DPDK SUMMIT
Powering the Future of Networking Software

12–13 May 2026

Develop With Confidence: Integrate the DPDK Test Suite With Your Development Workflow

Patrick Robb - DTS Maintainer

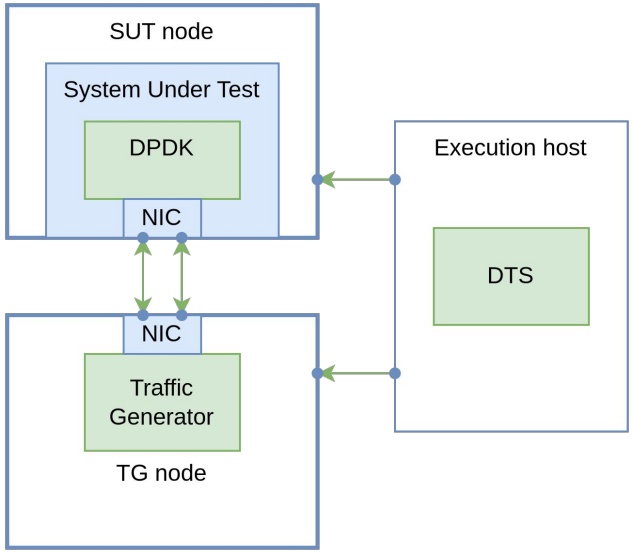
Introduction: Who am I and what is this talk about?

My name is Patrick Robb, and I am a maintainer of DTS, a testing harness and set of test suites for DPDK. I am also involved with the CI testing community, and previously was the DPDK Community Lab manager at the UNH-IOL.

The purpose of this talk is to show DPDK developers how they might integrate running DTS into their existing development workflow, and highlight the benefits of doing this.

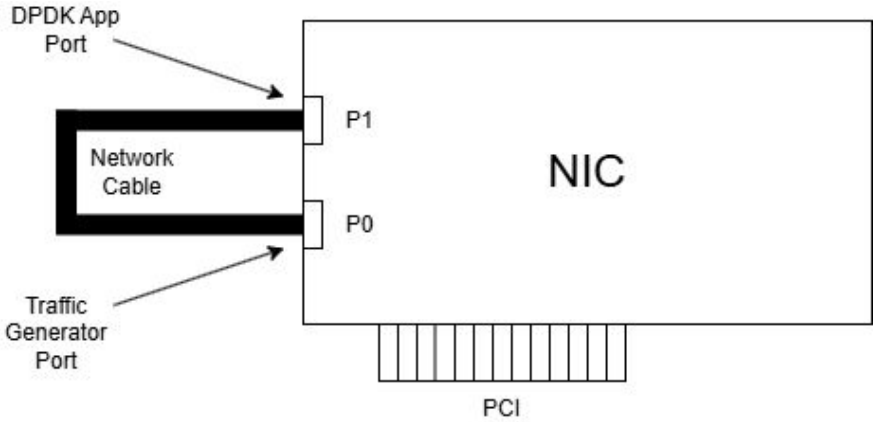
DTS Explained In 15 Seconds

- DTS is an E2E testing framework for DPDK which will run real workloads on DPDK enabled hardware (i.e. it will send real traffic to/from DPDK controlled network interfaces).
- For an in depth explanation of DTS, please refer to DTS talks from previous DPDK Summits or the DPDK docs (most up to date info).



*Disclaimer: The above represent **logical** nodes, not **physical** nodes. For instance, the DTS environment, the TG, and SUT could all be on the same physical host.

Simplest DTS Topology: 1 dual interface NIC on 1 host, port 0 and port 1 connected back to back. Traffic Generator runs on port 0, DPDK app runs on port 1. **This is what I am showing today.**



OR, enable extra tests by running DTS with 2 hosts connected to one another, and adding a second TG <-> SUT network cable link

Typical DPDK Development/Testing Workflow

1. Developer writes a patch for DPDK
2. Developer tests their patch locally with:
   a. app/test-pmd (ethdev), app/test-crypto-perf (cryptodev), and other /app/* applications
   b. app/dpdk-test (meson test for software tests of DPDK libraries and drivers)
   c. Custom DPDK applications
3. Developer runs checkpatches, check-git-log, etc.
4. Patch submitted to dev@dpdk.org
5. CI labs pull the patch and run testing:
   a. Software tests on a wide set of environments
   b. DTS testing on a diverse set of CPU architectures and NICs
6. If step 5 reveals an issue, developer goes back to step 1

Improved Workflow

1. Developer writes a patch for DPDK
2. Developer tests their patch locally with:
   a. app/test-pmd (ethdev), app/test-crypto-perf (cryptodev), and other /app/* applications
   b. app/dpdk-test (meson test for software tests of DPDK libraries and drivers)
   c. Custom DPDK applications

   AND/OR

   Developer runs DTS testing on their local environment.

3. Developer runs checkpatches, check-git-log, etc.
4. Patch submitted to dev@dpdk.org
5. CI labs pull the patch and run testing:
   a. Software tests on a wide set of environments
   b. DTS testing on a diverse set of CPU architectures and NICs
6. If step 5 reveals an issue, developer goes back to step 1

Today's Demo

1. First, I will show how I commit a patch updating rte_flow behavior in DPDK.
2. Then, I will run DTS on my laptop. DTS will remotely control a host at the DPDK Community Lab. On the test host we will use a single dual port NIC, with its 2 ports connected back to back.
    a. Running the rte_flow testsuite in this demo
3. DTS controls both ports on the NIC, transmitting traffic from port 0 (TG) with Scapy, and processing traffic with testpmd on port 1 (SUT).
4. Then, we will check the results of DTS, and whether my patch is safe to be submitted to Patchwork and merged into DPDK...

Okay, so we are patching flow offload behavior…

To prove a point, we will produce a patch that forces packets to be directed to queue 0 regardless of existing flow rules.

We will be using a Mellanox CX7 NIC, for the demonstration.

First, we write our patch

```
diff --git a/drivers/net/mlx5/mlx5_flow_dv.c b/drivers/net/mlx5/mlx5_flow_dv.c
index 32e75b063f..67769fd716 100644
--- a/drivers/net/mlx5/mlx5_flow_dv.c
+++ b/drivers/net/mlx5/mlx5_flow_dv.c
@@ -15265,7 +15265,9 @@ flow_dv_translate(struct rte_eth_dev *dev,
                case RTE_FLOW_ACTION_TYPE_QUEUE:
                        queue = actions->conf;
                        rss_desc->queue_num = 1;
-                       rss_desc->queue[0] = queue->index;
+                       /* Force queue to 0 for demo purposes. */
+                       (void)queue;
+                       rss_desc->queue[0] = 0;
                        action_flags |= MLX5_FLOW_ACTION_QUEUE;
                        dev_flow->handle->fate_action = MLX5_FLOW_FATE_QUEUE;
                        sample_act->action_flags |= MLX5_FLOW_ACTION_QUEUE;
```

First, we write our patch

```
diff --git a/drivers/net/mlx5/mlx5_flow_hw.c b/drivers/net/mlx5/mlx5_flow_hw.c
index bca5b2769e..1a9ebcd595 100644
--- a/drivers/net/mlx5/mlx5_flow_hw.c
+++ b/drivers/net/mlx5/mlx5_flow_hw.c
@@ -888,10 +888,14 @@ flow_hw_tir_action_register(struct rte_eth_dev *dev,
        };
        struct mlx5_hrxq *hrxq;

-       if (action->type == RTE_FLOW_ACTION_TYPE_QUEUE) {
-               const struct rte_flow_action_queue *queue = action->conf;
+       /*
+        * Force all queue actions to queue 0 for demo purposes.
+        * Use a static zero index so const_q has a valid pointer.
+        */
+       static const uint16_t demo_queue_zero = 0;

-               rss_desc.const_q = &queue->index;
+       if (action->type == RTE_FLOW_ACTION_TYPE_QUEUE) {
+               rss_desc.const_q = &demo_queue_zero;
                rss_desc.queue_num = 1;
        } else {
                const struct rte_flow_action_rss *rss = action->conf;
```

First, we write our patch

```
diff --git a/drivers/net/mlx5/mlx5_flow_verbs.c b/drivers/net/mlx5/mlx5_flow_verbs.c
index bb240d38d9..9069585189 100644
--- a/drivers/net/mlx5/mlx5_flow_verbs.c
+++ b/drivers/net/mlx5/mlx5_flow_verbs.c
@@ -1091,7 +1091,9 @@ flow_verbs_translate_action_queue(struct mlx5_flow_rss_desc *rss_desc,
 {
        const struct rte_flow_action_queue *queue = action->conf;

-       rss_desc->queue[0] = queue->index;
+       /* Force all queue actions to queue 0 for demo purposes. */
+       (void)queue;
+       rss_desc->queue[0] = 0;
        rss_desc->queue_num = 1;
 }
```

First, we write our patch

```
commit 9e103606973a3186426f469e4497b8239bfd83d1 (HEAD -> mlnx_test)
Author: Patrick Robb <patrick.robb@amd.com>
Date:   Sun May 3 22:30:52 2026 -0400

    drivers/mlx5: force all packets to queue 0

    This commit will force all packets to be sent to
    queue 0, regardless of flow offload rules set on
    the DPDK app.

    Signed-off-by: Patrick Robb <patrickrobb1997@gmail.com>
```

Great… our patch is complete. Since I’m a talented developer I know this totally won’t break anything in DPDK, and we can send the patch off to the dev mailing list, right?

WRONG

We should run DTS first… just in case.

Running DTS

Reminder: DTS is contained within the main DPDK repo. When your code changes are complete, just configure DTS to refer to the DPDK dir which contains it. It will copy its DPDK Repo to the SUT, and run DTS tests against your new code.

Steps:

1. Complete DTS setup (simple process, but I won't cover here - read the docs)
2. Configure test_run.yaml and nodes.yaml (DTS config files). I will show my config, but please refer to docs for a complete explanation of all fields.
3. Run "poetry run ./main.py"

My test_run.yaml config

```
dpdk:
  lcores: "11-14"
  memory_channels: 4
  build:
    dpdk_location:
      dpdk_tree: .. # Copy the DPDK dir which contains DTS to the SUT
      remote: false # Optional, defaults to false.
    build_options:
      compiler: gcc
      compiler_wrapper: ccache # Optional.
func_traffic_generator:
  type: SCAPY
perf: false # disable performance testing
func: true # enable functional testing
crypto: false
use_virtual_functions: false
skip_smoke_tests: true
test_suites:
  - rte_flow
system_under_test_node: "cx7-SUT"
traffic_generator_node: "cx7-TG"
port_topology:
  - sut.port-0 <-> tg.port-0  # explicit link. `sut` and `tg` are special identifiers
```

My nodes.yaml config

```
- name: "cx7-TG"
  hostname: "dell-amd-2.dpdklab.iol.unh.edu"
  user: probb
  os: linux
  ports:
    - name: port-0
      pci: "0000:21:00.0"
      os_driver_for_dpdk: mlx5_core
      os_driver: mlx5_core
- name: "cx7-SUT"
  hostname: "dell-amd-2.dpdklab.iol.unh.edu"
  user: probb
  os: linux
  ports:
    - name: port-0
      pci: "0000:21:00.1"
      os_driver_for_dpdk: mlx5_core
      os_driver: mlx5_core
```

Great. Let’s run DTS.

```
root@7b8b723dd418:/dpdk/dts#
root@7b8b723dd418:/dpdk/dts# poetry run ./main.py
2026/05/04 23:07:26 - pre_run - dts.cx7-TG - INFO - Connecting to probb@dell-amd-2.dpdklab.iol.unh.edu.
2026/05/04 23:07:26 - pre_run - dts.cx7-TG - INFO - Connection to probb@dell-amd-2.dpdklab.iol.unh.edu successful.
2026/05/04 23:07:26 - pre_run - dts.cx7-TG - INFO - Initializing interactive connection for probb@dell-amd-2.dpdklab.iol.unh.edu
2026/05/04 23:07:27 - pre_run - dts.cx7-TG - INFO - Interactive connection successful for probb@dell-amd-2.dpdklab.iol.unh.edu
2026/05/04 23:07:27 - pre_run - dts.cx7-TG - INFO - Sending: 'uname -m'
2026/05/04 23:07:27 - pre_run - dts.cx7-TG - INFO - Connected to node: cx7-TG
2026/05/04 23:07:27 - pre_run - dts.cx7-TG - INFO - Getting CPU information.
2026/05/04 23:07:27 - pre_run - dts.cx7-TG - INFO - Sending: 'lscpu -p=CPU,CORE,SOCKET,NODE|grep -v \#'
2026/05/04 23:07:27 - pre_run - dts.cx7-TG - INFO - Created node: cx7-TG
2026/05/04 23:07:27 - pre_run - dts.cx7-SUT - INFO - Connecting to probb@dell-amd-2.dpdklab.iol.unh.edu.
2026/05/04 23:07:28 - pre_run - dts.cx7-SUT - INFO - Connection to probb@dell-amd-2.dpdklab.iol.unh.edu successful.
2026/05/04 23:07:28 - pre_run - dts.cx7-SUT - INFO - Initializing interactive connection for probb@dell-amd-2.dpdklab.iol.unh.edu
2026/05/04 23:07:28 - pre_run - dts.cx7-SUT - INFO - Interactive connection successful for probb@dell-amd-2.dpdklab.iol.unh.edu
2026/05/04 23:07:28 - pre_run - dts.cx7-SUT - INFO - Sending: 'uname -m'
2026/05/04 23:07:28 - pre_run - dts.cx7-SUT - INFO - Connected to node: cx7-SUT
2026/05/04 23:07:28 - pre_run - dts.cx7-SUT - INFO - Getting CPU information.
2026/05/04 23:07:28 - pre_run - dts.cx7-SUT - INFO - Sending: 'lscpu -p=CPU,CORE,SOCKET,NODE|grep -v \#'
2026/05/04 23:07:28 - pre_run - dts.cx7-SUT - INFO - Created node: cx7-SUT
2026/05/04 23:07:28 - pre_run - dts.cx7-SUT dpdk_build - INFO - Connecting to probb@dell-amd-2.dpdklab.iol.unh.edu.
2026/05/04 23:07:29 - pre_run - dts.cx7-SUT dpdk_build - INFO - Connection to probb@dell-amd-2.dpdklab.iol.unh.edu successful.
2026/05/04 23:07:29 - pre_run - dts.cx7-SUT dpdk_build - INFO - Initializing interactive connection for probb@dell-amd-2.dpdklab.iol.unh.edu
2026/05/04 23:07:29 - pre_run - dts.cx7-SUT dpdk_build - INFO - Interactive connection successful for probb@dell-amd-2.dpdklab.iol.unh.edu
2026/05/04 23:07:29 - pre_run - dts - INFO - Not using first core
2026/05/04 23:07:29 - pre_run - dts - INFO - Running test run with SUT 'cx7-SUT'.
2026/05/04 23:07:29 - pre_run - dts - INFO - Moving from stage 'pre_run' to stage 'test_run_setup'.
2026/05/04 23:07:29 - test_run_setup - dts - INFO - Running on SUT node 'cx7-SUT'.
2026/05/04 23:07:29 - test_run_setup - dts - INFO - Initializing with random seed 2417556547.
2026/05/04 23:07:29 - test_run_setup - dts.cx7-SUT - INFO - Sending: 'mktemp -dp "" dts.XXXXX'
2026/05/04 23:07:30 - test_run_setup - dts.cx7-TG - INFO - Sending: 'mktemp -dp "" dts.XXXXX'
2026/05/04 23:07:30 - test_run_setup - dts.cx7-SUT - INFO - Sending: 'mkdir -p /tmp/dts.a6ozc/dpdk'
2026/05/04 23:07:30 - test_run_setup - dts - INFO - Copying DPDK source tree to SUT: '/dpdk' into '/tmp/dts.a6ozc/dpdk'.
2026/05/04 23:10:37 - test_run_setup - dts.cx7-SUT dpdk_build - INFO - Sending: 'tar tf /tmp/dts.a6ozc/dpdk/dpdk.tar.gz'
2026/05/04 23:10:38 - test_run_setup - dts.cx7-SUT dpdk_build - INFO - Sending: 'tar xfmv /tmp/dts.a6ozc/dpdk/dpdk.tar.gz -C /tmp/dts.a6ozc/dpdk --strip-com
ponents=1'
2026/05/04 23:10:39 - test_run_setup - dts.cx7-SUT dpdk_build - INFO - Sending: 'ls /tmp/dts.a6ozc/dpdk'
2026/05/04 23:10:39 - test_run_setup - dts.cx7-SUT dpdk_build - INFO - Sending: 'rm -f /tmp/dts.a6ozc/dpdk/dpdk.tar.gz'
2026/05/04 23:10:39 - test_run_setup - dts.cx7-SUT dpdk_build - INFO - Sending: 'gcc --version'
2026/05/04 23:10:39 - test_run_setup - dts.cx7-SUT dpdk_build - INFO - Configuring DPDK build from scratch.
2026/05/04 23:10:39 - test_run_setup - dts.cx7-SUT dpdk_build - INFO - Sending: 'rm -rf /tmp/dts.a6ozc/dpdk/x86_64-linux-gcc'
2026/05/04 23:10:39 - test_run_setup - dts.cx7-SUT dpdk_build - INFO - Sending: 'meson setup --default-library=static -Dlibdir=lib /tmp/dts.a6ozc/dpdk /tmp/
dts.a6ozc/dpdk/x86_64-linux-gcc' with env vars: '{'CC': "'ccache gcc'"}'
2026/05/04 23:10:48 - test_run_setup - dts.cx7-SUT dpdk_build - INFO - Building DPDK.
2026/05/04 23:10:48 - test_run_setup - dts.cx7-SUT dpdk_build - INFO - Sending: 'ninja -C /tmp/dts.a6ozc/dpdk/x86_64-linux-gcc' with env vars: '{'CC': "'cca
```

```
2026/05/05 02:20:49 - test_case - dts.cx7-SUT.TestPmd - INFO - Sending: 'sudo -- sh -c '/tmp/dts.THgq0/dpdk/x86_64-linux-gcc/app/dpdk-testpmd -l
11-12 --file-prefix=dpdk_4890_20260505021945 -a 0000:21:00.1 -- -i --port-topology=loop --rxq=4 --txq=4 --mask-event=intr_lsc''
2026/05/05 02:20:50 - test_case - dts.TestRteFlow - INFO - Testing: eth[src=02:00:00:00:00:00] -> drop
2026/05/05 02:20:50 - test_case - dts.cx7-SUT.TestPmd - INFO - Sending: 'flow validate 0 group 0 ingress pattern eth src is 02:00:00:00:00:00 / e
nd actions drop / end'
2026/05/05 02:20:50 - test_case - dts.cx7-SUT.TestPmd - INFO - Sending: 'flow create 0 group 0 ingress pattern eth src is 02:00:00:00:00:00 / end
 actions drop / end'
2026/05/05 02:20:50 - test_case - dts.cx7-SUT.TestPmd - INFO - Sending: 'show port info all'
2026/05/05 02:20:50 - test_case - dts.cx7-SUT.TestPmd - INFO - Sending: 'show port info 0'
2026/05/05 02:20:50 - test_case - dts.cx7-SUT.TestPmd - INFO - Sending: 'start'
2026/05/05 02:20:51 - test_case - dts.cx7-SUT.TestPmd - INFO - Sending: 'start'
2026/05/05 02:20:51 - test_case - dts.cx7-TG TrafficGeneratorType.SCAPY - INFO - Sending packet.
2026/05/05 02:20:51 - test_case - dts.cx7-TG TrafficGeneratorType.SCAPY - INFO - Building a list of packets to send.
2026/05/05 02:20:51 - test_case - dts.cx7-TG.scapy - INFO - Sending: 'packets = [Ether(src='02:00:00:00:00:00', dst='58:a2:e1:fa:89:19')/Raw(load
=b'XXXXX')]'
2026/05/05 02:20:51 - test_case - dts.cx7-TG.scapy - INFO - Sending: 'sendp(
    packets,
    iface='enp33s0f0np0',
    realtime=True,
    verbose=True
    )'
WARNING: PcapWriter: unknown LL type for NoneType. Using type 1 (Ethernet)
2026/05/05 02:20:52 - test_case - dts.cx7-SUT.TestPmd - INFO - Sending: 'stop'
2026/05/05 02:20:52 - test_case - dts.cx7-SUT.TestPmd - INFO - Sending: 'flow destroy 0 rule 0'
2026/05/05 02:20:52 - test_case - dts.TestRteFlow - INFO - Testing: eth[dst=02:00:00:00:00:02] -> drop
2026/05/05 02:20:52 - test_case - dts.cx7-SUT.TestPmd - INFO - Sending: 'flow validate 0 group 0 ingress pattern eth dst is 02:00:00:00:00:02 / e
nd actions drop / end'
2026/05/05 02:20:52 - test_case - dts.cx7-SUT.TestPmd - INFO - Sending: 'flow create 0 group 0 ingress pattern eth dst is 02:00:00:00:00:02 / end
 actions drop / end'
2026/05/05 02:20:52 - test_case - dts.cx7-SUT.TestPmd - INFO - Sending: 'show port info 0'
2026/05/05 02:20:52 - test_case - dts.cx7-SUT.TestPmd - INFO - Sending: 'start'
2026/05/05 02:20:52 - test_case - dts.cx7-SUT.TestPmd - INFO - Sending: 'start'
2026/05/05 02:20:52 - test_case - dts.cx7-TG TrafficGeneratorType.SCAPY - INFO - Sending packet.
2026/05/05 02:20:52 - test_case - dts.cx7-TG TrafficGeneratorType.SCAPY - INFO - Building a list of packets to send.
2026/05/05 02:20:52 - test_case - dts.cx7-TG.scapy - INFO - Sending: 'packets = [Ether(dst='02:00:00:00:00:02', src='58:a2:e1:fa:89:18')/Raw(load
=b'XXXXX')]'
2026/05/05 02:20:52 - test_case - dts.cx7-TG.scapy - INFO - Sending: 'sendp(
    packets,
```

```
Results
=======
test_suites: FAIL
  rte_flow: FAIL
    drop_action: PASS
    jump_action: FAIL
      reason: Flow rules failed:
        eth[src=02:00:00:00:00:00] -> jump -> ipv4[dst=192.168.1.2] -> queue 2 (group 1 -> queue)
          Pattern: ipv4 dst is 192.168.1.2
          Reason: eth[src=02:00:00:00:00:00] -> jump -> ipv4[dst=192.168.1.2] -> queue 2: Matching packet not received on queue 2 after jump
          Sent Packet: Ether / IP / UDP 192.168.1.1:5000 > 192.168.1.2:domain / Raw
        eth[dst=02:00:00:00:00:02] -> jump -> ipv4[src=192.168.1.1] -> queue 3 (group 1 -> queue)
          Pattern: ipv4 src is 192.168.1.1
          Reason: eth[dst=02:00:00:00:00:02] -> jump -> ipv4[src=192.168.1.1] -> queue 3: Matching packet not received on queue 3 after jump
          Sent Packet: Ether / IP / UDP 192.168.1.1:5000 > 192.168.1.2:domain / Raw
        ipv4[src=10.0.0.1] -> jump -> tcp[dst=443] -> queue 1 (group 1 -> queue)
          Pattern: ipv4 / tcp dst is 443
          Reason: ipv4[src=10.0.0.1] -> jump -> tcp[dst=443] -> queue 1: Matching packet not received on queue 1 after jump
          Sent Packet: Ether / IP / TCP 10.0.0.1:12345 > 10.0.0.2:https S / Raw
        ipv4[dst=172.16.0.1] -> jump -> udp[dst=123] -> queue 2 (group 1 -> queue)
          Pattern: ipv4 / udp dst is 123
          Reason: ipv4[dst=172.16.0.1] -> jump -> udp[dst=123] -> queue 2: Matching packet not received on queue 2 after jump
          Sent Packet: Ether / IP / UDP 172.16.0.100:5000 > 172.16.0.1:ntp / Raw
        eth[src=02:00:00:00:00:00] -> jump -> udp[dst=53] -> queue 3 (group 1 -> queue)
          Pattern: ipv4 / udp dst is 53
          Reason: eth[src=02:00:00:00:00:00] -> jump -> udp[dst=53] -> queue 3: Matching packet not received on queue 3 after jump
          Sent Packet: Ether / IP / UDP 192.168.1.1:5000 > 192.168.1.2:domain / Raw
        ipv4[tos=4] -> jump -> tcp[dst=80] -> queue 1 (group 1 -> queue)
          Pattern: ipv4 / tcp dst is 80
          Reason: ipv4[tos=4] -> jump -> tcp[dst=80] -> queue 1: Matching packet not received on queue 1 after jump
          Sent Packet: Ether / IP / TCP 10.0.0.1:54321 > 10.0.0.2:http S / Raw
    modify_field_action: PASS
    queue_action: FAIL
      reason: Flow rules failed:
        eth[src=02:00:00:00:00:00] -> queue
          Pattern: eth src is 02:00:00:00:00:00
          Reason: Packet not received on queue 2
          Sent Packet: 02:00:00:00:00:00 > None (0x9000) / Raw
        eth[dst=02:00:00:00:00:02] -> queue
          Pattern: eth dst is 02:00:00:00:00:02
          Reason: Packet not received on queue 2
```

```
Results
=======
test_suites: FAIL
  rte_flow: FAIL
    drop_action: PASS
    jump_action: FAIL
      reason: Flow rules failed:
        eth[src=02:00:00:00:00:00] -> jump -> ipv4[dst=192.168.1.2] -> queue 2 (group 1 -> queue)
          Pattern: ipv4 dst is 192.168.1.2
          Reason: eth[src=02:00:00:00:00:00] -> jump -> ipv4[dst=192.168.1.2] -> queue 2: Matching packet not received on queue 2 after jump
          Sent Packet: Ether / IP / UDP 192.168.1.1:5000 > 192.168.1.2:domain / Raw
        eth[dst=02:00:00:00:00:02] -> jump -> ipv4[src=192.168.1.1] -> queue 3 (group 1 -> queue)
          Pattern: ipv4 src is 192.168.1.1
          Reason: eth[dst=02:00:00:00:00:02] -> jump -> ipv4[src=192.168.1.1] -> queue 3: Matching packet not received on queue 3 after jump
          Sent Packet: Ether / IP / UDP 192.168.1.1:5000 > 192.168.1.2:domain / Raw
        ipv4[src=10.0.0.1] -> jump -> tcp[dst=443] -> queue 1 (group 1 -> queue)
          Pattern: ipv4 / tcp dst is 443
          Reason: ipv4[src=10.0.0.1] -> jump -> tcp[dst=443] -> queue 1: Matching packet not received on queue 1 after jump
          Sent Packet: Ether / IP / TCP 10.0.0.1:12345 > 10.0.0.2:https S / Raw
        ipv4[dst=172.16.0.1] -> jump -> udp[dst=123] -> queue 2 (group 1 -> queue)
          Pattern: ipv4 / udp dst is 123
          Reason: ipv4[dst=172.16.0.1] -> jump -> udp[dst=123] -> queue 2: Matching packet not received on queue 2 after jump
          Sent Packet: Ether / IP / UDP 172.16.0.100:5000 > 172.16.0.1:ntp / Raw
        eth[src=02:00:00:00:00:00] -> jump -> udp[dst=53] -> queue 3 (group 1 -> queue)
          Pattern: ipv4 / udp dst is 53
          Reason: eth[src=02:00:00:00:00:00] -> jump -> udp[dst=53] -> queue 3: Matching packet not received on queue 3 after jump
          Sent Packet: Ether / IP / UDP 192.168.1.1:5000 > 192.168.1.2:domain / Raw
        ipv4[tos=4] -> jump -> tcp[dst=80] -> queue 1 (group 1 -> queue)
          Pattern: ipv4 / tcp dst is 80
          Reason: ipv4[tos=4] -> jump -> tcp[dst=80] -> queue 1: Matching packet not received on queue 1 after jump
          Sent Packet: Ether / IP / TCP 10.0.0.1:54321 > 10.0.0.2:http S / Raw
    modify_field_action: PASS
    queue_action: FAIL
      reason: Flow rules failed:
        eth[src=02:00:00:00:00:00] -> queue
          Pattern: eth src is 02:00:00:00:00:00
          Reason: Packet not received on queue 2
          Sent Packet: 02:00:00:00:00:00 > None (0x9000) / Raw
        eth[dst=02:00:00:00:00:02] -> queue
          Pattern: eth dst is 02:00:00:00:00:02
          Reason: Packet not received on queue 2
```

DTS Output

- DTS provides a high level testsuite PASS/FAIL summary to stdout when it is finished executing (seen in the previous slide)
- For extra information, the DTS output dir also contains extended logs and artifacts
  - dts/output/**results_summary.txt** (human readable high level)
  - dts/output/**results_summary.json** (json high level for automation)
  - dts/output/**dts.log**
  - dts/output/**dts.verbose.log** (debug level info)
  - dts/output/**<test_suite_name>**/
    - Contains pcaps, test artifacts copied during execution time, and testsuite/testcase specific logs

Conclusion

- So, DTS has shown that our patch actually IS breaking DPDK functionality. Who knew? I guess I won't send it to the mailing list after all (until I make some updates to my patch)
- By running your new DPDK patch over DTS testsuites before sending it to the mailing list:
  - You tighten the testing feedback loop, getting test results much quicker
  - You will need to send fewer patch versions to the mailing list, reducing mailing list traffic, reducing test load on CI lab systems, and saving yourself time
  - You get the opportunity to compare your implementations against the community driven "expected functionality" as defined in the DTS test suites

Thank you

I would like to thank a few people who helped make this presentation possible.

1. **Dean Marx (UNH-IOL):** Wrote the (extended) rte_flow testsuite for 26.03. Congratulations Dean for graduating from UNH this Spring and starting a job at Canonical!
2. **Andrew Bailey(UNH-IOL):** Killing it on DTS tickets this year - the new king of DTS Bugzilla.
3. **Lincoln Lavoie (UNH-IOL):** Who has acted as mentor for me for the past few years, and has lent his many years of networking expertise towards DTS on many occasions.
4. **Luca Vizzarro (ARM):** Doing a wonderful job holding down the fort as DTS maintainer
5. And thank you to everyone who has gone out of their way to engage with the DTS effort and the students at UNH who have been working on this effort. Names like Morten Brorup and Bruce Richardson come to mind, but there have been others. Thanks everyone.


The End - Avsluta!