Inserting a Network Virtualization Layer between Layers 2 and 3

Benjamin Reagan McLemore V
Ira A. Fulton Schools of Engineering
Arizona State University
Tempe, Arizona, USA
brmclemo@asu.edu

I. Introduction and Motivation

For my project, I have chosen to build a virtualization layer on top L2 that would give a single software switch the ability to simultaneously support several independent and isolated L3 networks with potentially different architectures and protocols. The purpose of this innovation would be to allow new internet architectures to be set up and tested on already existing production infrastructure without interfering with actively running networks and to allow different internet architectures with different strengths and weaknesses to run over the same hardware. In the modern age of streaming and big data, the present architecture of the internet, defined by the TCP/IP stack and host-to-host connections, is beginning to show its age. As such, many are experimenting with novel alternatives, but the rigidity of existing infrastructure means that these experiments can only be done in isolated testbeds with specialized infrastructure and capabilities. With this virtualization, the public internet would be able to host any number of architectures, in addition to the TCP/IP stack, which would allow for innovations to be made without disruption or costly customized environments.

II. Related Work

Similar ideas to the one being proposed have been tested in both *Revitalizing the Public Internet by Making it Extensible* [1] and *FlowVisor: A Network Virtualization Layer* [2], though with slightly different goals. The former attempts to build on top of L3, i.e. within the current internet architecture, a means of virtualizing independent services of the internet so that many high-level features currently provided by the application layer such as encryption and caching can be added directly to the internet with minimal disruption. The latter attempts to virtualize networks by limiting the access of independent OpenFlow controllers to certain isolated segments of packet header values, for the primary purpose of isolation. What both of these projects have in common with mine and each other is their desire to make networks more flexible and powerful by moving complex, higher-level network features down the protocol stack to make them accessible at lower levels of the network. Where these projects differ from mine is that I am not attempting to change the way current networks operate, rather I am attempting to build an abstraction layer that would allow many networks to run on the same devices.

In addition, *P4: Programming Protocol-Independent Packet Processors* [3] describes a mechanism that allows for commodity switches to be programmed to support custom L3 protocols. Since my project will attempt to insert a new protocol layer between L2 and L3, P4 will likely be necessary to implement it, but even if not, it provides a roadmap for how switches can be programmatically instructed to handle different protocols, which is a feature a full implementation of my project would need to include.

Another paper, *Named Data Networking* [4], describes another internet architecture that could be virtualized alongside the traditional IP stack, and discusses some of the reasons new internet architectures are being sought out.

III. Experimental Set-up

For my experiment, I plan to first implement the protocol for software-defined switches, probably using P4. Specifically, the protocol will work by putting a new header, called the ”protocol header”, at the beginning of all outgoing L3 packets which will indicate which protocol or virtual network the packet belongs to. When each packet is received by its next switch, the switch will first look at this header to determine which protocol or virtual network the packet belongs and then send it to the appropriate forwarding mechanism. For this project, the likely P4 implementation will probably be monolithic and include built in implementation of IP, though in the future implementations may include a modular system which would allow protocols to be easily created, installed, and shared without having to write P4 code.

Because of the complexity and scale of the project, it will be set up in three phases, each with a more complex network that adds a new piece of the architecture. The first phase will involve creating a ”translation node”, which will sit between a host and the virtual network to add the protocol header to outgoing packets and remove the protocol header from incoming packets, so the host doesn't know that the network is virtualized. The test for this phase will involve creating a simple network with two hosts, each connected to a translation node for IP and the translation nodes connected to each other. A simple http client-server app will then be run between the hosts, to verify that the translation nodes function properly.

The second phase will involve the creation of a ”core node”, which will handle the core functions of the virtualized network, such as distinguishing between different protocols, packet forwarding, and protocol specific functions such as caching. This phase will have the same architecture as the first, except that a core node supporting IP will be placed in between the two translation nodes, so that all traffic will need to cross it. The experiment will also be identical to the first, to verify that the core node forwards traffic as intended

The final phase won't add any new components. Instead, its focus will be on verifying that the core node can indeed handle multiple virtual networks at the same time. To do this, two virtual IP networks will be created, each with a client and server node to run the application used in the previous experiments. Each client and server node will be attached to a translation node, and each translation node will be attached to a single core node. The translation nodes will virtualize/devirtualize traffic to/from its respective virtual network and the core node will forward the traffic between them, keeping packets within their own virtual network. As with before, the experiment will simply verify that the core node successfully forwards virtual network traffic and keeps it isolated.

IV. Results

Since this project is a proof of concept, rather than a traditional factorial experiment, the results from each phase of the experiment will simply include whether or not traffic reaches its destination and whether or not the virtual networks stayed isolated. At all phases of the experiment, I do expect that both will be the case.

References

[1] H. Balakrishnan et al., "Revitalizing the Public Internet by Making it Extensible," ACM SIGCOMM Computer Communication Review (SIGCOMM-CCR), Volume 51, Issue 2, April 2021, pp. 18-24.

[2] R. Sherwood et al., "FlowVisor: A Network Virtualization Layer,"

[3] P. Bosshart et al., "P4: Programming Protocol-Independent Packet Processors," ACM SIGCOMM Computer Communication Review (SIGCOMM-CCR), Volume 44, Issue 3, July 2014, pp. 88-95.

[4] L. Zhang et al., "Named Data Networking," ACM SIGCOMM Computer Communication Review (SIGCOMM-CCR), Volume 44, Issue 3, July 2014, pp. 66-73.