Devlink Port

`devlink-port` is a port that exists on the device. It has a logically separate ingress/egress point of the device. A devlink port can be any one of many flavours. A devlink port flavour along with port attributes describe what a port represents.

A device driver that intends to publish a devlink port sets the devlink port attributes and registers the devlink port.

Devlink port flavours are described below.

List of devlink port flavours

Flavour	Description
`DEVLINK_PORT_FLAVOUR_PHYSICAL`	Any kind of physical port. This can be an eswitch physical port or any other physical port on the device.
`DEVLINK_PORT_FLAVOUR_DSA`	This indicates a DSA interconnect port.
`DEVLINK_PORT_FLAVOUR_CPU`	This indicates a CPU port applicable only to DSA.
`DEVLINK_PORT_FLAVOUR_PCI_PF`	This indicates an eswitch port representing a port of PCI physical function (PF).
`DEVLINK_PORT_FLAVOUR_PCI_VF`	This indicates an eswitch port representing a port of PCI virtual function (VF).
`DEVLINK_PORT_FLAVOUR_PCI_SF`	This indicates an eswitch port representing a port of PCI subfunction (SF).
`DEVLINK_PORT_FLAVOUR_VIRTUAL`	This indicates a virtual port for the PCI virtual function.

Devlink port can have a different type based on the link layer described below.

List of devlink port types

Type	Description
`DEVLINK_PORT_TYPE_ETH`	Driver should set this port type when a link layer of the port is Ethernet.
`DEVLINK_PORT_TYPE_IB`	Driver should set this port type when a link layer of the port is InfiniBand.
`DEVLINK_PORT_TYPE_AUTO`	This type is indicated by the user when driver should detect the port type automatically.

PCI controllers

In most cases a PCI device has only one controller. A controller consists of potentially multiple physical, virtual functions and subfunctions. A function consists of one or more ports. This port is represented by the devlink eswitch port.

A PCI device connected to multiple CPUs or multiple PCI root complexes or a SmartNIC, however, may have multiple controllers. For a device with multiple controllers, each controller is distinguished by a unique controller number. An eswitch is on the PCI device which supports ports of multiple controllers.

An example view of a system with two controllers:

```
                 ---------------------------------------------------------
                 |                                                       |
                 |           --------- ---------         ------- ------- |
  -----------    |           | vf(s) | | sf(s) |         |vf(s)| |sf(s)| |
  | server  |    | -------   ----/---- ---/----- ------- ---/--- ---/--- |
  | pci rc  |=== | pf0 |______/________/       | pf1 |___/_______/     |
  | connect |    | -------                       -------                 |
  -----------    |     | controller_num=1 (no eswitch)                   |
                 ------|--------------------------------------------------
                 (internal wire)
                       |
                 ---------------------------------------------------------
                 | devlink eswitch ports and reps                        |
                 | ----------------------------------------------------- |
                 | |ctrl-0 | ctrl-0 | ctrl-0 | ctrl-0 | ctrl-0 |ctrl-0 | |
                 | |pf0    | pf0vfN | pf0sfN | pf1    | pf1vfN |pf1sfN | |
                 | ----------------------------------------------------- |
                 | |ctrl-1 | ctrl-1 | ctrl-1 | ctrl-1 | ctrl-1 |ctrl-1 | |
                 | |pf0    | pf0vfN | pf0sfN | pf1    | pf1vfN |pf1sfN | |
                 | ----------------------------------------------------- |
                 |                                                       |
                 |                                                       |
  -----------    |           --------- ---------         ------- ------- |
  | smartNIC|    |           | vf(s) | | sf(s) |         |vf(s)| |sf(s)| |
  | pci rc  |==| -------   ----/---- ---/----- ------- ---/--- ---/--- |
  | connect |    | | pf0 |______/________/       | pf1 |___/_______/     |
  -----------    | -------                       -------                 |
                 |                                                       |
                 |  local controller_num=0 (eswitch)                     |
                 ---------------------------------------------------------
```

In the above example, the external controller (identified by controller number = 1) doesn't have the eswitch. Local controller (identified by controller number = 0) has the eswitch. The Devlink instance on the local controller has eswitch devlink ports for both the controllers.

Function configuration

A user can configure the function attribute before enumerating the PCI function. Usually it means, user should configure function attribute before a bus specific device for the function is created. However, when SRIOV is enabled, virtual function devices are created on the PCI bus. Hence, function attribute should be configured before binding virtual function device to the driver. For subfunctions, this means user should configure port function attribute before activating the port function.

A user may set the hardware address of the function using 'devlink port function set hw_addr' command. For Ethernet port function this means a MAC address.

Subfunction

Subfunction is a lightweight function that has a parent PCI function on which it is deployed. Subfunction is created and deployed in unit of 1. Unlike SRIOV VFs, a subfunction doesn't require its own PCI virtual function. A subfunction communicates with the hardware through the parent PCI function.

To use a subfunction, 3 steps setup sequence is followed. (1) create - create a subfunction; (2) configure - configure subfunction attributes; (3) deploy - deploy the subfunction;

Subfunction management is done using devlink port user interface. User performs setup on the subfunction management device.

(1) Create

A subfunction is created using a devlink port interface. A user adds the subfunction by adding a devlink port of subfunction flavour. The devlink kernel code calls down to subfunction management driver (devlink ops) and asks it to create a subfunction devlink port. Driver then instantiates the subfunction port and any associated objects such as health reporters and representor netdevice.

(2) Configure

A subfunction devlink port is created but it is not active yet. That means the entities are created on devlink side, the e-switch port representor is created, but the subfunction device itself is not created. A user might use e-switch port representor to do settings, putting it into bridge, adding TC rules, etc. A user might as well configure the hardware address (such as MAC address) of the subfunction while subfunction is inactive.

(3) Deploy

Once a subfunction is configured, user must activate it to use it. Upon activation, subfunction management driver asks the subfunction management device to instantiate the subfunction device on particular PCI function. A subfunction device is created on the :ref:`Documentation/driver-api/auxiliary_bus.rst <auxiliary_bus>`. At this point a matching subfunction driver binds to the subfunction's auxiliary device.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\networking\devlink\(linux-master) (Documentation) (networking) (devlink)devlink-port.rst`, line 161); *backlink*

Unknown interpreted text role "ref".

Rate object management

Devlink provides API to manage tx rates of single devlink port or a group. This is done through rate objects, which can be one of the two types:

`leaf`
: Represents a single devlink port; created/destroyed by the driver. Since leaf have 1to1 mapping to its devlink port, in user space it is referred as `pci/<bus_addr>/<port_index>`;

`node`
: Represents a group of rate objects (leafs and/or nodes); created/deleted by request from the userspace; initially empty (no rate objects added). In userspace it is referred as `pci/<bus_addr>/<node_name>`, where `node_name` can be any identifier, except decimal number, to avoid collisions with leafs.

API allows to configure following rate object's parameters:

`tx_share`
: Minimum TX rate value shared among all other rate objects, or rate objects that parts of the parent group, if it is a part of the same group.

`tx_max`
: Maximum TX rate value.

`parent`

Parent node name. Parent node rate limits are considered as additional limits to all node children limits. `tx_max` is an upper limit for children. `tx_share` is a total bandwidth distributed among children.

Driver implementations are allowed to support both or either rate object types and setting methods of their parameters.

Terms and Definitions

Terms and Definitions

Term	Definitions
`PCI device`	A physical PCI device having one or more PCI buses consists of one or more PCI controllers.
`PCI controller`	A controller consists of potentially multiple physical functions, virtual functions and subfunctions.
`Port function`	An object to manage the function of a port.
`Subfunction`	A lightweight function that has parent PCI function on which it is deployed.
`Subfunction device`	A bus device of the subfunction, usually on a auxiliary bus.
`Subfunction driver`	A device driver for the subfunction auxiliary device.
`Subfunction management device`	A PCI physical function that supports subfunction management.
`Subfunction management driver`	A device driver for PCI physical function that supports subfunction management using devlink port interface.
`Subfunction host driver`	A device driver for PCI physical function that hosts subfunction devices. In most cases it is same as subfunction management driver. When subfunction is used on external controller, subfunction management and host drivers are different.