Network Research Workshop
Proceedings of the Asia-Pacific Advanced Network 2014 v. 38, p. 12-18.
http://dx.doi.org/10.7125/APAN.38.2 ISSN 2227-3026

Prototyping Media Distribution Experiments over OF@TEIN SDN-enabled Testbed

Aris Cahyadi Risdianto [1] and JongWon Kim [1,*]

1 School of Information and Communications, Gwangju Institute of Science & Technology, Gwangju, Korea

E-Mails: {aris,jongwon}@nm.gist.ac.kr

* Tel.: +82-62-715-2219

Abstract: The lifecycle of service realization experiment is composed of multiple stages, where tasks and responsibilities are well-defined between users and operators. In this paper, we prototype media distribution experiments over an OF@TEIN SDN (Software-Defined Networking)-enabled testbed while paying attention to the automated resource provisioning and experiment execution.

Keywords: Software-defined networking; lifecycle experiments; automated resource provisioning; experimental testbed.

1. Introduction

OF@TEIN Future Internet testbed project [2], which was launched in 2012 and aligned with GENI [1], connects 12 international sites spread over 7 countries (Korea, Indonesia, Malaysia, Thailand, Vietnam, Philippines, and Pakistan) as illustrated in Fig. 1. OF@TEIN SmartX Racks are deployed to simultaneously support computing and networking resources. They are coordinated by two separate planes: One for control and management and the other for datapath forwarding. Also, from early 2014, we began to simplify the design of SmartX Racks by minimizing the number of nodes for each rack and by logically connecting between computing and networking resources. For this, currently all SmartX Racks are upgraded through Chef DevOps (Developments and Operations) automation software, after small human-based wiring changes, by configuring remote power management and installation/configuration modules.

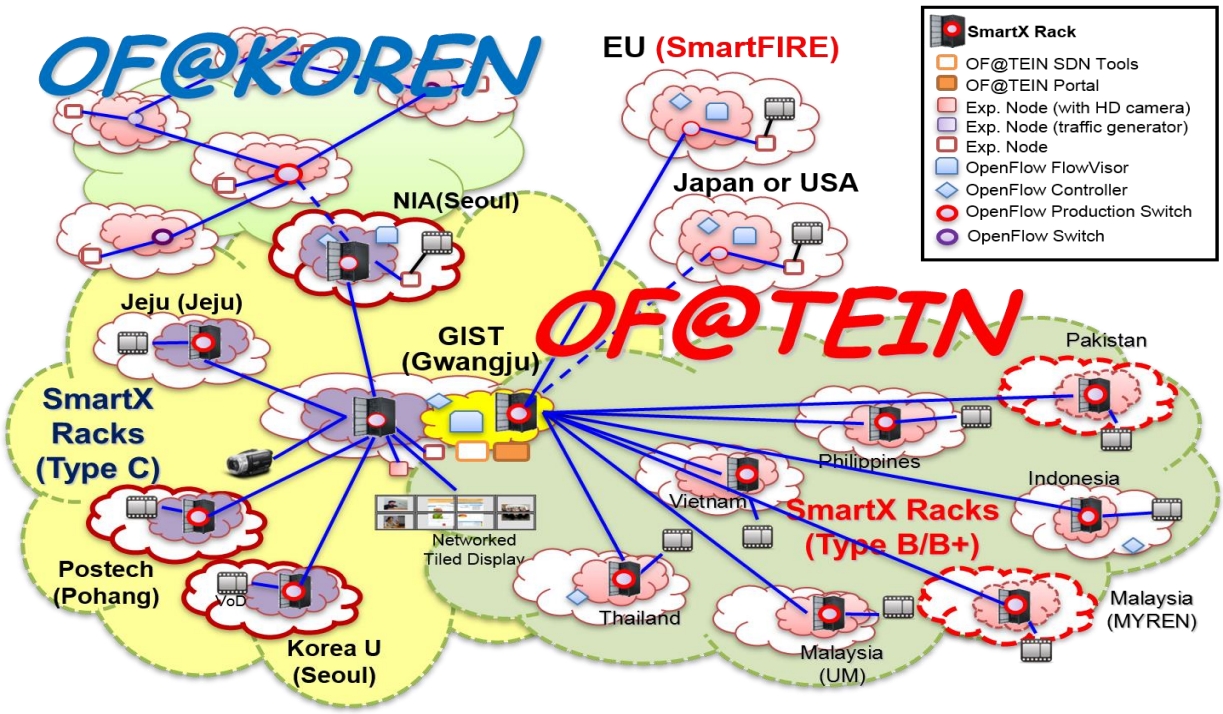

Figure 1. OF@TEIN testbed infrastructure (2012~2014).

Based on the systematic requirements for whole experiment stages (designing, provisioning, executing, monitoring, and finishing), we define lifecycle experiments to help the operators to provide the resources and instruments for user experiments and to help the users to understand the different stages of repeatable experiments. Specifically for OF@TEIN, we are required support for OF@TEIN operators to monitor, check, and recover the available resources before and during experiments. We are also required to support the users to play with predefined or others tools/scripts for their own experiments. This paper tries to extend our previous 'workflow-based control of experiment' work [3], by implementing the automated resource provisioning (i.e., installation/configuration of utilized resources) and automating the experiment execution via easy-to-use scripting. As a result, we can provide fast and efficient realization of lifecycle experiments over OF@TEIN.

2. OF@TEIN Lifecycle Experiments for Media Distribution

2.1. Lifecycle Experiments: Design and Workflow

Based on the example from GENI experiment lifecycle [4], the specific design of OF@TEIN lifecycle experiment is proposed with some modifications by dividing resource provisioning and

experiment execution stages and by adding detailed tasks with specific workflow for each stage. Fig. 2 shows that the first stage of lifecycle experiment is the design stage to prepare an experiment scenario and define required tasks before executing the experiment. The next stage will be the provisioning stage where the operators need to provide experiment resources for the users. Experiment execution stage is then defined for the users to fully control the experiment by using simple tools/scripts, which is followed by the finish stage to release allocated resources.

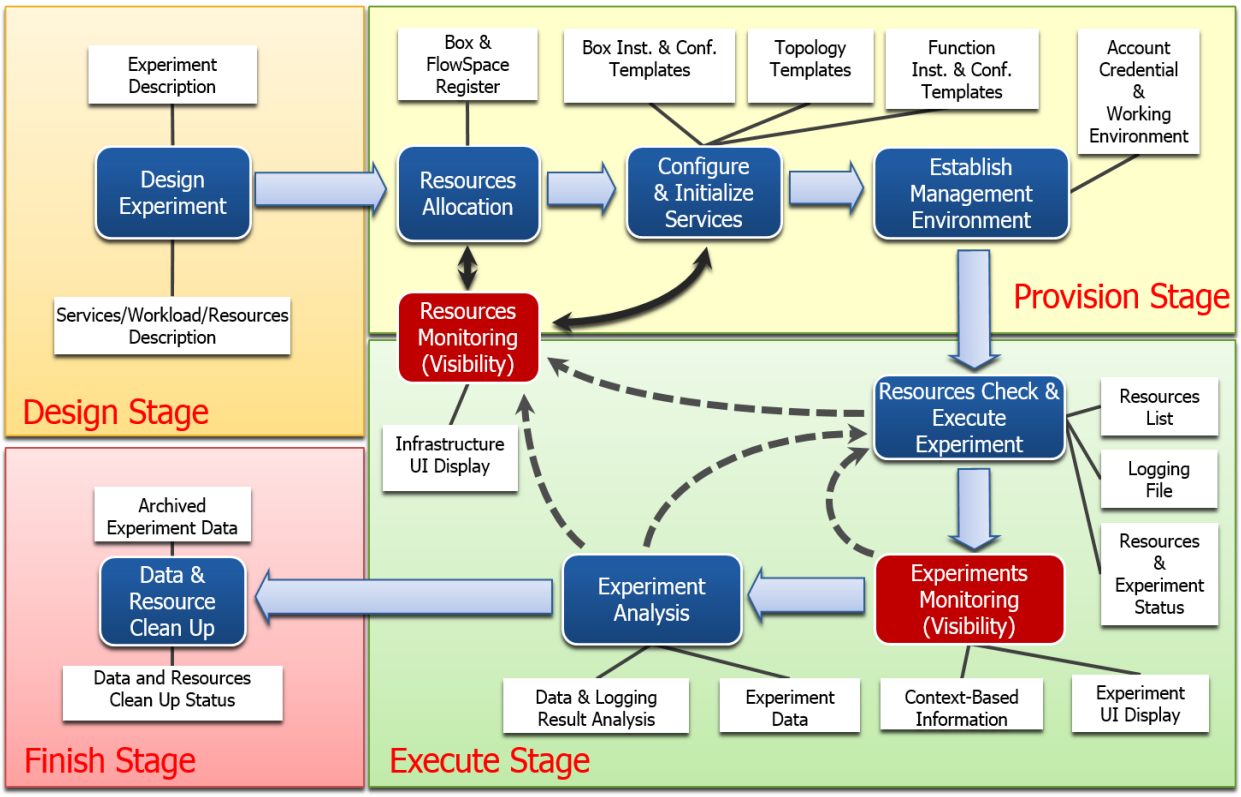

Figure 2. OF@TEIN preliminary design of lifecycle experiments.

This lifecycle experiment design includes privilege and responsibility distinction between the operators and the users. Almost all stages involve both the operators and the users. However, they cover different tasks and thus the associated workflows depend on the level of tasks (as well as privileges). That is, both users and operators should build 'DevOps' relationship, where the users are responsible to rapidly develop their services on the top of testbed infrastructure while the operators are efficiently managing the testbed infrastructure.

2.2. Typical Media Distribution Experiments

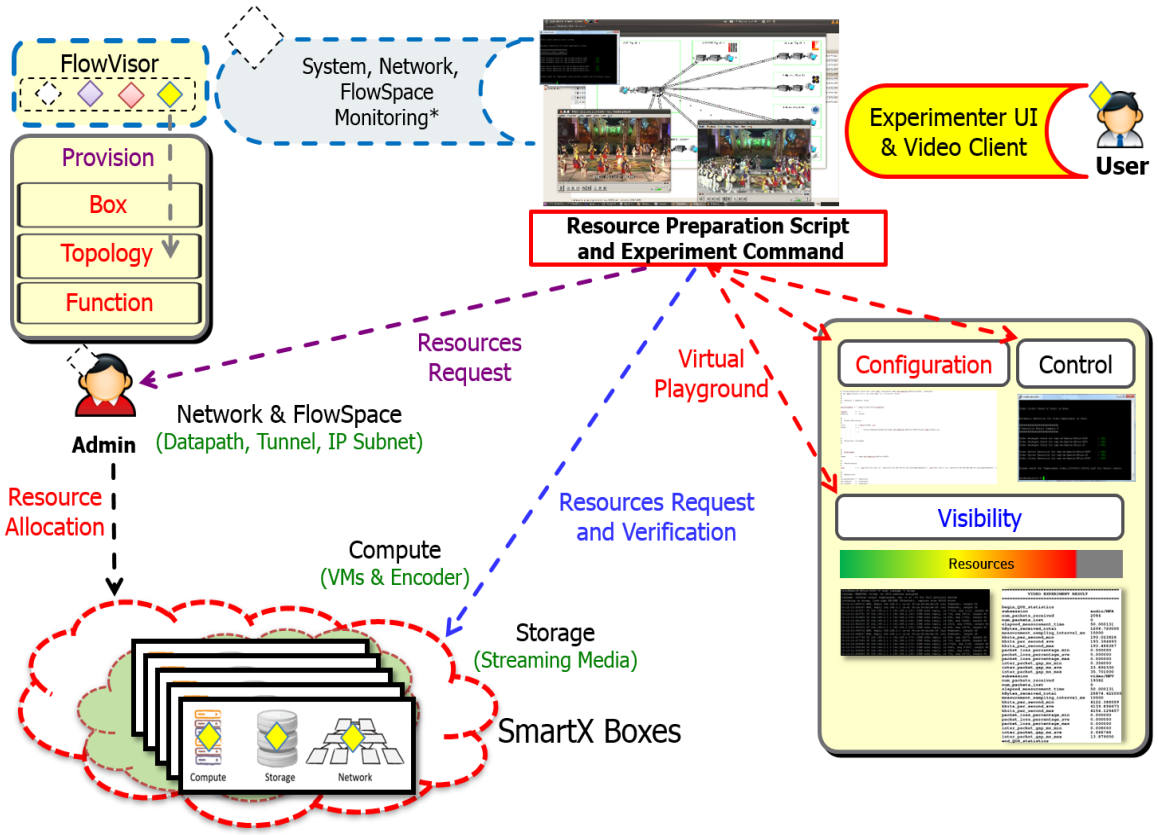

Figure 3. OF@TEIN media distribution experiment.

Similar to CDN (Content Delivery Network) services that enable content distribution to multiple users from a content distribution server [5], smoothly collecting content streams from multiple origin locations is quite challenging. A content collection server needs to merge multiple media streams and perform periodic quality checking for content adaptation. In this paper, by utilizing OF@TEIN lifecycle experiments as shown on Fig. 3, we show that this kind of media-distribution experiment can be repeatedly performed to identify the best approach.

3. Running Designed Experiments and Fixing Failures

3.1. Automated Resource Provisioning with DevOps Support Tools

As mentioned above, in order to have faster execution time and provide automatic execution, the resource provisioning uses the DevOps combination of Chef [6] installation/configuration and script-driven experiment execution [4]. Chef is effective in repeating the same installation/configuration based on Ruby[7]-written recipes from cookbooks, which can transform SmartX Rack nodes into ready-to-be-utilized resources. Like this, the above

combination supports flexibility and efficiency for lifecycle experiments by providing automatic resource provisioning and recovery.

3.2. Preparing for Media Distribution Experiments with Inter-connected Functions

With the provisioned and configured SmartX Rack resources, we now prepare the required functions and inter-connect them to construct the desired service chaining. For example, this task is handled by the experiment preparation script that first checks IP subnet and FlowSpace (tied with VLAN id for each user) resource allocation, networking resource (OpenFlow switches and tunnels) registration to the FlowVisor [8], and computing resource (virtual machines) creation. The IP subnets, as the parameters for user experiments, are allocated by the operators. Then, the experiment execution script will start ping tests among all virtual machine pairs, which can verify the resource availability. We finally create an output that details about automated resource checking. Similar experiment visibility is also provided via a SDN experimenter UI, Java-based GUI developed in [9], which automatically shows the allocated resources and active traffic flows.

3.3. Running Targeted Experiments with Performance Adaptation

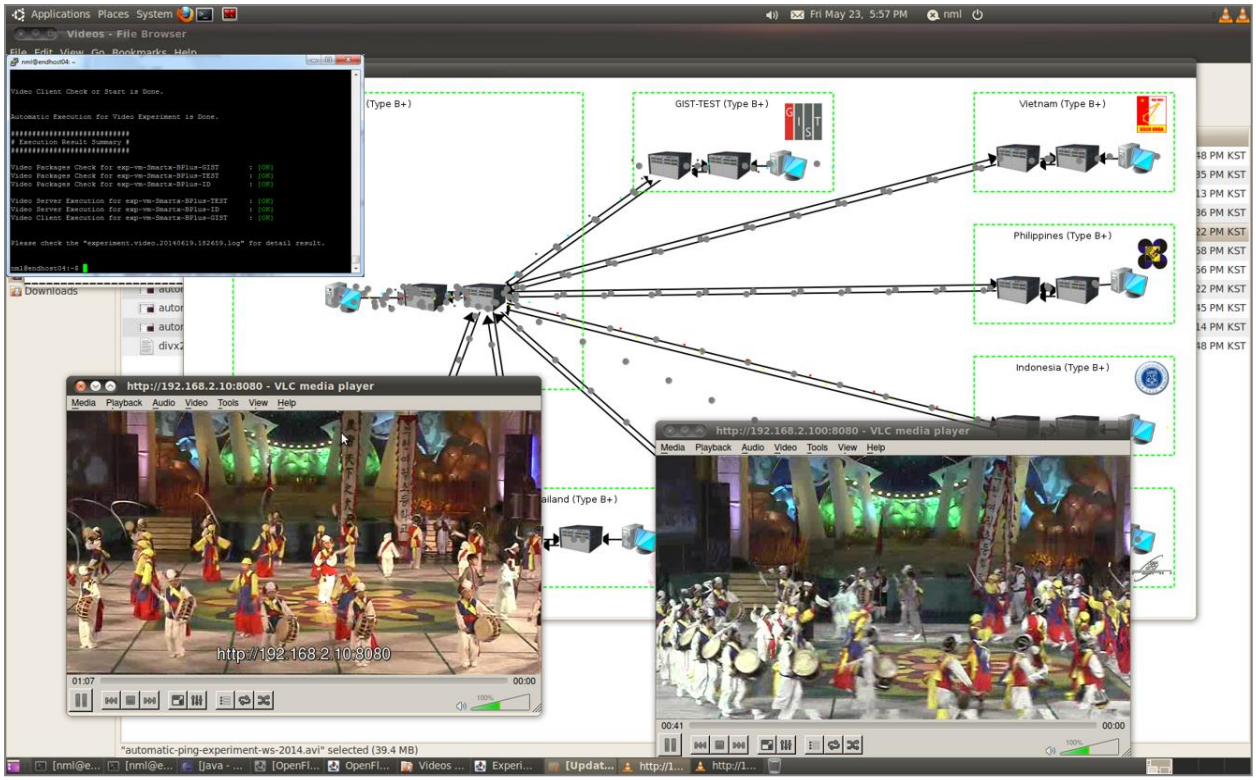

Figure 6. Media distribution experiment visualized with SDN Experimenter UI.

Next, the media distribution experiment is executed to verify the media distribution of HTTP-based streaming from two different origins. At both sides, specific VLC (VideoLan Client)-enabled virtual machines are employed to support the required media streaming capabilities and one another side played both streams into GUI terminal, as shown in Fig. 6. Note that, with VLC, we can send and receive video streams with or without GUI support. Also note that, even though this sample experiment can measure video streaming performance, the visibility-tied performance adaptation for video streaming is not yet covered.

3.4. Fixing Failures with Automated Resource Recovery

Another example lifecycle experiment shows how the operators can recover one of required resources in response to the user alarm report. To simulate the failure scenario, one node in a chosen SmartX Rack is intentionally broken. The operator then receives the alarm report from the user. In response to the report, the operator executes automated resource checking and recovery with the Chef-based installation/configuration [4].

4. Conclusions

To verify easy and fast lifecycle experiments over the SDN-enabled testbed, several automated provisioning and execution for OF@TEIN media distribution experiments is performed. In future, by fully utilizing DevOps-based tools, we are planning to continue our lifecycle experiments to refine the automation of selected key tasks of all stages for lifecycle experiments.

Acknowledgements

This work was supported by one of KOREN projects of National Information Society Agency (13-951-00-001). Also, this work was supported in part by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIP) (No. 2012R1A2A2A01014687).

References

1. M. Berman, J. S. Chase, L. Landweber, A. Nakao, M. Ott, D. Raychaudhuri, R. Ricci, I. Seskar, “GENI: a federated testbed for innovative network experiments”, *Computer Networks Special Issue on Future Internet Testbeds* - Part I **2014**, Volume 61, pp. 5-23.
2. J. Kim et al., “OF@TEIN: An OpenFlow-enabled SDN testbed over international SmartX Rack sites,” in *Proc. APAN - NRW*, Daejeon, Korea, August, 2013.

3. S.W. Han, N. Kim, J. Kim, “An experimental service composition tool for media-centric networked applications,” *Computer Networks Special Issue on Future Internet Testbeds* - Part II **2014**, Volume 63, pp. 188-204.
4. A.C. Risdianto, T. Na, J. Kim, “Running Lifecycle Experiments over SDN-enabled OF@TEIN Testbed,” in *Proc. 5th IEEE ICCE Conference*, July 2014.
5. J. Almeida, D.L. Eager, M. Ferris, M.K. Vernon, "Provisioning content distribution networks for streaming media," in *Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies Proceedings, INFOCOM **2002***, Volume 3.
6. Opscode, “Chef Documentation,” http://docs.opscode.com/.
7. Opscode, “Just enough Ruby for Chef,” http://docs.opscode.com/just_enough_ruby_for_chef.html.
8. R. Sherwood. G. Gibb, K. Yap, G. Appenzeller, M.Casado, N. McKeown, G. Parulkar, “FlowVisor: A network virtualization layer,” *Technical Report Openflow-tr-2009-1*, Stanford University, July 2009.
9. N. Kim and J. Kim, “Building NetOpen networking services over OpenFlow-based programmable networks,” *in Proc. ICOIN*, Jan. 2011.