LINCX - OF-DPA integration pilot

Maxim Kharchenko, Cloudozer LLP

13/06/2014

1 Overview

The document describes the architecture of the pilot implementation of the integration of the LINCX switch and OF-DPA layer. The notes regarding the current status of the project are interspersed throughout the document.

2 Constraints

LINCX requires Xen as a virtualization platform. It runs inside an unpriviledged user Xen domain (DomU). By default, unpriviledged Xen domains can not talk to hardware directly. OF-DPA should be able to access the custom ASIC and thus is limited to the priviledged Xen domain (Dom0). Due to this limitation LINCX and OF-DPA are integrated using RPC over a TCP connection.

3 Architecture

The project adds a new 'backend' to LINCX - linc_ofdpa. The backend responds to the controller messages. Currently all controller messages result in an error message sent back to the controller. See src/linc_ofdpa.erl. All paths in the document are relative to lincx/apps/linc_ofdpa.

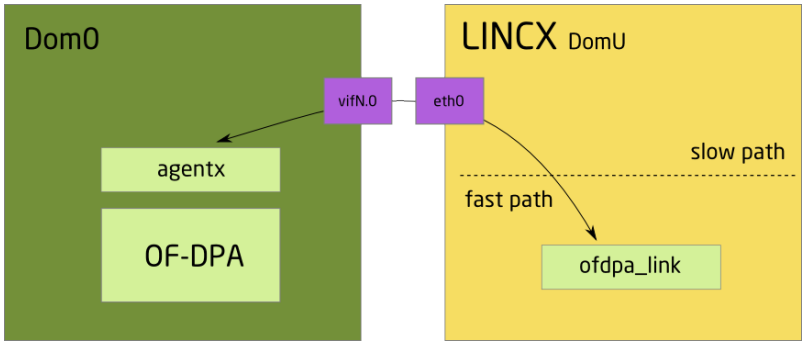

Figure 1: LINCX - OFDPA

Agentx is a simple server that accepts TCP connections from the LINCX switch and invokes the OF-DPA API functions according to the switch messages. Agentx must be linked

with the OF-DPA library to avoid yet another layer of RPC. Currently agentx is linked to a sham library. See agentx/ofdpa_lib.c.

In the LINCX domain all OF-DPA API entries are mapped to functions in the ofdpa module. See src/ofdpa.erl. For instance, ofdpaPortQueueStatsGet() maps to ofdpa:ofdpaPortQueueStats(). Stubs in the ofdpa module convert function arguments, pack them, and pass them to agentx using using the connection maintained by ofdpa_link process. They also unpack and convert back return values including parameters passed by reference.

A sample session:

```
dom0$ agentx
Agentx listens on 5005
linc_ofdpa connection accepted
Invoking function [161], cookie [93fe4bf7] arg_len [32]
Function invocation complete, ret_len [32]

--------------------------------------------------------------------------------

1> rr("priv/ofdpa.hrl").
[bridging_flow_entry,bridging_flow_match,flow_entry,
 flow_entry_stats,flow_event,flow_table_info,
 group_bucket_entry,group_entry,group_entry_stats,
 group_table_info,ingress_port_flow_entry,
 ingress_port_flow_match,l_2_interface_group_bucket_data,
 l_2_overlay_group_bucket_data,l_2_rewrite_group_bucket_data,
 l_3_interface_group_bucket_data,
 l_3_unicast_group_bucket_data,multicast_routing_flow_entry,
 multicast_routing_flow_match,packet,policy_acl_flow_entry,
 policy_acl_flow_match,port_event,port_feature,
 port_queue_stats,port_stats,termination_mac_flow_entry,
 termination_mac_flow_match,unicast_routing_flow_entry|...]
2>
2> ofdpa_link:start_link("192.168.0.1", 5005).
{ok,<0.826.0>}
3>
3> ofdpa:ofdpaQueueStatsGet(1, 2, #port_queue_stats{}).
{ok,[e_none,
     #port_queue_stats{txBytes = 1234,txPkts = 5678,
                       duration_seconds = 987654321}]}
4>
```

On the LINCX (Erlang) side enum values are represented as atoms. They are converted automatically to integer values and back. C structures are represented as (potentially nested) Erlang records. See include/ofdpa.hrl.

4 Protocol

An invocation of an OF-DPA API function starts with the CALL message originating from LINCX:

Table 1: CALL message

Field	Size	Notes
len	4	message length (without len field)
tag	2	= 0xca11
what	2	API function index
cookie	4	random (for testing)
args	-	call arguments

Agentx responds with the RETURN message:

Table 2: RETURN message

Field	Size	Notes
len	4	message length (without len field)
tag	2	= 0xbac6
what	2	API function index
cookie	4	random (for testing)
ret_vals	-	returned values

The API calls are synchronous. There is no more than one outstanding CALL request.

Arguments (and returned values) are interpreted depending on the function being called (returned from). Struct values are prefixed with a 32-bit length field. Scalar values, such as 32-bit unsigned integers, are packed without a prefix.

All integer values use little-endian representation.

5 Events

The file descriptors returned by certain OF-DPA API calls have no meaning for LINCX. All events originating from the OF-DPA layer should be captured by agentx. Agentx should then convert these events to asynchonous messages to ofdpa_link. Events are not currently implemented.

6 Generator

Portions of code of the integration layers are generated automatically from ofdpa_api.h and ofdpa_datatypes.h header files using a utility called genera. See genera directory. Run 'genera help' to get the (partial) list of possible code snippets the utility can emit.

Currently genera only emits Erlang code. It should be extended to generate code for packing/unpacking function arguments on the agentx side.

7 Alternative

The alternative to the approach selected for the pilot integration is to import the entire OF-DPA to LINCX. This will require PCI passthrough to allow the access to the ASIC for the LINCX DomU. This is the path that can be taken if the pilot shows promising results.