# OVS-KSelftest: A new way to test the kernel module

Aaron Conole <aconole@redhat.com>

November 8, 2022

# Introduction

## OVS has two different datapaths

- netdev
- netlink

## Netlink datapath

- decoupled from 'main' ovs development

## Netlink: less attention from the community

- Different tree, maintainers, etc.

# What is netlink datapath

## The 'kernel datapath'

- Controlled via netlink messages from userspace (get it?)

## In-tree for kernel

- developed on the netdev list
- Accepted in 2011 (mature)

## Recently removed from ovs tree

- Most development for kernel side needs to start in kernel ML anyway

# How do we interact with it?

## Primarily via the vswitchd

- There exists one utility - `ovs-dpctl` (disadvantages though)
- `ovs-appctl dpctl/...` is just a front-end via `ovs-vswitchd`

## All kernel side testing is contained in ovs tree testsuite

- `make check-kernel`

# Issues with existing approach

## Upstream changes are difficult to vet

- Kernel maintainers can't be expected to be running all of the userspace utilities
- Even individual developers can sometimes make changes without really knowing if they've broken things

## Having a test infrastructure that relies on ovs userspace insalls is somewhat error prone

- Difficult to get folks to install and run the tests
- Upstream maintainers test from lots of subsystems

# The approach

## Create a new utility that can be included in kernel tree

- Should not be coupled to the OVS userspace (minimize dependency)
- Should be able to run in an automated fashion to test changes

## Utility should be easy to extended and use

- Chose python over C because it is easier for extending
- Performance isn't as critical as correctness

# Initial version of dpctl utility accepted

## Current limitations

- Only creates / displays the DP right now
- Output format is close to the `ovs-dpctl` format

## Already included a test

- Used to trap a specific error condition related to a WARN() call
- Can also do some feature bit settings

# Initial test harness

## dpctl utility isn't the only thing

- Shell script that creates namespaces, interfaces, and datapaths
- Has hooks to save traffic and log commands, etc.
- Detects missing python libraries, and modules

## Future work

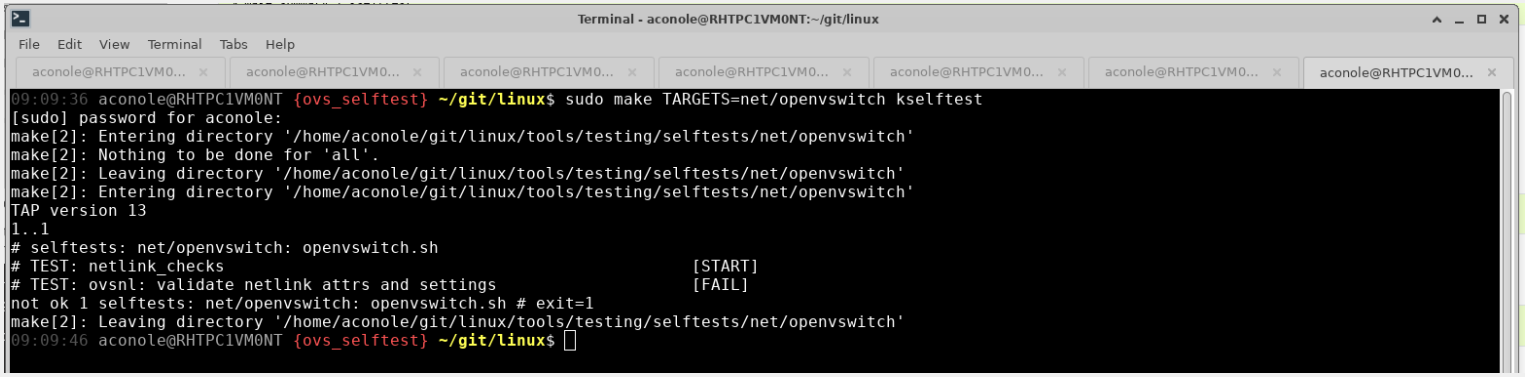
- Adding the ability to push / dump flows during testing
- More introspection would be great

# Running the suite

## Easy to do

```
$ sudo make TARGETS=net/openvswitch kselftest
```

## With a current kernel tree



# Adding some test

## simple test case

```
function test_addingflow() {
  sbx_add "test_addingflow" || return 1
  info "setting up some DPs"
  ovs_add_dp "test_addingflow" af0 || return 1
  ovs_add_netns_and_veths "test_addingflow" af0 \
                          left left0 l0 || return 1

  ovs_add_flow "test_addingflow" af0 \
               "in_port(1),eth(),eth_type(0x800)" \
               "drop" || return 1
  return 0
}
```

# Adding some test (cont"d)

**adding to the harness**

```
@@ -11,7 +11,8 @@ VERBOSE=0
 TRACING=0

 tests="
-        netlink_checks                                    ovsnl
+        netlink_checks                                    ovsnl
+        addingflow                                  ovsnl: ad

 info() {
     [ $VERBOSE = 0 ] || echo $*
```

# Future work - current patches to be submitted

## dpctl utility

- Configure interfaces to the datapath
- Add some static flows

## Test harness script

- Do some testing for various actions, including nested actions
- Testing upcalls as well

# Future work - ovs tree

## Deprecation?

- deprecate the in-tree `ovs-dpctl` utility
- We would much prefer `ovs-appctl dpctl/...` since it won't reconfigure the datapath

## Upstream

- 'pyroute2' could be extended with the internal classes from our utility
- Would make other projects able to integrate to the kernel side easier

## Additional

- At least share some of this with Adrian Moreno's tracing and monitoring work