UPSTREAM KERNEL CI USING PATCHEW AND TDC

DAVIDE CARATTI
Software Engineer, Networking Services Team
We are going to...

- Test the control plane of ‘bpf’ action using TDC selftests
- Automate the test using patchew
- Overview what’s next with regards to TDC
- Questions, if any
Motivation

• TC code is in the Linux kernel since decades, but it’s constantly evolving (new qdiscs, modifications to the classifiers / actions, performance improvements)

• Users don’t like undesired changes of the behavior

• In netdev 1.2, TDC has been introduced to unit-test the TC control plane

• Patchew project recently landed upstream to provide CI for QEMU
Credits to:

- Lucas Bates, Alexander Aring, Keara Leibovitz, Roman Mashak, for maintaining TDC in upstream Linux tree
- Paolo Bonzini, Fam Zheng and QEMU developers for developing patchew (and helping me a lot)
- You, if you want to contribute!

---

Date: 2018-10-16 5:38:05
Message-ID: 20181015.223805.1273944578999894952.davem () davemloft ! net
[Download RAW message or body]

From: Stefano Brivio <sbrivio@redhat.com>
Date: Fri, 12 Oct 2018 23:54:12 +0200

> This series adds a couple of features useful for debugging: 1/2
> allows selecting single tests and 2/2 adds optional traffic captures.
> Semantics for current invocation of test script are preserved.

MOAR SELF TESTS!
I love it.
Keep them coming.
Series applied, thanks.
TC: the Linux Traffic Control tool

- can create/modify/delete/dump info on qdiscs/filters/actions
- sends configuration to the kernel using netlink messages
- (a lot of?) documentation available

```bash
# strace -xx -e verbose=%network tc action show action simple
...
recvmsg(3, {msg_name={sa_family=AF_NETLINK, nl_pid=0, nl_groups=00000000}, msg_namelen=12, msg_iov=[{iov_base={len=732, type=RTM_GETACTION, flags=NLM_F_MULTI, seq=1547655914, pid=31917}, {tca_family=AF_UNSPEC}, [{nla_len=8, nla_type=TCA_ACT_INDEX}, 5], [{nla_len=704, nla_type=TCA_ACT_KIND}, "\x8c"}], iov_len=732}], msg_iovlen=1, msg_controllen=0, msg_flags=0}, 0) = 732
```

- now consider: there are 35 qdisc, 12 classifiers, 15 actions
Unit tests for TC: an example

What happens when I install a dummy eBPF action?

```
# tc action add action bpf \
> obj action.o section foo index 2
# tc action show action bpf
[...]
action order 0: bpf action.o:[foo] id 34 tag
a04f5eef06a7f555 jited default-action pipe
index 2 ref 1 bind 0
```

- `tc_bpf_init()`
- `tc_bpf_dump()`
- `TCA_ACT_BPF_NAME`
- `TCA_ACT_BPF_FD`
- `TCA_ACT_KIND`
- `TCA_ACT_INDEX`
Unit tests for TC: an example, with TDC

```bash
# cd tools/testing/selftests/tc-testing
# ./tdc.py -l -c bpf

[...]
e939: (actions, bpf) Add eBPF action with valid object-file
# ./tdc.py -B -e e939
Test e939: Add eBPF action with valid object-file

All test results:

1..1
ok 1 e939 - Add eBPF action with valid object-file
```
### How test cases are written (#1)

- Each TC component has a JSON file containing a list of test cases
- Each test is composed by setup, command, verify, teardown phases

```json
{
    "id": "e939",
    "name": "Add eBPF action with valid object-file",
    ...
    "cmdUnderTest": "$TC action add action bpf object-file $EBPFDIR/action.o section action-ok index 667",
    "expExitCode": "0",
    "verifyCmd": "$TC action get action bpf index 667",
    "matchPattern": "action order [0-9]*: bpf action.o:\[action-ok\] id [0-9]* tag [0-9a-f]{16}( jited)? default-action pipe.*index 667 ref",
    "matchCount": "1",
    ...
}
```
How test cases are written (#2)

- Each TC component has a JSON file containing a list of test cases
- Each test is composed by setup, command, verify, teardown phases

```bash
# tc action add action bpf
> obj action section foo index 2
# tc action show action bpf
[...]
action order 0: bpf action.o:[foo] id 34 tag
a04f5eef06a7f555 jited default-action pipe
index 2 ref 1 bind 0
```

- cmdUnderTest
- expExitCode
- verifyCmd
- matchPattern
- matchCount
Plugins, and more test cases

- plugins can be used (valgrind, nsPlugin, ...)
- many test cases have been added over the past year, many more are going to be added
Automate it with patchew (#1)

- A lightweight tool meant to test single patches as long as they are sent to a mailing list
- Used by QEMU project
- open source (see https://github.com/patchew-project/patchew)
Automate it with patchew (#2)

- The importer runs an email client, receiving messages from netdev@vger.kernel.org. A filter configured for the inbox moves messages to a designated IMAP folder based on the List-id and on the body.

- The importer parses the email content to match series, single patches and followup messages (such as ‘Reviewed-by’ tag)

```plaintext
From: xiangxia.m.yue@gmail.com
To: netdev@vger.kernel.org, xiyou.wangcong@gmail.com
Cc: Tonghao Zhang <xiangxia.m.yue@gmail.com>
Subject: [PATCH net-next] net: sched: add checks for tbf classid
Date: Tue, 18 Dec 2018 23:08:25 -0800
Message-Id: <1545203305-68572-1-git-send-email-xiangxia.m.yue@gmail.com>
Sender: netdev-owner@vger.kernel.org
Precedence: bulk
List-ID: <netdev.vger.kernel.org>
```
Automate it with patchew (#3)

- A github repository mirrors Linux net-next tree. After the importer detects messages containing patches/series, it pushes them to the git repository with a tag.
- The importer updates the ‘apply status’ on the dashboard with G/G, and also ?, O or R.

![Diagram showing importer pushing to git repository with AP/I](https://github.com/dcaratti/net-next/tags/<message>)
Automate it with patchew (#4)

- the tester periodically polls the server for new patches, rebuilds the kernel for each untested series, then it launches the TDC test in a VM, using the newly built kernel binary
- Test results are then uploaded to the dashboard with T or T
A look at the dashboard (#1)

### patchew

**project:** TDC

#### All series for TDC

<table>
<thead>
<tr>
<th>Status</th>
<th>Subject</th>
<th>Author</th>
<th>Age</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>[Patch net-next] net_sched: add performance counters for basic filter</td>
<td>Cong Wang</td>
<td>5 days</td>
</tr>
<tr>
<td>1-0</td>
<td>[Patch net-next] net_sched: add hit counter for matchall</td>
<td>Cong Wang</td>
<td>6 days</td>
</tr>
<tr>
<td>1-0</td>
<td>[PATCH net-next] net: sched: add checks for tbf classid</td>
<td><a href="mailto:xiangxia.m.yue@gmail.com">xiangxia.m.yue@gmail.com</a></td>
<td>5 weeks</td>
</tr>
</tbody>
</table>

---

15
A look at the dashboard (#2)

[Patch net-next] net_sched: add hit counter for matchall

Cong Wang posted 1 patch 6 days ago | Download mbox

✓ Patches applied successfully (tree, apply log)
git fetch https://github.com/dcaratti/net-next.git tags/patchew/26196317284425.27759-1-xyou.wangcong@gmail.com

✓ Test run_tdc passed

Maintainers: Jamal Hadi Salim <jhs@mojatatu.com>, Cong Wang <xyou.wangcong@gmail.com>, Jiri Pirko <jiri@resnulli.us>, "David S. Miller" <davem@davemloft.net>

include/uapi/linux/ptk_cls.h | 6 ++++++
net/sched/clis_matchall.c | 24 ++++++++************
2 Files changed, 30 insertions(+)

Although matchall always matches packets, however, it still relies on a protocol match first. So it is still useful to have...
Everything worked smoothly, right?

- Scripts I used to create the tester at [https://github.com/dcaratti/patchew-deploy](https://github.com/dcaratti/patchew-deploy)

- Traffic on netdev mailing list was really low, almost zero

- Many TDC tests were broken!

- There is no dashboard (yet) on [https://patchew.org](https://patchew.org), a draft dashboard (with semibroken importer) runs at [http://next.patchew.org/Linux-net-next](http://next.patchew.org/Linux-net-next). Stay tuned!
What’s next?

For TDC:

• functional tests of the datapath
• performance tests
• more test cases (qdiscs)
• Inclusion in CKI project
  https://github.com/CKI-project/tests-beaker
  when there is enough coverage
• Loose dependency on iproute2

For patchew:

https://github.com/patchew-project/patchew/issues
Questions?
THANK YOU
EXTRA
Extra content #1: git configuration

- Test run_tdc passed
  - View 3 series
    - Expand all
    - Fold all
  - Testing configuration
  - Email notifications
  - Git configuration
    - Push remote
      - git@github.com:dcaratti/net-next.git
    - Public repo
      - https://github.com/dcaratti/net-next.git
    - URL template
      - https://github.com/dcaratti/net-next/tree/%t

Save
Extra content #2: the test script

cp $HOME/tdc_kernel_config .config
export curwd=$(pwd)
yes "" | make oldconfig
make -j4
qemu-system-x86_64 -nographic -nodefaults -machine q35,accel=kvm -m 1g \
  -drive "file=${HOME}/f29.qcow2,media=disk,format=qcow2,if=virtio" \
  -netdev bridge,br=virbr0,id=hostnet0 \
  -chardev stdio,signal=off,id=channel0,mux=on \
  -mon chardev=channel0 \
  -device virtio-net-pci,netdev=hostnet0 \
  -device virtio-serial-pci,id=virtio-serial0 \
  -device virtio-balloon-pci \
  -device virtconsole,bus=virtio-serial0.0,chardev=channel0 \
  -virtfs local,path=${curwd},mount_tag=share,security_model=none \
  -kernel ${curwd}/arch/x86/boot/bzImage -append "ro root=/dev/vda4 console=hvc0
noquiet"
if grep 'not ok' ${curwd}/tools/testing/selftests/tc-testing/test-results.tap; then
  exit 1
fi
exit 0
Extra content #3: commits on net/sched

```
for i in `seq 13 19`;
do
  n=`git log v4.$i..v4.$((i+1)) --no-merges --pretty=oneline --
  net/sched include/linux/tc.h include/uapi/linux/pkt_sched.h \ 
  include/uapi/linux/pkt_cls.h include/uapi/linux/tc_act \ 
  include/uapi/linux/tc_ematch | wc -l` ; echo v4.$((i+1)) $n
done
```
Extra content #4: why did the test fail?