Porting of drivers/net/ethernet/broadcom/bnxt/bnxt.c

May 25, 2017

1 Introduction

The driver drivers/net/ethernet/broadcom/bnxt/bnxt.c was introduced in 2013 in the commit c0c050c. It is a Broadcom ethernet driver. This driver was accompanied by changes in Kconfig and two make files, as well as two other .c files and 6 header files. Gcc produces 1 warning.

2 struct net_device_ops.ndo_setup_tc

Gcc reports that the field `ndo_setup_tc` of the structure `struct struct net_device_ops` is initialized to a value of the wrong type. We try the following patch query (step1.cocci):

```
@r1 depends on before@
identifier i,f1;
@@

struct net_device_ops i = {
  .ndo_setup_tc = \(f1\|&f1\),
};

@r2 depends on before expression@
struct net_device_ops e;
struct net_device_ops *ep;
identifier f2;
@@

  <+...\(e.ndo_setup_tc\|ep->ndo_setup_tc\)...+> = \(f2\|&f2\)

@r depends on before@
identifier f,r1.f1,r2.f2;
@@

(
(
 f1
|f2
)
&
 f
)

@@
type T;
```

```
identifier r.f;
@@

- T
  f(...) { ... }

@@
identifier r.f;
parameter p;
@@

f(...,
- p
  ,...) { ... }

@@
identifier r.f;
parameter p;
@@

f(...,
+ p
  ,...) { ... }
```

The first result is 0c71de6 at 100%. It shows that the fourth argument of the setup tc callback function should have a pointer type. This doesn't seem to be the right change, because our driver's callback function has only two arguments.

The next result is 6e2a60b at 100%. This function makes the callback static. This doesn't seem like a relevant change. In any case, the affected function already has 4 arguments.

The next result is 16e5cc6 at 20%. This commit adds a new third parameter to the callback function having type `__be16`. None of the examples use it. This would seem to help us to get from 2 parameters to 4 parameters, but there are currently 3 parameters, so this is at least not the first change we should apply.

The last result is e4c6734 at 8%. Here the parameter list starts with the same types as we have in our code.

We also observe that the first two patches had four parameters, but where the last one did not have u8 type, but rather a structure pointer type. We don't have a commit that shows that change, but we could make it anyway.

```
@r1 depends on before@
identifier i,f;
@@

struct net_device_ops i = {
  .ndo_setup_tc = f,
};

@@
identifier r1.f;
@@

f(p1,
+ u32 handle, __be proto, struct tc_to_netdev *
- u8
  tc) {
... when != S1
```

```
+ if (handle != TC_H_ROOT) return -EINVAL;
S2
...}
```

The provided patches don't illustrate any use of proto, nor any other use of handle. There are many uses of tc, though, and since we did not have a patch changing its type, we don't know how to change the uses.

Since we are aware of the need for the change from our examples, it seems fair enough to search for more examples that may illustrate this issue.

```
@r1 depends on before@
identifier i,f1;
@@

struct net_device_ops i = {
  .ndo_setup_tc = \(f1\|&f1\),
};

@r2 depends on before expression@
struct net_device_ops e;
struct net_device_ops *ep;
identifier f2;
@@

  <+...\(e.ndo_setup_tc\|ep->ndo_setup_tc\)...+> = \(f2\|&f2\)

@r depends on before@
identifier f,r1.f1,r2.f2;
@@

(
(
 f1
|f2
)
&
 f
)

@@
identifier r.f,i;
@@

f(...,
- u8 i
+ struct tc_to_netdev *i
 ) { ... }
```

There is one result, 16e5cc6 at 5%. Our code has many examples of uses of the tc parameter. Based on some of the examples, it seems that the following could be a reasonable transformation:

```
@r1 depends on before@
identifier i,f;
@@

struct net_device_ops i = {
```

```
  .ndo_setup_tc = f,
};

@@
identifier r1.f;
@@

f(p1,
+ u32 handle, __be proto, struct tc_to_netdev *tc_to_netdev
- u8 tc
  ) {
... when != S1
+ u8 tc;
+ if (handle != TC_H_ROOT || tc_to_netdev->type != TC_SETUP_MQPRIO)
+     return -EINVAL;
+ tc = tc_to_netdev->tc;
S2
...}
```

This is the same as the manual change except for two respects. First, the name of the `struct tc_to_netdev *` parameter is actually `ntc`, but this has no impact on the behavior. Second, the test `handle != TC_H_ROOT` was found to be correct and was dropped in multiple files in a later patch (5eb4dce3b347).

== failure - multichange