DPDK

DATA PLANE DEVELOPMENT KIT

DPDK USERSPACE 2019, BORDEAUX

Lib1Net

OPTIMISING VNF PERFORMANCE AND DENSITY AT THE ENTERPRISE EDGE

ANTHONY FEE

PRINCIPAL NETWORK SOFTWARE ENGINEER

emutex

where software meets silicon

DPDK
DATA PLANE DEVELOPMENT KIT

Overview

- Who am I?
- Enterprise Edge Characteristics
- Lib1Net and its Benefits to VNF deployments
- Lib1Net Features
- Q&A

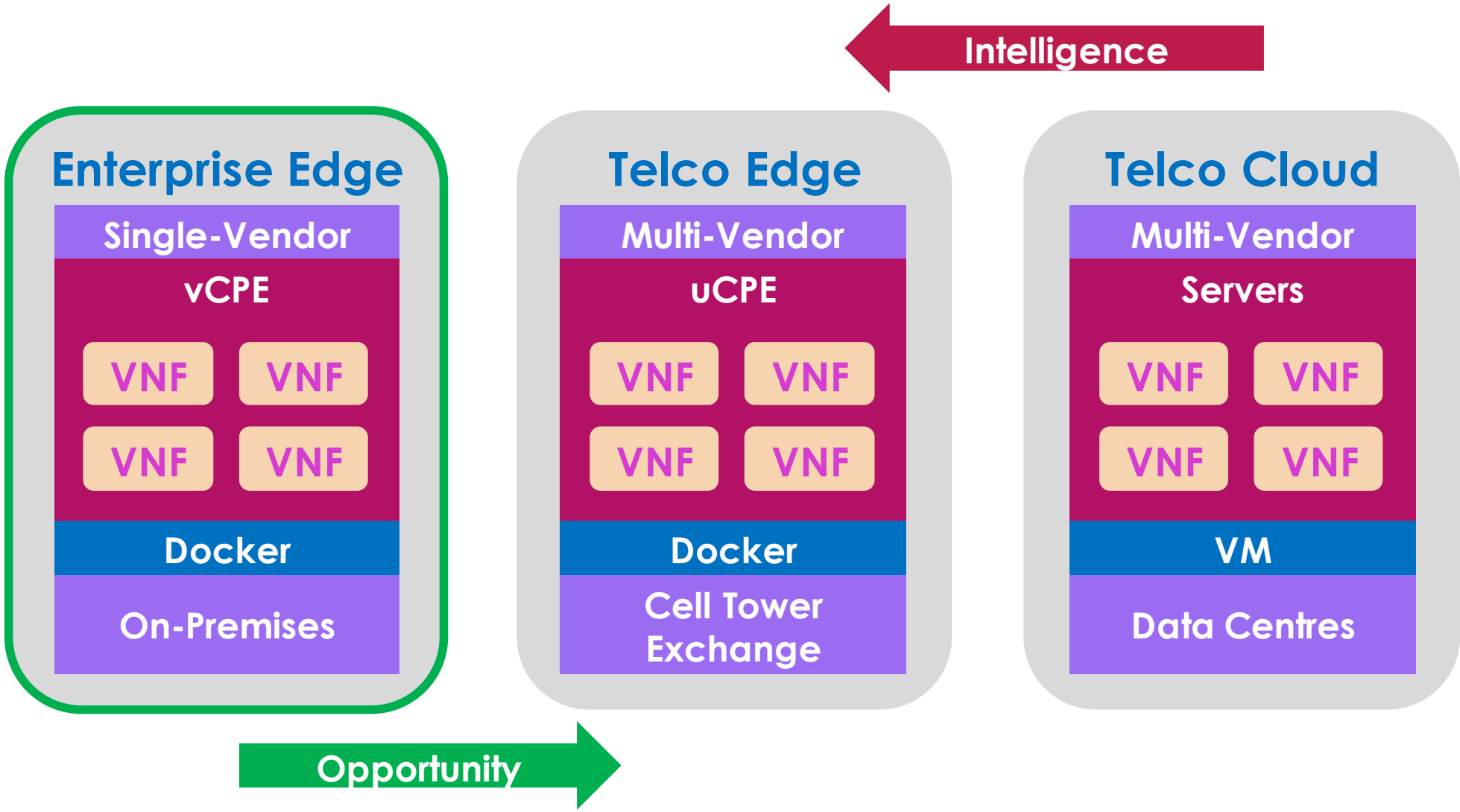
Introduction

- Anthony Fee, Principal Network Software Engineer at Emutex Madrid.

- DPDK experience:
  - Developer of high performance DPDK-based network applications and VNFs.
  - Co-author of initial vhost-user example application.
  - Integrator of vhost-user with OVS-DPDK (Open vSwitch).

- Emutex:
  - Developer of high-performance DPDK network applications on Intel architecture.
  - Developer of Linux Kernels, drivers and distributions for embedded systems.

Enterprise Edge Characteristics

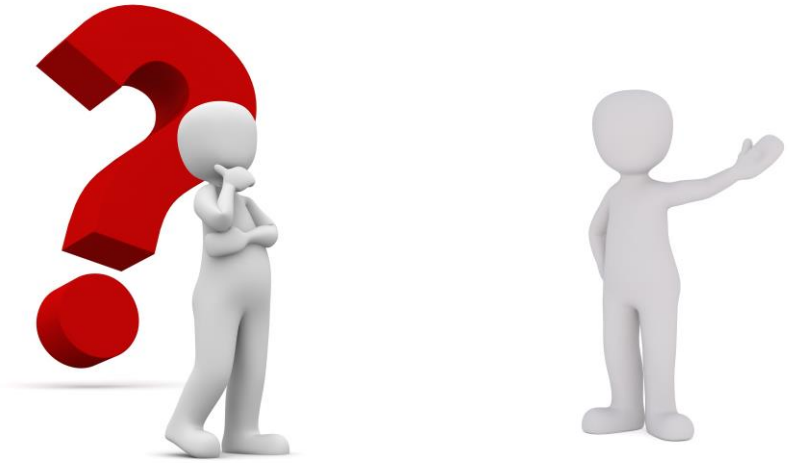


Enterprise Edge Characteristics

- Single vendor VNF solutions (e.g. Security Appliance).
- Hardware cost has significant impact on solution cost.
- Maximising network packet throughput is a challenge.
- Service chain configuration is common.
- Trending towards use of Docker containerised VNFs.

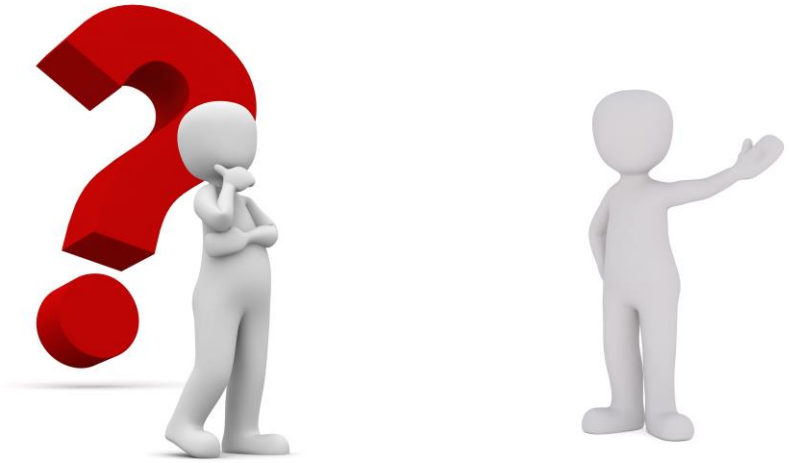
Lib1Net

How can I maximise network packet throughput and VNF density in my Enterprise Edge solution?



Lib1Net

Build your Docker containerised VNFs using **Lib1Net** and **DPDK!**



Lib1Net

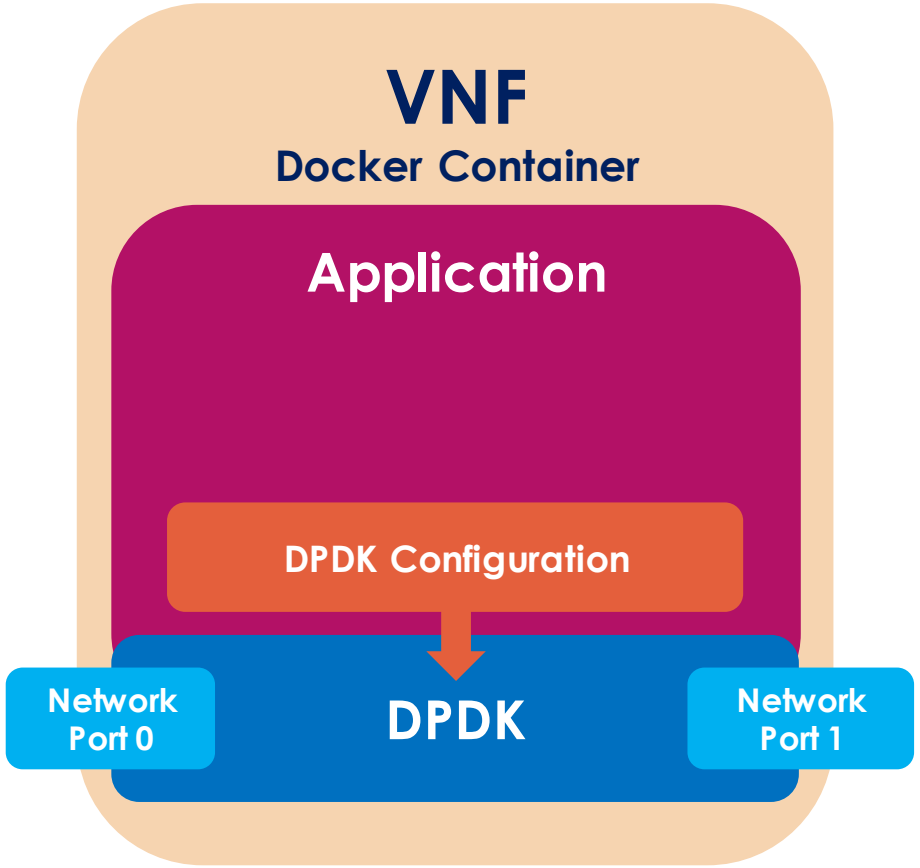
- Light-weight DPDK-based software library.
- Simplifies the integration of DPDK packet acceleration.
- Exposes a simplified API focussed on VNF requirements.
- Targets enterprise edge Docker containerised VNFs.

Lib1Net - Benefits

- Abstracts VNF applications from DPDK configuration.
- Optimises packet receive polling and transmission.
- Maximises hardware utilisation for service chaining.
- Simplifies port management through JSON configuration.
- Standardises how configuration is provided to VNFs.

Lib1Net - DPDK Configuration

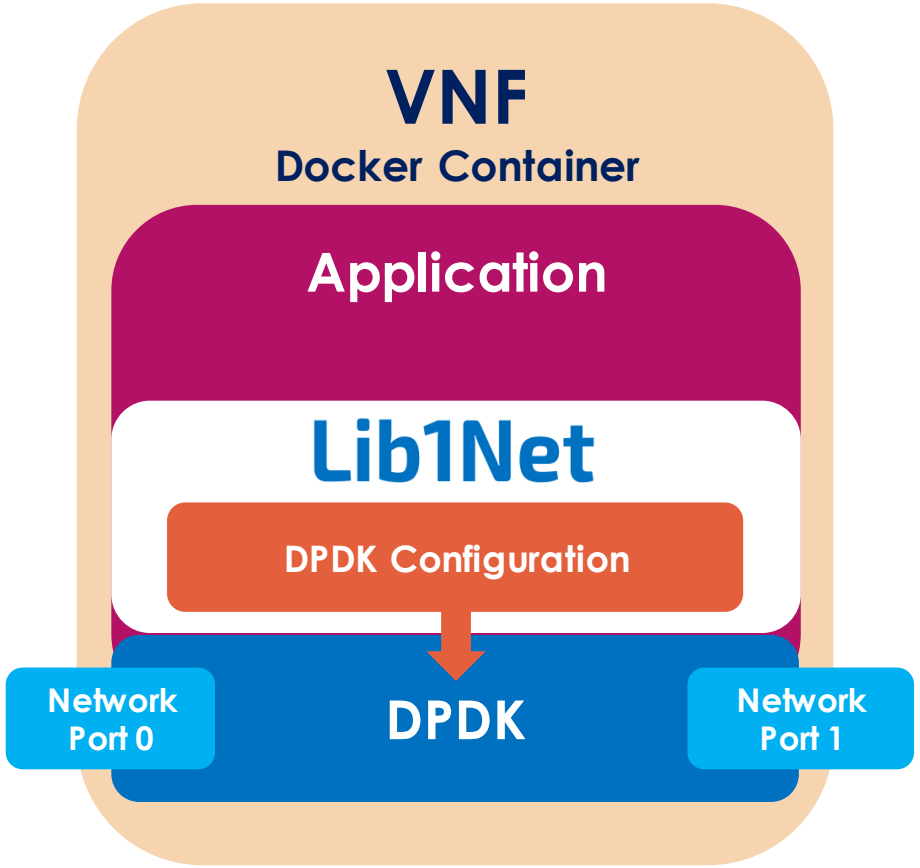
How can I configure **DPDK** to maximise the packet processing performance of my VNF?



Lib1Net - DPDK Configuration

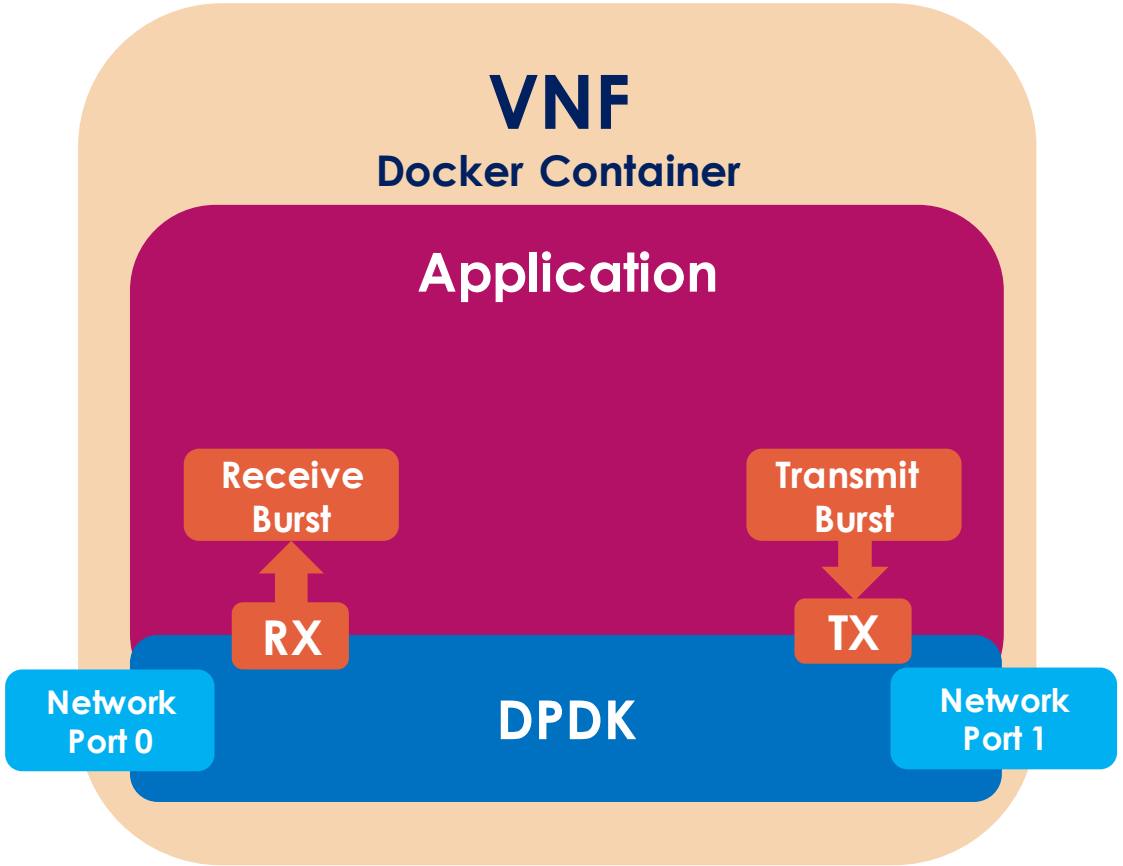
Lib1Net does this for you by:

- Configuring DPDK Memory
- Configuring DPDK Ports
- Initialising DPDK



Lib1Net - DPDK Packet Handling

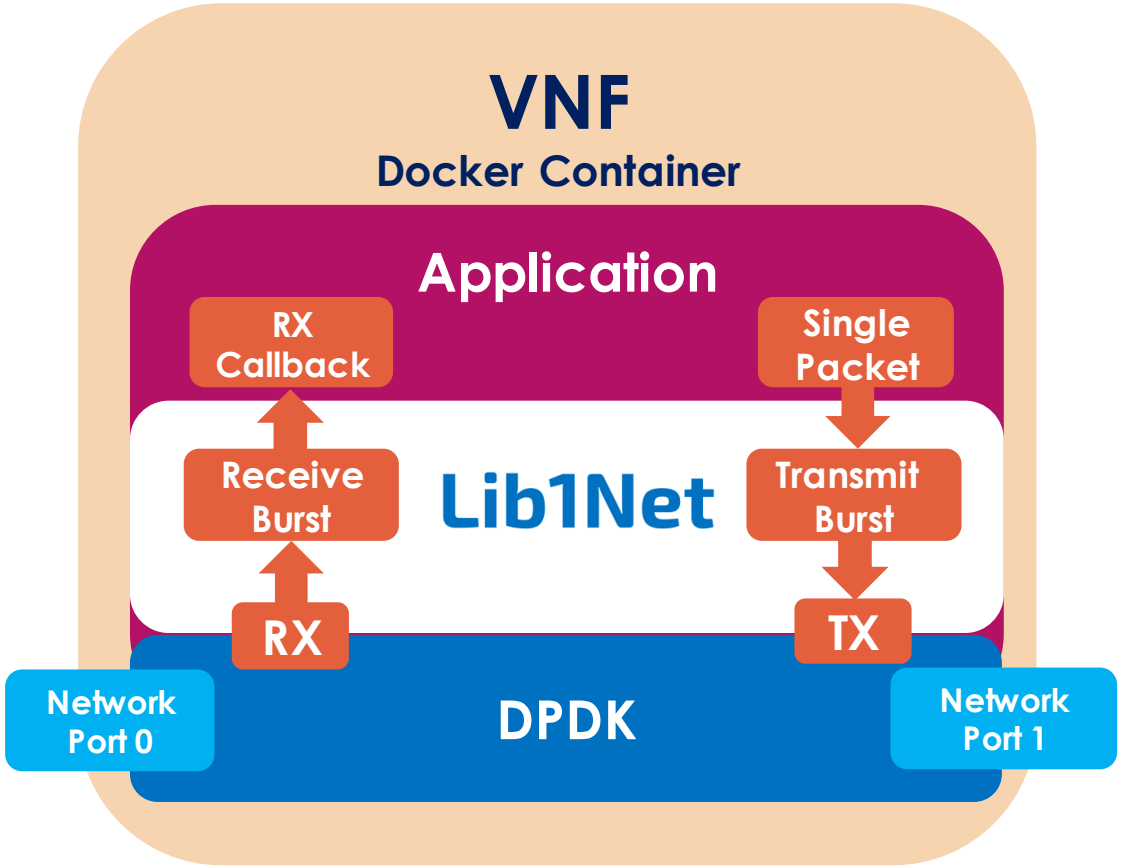
What is the most efficient method of receiving and transmitting packets with **DPDK**?



Lib1Net - DPDK Packet Handling

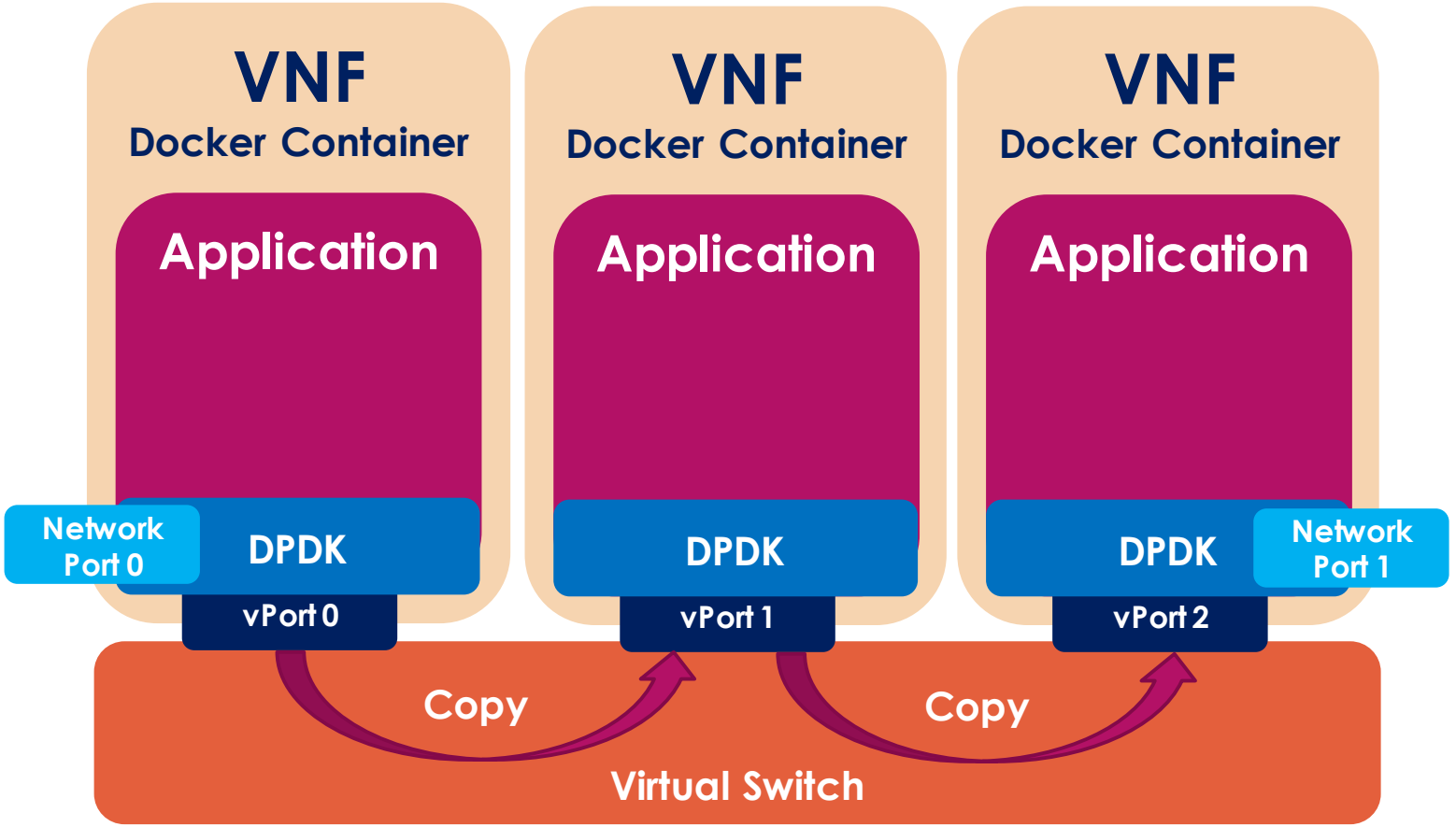
Lib1Net manages RX/TX by:

- Handling RX polling.
- Passing packets to the application through an RX callback.
- Internally buffer TX packets to efficiently transmit bursts.



Lib1Net - Service Chains

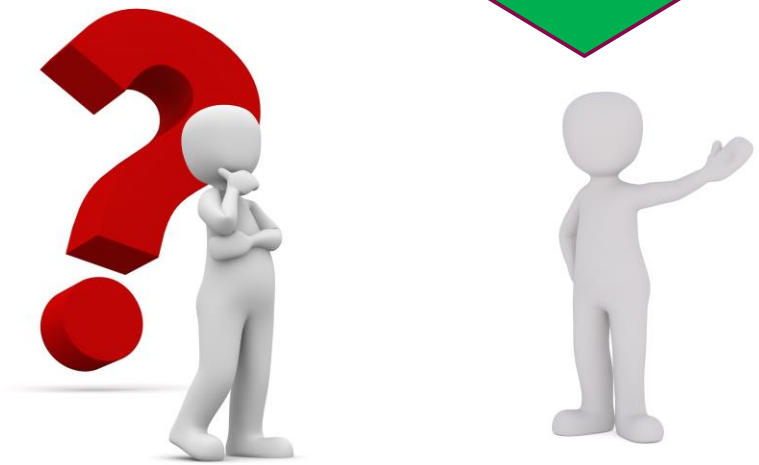
How can I maximise network packet performance and VNF density in my service chain?

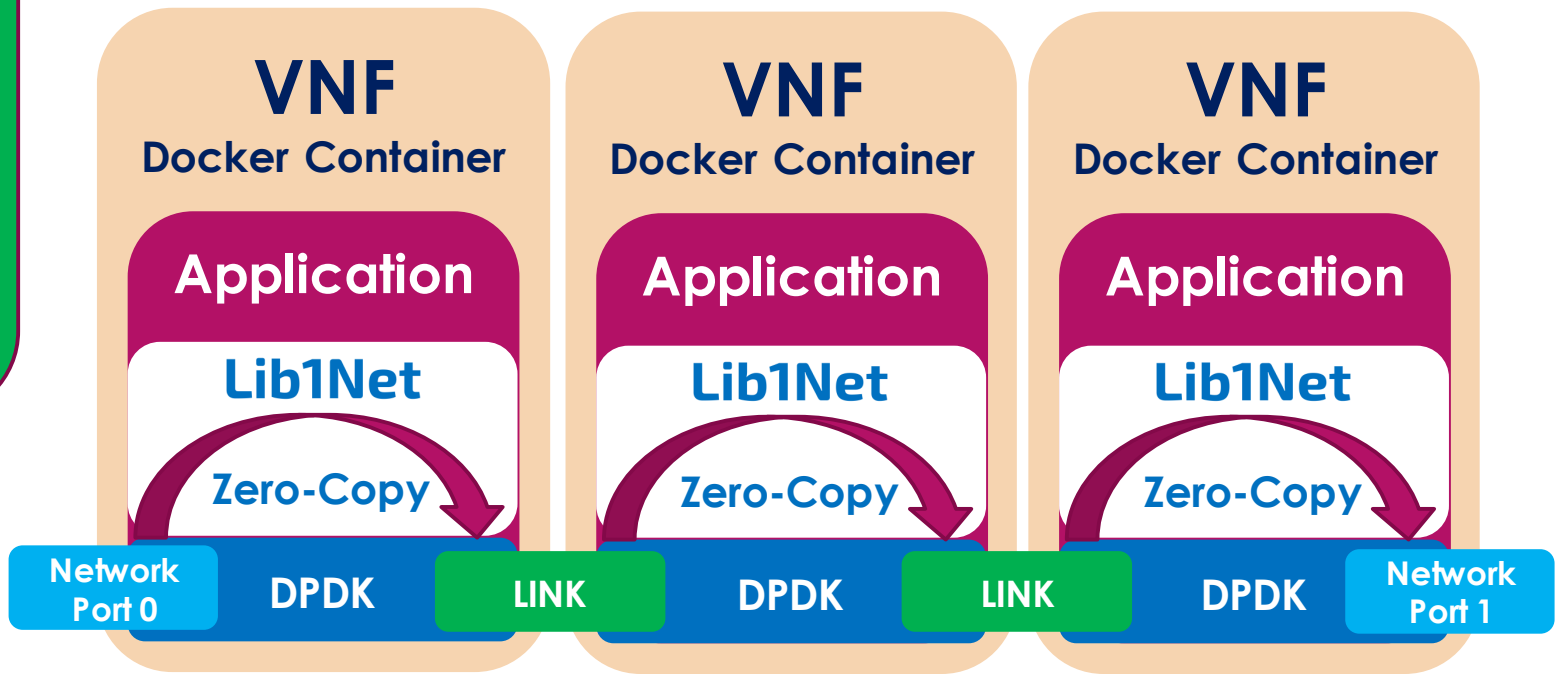


Lib1Net - Service Chains

The **Lib1Net LINK** network interface:

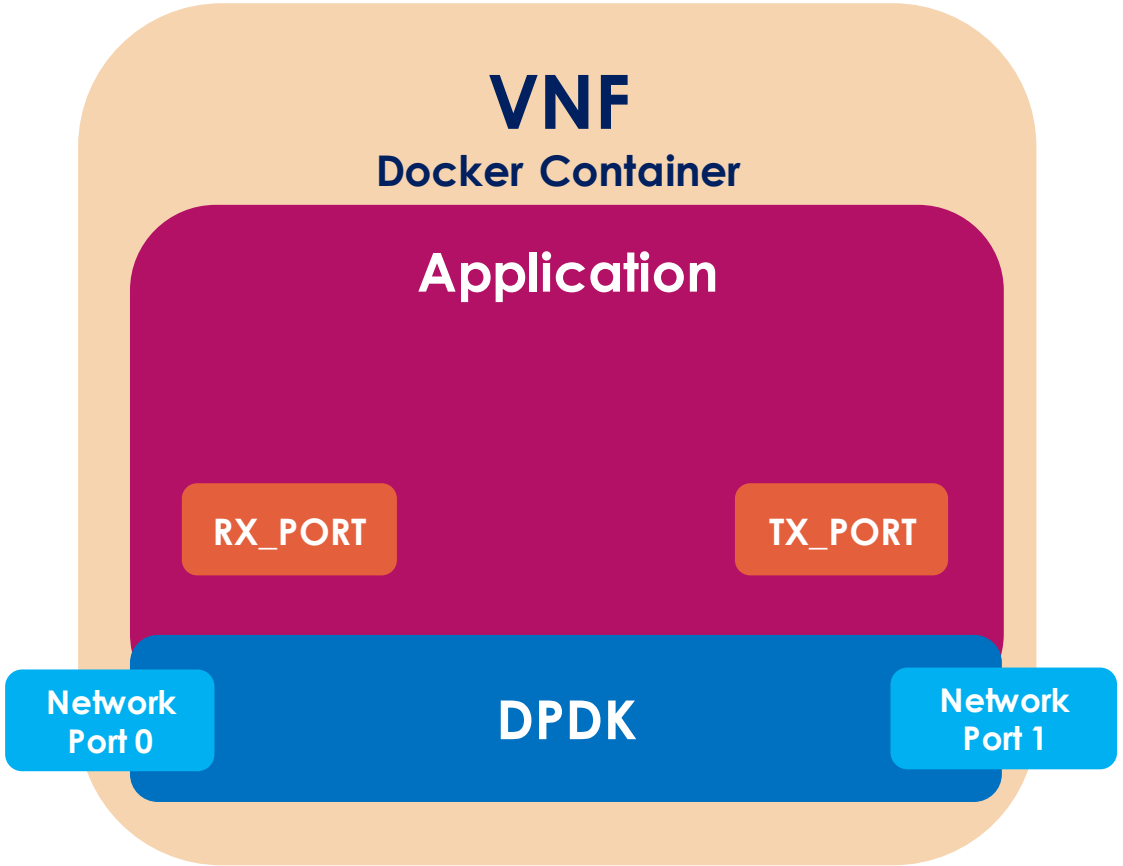
- enables **zero-copy** through the entire service chain.
- Frees system resources by removing the virtual switch.
- Hides the management of DPDK primary/secondary processes.





Lib1Net - Port Management

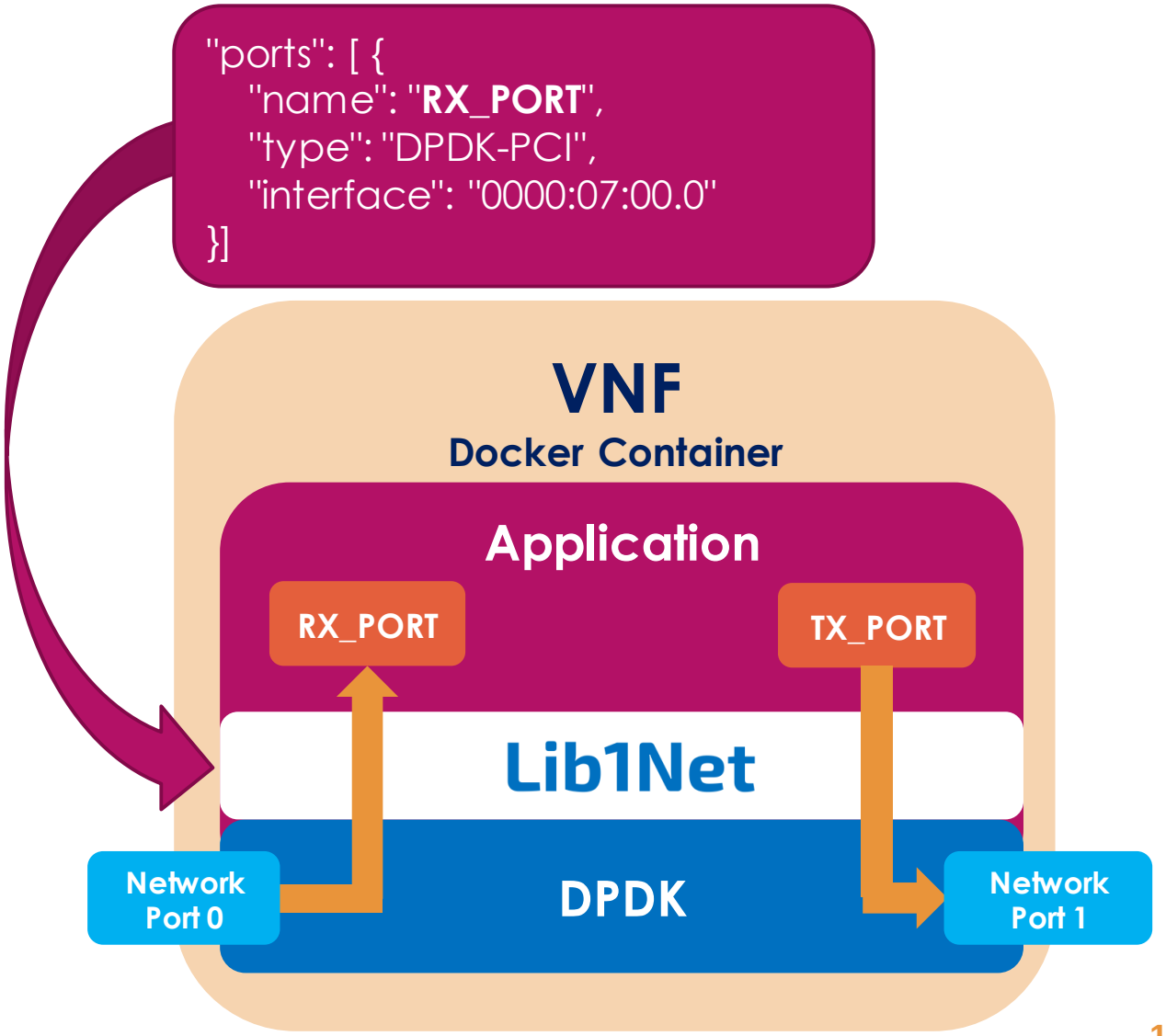
How can I attach network interfaces to specific application ports?



Lib1Net - Port Management

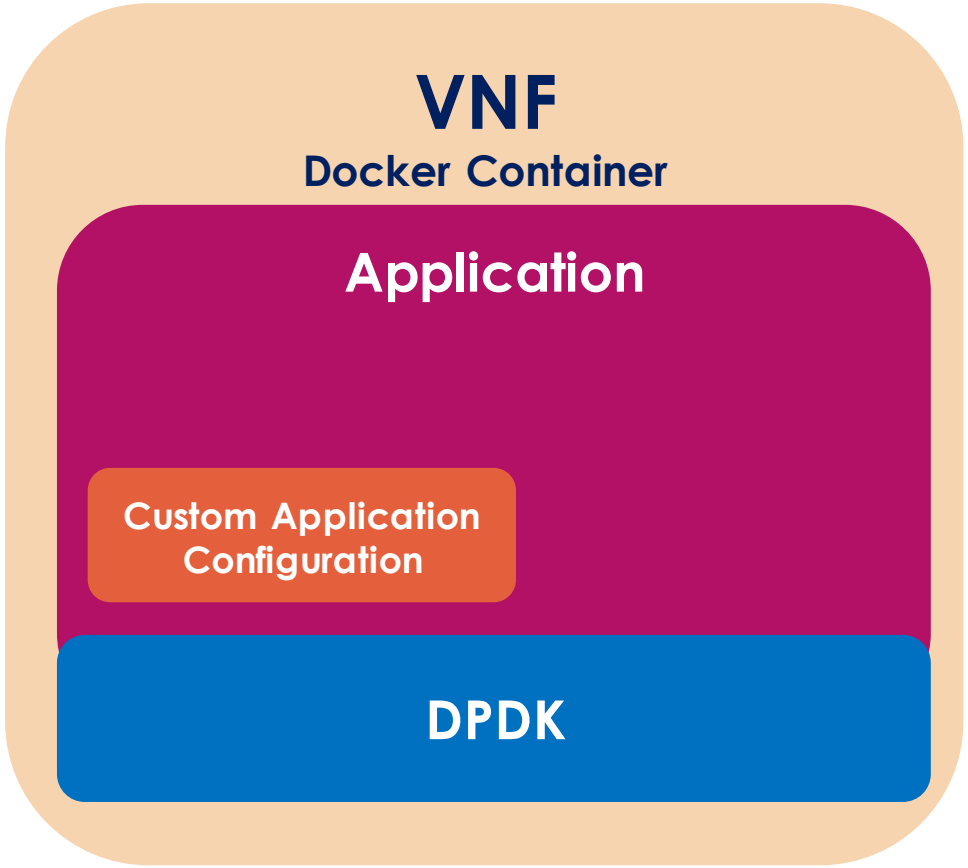
Lib1Net makes this easy by:

- Defining application port names within the application.
- Mapping port names to network interfaces through JSON configuration.



Lib1Net - App Configuration

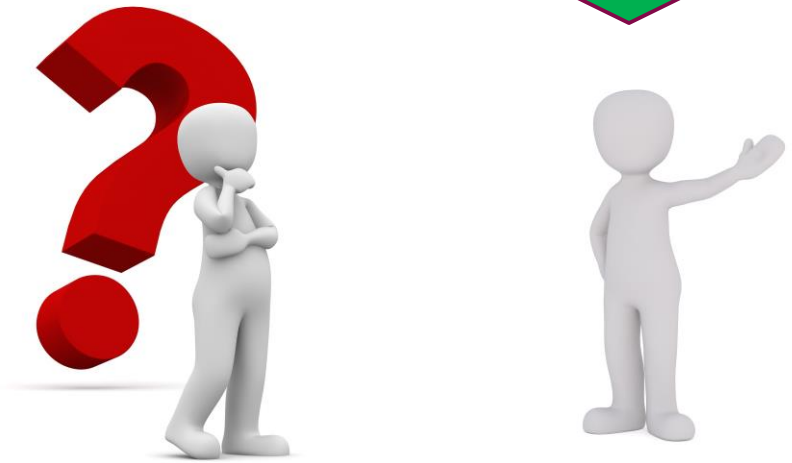
How can I manage my VNF configuration?

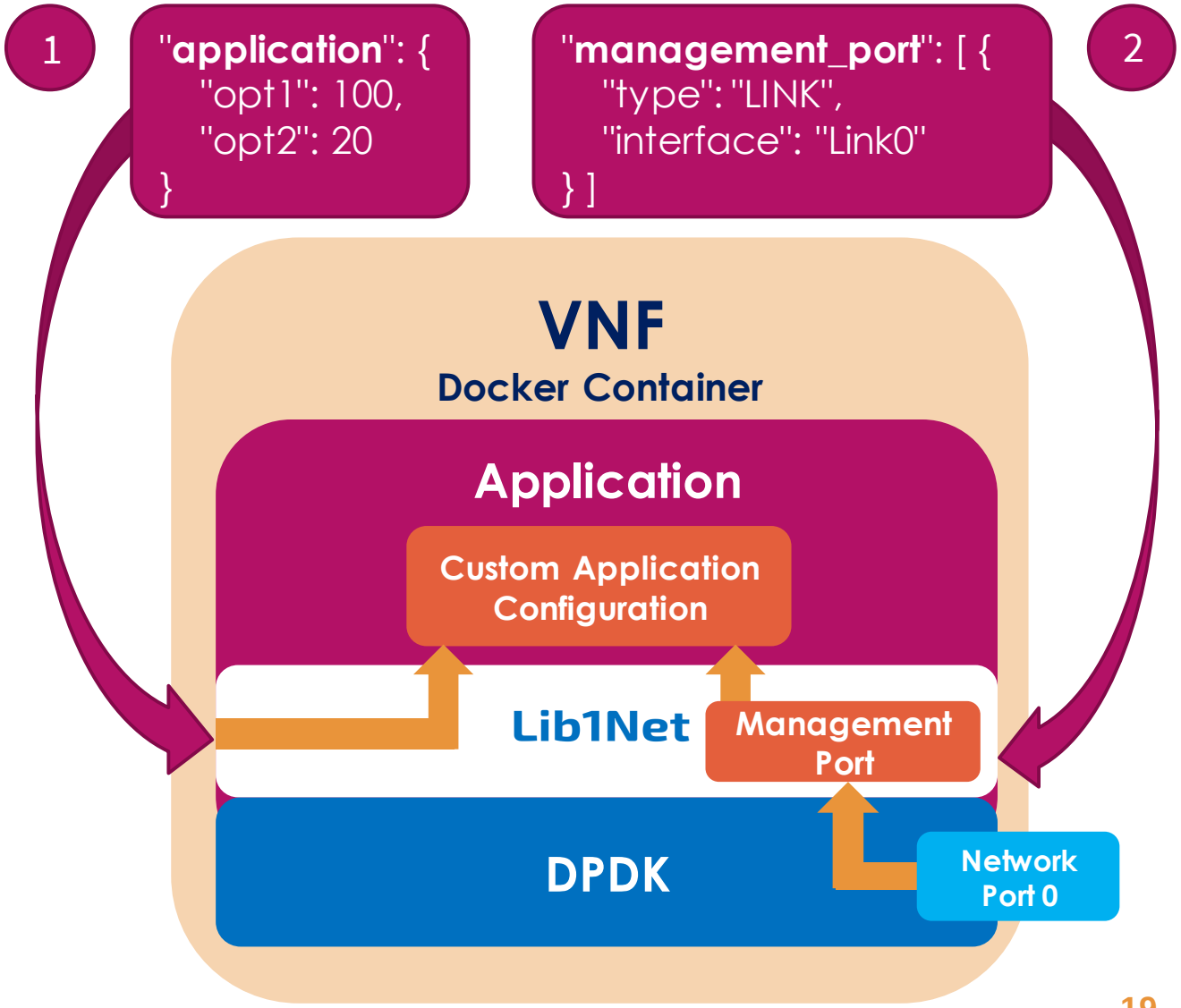


Lib1Net - App Configuration

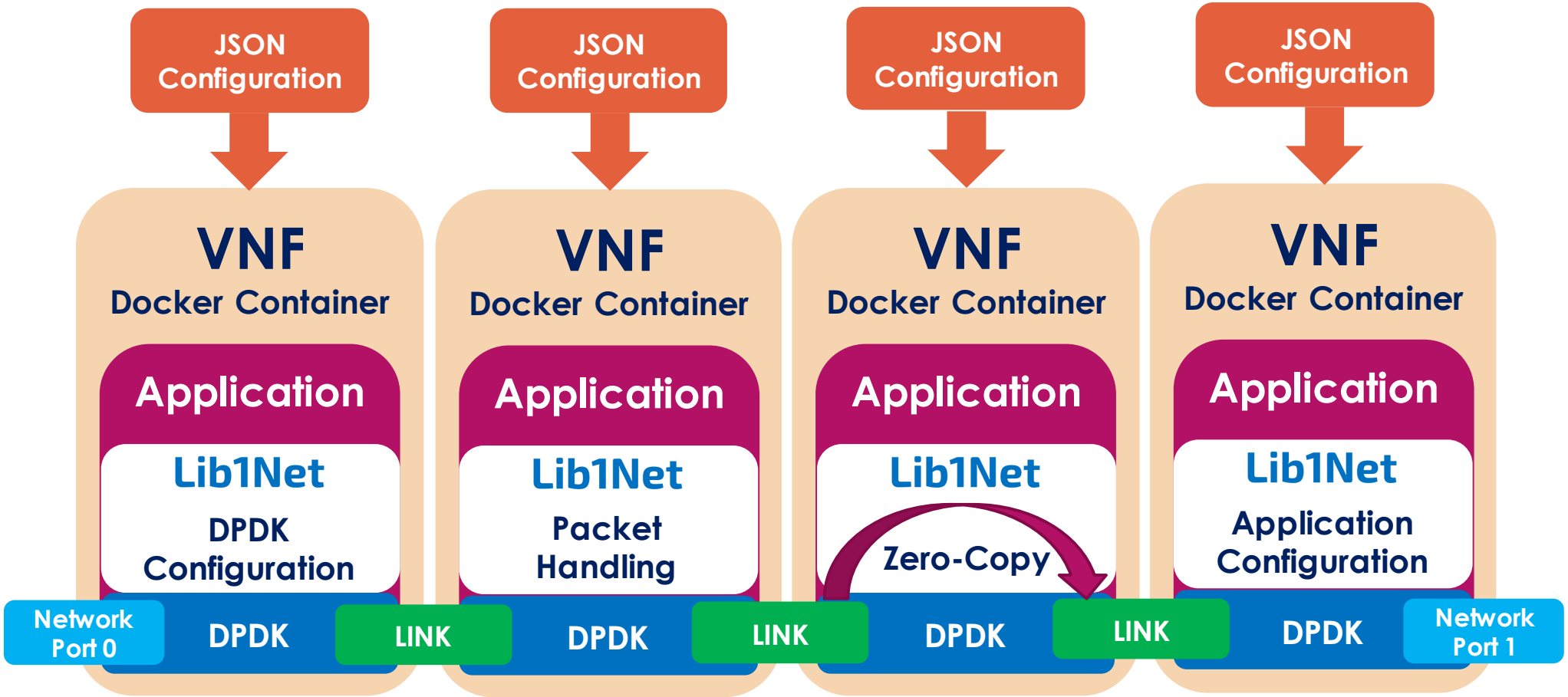
Lib1Net includes two methods for passing configuration messages to your VNF :

1. A reserved field is included in the Lib1Net JSON configuration file.
2. A management port can be attached to the VNF.





Lib1Net - Enterprise Edge Solution



Summary

- Enterprise Edge solutions developed using Lib1Net benefit from:
  - Increased VNF density by optimising hardware usage when service chaining.
  - Maximised network throughput using the Lib1Net LINK interface.
  - Configuration of DPDK and optimised RX/TX packet handling.
  - Simplified mechanism to pass custom application configuration to VNFs.

- For further information:
  - anthony.fee@emutex.com
  - www.emutex.com

Q&A

Backup

DPDK
DATA PLANE DEVELOPMENT KIT

Lib1Net Packet Forwarding Example

main.c

main.c

```c
#include "ports_definition.h"

/** Simple receive callback that forwards all packets to a single interface. */
static void rx_callback(unsigned port_id, liblnet_buffer_t *buffer) {
    liblnet_tx(TX_PORT, buffer);
}

liblnet_config_t config = {.static_config = &static_config, .rx_callback = rx_callback};

int main(void) {
    liblnet_init(&config);

    return 0;
}
```

DPDK
DATA PLANE DEVELOPMENT KIT

Lib1Net Packet Forwarding Example

ports_definition.h

ports_definition.h

```c
#include "lib1net.h"

/**
 * Ports must be defined as an enum. This is used to define the ports in the static_config below.
 */
enum { RX_PORT, TX_PORT, NUM_PORTS };

/**
 * The Lib1Net configuration should be defined at the beginning of application development. It
 * describes the ports as well as configuration file paths which should not need to change once
 * defined.
 */
const static lib1net_static_config_t static_config = {
    /** RX port definition */
    .port_definitions[RX_PORT] =
    {
        /** Define the same port name as set in the enum above */
        .name = "RX_PORT",

        /** Define the port direction as RX only */
        .direction = lib1net_direction_rx,

        /** Set the port as required as it is always needed for the application to function */
        .required = true
    },

    /** TX port definition */
    .port_definitions[TX_PORT] =
    {
        /** Define the same port name as set in the enum above */
        .name = "TX_PORT",

        /** Define the port direction as RX only */
        .direction = lib1net_direction_tx,

        /** Set the port as required as it is always needed for the application to function */
        .required = true
     },

    /** Define the number of ports used by the application */
    .num_ports = NUM_PORTS
};
```

DPDK
DATA PLANE DEVELOPMENT KIT

Lib1Net Packet Forwarding Example

vnf_configuration.json

vnf_configuration.json

```
{
    "container_name": "pkt_fwd",
    "core": 2,
    "ports": [
        {
            "name": "RX_PORT",
            "type": "DPDK-PCI",
            "interface": "0000:04:00.0"
        },
        {
            "name": "TX_PORT",
            "type": "DPDK-PCI",
            "interface": "0000:05:00.0"
        }
    ]
}
```