In-Network Processing, User-Level Stacks and the Future of Internet Evolution

Felipe Huici*, Michio Honda*, Costin Raiciu†
NEC Europe Ltd.*, University of Bucharest†
{felipe.huici, michio.honda}@neclab.eu
costin.raiciu@cs.pub.ro

1. INTRODUCTION

In past years, it has become increasingly evident that the venerable end-to-end model often taught in networking courses has more to do with Internet lore than with the reality of the network today. Over time, operators have deployed a vast array of middleboxes to enhance the capabilities of the network, ranging from security (firewalls, IDSes, traffic scrubbers), traffic shaping (rate limiters, load balancers), dealing with address space exhaustion (NATs) or improving performance (traffic accelerators, caches, proxies), to name a few. Middleboxes are ubiquitous: a third of access networks show symptoms of stateful middlebox processing [4] and in enterprise networks there are as many middleboxes deployed as routers and switches [6].

Middleboxes are here to stay, and their usefulness, at least to operators, is beyond question. However, their existence restricts, in an often opaque way, what is considered "correct" traffic: packets that do not use common protocols or that use unusual header field values risk being modified or dropped altogether. This inevitably forces applications to use a few (sometimes inadequate) protocols such as HTTP or worse, HTTPS, to ensure that traffic makes it to their destination.

It might be tempting to single out middleboxes as the root of the Internet's ossification problem, but their existence is not a fundamental show stopper. Rather, the issue lies in the fact that their presence in a path is often unknown to the end points, that some of them may modify or filter protocols in ways unexpected to end points, and that there is no mechanism to explicitly address them in order to negotiate and resolve the tussle between what the operator needs in order to ensure the correct functioning of its network and what end users would expect in terms of the service they would like to experience.

Middleboxes are not the only factor contributing to the Internet's ossification. A recent study [2] has shown that as many as 86% of Internet paths still allow TCP extensions despite the existence of a large number of middleboxes. Indeed, over the past years a respectable number of new protocols have surfaced (e.g., SCTP, DCCP, MPTCP, improved versions of TCP), many of which have become part of mainstream OSes and distributions. However, making into an OS does not imply the immediate availability of a protocol: OS upgrades can take years to trickle down to end user devices, and even when they do, they are sometimes not automatically enabled [3]. In short, it is rather difficult to deploy novel network stacks and their corresponding protocols on end user devices, compounding the previously mentioned tendency towards using a select few protocols.

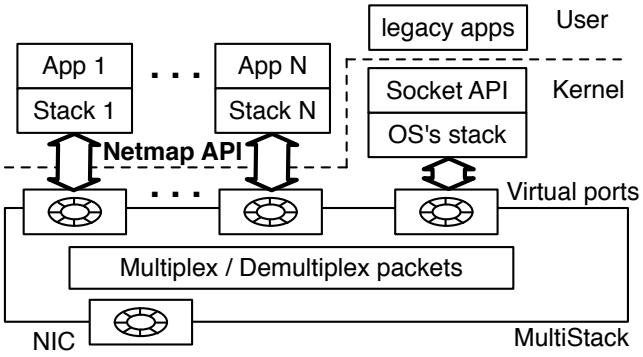

Figure 1: MultiStack software architecture.

How do we get ourselves out of this mess and *incrementally* into a world that strikes a balance between the operational realities of end-to-end paths in today's Internet and the Internet's intended, original end-to-end model? Fixing the Internet's ossification is a tall order, but incremental progress can be made by focusing on specific issues and their potential solutions.

In this position paper we take such an approach, and discuss two components we believe to be crucial to the future evolution of the Internet. First, we argue for user-level network stacks as a means to allow quicker and easier deployment of novel protocols. Second, we introduce the concept of *in-network processing*, whereby an operator deploys a set of (virtualized) platforms able to run their middlebox software as well as that of its customers and third-parties. Such platforms can be addressed explicitly, and allow the operators and end users to reason about the interaction between traffic, the middleboxes and network policies.

In the rest of this short paper we outline the main principles behind each of these two proposals and our implementation of them. We finish by addressing potential ways of going forward in order to improve the current state of affairs.

2. USER-LEVEL NETWORK STACKS

Placing network stacks in user space would have the clear advantage of not having to wait for OS upgrades for new protocols to reach the general public. But can this be achieved without sacrificing the performance and security afforded by traditional, in-kernel stacks?

In previous work [3], we presented MultiStack, operating system support for user-level network stacks (figure 1). MultiStack's implementation revolves around four design

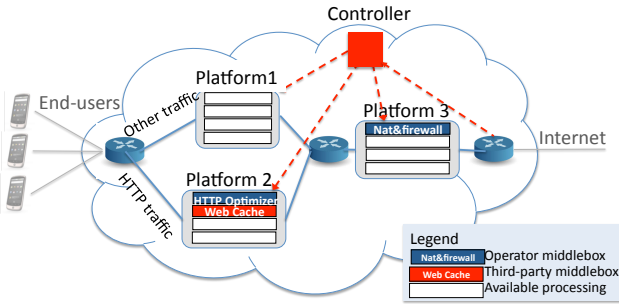

Figure 2: CHANGE Architecture: Access operators deploy processing platforms (micro-datacenters) where they run their own middleboxes and processing from third parties. A controller deployed by the operator knows the network topology, router and middlebox configurations. Client requests are deployed by the controller after they are statically checked for safety.

principles. First, MultiStack supports a large number of dedicated stacks, to allow for having per-application stacks. Second, each stack has isolated access to the NIC, preventing stacks, some of which may contain untested or beta code, from interfering with each other. Third, MultiStack provides supports namespace isolation, whereby stacks register `<src IP, dst port, protocol type>` 3-tuples which are then used to multiple incoming packets and to validate outgoing traffic for each application/stack instance. Finally, the system is able to accommodate legacy (in-kernel) stacks and application, providing an incremental deployment path. In terms of performance, our tests show that MultiStack is able to achieve 10Gb/s line rate for almost all packet sizes with a single CPU core, and line rate for minimum-sized ones when using 2 CPU cores.

Clearly, MultiStack would require changes to the kernel. However, these changes would only need to happen *once*, after which novel stacks could be deployed without requiring the long deployment cycles related with OS upgrades. Further, because it is implemented as a module, MultiStack does not need changes to the kernel itself. We believe that MultiStack shows the viability of user-level network stacks as a means to re-invigorate network stack development.

3. IN-NETWORK PROCESSING

The availability of a new stack on end user devices is not sufficient for its successful deployment. As discussed earlier, the ubiquitous presence of middleboxes means that packets belonging to a new protocol will often be modified or dropped, affecting the behavior or performance of the applications that make use of it.

While we cannot easily modify existing hardware-based middleboxes, there is a current trend by major operators and vendors towards turning them into software-based boxes running on inexpensive x86 servers, including turning them into virtualized instances [1]. The trend goes further, with operators like Deutsche Telekom deploying racks of x86 servers in their core and at regional POPs in order to deploy their (virtualized) middleboxes, but also with views towards offering such servers as a platform for middlebox deployment by third parties.

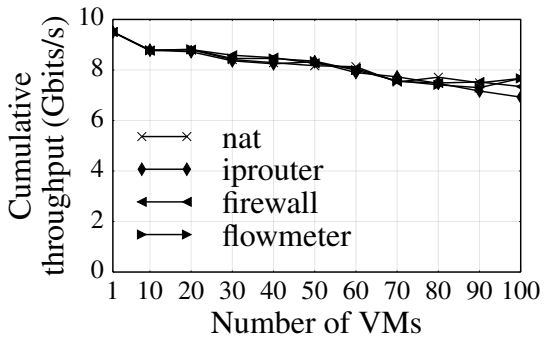

Figure 3: High throughput when running 100 middlebox VMs on a single CPU core

In a technical report [CITE], we make the case for IN-NET, an architecture to enable such *in-network processing* (figure 2). IN-NET provides (1) a set of security rules that ensure that the middleboxes are safe to run in terms of the operator's network, other middleboxes and the Internet at large; (2) an API that allows operators and users to specify and negotiate policy; (3) the use of static analysis tools to automatically check whether client processing satisfies provider policy and security requirements, and whether the client's own requirements are met; and (4) a high performance, virtualized middlebox platform.

To show the platform's feasibility, we implemented it using ClickOS [5]. As an example of the sort of performance that can be expected from a IN-NET platform, we ran a test with up to 100 VMs configured as different kinds of middleboxes. As shown in figure 3, a IN-NET platform can achieve rates of up to about 8 Gb/s when running 100 VMs (for more extensive test results, we refer the reader to [CITE]).

While IN-NET is no more than a research prototype at this stage, we claim that an architecture of its sort would foster a more open environment for middlebox deployment, and would allow for negotiation between what end clients (or their stacks) expect and what the network can or is willing to provide, leading to win-win situations.

4. CONCLUSION

In this short paper we have discussed the concept of user-level network stacks and in-network processing, along with prototypical implementations of each. The former lowers the barrier for deploying novel protocols in end user devices, while the latter does so for the middleboxes along those protocols' paths. Both are incrementally deployable, and we believe that while they are at this stage only prototypes, they represent a first step towards the future evolution of protocols on the Internet.

5. REFERENCES

[1] ETSI Portal. Network Functions Virtualisation: An Introduction, Benefits, Enablers, Challenges and Call for Action . `http://portal.etsi.org/NFV/NFV_White_Paper.pdf`, October 2012.

[2] M. Honda, Y. Nishida, C. Raiciu, A. Greenhalgh, M. Handley, and H. Tokuda. Is it Still Possible to Extend TCP? In *Proc. ACM IMC*, pages 181–192, 2011.

[3] Michio Honda, Felipe Huici, Costin Raiciu, Joao Araujo, and Luigi Rizzo. Rekindling network protocol innovation with user-level stacks. *SIGCOMM Comput. Commun. Rev.*, 44(2):52–58, April 2014.

[4] Michio Honda, Yoshifumi Nishida, Costin Raiciu, Adam Greenhalgh, Mark Handley, and Hideyuki Tokuda. Is it still possible to extend tcp? In *Proc. ACM IMC*, 2011.

[5] Joao Martins, Mohamed Ahmed, Costin Raiciu, and Felipe Huici. Enabling fast, dynamic network processing with clickos. In *Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking*, HotSDN '13, pages 67–72, New York, NY, USA, 2013. ACM.

[6] J. Sherry, S. Hasan, C. Scott, A. Krishnamurthy, S Ratsanamy, and V Sekarl. Making middleboxes someone else's problem: Network processing as a cloud service. In *Proc. ACM SIGCOMM*, 2012.