTTP: Run IDS → Proxy

What fraction of traffic of class HTTP from N1 to 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
Network-wide Optimization

\[
\text{Minimize } \max_{r,n} \{ \text{load}_{n,r} \}, \quad \text{subject to } \quad (1)
\]

\[
\forall n, r: \text{load}_{n,r} = \sum_{c: n \in \text{path}_c} \frac{d_{c,n}|T_c|F_{h_c,r,n}}{\text{Prov}_{n,r}} \quad (2)
\]

\[
\forall c: \sum_{n \in \text{path}_c} d_{c,n} = 1 \quad (3)
\]

\[
\forall c, n: 0 \leq d_{c,n} \leq 1 \quad (4)
\]

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

Applications

IDS

Core1

Proxy

Core4

Policy Enforcer

IDS → Proxy

Classification:

HTTP

Policy Shim (Pshim)

NIC

Traffic

Challenges:

Performance
Parallelize
Isolation

Challenges:

Lightweight
Parallelize

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

HyperApp-per-core is better or comparable
Contention does not seem to matter!
Discussion

• Changes traditional vendor business
  – Already happening (e.g., “virtual appliances”)
  – Benefits imply someone will do it!
  – May already have extensible stacks internally!

• Isolation
  – Current: rely on process-level isolation
  – Get reuse-despite-isolation?
Conclusions

• Network evolution occurs via middleboxes

• Today: Narrow “point” solutions
  – High CapEx, OpEx, and device sprawl
  – Inflexible, difficult to extend

• Our proposal: Consolidated architecture
  – Reduces CapEx, OpEx, and device sprawl
  – Extensible, general-purpose

• More opportunities
  – Isolation
  – APIs (H/W—Apps, Management—Apps, App Stack)
More NFV challenges & solutions

• Performance

• Session management
  – How to chain the NFV together?

• More applications
NFP: Enabling Network Function Parallelism in NFV

Chen Sun
Tsinghua University
c-sun14@mails.tsinghua.edu.cn

Jun Bi
Tsinghua University
junbi@tsinghua.edu.cn

Zhilong Zheng
Tsinghua University
zhengzl15@mails.tsinghua.edu.cn

Heng Yu
Tsinghua University
hengyu1215@163.com

Hongxin Hu
Clemson University
hongxin@clemson.edu

ABSTRACT

Software-based sequential service chains in Network Function Virtualization (NFV) could introduce significant performance overhead. Current acceleration efforts for NFV mainly target on optimizing each component of the sequential service chain. However, based on the statistics from real-world enterprise networks, we observe that 53.8% network function (NF) pairs can work in parallel. In particular, 41.5% NF pairs can be parallelized without causing extra resource overhead. In this paper, we present NFP, a high-performance framework that innovatively enables network function parallelism to improve NFV performance. NFP consists of three logical components. First, NFP provides a policy specification scheme for operators to intuitively describe sequential or parallel NF chaining intents. Second, NFP orchestrator intelligently identifies NF dependency and automatically compiles the policies into high-performance service graphs. Third, NFP infrastructure performs lightweight packet copying, distributed parallel packet delivery, and load-balanced merging of packet copies to support NF parallelism. We implement an NFP prototype based on DPDK in Linux containers. Our evaluation results show that NFP achieves significant latency reduction for real-world service chains.

CCS CONCEPTS

- Networks → Middle boxes / network appliances; Network performance analysis; Network control algorithms;

KEYWORDS

NFV, network function parallelism, service chain

ACM Reference Format:
https://doi.org/10.1145/3090822.3090826

*Chen Sun, Jun Bi, Zhilong Zheng, and Heng Yu are with Institute for Network Sciences and CyberSpace, Tsinghua University. Department of Computer Science, Tsinghua University, and Tsinghua National Laboratory for Information Science and Technology (TNList). Jun Bi is the corresponding author.

Permission to make digital or hard copies of all or part of this work for personal or

• Use parallel processing to shorten NFV latency

1 INTRODUCTION

Network Functions Virtualization (NFV) addresses the problems of traditional proprietary middleboxes [61] by leveraging virtualization technologies to implement network functions (NFs) on commodity hardware, in order to enable rapid creation, destruction, or migration of NFs [24]. In operator networks [52], data centers [32, 36], mobile networks [25] and enterprise networks [60].
Use a session protocol to set up NFV chain and dynamically manage the chaining
A user space NF scheduler that uses backpressure to shed load early in service chain.
A High Performance Packet Core for Next Generation Cellular Networks

Zafar Ayyub Qazi  
UC Berkeley

Vyasa Sekar  
Carnegie Mellon University

Melvin Walls  
Nefeli Networks, Inc.

Sylvia Ratnasamy  
UC Berkeley

Aurojit Panda  
UC Berkeley

Scott Shenker  
UC Berkeley

ABSTRACT

Cellular traffic continues to grow rapidly making the scalability of the cellular infrastructure a critical issue. However, there is mounting evidence that the current Evolved Packet Core (EPC) is ill-suited to meet these scaling demands: EPC solutions based on specialized appliances are expensive to scale and recent software EPCs perform poorly, particularly with increasing numbers of devices or signaling traffic.

In this paper, we design and evaluate a new system architecture for a software EPC that achieves high and scalable performance. We postulate that the poor scaling of existing EPC systems stems from the manner in which the system is decomposed which leads to device state being duplicated across multiple components which in turn results in frequent interactions between the different components. We propose an alternate approach in which state for a single device is consolidated in one location and EPC functions are (re)organized for efficient access to this consolidated state. In effect, our design "slices" the EPC by user.

We prototype and evaluate PEPC, a software EPC that implements the key components of our design. We show that PEPC achieves 3-7x higher throughput than comparable software EPCs that have been implemented in industry and over 10x higher throughput than a popular open-source implementation (OpenAirInterface). Compared to the industrial EPC implementations, PEPC sustains high data throughput for 10-100x more users per core, and a 10x higher ratio of signaling-to-data traffic. In addition to high performance, PEPC’s by-user organization enables efficient state migration and customization of processing pipelines. We implement user migration in PEPC and show that state can be migrated with little disruption, e.g., migration adds only up to 4µs of latency to median per packet latencies.

CSCS CONCEPTS

- Networks  → Network components; Middle boxes / network appliances;

KEYWORDS

Cellular Networks, EPC, Network Function

ACM Reference format:


https://doi.org/10.1145/3098822.3098848

1 INTRODUCTION

Cellular networks are experiencing explosive growth along multiple dimensions: (i) traffic volumes (e.g., mobile traffic grew by 74% in 2015), (ii) the number and diversity of connected devices (e.g., projections show that by 2020 there will be 11.6 billion mobile connected devices including approximately 3 billion IoT devices [11]), and (iii) signaling traffic (e.g., signaling traffic in the cellular network is reported to be growing 50% faster than data traffic [31]).

These trends impose significant scaling challenges on the cellular infrastructure. In particular, there is growing concern regarding the scalability of the cellular evolved packet core (EPC) [21] infrastructure. The EPC is the portion of the network that connects the base stations to the IP backbone and implements cellular-specific processing on user's data and signaling traffic. Recent industrial and academic studies have provided mounting anecdotal and empirical evidence showing that existing EPC implementations cannot keep up with the projected growth in cellular traffic [10, 18, 19, 23, 37].

We postulate that the poor scaling of existing solutions stems from the manner in which existing EPC systems have been decomposed. More specifically, EPC systems today are factored based on traffic type, with different components to handle signaling and data traffic: the Mobility Management Entity (MME) handles signaling traffic from mobile devices and base stations, while the Serving and Packet Gateways (S-GW and P-GW) handle data traffic. The problem with this factoring is it complicates how state is decomposed and managed. As we elaborate on in §2, current designs lead to three problems related to state management:

(1) Duplicated state leads to frequent synchronization across components. In current EPCs, per-user state is often duplicated between components. For example, a user request to establish a cellular connection is processed by the MME which instantiates user state and then communicates this addition to the S-GW which in turn locally instantiates similar per-user state. A similar interaction takes place when user state is updated after mobility events. This duplication introduces complexity (e.g., implementing the protocols
Microboxes: High Performance NFV with Customizable, Asynchronous TCP Stacks and Dynamic Subscriptions

Guyue Liu*, Yuxin Ren*, Mykola Yurchenko*, K.K. Ramakrishnan†, Timothy Wood*
*George Washington University, †University of California, Riverside

ABSTRACT
Existing network service chaining frameworks are based on a “packet-centric” model where each NF in a chain is given every packet for processing. This approach becomes both inefficient and inconvenient for more complex network functions that operate at higher levels of the protocol stack. We propose Microboxes, a novel service chaining abstraction designed to support transport- and application-layer middleboxes, or even end-system-like services. Simply including a TCP stack in an NFV platform is insufficient because there is a wide spectrum of middlebox types—from NFs requiring only simple TCP bytestream reconstruction to full endpoint termination. By exposing a publish/subscribe-based API for NFs to access packets or protocol events as needed, Microboxes eliminates redundant processing across a chain and enables a modular design. Our implementation on a DPDK-based NFV framework can double throughput by consolidating stack operations and provide a 51% throughput gain by customizing TCP processing to the appropriate level.

CCS CONCEPTS
• Networks → Middleboxes / network appliances;

KEYWORDS
Middleboxes, NFV, Networking Stack, Service Chain

1 INTRODUCTION
Today’s enterprise and wide-area networks are filled with middleboxes [27] providing a wide range of functionality from simple firewalls to complex Evolved Packet Core (EPC) functions in cellular networks. Network Function Virtualization (NFV) platforms provide high performance packet processing by leveraging kernel bypass I/O libraries such as DPDK [1] and netmap [25]. However, these systems are packet-centric: they focus on providing efficient movement of packets through a chain of network functions that operate on each packet as it arrives. While this model can make sense for simple layer-2/3 processing, it becomes inefficient and inconvenient when building more complex functions operating at higher levels of the protocol stack.

Network functions that operate at the transport layer need to perform additional processing such as TCP bytestream reconstruction. This is a relatively heavyweight function since it involves copying packet data into a buffer, an action that is avoided in many layer-2/3 middleboxes that rely on “zero-copy” to achieve high throughput. High performance, user-space TCP stacks [12, 13] can be used by NFs to simplify this operation, but these libraries must be used individually by each NF, resulting in redundant computation if a chain of functions each perform TCP processing.

To illustrate the high cost of redundant TCP processing, Figure 1 shows the processing latency for a chain of NFs that perform TCP bytestream reconstruction using mOS [12] or simply forward individual packets at layer 2 (fwd in figure); to maximize performance, each NF runs on its own core. As the chain length increases, the latency for the NFs performing TCP processing increases substantially compared to that for

• Change from packet-layer to transport-layer processing for performance improvement
Summary

• Network Function Virtualization

• A case study
  – CoMb

• More challenges and solutions