Link Aggregation Configuration and Usage in Open vSwitch\* with DPDK

This article explains the link aggregation feature of Data Plane Development Kit (DPDK) ports on Open vSwitch\* (OVS), and shows how to configure them. Link aggregation can be used for high availability, traffic load balancing and extending the link capacity using multiple links/ports. Link aggregation combines multiple network connections in parallel in order to increase the throughput beyond what a single connection could sustain, and to provide redundancy in case one of the links should fail. Link aggregation support for OVS-DPDK is available in OVS 2.4 and later.

Test Environment

---

**Figure 1: OVS-DPDK link aggregation test setup**

The test setup uses two hypervisors (physical host machines) both running OVS 2.6 with DPDK 16.7 and QEMU 2.6. The VMs (VM1 and VM2, respectively) running on each hypervisor are connected to a bridge named br0. The two hypervisors are connected to each other using an aggregated link consisting of two physical interfaces named dpdk0 and dpdk1. The member ports (dpdk0, dpdk1) on each host must have the same link properties, such as speed and bandwidth, to form an aggregated link. However, it is not necessary
that the port names should be the same on both hosts to form an aggregated link. The VMs in each hypervisor can reach each other via the aggregated ports between the host machines.

**Link Aggregation in OVS with DPDK**

At the time of writing, OVS considers each member port in an aggregated port as an independent OpenFlow™ port. When a user issues the following command to see the available OpenFlow ports in OVS-DPDK, the member ports are displayed separately, without any bond interface information.

- `ovs-ofctl show br0`

This makes it impossible to program OpenFlow rules on bond ports, and also limits the OVS to operate only in the NORMAL action mode. In the NORMAL action mode, OVS operates like a traditional MAC learning switch.

The following link aggregation modes are supported in OVS with DPDK.

**Active-Backup (Active/Standby)**

Active/standby failover mode is where one of the ports in the link aggregation port is active and all others are in standby mode. One MAC address (MAC address of the active link) is used as the MAC address of the aggregated link.

Note: No traffic load balancing is offered in this mode.

**balance-slb**

Load balance the traffic based on source MAC and VLAN. This mode uses a simple hashing algorithm on source MAC and VLAN to choose the port in an aggregated link to forward the traffic. This mode is a simple static link aggregation similar to the mode-2 bonds in Linux® bonding driver [1].

**balance-tcp**

The preferred load-balancing mode. It uses 5-tuple (source and destination IP, source and destination port, protocol) to balance traffic across the ports in an aggregated link. This mode is similar to mode-4 bonds in
Linux bonding driver\(^1\). It uses Link Aggregation Control Protocol (LACP)\(^2\) (for signaling/controlling the link aggregation between switches. LACP offers high resiliency for link failure detection and additional diagnostic information about the bond. It is observed that the balance-tcp is less performant due to the overhead of hashing on more header fields when compared to the balance-slb.

### Link Aggregation Configuration and Testing

The test setup uses two identical host machines with the following configuration:

**Hardware:** Intel® Xeon® processor E5-2695 V3 product family, Intel® Server Board S2600WT2, and Intel® 82599ES 10-G SFI/SFP+ (rev 01) NIC.

**Software:** Ubuntu* 16.04, Kernel version 4.2.0-42-generic, OVS 2.6, DPDK 16.07, and QEMU2.6.

To test the configuration, make sure iPerf* is installed on both VMs. iPerf can be run in client mode or server mode.

To set up the link aggregation, run the following commands on each hypervisor (physical host 1 and physical host 2):

- Create a bridge named br0 of type netdev:
  
  ```bash
  ovs-vsctl add-br br0 -- set bridge br0 datapath_type=netdev
  ```

- Create a link aggregation port using the DPDK physical ports dpdk0 and dpdk1:
  
  ```bash
  ovs-vsctl add-bond br0 dpdkbond1 dpdk0 dpdk1 -- set Interface dpdk0 type=dpdk -- set Interface dpdk1 type=dpdk
  ```

- Add the vhost-user port of the VM to the bridge:
  
  ```bash
  ovs-vsctl add-port br0 vhost-user1 -- set Interface vhost-user1 type=dpdkvhostuser
  ```

- Delete all the flows and set the bridge to NORMAL forwarding mode:
  
  ```bash
  ovs-ofctl del-flows br0
  ovs-ofctl add-flow br0 actions=NORMAL
  ```

- Start the VM on each hypervisor using the vhost-user interface vhost-user1.
Active-Backup

- The default mode of link aggregation (bond) interface is active-backup. To set the mode explicitly:
  - `ovs-vsctl set port dpdkbond1 bond_mode=active-backup`
- Verify the bond interface configuration by using:
  - `ovs-appctl bond/show`
- Assign Ip address to the VNIC interfaces of VM1 and VM2:
  - `ip addr flush eth0`
  - `ip addr add <ip_addr> dev eth0`
    In this example, 10.0.0.1/24 and 10.0.0.5/24 are the `<ip_addr>` for VM1 and VM2, respectively.
  - `ip link set dev eth0 up`
- Run iPerf server on VM1 in UDP mode at port 8080:
  - `iperf -s -p 8080`
- Run iPerf client on VM2 in UDP mode at port 8080:
  - `iperf -c 10.0.0.1 -p 8080`

After 10 seconds the client shows a series of results for traffic between VM1 and VM2, which is similar to the following, Figure 2, though the numbers may vary.

```
[root@ovdk_guest ~]# iperf -c 10.0.0.1 -p 8080
-------------------------------------------------------------
Client connecting to 10.0.0.1, TCP port 8080
TCP window size: 22.9 KByte (default)
-------------------------------------------------------------
[  3] local 10.0.0.5 port 56289 connected with 10.0.0.1 port 8080
[ ID] Interval      Transfer      Bandwidth
[  3]   0.0-10.0 sec  1.89 GBytes  1.62 Gbits/sec
[root@ovdk_guest ~]#
```

**Figure 2: Screenshot of iPerf client on VM2 in Active-Backup mode**

Only the active port in the bond interface is used for traffic forwarding. In this example, the stats of dpdk1 (port: 2, active port) shows all traffic is on dpdk1. A small number of packets on port dpdk0 (port: 1) are
related to link negotiation. The OpenFlow port numbers assigned to ports dpdk0 and dpdk1 are port: 1 and port: 2, respectively.

```
$ sudo ./utilities/ovs-ofctl show br0
DPPT FEATURES_REPLY (xid=0x2): dpid:00000000000000100
n_tables:254, n_buffers:0
Capabilities: FLOW_STATS TABLE_STATS PORT_STATS QUEUE_STATS ARP_MATCH_IP
Actions: output enqueue set vlan vid set vlan pcp strip_vlan pcp strip_vlan pcp drop
1(dpdk0): addr:00:00:00:00:01:00
  config: 0
  state: 0
  current: 10GB-FD AUTO_NEG
  advertised: 10MB-FD 100MB-FD 100MB-FD 1GB-HD 10GB-FD AUTO_PAUSE ASYM
  supported: 10MB-HD 10MB-FD 1GB-HD COPPER FIBER AUTO_NEG AUTO_PAUSE
  peer: 1GB-HD AUTO_NEG AUTO_PAUSE
  speed: 10000 Mbps now, 10000 Mbps max
2(dpdk1): addr:00:00:00:00:01:01
  config: 0
  state: 0
  current: 10GB-FD AUTO_NEG
  advertised: 10MB-FD 100MB-FD 100MB-FD 1GB-HD 10GB-FD AUTO_PAUSE ASYM
  supported: 10MB-HD 10MB-FD 1GB-HD COPPER FIBER AUTO_NEG AUTO_PAUSE
  peer: 1GB-HD AUTO_NEG AUTO_PAUSE
  speed: 10000 Mbps now, 10000 Mbps max
```

**Figure 3:** OpenFlow port statistics on physical host-1 in Active-Backup mode

**balance-slb**

- Set the link aggregation mode to balance-slb:
  - `ovs-vsctl set port dpdkbond1 bond_mode=balance-slb`
- Verify the bond interface configuration by using:
  - `ovs-appctl bond/show`
- Create two VLAN logical interfaces on the VNIC port of each VM; the balance-slb load balances the traffic based on the VLAN and source MAC address:
  - `ip link add link eth0 name eth0.10 type vlan id 10`
  - `ip link add link eth0 name eth0.20 type vlan id 20`
- Assign IP address to the VNIC interfaces of VM1 and VM2:
  - `ip addr flush eth0`
- `ip addr flush eth0.10`
- `ip addr add <ip-addr> dev eth0.10`
  10.0.0.1/24 and 10.0.0.5/24 are the <ip-addr> for VM1 and VM2, respectively, for the logical interface eth0.10.
- `ip addr flush eth0.20`
- `ip addr add <ip-addr> dev eth0.20`
  20.0.0.1/24 and 20.0.0.5/24 are the <ip-addr> for VM1 and VM2, respectively, for the logical interface eth0.20.
- `ip link set dev eth0.10 up`
- `ip link set dev eth0.20 up`
- Run iPerf server on VM2 in UDP mode at port 8080:
  - `iperf –s –u –p 8080`
- Run two iPerf client streams on VM1 in UDP mode at port 8080:
  - `iperf -c 10.0.0.5 -u -p 8080 –b 1G`
  - `iperf -c 20.0.0.5 -u -p 8080 –b 1G`

In this example each stream uses a separate port in the bond interface for the traffic. The port stats show the same.
Set link aggregation mode to balance-tcp and enable LACP:

- `ovs-vsctl set port dpdkbond1 bond_mode=balance-tcp`
- `ovs-vsctl set port dpdkbond1 lacp=active`

Disabling LACP will fall back the balance-tcp bond interface to the default mode (active-standby). To disable LACP on bond interface:

- `ovs-vsctl set port dpdkbond1 lacp=passive`

Verify the bond interface configuration by using:

- `ovs-appctl bond/show`

Assign Ip address to the VNIC interfaces of VM1 and VM2:

- `ip addr flush eth0`
- `ip addr add <ip-address> dev eth0`

---

**Figure 4:** OpenFlow port statistics at physical host-1 in balance-slb mode
In this example, 10.0.0.1/24 and 10.0.0.5/24 are the <ip-addr> for VM1 and VM2, respectively.

- `ip link set dev eth0 up`

- Run iPerf server instance on VM2 in TCP mode at port 9000; logical interfaces are not needed here as load balancing is performed on layer-4:
  - `iperf -s -p 9000`

- Run two iPerf client instances on VM1 in TCP mode:
  - `iperf -c 10.0.0.5 -p 9000`
  - `iperf -c 10.0.0.5 -p 9000`

The two independent TCP streams are load balanced between two ports in the bond interface as the iPerf client uses different source ports for each stream.

![Screenshot of iPerf server on VM2 in balance-tcp mode.](image)

**Figure 5:** Screenshot of iPerf server on VM2 in balance-tcp mode.

The statistics of bond member ports (highlighted in Figure 6) show that the streams are balanced between the ports.
Additional Configuration and Display Options for Link Aggregation

- Setting LACP mode to passive/off:
  - `ovs-vsctl set port dpdkbond1 lacp=passive`
  - `ovs-vsctl set port dpdkbond1 lacp=off`

- Setting LACP behavior to switch to bond_mode=active_backup as a fallback:
  - `ovs-vsctl set port dpdkbond1 other_config:lacp-fallback-ab=true`

- Setting LACP negotiation time interval either fast (30 ms) or slow (1 second); the default is slow:
  - `ovs-vsctl set port dpdkbond1 other_config:lacp-time=fast`
  - `ovs-vsctl set port dpdbond1 other_config:lacp-time=slow`

- The number of milliseconds for a port to be up before activating in the bond interface:
  - `ovs-vsctl set port dpdkbond1 other_config:bond_updelay=1000`

- The time interval in milliseconds to rebalance the flow between bond member ports; set to zero to disable:
• `ovs-vsctl set port dpdkbond1 other_config:bond-rebalance-interval=10000`

• To display the bond interface configuration details:
  • `ovs-appctl bond/show`
  • `ovs-appctl bond/show dpdkbond0`

The following bond interface information is displayed for the given test setup in a balance-tcp mode.

![bond show output]

**Figure 7**: ‘bond show’ on physical host in balance-tcp mode

**Summary**

Link aggregation is a useful method for combining multiple links to form a single (aggregated) link. The main features of link aggregation are:

- Increased link capacity/bandwidth
- Traffic load balancing
- High availability (automatic failover in the event of link failure)

OVS-DPDK offers three modes of link aggregation:

- **Active/Standby (Active-Backup)**
  No load balancing is offered in this mode and only one of the member ports is active/used at a time.
- **balance-slb**
  Considered as a static load-balancing mode. Traffic is load balanced between member ports based on the source MAC and VLAN.
- **balance-tcp**
This is the preferred bonding mode. It offers traffic load balancing based on 5-tuple header fields. LACP must be enabled at both endpoints to use this mode. The aggregate link will fall back to default mode (active-passive) in the event of LACP negotiation failure.

About the Author

Sugesh Chandran is a network software engineer with Intel. His work is primarily focused on accelerated software switching solutions in the user space running on Intel® architecture. His contributions to Open vSwitch with DPDK include tunneling acceleration and enabling hardware acceleration in OVS-DPDK.

Additional Information

1. Link Aggregation Wiki
2. Link Aggregation Control Protocol Wiki
3. Linux Bonding Documentation
4. Open vSwitch Documentation and Manpages
5. Open vSwitch with DPDK Installation guide

Notices

No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document.

Intel disclaims all express and implied warranties, including without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement, as well as any warranty arising from course of performance, course of dealing, or usage in trade.

This document contains information on products, services and/or processes in development. All information provided here is subject to change without notice. Contact your Intel representative to obtain the latest forecast, schedule, specifications and roadmaps.

The products and services described may contain defects or errors known as errata which may cause deviations from published specifications. Current characterized errata are available on request.

Copies of documents which have an order number and are referenced in this document may be obtained by calling 1-800-548-4725 or by visiting www.intel.com/design/literature.htm.

This sample source code is released under the Intel Sample Source Code License Agreement.

Intel, the Intel logo and Xeon are trademarks of Intel Corporation in the U.S. and/or other countries.

*Other names and brands may be claimed as the property of others.

© 2016 Intel Corporation