Heterogeneous network policy enforcement in data centers

This item was submitted to Loughborough University’s Institutional Repository by the/an author.

Citation: CUI, L., TSO, F.P. and JIA, W., 2017. Heterogeneous network policy enforcement in data centers. IFIP/IEEE International Symposium on Integrated Network Management (IM 2017), Lisbon, Portugal, 8th-12th May 2017, pp. 552-555.

Additional Information:

- © 2017 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.

Metadata Record: https://dspace.lboro.ac.uk/2134/24640

Version: Accepted for publication

Publisher: © IEEE

Please cite the published version.
Heterogeneous Network Policy Enforcement in Data Centers

Lin Cui∗, Fung Po Tso†, Weijia Jia‡,
∗Department of Computer Science, Jinan University, Guangzhou, China
†Department of Computer Science, Loughborough University, UK
‡Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai China.
Email: tcuilin@jnu.edu.cn; f.p.tso@lboro.ac.uk; jia-wj@cs.sjtu.edu.cn

Abstract—With the emergence of network function virtualization, data center start to deploy a variety of network function boxes (NFBs) in both physical and virtual form factors in order to combines inherent efficiency offered by physical NFBs with the agility and flexibility of virtual ones. However, existing schemes are limited to exclusively consider physical or virtual NFBs, which may reduce the performance efficiency of services running atop. In this paper, we propose a Heterogeneous Network Policy Enforcement scheme (HOPE) to overcome these challenges. An efficient algorithm that can closely approximate optimal latency-wise NF service chaining is proposed. The experimental results have also shown that HOPE can outperform greedy algorithm by 25% in terms of network latency and is 56x more efficient than naive depth-first search algorithm.

I. INTRODUCTION

Cloud data centers deploy a great variety of network functions (NFs), such as firewall (FW), intrusion prevention/detection system (IP/DS), deep packet inspection (DPI), load balancer (LB), and etc., at various points in the network topology to safeguard networks and/or improve application performance [1]. Each NF is responsible for specific treatment of received packets, including forwarding, dropping, rate-limiting, inspecting and so on. Various permutations of NFs form an ordered chain, i.e., service chain [2], which is defined by network policy [3] and must be applied to packets.

Nowadays, NFs are either embedded in purpose-built proprietary hardware middleboxes or run as virtual instances on top of commodity servers through network function virtualization (NFV). We term both hardware middleboxes and virtualized servers for NFs as Network Function Boxes (NFBs). Physical NFBs are more efficient because they are built with dedicated hardware for optimizing the performance of specific functions but are proprietary and hence less extensible. On the other hand, virtualized NFBs have the agility for rapid on-demand deployment and programmability for software automation but they are less efficient due to virtualization overhead, resource sharing, and general-purpose hardware they sit atop [4]. Obviously, a NF can be independently allocated to different NFBs in the network or collocated with other NFs within a single NFB [2]. In addition to hardware middleboxes and general-purpose NFV servers, some simple NFs such as firewall and NAT can also be efficiently implemented in SDN switches [5].

Clearly, these distinctively different NFBs present an enormous opportunity for data centers by adopting mixture of both form factors in order to capitalize on both efficiency and flexibility [2]. Nevertheless, coming with this heterogeneous are significant challenges on the correct implementation of network policies: (1) Support for deployment of network policies is limited exclusively to physical or virtualized NFBs, there is no tools for supporting mixture of both [3]; (2) Migration of virtual instances of NFs will involve change of end-to-end path that will break legacy VLAN partition [6]; (3) The performance of virtual NFs are subject to the compute and storage capacities of commodity servers. This will lead to unpredicted performance such as end-to-end latency [7].

While the first and second challenges can be partially tackled with small tweaks on top of our previous work Sync [6], the third challenge remains the most prominent and needs solving urgently because data centers host applications that are largely latency-sensitive and are prone to unpredictable slowdown along the end-to-end links [8]. Existing works only combat latency in network switches and/or protocols running atop and overlook latency arises from NFBs [9].

In this paper, we propose a Heterogeneous Network Policy Enforcement scheme (HOPE) which meets this requirement. Our experimental evaluation demonstrates that HOPE can achieve optimal placement of NFs amongst heterogeneous NFBs.

The remainder of this paper is structured as follows. Section II describes the problem formulation and the model of HOPE. An efficient greedy scheme is proposed in Section III, followed by the performance evaluation of HOPE in Section IV. Section V concludes the paper.

II. PROBLEM MODELING

A. Overview

We consider a heterogeneous environment where NFs can be implemented different kinds of NFBs: (1) Hardware middleboxes are vendor specific, proprietary boxes for providing specific NFs. They are optimized for performance and less extensible. (2) NFV servers are virtualized that can run multiple, and theoretically, any types of virtual NFs. (3) Some simple NFs can also be implemented on switches or routers such as VPN, simple firewalls, LBs. They are amongst hardware middleboxes. However, SDN also allow us to exploit the OpenFlow switches to increase the performance of service chain by installing some rules (i.e., NF) to their tables [5].
We anticipate that the heterogeneous implementation of NFs will exist for the foreseeable future.

Denote \( B = \{b_1, b_2, \ldots \} \) to be the set of all NFBs in a data center. For a NFB \( b_i \), \( b_i, \text{capacity} \) denotes the maximum processing capability of \( b_i \), measured in number of packets per second (pps), e.g., 3800 pps [10]. \( b_i, \text{typeset} \) specifies the set of supported NF types on \( b_i \). NFV servers theoretically support all types of NF, while hardware MBs and switches can only support one or few types of NFs. We assume that memory space of NFBs are enough to accommodate states information of all NFs, i.e., bottleneck is the processing capacity.

Let \( N = \{n_1, n_2, \ldots \} \) be the set of all NF instances in data center. For a NF \( n_i \), the property \( n_i, \text{type} \) defines the function of \( n_i \), e.g., IP/DS, LB, or FW. \( n_i, \text{capacity} \) is essentially the processing capacity requirement of \( n_i \) in pps. \( n_i, \text{location} \) is the NFB that currently hosts \( n_i \). This set of NFs in \( N \) may belong to different applications, and are deployed and configured by a centralized Policy Controller [11].

Traffic in data center is largely flow-based [12]. In light of this, we define data center traffic as \( P = \{f_1, f_2, \ldots \} \). For each flow \( f_i \in P \), \( f_i, \text{src} \) and \( f_i, \text{dst} \) specify the source and destination VMs of \( f_i \) respectively, e.g., \( f_i, \text{src} = v_1 \) and \( f_i, \text{dst} = v_2 \). The data rate of \( f_i, \text{rate} \) is represented by data exchanged from VM \( f_i, \text{src} \) to VM \( f_i, \text{dst} \) per time unit.

The set of network policies is \( P \), which can be defined by administrators. For each \( f_i \in P \), there is a policy \( p_i \). \( p_i, \text{chain} \) defines the sequence of NFB types that all flows matching policy \( p_i \) should traverse in order, e.g., \( p_i, \text{chain} = \{n_1, n_2, n_3\} \), where, for example, \( n_1, \text{type} = \text{FW}, n_2, \text{type} = \text{IPS}, n_3, \text{type} = \text{Proxy} \). Specially, \( p_i = \emptyset \) means \( f_i \) is not governed by any policies. \( p_i, \text{len} \) is the length of \( p_i, \text{chain} \).

\( p_i, \text{chain} \) must be assigned to appropriate NFBs beforehand, and we assume there are enough NFBs to accommodate all required NFs. Since we consider heterogeneous NFs, there are various possible locations for each NF in \( p_i, \text{chain} \). For example, in the above example of \( p_i, n_1, \text{location} \) could be a core router, \( n_2, \text{location} \) could be a hardware NFB, and \( n_3, \text{location} \) could be a NFV server. An example of service chain is given in Figure 1.

**B. Delays with network functions**

There are many metrics to measure the efficiency of NF placement (service function chaining) for a policy such as communication cost [6][14]. In this paper, we mainly focus on the latency of a policy flow.

The total delay of a flow includes transmission delay among adjacent NFs and processing delay of NFs in the service chain.

1) Transmission delays: In order to steer traffic to the service chain, either Policy Based Routing (PBR) or VLAN stitching can be used in data centers [2]. For either case, the intended solution in this paper should be unaware of these schemes and is general and applicable to the schemes. So, we do not consider the detailed routing between two NFBs.

![Fig. 1: Service Chain Example: FW → LB → IPS → Monitor](image)

Since, in production data centers, the transmission delay of links in its path are relatively stable and can be easily obtained/estimated through large-scale measurement [15], we assume the transmission delay between two NFs is known and can be obtained through the controller. The controller will maintain a transmission delay matrix \( D \), \( D(n_i, n_j) = D(n_i, n_j) \) is the delay between \( n_i \) an \( n_j \). \( D(n_i, n_j) = -1 \) if the delay is unknown or they are unreachable.

2) Processing delays of network functions: We define service time \( t^i_s \) as the time that \( n_i \) takes to process a packet. Since many NFs such as firewalls and load balancers only process packet headers of which sizes are fixed, ignoring variable length data payloads. Thus, the service time \( t^i_s \) is a constant [16]. Specially, considering the processing capacity \( n_i, \text{capacity} \) of \( n_i \), \( t^i_s = 1/n_i, \text{capacity} \).

**C. Heterogeneous network policy enforcement problem**

The expected delay for flow constrained by policy \( p_i \) is:

\[
T(p_i) = D(f_i, \text{src}, p_i, \text{chain}[1]) + \sum_{j=1}^{p_i, \text{len} - 1} (D(p_i, \text{chain}[j], p_i, \text{chain}[j + 1]) + t_p(p_i, \text{chain}[j])) + D(p_i, \text{chain}[p_i, \text{len}], f_i, \text{dst})
\]

(1)

We aims to reduce the total delay by efficiently placing NFs onto heterogeneous NFBs while strictly adhering to network policies. Let \( A \) be an allocation of NFs. \( A(n_i) \) is the NFB which hosts \( n_i \), and \( A(b_j) \) is the set of NFs hosted by \( b_j \).

The Heterogeneous Network Policy Enforcement problem is defined as follows:

**Definition 1.** Given the set of \( F, P, \text{NFBs} B \) and \( D \), we need to find an appropriate allocation of NFs \( A \), which minimizes the total expected end-to-end delays of the network:

\[
\min \sum_{f_i \in F} T(p_i)
\]

s.t. \( A(n_i) \neq \emptyset \) \&\& \( |A(n_i)| = 1, \forall n_i \in p_i, \text{chain}, \forall p_k \in P \)

\[
\sum_{n_i \in A(b_j)} n_i, \text{capacity} < b_j, \text{capacity}, \forall b_j \in B
\]

(2)
Fig. 2: Example of service chain network for flow $f$ and $l = 3$.

The first constraint ensure that NFs of all service chains are appropriately accommodated by one NFB. The second constraint is the capacity constraint of all NFBs. It can be easily proven that the above problem is NP-Hard, by reducing from the Multiple Knapsack Problem (MKP).

III. HETEROGENEOUS POLICY ENFORCEMENT

In this section, we introduce HOPE, a Heterogeneous Network Policy Enforcement scheme.

A. Service chain network

We consider an online solution which process one service chain at a time when a new policy/flow requirement arrives. Suppose a flow $f_i$, constrained by $p_i$, arrives and we need to find appropriate NFBS to accommodate all NFs in $p_i$.chain with an objective to minimize its total expected delay $T(p_i)$. We define its Service Chain Network $G = (V, E)$, which is a $p_i$.len-tier directed graph. Nodes in the $j$th tier are NFBS defined by $B_j$:

$$B_j = \{b_k[p_i].chain[j] \in b_k.typeset \text{ and } \sum_{n \in A(b_k)} n.capacity + capacity \leq b_k.capacity, \forall b_k \in B_j\}$$

For a node $x$ in $j$th ($j \leq p_i$.len $- 1$) tier and $y$ in $(j + 1)$th tier, there is a directed edge from $x$ to $y$ if $y$ is reachable from $x$ and weight of the edge is $D(x, y) + t_p(y)$. It is possible that both $x$ and $y$ are the same NFB. In this case, $D(x, y) = 0$. Flow originates from the source ($f_i$.src) and terminate at the sink ($f_i$.dst). For a node $x$ in 1st tier, the weight of edges from $f_i$.src to $x$ is $D(f_i.src, x) + t_p(s)$. For a node $y$ in $l$th tier, the weight of the directed edges from $y$ to $f_i$.dst is $D(y, f_i$.dst$)$. Figure 2 shows an example of service chain network.

B. Shortest service chain path

Clearly, the route with smallest expected latency for a flow is the shortest path from source to sink, which we referred as ssp (Shortest service chain path). However, since nodes in different tiers of the service chain network can be the same NFB with limited capacity, we can not simply re-use traditional shortest first path algorithms, e.g., Dijkstra, Floyd.

The difficulty here is that two nodes that belong to different tiers in the service chain network, say $x$ and $y$, may be in the same NFB and share the same capacity. If we assign $f_i$ to $x$, it may saturate the NFB such that $y$ can not further accept $f_i$. In this case, we call them conflict nodes. A path from the source will be blocked by the latter one of the conflict nodes.

Hence, we design the SSP (Shortest Service Chain Path) to find the shortest path in this situation, as shown in Algorithm 1. Conflict nodes are handled in line 14. The shortest service chain path are maintained in $prev$ and can be obtained through function $getPath()$. We can easily derived that Algorithm 1 can always output a shortest service chain path. Due to page limitation, detailed proof is not shown here.

IV. EVALUATION

We have evaluated the performance of HOPE through extensive simulations in a fat-tree data center topology with factor $k$ ranged from 4 to 20 meaning that there are at most 2000 servers and 500 switches in these setups. Each NFB in our simulations is modeled with random residual capacity (number of packets it can process per second) and a set of NF types that it supports. NFBS are implemented through OpenFlow switches, hardware middleboxes or NFV servers. Each service chain is comprised of 1–4 NFs including FW, IPS, RE, LB and (traffic) Monitor [2]. A centralized controller is implemented to collect all network information that is needed, as defined in Section II to perform the HOPE scheme. To compare the performance of HOPE, we have also implemented two other approaches: Greedy, which always tries to assign NFBS with the lowest estimated latency, and Brute-force, which gives the optimal results but is not suitable for large-scale network.

Figure 3a shows the average latency of all service chain under different network scales with the factor $k$ of fat-tree ranging from 4 to 20. It shows that on average HOPE can
always find a service chain path with the same latency as Brute-force, which is optimal. The Greedy approach can deviate from both HOPE and Brute-force by up to 23%.

Figure 3b shows that HOPE and Brute-force schemes have identical CDF of latency for all policies for a large scale network when \( k = 20 \). Particularly, they can outperform Greedy scheme by 38% at 99 percentile. Figure 3c reveals that average latency increase linearly with the length of service chain when all NFBs have sufficient capacity for accommodating all NFs.

Figure 3d shows the average total running time to process a policy increases exponentially for all schemes. Greedy is the fastest methods, consuming only 50ms and 63ms for \( k = 18 \) and \( k = 20 \) respectively to complete a cycle. On the contrary, HOPE can complete within 1s for most scenarios, and Brute-force is much more slower. The results indicate that HOPE can be nearly 9 times slower as compared to Greedy but is 56 times faster than Brute-force. However, given the remarkable performance gain of HOPE over Greedy, we believe this trade-off in system running time is acceptable.

V. CONCLUSION

Network policies and service chains are important for the security and reliability of data centers. NFs of policies can be deployed in different environment, e.g., OpenFlow switches, hardware middleboxes and NFV servers. Such heterogeneous environment for policy allocation remain unexplored in previous works. In this paper, we study the Heterogeneous Policy Enforcement Problem with a focus on the latency. We proposed HOPE, which can find the optimal service chain path for each policy. Extensive simulation results have demonstrated the effectiveness and optimality of HOPE.

VI. ACKNOWLEDGEMENTS

This work is partially supported by Chinese National Research Fund (NSFC) Project No. 61402200; NSFC Key Project No. 61532013; National China 973 Project No. 2015CB352401; the UK Engineering and Physical Sciences Research Council (EPSRC) grants EP/P004407/1 and EP/P004024/1; Shanghai Scientific Innovation Act of STCSM No.15JC1402400 and 985 Project of SJTU with No. WF220103001.

REFERENCES


