
Kamuee Users Guide

Revision 1.1 / 0.6.3t

2020 年 4 月 22 日

目次

目次	1
第 1 章 はじめに	1
1.1 Kamuee とは	1
1.2 問い合わせ先・バグレポート	2
1.3 本マニュアルの更新履歴	2
1.4 コマンドシンタックスの表記方式	2
1.5 コマンドリファレンスの表記例	3
1.6 アイコン	3
第 2 章 前提条件	5
2.1 テスト済みハードウェア一覧	5
2.2 CPU アーキテクチャとオペレーティングシステム	6
2.3 本パッケージに含まれる関連ソフトウェア	6
2.4 本パッケージが前提とする関連ソフトウェア	7
2.5 事前準備: hugepage 設定	7
2.6 Intel NIC	7
2.7 Mellanox NIC	8
2.8 (オプション) KVM/QEMU	9
第 3 章 インストール	11
3.1 ダウンロード	11
3.2 インストール	11
3.3 インストールの確認	11
3.4 kamuee の停止	12
3.5 アップグレード	12
3.6 kamuee の起動	12
3.7 kamuee の起動の確認	12
3.8 (オプション) FRRouting のインストール	12
第 4 章 基本コンフィギュレーション	15

4.1	起動時設定ファイル: kamuee.conf	15
4.2	CPU coremap および スレッドの NIC port への割り当て	16
4.3	管理コンソール / コマンドラインインターフェイス (CLI)	18
4.4	物理インターフェイスの確認と設定	19
4.5	物理デバイス・ポートの動的設定例	20
4.6	Jumbo フレームと MTU の設定	22
4.7	論理インターフェイスの確認と設定	23
4.8	VLAN インターフェースの設定	24
4.9	IP アドレスの設定と確認	25
4.10	IP 経路の設定と変更	25
4.11	VRF 設定	29
第 5 章	確認とモニタリング	31
5.1	トラフィック統計	31
5.2	トラフィックモニタリング機能	32
5.3	ログ機能	32
第 6 章	その他	35
6.1	マルチキャストルーター機能	35
6.2	仮想環境での利用	35
第 7 章	コマンドリファレンス	37
7.1	port attach: ポート生成機能	37
7.2	port start: ポート起動機能	38
7.3	port stop: ポート停止機能	39
7.4	port detach: ポート削除機能	39
7.5	show port config: ポート設定確認機能	40
7.6	show devices: デバイス一覧取得機能	40
7.7	device bind: デバイス bind 機能	41
7.8	device unbind: デバイス unbind 機能	41
7.9	show device config: デバイス設定確認機能	42
7.10	write file: コンフィグレーション保存機能	42
第 8 章	アプライアンス製品例	45
8.1	設置と配線	45
8.2	起動とログイン	47
8.3	管理用ポートの IP アドレス設定	47
8.4	kamuee で使用するポート割り当て	48
8.5	ポートの確認と CPU 割り当て	50

1.1 Kamuee とは

kamuee は NTT コミュニケーションズが開発した、高速な経路検索とパケット転送を提供する、ソフトウェアのパケットフォワーディングエンジンです。Kamuee は、「汎用 CPU のみを前提とし、専用のハードウェアを前提としない」というピュアソフトウェアアプローチをとっています。DPDK ライブラリを利用しており、DPDK が動作する CPU アーキテクチャ、Network Interface Card (NIC) であれば、大抵のハードウェアで kamuee が動作すると期待できます。Kamuee は独自の高速経路検索アルゴリズム「Poptrie」を採用しており、OSPF や BGP などの経路制御プロトコルを動作させ、大規模な数の経路を扱うような、full-fledged な（本格的な）ソフトウェアルータとして利用可能です。

kamuee は以下のような利点を持つことを理想とした設計がなされています。

1. 既存のソフトウェア資産を再利用・有効活用すること。kamuee は Linux 上で動作する各種経路制御ソフトウェアと経路情報を共有し、連携することができます。
2. ソフトウェアが性能向上の足かせにならないこと。トラフィック処理をマルチコア CPU 上に並列分散処理させることが可能で、任意の数の CPU スレッドをパケット転送に割り当てることができます。このことから、大量の CPU コアを投入することで、必要とされる性能を容易に準備することができますという期待があります。（コア数を増やすことで対応できるため、トラフィック処理を複数の CPU コアにスケールアウトさせることができる、ということです。）
3. ハードウェアメーカーによるベンダロックインの回避。ソフトウェアルータの特徴を生かし、CPU の交換・増速によるパケット転送性能を調節したり、通信インターフェイス（NIC）の交換・追加・高速化をするなど、ハードウェアの自由な選択と、経済的コストの調整が実現できます。

kamuee は次の複数の要素から構成されています。

- オープンソースの高速パケット転送機構（DPDK ライブラリ）
- 独自開発の超高速経路検索アルゴリズム（poptrie）
- オープンソースの経路制御ソフトウェア

これらの要素技術は単独ではルータとして機能できませんが、kamuee がこれらの機能を制御することで、そ

れぞれを超高速インターネットルータのフォーワーディングエンジンとして利用できるようになりました。結果、Linux 上で動作する各種経路制御ソフトと経路情報を共有し、各エンジンにてパケットの高速転送を提供します。

1.2 問い合わせ先・バグレポート

- kamuee-contact@ml.ntt.com

1.3 本マニュアルの更新履歴

表 1.1 :align: left

バージョン	日付	対象 kamuee ver.	内容
1.1	2020/04/22	0.6.3t	初期バージョン

1.4 コマンドシンタックスの表記方式

コマンドのシンタックスでは、太文字でコマンド文を表記し、円かっこ () と 三角かっこ <> で囲まれた 2 種類のパラメータを表記します。円かっこ () 内に記されたパラメータはオプションパラメータで、任意のパラメータです。三角かっこ <> 内に記載されたパラメータは数値パラメータで、指定された範囲の数値、または指定された ID を入力します。また、円かっこ () 内のパラメータはパイプ文字 | でパラメータが区切られており、パラメータを選択する形式になります。

例 1) `show ipv4 route length (|le|lt|gt|ge) <0-32>`

とシンタックスが表記されていた場合、(|le|lt|gt|ge) はオプションパラメータ、<0-32> は数値パラメータになります。(|le|lt|gt|ge) は選択式で、実際のコマンド入力時には、「le、lt、gt、ge のどれか一つを選ぶ」、または、「パラメータを入力しない」、のどれかを選びます。<0-32> では 0 から 32 の間の数値を記入します。

例 2) `(|no) info fib`

とシンタックスが表記されていた場合、(|no) はオプションパラメータで、no を記入する・記入しないの 2 択になります。no を選択した場合、後ろに続くコマンドの実行内容を解除するコマンドに変わります。

例 3) `show port statistics (|<0-255>|all) (|pps|bps|bytes)`

とシンタックスが表記されていた場合、(|<0-255>|all) と (|pps|bps|bytes) の 2 か所でオプションパラメータを選択します。一つ目のオプションパラメータは、0 から 255 の間の数値を記入する、all を記入する、何も記入しない、のどれかを選択します。二つ目のオプションパラメータは、pps、bps、bytes のどれかを選択する、または、何も選択しない、のどれかを選択します。

1.5 コマンドリファレンスの表記例

コマンドリファレンスの表記項目を [表 1.2](#) にまとめます。

表 1.2 出力例の解説

用語	解説
CLI 形式	コマンドラインインターフェイス (CLI) のコマンド形式を表します。
説明	コマンドの機能の概要を説明します。
処理	コマンドが実行する処理をリストします。
引数	パラメーターとして指定できる値とその意味を説明します。
前提条件	コマンドが前提とする条件を説明します。
備考	その他注意事項があれば記載します。

コマンドリファレンスの表記例は以下のようになります。

- CLI 形式





```
show port
```

- 説明
 -
- 処理
 -
- 引数
 -
- 前提条件
 -
- 備考
 -

1.6 アイコン

本文中でいくつかアイコンを使用する場合があります。使用する各アイコンの概要を [表 1.3](#) にまとめます。

表 1.3 アイコンの説明

アイコン	説明
 (メモ)	注釈、または、補足情報を記載する場合に使用するアイコンです。
 (注意)	注意事項を記載する場合に使用するアイコンです。
 ワンポイント・アドバイス	ワンポイントアドバイスを記載する場合に使用するアイコンです。
 ヒント	ヒントを記載する場合に使用するアイコンです。

2.1 テスト済みハードウェア一覧

2.1.1 Tested NICs

- Intel Ethernet Converged Network Adapter X550-T2 (10G-T)
- Intel Ethernet Converged Network Adapter X540-T2 (10G-T)
- Intel Ethernet Converged Network Adapter X710 DA-2/DA-4 (10G)
- Intel Ethernet Converged Network Adapter XL710 (40G)
- Mellanox ConnectX-5 VPI (MCX556A-ECAT) (100G)

2.1.2 Tested Server Hardware

- Supermicro SYS-7049GP-TRT (4U)
 - Motherboard: X11DPG-QT
 - CPU: Intel Xeon Platinum 8180 (2.5GHz, 28 cores/56 threads) x 2
 - Mem: 192GB (DDR4-2666 16GB x 12)
 - NIC: Intel X710 DA-4, Intel X550/540-T2, Mellanox ConnectX-5 VPI
- Supermicro SYS-2028R-TXR (2U)
 - Motherboard: X11DRX
 - CPU: Intel Xeon E5-2699A v4 (22 cores/44 threads) x 2
 - Mem: 128GB (DDR4-2400 8GB x 16)
 - NIC: Intel XL710, Intel X710 DA-4

- Fujitsu Primergy RX200 S8 (1U)
 - CPU: Intel Xeon E5-2620 v2 (6 cores/12 threads) x 2
 - Mem: 32GB
- Fujitsu Primergy RX2530 M2 (1U)
 - CPU: Intel Xeon E5-2683 v4 (16 cores/32 threads) x 2
 - Mem: 256GB
- LANNER NCA-5710C (1U)
 - CPU: Intel Xeon Scalable Processor x 2
 - NIC: NC2S-MXH01 (Mellanox ConnectX-4)
- Ampere Computing eMAG 8180 server (2U)
 - CPU: eMAG 8180 ARMv8 CPU (32 cores) x 1
 - Mem: 64GB
 - NIC: Intel X710 DA-4

2.2 CPU アーキテクチャとオペレーティングシステム

kamuee は、以下の環境で動作することが確認されています。

- PC アーキテクチャ: amd64/x86_64
- ARM アーキテクチャ: arm64/armv8a)

以下のオペレーティングシステムがテストされています。


- Ubuntu 18.04.2 LTS

本パッケージに付属の DPDK/igb_uio ドライバを使う場合には、以下の Linux カーネルを利用する必要があります。

- Linux kernel: linux-image-4.15.0-50-generic

2.3 本パッケージに含まれる関連ソフトウェア

- DPDK 18.11.2 (x86_64-nehalem-linuxapp-gcc)
- DPDK 19.02.0 (arm64-armv8a-linuxapp-gcc)

 (注意)

- libpci, libkmod 開発パッケージについて

GPL 違反している可能性があります

libpci, libkvm とともに Kamuee の実行形式バイナリに static リンクされるようになっているため、実行ホストには libpci, libkvm の実行時パッケージは必ずしも必要ありません。

ビルドの際には、libpci-dev libkvm-dev のパッケージが構築ホストに必要です（デバイス関連の機能が libpci, libkvm のパッケージのライブラリ関数を利用しているため）。

```
apt install libpci-dev libkvm-dev
```

2.4 本パッケージが前提とする関連ソフトウェア

debian パッケージ (dpkg) で提供される本パッケージでは、必要なソフトウェアは自動でインストールされますので、特別な準備は必要ありません。

```
libc6 (>= 2.27), liburcu-dev (>= 0.10.1), pkg-config (>= 0.29.1), libjson-c-dev (>= 0.12.1), libmnl-dev (>= 1.0.4), libnuma-dev (>= 2.0.11)
```

2.5 事前準備: hugepage 設定

DPDK のアプリケーションを動かすためには、Hugepage の設定が必要となります。本マニュアルでは、1GB Hugepage の設定を推奨しています。

本パッケージに付属のシェルスクリプト (dpdk-hugepage-1G.sh) を利用して、1GB Hugepage を設定できます。

- /usr/local/share/kamuee/dpdk-hugepage-1G.sh

```
sh dpdk-hugepages-1G.sh
reboot
```

注釈: You can check the hugepages are configured with `cat /proc/meminfo | grep -i huge`.

2.6 Intel NIC

2.6.1 uio_pci_generic の利用

```
# lshw -class net -businfo

Bus info          Device          Class          Description
=====
```

(次のページに続く)

(前のページからの続き)

```

pci@0000:01:00.0  eno1      network      Ethernet Controller 10G X550T
pci@0000:01:00.1  eno2      network      Ethernet Controller 10G X550T
pci@0000:18:00.0  enp24s0f0 network      Ethernet Controller 10-Gigabit X540-
↳AT2
pci@0000:18:00.1  enp24s0f1 network      Ethernet Controller 10-Gigabit X540-
↳AT2

# modprobe uio_pci_generic
# /usr/local/share/dpdk/usertools/dpdk-devbind.py -b uio_pci_generic 18:00.0
# /usr/local/share/dpdk/usertools/dpdk-devbind.py -b uio_pci_generic 18:00.1
# /usr/local/share/dpdk/usertools/dpdk-devbind.py -s
..(snip)
Network devices using DPDK-compatible driver
=====
0000:18:00.0 'Ethernet Controller 10-Gigabit X540-AT2' drv=uio_pci_generic_
↳unused=ixgbe
0000:18:00.1 'Ethernet Controller 10-Gigabit X540-AT2' drv=uio_pci_generic_
↳unused=ixgbe

Network devices using kernel driver
=====
..(snip)

```

2.6.2 igb_uio.ko

```

# modprobe uio
# insmod /usr/local/share/dpdk/$(RTE_TARGET)/kmod/igb_uio.ko
# /usr/local/share/dpdk/usertools/dpdk-devbind.py -b igb_uio 18:00.0
# /usr/local/share/dpdk/usertools/dpdk-devbind.py -b igb_uio 18:00.1

```

2.7 Mellanox NIC

Install MLX OFED

ofed-version: 4.5-1.0.1.0

```

$ sudo apt update && sudo apt install --yes \
    unzip quilt debhelper libnl-route-3-200 \
    chrpath python-libxml2 dkms libltdl-dev \
    swig dpatch graphviz
$ mkdir $HOME/mlnx
$ OFED_VER=4.5-1.0.1.0
$ OFED_URL=http://content.mellanox.com/ofed/MLNX_OFED-${OFED_VER}
$ OFED_FILE=MLNX_OFED_LINUX-${OFED_VER}-ubuntu18.04-x86_64.tgz
$ OFED_FILE_BASE=${OFED_FILE%.*}
$ wget --quiet ${OFED_URL}/${OFED_FILE} -O $HOME/${OFED_FILE}
$ tar xvf $HOME/${OFED_FILE} -C $HOME/mlnx

```

(次のページに続く)

(前のページからの続き)

```
$ cd $HOME/mlnx/${OFED_FILE_BASE}/
$ sudo ./mlnxofedinstall --force --upstream-libs --dpdk
```

Hardware の FW を reset して再起動.

```
$ sudo /etc/init.d/openibd restart
$ sudo mlxconfig -d 3b:00.0 reset
$ sudo reboot
```

2.8 (オプション) KVM/QEMU

2.8.1 QEMU with PCI-passthrough

```
echo 1 > /sys/class/net/enp0s1f0/devices/sriov_numvfs
echo 1 > /sys/class/net/enp0s1f1/devices/sriov_numvfs
dpdk_nic_bind -b vfio-pci 01:00.2
dpdk_nic_bind -b vfio-pci 01:00.3
qemu-system-x86_64 \
  -cpu host -enable-kvm \
  -smp 8 -m 8000 -hda $IMAGE \
  -net nic,macaddr=52:54:00:22:22:22 \
  -net tap,script=ifup.sh,downscript=ifdown.sh \
  -vnc :1,password -monitor stdio \
  -device vfio-pci,host=01:00.2 \
  -device vfio-pci,host=01:00.3
```

2.8.2 QEMU with SR-IOV

```
linux cmdline
"intel_iommu=on pci=assign-busses pci=realloc"

sudo mst start    ## start mst daemon, this enables us to check mlx5 config and
↳ registers
sudo mlxconfig -d /dev/mst/mtXXXX_pciconf0 q | grep SRIOV    ## check sriov is
↳ enabled on mlx5
sudo  mlxconfig -d /dev/mst/mt4119_pciconf0 q | grep -e "NUM_OF_VFS" -e "SRIOV"
sudo  mlxconfig -y -d /dev/mst/mt4119_pciconf4 set SRIOV_EN=1 NUM_OF_VFS=16
```

- PROS:
 - Good Performance
- CONS:
 - Slow to start VM
 - Difficult to migrat VM

- Need restart when sriov-config is changed

3

インストール

3.1 ダウンロード

```
% wget http://www.kamuee.net/download/kamuee_0.6.3t_amd64.deb
```

パッケージの種類には、以下があります。

- kamuee_0.6.3t_amd64.deb: amd64/x86_64 アーキテクチャ用 Intel NIC 用 debian パッケージ
- kamuee_0.6.3tm_amd64.deb: amd64/x86_64 アーキテクチャ用 Intel/Mellanox NIC 用 debian パッケージ
- kamuee_0.6.3t_arm64.deb: arm64/armv8a アーキテクチャ用 Intel NIC 用 debian パッケージ

3.2 インストール

```
# apt install ./kamuee_0.6.3t_amd64.deb
```

3.3 インストールの確認

```
# apt show kamuee
Package: kamuee
Version: 0.6.3t
Status: install ok installed
Priority: optional
Section: non-free/net
Maintainer: Yasuhiro Ohara <yasu@nttv6.jp>
Installed-Size: unknown
Depends: libc6 (>= 2.27), liburcu-dev (>= 0.10.1), pkg-config (>= 0.29.1), libjson-
↳c-dev (>= 0.12.1), libmnl-dev (>= 1.0.4), libnuma-dev (>= 2.0.11)
Download-Size: unknown
APT-Manual-Installed: yes
```

(次のページに続く)

(前のページからの続き)

```
APT-Sources: /var/lib/dpkg/status
Description: Kamuee software router
```

3.4 kamuee の停止

```
# systemctl stop kamuee
```

3.5 アップグレード

```
# apt remove kamuee
# apt install ./kamuee_0.6.3t_amd64.deb
```

3.6 kamuee の起動

```
# systemctl start kamuee
```

3.7 kamuee の起動の確認

```
# systemctl status kamuee
```

```
# ps ax | grep kamuee
```

```
# tail -f /var/log/syslog
```

```
# telnet localhost 9077
```

3.8 (オプション) FRRouting のインストール

```
wget https://github.com/FRRouting/frr/releases/download/frr-6.0.2/frr_6.0.2-0.
↪ubuntu18.04.1_amd64.deb
apt install -y ./frr_6.0.2-0.ubuntu18.04.1_amd64.deb
systemctl start frr
```

```
vtysh -c 'show version'
FRRouting 6.0 (96c021364c0b).
Copyright 1996-2005 Kunihiro Ishiguro, et al.
configured with:
```

(次のページに続く)

(前のページからの続き)

```

'--build=x86_64-linux-gnu' '--prefix=/usr' \
'--includedir=${prefix}/include' '--mandir=${prefix}/share/man' \
'--infodir=${prefix}/share/info' '--sysconfdir=/etc' \
'--localstatedir=/var' '--disable-silent-rules' \
'--libexecdir=${prefix}/lib/frr' '--disable-maintainer-mode' \
'--disable-dependency-tracking' \
'--enable-exampledir=/usr/share/doc/frr/examples/' \
'--localstatedir=/var/run/frr' '--sbindir=/usr/lib/frr' \
'--sysconfdir=/etc/frr' '--disable-snmp' '--enable-ospfapi=yes' \
'--enable-multipath=256' '--enable-ldpd' '--enable-fpm' \
'--enable-user=frr' '--enable-group=frr' '--enable-vty-group=frrvty' \
'--enable-configfile-mask=0640' '--enable-logfile-mask=0640' \
'--enable-werror' '--with-libpam' '--enable-systemd=yes' \
'--enable-poll=yes' '--enable-cumulus=no' '--enable-pimd' \
'--enable-dependency-tracking' '--enable-bgp-vnc=yes' \
'--disable-rpki' '--enable-bfdd' \
'CFLAGS=-g -O2 -fdebug-prefix-map=/home/ci/cibuild.6/debwork/frr-6.0=. \
-fstack-protector-strong -Wformat -Werror=format-security' \
'CPPFLAGS=-Wdate-time -D_FORTIFY_SOURCE=2' 'CXXFLAGS=-g -O2 \
-fdebug-prefix-map=/home/ci/cibuild.6/debwork/frr-6.0=. \
-fstack-protector-strong -Wformat -Werror=format-security' \
'FCFLAGS=-g -O2 -fdebug-prefix-map=/home/ci/cibuild.6/debwork/frr-6.0=. \
-fstack-protector-strong' 'FFLAGS=-g -O2 \
-fdebug-prefix-map=/home/ci/cibuild.6/debwork/frr-6.0=. \
-fstack-protector-strong' 'GCJFLAGS=-g -O2 \
-fdebug-prefix-map=/home/ci/cibuild.6/debwork/frr-6.0=. \
-fstack-protector-strong' 'LDFLAGS=-Wl,-Bsymbolic-functions \
-Wl,-z,relro -Wl,-z,now' 'OBJCFLAGS=-g -O2 \
-fdebug-prefix-map=/home/ci/cibuild.6/debwork/frr-6.0=. \
-fstack-protector-strong -Wformat -Werror=format-security' \
'OBJCXXFLAGS=-g -O2 -fdebug-prefix-map=/home/ci/cibuild.6/debwork/frr-6.0=. \
-fstack-protector-strong -Wformat -Werror=format-security' \
'build_alias=x86_64-linux-gnu'

```


基本コンフィギュレーション

kamuee で高速な経路検索とパケット転送を利用するには、ネットワークインターフェースを kamuee 用に割り当て、kamuee から各種パラメータを設定する必要があります。それにより、iproute2 や FRR などのサードパーティルーティングソフトウェアからでも高速な経路検索とパケット転送を利用できます。

4.1 起動時設定ファイル: kamuee.conf

kamuee の設定は `/etc/kamuee/kamuee.conf` に保存されています。起動時にはこの設定ファイルを読み込んで、自動的にルーティングの転送設定を DPDK に反映します。付属のサンプル設定（初期設定）を下記に示します。

```
set thread 0 os
set thread 1 os

set thread 2 lthread_scheduler
set thread 3 snmp_manager

set thread 4 forwarder port 0 rx-queue 0
set thread 5 forwarder port 0 rx-queue 1

set thread 6 forwarder port 1 rx-queue 0
set thread 7 forwarder port 1 rx-queue 1

set thread 8 forwarder port 2 rx-queue 0
set thread 9 forwarder port 2 rx-queue 1

set thread 10 forwarder port 3 rx-queue 0
set thread 11 forwarder port 3 rx-queue 1

set thread 28 rib_manager
set thread 29 tap_manager

set thread 30 forwarder port 4 rx-queue 0
set thread 31 forwarder port 4 rx-queue 1
```

(次のページに続く)

(前のページからの続き)

```

set thread 32 forwarder port 5 rx-queue 0
set thread 33 forwarder port 5 rx-queue 1

set thread 34 forwarder port 6 rx-queue 0
set thread 35 forwarder port 6 rx-queue 1

set thread 36 forwarder port 7 rx-queue 0
set thread 37 forwarder port 7 rx-queue 1

```

kamuee.conf に記載されるのは、kamuee CLI で実行可能なコマンドに過ぎません。各コマンドは、以降の節で個別に説明します。

4.2 CPU coremap および スレッドの NIC port への割り当て

CPU coremap およびスレッドの NIC port への割り当ては、基本的に設定ファイル kamuee.conf で行われます。動的な coremap やスレッドの NIC port への割り当て変更は、未だサポートされていません。以降のコマンドは、kamuee.conf に記述し、kamuee 起動時の初期に実行されるようにしてください。

ここでは、前節の kamuee.conf を例に取り、各コマンドを順に説明します。

set thread X os コマンドは、オペレーティングシステム（OS）用に core X を確保し、何も動作させないためのダミーコマンドです。オペレーティングシステム用には、2~4 コアほど残しておけば良いでしょう。

kamuee では、native thread で動作させる必要のある thread がいくつかあります。それらは、lthread_scheduler、snmp_manager、rib_manager、tap_manager です。lthread_scheduler は、バーチャルターミナル（VTY）や netlink 受信 thread を含む、多数の重要 thread を仮想的に動作する thread です。lthread_scheduler によってこれらの重要 thread が lthread として実行されます。snmp_manager は、Net-SNMP への送受信を処理する thread であり、SNMP を動作させるのでなければ、設定する必要はありません。rib_manager および tap_manager はそれぞれ、routing socket および TUN/TAP I/F とのデータ送受信を処理する thread であり、パフォーマンスバランスのために、native thread として実行する必要があります。これらの 4 つの thread は、NUMA に関係なく設定することができます。

set thread X forwarder port Y rx-queue Z は、port Y 用の forwarder を core X に設定するコマンドです。port には任意の数の forwarder を設定でき、これらの thread が port 向けトラフィックの受信を並列分散処理します。port Y 用の複数 forwarder は異なる rx-queue に設定する必要があります。rx-queue 番号 Z は、0 から連続で順番に設定する必要があります。コア番号が昇順で飛ぶ（間隔を開ける）ことは問題ありませんが、コア番号が戻ることは許されていません（テストされておらず、問題が出ます）。forwarder の設定は、NUMA に関係なく設定することもできますが、NUMA 的に非効率な設定をすると、10% ほど性能が低下すると思われます。

thread の設定は、kamuee 管理コンソール（CLI）にて、show thread info で確認することができます。（CLI へのアクセス方法は、次節で説明します。）

```

kamuee[vty0]> show thread info
core[id]: v thread name          lcpu pcpu port  rxq funcp
core[0]:  1 OS                  0    0    0    0 (nil)
core[1]:  1 OS                  0   --   --   -- (nil)

```

(次のページに続く)

(前のページからの続き)

```

core[2]: 1 lthread_scheduler      0  --  --  -- 0x5555558fe8e
core[3]: 1 snmp_manager          0  --  --  -- (nil)
core[4]: 1 forwarder             0  0  0  0 0x5555556b429a
core[5]: 1 forwarder             0  0  0  1 0x5555556b429a
core[6]: 1 forwarder             0  0  0  2 0x5555556b429a
core[7]: 1 forwarder             0  0  0  3 0x5555556b429a
core[8]: 1 forwarder             0  0  0  4 0x5555556b429a
core[9]: 1 forwarder             0  0  0  5 0x5555556b429a
core[10]: 1 forwarder            0  0  0  6 0x5555556b429a
core[11]: 1 forwarder            0  0  0  7 0x5555556b429a
core[12]: 1 forwarder            0  --  --  -- 0x5555556b429a
core[13]: 1 forwarder            0  --  --  -- 0x5555556b429a
core[14]: 1 forwarder            0  --  --  -- 0x5555556b429a
core[15]: 1 forwarder            0  --  --  -- 0x5555556b429a
core[16]: 1 forwarder            0  0  2  0 0x5555556b429a
core[17]: 1 forwarder            0  0  2  1 0x5555556b429a
core[18]: 1 forwarder            0  0  2  2 0x5555556b429a
core[19]: 1 forwarder            0  0  2  3 0x5555556b429a
core[20]: 1 forwarder            0  0  2  4 0x5555556b429a
core[21]: 1 forwarder            0  0  2  5 0x5555556b429a
core[22]: 1 forwarder            0  0  2  6 0x5555556b429a
core[23]: 1 forwarder            0  0  2  7 0x5555556b429a
core[24]: 1 forwarder            0  --  --  -- 0x5555556b429a
core[25]: 1 forwarder            0  --  --  -- 0x5555556b429a
core[26]: 1 forwarder            0  --  --  -- 0x5555556b429a
core[27]: 1 forwarder            0  --  --  -- 0x5555556b429a
core[28]: 1 rib_manager          1  --  --  -- 0x5555556385d4
core[29]: 1 tap_manager          1  --  --  -- 0x5555557d33c1
core[30]: 1 forwarder            1  1  4  0 0x5555556b429a
core[31]: 1 forwarder            1  1  4  1 0x5555556b429a
core[32]: 1 forwarder            1  1  4  2 0x5555556b429a
core[33]: 1 forwarder            1  1  4  3 0x5555556b429a

```

(略)

CPU スレッドの統計情報を確認したい場合は、下記のコマンドで確認できます。

```
kamuee-vty[0] > show thread statistics pps
```

kamuee 設定ファイル (kamuee.conf) は、kamuee CLI のコマンドの羅列に過ぎません。後述の、vlan 関連や jumbo-frame、mtu 関連のコマンドも、設定ファイルに記述しておくことによって、起動時に自動で設定されるようになります。

⚠ (注意)

mirror コマンド (set port X mirror X) だけは、kamuee 設定ファイルでは有効化になりません。起動してから kamuee CLI で実行してください。

4.3 管理コンソール / コマンドラインインターフェイス (CLI)

4.3.1 CLI へのアクセス

kamuee ではコマンドラインインターフェイスでデーモンを操作します。コマンドラインインターフェイスは、kamuee デーモンをフォラグラウンドで実行した際のコンソール（標準入出力端末（stdin/stdout））に現れるほか、telnet で接続することもできます。

コマンドを実行する場合は、実行デーモン（kamuee デーモン）にログインしてコマンドを入力します。kamuee デーモンへのログイン方法は、コンソールまたは ssh コマンドなどでアプライアンスにログインした後、下記のように **telnet** コマンドを実行します。

```
kamuee@kamuee:~$ telnet localhost 9077
```

ログインすると次のようにコマンドプロンプトが表示されます。

```
kamuee-vty[0]>
```

CLI からのログアウトは、**logout**, **exit**, **quit**, いずれかのコマンドで行えます。次は **logout** を入力した場合の出力例です。

```
kamuee[vty0]> logout
vty exit !
Connection closed by foreign host.
kamuee@kamuee:~$
```

(メモ)

kamuee には、ログイン時のパスワード入力はありません。また、特権モードやコンフィグモードなどありません。ご注意ください。

4.3.2 コマンドヘルプ、コマンド候補の表示、コマンドの補完

コマンド入力で次に続くパラメータがわからない場合、<?> キーを入力すると、パラメータの候補が表示されます。

```
kamuee[vty0]> show port
```

と表示されている状態で ? キーを入力すると、

```
kamuee[vty0]> show port
<cr>                Port information
<0-127>             Specify the port ID.
all                 Show all ports.
statistics          Port statistics
kamuee[vty0]> show port
```

と、**show port** に続くパラメータ候補が表示されます。

kamuee では、コマンドが一意に定まる場合、TAB キーにより補完できます。

```
kamuee[vty0]> show th
```

と入力した状態で TAB キーを押すと補完され

```
kamuee[vty0]> show thread
```

と入力された状態になります。

4.4 物理インターフェースの確認と設定

kamuee 側で物理インターフェース情報を確認する場合は、下記のコマンドを実行します。

```
kamuee[vty0]> show port
port      id c  if tap mac-addr      drvtr  speed supported
port-3b-0-0  0 0  4 273 50:6B:4B:08:6C:56 mlx5 100000 100G/50G/40G/25G/10G/1G
port-3b-0-1  1 0  5 274 50:6B:4B:08:6C:57 mlx5    0 40G/10G/1G
port-5e-0-0  2 0  6 275 50:6B:4B:B6:C6:F8 mlx5 100000 100G/50G/40G/25G/10G/1G
port-5e-0-1  3 0  7 276 50:6B:4B:B6:C6:F9 mlx5    0 40G/10G/1G
port-86-0-0  4 1  8 277 50:6B:4B:08:61:DE mlx5 100000 100G/50G/40G/25G/10G/1G
port-86-0-1  5 1  9 278 50:6B:4B:08:61:DF mlx5    0 40G/10G/1G
port-af-0-0  6 1 11 279 50:6B:4B:B6:C8:E0 mlx5 100000 100G/50G/40G/25G/10G/1G
port-af-0-1  7 1 12 280 50:6B:4B:B6:C8:E1 mlx5    0 40G/10G/1G
```

特定の物理ポートの詳細情報を表示する場合は、**show port** コマンドで表示する物理ポートの ID を指定します。

```
kamuee-vty[0]> show port 0
port0:
  new name: kni-3b-0-0
  valid: 1
  vrf: 1 (router)
  flags: <RUNNING|OURS-TAP>
  kni ifindex: -1
  tap ifindex: 273
  tap name: port-3b-0-0
  tap sockfd: 288
  promiscuous:      1      allmulticast:      1
  nrxq:              8      ntxq:              56
  hwaddr: 50:6B:4B:08:6C:56
  device info:
    pci_dev_name: 0000:3b:00.0
    driver_name: net_mlx5
    if_index: 4
    min_rx_bufsize: 32      max_rx_pktlen: 65536
    max_rx_queues: 65535    max_tx_queues: 65535
    nb_rx_queues: 8      nb_tx_queues: 56
```

(次のページに続く)

(前のページからの続き)

```

    reta_size:          8      hash_key_size:      40
    speed_capa: 100G/50G/40G/25G/10G/1G
    rx_offload_capa: <VLAN-STRIP|IPV4-CKSUM|UDP-CKSUM|TCP-CKSUM>
    tx_offload_capa: <VLAN-INSERT|IPV4-CKSUM|UDP-CKSUM|TCP-CKSUM|TCP-
↪TSO|OUTER-IPV4-CKSUM|VXLAN-TNL-TSO|GRE-TNL-TSO>
    default_txconf:
        tx_free_thresh: 0
    ifaddrs[ipv4]:
        192.85.2.1/24
    ifaddrs[ipv6]:
        ::/0
        2001:0:1::1/64
        fe80::526b:4bff:fe08:6c56/64
    mirror port:      none
(略)

```

4.5 物理デバイス・ポートの動的設定例

kamuee では、物理デバイスは `dppdk-devbind.py` 等であらかじめ設定しておき、コア・スレッドは `kamuee.conf` で起動時初期に設定するのが、現状ではもっとも検証された、安全・基本的なやり方です。この節では、それほど検証されていないが、ベータ機能として実装されている、物理デバイス・ポートの動的設定方法を説明します。

一般的に Kamuee ポートを起動してフレーム転送を開始する場合には、以下の 4 つの手順が必要となります。

1. デバイスを DPDK PMD に bind する
2. デバイスに関連してポートを生成する
3. ポートのコンフィグレーションコマンドを投入する
4. ポートを実際に起動する

一方、フレーム転送を終了し、Kamuee ポートを停止する場合には、以下の 3 つの手順が必要となります。

5. ポートを実際に停止する
6. ポートを削除する
7. デバイスを DPDK PMD から unbind する

本件では、必須の新規追加 CLI として、上記手順のうち (2), (4), (5), (6) についての機能を提供しています。また、オプションの新規追加 CLI として、手順 (1), (7) についての機能を提供しています。手順 (2) については、従来の Kamuee ポート関連 CLI コマンドをそのまま利用できるものとします。

4.5.1 例 1: 完全初期状態からのポート起動

物理デバイスが一切 DPDK PMD に bind されておらず、kamuee.conf も存在しない状態から kamuee サービスを起動し、ポート利用可能とする場合の CLI 設定例。

```
kamuee> device bind 0000:05:00.0 igb_uio
kamuee> port attach 0000:05:00.0
kamuee> set thread 4 forwarder port 0 rx-queue 0
kamuee> set thread 5 forwarder port 0 rx-queue 1
kamuee> port start 0
```

4.5.2 例 2: kamuee.conf を引き継いだポート起動

物理デバイスが一切 DPDK PMD に bind されておらず、kamuee.conf に forwarder 設定だけは記述されている状態から kamuee サービスを起動し、ポート利用可能とする場合の CLI 設定例。

```
$ cat /etc/kamuee/kamuee.conf
set thread 4 forwarder port 0 rx-queue 0
set thread 5 forwarder port 0 rx-queue 1

kamuee> device bind 0000:05:00.0 igb_uio
kamuee> port attach 0000:05:00.0
kamuee> port start 0
```

4.5.3 例 3: kamuee.conf を無視してのポート起動

物理デバイスが一切 DPDK PMD に bind されておらず、kamuee.conf に forwarder 設定だけは記述されている状態から kamuee サービスを起動し、完全に新規ポートとして利用可能とする場合の例。

```
$ cat /etc/kamuee/kamuee.conf
set thread 4 forwarder port 0 rx-queue 0
set thread 5 forwarder port 0 rx-queue 1

kamuee> device bind 0000:05:00.0 igb_uio
kamuee> port attach 0000:05:00.0 default
kamuee> set thread 8 forwarder port 0 rx-queue 0
kamuee> set thread 9 forwarder port 0 rx-queue 1
kamuee> port start 0
```

4.5.4 例 4: ポート停止

kamuee.conf の設定、もしくは port attach/port start コマンドによって起動中のポートを停止させ、物理デバイスの DPDK PMD を unbind する場合の例。

```
kamuee> port stop 0
kamuee> port detach 0
kamuee> device unbind 0000:05:00.0
```

4.5.5 例 5: ポート強制停止

kamuee.conf の設定、もしくは port attach/port start コマンドによって起動中のポートが利用中であって停止させる場合の例。

```
kamuee> port detach 0 force
kamuee> device unbind 0000:05:00.0
```

4.5.6 例 6: 物理デバイス一覧表示

Kamuee から利用可能なすべての物理デバイスの一覧を表示する場合の例。以下の例は、物理デバイス 0000:05:00.0 が Kamuee の port 0 として利用される場合の表示内容です。

```
kamuee> show devices
```

id	device	name	driver	unused
-	0000:03:00.0	Ethernet Controller 10-Gigabit X540-AT2	ixgbe	igb_uio
-	0000:03:00.1	Ethernet Controller 10-Gigabit X540-AT2	ixgbe	igb_uio
0	0000:05:00.0	Ethernet Controller 10-Gigabit X540-AT2	igb_uio	ixgbe
-	0000:05:00.1	Ethernet Controller 10-Gigabit X540-AT2	ixgbe	igb_uio
-	0000:08:00.0	I350 Gigabit Network Connection	igb	igb_uio
-	0000:08:00.1	I350 Gigabit Network Connection	igb	igb_uio

4.6 Jumbo フレームと MTU の設定

Jumbo フレームをポートに設定するには、次のように **set port** コマンドにより MTU の値を **9140** で設定します。

```
kamuee-vty[0] > set port 0 jumbo-frame
kamuee-vty[0] > set port 0 mtu 9140
kamuee-vty[0] > set port 2 jumbo-frame
kamuee-vty[0] > set port 2 mtu 9140
kamuee-vty[0] > set port 4 jumbo-frame
kamuee-vty[0] > set port 4 mtu 9140
```

Jumbo フレームを vport (VLAN インターフェース) に設定する場合は、MTU の値を **9118** で設定します。次の例では vport 0 と vport 1 の MTU を 9118 に設定しています。

```
kamuee-vty[0] > set vport 0 mtu 9118
kamuee-vty[0] > set vport 1 mtu 9118
```

設定した MTU 値は **ip link** コマンドでベース OS 側でも確認できます。

```

kamuee@kamuee:~$ ip link
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode_
↳DEFAULT group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
3: eno1: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN mode DEFAULT_
↳group default qlen 1000
    link/ether 70:e2:84:09:f1:0c brd ff:ff:ff:ff:ff:ff
4: eno2: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP mode_
↳DEFAULT group default qlen 1000
    link/ether 70:e2:84:09:f1:0d brd ff:ff:ff:ff:ff:ff
6: port-2-0-0: <BROADCAST,MULTICAST> mtu 9140 qdisc noop state DOWN mode_
↳DEFAULT group default qlen 1000
    link/ether a0:36:9f:ba:3b:9c brd ff:ff:ff:ff:ff:ff
7: port-2-0-1: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN mode_
↳DEFAULT group default qlen 1000
    link/ether a0:36:9f:ba:3b:9d brd ff:ff:ff:ff:ff:ff
8: vlan0040: <BROADCAST,MULTICAST> mtu 9118 qdisc noop state DOWN mode_
↳DEFAULT group default qlen 1000
    link/ether b2:cd:8a:d8:c3:ae brd ff:ff:ff:ff:ff:ff
9: vlan0041: <BROADCAST,MULTICAST> mtu 9118 qdisc noop state DOWN mode_
↳DEFAULT group default qlen 1000
    link/ether 4a:fa:2f:f8:fb:3e brd ff:ff:ff:ff:ff:ff
kamuee@kamuee:~$

```

4.7 論理インターフェイスの確認と設定

論理インターフェイス（vport）を確認するには **show vport** コマンドを実行します。

```

kamuee[vty0]> show vport
vport      id tap mac-addr      drvr  phy-name    phy vlan
vlan0040    0  8 B2:CD:8A:D8:C3:AE vlan  port-2-0-0    0  40
kamuee[vty0]>

```

特定の vport の詳細を確認するには **show vport** コマンドで vport ID を指定します。

```

kamuee[vty0]> show vport 0
vport0:
  name: vlan0040
  new name: vlan0040
  vrf: 0 (vrf0)
  flags: <OURS-TAP|MACADDR-RANDOM>
  kni ifindex: -1
  tap ifindex: 8
  tap sockfd: 120
  tx_ol_flags: <>
  hwaddr: B2:CD:8A:D8:C3:AE
  vlan-id: 40
  physical port: 0
  ifaddrs[ipv4]:

```

(次のページに続く)

(前のページからの続き)

```

    ifaddrs[ipv6]:
    mirror port:      none
    tapmirror-rx:     no
    tapmirror-tx:     no
kamuee[vty0]>

```

全ての vport の詳細を確認するには **show vport all** コマンドを実行します。

```

kamuee[vty0]> show vport all
vport0:
  name: vlan0040
  new name: vlan0040
  vrf: 0 (vrf0)
  flags: <OURS-TAP|MACADDR-RANDOM>
  kni ifindex: -1
  tap ifindex: 8
  tap sockfd: 120
  tx_ol_flags: <>
  hwaddr: B2:CD:8A:D8:C3:AE
  vlan-id: 40
  physical port: 0
  ifaddrs[ipv4]:
  ifaddrs[ipv6]:
  mirror port:      none
  tapmirror-rx:     no
  tapmirror-tx:     no
vport1:
  name: vlan0041
  new name: vlan0041
  vrf: 0 (vrf0)
  flags: <OURS-TAP|MACADDR-RANDOM>
  kni ifindex: -1
  tap ifindex: 9
  tap sockfd: 121
  tx_ol_flags: <>
  hwaddr: 4A:FA:2F:F8:FB:3E
  vlan-id: 41
  physical port: 0
  ifaddrs[ipv4]:
  ifaddrs[ipv6]:
  mirror port:      none
  tapmirror-rx:     no
  tapmirror-tx:     no
kamuee[vty0]>

```

4.8 VLAN インターフェースの設定

kamuee ではパケットフォワーディングに DPDK を利用しているため、VLAN の設定も Linux 標準の設定ではなく DPDK 側の設定を行う必要があります。kamuee では DPDK の VLAN 設定を行うためのインターフェ

イスを提供しています。ここでは、この設定について解説します。

kamuee 上では VLAN インターフェースは **vport** として扱われます。vport への VLAN 番号割り当ては **set vport** コマンドで設定します。次の例では、ポート **0** 番に対して、VLAN 番号 **40** を認識する vport ID **0** 番の vport を作成しています。

```
kamuee[vty0]> set vport 0 physical-port 0 vlan 40
```

設定した vport は **ip link** コマンドでベース OS 側でも VLAN インターフェースとして確認できます。次の出力例では **vlan0040**、**vlan0041** というインターフェース名で表示されています。

```
kamuee@kamuee:~$ ip link
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode_
↳DEFAULT group default qlen 1000
   link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
3: eno1: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN mode _
↳group default qlen 1000
   link/ether 70:e2:84:09:f1:0c brd ff:ff:ff:ff:ff:ff
4: eno2: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP mode_
↳DEFAULT group default qlen 1000
   link/ether 70:e2:84:09:f1:0d brd ff:ff:ff:ff:ff:ff
6: port-2-0-0: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN mode_
↳DEFAULT group default qlen 1000
   link/ether a0:36:9f:ba:3b:9c brd ff:ff:ff:ff:ff:ff
7: port-2-0-1: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN mode_
↳DEFAULT group default qlen 1000
   link/ether a0:36:9f:ba:3b:9d brd ff:ff:ff:ff:ff:ff
8: vlan0040: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN mode_
↳DEFAULT group default qlen 1000
   link/ether b2:cd:8a:d8:c3:ae brd ff:ff:ff:ff:ff:ff
9: vlan0041: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN mode_
↳DEFAULT group default qlen 1000
   link/ether 4a:fa:2f:f8:fb:3e brd ff:ff:ff:ff:ff:ff
kamuee@kamuee:~$
```

OS 側に公開された VLAN インターフェースは、**iproute2** コマンドや **FRR** の論理インターフェースとして利用可能です。

4.9 IP アドレスの設定と確認

4.10 IP 経路の設定と変更

kamuee 側で DPDK の RIB, FIB に直接経路設定を書き込むことで、ベース OS のカーネルやルーティングデーモンには届かない形で、経路設定を強制的に変更できます。緊急避難的なブラックホールルーティングの設定に利用できます。

経路変更を設定する場合は、**ip route** コマンドを実行します。下記は、10.0.0.0/24 の経路を /dev/null に設定するブラックホールルーティングの例です。

```
kamuee-vty[0] > ipv4 route 10.0.0.0/24 ???? (要確認)
```

経路変更を解除する場合は、下記のコマンドを実行します。

```
kamuee-vty[0] > no ipv4 route 10.0.0.0/24 ???? (要確認)
```

ECMP (Equal Cost Multipath) を確認したい場合は、次のコマンドで表示できます。

```
kamuee-vty[0] > show ipv4 route multipath
```

IPv6 の場合は次のようになります。

```
kamuee-vty[0] > show ipv6 route multipath
```

マルチキャストの経路を確認したい場合は、次のコマンドで表示できます。

```
kamuee-vty[0] > show ipv4 route type multicast
```

```
kamuee-vty[0] > show ipv6 route type multicast
```

IP アドレス設定と経路設定は、基本的には **ip** コマンドやサードパーティの経路制御ソフトウェアから行います。基本的には kamuee が自動的に OS に設定された経路情報を読み込み kamuee 側の経路テーブルにも利用します。すなわち、ここまでの設定が完了していれば、**ip route** や **quagga FRR** 等の経路制御ソフトを普通の Linux ルータ同様に設定するだけで、kamuee の高速ルーティングは動作します。本節では例として **iproute2** を利用した方法と、kamuee に入力された経路の確認方法を解説します。

はじめに **ip** コマンドを使ったスタティックルートに対する kamuee の状態を説明します。まず、ベース OS 側で **ip** コマンドでインターフェースの IPv4 アドレスとスタティックルートを設定します。

```
kamuee@kamuee:~# ip addr add 192.168.10.1/24 dev port-2-0-0
kamuee@kamuee:~# ip route add 172.16.20.0/24 via 192.168.10.1
```

上記のコマンドを Ubuntu 上で実行すると同時に kamuee 側でも設定された経路が確認できます。**show ipv4 route** コマンドで設定した経路情報を確認できます。このように経路情報が確認できていれば kamuee 占有として割り当てたポートに入力されたパケットは、経路表に従い転送されます。

```
kamuee[vty0]> show ipv4 route
vrf[0]: vrf0
Destination      Nexthop          MAC-address      I/F              <Flags>
127.0.0.0/8      0.0.0.0          00:00:00:00:00:00 port-2-0-0       <OURS>
127.0.0.0/32     0.0.0.0          00:00:00:00:00:00 port-2-0-0       <BLACKHOLE>
127.0.0.1/32     0.0.0.0          00:00:00:00:00:00 port-2-0-0       <OURS>
127.255.255.255/32 0.0.0.0          00:00:00:00:00:00 port-2-0-0       <BLACKHOLE>
172.16.20.0/24   192.168.10.1     00:00:00:00:00:00 port-2-0-0       <>
192.168.1.0/24   0.0.0.0          00:00:00:00:00:00 port-2-0-1       <CONNECTED>
192.168.1.0/32   0.0.0.0          00:00:00:00:00:00 port-2-0-1       <BLACKHOLE>
192.168.1.228/32 0.0.0.0          00:00:00:00:00:00 port-2-0-1       <OURS>
192.168.1.255/32 0.0.0.0          00:00:00:00:00:00 port-2-0-1       <BLACKHOLE>
```

(次のページに続く)

(前のページからの続き)

```

192.168.10.0/24    0.0.0.0          00:00:00:00:00:00 port-2-0-0 <CONNECTED>
192.168.10.0/32    0.0.0.0          00:00:00:00:00:00 port-2-0-0 <BLACKHOLE>
192.168.10.1/32    0.0.0.0          00:00:00:00:00:00 port-2-0-0 <OURS>
192.168.10.2/32    192.168.10.2     80:2A:A8:9C:AE:99 port-2-0-0 <>
192.168.10.255/32  0.0.0.0          00:00:00:00:00:00 port-2-0-0 <BLACKHOLE>
vrf[0]: vrf0: routes: 19 from 19
kamuee[vty0]>

```

複数の IPv4 経路を設定している場合で、詳細情報が必要ない場合は、**show ipv4 route summary** コマンドでサマリを表示できます。

```

kamuee[vty0]> show ipv4 route summary
vrf[0]: vrf0
vrf[0]: vrf0: afi: ipv4: routes: 19 from 19
  multipath: 0 routes
  route_type: CONNECTED: 2 routes
  route_type: REJECT: 0 routes
  route_type: BLACKHOLE: 6 routes
  route_type: DISABLE: 0 routes
  route_type: OURS: 4 routes
  route_type: MPATH: 0 routes
  route_type: MCAST: 0 routes
  route_type: NORMAL: 7 routes
  route to port[0]: port-2-0-0: 10 routes
  route to port[1]: port-2-0-1: 9 routes
kamuee[vty0]>

```

IPv6 のスタティックルートの設定も IPv4 の場合と同様に行えます。

```

kamuee@kamuee:~$ sudo ip addr add 2001:db8:0::1/64 dev port-2-0-0
kamuee@kamuee:~$ sudo ip route add 2001:db8:20::/64 via 2001:db8:10::2 dev
➡port-2-0-0

```

上記のコマンドを実行後、kamuee 側では **show ipv6 route** コマンドで設定した経路情報を確認します。

```

kamuee[vty0]> show ipv6 route
vrf[0]: vrf0
Destination                                Nexthop                                I/F                                <Flags>
::1/128                                    ::                                      port-2-0-0                        <OURS>
2001:db8:10::/64                          ::                                      port-2-0-0                        <CONNECTED>
2001:db8:10::1/128                        ::                                      port-2-0-0                        <OURS>
2001:db8:20::/64                          2001:db8:10::2                       port-2-0-0                        <>
fe80::2:56dd:3875:d806/128                18:F1:D8:62:46:22                    port-2-0-1                        <>
fe80::80b:dbff:fe83:c12b/128               0A:0B:DB:83:C1:2B                    port-2-0-1                        <>
fe80::822a:a8ff:fe9c:ae99/128              80:2A:A8:9C:AE:99                    port-2-0-0                        <>
fe80::822a:a8ff:fe9e:4e61/128              80:2A:A8:9E:4E:61                    port-2-0-1                        <>
fe80::a236:9fff:feba:3b9c/128              00:00:00:00:00:00                    port-2-0-0                        <OURS>
fe80::a236:9fff:feba:3b9d/128              00:00:00:00:00:00                    port-2-0-1                        <OURS>
fe80::f452:53ff:fee3:e348/128              00:00:00:00:00:00                    vlan0040                          <OURS|VPORT>
fe80::f4cc:ba15:9c8a:2498/128              E4:70:B8:E2:36:E1                    port-2-0-1                        <>

```

(次のページに続く)

(前のページからの続き)

```
fe80::f830:85ff:fe50:1b85/128 00:00:00:00:00:00 vlan0041 <OURS|VPORT>
vrf[0]: vrf0: routes: 13 from 13
kamuee[vty0]>
```

IPv4 の場合と同様に詳細情報が必要ない場合は、**show ipv6 route summary** コマンドでサマリを表示できます。

```
vrf[0]: vrf0: afi: ipv6: routes: 13 from 13
  multipath: 0 routes
  route_type: CONNECTED: 1 routes
  route_type: REJECT: 0 routes
  route_type: BLACKHOLE: 0 routes
  route_type: DISABLE: 0 routes
  route_type: OURS: 6 routes
  route_type: MPATH: 0 routes
  route_type: MCAST: 0 routes
  route_type: NORMAL: 6 routes
  route to port[0]: port-2-0-0: 6 routes
  route to port[1]: port-2-0-1: 5 routes
  route to vport[0]: vlan0040: 1 routes
  route to vport[1]: vlan0041: 1 routes
kamuee[vty0]>
```

IPv4 と IPv6 の経路情報サマリを一括で表示したい場合は、**show ip route summary** コマンドを実行します。

```
kamuee[vty0]> show ip route summary
vrf[0]: vrf0
vrf[0]: vrf0: afi: ipv4: routes: 19 from 19
  multipath: 0 routes
  route_type: CONNECTED: 2 routes
  route_type: REJECT: 0 routes
  route_type: BLACKHOLE: 6 routes
  route_type: DISABLE: 0 routes
  route_type: OURS: 4 routes
  route_type: MPATH: 0 routes
  route_type: MCAST: 0 routes
  route_type: NORMAL: 7 routes
  route to port[0]: port-2-0-0: 10 routes
  route to port[1]: port-2-0-1: 9 routes
vrf[0]: vrf0: afi: ipv6: routes: 13 from 13
  multipath: 0 routes
  route_type: CONNECTED: 1 routes
  route_type: REJECT: 0 routes
  route_type: BLACKHOLE: 0 routes
  route_type: DISABLE: 0 routes
  route_type: OURS: 6 routes
  route_type: MPATH: 0 routes
  route_type: MCAST: 0 routes
  route_type: NORMAL: 6 routes
```

(次のページに続く)

(前のページからの続き)

```

route to port[0]: port-2-0-0: 6 routes
route to port[1]: port-2-0-1: 5 routes
route to vport[0]: vlan0040: 1 routes
route to vport[1]: vlan0041: 1 routes
kamuee[vty0]>

```

🕒 ワンポイント・アドバイス

quagga や FRR などを用いて経路設定を行った場合も、同様の手順で経路情報を kamuee 側で確認できます。

4.11 VRF 設定

kamuee は Linux のユーザスペースプログラムとして実装されているため、管理・マネージメント用の IP アドレス・経路設定は、Linux のデフォルト経路設定として実現すべきです。こうしておけば、kamuee や FRR の不具合からアドレスや経路が消えて Linux ホストにアクセスできなくなることを防げます。

また、kamuee ルータのアドレス・経路設定と、Linux のアドレス・経路設定を分離するために、kamuee には別の VRF を割り当てるべきです。こうすることで、kamuee がクラッシュしたときに、ルータのトラフィックが Linux OS によって Linux ホストのマネージメント用 I/F からホスト用デフォルト経路に誤って転送してしまうことなどを防げます。

現状では、kamuee の機能未実装などの関係から、少々複雑な起動方法・設定の順序を取らなくてはなりません。

Linux ホストで別のユーザプログラムの経路制御に利用するため、kamuee は物理・論理ポートに一対一対応する TUN/TAP I/F を Linux ホストに作成します。これが、port-X-X-X の I/F や、vlanXXXX の I/F です。これらの I/F は kamuee 起動後に kamuee が作成するため、まずは kamuee を起動しなくてはなりません。kamuee の起動は、通常通りであり、特に追加設定は必要ありません。

```
# systemctl start kamuee
```

その後、iproute2 で VRF m3dev の作成と、上記 kamuee 配下の TUN/TAP I/F (e.g., port-X-X-X) の VRF への紐付け (enslave) を実行します。これが必要になる理由は、1) kamuee にこの機能が未実装であること、2) FRR が interface の VRF の動的変更に未対応であること、の二つです。FRR を起動するまえに iproute2 で VRF の設定を実行しましょう。

```
# sh -x ip_vrf.sh
```

```

# cat ip_vrf.sh
ip link add router type vrf table 100
ip link set router up

ip link set port-3b-0-0 vrf router
ip link set port-5e-0-0 vrf router

```

(次のページに続く)

(前のページからの続き)

```
ip link set port-86-0-0 vrf router
ip link set port-af-0-0 vrf router
```

その後、FRR を実行します。

```
# systemctl start frr
```

FRR の設定では、interface コマンド、ip route コマンド、bgp コマンドのそれぞれに、vrf を指定する必要があります。ip forwarding は no に設定しておいた方が安全です。Linux ホストはパケットを転送せず、kamuee のみにパケットを転送させるという動作が実現できます。

```
# cat /etc/frr/frr.conf
frr version 7.2
    : (略)
no ip forwarding
    : (略)
!
vrf router
    ip route 0.0.0.0/0 192.85.2.4
    exit-vrf
!
    : (略)
interface port-3b-0-0 vrf router
    ip address 192.85.2.1/24
    : (略)
router bgp 65000 vrf router
    bgp router-id 10.1.1.1
    neighbor 192.85.1.4 remote-as 1
    neighbor 192.85.2.4 remote-as 2
    : (略)
```

FRR では、show bgp vrf router summary など、vrf を指定して情報を確認できます。

Kamuee では、show ipv4 route で、全 vrf の経路が表示されます。

iproute2 では、ip route show vrf router など、vrf を指定する必要があります。

確認とモニタリング

5.1 トラフィック統計

ポートの転送パケット数（pps）を確認したい場合は、**show port statistics all pps** コマンドを実行します。

```
kamuee[vty0]> show port statistics all pps
I/F          ipackets          ierrs opackets          oerrs imiss  nombuf
port-2-0-0    0                  0      0                  0      0      0
port-2-0-1    28                 0      0                  0      0      0
total         28                 0      0                  0      0      0
kamuee[vty0]>
```

CPU スレッドでの転送パケット数（pps）を確認したい場合は、**show thread statistics pps** コマンドで確認できます。

title	rxframes rxpackets drop	txframes txpackets noroute	discard forwarded malformed	ours filtered
core[2]:	0	0	0	
	0	0	0	0
	0	0	0	0
core[3]:	0	0	0	
	0	0	0	0
	0	0	0	0
core[4]:	0	0	0	
	0	0	0	0
	0	0	0	0

CPU スレッドの情報を確認したい場合は、**show thread info** コマンドで確認できます。

```
kamuee[vty0]> show thread info
core[id]: v thread name          lcpu pcpu port  rxq funcp
core[0]:  1 (null)                0    0    0    0 (nil)
core[1]:  1 OS                    0    --    --    -- (nil)
core[2]:  1 lthread_scheduler     0    --    --    -- 0x55da75882840
```

(次のページに続く)

(前のページからの続き)

```

core[3]:  1 forwarder          0  --  --  -- 0x55da757a5540
core[4]:  1 forwarder          0   0   0   0 0x55da757a5540
core[5]:  1 forwarder          0   0   0   1 0x55da757a5540
core[6]:  1 rib_manager        1  --  --  -- 0x55da7577c640
core[7]:  1 tap_manager        1  --  --  -- 0x55da75817a50
core[8]:  1 forwarder          1   0   0   2 0x55da757a5540
core[9]:  1 forwarder          1   0   0   3 0x55da757a5540
core[10]: 1 forwarder          1   0   1   0 0x55da757a5540
core[11]: 1 forwarder          1   0   1   1 0x55da757a5540
core[12]: 1 forwarder          0   0   1   2 0x55da757a5540
core[13]: 1 forwarder          0   0   1   3 0x55da757a5540
core[14]: 1 forwarder          0  --  --  -- 0x55da757a5540
core[15]: 1 forwarder          0  --  --  -- 0x55da757a5540
core[16]: 1 forwarder          0  --  --  -- 0x55da757a5540
core[17]: 1 forwarder          0  --  --  -- 0x55da757a5540
core[18]: 1 forwarder          1  --  --  -- 0x55da757a5540
core[19]: 1 forwarder          1  --  --  -- 0x55da757a5540
core[20]: 1 forwarder          1  --  --  -- 0x55da757a5540
core[21]: 1 forwarder          1  --  --  -- 0x55da757a5540
core[22]: 1 OS                 1  --  --  -- (nil)
core[23]: 1 OS                 1  --  --  -- (nil)
kamuee[vty0]>

```

5.2 トラフィックモニタリング機能

5.2.1 ポートミラー設定

kamuee では、ポートミラーを設定することもできます。別のポートに出力する場合は、次のコマンドを実行します。

```
kamuee-vty[0] > set port 0 mirror port 1
```

ベース OS の tap インターフェースに出力する場合は、次のように設定します

```
kamuee-vty[0] > set port 0 mirror tap mon0
```

設定後は、ベース OS 側で tcpdump コマンドなどにてパケットキャプチャが行えるようになります。

```
localhost > sudo tcpdump -nli mon0
```

5.3 ログ機能

5.3.1 ログ出力情報の変更 (★1)

下記のコマンドを実行すると、kamuee で全ての転送情報がログに表示されます。

```
kamuee-vty[0] > info all
```

下記のコマンドを実行すると、kamuee の FIB (Forwarding Information Base) 内の情報がログに表示されます。

```
kamuee-vty[0] > info fib
```

下記のコマンドを実行すると、kamuee の RIB (Routing Information Base) 内の情報がログに表示されます。

```
kamuee-vty[0] > info rib
```

また、デバッグ情報を出力したい場合には下記のコマンドを実行してください。

```
kamuee-vty[0] > debug trace
```

5.3.2 経路の再計算 (★ 2)

手動にて、`poptrie` を使った経路の再計算を行うことができます。下記のコマンドを実行してください。

```
kamuee-vty[0] > ip fib update
```

下記のように **info all** を実行後に再計算を行うと、再計算にかかった時間を **SYSLOG** に出力します。

```
kamuee-vty[0] > info all  
kamuee-vty[0] > ip fib update
```


6.1 マルチキャストルーター機能

6.2 仮想環境での利用

virtio-net での動作を確認しています。

virtio-net のドライバの仕様から、memory leak が起きて長期安定運用できない不具合が確認されています。その場合には、以下のように、rxdesc の値を 256 固定にすると、memory leak が解消されます。

```
set port 0 nrxdesc 256
set port 0 ntxdesc 256
set port 1 nrxdesc 256
set port 1 ntxdesc 256
```

- 物理 NIC: intel 82599
- VM 上。VM とは OVS-DPDK 経由 (vhost-user)
- ethtool -i: driver:virtio_net

コマンドリファレンス

7.1 port attach: ポート生成機能

- CLI 形式

```
port attach DEVICE (|restore|default)
```

- 説明

本機能は、DPDK のドライバにバインドされた物理デバイスを Kamuee のポートとして利用できるように、Kamuee 内部のデータ構造などを初期化して準備します。

- 処理

- 引数 DEVICE に指定されたデバイスを Kamuee ポートとして生成します。
- 指定されたデバイスが以前にも使用されていたものであれば、前回のポート設定をそのまま引き継ぐ (restore) か、新規ポートとしてデフォルト設定で再初期化する (default) かを選択可能とします。

- 引数

- 引数 DEVICE 部には 物理 PCI デバイス識別子 (BDF 形式文字列) を指定します。
- 引数 restore は、前回起動時のポート設定をそのまま引き継ぐことを指示します。
- 引数 default は、前回起動時のポート設定は引き継がず、新規ポートとして再初期化することを指示します。
- 引数 restore, default のいずれも省略された場合の処理は、前回ポート起動時の設定の有無によって動的に判断します。

- 前提条件

- 物理 PCI デバイスのドライバは事前に DPDK PMD に bind 済であると想定します (Kamuee の device bind コマンドや DPDK に付属の外部コマンド dpdk-devbind.py の利用を想定します)。

- 新規ポート ID(ポート番号) は、DPDK により動的に配番された値を Kamuee でも採用するものとします。
 - すでに Kamuee にアタッチ済のポートについては何も処理を行いません。
 - 引数 DEVICE に指定するデバイスアドレス文字列は DPDK API 関数 `rte_pci_addr_parse()` によって解釈可能な形式に限ります。
 - 実際のポート起動前にポートコンフィグレーション用コマンドを投入できるように、ポートの起動は `port start` コマンドによって別途行うものとします。
- 備考
 - ポートを再初期化して生成した場合、`port start` 前に最低でも `fowarder` スレッドの設定だけは投入し直す必要があります。
 - DPDK が空いているポート番号から若番順に配番するため、同じ物理デバイスであってもポート番号が変わる可能性があります。

7.2 port start: ポート起動機能

- CLI 形式

```
port start PORTID
```

- 説明

本機能は、生成済み Kamuee のポートの利用を実際に開始します。本機能を使用してはじめてフレームの転送処理が開始されます。

- 処理
 - 引数 `PORTID` に指定されたポートの利用を開始します。
- 引数
 - 引数 `PORTID` 部には Kamuee ポート番号を指定します。
- 前提条件
 - `port attach` コマンドによってアタッチ済みのポートのみを処理対象とします。
 - すでに起動済のポートについては何も処理を行いません。
- 備考
 - 内部スレッドの起動を伴う点で `port no shutdown` コマンドとは異なります。
 - ポート毎の `statistics` 情報はゼロクリアされます。

7.3 port stop: ポート停止機能

- CLI 形式

```
port stop PORTID
```

- 説明

本機能は、起動状態 Kamuee のポートを停止します。本機能を使用するとフレームの転送処理が停止し、ポートの削除ができるようになります。

- 処理

- 引数 PORTID に指定されたポートの利用を停止します。

- 引数

- 引数 PORTID 部には Kamuee ポート番号を指定します。

- 前提条件

- startup 時に起動されたポートも含め、すべての起動済ポートが処理対象です。
- すでに停止中のポートについては何も処理を行いません。

- 備考

- 内部スレッドの停止を伴う点で port shutdown コマンドとは異なります。

7.4 port detach: ポート削除機能

- CLI 形式

```
port detach PORTID (force|)
```

- 説明

本機能は、Kamuee ポートとして使用中の物理デバイスの利用を終了し、内部データなどを解放します。

- 処理

- 引数 PORTID に指定されたポートを削除 (Kamuee の管理下から除外) します。

- 引数

- 引数 PORTID 部には Kamuee ポート番号を指定します。
- force オプションを付加した場合、起動中のポートも強制的に削除します。

- 前提条件

- force オプションを付加しない限り、port stop コマンドによって停止済みのポートだけが処理対象となります。

- ポートに関連するルーティング情報なども連動してすべて削除されます。
- 備考
 - 物理 PCI デバイスへのドライバのバインド状態は変更しません。

7.5 show port config: ポート設定確認機能

- CLI 形式

```
show port config (PORTID|all|)
```

- 処理
 - ポートに投入済の設定を CLI コマンド形式で表示します。
- 引数
 - 引数 PORTID 部には Kamuee ポート番号を指定します。
 - 引数 PORTID を省略した場合、もしくは、PORTID に all を指定した場合は、すべてのポートの設定情報を表示します。
- 前提条件
 - 現在利用中 (running) のポートでなくても、CLI でなにか設定を投入してある (dirty な) ポートであれば表示対象となります。
- 備考
 - show port config の出力結果を kamuee.conf にコピー&ペーストしておけば、次回 Kamuee 起動時にそのまま適用可能です。
 - 将来的に Zebra の write memory コマンドのような使い方ができるように機能拡張する場合へのサブルーチンを準備する意味合いもあります。

7.6 show devices: デバイス一覧取得機能

- CLI 形式

```
show devices
```

- 処理
 - Kamuee ポートして利用可能なデバイスの一覧を表示します。(dpdk-devbind.py -status の出力結果と似た内容を表示)。
- 引数
 - 可変パラメータはありません。

7.7 device bind: デバイス bind 機能

- CLI 形式

```
device bind DEVICE (DRIVER|none)
```

- 説明

本機能は、システム内に存在する物理デバイスに対して DPDK のボールモードドライバ (PMD) をバインドし、Kamuee ポートとして利用できるようにします。

- 処理

- 指定されたデバイスにドライバをバインド/アンバインドします (dpdk-devbind.py –bind=DRIVER DEVICE と同等の処理を実行します)。

- 引数

- 引数 DEVICE 部には 物理 PCI デバイスの識別子 (BDF 形式文字列) を指定します。
- 引数 DRIVER 部にはドライバ名称 (典型的には igb_uio) を指定します。
- 引数 DRIVER 部分に none を指定した場合は unbind 処理を行います。

- 前提条件

- Kamuee ポートとして利用中のデバイスは再 bind しません。
- すでに同じドライバにバインド済の場合は再 bind しません。
- 指定のドライバとは違うドライバにバインド済の場合は、一旦 unbind した後 bind が実行されます。
- 引数 DEVICE に指定するデバイスアドレス文字列は DPDK 関数 rte_pci_addr_parse() によって解釈可能な形式に限られます。

7.8 device unbind: デバイス unbind 機能

- CLI 形式

```
device unbind DEVICE
```

- 説明

本機能は、PMD にバインド済みの物理デバイスのドライバをバインド解除します。

- 処理

- 指定されたデバイスのドライバをアンバインドします (dpdk-devbind.py –unbind DEVICE と同等)。

- 引数

- 引数 DEVICE 部には物理 PCI デバイスの識別子 (BDF 形式文字列) を指定します。

- 前提条件
 - Kamuee ポートとして利用中のデバイスは unbind しません。
 - Kernel モードポートとして利用中のデバイスも unbind しません。
 - 引数 DEVICE に指定するデバイスアドレス文字列は DPDK 関数 `rte_pci_addr_parse()` によって解釈可能な形式に限られます。

7.9 show device config: デバイス設定確認機能

- CLI 形式

```
show device config (all|)
```

- 説明

本機能は、現在実行中のポートやデバイスの状態を表示します。

- 処理
 - デバイスに関する設定を CLI コマンド形式で表示します。
- 引数
 - オプション引数 `all` を付加しない場合、Kamuee ポートとして利用中のデバイス設定だけを表示します。
 - オプション引数 `all` を付加した場合は、システム上で利用可能すべてのデバイス設定を表示します。
- 備考
 - `show device config` の出力結果を `kamuee.conf` にコピー&ペーストしておけば、次回 Kamuee 起動時にそのまま適用可能です。
 - 将来的に Zebra の `write memory` コマンドのような使い方ができるように機能拡張する場合へのサブルーチンを準備する意味合いもあります。

7.10 write file: コンフィグレーション保存機能

- CLI 形式

```
write (file|memory|terminal) (FILE|)
```

- 説明

本機能は、現在実行中のポートやデバイスの状態を Kamuee コマンドライン形式で保存し、次回 Kamuee 起動時に利用できるようにします。

- 処理

- 現在のポートコンフィグレーション状態を Kamuee CLI 形式でファイルや端末に書き出します。

- 引数

- 第二引数が `file` の場合は、`FILE` に情報を書き出す。`FILE` が省略された場合はデフォルトコンフィグレーションファイル (`/etc/kamuee/kamuee.conf`) に情報を書き出します。`kamuee.conf` がすでに存在する場合は、古いファイル内容が `/etc/kamuee/kamuee.conf.save` に保存されます。
- 第二引数が `memory` の場合は、デフォルトコンフィグレーションファイル (`/etc/kamuee/kamuee.conf`) に情報を上書きします。`kamuee.conf` がすでに存在する場合でも強制的に上書きされます。
- 第二引数が `terminal` の場合は、端末に情報を書き出します。

- 制限事項

- 現時点では `device` 設定、`port` 設定だけが情報として書き出されます。

アプライアンス製品例

8.1 設置と配線

表 8.1 は、kamuee を動作させるハードウェアの構成例です。本稿では 表 8.1 で示すハードウェアを例として設定方法を解説します。

表 8.1 ハードウェア構成例

名称	機種・型番	備考
筐体	FUJITSU PRIMERGY RX200 S8/D3302-A1	1U (高さ x 幅 x 奥行 = 4.3 cm x 43.11 cm x 76.88 cm)
CPU	Intel(R) Xeon(R) CPU E5-2620 v2	2.10GHz 6 コア 12 スレッド
メモリ		32 GB
ストレージ		800 GB
ネットワークインターフェース (内蔵インターフェース)	Intel(R) I350 Gigabit Network Connection 1521	1 Giga bit インターフェース x 2
ネットワークインターフェース (PCI スロット 1)	Intel(R) 10G X550T 1563	10 Giga bit インターフェース x 2
管理用 リモートコンソールポート	iRMC	

図 8.1 と 図 8.2 は 表 8.1 で示すハードウェアの前面と背面の写真です。図 8.3 は 図 8.2 のポート構成を図にしたものです。図 8.3 の内容を 表 8.2 にまとめます。



図 8.1 kamuee ハードウェア（前面）



図 8.2 kamuee ハードウェア（背面）

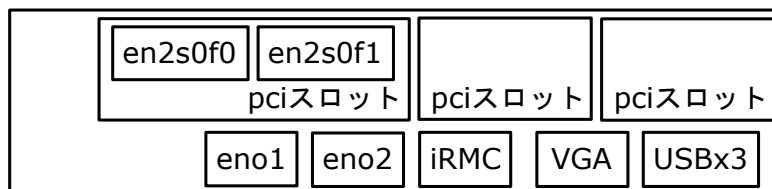


図 8.3 kamuee ハードウェア例の背面側ポート構成

表 8.2 ポート構成

ポート名	説明	備考
iRMC	リモートコンソール専用ポート	BIOS で IP アドレス設定が必要
eno1	1Gbps ネットワークインターフェース（内蔵）	予備
eno2	1Gbps ネットワークインターフェース（内蔵）	管理用ポート
enp2s0f0	10G-T ネットワークインターフェース（PCI スロット 2）	パケット転送ポート用
enp2s0f1	10G-T ネットワークインターフェース（PCI スロット 2）	パケット転送ポート用

iRMC は機器に内蔵されリモートコンソール専用ポートで、BIOS で IP アドレスを設定することで利用できます。**eno1** は予備ポートとします。**eno2** は kamuee 管理用の SSH ログイン用ポートとし DHCP の設定を行います。**enp2s0f0** と **enp2s0f1** はパケット転送用ポートとして設定します。

ラックマウント後 図 8.4 のように、**iRMC**、**eno2** に UTP を配線し、管理用のネットワークに接続します。また、初期設定をサーバから直接行う場合、VGA にモニタを接続し、USB ポートに USB キーボードを接続して下さい。



図 8.4 kamuee サーバ（配線後）

⚠（注意） USB キーボードの種類によっては、認識されないことがあります。その際には別のキーボードを利用してください。（試験環境では Lenovo のトラックポイント付き USB キーボードが利用できませんでした。）

8.2 起動とログイン

電源投入後、モニターにベース OS である Ubuntu のログイン画面が確認できます。初期アカウントとパスワードを入力してログインしてください。アカウント情報を次の表 8.3 に示します。

表 8.3 初期アカウント

アカウント名	権限	パスワード
kamuee	wheel	kamuee

⚠（注意） パスワードはログイン後に必ず変更してください。初期パスワードのまま運用することは潜在的なセキュリティホールになる可能性があります。

8.3 管理用ポートの IP アドレス設定

kamuee では、ルータのパケット転送に利用するポートは別途、占有することを指定します。占有指定していない場合には通常の Ubuntu で利用できる NIC として利用できます。本節では通常の Ubuntu NIC 設定について説明しますので、既知の場合には読み飛ばしてください。デフォルトでは、管理用ポートとして設定したいポートに **netplan** 経由で DHCP を設定し、SSH ログインなどに利用できます。管理用ポートをネットワークに接続後、**ip addr** コマンドで割当られた DHCP アドレスを確認します。

✎（メモ） **netplan** の設定ファイルは **/etc/netplan/01-netcfg.yaml** に配置されています。次の例では、**eno2** に DHCPv4 を設定しています。

```
# This file describes the network interfaces available on your system
# For more information, see netplan(5).
network:
  version: 2
  renderer: networkd
  ethernets:
    eno2:
      dhcp4: yes
```


固定 IP を割り当てる場合は、**/etc/netplan/01-netcfg.yaml** を次のような形で編集します。

```
network:
  version: 2
  renderer: networkd
  ethernets:
    eno2:
      dhcp4: no
      addresses: [10.0.2.100/24]
      gateway4: 10.0.2.1
      nameservers:
        addresses: [10.0.2.1]
```

編集後、**sudo netplan apply** コマンドを実行して適用します。

```
sudo netplan apply
```

設定の確認は **ip addr** コマンドで行います。

 **NTTCOM TODO** インストール済みサーバを出荷する場合、

- eno2 を常に mgmt にする
- 可変にしておき裏にテプララベルで「kamuee-mgmt」と貼ったポートが SSH 有効ポートですとマニュアルに書く

など考えられます。

ssh はデフォルトで有効になっています。ssh が起動していない場合は、下記のコマンドを実行してください。

```
sudo systemctl enable ssh
sudo systemctl start ssh
```

8.4 kamuee で使用するポート割り当て

kamuee で高速パケット転送を行うため、ネットワークインターフェイスを DPDK に登録します (kamuee で利用するよう占有指定します)。表 8.1 のハードウェア構成の場合、インターフェース名とその PCI 番号が表 8.4 のようになります。

表 8.4 インターフェース名と PCI 番号

インターフェース名	スピード	PCI 番号
eno1	1G	0000:07:00.0
eno2	1G	0000:07:00.1
enp2s0f0	10G/5G/2.5G/1G/100M	0000:02:00.0
enp2s0f1	10G/5G/2.5G/1G/100M	0000:02:00.1

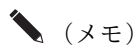
ここで、10G インターフェースの enp2s0f0 と enp2s0f1 をポートとして設定して、kamuee で管理するとし

ます。このとき、次のコマンドを実行して、DPDK の設定を行い、kamuee を **systemctl** コマンドで再起動します。

```
kamuee@kamuee:~$ sudo /usr/local/share/dpdk/usertools/dpdk-devbind.py -b igb_
↳uio 0000:02:00.0
kamuee@kamuee:~$ sudo /usr/local/share/dpdk/usertools/dpdk-devbind.py -b igb_
↳uio 0000:02:00.1
kamuee@kamuee:~$ sudo systemctl restart kamuee
```

実行後、**ip link** コマンドで確認すると、DPDK を設定したインターフェースは **port-2-0-1** のように **port-PCI** 番号 にリネームされています。

```
kamuee@kamuee:~$ ip link
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode_
↳DEFAULT group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
3: eno1: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN mode DEFAULT_
↳group default qlen 1000
    link/ether 70:e2:84:09:f1:0c brd ff:ff:ff:ff:ff:ff
4: eno2: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP mode_
↳DEFAULT group default qlen 1000
    link/ether 70:e2:84:09:f1:0d brd ff:ff:ff:ff:ff:ff
6: port-2-0-0: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN mode_
↳DEFAULT group default qlen 1000
    link/ether a0:36:9f:ba:3b:9c brd ff:ff:ff:ff:ff:ff
7: port-2-0-1: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN mode_
↳DEFAULT group default qlen 1000
    link/ether a0:36:9f:ba:3b:9d brd ff:ff:ff:ff:ff:ff
```



(メモ)

インターフェースの PCI 番号はハードウェア構成により変わります。PCI 番号は **dmesg** コマンドで確認できます。次は該当箇所の出力例です。

```
[ 19.618978] ixgbe 0000:02:00.0: PCI Express bandwidth of 32GT/s available
[ 19.718017] ixgbe 0000:02:00.0: (Speed:8.0GT/s, Width: x4, Encoding Loss:<2
↳%)
[ 19.931708] ixgbe 0000:02:00.0: MAC: 4, PHY: 0, PBA No: H86377-005
[ 20.033443] ixgbe 0000:02:00.0: a0:36:9f:ba:3b:9c
[ 20.296453] ixgbe 0000:02:00.0: Intel(R) 10 Gigabit Network Connection
[ 21.096840] ixgbe 0000:02:00.1: Multiqueue Enabled: Rx Queue count = 24,
↳Tx Queue count = 24 XDP Queue count = 0
```

🕒 NTTOM TODO

アプライアンスとしてインストール済みサーバを販売する場合、出荷に合わせて、dpdk-devbind.py を実行する Shell スクリプトファイルの設定を追記してください。(リブートすると DPDK の設定が消えてしまう)

8.5 ポートの確認と CPU 割り当て

2.4 節での kamuee へのポート割当てが正しく実施されているかを確認するには、**show port** コマンドを実行します。

```
kamuee[vty0]> show port
port          id c if tap mac-addr          drvr    speed supported
port-2-0-0    0 0 0  6 A0:36:9F:BA:3B:9C ixgbe    0 10G/5G/2.5G/1G/100M
port-2-0-1    1 0 0  7 A0:36:9F:BA:3B:9D ixgbe    0 10G/5G/2.5G/1G/100M
kamuee[vty0]>
```

特定のポートの詳細を確認するには **show port** コマンドでポートの ID を指定します。

```
kamuee[vty0]> show port 1
port1:
  new name: kni-2-0-1
  valid: 1
  vrf: 0 (vrf0)
  flags: <RUNNING|CHANGED|OURS-TAP>
  kni ifindex: -1
  tap ifindex: 7
  tap name: port-2-0-1
  tap sockfd: 119
  promiscuous:      1      allmulticast:      1
  nrxq:              4      ntxq:              24
  hwaddr: A0:36:9F:BA:3B:9D
  (割愛)
```

全ポートの詳細を確認するには **show port all** コマンドを実行します。

```
kamuee[vty0]> show port all
port0:
  new name: kni-2-0-0
  valid: 1
  vrf: 0 (vrf0)
  flags: <RUNNING|OURS-TAP>
  kni ifindex: -1
  tap ifindex: 6
  tap name: port-2-0-0
  tap sockfd: 118
  promiscuous:      1      allmulticast:      1
  nrxq:              4      ntxq:              24
  hwaddr: A0:36:9F:BA:3B:9C
  device info:
  (割愛)
```

kamuee では、CPU スレッドをポートの rx-queue に割り当てる必要があります。現在の割り当て状態を確認する場合は、次のように **show thread info** コマンドを実行してください。

```

kamuee[vty0]> show thread info
core[id]: v thread name          lcpu pcpu port  rxq funcp
core[0]:  1 (null)                0   0   0    0 (nil)
core[1]:  1 OS                    0  --  --   -- (nil)
core[2]:  1 lthread_scheduler      0  --  --   -- 0x56490aab7840
core[3]:  1 forwarder             0  --  --   -- 0x56490a9da540
core[4]:  1 forwarder             0   0   0    0 0x56490a9da540
core[5]:  1 forwarder             0   0   0    1 0x56490a9da540
core[6]:  1 rib_manager            1  --  --   -- 0x56490a9b1640
core[7]:  1 tap_manager            1  --  --   -- 0x56490aa4ca50
core[8]:  1 forwarder             1   0   0    2 0x56490a9da540
core[9]:  1 forwarder             1   0   0    3 0x56490a9da540
core[10]: 1 forwarder             1   0   1    0 0x56490a9da540
core[11]: 1 forwarder             1   0   1    1 0x56490a9da540
core[12]: 1 forwarder             0   0   1    2 0x56490a9da540
core[13]: 1 forwarder             0   0   1    3 0x56490a9da540
core[14]: 1 forwarder             0  --  --   -- 0x56490a9da540
core[15]: 1 forwarder             0  --  --   -- 0x56490a9da540
core[16]: 1 forwarder             0  --  --   -- 0x56490a9da540
core[17]: 1 forwarder             0  --  --   -- 0x56490a9da540
core[18]: 1 forwarder             1  --  --   -- 0x56490a9da540
core[19]: 1 forwarder             1  --  --   -- 0x56490a9da540
core[20]: 1 forwarder             1  --  --   -- 0x56490a9da540
core[21]: 1 forwarder             1  --  --   -- 0x56490a9da540
core[22]: 1 OS                    1  --  --   -- (nil)
core[23]: 1 OS                    1  --  --   -- (nil)
kamuee[vty0]>

```

ポートに CPU スレッドが割り当たっていない場合は、**set port** コマンドを実行します。CPU スレッドはポートの **nrqx** の数だけ割り当てることができます。次の例では、ポート 0 番の rx-queue 0 番から 3 番に対して論理 CPU 4 番から 7 番を割り当てる場合のコマンドです。

```

kamuee-vty[0]> set port 0 rx-queue 0 lcore 4
kamuee-vty[0]> set port 0 rx-queue 1 lcore 5
kamuee-vty[0]> set port 0 rx-queue 2 lcore 6
kamuee-vty[0]> set port 0 rx-queue 3 lcore 7

```

🔍 ヒント

CPU スレッドの割り当ては、ベース OS や各種マネージャにも割り当てることができます。kamuee の CPU スレッド設定は **/etc/kamuee/kamuee.conf** に保存されています。起動時にはこの設定ファイルを読み込んで自動的にルーティングの転送設定を DPDK に反映します。表 8.1 のハードウェア構成の場合のサンプル設定を次に示します。

```

set thread 2 lthread_scheduler
set thread 6 rib_manager
set thread 7 tap_manager

set port 0 rx-queue 0 lcore 4
set port 0 rx-queue 1 lcore 5

```

(次のページに続く)

(前のページからの続き)

```
set port 0 rx-queue 2 lcore 8
set port 0 rx-queue 3 lcore 9
set port 1 rx-queue 0 lcore 10
set port 1 rx-queue 1 lcore 11
set port 1 rx-queue 2 lcore 12
set port 1 rx-queue 3 lcore 13
```

⚠ (注意) CPU スレッド番号 6 は rib_manager に、CPU スレッド番号 7 は tap_manager に必ず割り当てられます。また、rib_manager、tap_manager に他の CPU スレッド番号を割り当てることはできません。