CLICK-UP: Towards Software Upgrades of Click-driven Stateful Network Elements

Junxiao Wang, Yuchen Huang, Heng Qi, Keqiu Li*
Dalian University of Technology

CCS CONCEPTS

• **Networks** → **Programmable networks**;

KEYWORDS

click, software upgrades, stateful network elements

ACM Reference Format:
Junxiao Wang, Yuchen Huang, Heng Qi, Keqiu Li. 2018. CLICK-UP: Towards Software Upgrades of Click-driven Stateful Network Elements. In *Proceedings of SIGCOMM Posters and Demos '18*. ACM, New York, NY, USA, 3 pages. https://doi.org/10.475/123_4

1 INTRODUCTION

In the productive scenarios of virtualized network function (vnf), active network elements are always accompanied by the software upgrade, due to the off-the-shelf ones cannot provide satisfactory functionalities, some high-level reasons e.g. business upgrading. Thus original network elements have to be re-formulated and adapt to their future work. First, we should say thanks to the evolution brought by vnf. Its appearance actually gives us a new chance to replace traditional hardware-based upgrades with software style, increasing the cost-efficiency as well the technology iterative speed.

Second, the Click-driven platform [3] has been one of the most popular vnf platforms that allow flexible composition of packet-processing functionalities. It has been used in a number of application prototypes such as a router for the future Internet architecture, redundant traffic elimination systems, a scalable middlebox platform, and a cluster-based high performance software router, just to name a few.

The key strength of Click is its inherent extensibility for functional components: a new feature can be easily implemented by module integration. While Click's flexible design has satisfied many demands for rapid prototyping, its internal architecture has not caught up with the potential software upgrade. Traditional Click-driven upgrades for network elements have some significant ***drawbacks*** as following:

D1: Integrating new modules with upgraded network elements is a time-consuming processing. During this processing, however, the packet-processing functionalities are out-of-work. This will bring several insufficiencies including the inability to elastically scale out network functions on demand and to quickly recover from downtime.

D2: With no mechanism to reconstruct lost states for network elements, stateful functionalities are unable to correctly handle packets after upgrade, leading to service disruption. This may involve with states such as connection information in a stateful firewall, substring matches in an intrusion detection system, address mappings in a network address translator, or server mappings in a stateful load balancer.

D3: Development of new modules requires intimate knowledge of complex library implementation, let alone correctly deal with states. It is frustrating that the visibility of current upgrade is pretty poor and the learning curve is rather steep.

In this paper, in order to address these problems and satisfy practical requirements about stateful network element upgrades, we present CLICK-UP [2], which is, to the best of our knowledge, the first research effort towards software upgrades of Click-driven stateful network elements.

For one thing, we notice the root of ***D1*** is that software integration of upgrades didn't stick religiously to the service context. A series of functionality-neutral modules are redundantly shipped with essential modules. Thus, we argue that explicitly integrating essential modules in a service context aware manner can cut down upgrade overheads. (solution of ***D1***). For another, current Click modularity still poses severe challenges in maintaining network states during software upgrades. Instead of expecting the operators who originate network elements to manage any problem, we argue that a state synchronization scheme which enforcedly integrated into modules is essential. (solution of ***D2***). At last, as the exoskeleton of Click modularity, we argue that a lightweight runtime library of software upgrades is also necessary to be embraced for clarifying service semantics, simplifying tedious orchestration and unifying interfaces. (solution of ***D3***).

Besides the agile, stateful and tractable features on software upgrades, CLICK-UP also stands out because of its strong compatibility. The seamless instead of invasive integration style can be easily compatible with current Click-driven techniques, such as ClickOS [5], CliMB [4] and FastClick [1].

2 DESIGN OF CLICK-UP

Design Overview: As shown in Figure 1, the framework of CLICK-UP is divided into the top-down three layers.

*Junxiao Wang, Yuchen Huang, Heng Qi, and Keqiu Li are with the School of Computer Science and Technology, Dalian University of Technology. This work is supported by the National Key R&D Program of China (2016YFB1000205), the State Key Program of National Natural Science of China (No.61432002) and the National Science Foundation of China (No.61772112, No.61672379, No.U1701263, No.61702365, No.61425002, No.61751203). Heng Qi is the corresponding author.

SIGCOMM Posters and Demos '18, Budapest, Hungary

ACM ISBN 123-4567-24-567/08/06...$15.00
https://doi.org/10.475/123_4

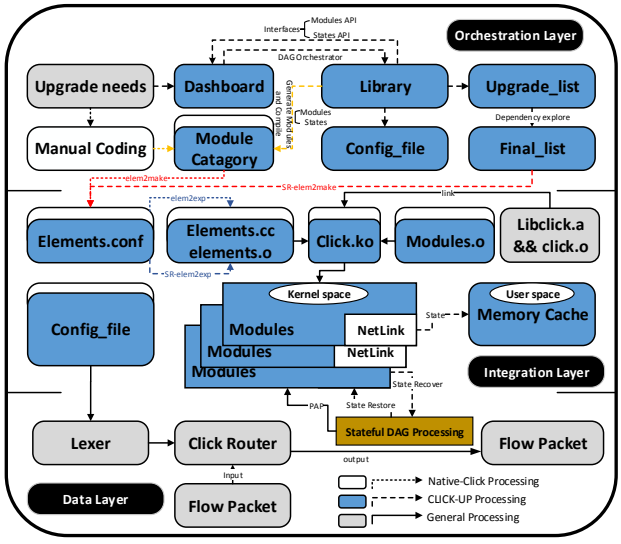

Figure 1: CLICK-UP framework overview.

In contrast with native processing, operators directly interact with the DAG orchestrator and formulate their required atom-based pipeline in a view of runtime library, getting ride of previous inefficient manual manners. During the alternative processing, all the dependency of essential modules are fully explored according to the determined service context in DAG pipeline. This can help in cutting down unnecessary overheads of module integration, and speed up the recovery from downtime as far as possible.

The reconstruction scheme for lost states is implemented by periodic synchronization between modules and persistent storage, which respectively locates at kernel space and user space. With recovery bootstrap integrated inside all the modules, the states can thus be recaptured after upgrade.

Workflow: The procedures for CLICK-UP to handle a software upgrade of Click-driven stateful network elements are shown as arrow direction in Figure 1: (1) the dashboard exposes DAG orchestrator to operators, allowing operators to define their upgrade needs as a DAG. The DAG is based on well-known semantics and consists of a series of pipeline processing related atom functionalities as well their required service states. (2) the DAG should be parsed to a set of Click modules (called elements), and its new state collection is integrated into corresponding modules. All the modules have a state synchronization mechanism and a state reconstruction bootstrap. (3) The modules are complied, built into kernel space, and the persistent storage in user space is initialized with new version number, meanwhile sending back former version related states. (4) The new configuration is created and upgraded network element is reboot from downtime with former service states fulfilled by a recovery bootstrap.

Atom-based Orchestration: As shown in Figure 2, the DAG orchestration of CLICK-UP is based on a series of core functionalities called atom functionalities, e.g. packet parsing, payload modification, and the like, each of which with well-known semantics. Operators can thus represent their upgrade

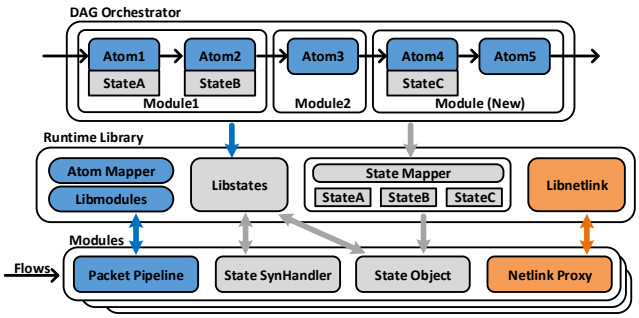

Figure 2: Atom-based upgrade orchestration.

needs as a DAG with stateful or stateless atoms linked. Via exploring all the dependency of essential modules, runtime library will parse the atom-based graph to a module-based graph that has identical service contexts and pipeline functionalities. This realizes an unified interface for operators, at the same time provides the compatibility between CLICK-UP and other Click-driven techniques that based on module-style graph. Moreover, since the orchestration sticks religiously to the service context, thus a series of functionality-neutral modules won't shipped with essential modules any more, significantly reducing the integration time.

Runtime Library: As shown in Figure 2, the runtime library of CLICK-UP consists of a set of interfaces exposed to the orchestration. They specify available atom functionalities as well the mapping between atoms and module category. These interfaces also provide a state management toolset to deal with the state synchronization and the reconstruction bootstrap. Through top-down integration, all the modules of upgraded network elements can thus recover from downtime in an agile manner and take back their service states correctly. Developers can also enlarge current library stack in scalability by implementing new interfaces for new atoms or states, in order to handle the cases that atom-based graph doesn't fit comfortably within upgrade needs' range.

3 DEMO

In order to further demonstrate the effectiveness and convenience of CLICK-UP[1], we offer a software upgrade walkthrough on a firewall network element. We hope to demonstrate the agility of module integration by upgrading a certain number of new modules during this demo, and checking its downtime compared to the native manners. We also plan to demonstrate the state persistency by implementing tcp session states in the demo, and observing the negative influence on state loss compared to our state reconstruction case.

During demonstration, we are going to demonstrate more details as described above by using a laptop (with dashboard and http benchmark on it) and a raspberry pi (with whitelist firewall and CLICK-UP server on it). On site, we will showcase at length how to apply CLICK-UP to the software upgrades of Click-driven stateful network elements.

[1]Source codes are now available at https://github.com/click-up

REFERENCES

[1] Tom Barbette, Cyril Soldani, and Laurent Mathy. 2015. Fast Userspace Packet Processing. In *Proc. of ACM/IEEE ANCS*.
[2] CLICK-UP. 2018. Project Website. (2018). https://click-up.github.io.
[3] Eddie Kohler, Robert Morris, Benjie Chen, John Jannotti, and M. Frans Kaashoek. 2000. The click modular router. *ACM Transactions on Computer Systems* 18, 3 (2000), 263–297.
[4] Rafael Laufer, Massimo Gallo, Diego Perino, and Anandatirtha Nandugudi. 2016. CliMB: Enabling Network Function Composition with Click Middleboxes. In *Proc. of ACM HotMiddlebox*.
[5] Joao Martins, Mohamed Ahmed, Costin Raiciu, Vladimir Olteanu, Michio Honda, Roberto Bifulco, and Felipe Huici. 2014. ClickOS and the art of network function virtualization. In *Proc. of USENIX NSDI*.