沖縄オープンラボラトリご紹介と kata containers で動かすfree5gc

2021/7/13 Open Mobile Network Infra Community Meetup#3

Yoshifumi Sumida

OKINAWA OPEN LABORATORY

自己紹介

- **名前**
  - 角田 佳史
- **所属**
  - NTT Communications
  - Okinawa Open Laboratory (常駐)
- **最近の業務**
  - (沖縄オープンラボラトリでの)
    モバイル関連OSS調査など
- **趣味**
  - 海遊び
  - 自然めぐり

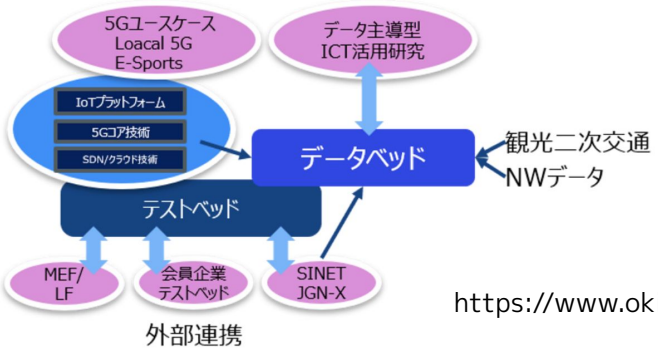
Okinawa Open Laboratory

SDNやNFVを皮切りに、IoTや5Gといった技術要素をベースに幅広く活動しています

沖縄オープンラボでは通信技術と情報技術を融合させたICT(Information and Communication Technology)基盤の研究と、そのICT基盤上に各種データを収集・蓄積し、活用するデータ利活用研究を行っています。

ICT基盤技術研究

沖縄オープンラボではオープンソースソフトウェア(OSS)を活用したICT基盤の研究開発、検証、実証を推進しています。ネットワークを仮想化し、ソフトウェアによる制御を可能とするSDN(Software-Defined Networking)やNFV(Network Functions Virtualization)、およびコンピュータ資源を仮想化する仮想マシンやコンテナ技術を用いてICT基盤の研究開発・検証実証を行ってきました。
2018年度からはIoT(Internet of Things)や第5世代無線技術（5G）といった新しい技術領域もOSSを活用して研究活動を行っています。
これらの活動で得られた研究成果や検証結果を公開する事で次世代ICT基盤技術の実用化・普及を促進しています。
また、検証・研究活動の基盤であるテストベッドを構築し、会員の皆様に提供しています。さらに現実世界に存在する様々なデータを収集・蓄積・活用する基盤としてデータベッドを構築し、データ利活用研究の基盤として運用しています。



https://www.okinawaopenlabs.org/research-development

OKINAWA OPEN LABORATORY

@OkinawaOpenLab

Okinawa Open Laboratory

様々な企業が集まり、様々なプロジェクトで連携しています

IoT Platform

- ➜ FIWARE/X-ROADの検証
- ➜ FaaSとの統合
- ➜ タクシー事業者との共同研究PJと連携
  （ドライブレコーダデータの分析）

Testbed

- ➜ 拠点間通信の改善
- ➜ Cephの利用 (データベッドでの活用)
- ➜ 仮想基盤サービスの提供
- ➜ テストベッドサービスのUI改善

5G

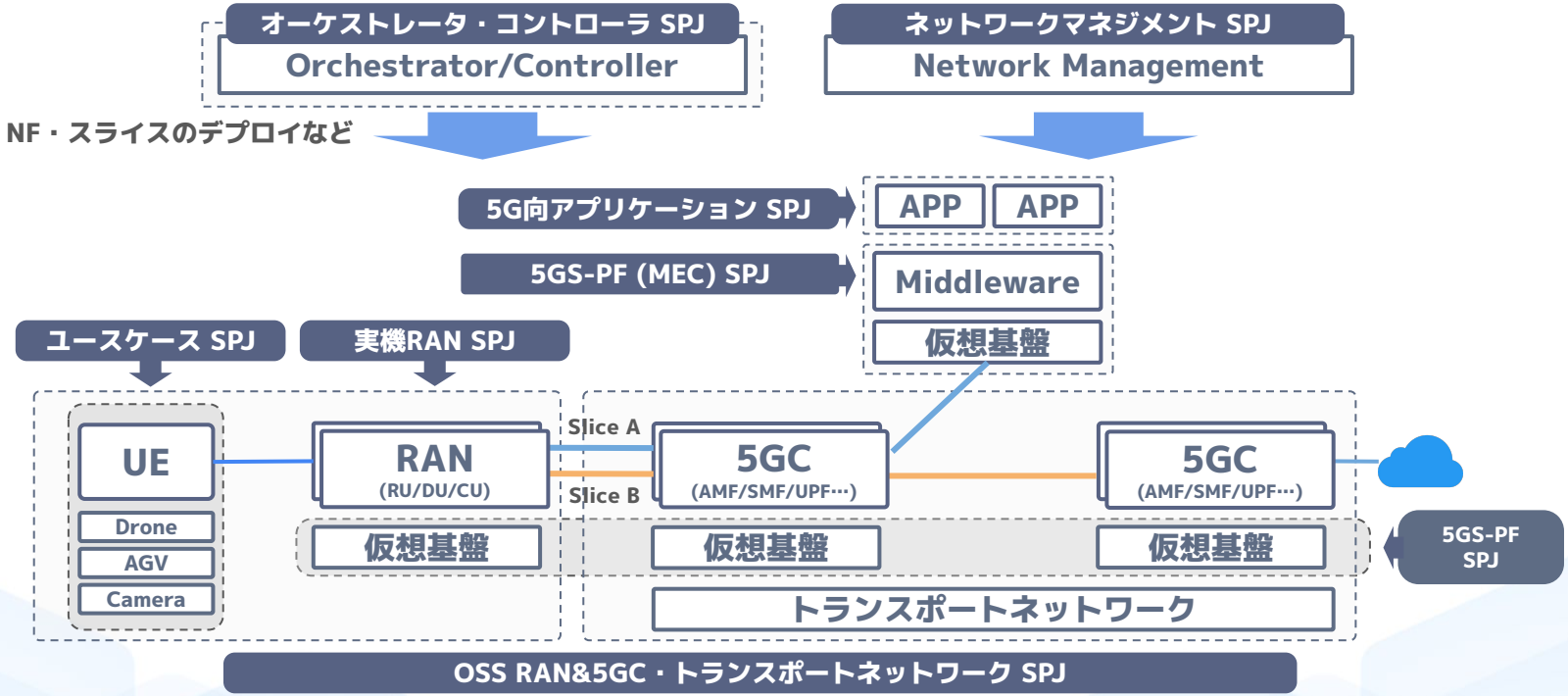
- ➜ 5Gコアネットワーク基盤の検証
  オープンソースのモバイル関連OSS検証
- ➜ 5Gユースケース (MEC/RAN) の検証
  ※ ローカル5Gもスコープに
- ➜ 外部組織との連携 (OMNI JPなど)

Databed

- ➜ データ利活用基盤 (データベッド) 整備
- ➜ データ利活用ユースケース研究
- ➜ 観光2次交通情報のオープンデータ化
- ➜ タクシー走行データ分析

Okinawa Open Laboratory: 5GPJ

技術ドメイン毎にサブプロジェクトを展開し、並行して検証を進めています



Okinawa Open Laboratory: 5GPJ

2020年度の実施内容の概要など

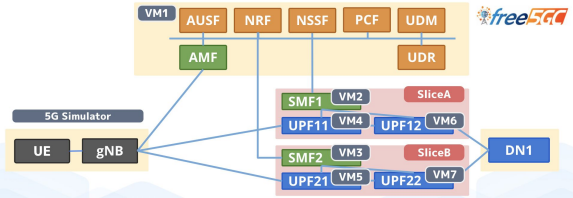
- **Free5GCを用いたOSS 5GC基盤の評価**
  - スライス識別子を用いた経路選択の機能に対応
  - QoS管理や運用管理に関する機能の不足
  - 実運用での利用はバグ修正、機能修正・追加などの対応が必要
- **IoT PJ と連携した擬似E2Eローカル5G基盤**
  - ベアメタルやコンテナ上でOSS 5GC基盤やMEC基盤を構築
  - 5GCやMECに適した基盤やコンポーネント再配置の仕組みの検討の必要性有り
- **MEC上での低遅延処理の実現に向けたコンテナベースMEC基盤の評価**
  - CentOS、Kubernetes (ContivVPP + DPDKを利用) を用いたMEC基盤を構築
  - ContivVPPによる安定したスループットなどの実現（Pod/Node間で10Gbps近くを確認）
  - K8S API との QoS 機能とスライス連携や複数NICでのQoSポリシ適用に未対応などの部分に課題有り
- **E2Eローカル5G基盤の管理・制御に向けたOSSオーケストレータ (ONAPを選択) の評価**
  - Closed-Loop のための自動化機構やAI拡張フレームワークなどの様々な機能に対応
  - 必要リソースの多さ、構築難易度の高さ、Helm関連リソースの習熟の必要性など構築・運用上の課題有り

Okinawa Open Laboratory: 5GPJ

5GC検証サブPJ

5GC・NWスライシング検証環境 (2)

- NWスライシングを想定しNFを複数VMで構築



NWスライシング検証サブPJ

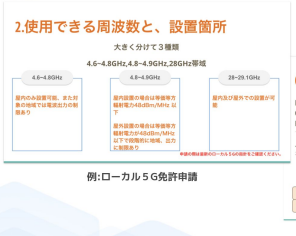
NWスライシング PJ 検証まとめ

- **検証成果**
  - スライス検証におけるリファレンス構成の明確化・対応パッチの開発
    Free5GC v3.0.4の実装状況の明確化
- **備考**
  - バグ報告の結果、Free5GC v3.0.5(2021/1/19リリース)ではスライス経路制御/バグが修正されている

検証項目	実装状況	対応	対応後の検証
スライス経路制御	△ 機能は実装されているがバグで動作しない	バグ修正パッチを作成して経路制御を動作可能とした	パッチで経路制御可能なことを確認
スライス品質保証	× 品質保証関連機能は実装されていない	UPF間にTCで帯域制御を実施することで品質保証を実施	スライス別に帯域制御可能なことを確認
スライス追加削除	× スライスの追加削除機能は実装されていない	コンフィグ再読み込み機能を実装することで対応	スライス追加削除が可能なことを確認

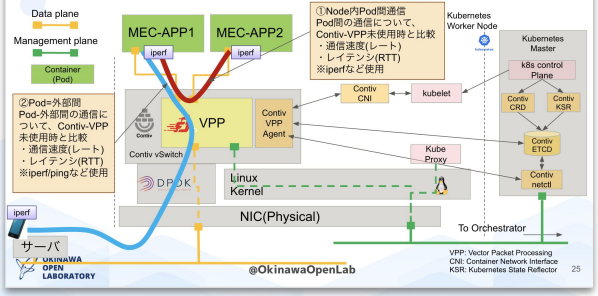
RAN検証サブPJ

OSS RAN PJ - 調査資料



MEC-PFサブPJ

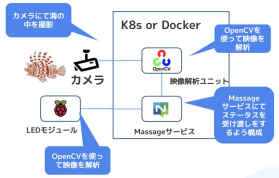
ContivVPPアーキおよび性能検証構成・内容



MECアプリサブPJ

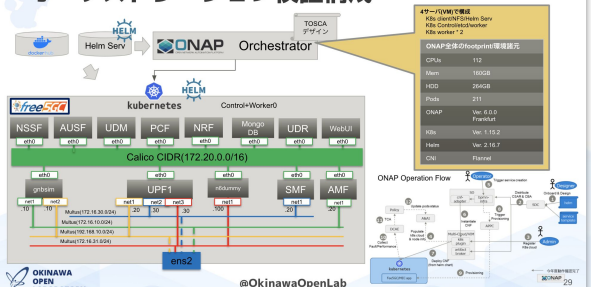
MECアプリケーション PJ

- MEC環境で利用できるサンプルプログラムを開発
- アプリの内容
  - 離島の海水浴場などの海中にいるミノカサゴといった危険生物を検知してLEDにて出現をお知らせする
- 動作概要
  - カメラに写っている映像を映像解析ユニットでOpenCVを使い解析する。その後、ミノカサゴを検知してMessageサービスにステータスを送信する
  - LEDモジュールはMessageサービスに送信されたステータスが変化するとLEDを光らせて知らせる仕組み
- MEFのPoCで使用
- サブプロジェクトの今後の内容はユースケースを含め検討中



オーケストレータ・コントローラサブPJ

オーケストレーション検証構成



Okinawa Open Laboratory

ご興味がある方は下記までご連絡ください😎

- **沖縄オープンラボラトリ・公式ウェブサイト**
  - https://www.okinawaopenlabs.org
- **Facebook**
  - https://www.facebook.com/okinawaopenlabs/
- **Twitter**
  - https://twitter.com/OkinawaOpenLab

kata-containerで動かすfree5gc

OKINAWA OPEN LABORATORY

Kata Containers

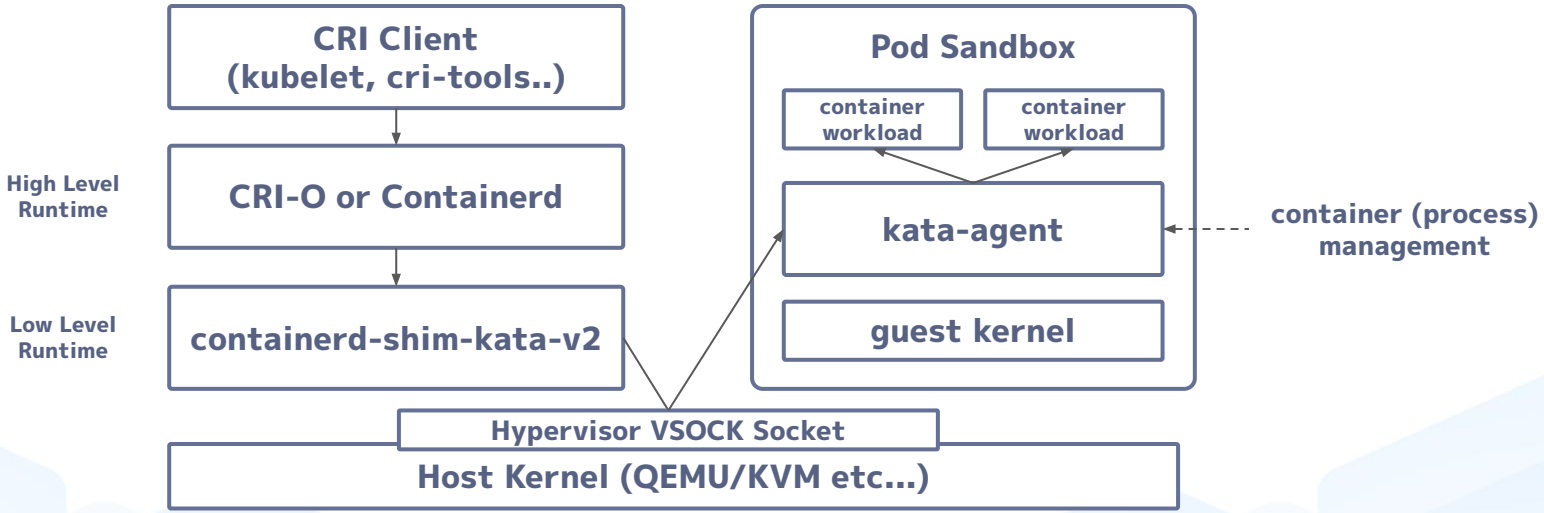
- OpenStack Foundation がオープンソースで開発している
  セキュアなコンテナランタイム (低レベルランタイム) の一つ

- 軽量なVM上を用いてコンテナ化されたワークロードの分離を強化
  - 複数のハイパーバイザに対応
    - QEMU/KVM, Cloud Hypervisor/KVM, Firecracker/KVM など
  - 想定ケースや対応機能に違い有り
    - kata-containers/docs/hypervisors
    - kata-containers/docs/design/virtualization

- プロダクションでの導入実績もある (主に中華系企業)
  - Baidu, Alibaba, Huawei

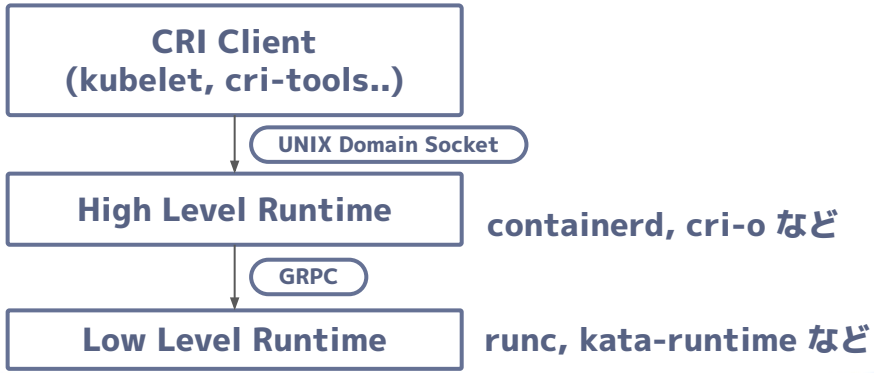
Kata Containers Architecture (v2.x)

- Kata Runtime (containerd-shim-kata-v2) が高レベルランタイムから受け取った命令を vSOCK経由でVM内のkata-agentへ転送し、コンテナの作成などを行う



High Level Runtime, Low Level Runtime

- 高レベルランタイム
  - CRI Client から Pod 作成や削除などの命令を受け取る
  - コンテナイメージの管理やネットワーク管理等を行う
- 低レベルランタイム
  - 高レベルランタイムからの命令でコンテナの起動や停止などの制御を行う
    - 各種ネームスペースの作成、cgroup によるリソース制御, pivot root など...



Ref: コンテナユーザなら誰もが使っているランタイム「runc」を俯瞰する[Container Runtime Meetup #1発表レポート]

実行環境

- **Host**
  - OS: Ubuntu 20.04 LTS
  - Kernel: 5.4.0-65-generic
  - Containerd: v1.5.2
  - Kata Runtime: v2.1.1
  - vSock: 1.0.2.0-k
  - qemu: v5.2.0 (kata-static)
- **Guest**
  - OS: Ubuntu 18.04LTS
  - Kernel: v5.10.25
- **Free5GC**
  - Free5GC nightly
  - gtp5g nightly

High Level Runtime, Low Level Runtime

- **Host**
  - OS: Ubuntu 20.04 LTS
  - Kernel: 5.4.0-65-generic
  - Containerd: v1.5.2
  - Kata Runtime: v2.1.1
  - vSock: 1.0.2.0-k
  - qemu: v5.2.0 (kata-static)
- **Guest**
  - OS: Ubuntu 18.04LTS
  - Kernel: v5.10.25
- **Free5GC (Ubuntu Focal ベースのコンテナイメージを作成)**
  - Free5GC - commit id: b1172c3c4b9d3e1467ec1fbc349194279bc18ada
    - upf - commit id: bdbff7e55851b0079e15befbb72e8a2738311716
    - amf - commit id: 40812d2279530dd967271e8fea2da221b8239aa5
  - gtp5g nightly

環境構築 (containerd / kata-containers)

- Containerd / Default CNI / 起動ファイルの展開

```
$ wget https://github.com/containerd/containerd/releases/download/v1.5.2/cri-containerd-cni-1.5.2-linux-amd64.tar.gz -O - |
tar zxvf -
$ sudo cp -rf etc/* /etc/
$ sudo cp -rf opt/* /opt/
$ sudo cp -rf usr/* /usr/
$ sudo mkdir -p /etc/containerd
$ containerd config default | sudo tee /etc/containerd/config.toml
```

- Kata Containers の展開

```
$ wget https://github.com/kata-containers/kata-containers/releases/download/2.1.1/kata-static-2.1.1-x86_64.tar.xz -O - | tar Jxvf
-
$ sudo cp -rf opt/* /opt/
$ echo “export PATH=\$PATH:/opt/kata/bin” >> ~/.bashrc
$ sudo ln -s /opt/kata/bin/containerd-shim-kata-v2 /usr/local/bin/containerd-shim-kata-v2
$ sudo mkdir -p /etc/kata-containers
# sudo cp /opt/kata/share/defaults/kata-containers/configuration-qemu.toml /etc/kata-containers/configuration.toml
```

設定関連 (/etc/containerd/config.toml)

kata-containers に
関する設定を追加
Default CRI は適宜変更

Free5GCを動作させる上で必要な準備

- デフォルトで用意されている Guest Kernel や Guest Image には
  必要最低限のパッケージやカーネルモジュールしか含まれていない
  - SCTPも無効化されている上、実行中VMでのモジュールの動的ロードは不可

- Free5GCの各NF(主にAMFとUPF)を動作させるためには、以下を行う必要がある
  - AMF用にSCTPを有効化した Guest Kernel の用意
  - UPF用にgtp5g module を組み込んだ Guest Kernel か Guest Image の用意
    - カーネルモジュールを追加する方法など
      - Guest Kernel に built-in で組み込む
      - Guest Image にビルド済み Kernel module を組み込み、
        Kata Containers の設定ファイルに指定した特定のモジュールを
        VM 起動時にロードする

Free5GCを動作させる上で必要な準備

- Guest Kernel や Guest Image の作成にあたって、
  Kata Containers 公式で便利ツールが用意されている
  - kata-containers/tools/os-builder/image/rootfs.sh
  - kata-containers/tools/os-builder/image-builder/image_builder.sh
  - kata-containers/tools/packaging/kernel/build-kernel.sh

Free5GCを動作させる上で必要な準備

- 必要なファイルを作成するまでの流れ
  - Guest Kernel のビルド
    - 上記 Kernel を利用して gtp5g モジュールをビルド
  - Rootfs の作成
    - ビルド済み gtp5g モジュールを Rootfs 内に配置
  - Guest Image の作成
    - Rootfsを元に Guest Image を作成
  - Kata Containers 設定ファイルの変更
    - [hypervisor.qemu] 下の kernel と image の変更
    - [agent.kata] 下の kernel_modules に gtp5g を追加

各種ツールの概要

- kata-containers/tools/os-builder/image/rootfs.sh
  - Guest Image を作成する際の Rootfs を作成する
  - ビルド済みカーネルモジュールはここへ配置する
  - 追加で必要なパッケージなどはこのタイミングでもインストール可能
- kata-containers/tools/os-builder/image-builder/image_builder.sh
  - 上記で作成した Rootfs をベースに Guest Image を作成する
- kata-containers/tools/packaging/kernel/build-kernel.sh
  - Guest Kernel をビルドする
    - kata 2.x でデフォルトのバージョンは5.10.25
  - 自前で有効化したいカーネルオプションなどがある際に利用

Guest Kernelのビルド

- build-kernel.sh を利用する
  - kata-containers のソースとビルドに必要なパッケージを取得

```
$ mkdir ~/workdir && cd ~/workdir
$ wget https://github.com/kata-containers/kata-containers/archive/refs/tags/2.1.1.tar.gz -O - | tar zxvf -
$ sudo apt install -y flex bison libelf-dev libssl-dev
```

  - 下記の設定ファイル (*.conf) を↓のディレクトリへ配置する
    - ~/workdir/kata-containers/tools/packaging/tools/kernel/configs/fragments/x86_64

```
CONFIG_IP_SCTP=y
CONFIG_MODULES=y            (VM起動時にカーネルモジュールを読み込むために必要
CONFIG_MODULE_SIG=y
CONFIG_NET_IP_TUNNEL=y
CONFIG_NET_UDP_TUNNEL=y (GTP5Gの動作に必要 これのみを有効化することは出来ないためGTPを有効化)
CONFIG_GTP=y
```

Guest Kernelのビルド

- build-kernel.sh を利用する
  - Kernel をビルドし、bzImage を vmlinuz としてコピーしておく
    - setup オプションで Linux Kernel の取得と Kernel Config を設定
    - build オプションでカーネルをビルドする

```
$ cd ~/workdir/kata-containers/packaging/kernel/
$ sudo ./build-kernel.sh -a x86_64 -v 5.10.25 setup
$ sudo ./build-kernel.sh -k kata-linux-5.10.25-85 -a x86_64 -v 5.10.25 build
$ sudo cp kata-linux-5.10.25-85/arch/x86/boot/bzImage /opt/kata/share/kata-containers/new-vmlinuz
```

Gtp5gモジュールのビルド

- gtp5g モジュールを Guest Kernel のバージョンでビルドする必要がある
  - Makefile の INCLUDE_DIR を Guest Kernel のルートディレクトリへ変更する

```
< INCLUDE_DIR = /usr/src/linux-headers-$(KVERSION)/
> # INCLUDE_DIR = /usr/src/linux-headers-$(KVERSION)/
> INCLUDE_DIR = /home/user/workdir/kata-containers/tools/packaging/kernel/kata-linux-5.10.25-85
```

  - gtp5g モジュールをビルドする

```
$ cd gtp5g
$ sudo make
```

Rootfsの作成

- Guest Image を作成するためにベースとなる Rootfs を作成する
  - Rootfs を作成する際に、gtp5g.ko を何かしらのディレクトリへ配置し、depmod をコマンドを実行して依存関係に関する情報を更新する
    - depmod 実行時にWARNINGが表示されるが気にしなくても良い
  - Docker 上を用いた Rootfs を作成可能 (USE_DOCKERオプション)

```
$ cd ~/workdir/kata-containers/osbuilder/rootfs-builder
$ sudo OS_NAME=bionic OS_VERSION=18.04 AGENT_BIN=kata-agent USE_DOCKER=yes AGENT_INIT=yes
SECCOMP=no ./rootfs.sh ubuntu
$ sudo mkdir ./rootfs-bionic/lib/modules/5.10.25/kernel/drivers/net
$ sudo cp ~/gtp5g/gtp5g.ko ./rootfs-bionic/lib/modules/5.10.25/kernel/drivers/net
$ depmod -a -b rootfs-bionic 5.10.25
```

Guest Imageの作成

- VM を起動する際に利用する Guest Image を作成する
  - 作成した Rootfs を利用する
  - Docker 上を用いた Rootfs を作成可能 (USE_DOCKERオプション)

```
$ cd ~/workdir/kata-containers/osbuilder/image-builder
$ sudo AGENT_INIT=yes USE_DOCKER=yes ./image_builder.sh ../rootfs-builder/rootfs-bionic
$ sudo cp -rf kata-containers.img /opt/kata/share/kata-containers/new-kata.img
```

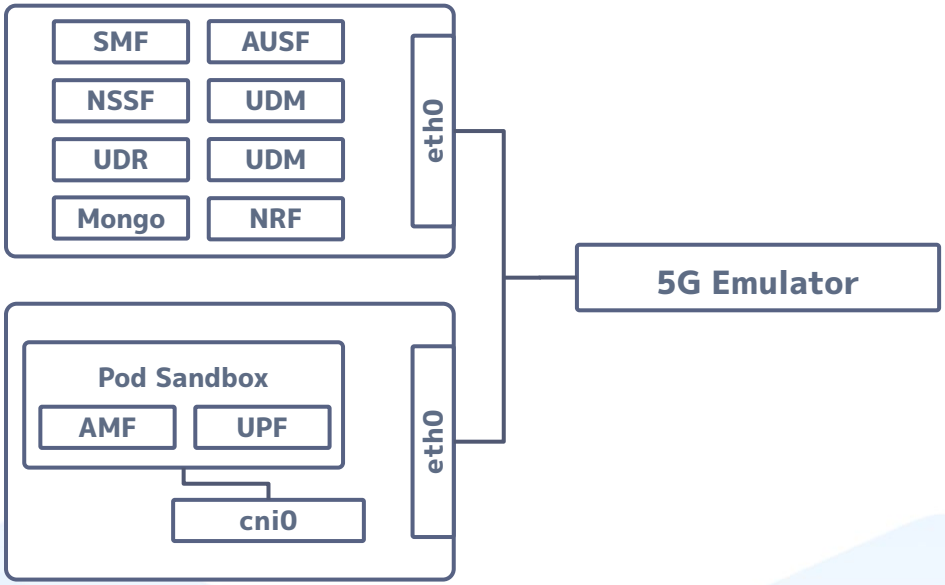
Kata Containers 設定ファイルの修正

- SCTPやgtp5gを適用した Guest Kernel と Guest Image を指定する
  - [agent.kata] 下の kernel_modules に gtp5g を指定することで、VM起動時に動的に gtp5g モジュールをロード出来る
  - 存在しないモジュールを指定している場合はVMの起動に失敗する

```
[hypervisor.qemu]
path = "/opt/kata/bin/qemu-system-x86_64"                    (qemuを使用)
kernel = "/opt/kata/share/kata-containers/new-vmlinuz"       (新たに作成したGuest Kernel を指定)
image = "/opt/kata/share/kata-containers/new-kata.img"       (新たに作成したGuest Image を指定)
machine_type = "pc"

<..snip..>

[agent.kata]
kernel_modules=["gtp5g"]                                     (gtp5gを指定)

<..snip..>
```

動作確認

- AMFとUPFのみ Kata Containers 上で動作させ、接続確認を実施
  - kubernetes などは用いず、cri-tools で手作業で動作確認

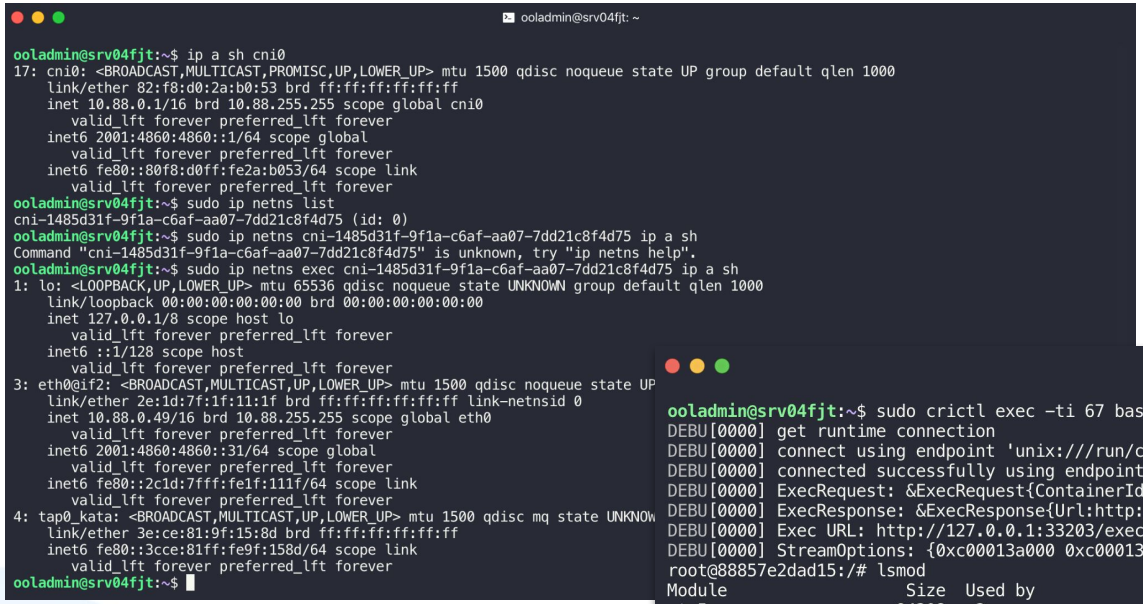


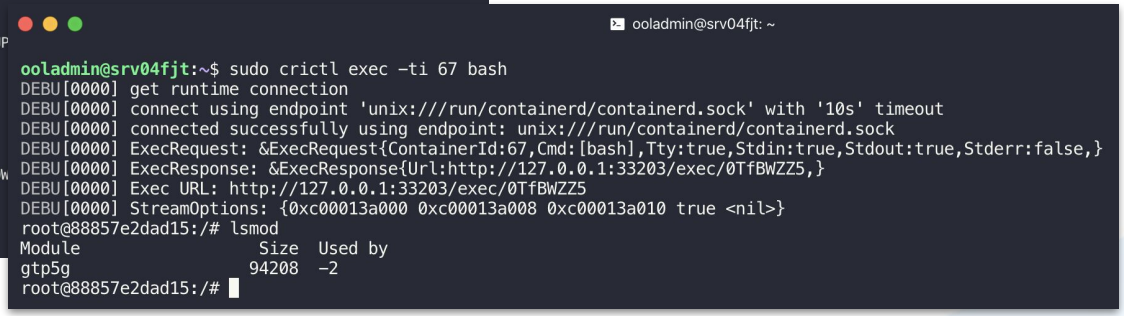
AMF/UPF ノード (起動確認1)

```
ooladmin@srv04fjt: ~

ooladmin@srv04fjt:~$ sudo crictl ps -a
DEBU[0000] get runtime connection
DEBU[0000] connect using endpoint 'unix:///run/containerd/containerd.sock' with '10s' timeout
DEBU[0000] connected successfully using endpoint: unix:///run/containerd/containerd.sock
DEBU[0000] get image connection
DEBU[0000] connect using endpoint 'unix:///run/containerd/containerd.sock' with '10s' timeout
DEBU[0000] connected successfully using endpoint: unix:///run/containerd/containerd.sock
DEBU[0000] ListContainerRequest: &ListContainersRequest{Filter:&ContainerFilter{Id:,State:nil,PodSandboxId:,Label
Selector:map[string]string{},},}
DEBU[0000] ListContainerResponse: &ListContainersResponse{Containers:[]*Container{&Container{Id:67fe8028bf7719363
52b4c11d82e4efc26455c11cda5f4d953c9e82ed3102fd7,PodSandboxId:f2eb50372bb6
b421254c1173,Metadata:&ContainerMetadata{Name:amf,Attempt:0,},Image:&Imag
ly-focal,Annotations:map[string]string{},},ImageRef:sha256:5d0301de4ed794
18d7b74f0a,State:CONTAINER_RUNNING,CreatedAt:1626160474813331635,Labels:m
g]string{},},&Container{Id:58c49f95d83a0455ab95020406e78c655924f2793698c0
2bb6fa60059a6c2ace13bca9d45f7cf9001f2e457e16b421254c1173,Metadata:&Contai
ImageSpec{Image:sumichaaan/free5gc-aio:nightly-focal,Annotations:map[stri
d7948ac70ce5cf6735db633d56d382ba684e5e2ab8e018d7b74f0a,State:CONTAINER_RU
ls:map[string]string{},Annotations:map[string]string{},},},}
CONTAINER           IMAGE                                CREATED
      ATTEMPT             POD ID
58c49f95d83a0       sumichaaan/free5gc-aio:nightly-focal   8 minutes ago
      0                   f2eb50372bb6f
67fe8028bf771       sumichaaan/free5gc-aio:nightly-focal   24 minutes ago
      0                   f2eb50372bb6f
ooladmin@srv04fjt:~$
```

```
ooladmin@srv04fjt: ~

ooladmin@srv04fjt:~$ ps aux | grep qemu
root       17848  0.0  0.0  76340  1752 ?        Sl   07:14   0:00 /opt/kata/libexec/kata-qemu/virtiofsd --fd=3 -o source=/run/kata-co
ntainers/shared/sandboxes/f2eb50372bb6fa60059a6c2ace13bca9d45f7cf9001f2e457e16b421254c1173/shared -o cache=auto --syslog -o no_posix_l
ock -f --thread-pool-size=1
root       17854  0.4  0.6 2500808 214348 ?      Sl   07:14   0:07 /opt/kata/bin/qemu-system-x86_64 -name sandbox-f2eb50372bb6fa60059a
6c2ace13bca9d45f7cf9001f2e457e16b421254c1173 -uuid f60df32f-6f38-403a-8151-1cbed776684e -machine pc,accel=kvm,kernel_irqchip,nvdimm=on
 -cpu host,pmu=off -qmp unix:/run/vc/vm/f2eb50372bb6fa60059a6c2ace13bca9d45f7cf9001f2e457e16b421254c1173/qmp.sock,server=on,wait=off -
m 2048M,slots=10,maxmem=32965M -device pci-bridge,bus=pci.0,id=pci-bridge-0,chassis_nr=1,shpc=on,addr=2 -device virtio-serial-pci,disa
ble-modern=false,id=serial0 -device virtconsole,chardev=charconsole0,id=console0 -chardev socket,id=charconsole0,path=/run/vc/vm/f2eb5
0372bb6fa60059a6c2ace13bca9d45f7cf9001f2e457e16b421254c1173/console.sock,server=on,wait=off -device nvdimm,id=nv0,memdev=mem0 -object
memory-backend-file,id=mem0,mem-path=/opt/kata/share/kata-containers/new-kata.img,size=134217728 -device virtio-scsi-pci,id=scsi0,disa
ble-modern=false -object rng-random,id=rng0,filename=/dev/urandom -device virtio-rng-pci,rng=rng0 -device vhost-vsock-pci,disable-mode
rn=false,vhostfd=3,id=vsock-2816296907,guest-cid=2816296907 -chardev socket,id=char-65146699f87fcb22,path=/run/vc/vm/f2eb50372bb6fa600
59a6c2ace13bca9d45f7cf9001f2e457e16b421254c1173/vhost-fs.sock -device vhost-user-fs-pci,chardev=char-65146699f87fcb22,tag=kataShared -
netdev tap,id=network-0,vhost=on,vhostfds=4,fds=5 -device driver=virtio-net-pci,netdev=network-0,mac=2e:1d:7f:1f:11:1f,disable-modern=
false,mq=on,vectors=4 -rtc base=utc,driftfix=slew,clock=host -global kvm-pit.lost_tick_policy=discard -vga none -no-user-config -nodef
aults -nographic --no-reboot -daemonize -object memory-backend-file,id=dimm1,size=2048M,mem-path=/dev/shm,share=on -numa node,memdev=d
imm1 -kernel /opt/kata/share/kata-containers/new-vmlinuz -append tsc=reliable no_timer_check rcupdate.rcu_expedited=1 i8042.direct=1 i
8042.dumbkbd=1 i8042.nopnp=1 i8042.noaux=1 noreplace-smp reboot=k console=hvc0 console=hvc1 cryptomgr.notests net.ifnames=0 pci=lastbu
s=0 root=/dev/pmem0p1 rootflags=dax,data=ordered,errors=remount-ro ro rootfstype=ext4 quiet systemd.show_status=false panic=1 nr_cpus=
16 systemd.unit=kata-containers.target systemd.mask=systemd-networkd.service systemd.mask=systemd-networkd.socket scsi_mod.scan=none -
pidfile /run/vc/vm/f2eb50372bb6fa60059a6c2ace13bca9d45f7cf9001f2e457e16b421254c1173/pid -smp 1,cores=1,threads=1,sockets=16,maxcpus=16
root       17857  0.0  0.2 2599492 70476 ?       Sl   07:14   0:00 /opt/kata/libexec/kata-qemu/virtiofsd --fd=3 -o source=/run/kata-co
ntainers/shared/sandboxes/f2eb50372bb6fa60059a6c2ace13bca9d45f7cf9001f2e457e16b421254c1173/shared -o cache=auto --syslog -o no_posix_l
ock -f --thread-pool-size=1
ooladmin   18401  0.0  0.0   6432   672 pts/2    R+   07:41   0:00 grep --color=auto qemu
ooladmin@srv04fjt:~$
```

AMF/UPF ノード (起動確認2)





UPFを動作させる上での課題

- PDU Session を開放するシーケンスでコンテナが停止する
  - gtp5g nightly の gtp5g.c の1532〜1550行目
  - static void pdr_context_free(struct rcu_head *head) 関数内の Packet Detection Information 構造体の Service Data Flow の開放処理部分
  - この箇所をコメントアウトすれば開放シーケンスが正常に終了する
- Free5GC公式による推奨環境とはやや異なるため、対応が入るかは分からない
  - Free5GC Recommended Environment
    - https://github.com/free5gc/free5gc/wiki/Environment
  - Gtp5g (gtp5gの推奨環境は、Free5GC公式の推奨環境と少し差異がある)
    - https://github.com/free5gc/gtp5g
      - *5.0.0-23-generic or upper than 5.4 (Ubuntu 20.04)* と記載されている

```
1532        sdf = pdi->sdf;
1533        if (sdf) {
1534            if (sdf->rule) {
1535                if (sdf->rule->sport)
1536                                        kfree(sdf->rule->sport);
1537                if (sdf->rule->dport)
1538                                        kfree(sdf->rule->dport);
1539                if (sdf->rule)
1540                                        kfree(sdf->rule);
1541            }
1542            if (sdf->tos_traffic_class)
1543                                    kfree(sdf->tos_traffic_class);
1544            if (sdf->security_param_idx)
1545                                    kfree(sdf->security_param_idx);
1546            if (sdf->flow_label)
1547                                    kfree(sdf->flow_label);
1548            if (sdf->bi_id)
1549                                    kfree(sdf->bi_id);
1550        }
1551    }
```

まとめと感想

- Kata Containers を使い、Free5GCのNFにおけるAMF及びUPFの動作確認を実施
  - お試し程度に動かしただけなので、時間が合えば PCI Passthrough を使った SRIOV NICなどの物理デバイスのアタッチなどを試してみたい

- コンテナを動作させるVMが挟まるので、VMを動作させる上で様々な考慮が必要
  - 複数ソケットを持つ物理サーバにおいて、VMに割り当てるCPU範囲を決めるなど、NFVにおけるノウハウが必要になる... (どこまでセキュアにしたいかとの兼ね合い)

- UPFをコンテナ化する旨味があるのかは考えていきたい
  - 物理デバイスに密接に紐づく可能性が高い点から見たコンテナとの相性
    - SRIOV, HugePage, CPU Pinning, (k8sだと) Multus など
  - 上記を実現するツールやミドルウェアは多岐に渡るが、長期的に安定運用は出来るのか
    - k8s 自体の更新(1回/6month)、Operatorなどのミドルウェア... 影響範囲は?
  - コンテナ関連の様々なエコシステムに乗り、拡張出来るのは一つのメリットか