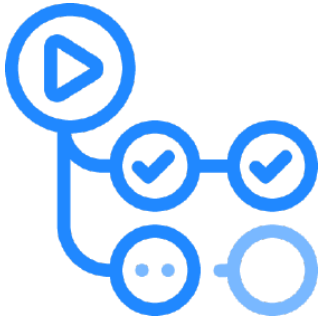
Netdev Conference 2022

P4-TC CI/CD and Test Framework

Deb Chatterjee - Sr. Director Intel
Khan, Mohd Arif <mohd.arif.khan@intel.com> - Cloud Engineer, Intel
Pottimurthy, Sathya Narayana - Cloud Engineer, Intel
Sambasivam, Balachandher - Director, Intel
Pedro Tammela - SW Engineer, Mojatatu
Victor Nogueira - SW Engineer, Mojatatu


intel

Github Actions



For every commit in a GitHub PR we run:

- Checkpatch.pl
- 32/64-bit builds on GCC
- 32/64-bit builds on Clang
- Sparse
- Clang static analyzer

These actions have been running since day one

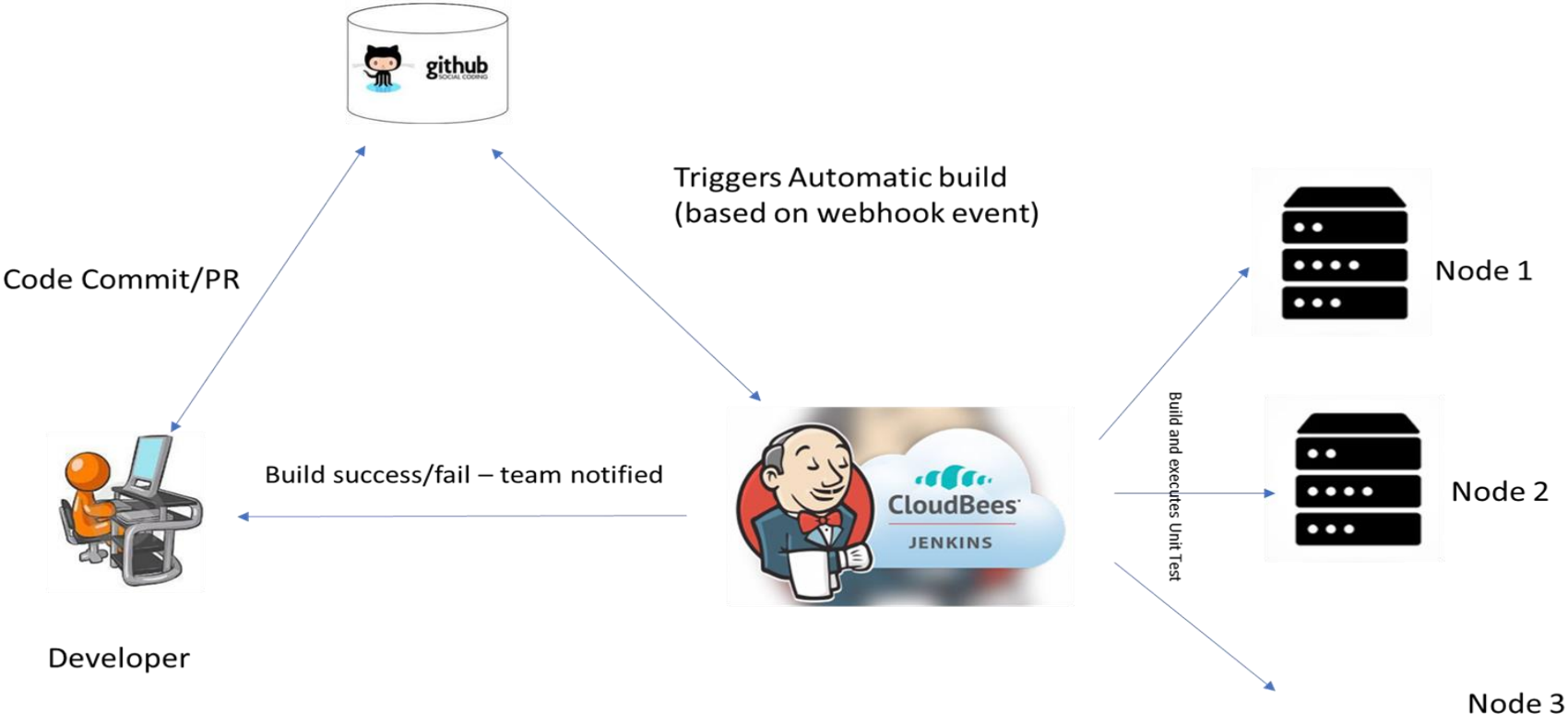
- Warned us about many code style issues
- Caught a few bugs
- Makes sure every commit is buildable

P4-TC CI/CD

- Jenkins receives GitHub PR notifications via webhooks
- Job triggers for GitHub pull_request event (s)
- Unit Test Integration job runs based on Jenkins stages
  - Each job runs in a dedicated machine (baremetal)
  - Clones P4TC kernel and IProute2 repos
  - Invoke vm.sh with unit tests as an argument, parse the results and notify developer (committer)

```
09:45:32  + ./vm.sh -c 6 -m 6G -- ./tdc.py -c p4tc
```

CI/CD Flow


github
Triggers Automatic build
(based on webhook event)
Code Commit/PR
Node 1
Node 2
Node 3
Build and executes Unit Test
Build success/fail – team notified
CloudBees
JENKINS
Developer

TDC tests

- Control path tests for each object (Pipeline, metadata, tables ...)
- As of now we have 376 tests
- Using TC's JSON output to verify the outputs
- We've been adding tests since day one
- Looking to also add data path tests

P4-TC Fuzzy Testing with syzkaller

P4-TC Template for syzkaller

System calls and resources used for fuzzing

socket$nl_route_p4 => Returns the netlink route socket

sendmsg$nl_route_p4_sched => Creates the p4tc netlink messages and sends to kernel

size_t sendmsg(int sockfd, const struct msghdr *msg, int flags)

P4TC netlink messages for following objects

Pipeline

Action

Table Class

Table Instance

Table Entry

Important files being used/added

include/uapi/linux/p4tc.h => kernel inclusion file

sys/linux/socket_netlink_route_p4_sched.txt => netlink message description for Syzkaller

Note: Currently syzkaller runs on 3 Machines(112 cpus, 95 GB) 24x7 with 32 VM's(32 CPUs, 3GB).

We are integrating with CI/CD to run with every pull request.

p4tc.cfg: "enable_syscalls": ["sendmsg$nl_route_p4_sched","socket$nl_route_p4"],

Fuzzy testing - NL Msg Definition

```
include <uapi/linux/p4tc.h>

resource sock_nl_route_p4[sock_netlink]
socket$nl_route_p4(domain const[AF_NETLINK], type const[SOCK_RAW],
                   proto const[NETLINK_ROUTE]) sock_nl_route_p4
sendmsg$nl_route_p4_sched(fd sock_nl_route_p4,
                          msg ptr[in, msghdr_netlink[netlink_msg_route_p4_sched]], f flags[send_flags])
netlink_msg_route_p4_sched [
        newp4pipeline   netlink_msg[RTM_CREATEP4TEMPLATE, p4tcmsg[P4TC_OBJ_PIPELINE], p4tc_root[p4tc_pipeline_policy]]
        delp4pipeline   netlink_msg[RTM_DELP4TEMPLATE,    p4tcmsg[P4TC_OBJ_PIPELINE], p4tc_root[p4tc_pipeline_policy]]
        newp4tclass     netlink_msg[RTM_CREATEP4TEMPLATE, p4tcmsg[P4TC_OBJ_TABLE_CLASS], p4tc_root[p4tc_tclass_policy]]
        delp4tclass     netlink_msg[RTM_DELP4TEMPLATE,    p4tcmsg[P4TC_OBJ_TABLE_CLASS], p4tc_root[p4tc_tclass_policy]]
        newp4tinst      netlink_msg[RTM_CREATEP4TEMPLATE, p4tcmsg[P4TC_OBJ_TABLE_INST], p4tc_root[p4tc_tinst_policy]]
        delp4tinst      netlink_msg[RTM_DELP4TEMPLATE,    p4tcmsg[P4TC_OBJ_TABLE_INST], p4tc_root[p4tc_tinst_policy]]
        newp4acttmpl    netlink_msg[RTM_CREATEP4TEMPLATE, p4tcmsg[P4TC_OBJ_ACT], p4tc_root[p4tc_tmpl_action_policy]]
        delp4acttmpl    netlink_msg[RTM_DELP4TEMPLATE,    p4tcmsg[P4TC_OBJ_ACT], p4tc_root[p4tc_tmpl_action_policy]]
        newp4tabentry   netlink_msg[RTM_CREATEP4TBENT,    p4tcmsg[P4TC_OBJ_TABLE_ENTRY], p4tc_root[p4tc_tabentry_policy]]
        delp4tabentry   netlink_msg[RTM_DELP4TBENT,       p4tcmsg[P4TC_OBJ_TABLE_ENTRY], p4tc_root[p4tc_tabentry_policy]]
] [varlen]

type p4tc_id int32[0:4]
type p4tcmsg[P4_OBJ_TYPE] {
        id      p4tc_id
        obj     const[P4_OBJ_TYPE, int32]
} [packed, align[4]]

type p4tc_root[PARAMS] {
        P4TC_ROOT       nlnest[P4TC_ROOT, array[nlattr_p4tc_batch[p4tc_policy[PARAMS]], 1]]
        P4TC_ROOT_PNAME nlattr[P4TC_ROOT_PNAME, string]
} [packed, align[4]]

# PARAMS must be a struct
type p4tc_policy[PARAMS] {
        P4TC_PATH       nlattr[P4TC_PATH, array[p4tc_id, 1:2]]
        P4TC_PARAMS     nlnest[P4TC_PARAMS, PARAMS]
} [packed, align[4]]

p4tc_pipeline_policy {
        P4TC_PIPELINE_MAXRULES          nlattr[P4TC_PIPELINE_MAXRULES, int32[0:513]]
        P4TC_PIPELINE_NUMTCLASSES       nlattr[P4TC_PIPELINE_NUMTCLASSES, int16[0:33]]
        P4TC_PIPELINE_PREACTIONS        nlnest[P4TC_PIPELINE_PREACTIONS, array[p4tc_actions, 1:4]]
        P4TC_PIPELINE_POSTACTIONS       nlnest[P4TC_PIPELINE_POSTACTIONS, array[p4tc_actions, 1:4]]
} [packed, align[4]]

# Hack for a nest of nests, which behave as a dynamic array
type nlattr_p4tc_batch[PAYLOAD] nlattr_tt[int16:14[0:P4TC_MSGBATCH_SIZE], 0, 0, PAYLOAD]
type nlattr_p4tc_metact_batch[PAYLOAD] nlattr_tt[int16:14[0:TCA_METACT_LIST_MAX], 0, 0, PAYLOAD]

type p4tc_action_policy[NAME, VALUES] {
        TCA_ACT_KIND            nlattr[TCA_ACT_KIND, string[NAME]]
        TCA_ACT_OPTIONS         nlnest[TCA_ACT_OPTIONS, array[VALUES, 1]]
        TCA_ACT_COOKIE          nlattr[TCA_ACT_COOKIE, array[int8, 16]]
        TCA_ACT_FLAGS           nlattr[TCA_ACT_FLAGS, nla_bitfield32[tcf_action_policy_flags]]
        TCA_ACT_HW_STATS        nlattr[TCA_ACT_HW_STATS, nla_bitfield32[tcf_action_policy_hw_stats]]
} [packed, align[4]]

# One of these actions
p4tc_actions [
        action_metact   nlattr_tca_actions[p4tc_action_policy["metact", p4tc_metact_policy]]
] [varlen]
```

Netlink message format

- Netlink Header
- IP Service Template
- IP Service specific TLV's

```
Netlink header
    length
    msg_type => Pass
    flags, sequence number, pid
IP Service Template
    obj => Parameter
    id => u32[0..4]
IP Service specific data TLVs
    name => String
    nesting start => object1
       TLV1 => max rules (0..513)
            ...
       TLVn => preactions (Nested attribute)
       nesting start
           gact or metaact (action kind)
              ...
           action options
       nesting ends
            ...
    nesting end #endof of object1
            ...
    nesting start #Start of objectN
       TLV1 => max rules (0..513)
            ...
       TLVn => preactions (Nested attribute)
       nesting start
           gact or metaact (action kind)
              ...
           action options
      nesting ends
   nesting ends #end of object
```

Crashes Observed

Crash Type	Statistics Root cause , Instances
Dereference of Null Pointer	1,5
Stack out of bounds	1,7
Use after free	2,16
General protection fault	1,3
Memory leaks	1,1
RCU Stall during Object dump	1,1
Soft lockups, work-queue stalls	1,1

We reproduced the crashes using syz-repro utility and the generated c-code helps to fix the crashes

Use-After-free Bug Walkthrough

```
139 tcf_pipeline_create(struct net *net, ..,const char *p_name,
141                     u32 pipeid, struct netlink_ext_ack *extack)
142 {
..
153         pipeline = kmalloc(sizeof(*pipeline), GFP_KERNEL);
..
176
177         if (pipeid) {
178                 ret = idr_alloc_u32(&pipeline_idr, pipeline, &pipeid, pipeid,
179                                     GFP_KERNEL);
..
186                 pipeline->common.p_id = pipeid;
187         } else {
188                 pipeline->common.p_id = 1;
189
190                 ret = idr_alloc_u32(&pipeline_idr, pipeline,
191                                     &pipeline->common.p_id, UINT_MAX,
192                                     GFP_KERNEL);
..
198         }
199
..
212         if (tb[P4TC_PIPELINE_PREACTIONS]) {
213                 pipeline->preacts = kcalloc(TCA_ACT_MAX_PRIO,
214                                             sizeof(struct tc_action *),
215                                             GFP_KERNEL);
216                 if (!pipeline->preacts) {
217                         ret = -ENOMEM;
218                         goto idr_rm;
219                 }
220
..
278 idr_rm:
279         idr_remove(&pipeline_idr, pipeid);
280
281 err:
282         kfree(pipeline);
283         return ERR_PTR(ret);
284 }
```

```
==================================================================
BUG: KASAN: use-after-free in strncmp+0xc2/0xd0
Read of size 1 at addr ffff88805d749000 by task syz-executor.9/25815

CPU: 15 PID: 25815 Comm: syz-executor.9 Tainted: G        W         6.0.0-rc4P4TC-g88cbb83d1e8a-dirty #4
Hardware name: QEMU Standard PC (i440FX + PIIX, 1996), BIOS 1.13.0-1ubuntu1.1 04/01/2014
Call Trace:
 <TASK>
 dump_stack_lvl+0x8b/0xb3
 print_report.cold+0xb2/0x6bb
 kasan_report+0x8a/0x190
 strncmp+0xc2/0xd0
 pipeline_find_name+0x7d/0x120
SELinux: unrecognized netlink message: protocol=6 nlmsg_type=124 sclass=netlink_xfrm_socket pid=25823 comm=syz-executor.3
 tcf_pipeline_cu+0x7fe/0x1ae0
 tcf_p4_tmpl_cu_n+0x8c3/0x1620
 tc_ctl_p4_tmpl_cu+0x14a/0x250
 rtnetlink_rcv_msg+0x43d/0xd70
 netlink_rcv_skb+0x148/0x440
 netlink_unicast+0x54b/0x760
 netlink_sendmsg+0x93d/0xe50
 sock_sendmsg+0x152/0x190
 ____sys_sendmsg+0x710/0x880
 ___sys_sendmsg+0xff/0x170
 __sys_sendmsg+0xf3/0x1c0
 do_syscall_64+0x38/0x90
 entry_SYSCALL_64_after_hwframe+0x63/0xcd
RIP: 0033:0x7f60bd44de4d
Code: 02 b8 ff ff ff ff c3 66 0f 1f 44 00 00 f3 0f 1e fa 48 89 f8 48 89 f7 48 89 d6 48 89 ca 4d 89 c2 4d 89 c8 4c 8b 4c 24
RSP: 002b:00007f60bcbbebf8 EFLAGS: 00000246 ORIG_RAX: 000000000000002e
RAX: ffffffffffffffda RBX: 00007f60bd57bf80 RCX: 00007f60bd44de4d
RDX: 0000000000000000 RSI: 00000000200003c0 RDI: 0000000000000003
RBP: 00007f60bd4bb57c R08: 0000000000000000 R09: 0000000000000000
R10: 0000000000000000 R11: 0000000000000246 R12: 0000000000000000
R13: 00007ffc2518ac6f R14: 00007ffc2518ae10 R15: 00007f60bcbbed80
 </TASK>
```

Fix:

```
278 idr_rm:
279         idr_remove(&pipeline_idr,pipeline->common.p_id);
```

Stack out of bounds

```
-       if ((nla_len(tb[P4TC_PATH]) / sizeof(u32)) > P4TC_PATH_MAX - 1) {
                NL_SET_ERR_MSG(extack, "Path is too big");
                return -E2BIG;
+       if (nla_len(tb[P4TC_PATH])  > ((P4TC_PATH_MAX - P4TC_TBCID_IDX) * sizeof(u32))) {
 memcpy(&ids[P4TC_TBCID_IDX], arg_ids, nla_len(p4tca[P4TC_PATH]));
```

P4-TC Control Path Performance

Control Path Performance - (1/2)

1. Triggered by CI/CD during every PR to catch control performance related issues

2. For this we have added a utility “perf_app” which will give us
   - Update rate (rate of update of table entries)/throughput)
     - Synchronous : waits for acknowledgement
     - Asynchronous: Do not wait for acknowledgement (Slightly better performance than Synchronous)

```
./perf_app --rate --batch 16 --sync
```

   - Latency values (time to add a entry in kernel tables)

```
./perf_app --latency --batch 16 --sync
```

1 Preloads the netlink messages

Control Path Performance - (2/2)

1.Throughput evaluation

- Create 1 pipeline, 1table class, 32 table instances
- Keep adding 60K Rules in each table by changing key and record throughput

2.Observations

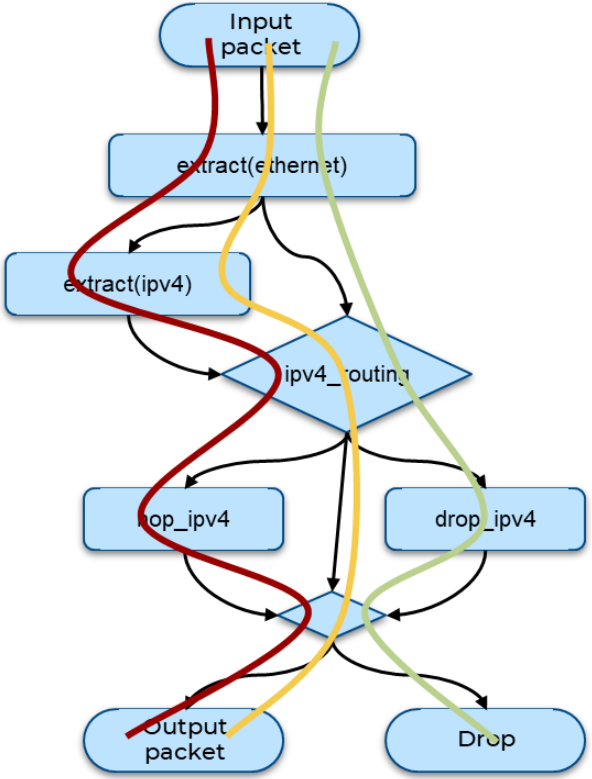
- Asynchronous mode throughput significantly higher than synchronous mode for lower batch sizes
- Both are approximately same at batch sizes 16
- Throughput is directly proportional to batch size

Note: Expect to publish results shortly

P4-TC Testgen

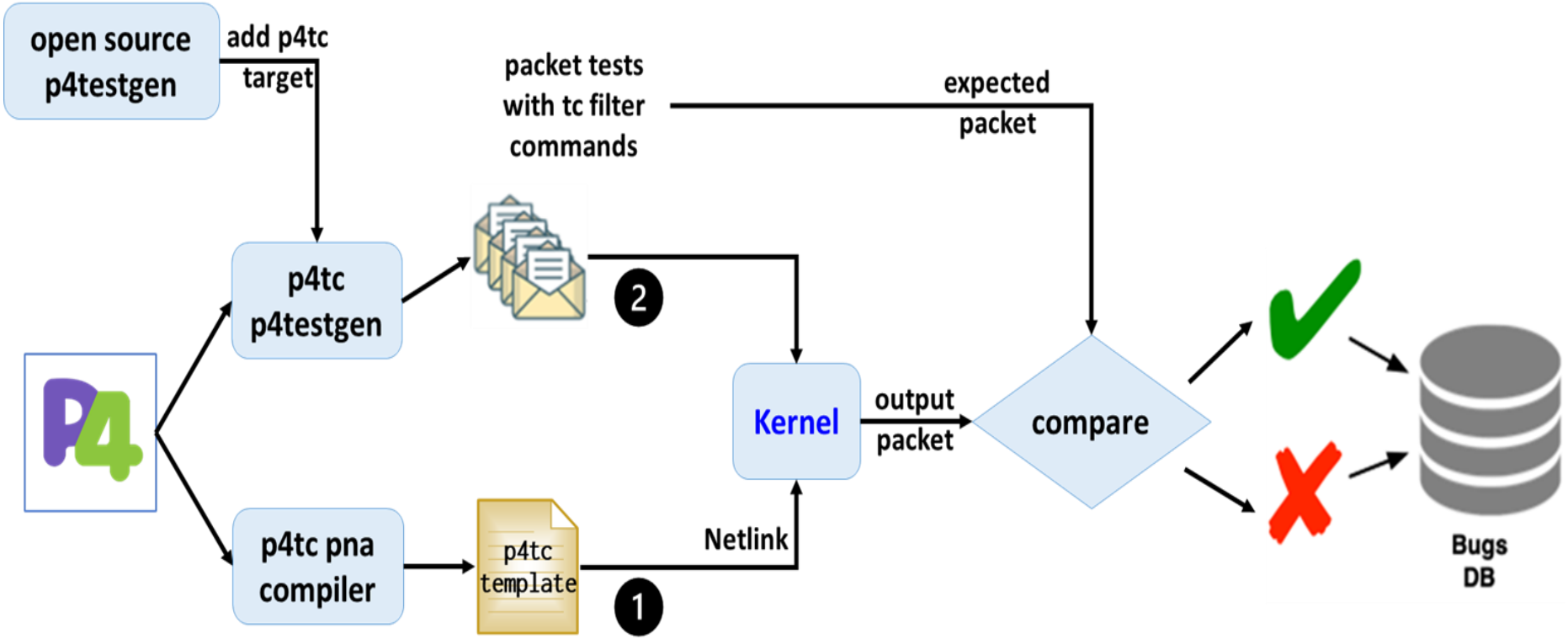
P4 TestGen Overview

- P4Testgen is an automated P4 verification tool for generating test cases for P4 programs.
- Purpose is to cover all path and branches.
- Given a P4 testcase as input to p4testgen tool, p4testgen creates packet tests for all path in the input P4 testcase.
- Number of tests can be controlled by max-tests option.



https://github.com/p4lang/p4c/tree/main/backends/p4tools/testgen

P4 testgen flow for P4-TC



Thank You