A Model-Driven Graybox Approach to Rehoming Service Chains

Muhammad Wajahat*, Bharath Balasubramanian †, Anshul Gandhi*, Gueyoung Jung†,
Shankaranarayanan Puzhavakath Narayanan†
*Stony Brook University{*mwajahat,anshul*}*@cs.stonybrook.edu*
†AT&T Labs - Research {*bharathb,gjung,snarayanan*}*@research.att.com*

***Abstract*—Network clouds are typically private clouds owned by the network provider, consisting of a large number of geo-distributed sites with heterogeneous capabilities and small capacities. Each of these small clouds often run specialized service chains of Virtual Network Functions (VNFs), which need to meet strict Service Level Objectives (SLOs), especially along the lines of availability (e.g., First responder services). Hence, VNFs in such thinly provisioned clouds may need to be moved (*rehomed*), both within and across sites, much more frequently than in traditional public clouds (like Amazon's EC2 cloud), in order to meet the performance SLOs, when reacting to various cloud events like hotspots, interference from co-located VMs, failures and upgrades. Rehoming is also required by the infrastructure (platform) providers for various other reasons such as consolidation of resources for saving energy and improving the platform utilization. In this paper, we propose a model-based approach to show that naive strategies for rehoming, applied uniformly across all VNFs of the service chain, are often sub-optimal when considering different metrics like user-perceived service disruption time and the time taken to complete the rehoming action. Our model leverages the transparency between the services and platforms on private clouds (*grayness*), and provides appropriate rehoming recommendations based on various factors including service characteristics and runtime platform dynamics. We validate our models using a simple, yet ubiquitously deployed service chain, and using out-of-the-box rehoming options provided by Openstack, the most commonly used open-source cloud. Our results show that our graybox approach is able to achieve significant reductions in service disruption times and time taken for the rehoming action.**

I. Introduction

The drive towards network virtualization by large network providers like AT&T and Verizon [1] is replacing specialized networking hardware with commodity hardware running networking software. Due to this drive, buildings and central offices hosting networking equipment now morph into datacenters/sites of varying sizes hosting servers, leading to widely geo-distributed *private* clouds all the way from the center to the edge. Network provider deployments often have a few hundred such sites spread across tens of countries, with each site hosting anywhere between 10 and 500 compute servers [2]. Further, these sites are thinly provisioned and operate in resource constrained environments as they consist of limited (possibly heterogeneous) hardware capabilities based on the types of services they are envisioned to support.

The networking services hosted on these cloud sites often include complex service chains of Virtual Network Functions (VNFs). Figure 1 shows the example of a service chain at the provider edge that optimizes the video traffic through the Video Service Path, and lets all other traffic flow through the Regular Service Path. These service chains typically have stringent Service Level Objectives (SLOs) that need to be met, especially along the lines of availability. For example, first responder services (EMS, police, fire) require "5-9s" system availability [3], [4].

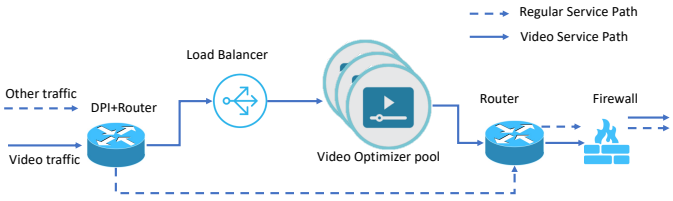


Fig. 1. A video optimizing service chain.

An important challenge in maintaining such stringent SLOs, especially in our thinly provisioned clouds, is the ability to react quickly to various cloud events that affect the availability and performance of these services. While events like hotspots and interference due to workload shifts in colocated VMs, lower the service performance, failures and upgrades lower the availability of the service. In order to maintain the SLOs, Network Providers often *rehome* (or move) one or more VNFs (or VMs, used interchangeably) of the service chain within the same site to a different host and sometimes even to a different cloud site. While some of these events, such as upgrades, are predictable, others are unpredictable and require reactive measures to quickly restore SLO compliance. Rehoming is also required by cloud operators for improving resource and energy efficiency (via consolidation of VNFs), and for leveraging newer hardware capabilities (via migration of service chains to the new infrastructure).

The typical approach to rehoming involves applying the same corrective action, such as VM migration, to all VNFs in a service chain. However, this may not be the optimal approach given the heterogeneity across VNFs in a chain. For example, consider a hotspot in one of the physical servers on which the video optimizer VNFs are hosted for the service chain in Figure 1. Knowing that the video optimizer VNFs are stateless makes rebuilding (which shuts down and recreates the VM) a quicker option when compared to the more time-consuming cold or live migration action. Further, the optimal

set of actions depend on the exact metric(s) being considered – if optimizing for the time taken to complete the rehoming action, rebuild may be a better option; however, if optimizing for the application downtime, migration may be better as it can minimize the disruption of user traffic. Finally, the optimal rehoming actions could change dynamically as the state of the VMs (e.g., the disk size) or the state of the underlying platform (e.g., total traffic) evolve over time.

In this paper, we present a model-based approach to identify the optimal set of rehoming actions across all VNFs for a given service chain. We adopt a graybox approach to the cloud and leverage the exchange of information across the platform and the services in private clouds to minimize the cost of rehoming. Specifically, given a service chain of VNFs, and a set of rehoming actions, we find the best combination of actions for the different VNFs of the chain. The key idea behind our approach is to develop models for each action that capture the impact of various parameters that affect the service downtime and the time taken for the rehoming action of the VNFs. Our models leverage information exchange between the services and platforms on private clouds, such as specifications on whether a service is stateful (hence has to be migrated) or stateless (hence can potentially be rebuilt).

We experimentally validate our models and rehoming recommendations using a simple service chain deployment on an Openstack cloud, and using out-of-the box rebuild and migrate actions provided by Openstack. Our results show that, by understanding the characteristics of the VNFs, our graybox approach to rehoming is able to reduce the time taken for the rehoming action by as much as 42% and the service downtime by as much as 8%. Our results also show that our models can estimate the time taken by a rehoming action for a single VNF or all the VNFs in the service chain with reasonable accuracy. We conclude by illustrating the challenges involved in estimating the service downtime for the entire chain due to dependencies between the VNFs and the impact of a rehoming action on the shared infrastructure resources like bandwidth.

II. Why does service-specific information matter to rehoming?

In this section, we motivate the need for a graybox approach, that is, the need for service information in helping the network provider perform optimal rehoming actions for service chains. We perform a simple, yet illuminating experiment on a real testbed (details of the experimental setup in Section V) with a service chain consisting of two ubiquitous VNFs: a network firewall filtering packets to a web service hosted on an Apache Tomcat web server. In the absence of any information from the client who owns the service chain, when these service chains have to be rehomed, the provider has no choice but to migrate both VNFs, thus preserving all their state. Clearly, the firewall is a stateful service given its stored filtering rules, and hence the only possible rehoming action for it is migration. However, if the service can provide a simple piece of information to the network provider that its web server VM is stateless, then the provider has two potential rehoming actions: migrating the web server VM or rebuilding it from scratch on a new host.

Action	Firewall Action Time	Web Server Action Time	Connectivity Downtime
Migrate Both	245s	197s	336s
Migrate Firewall Rebuild Web Server	226s	191s	324s

TABLE I
Results for the illustrative example in Section II.

Our experimental results confirm that these actions have implications on several metrics, affecting both VNFs of the service chain. In Table I, we illustrate the effect of two rehoming combinations: (a) migrate both VMs, and (b) migrate firewall, rebuild web server on three different metrics: (i) firewall action time, i.e., time taken for the rehoming action of firewall to complete, (ii) web server action time, and (iii) the connectivity downtime of the service chain as a whole. All our results have been averaged across multiple independent runs (see Section V for full details). The table clearly shows that for all three metrics the "migrate firewall, rebuild web server" option achieves better performance than "migrate both". Hence, armed with service information, for this example, the provider can perform the more intuitively optimal action of migrating the (stateful) firewall and simply rebuilding the (stateless) web server. Our graybox approach enables us to understand the characteristics of the VNF like replication, statefulness, etc., which are defined along with the VNF models as part of the VNF certification process [5], [6].

III. Solution Design

Our solution to rehoming relies on a model-driven approach to predicting the costs of various rehoming actions. Specifically, we first perform experiments on each VM hosting the VNF for various actions (rebuild and migrate), and then, using the empirical data, we model the various costs of rehoming as a function of the VM and cloud infrastructure parameters. Finally, we leverage these models at run-time to predict the right set of (possibly heterogeneous) *simultaneous* rehoming actions to perform for the different VNFs of a service chain to optimize for the specified cost metric.

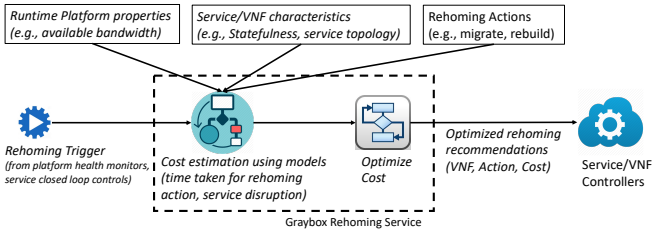
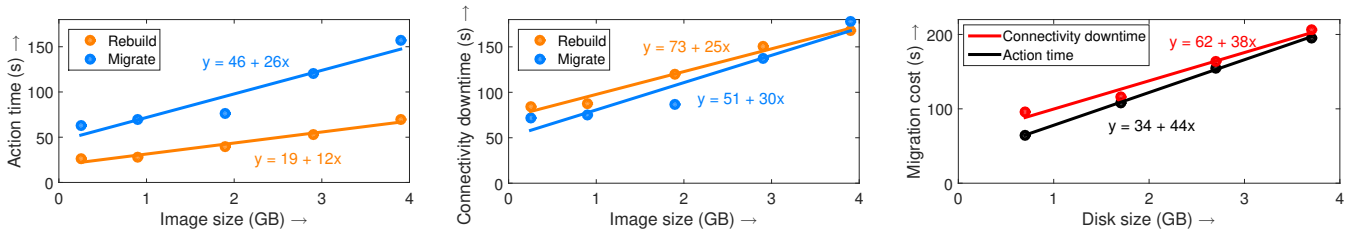


Fig. 2. Illustration of our solution design.

Figure 2 shows the design of our solution. We assume that the decision to rehome is triggered by an external service, such as a monitoring system (e.g., Ceilometer in OpenStack [7]). Once triggered, our solution leverages the pre-existing cost models to determine the optimal rehoming actions for each VNF in the service chain based on the provided cost/utility



(a) Action time as a function of image size. (b) Connectivity time as a function of image size. (c) Costs as a function of disk size.

Fig. 3. Empirical and modeled results for action time and connectivity downtime of rebuild and cold migrate as a function of the image size and disk size. For Figure 3(c), the base image size is 0.25GB.

function. These rehoming decisions are then relayed to the service or platform controller (e.g., OpenStack Nova), for execution. Finally, logs from the execution are parsed and the relevant statistics are passed to the model to update it for use in future rehoming decisions.

IV. Rehoming Cost Analysis and Modeling

We now present one of our key contributions — analysis and modeling of the rehoming cost for different rehoming actions. We specifically consider the rebuild and cold migrate rehoming actions, and model two different costs: (i) time to perform the rehoming action, as logged by OpenStack, referred to as **action time**, and (ii) time until the service chain's end-to-end connectivity is restored, referred to as **connectivity downtime**; we measure this by calculating the delay between successful pings from client to server in the service chain. In general, the rehoming cost can be expressed as a utility function in terms of various individual costs; we focus on these two costs in this paper as illustrative examples.

We consider a simple iPerf [8] service chain deployed in CloudLab [9], consisting of an iPerf client VM, a router VM, a firewall VM, and an iPerf server VM, to collect empirical data for modeling (see Section V-A for details of our experimental setup). We focus on modeling the rehoming costs for the firewall VM; we expect similar results for other VMs as well, though the post-bootup process may be different, depending on the VM's functionality. It is important to note that the results in this section are for a single generic VM being rehomed; we show in Section V-D how these results can be applied to predict the optimal simultaneous rehoming actions for multiple VMs in the service chain.

A. Empirical analysis of rehoming costs

Figures 3(a) and 3(b) show our empirical results (circles) for the rehoming costs of rebuild and cold migrate of the firewall VM as a function of the VM image size; we modify the image size by adding some data to the base image and taking snapshots. We see that the various costs (action time and connectivity downtime) increase with the image size, as expected. Also, connectivity downtime is typically higher than action time as connectivity typically requires some post-boot processes to execute, such as network configuration. However, surprisingly, the superiority of one rehoming approach over another *depends* on the rehoming cost metric.

In Figure 3(a), we see that the *action time is higher for migrate than for rebuild*. This is to be expected as migrate requires data transfer over the network, including contents of the VM, from one host (source) to another (target), whereas rebuild involves rebuilding the VM on the same host from the base image. In Figure 3(b), we see that the *connectivity downtime is typically higher for rebuild than migrate*. This is because rebuild necessitates post-boot (re)configuration, such as reconfiguring the network, which impacts the time to bring up the VM. Specifically, we found that host-ssh keys need to be regenerated on rebuild as the VM is treated as a first boot; this is not the case for migration where the VM is not treated as a fresh VM. This shows that the optimal rehoming action depends on the specific cost (or utility) function being considered. Figure 3(c) shows our empirical results (circles) for the rehoming costs of cold migrate as a function of the VM disk size; we use a base image size of 0.25GB for this experiment and modify the disk size by adding software and data. We see that all costs increase with the disk size under migration, as expected. Note that the disk size does not matter for rebuild as only the image is rebuilt and disk content is lost.

B. Modeling the rehoming costs

To leverage the above empirical results in practice, we now build regression models for each rehoming action to allow prediction of rehoming costs for other VMs, as in Section V. The solid lines in Figures 3(a) and 3(b) show our regression fit for the action time and connectivity downtime, respectively, as a function of image size for rebuild and migrate. We find that a linear model works well for these observations with an R^2 value of 0.97 and 0.91 for the action time of rebuild and migrate, respectively, and an R^2 value of 0.98 and 0.91 for the connectivity downtime of rebuild and migrate, respectively. The linear trend suggests that the rehoming costs are proportional to the image size, as expected (since the image needs to be copied in case of rebuild and migrate).

Finally, Figure 3(c) shows our regression fit (solid lines) for the various migrate costs as a function of disk size. A linear fit works very well in this case, with an R^2 value of 0.99 and 0.98 for the action time and connectivity downtime, respectively. Recall that disk size does not impact the cost of rebuild, so we do not model this case.

Given the dependence of migration costs on image size and disk size, we also build multiple linear regression models for

migration costs using the empirical results obtained above. Our final migration cost model for action time is $8 + 49D + 24I$ (R^2 value of 0.77), and that for connectivity downtime is $22 + 47D + 23I$ (R^2 value of 0.78), where I is the image size and D is the disk size, in GB. For rebuild cost models, we use the regression fits listed in Figures 3(a) and 3(b): $19 + 12I$ for action time and $73 + 25I$ for connectivity downtime.

Note that the above modeling approach relies on our empirical results, which are specific to the underlying infrastructure. We will investigate the possibility of making these models infrastructure-independent in future work by including hardware-level details, such as available bandwidth and CPU processor speed, as part of the modeling step.

V. Experimental Results for Chain Rehoming

We now present our experimental results for rehoming of a service chain wherein multiple VNFs of the service chain are simultaneously rehomed; this scenario mimics a real deployment where the entire chain needs to be rehomed in response to either a hotspot or a maintenance or failure issue. We will also present evaluation results of our model-based rehoming approach in this section. We first describe our experimental setup, and then discuss our evaluation results.

A. Experimental Setup

Our experimental testbed comprises of bare metal servers in the CloudLab OpenStack setup (Clemson site). Each server has two Intel E5-2683 CPUs with 256GB Memory and a dual-port Intel 10Gbe NIC (X520). For the OpenStack (Mitaka) setup, one of the machines acts as the controller and the remaining are configured as compute nodes. We deploy our service chain on VMs hosted on this OpenStack setup.

Figure 4 illustrates our service chain composed of a client VM (outside the network provider cloud), a firewall VM, and an application/web server (Tomcat) VM; similar chains are commonly employed in practice to securely serve end-users [10]. The client VM employs httperf [11] as the web load generator and logs end-to-end response times. The client VM has 4vCPU, 8GB RAM, and 80GB Disk, the firewall VM has 2vCPU, 4GB RAM, and 40GB Disk, and the web server VM has 4vCPU, 8GB RAM, and 80GB Disk. We use Ubuntu 14.04 server (kernel version 3.13.0-123) as the OS for all VMs.

Each VM has multiple NICs and IP forwarding enabled in order to route traffic through the chain; static routes are configured on each VM to properly route traffic. The corresponding ports for these NICs are also configured in neutron to allow traffic pass through. The firewall VM has static routes and uses IPTables to enforce rules to allow only certain traffic meant for the application server. The web server VM serves client content using Tomcat via a Java servlet that uses JSP.

B. Methodology

Our focus in the experiments will be on the action time of rebuild and migrate for the firewall VM, referred to as **FW**, and the Tomcat web server, referred to as **WS**. Additionally, we will consider the connectivity downtime of the service chain (time until the client VM can ping the WS), and the **application downtime**, which is the time until the client can access the files hosted on the WS. Note that application downtime is a chain specific metric, and in our case is expected to be typically higher than connectivity downtime as the WS needs to start up Tomcat and other relevant services before the client can access the content hosted by Tomcat on the WS.

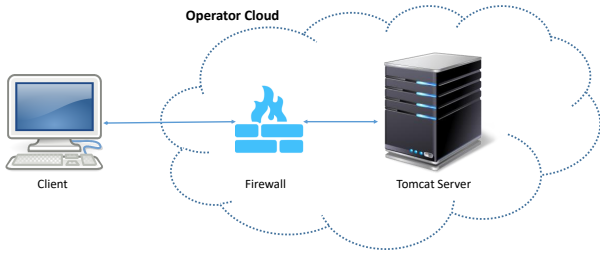


Fig. 4. Illustration of our service chain used for experimental results comprising a client VM, a stateful firewall VM (FW), and a Tomcat application/web server VM (WS), deployed in the network provider cloud.

Under our graybox approach, we consider that the network provider knows about the statefulness of FW and the statelessness of WS (see Section II). Thus, the only possible rehoming action for FW is migrate (since rebuild will result in loss of state), whereas WS can be migrated or rebuilt. Based on the above, we experiment with the following simultaneous rehoming options:

- **Migrate Both (MB):** In this rehoming approach both the FW and the WS are simultaneously migrated. Note that OpenStack decides the target host for migration.
- **Migrate FW Rebuild WS (MFRW):** In this rehoming approach we migrate FW but rebuild WS.

We repeat each experiment 6 times and report the average numbers, after removing outliers.

C. Experimental Results

Tables II (columns "FW action" and "WS action") and III (column "Downtime") show our experimental results for action time and connectivity downtime, respectively, for various scenarios under Migrate Both (MB) and Migrate FW Rebuild WS (MFRW) rehoming actions. Note that the scenarios listed in these tables are different from those analyzed in Section IV because: (i) they have different image sizes (larger than those considered before), (ii) two VNFs are being simultaneously rehomed, and (iii) the service chain is different. Thus, we consider these experiments as our test set when evaluating our model-based approach (that was trained on the experimental observations in Section IV) in Section V-D.

1) Action time: Starting with Table II, we see that MFRW has lower WS action time than MB in all cases, with an average reduction of about 20%. This shows that rebuild has a lower action time than migrate, which we also observed in Section IV. For FW action time, we see that MB and MFRW

FW image	WS image	FW disk	WS disk	Rehoming	FW action	WS action	FW model	WS model	opt prediction
6GB		12MB	16MB	**MB**	137s	154s	153s (11.5%)	153s (0.4%)	**yes**
				MFRW	143s	133s	153s (6.7%)	91s (31.7%)	
		2GB	146MB	**MB**	210s	127s	250s (19.5%)	159s (25.7%)	**yes**
				MFRW	236s	99s	250s (6.0%)	91s (8.0%)	
10GB		696MB	33MB	**MB**	245s	197s	287s (17.2%)	254s (28.9%)	**yes**
				MFRW	226s	191s	287s (26.8%)	140s (26.6%)	
		3.7GB	47MB	**MB**	372s	279s	434s (16.8%)	255s (8.6%)	**yes**
				MFRW	374s	161s	434s (16.0%)	140s (12.8%)	

TABLE II
EXPERIMENTAL AND MODEL-PREDICTED RESULTS FOR THE ACTION TIME OF FW AND WS UNDER VARIOUS SCENARIOS.

FW image	WS image	FW disk	WS disk	Rehoming	Downtime	FW model	WS model	Downtime model	opt prediction
6GB		12MB	16MB	**MB**	256s	161s	162s	162s (36.8%)	**no**
				MFRW	241s	161s	221s	221s (8.3%)	
		2GB	146MB	**MB**	281s	255s	168s	255s (9.1%)	**-**
				MFRW	259s	255s	221s	255s (1.4%)	
10GB		696MB	33MB	**MB**	336s	290s	259s	290s (13.7%)	**no**
				MFRW	324s	290s	323s	323s (0.4%)	
		3.7GB	47MB	**MB**	466s	431s	259s	431s (7.4%)	**-**
				MFRW	453s	431s	323s	431s (4.8%)	

TABLE III
EXPERIMENTAL AND MODEL-PREDICTED RESULTS FOR THE CONNECTIVITY DOWNTIME OF OUR SERVICE CHAIN UNDER VARIOUS SCENARIOS.

have similar costs, with an average difference of 2.6% across all 4 cases, with a maximum of 12.7% difference for the 6GB image size and 2 GB FW disk size scenario. This is to be expected as the FW action in MB and MFRW is the same — migrate. The observed difference is due to the variation in experiments, despite the several runs.

Without a graybox approach, the network provider would have to assume that both VNFs are stateful and only consider MB as a viable option, as otherwise there is the risk of losing data when performing a state-oblivious rehoming action. However, with the graybox approach, we consider the stateful/stateless nature of the VNFs, and realize that MFRW is also a viable option. Thus, using partial information from the tenant, the network provider can save significantly on action time, by as much as 42% in case of WS action time for 10GB image size and 47MB WS disk size experiments. This result also shows that homogeneous rehoming actions (such as MB) are not always optimal, and heterogeneous actions (such as MFRW) should also be considered.

2) Connectivity downtime: We see, from Table III, that MFRW provides slightly lower connectivity downtime than MB for all scenarios. The average reduction in downtime is about 5%, with a peak reduction of 8% for the 6GB image size and 2GB FW disk size experiments. This again shows that a graybox approach can provide better rehoming results, using the MFRW option, than naive (homogeneous) rehoming, which would only use MB. We relied on this argument when motivating the graybox approach in Section II.

3) Application downtime: While not shown here, we find that the application downtime numbers are similar to the connectivity downtime numbers reported under the "Downtime" column of Table III. In particular, the application downtime (time until the client VM can access the content hosted by Tomcat on WS) continues to be slightly lower under MFRW than under MB.

However, when we augment the Tomcat server of WS to also install the JDK 7 SDK and additional application code hosted online (to deploy additional WAR files), we find that the application downtime is *lower* under MB than under MFRW. This is because under MFRW the WS is rebuilt, and since we now require Tomcat to use SDK and host additional WAR files, the WS has to (apt-get) install SDK and download the additional code upon reboot, resulting in higher application downtime; note that this does not affect connectivity downtime which is the time until client can ping the WS. By contrast, under MB, the WS is migrated and so contains the SDK and additional (already compiled) WAR files after migration, thus obviating the need to (re)install these components. This further motivates the graybox approach which can leverage such information (additional post-bootup requirements) to make a more informed rehoming decision; in this case, MB is preferred over MFRW.

D. Model Evaluation

We now evaluate the efficacy of our model-based rehoming recommendations. We follow the approach detailed in Section III to make our recommendations for the rehoming action for each VNF of the service chain in Tables II and III (test data) based on the regression models built in Section IV using single VNF rehoming experiments (training data).

1) Action time prediction: Starting with the action time predictions in Table II (columns "FW model" and "WS model"), we find that our average prediction error for the FW rehoming action is 15.1% and that for the WS rehoming action is 17.9%, across all rows. We also use our predictions to recommend the optimal rehoming actions; in this case, the options are MB and MFRW. For this, we use the sum of action times as the metric to decide on the rehoming actions. Column "opt prediction" in Table II shows whether we correctly predict the optimal rehoming actions by comparing with the empirical values of action times obtained for FW action and WS action.

We see that our model is able to *accurately predict the optimal rehoming option for the action time metric in all scenarios*.

2) Connectivity downtime prediction: Our average prediction error for the connectivity downtime (column "Downtime model") is 10.3% across all rows of Table III. We now apply our predictions to recommend the optimal rehoming action for connectivity downtime as the metric. For this, we use the maximum of the predicted FW downtime and WS downtime. We find that our recommendation does not match empirical observations (column "opt prediction" in Table III). In some cases (marked as "-"), a comparison cannot be made as our model predicts that MB and MFRW should have similar downtime, but there is some difference observed in practice.

Looking at the empirical and model-predicted downtimes in Table III, we see that the model consistently underpredicts the connectivity downtime. This is to be expected as our model does not yet take network contention into account; this is because the model is trained on single VM rehoming but is being applied to simultaneous rehoming which incurs higher network contention. This result also shows that it is not trivial to extrapolate results observed for a single VM rehoming to predict those for multi VM rehoming. These observations motivate the need for further work in this direction, which we discuss in Section VII.

VI. Related Work

The prior work in this area has mostly been on model-driven approaches, but focused only on VM migration. For example, Nathan et al. [12] perform a thorough evaluation of existing models to predict VM migration time and propose a new model that takes into account other important factors such as the writable working set size and pages that are frequently dirtied. Mistral [13] attempts to optimize the overall *data center utility* by choosing adaptation actions such as increasing the CPU allocation, migrating VMs, and restarting hosts. Hence, Mistral may lead to sub-optimal decisions from the perspective of each service chain. On the other hand, we focus on the specific actions that can be taken for rehoming service chains to optimize *chain-specific* metrics such as action time and connectivity downtime. Wood et al. [14] espouses a graybox approach for VM migration taking into account OS and application-level statistics. Our graybox rehoming simply involves information from the user as to the nature of the VMs (stateful/stateless) in the service chain. While there has been work on holistic models for service chains, their focus has mostly been on various factors that influence *initial placement* such as the hardware and resource constraints [15], [16].

VII. Conclusion and Future Work

Service chain rehoming, in response to hotspots, upgrades or failures, is an important aspect of cloud management. In thinly provisioned network provider clouds, rehoming has to be performed much more often than in public clouds and thus has considerable implications on both cloud utilization and service availability. However, network clouds are typically private clouds wherein the service owner (customer) can provide information to the network provider to enable more informed, and hence optimal rehoming actions. In this paper, we present a graybox approach to rehoming service chains in private cloud environments. By leveraging information about the service, we design a model-based approach to determining the optimal rehoming action for each VM of a customer's service chain deployment. Our experimental results on a CloudLab testbed highlight both the efficacy of our model and the performance improvement that can be achieved by a graybox solution.

	Image Size	**Disk Size**	**Band-width**	**Instance Size**	**Page Dirty Rate**
Migrate	Yes	Yes	Future	Future	Future
Rebuild	Yes	No	Future	Future	No
Live Migrate	Future	No	Future	Future	Yes
Drain & Move	Future	Future	Future	Future	Future

TABLE IV
Factors affecting different rehoming actions.

We have several concrete steps of future work. First, we wish to extend the set of possible rehoming actions. For example, if a service VM is already replicated, then a potential rehoming action is to just *drain and move* the VM traffic to its replica. Second, we want to refine our VM-specific models to incorporate more relevant infrastructure parameters, like the available virtual memory and bandwidth, and provide unified models that *compose* different rehoming actions to allow us to predict the rehoming performance of the service chain as a whole, as mentioned in Section V.

Table IV lists the various actions and factors that we wish to consider for rehoming. The entries marked as "Yes" were studied in this paper, whereas those marked "No" were found to not have much impact on rehoming, and were thus omitted from discussion. The entries marked as "Future" are the ones we wish to explore as part of future work. We did experimentally investigate live migration, but found that the results for action time and connectivity downtime were noisy (likely due to the variance in scheduling times of the different subphases of live migration in OpenStack); we will explore live migration more thoroughly in future work. Finally, we wish to fine-tune our models to predict the rehoming performance of several commonly found service chains, thereby creating a repository of easily accessible information for customers and network providers. The service chain used in our evaluation, though representative of real deployments, is only one example; we will experiment with other VNF service chains with different functionalities and resource usage characteristics as part of future work. Our eventual goal is to build a comprehensive graybox solution that leverages service-specific information to seamlessly manage customer service chains to maintain performance SLOs in response to changing workload and platform conditions. Our efforts in this paper represent the first step towards this goal.

Acknowledgements

This work was partially supported by NSF CNS grants 1717588 and 1617046.

REFERENCES

[1] "Unraveling AT&Ts and Verizons Virtualization Vendors," https://www.sdxcentral.com/articles/news/unraveling-att-and-verizons-virtualization-vendors/2016/08/.

[2] "AT&T DataCenter locations," https://www.business.att.com/solutions/Service/cloud/colocation/data-center-locations/.

[3] "First Responder Network," https://www.firstnet.gov.

[4] "Public Safety Grade Features in FirstNet," https://www.illinois.gov/firstnet/resources/documents/faqs.pdf.

[5] "Momentum has grown for vnf certification," https://www.redhat.com/en/blog/momentum-has-grown-vnf-certification.

[6] "Cisco enterprise nfv open ecosystem and qualified vnf vendors," https://www.cisco.com/c/en/us/solutions/collateral/enterprise-networks/enterprise-network-functions-virtualization-nfv/nfv-open-ecosystem-qualified-vnf-vendors.html.

[7] Openstack.org, "Ceilometer," https://wiki.openstack.org/wiki/Ceilometer.

[8] "iperf," https://sourceforge.net/projects/iperf.

[9] "Cloudlab," https://www.cloudlab.us/.

[10] "Load balancing/Firewalling Web server using a service chain," https://www.ctl.io/knowledge-base/network/how-to-view-source-ip-in-web-server-logs-when-using-load-balancing/.

[11] D. Mosberger and T. Jin, "httperf—A Tool for Measuring Web Server Performance," *ACM Sigmetrics: Performance Evaluation Review*, vol. 26, pp. 31–37, 1998.

[12] S. Nathan, U. Bellur, and P. Kulkarni, "Towards a comprehensive performance model of virtual machine live migration," in *Proceedings of the Sixth ACM Symposium on Cloud Computing*, ser. SoCC '15. New York, NY, USA: ACM, 2015, pp. 288–301. [Online]. Available: http://doi.acm.org/10.1145/2806777.2806838

[13] G. Jung, M. A. Hiltunen, K. R. Joshi, R. D. Schlichting, and C. Pu, "Mistral: Dynamically managing power, performance, and adaptation cost in cloud infrastructures," in *2010 IEEE 30th International Conference on Distributed Computing Systems*, June 2010, pp. 62–73.

[14] T. Wood, P. Shenoy, A. Venkataramani, and M. Yousif, "Black-box and gray-box strategies for virtual machine migration," in *Proceedings of the 4th USENIX Conference on Networked Systems Design & Implementation*, ser. NSDI'07. Berkeley, CA, USA: USENIX Association, 2007, pp. 17–17. [Online]. Available: http://dl.acm.org/citation.cfm?id=1973430.1973447

[15] B. Addis, D. Belabed, M. Bouet, and S. Secci, "Virtual network functions placement and routing optimization," in *2015 IEEE 4th International Conference on Cloud Networking (CloudNet)*. IEEE, Oct. 2015, pp. 171–177. [Online]. Available: http://dx.doi.org/10.1109/cloudnet.2015.7335301

[16] H. Moens and F. D. Turck, "Vnf-p: A model for efficient placement of virtualized network functions," in *10th International Conference on Network and Service Management (CNSM) and Workshop*, Nov 2014, pp. 418–423.