ABSTRACTION LAYER
FOR IMPLEMENTATION OF EXTENSIONS
IN PROGRAMMABLE NETWORKS

The ALIEN Cookbook

Whitepaper

Build your own device

This white paper adds information over the ALIEN deliverables D2.3 and D3.2 that is intended to help anyone willing to port the eXtensible DataPath Daemon (xdpd) to a new hardware platform. The document contains two chapters: The first describes how to create a new platform driver from scratch and how to upgrade existing ones to new versions of xdpd. The second chapter describes how the new platform can be managed using the newly developed netconf interface to xdpd. The document adds a yang model for xdpd and some references.

Authors:
Tobias Jungel, Marc Sune, Hagen Woesner (BISDN GmbH, Berlin)

1. Platform drivers

1.1 Introduction

Platform drivers encapsulate the device (platform) specific code. xDPd [1] drivers may control hardware fast path devices (e.g. ASICs), control other software components that do the packet processing, or implement software fast path elements which process packets (e.g. user-space GNU/Linux or GNU/Linux DPDK).

1.2 How to port your platform

Disclaimer: basic autotools knowledge is assumed in this short tutorial

1) Check first the introductory slides of xDPd's architecture, ROFL-pipeline and ROFL-hal doxygen documentation.

2) Decide a code name for the new platform.
   The platform code name must use '-' as a separator xxx-yyy-zzz. Try to use a short code name. For our tutorial we will choose 'my-asic'.

3) Copy the example folder to 'my_asic/' (NOTE that '_' must be used instead of '-' in the folder name)

```
sh# cp src/xdpd/drivers/example src/xdpd/drivers/my_asic/ -R
```

4) Edit README and put the relevant information.

5) Edit `configure.ac` and change the name of the package accordingly.

6) Edit all `Makefile.am` and substitute 'example' for 'my_asic'. Example of substitution in vim:

```
%s/example/my_asic/c
```

7) Now add the platform to the list of available platforms in `config/hw.m4`. The diff should look like:

```
diff --git a/config/hw.m4 b/config/hw.m4
index 93881ae..43fc18b 100644
--- a/config/hw.m4
+++ b/config/hw.m4
@@ -11,6 +11,7 @@ AC_ARG_WITH(hw-support, AS_HELP_STRING([--with-hw-
support="driver-name"],[Compil
                [bcm: compile Broadcom hardware support]
                [octeon: compile OCTEON hardware support]
                [netfpga10g: compile NetFPGA 10G hardware code (HW code not
included)]
+                               [my-asic: Tutorial ASIC driver]
                [example: compile the example forwarding module code]
 ,
  [
@@ -52,6 +53,12 @@ if( test "$HW" = "netfpga10g");then
     PLATFORM=netfpga10g
     AC_CONFIG_SUBDIRS([src/xdpd/drivers/netfpga10g])
 fi
+if( test "$HW" = "my-asic");then
+       msg="$msg MY ASIC"
+       AC_DEFINE(HW_MY_ASIC)
+       PLATFORM=my_asic
```

```
+       AC_CONFIG_SUBDIRS([src/xdpd/drivers/my_asic])
+fi
 if( test "$HW" = "example");then
    msg="$msg Example platform"
    AC_DEFINE(HW_EXAMPLE)
```

8) Run `autogen.sh`. At this point the code should compile. Remember that to compile a hw platform different than the default one, you must specify --with-hw-support='my-asic' (Note '-'):

```
sh# bash autogen.sh
sh# cd build
sh# ../configure --enable-debug --disable-silent-rules --with-hw-support="my-asic"
sh# make
```

9) Now add the necessary functionality behind the HAL calls. For that, you can take as a reference:
   - gnu-linux/gnu-linux-dpdk: software switch
   - netfpga10g: ASIC port

If you are using the software packet processing API of ROFL-pipeline (as a software switch or for Packet_outs) you must add the necessary code under `my_asic/pipeline-imp/packet.c`. You can reuse the packet classifier and other components from the reference gnu-linux platform driver. Take a look at the rest of drivers to see how this is leveraged.

10) If you wish, contribute back your platform port so that the community can benefit from your work.

1.3 How to port your driver from 0.4 to 0.5

Please take always GNU/Linux driver as a reference for any port to 0.5. There have been some minor changes to the HAL API

1.4 Adapting to the new HAL API

The following functions have been removed:

```
-hal_result_t hal_driver_of1x_fetch_group_table(uint64_t dpid, of1x_group_table_t *group_table);

-of1x_stats_group_msg_t* hal_driver_of1x_get_group_all_stats(uint64_t dpid, uint32_t id);
```

…and have been replaced by this one:

```
+of1x_stats_group_desc_msg_t *hal_driver_of1x_get_group_desc_stats(uint64_t dpid);
```

1.5 Adapting to the new ROFL-pipeline behavior for capabilities

The ROFL-pipeline now by default marks ALL supported capabilities by the table matching algorithm in the table configuration (matches & wildcards). The driver MUST unset the ones not supported in `platform_post_init_of1x_switch()`, like this:

```
for(i=0; i<sw->pipeline.num_of_tables; i++)
{
of1x_flow_table_config_t *config = &(sw->pipeline.tables[i].config);
of1x_group_table_config_t *group_config = &(sw->pipeline.groups->config);
/*
* Lets set to zero the unsupported matches and actions.
/*

/Matches
bitmap128_unset(&config->match, OF1X_MATCH_IPV6_EXTHDR);
bitmap128_unset(&config->wildcards, OF1X_MATCH_IPV6_EXTHDR);
```

From that point on, the built-in flowmod and groupmod validation will reject any flowmod which is not supported by the table.

2. NETCONF and OF-Config for xDPd

2.1 NETCONF using libnetconf and netopeer

libnetconf [2] implements client and server functionality for the NETCONF protocol [3] in C. Therefore it provides NETCONF message parsing and their transport via SSH. In addition the library provides an implementation to store and work with the configuration data in a datastore. Libnetconf defines the transaction API (transAPI), which allows to define callbacks on configuration changes. To implement these callbacks the lnctool is provided. The module itself can be loaded from a NETCONF server implementing the transAPI.
The Netopeer project [4] implements a server using the transAPI for the modules. To manage transAPI modules of the netopeer-server the tool netopeer-manager is provided. SSH communication to the server is enabled by the netopeer-agent, that operates an an SSH subsystem and passing incoming NETCONF messages to the netopeer-server. In addition there is netopeer-configurator to configure the server (e.g. certificates).

2.1.3 Installation Libnetconf

```
git clone https://code.google.com/p/libnetconf/
cd libnetconf && ./configure && make && make install
```

2.1.4 Installation Netopeer

```
git clone https://code.google.com/p/netopeer/
cd netopeer/server && ./configure --disable-dbus && make && make install
```

The enabled switch --disable-dbus is not mandatory. By disabling D-Bus with this switch the communication between the netopeer-agent and the netopeer-server is defaulting to UNIX sockets.
Modify the file /etc/ssh/sshd_config and add the following:

```
Port      830
Subsystem netconf     /usr/local/bin/netopeer-agent
```

Start the NETCONF server:

```
sudo /usr/local/bin/netopeer-server
```

2.2 OF-Config for xDPd

OF-Config [5] for xDPd is implemented as a transAPI module. To generate a transAPI module for netopeer libnetconf provides the lnctool, which will generate a C stub for the module. The lnctool requires the yang model and a set of paths. The latter is a file of XPath expressions, which match the corresponding yin XML tree. XPath expressions that occur in the paths-file are transformed to the callbacks for the transAPI module. Besides the C stub a template Makefile and the necessary files for validation are created. The following call will create all the files:

```
lnctool --model of-config1.1.1.yang transapi --paths ./of-config1.1.1.paths
```

Once the module is build, it can be attached to the NETCONF server using the following command:

```
sudo /usr/local/bin/netopeer-manager add --name of-config1.1.1 \
--model /PATH/TO/of-config1.1.1.yin --transapi /PATH/TO/of-config1.1.1.so \
--datastore /var/lib/libnetconf/ofc_datastore.xml
```

The communication between the module and xDPd is then using xDPd's management protocol XMP. Since not all items of OF-Config can be configured by XMP just a subset of OF-Config is currently implemented. The subset is reflected in the of-config1.1.1.paths file.
OF-Config 1.1.1 currently does not support the creation of logical switch instances. Since xDPd is capable of doing so, we have changed the model of OF-Config to make this possible.

2.3 Extension for xDPd

OF-Config 1.1.1 does as well not reflect the connections between logical switch instances. Therefore an additional yang module was created to augment the OF-Config module.
The new module has to be part of the main module and can be created using lnctool like before:

```
lnctool --model of-config1.1.1.yang --augment-models xdpd-mgmt.yang --output-
dir . transapi --paths ./of-config1.1.1.paths
```

Additionally the augment module has to be added to the netopeer-server using:

```
sudo /usr/local/bin/netopeer-manager add --name of-config1.1.1 --augment
/PATH/TO/xdpd-mgmt.yin
```

Eventually, the netopeer-server needs a restart.

2.4 Source Code

The current repository is located at git://git.bisdn.de/xdpd-netconf-module.git. It will be soon merged into the github repositories of BISDN and licensed as MPL.

3. Appendix

3.1 xdpd-mgmt.yang

```
module xdpd-mgmt {
  namespace "urn:xdpd:mgmt:yang";
  prefix xdpd-mgmt;
  import of-config1.1.1  { prefix ofc; }
  organization "BISDN GmbH";
  contact "info@bisdn.de";
 description "";
  revision 2014-10-01 {
    description "First Draft";
    reference "nn";
  }
 augment "/ofc:capable-switch" {
    container cross-connections {
      description "";
      presence "";
      list cross-connection {
        description "";
        key "name";
        leaf name {
          type string;
        }
        list switch {
          description "";
          key "id";
          min-elements 2;
          max-elements 2;
          leaf id {
            type leafref {
              path "/ofc:capable-switch/ofc:logicalswitches/ofc:switch/ \
                                                                  ofc:id";
            }
          }
          leaf portname {
            config false;
            type string;
          }
        }
      }
    }
  }
}
```

4. References

[1] BISDN. [Online]. Available: http://www.xdpd.org/.

[2] CESNET. [Online]. Available: https://code.google.com/p/libnetconf/.

[3] “rfc6241,” [Online]. Available: https://tools.ietf.org/html/rfc6241.

[4] CESNET. [Online]. Available: https://code.google.com/p/netopeer/.

[5] ONF, “OF-Config,” [Online]. Available: https://www.opennetworking.org/sdn-resources/onf-specifications/openflow-config.