Platform for Evaluating Real-Time Resource Management Algorithms for Network Function Virtualization

Dept. of CIS - Senior Design 2014-2015*†

Alex Brashear
brashear@seas.upenn.edu
Univ. of Pennsylvania
Philadelphia, PA

Alex Brashear
brashear@seas.upenn.edu
Univ. of Pennsylvania
Philadelphia, PA

Alex Brashear
brashear@seas.upenn.edu
Univ. of Pennsylvania
Philadelphia, PA

Alex Brashear
brashear@seas.upen
Univ. of Pennsylvan
Philadelphia, PA

ABSTRACT

Network Functions Virtualization (NFV) is an approach to implementing network infrastructure that emerged in 2012. NFV aims to address the inflexibility of classical network hardware by leveraging virtualization technology to consolidate network equipment onto industry standard servers. To date, several reports have demonstrated the feasibility of virtualizing network functions. In order for NFV to succeed in practice, virtualized network systems must account for the unique real-time requirements of network functions when approaching such problems as resource management. To that end, Professor Linh Phan and her research group at Penn have developed algorithms to manage resources in the context of NFV. One use case of interest is to provide network services that meet functional requirements, which are typically end-to-end latency requirements. In this project, we propose to design, implement, and evaluate a testing platform that can simulate real-time scheduling of virtual machines as well as functional and nonfunctional services in order to evaluate the effectiveness of the algorithm. We will collect metrics to understand, validate, and later improve the real-time scheduling algorithms. On the hardware side, we will deploy the platform on four 32-core computers with network switches connecting them. The testing platform will consist of an orchestration layer that uses their algorithms to manage resources and a cluster of computers that provide computing and networking resources. This system will enable the evaluation of the algorithms and aid in the development of future algorithms for efficiently implementing network functions.

1. INTRODUCTION

A vast number of hardware equipments (known as middleboxes) provide important services for the network such as mediating data packets being sent to and from one network node to another. These critical services filter or manipulate packets in order to improve efficiency and maintain security. For instance, a firewall is one such example of a middlebox that filters out malicious or irrelevant traffic. However, network hardware poses a growing problem because it is difficult to install and maintain. Furthermore, middleboxes burden enterprises with financial and administrative cost [white paper]. In the same way that cloud computing freed service providers from the management of physical hardware, Network Functions Virtualization (NFV) has been proposed as a solution to move network hardware functions to the cloud, where hardware services will be managed by software. Through NFV, it is much easier to install, manage, and upgrade hardware services [white paper].

Now, suppose a service provider wants to virtualize hardware network functions in order to take advantages of NFV such as economic savings, automation, and scalability. To satisfy its customer, the service provider has to finish processing requests with a reasonable latency performance. If a customer imposes a certain latency requirement for their service request, how can the service provider schedule services in the cloud in order to meet this criteria? Although research has shown that it can be beneficial to move network hardware to the cloud, few studies have been conducted to suggest ways to optimally manage resources for network functions in the cloud in real-time systems.

Recently, Dr. Phan at the University of Pennsylvania has developed an algorithm that can schedule and manage NFV services in the cloud in real-time, meaning services will be effectively scheduled on virtual machines per each new customer requirement. The algorithmâĂŹs main goal is to create effective configuration of virtual machines in the cloud and schedule services in a way that minimizes latency in order the meet the latency requirement. At a high level, the algorithm uses a combination of linear programming and real time analysis to determine the assignment of services to virtual machines in the cloud. In middleboxes, a data packet coming from a customer will be processed by the first service, and then the output of that will be sent to the second service to be processed, then the third, and so on. When scheduling services in the cloud, it is optimal to place services within close proximity to one another in order to minimize the latency that will be incurred with the distance that packets have to travel from service to another, after being processed. Another important consideration that the algorithm looks at is effective ways to schedule CPU and network resources on the cloud. Both allocations of CPU and network resources occur in real time based on the latency requirement imposed by customers. The algorithm tries to ensure that the end-to-end latency, or the total latency acquired from processing a data packet through the chain of services from the virtual machines, is minimized.

Currently, no analysis platform exists to evaluate effec-

*Advisor: Insup Lee (lee@cis.upenn.edu).

†Do not list your advisors amongst the authors as that may cause Google Scholar to add this work to their list of publications. Your advisor must also sign a hard-copy of your proposal.

tive resource management algorithms for virtualized network functions on the cloud. As our research project, we propose a testing platform that can be used to analyze the ability of network service scheduling algorithms to meet customer latency requirements. Specifically, we propose to create software that simulate a service provider in the cloud handling customer requests. Through careful design of this software, it can be used to test and prove the correctness of real-time scheduling algorithms. On a broader standpoint, this software will ultimately help cloud service provider make real-time latency guarantees to customers. The specifics of this system is described below.

2. RELATED WORK

Although research into Network Functions Virtualization is a recent phenomenon, substantial progress in the field has led to numerous technological innovations. The following research examples serve as a collective âĂIJproof-of-concept,âĂİ demonstrating that, among other things, network functions can be virtualized and algorithms can be designed to minimize latency. Furthermore, they suggest that NFV is not only technologically feasible but economically viable as well.

A white paper published by the European Telecommunications Standards Institute defined Network Functions Virtualization and outlined the benefits and challenges associated [1]. This paper covers many practical aspects of NFV and is a key motivation for research in this area, and especially useful to our research because it clearly defines the goals of NFV in which we can use to evaluate our system and measure our progress.

Making Middleboxes Someone ElseâĂŹs Problem: Network Processing as a Cloud Service [2] is one of the earliest practical studies of moving middlebox processing to the cloud. Like the white paper, this publication gives strong motivation for the field by surveying the current state of network enterprises. This research highlights several design choices of the system they implemented which successfully mitigates potential weaknesses of NFV. This study touches upon scaling middleboxes in response to load, but does so on a small scale and does not impose time constraints on the network functions.

One specific technology that is relevant to our research is ClickOS and the Art of Network Function Virtualization [3], which deals specifically with high-performance virtualization of network functions. The strategies they used to optimize hypervisors and operating systems for network processing will be useful for building a performant system, and the fact that their patches to software such as Xen are open source means that we can use components of their work for high-performance NFV.

Design and Implementation of a Consolidated Middlebox Architecture [4] will also be invaluable for reference because it deals specifically with deploying multiple virtualized middleboxes, which is exactly what we will be dealing with. The general techniques employed here will be useful, but they still do not impose time constraints and are more interested in aggregate statistics than specific reliability and performance quotas.

Additional research such as Theory of Scheduling and Routing [5] and Real-Time Multi-Core Virtual Machine Scheduling in Xen [6] will help us understand the theoretical limits and state-of-the-art of scheduling.

This project, which will draw heavily upon the aforementioned research and technology as groundwork, gives unique consideration to performance guarantees in real-time systems. Current NFV resource-management algorithms seek to reduce average latency across all requests, but they cannot guarantee that a specific task will complete given a time constraint with minimal variance. This project establishes a platform for evaluating algorithms which seek compliance with various specific performance constraints.

3. PROJECT PROPOSAL

Our group will develop software to configure a cluster of computers to act as a network functions service provider. Real-time resource management algorithms can then be evaluated by using them to manage the providerâĂŹs computing and networking resources and simulating customers interacting with the provider. By measuring the success rate of the service provider to handle traffic within latency constraints, we will be able to analyze and improve the algorithms.

3.1 Anticipated Approach

The service provider that we will be building to test the algorithms will consist of two major components: an orchestration layer and the hardware that provides computing and networking resources. The orchestration layer consists of a state manager, request handler, algorithm solver, and analytics monitor. The hardware consists of high-volume servers which host virtual machines and network switches which allow the machines to send and receive data.

The state manager will keep track of the computing fleet in terms of the software configuration and activity on each computer. When changes need to be made to the fleet, the state manager will put them into effect by creating, modifying or destroying virtual computers.

The request handler will connect to the outside network and handle requests for services that specify functionality and performance requirements of the services. Depending on the service providerâĂŹs capacity and usage, requests for services can either be accepted or rejected.

The algorithm solver will solve for resource management strategies given the customer requirements and the state of the fleet. The algorithm used by the solver can be switched out to evaluate different resource management algorithms across separate trials. The algorithm solver communicates with the state manager to enact the strategies it produces.

The analytics component will track of how the system is performing over time and if it is meeting the requirements imposed by the customers. Once a trial has finished, the analytics component will quantitatively evaluate the resource management algorithms for review by the researchers.

The hardware will be provided by Professor PhanâĂŹs research group as four 32-core computers and network switches to connect them.

Trials will start with orchestration layer running and only the base hypervisor installed on the computing hardware. Simulated customers will begin to make requests with information about the accepted services, the acceptable latency (e.g. 20 ms), and the amount of data they will send (e.g. 100 mb at 1 mb / second). The request handler will accept or reject the requests. Once a request has been accepted, the customer will begin sending data and performance analytics will be collected by the service provider. When a trial is over, a report of the algorithmâĂŹs performance over that

trial will be generated.

3.2 Technical Challenges

The nature of Network Function Virtualization requires the deliberate integration of complex hardware and software. This design complicates latency constraint handling because CPU resources and network resources must be managed simultaneously. A major design challenge in lieu of meeting real-time end-to-end latency guarantees is optimizing and measuring both processor- and network-sourced latencies. In order to achieve minimal end-to-end latency, our design will need to a) minimize the overhead of critical functions, and b) keep non-critical functions off of the critical path.

A related design challenge is achieving repeatable test results so that algorithms may be analyzed and compared to one another. In order to accomplish this, the system must also provide consistent latency. Our design also presents additional challenges including maintaining overall system reliability and recovering from failures in either hardware (such as internal network switches) or software (such as virtual machine) components.

3.3 Evaluation Criteria

We consider several evaluation metrics. First, we will use output from the analytics component of the platform to understand and validate the performance of the real-time scheduling algorithm. Specifically, we compare end-to-end latency produced from the chain of services scheduled in multicore computers against the algorithmâĂŹs predicted latency. We plan to set an appropriate significance level in order to set a benchmark for acceptable latency performance. Since the algorithm schedules services in real-time based on the serviceâĂŹs functional requirements (in latency), we will validate how the algorithm performs given a wide range of functional requirements.

Second, when building this platform, it is important to consider how much overhead our system adds to the providing services. If our systemâĂŹs design plays a significant factor in increasing end-to-end latency, then the software and design of our platform must be optimized to minimize unnecessary and significant overhead. In the next few weeks of designing our platform, we plan to define an acceptable measure of overhead. One obvious constraint to this measure is that the added overhead should not increase the end-to-end latency of the services to more than the latency requirement imposed by clients.

Lastly, we evaluate our platform based on its scalability. As mentioned above, we plan on building this testing platform on 4 multi-core computers that will simulate servers. When talking about the scalability of this system, we need to ensure that the system is able to steadily and efficiently handle large incoming traffic (substantial packets/second) as well as deal with strict functional requirements on the system.

4. RESEARCH TIMELINE

As there is already a research group working on this subject, we will spend the first few weeks understanding the progress that has already been made by the group. Since this project involves a good deal of development, we will also spend this time on determining a set of frameworks and tools for the project. Since tools such as RT-Xen have already been used by the research group, it will be up to our senior design group to gain familiarity of these tools. This background work should be mostly completed by Thanksgiving break.

By Winter break, we will specify all the components and begin developing and testing each. We will also begin connecting components in an environment such as AWS EC2. Although performance metrics are unreliable as we are not in full control of the environment, this will allow us to make sure things are working as smoothly as possible.

Early in the spring semester, we will begin installing the components on the physical machines and begin collecting preliminary benchmarks. We will collect our first metrics by February and identify bottlenecks and architectural issues.

The rest of the year will be spent iterating on the system and performing trials with different resource management algorithms and analyzing the results. By this point we should have enough understanding of the algorithms that we can propose improvements to resource management.

As time permits, we will build an analytics dashboard for monitoring performance in real time for further insight, and contribute back to the open source tools that we used during our research.

APPENDIX

A. OTHER SPECIFICS

Your proposal need not have appendices like this section and the next but we still have info to share:

1. PROPOSAL LENGTH: We require that your proposal be 4–5 pages in length, bibliography included. Be careful, LaTeX and our style-file in particular are *extremely* space efficient. An 9-page MS-Word document could easily become a 5-page LaTeX one.
2. PLAGARISM: **DO NOT** plagarize. If you are caught, you will fail the class (*i.e.*, not graduate), or worse.

B. LaTeX EXAMPLES

At this point, the proposal specification is complete. From here on out, we are just going to show off some commonly used LaTeX technique. Be sure to look at the 'code behind' and see Tab. 1, Eqn. 1 and Fig. 1 for the output! Keep in mind that the appendix is usually not a good place for your figures. Place them where you need them and remember to refer to them in the body of your text; otherwise, the reader will keep reading and will miss them!

$$M(p) = \int_0^\infty (1 + \alpha x)^{-\gamma} x^{p-1} dx \qquad (1)$$

User Type	Cleanup%	Honesty%
Good	90-100%	100%
Purely Malicous	0-10%	0%
Malicious Provider	0-10%	100%
Feedback Malicous	90-100%	0%
Disguised Malicous	50-100%	50-100%
Sybil Attacker	0-10%	Irrelevant

Table 1: Example Table

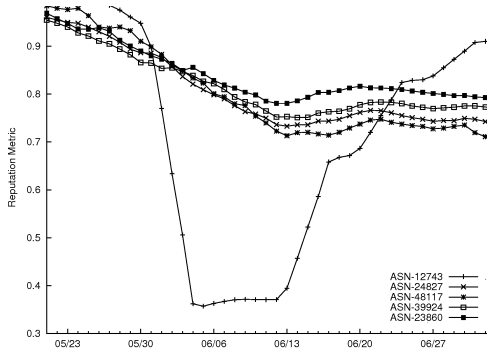

Figure 1: Example Figure/Graph