yanc: Yet Another Network Controller

Matthew Monaco (student), Eric Keller
University of Colorado, Boulder
{matthew.monaco, eric.keller}@colorado.edu

With the rise of software-defined networking, there has been a great deal of focus on the design of the centralized controller (both commercially and academically). This controller serves the role of an operating system for networks [2] which provides an interface to program the entire network.

In this poster, we explore the question: Is a network operating system fundamentally that different from an operating system such that it requires a completely new (network) operating system? The current offerings, while extensible, are geared towards single, monolithic network applications more analogous to a single operating system process with optional plug-ins. We argue that instead of building custom network operating systems, we should leverage existing operating system technology in building a network operating system – effectively making Linux the network operating system.

We will present `yanc`, a software-defined networking controller in early development. The main insight in `yanc` is that the network can be managed by standard file I/O, and we do not need special applications which sit at some privileged point between the network OS and hardware. Program dependencies and ordering is configured through text files – e.g., a program can be packaged with a config file in `/usr/share/yanc/conf.d/` with a default priority and relative ordering to other applications, but can be tweaked by an admin under `/etc/yanc/conf.d/`.

The `yanc` file system is mounted at `/net`. Network state such as switches, ports, queues, flow tables, and counters are organized as files and directories. To expand on this, we give a few examples here which reflect our current thinking to illustrate the possibilities, but is sure to evolve over time. Devices are configured by e.g., writing *100* into `/net/switches/`*id*`/queues/`*id*`/size`. The datapath is configured by writing a plain-text string e.g., *prio=16 match=in:10,ethsrc=mac,vlan=4 pop-vlan out=flood* to `/.../`*switchid*`/flows`. Error numbers are returned indicating success or failure – e.g., for incorrectly overlapping another flow. Individual flows have file representations which can be altered and deleted. Finally, network events are exposed through files which represent the type of event. For example, `/.../`*switchid*`/packet_in` can be used to retrieve PACKET-IN events, which can be read and written by applications – applications can write to it to alter the view other applications see of that event (as configured by the administrator, not hard coded into the applications).

Directories represent a hierarchy of network state (or network slices [3]). The slice will have some (configurable) default form – e.g., a read-only copy of the entire network. It is then configured by writing plain-text strings to a file such as *ipnet=10.0.0.0/24* With directories, access to portions of the network state can be restricted using file permissions.

Confining the controller's north-bound API to a file system enables a rich variety of applications which can be implemented in any language supported by the compilers and interpreters installed on the system. The network is ultimately controlled by daemons, `cron` jobs, interactive utilities, and even ad-hoc scripts.

`Yanc` differs from other controllers such as NOX [2] and Floodlight [1] in that it allows for applications to take a wide variety of forms by relying heavily on existing operating system (Linux) features. This results in a north-bound API with a stable and well-understood set of semantics. With this, a `yanc` network can rely on existing software which follows the same file system semantics from `grep` and `mkdir` to distributed file systems.

In the poster we will elaborate on the design and discuss how *yanc* can be used to perform the variety of network tasks needed by network admins.

References

[1] Floodlight. http://floodlight.openflowhub.org, 2013.

[2] N. Gude, T. Koponen, J. Pettit, B. Pfaff, M. Casado, N. McKeown, and S. Shenker. Nox: towards an operating system for networks. *ACM SIGCOMM Computer Communication Review*, 38(3):105–110, 2008.

[3] R. Sherwood, G. Gibb, K. Yap, G. Appenzeller, M. Casado, N. McKeown, and G. Parulkar. Flowvisor: A network virtualization layer. *OpenFlow Switch Consortium, Tech. Rep*, 2009.