

OmniUPF API

概要

OmniUPF API は RESTful 形式で eBPF を介して API を UPF に呼び出すための API です。

API

基本

- **PFCP** を介して UE IP と TEID を
- **PFCP** を介して

機能

- **PDR** (Policy Data Rule) を介して (IPv4/IPv6)
- **FAR** (Forwarding Action Rule) を介して
- **QoS** (Quality of Service) を介して QoS
- **URR** (Usage Reporting Rule) を介して

API

- **FAR** を介して (GET /buffer, GET /buffer/:far_id)
- **FAR** を介して (POST /buffer/:far_id/flush, DELETE /buffer/:far_id, DELETE /buffer)
- **FAR** を介して (POST /buffer/:far_id/notify)
- **DLDR** を介して (GET /buffer/notifications)

機能

- **IP** (Internet Protocol) (GTP, IP, TCP, UDP, ICMP, ARP)
- **XDP** (eXpress Data Path) (eXpress Data Path)
- **N3/N6** (Network Interface) (Network Interface)

- **国際FIB 加盟 (加盟国)**

□□□□□

- **UE** 回線側 UE IP から gNB へ問い合わせ (GET /routes)
- **FRR** 回線側 Free Range Routing へ問い合わせ (POST /routes/sync)
- 回線側セッション管理 (GET /routing/sessions)
- **OSPF** 回線側 OSPF へ問い合わせ (GET /ospf/database/external)

103

- **UPF** `libbpf` (GET /config, POST /config)
- `libbpf` (GET /dataplane_config)
- **XDP** `libbpf` XDP `libbpf` (GET /xdp_capabilities)
- **eBPF** `libbpf` (GET /map_info)

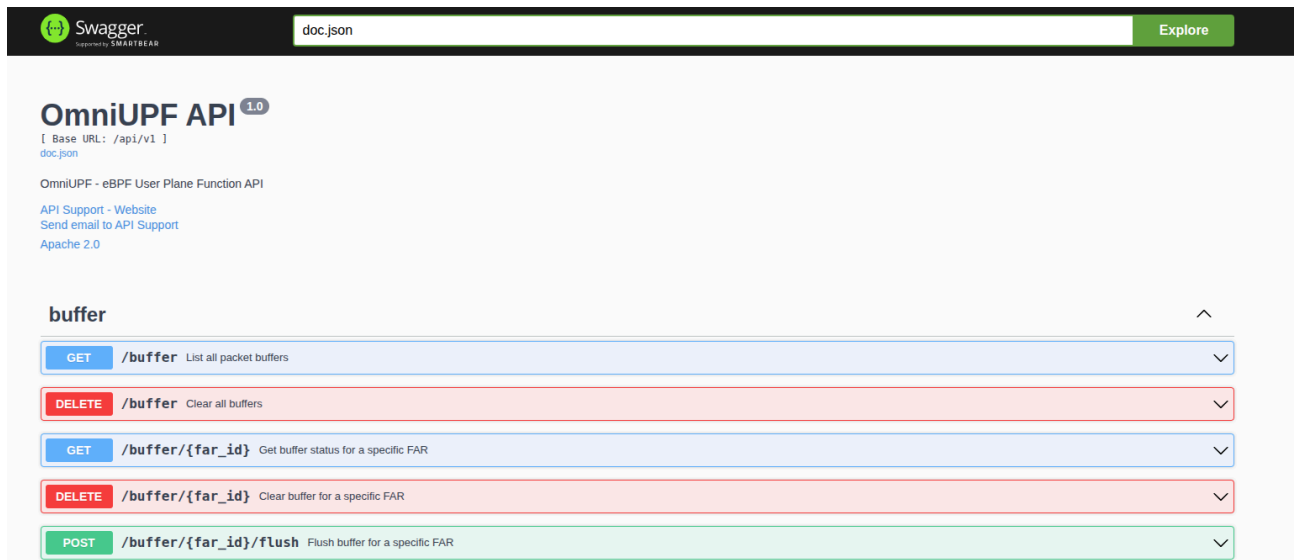
Web UI ☐ ☐

OmniUPF Web UI  API  API  Web UI  

Swagger API ☐

API OpenAPI 3.0 (Swagger) Swagger UI

- 問題/課題
- API 仕様
- 仕様
- HTTP 仕様



Swagger UI for OmniUPF API

Swagger UI

Swagger

```
http://<upf-host>:8080/swagger/index.html
```

```
http://10.98.0.20:8080/swagger/index.html
```

API

API

```
/api/v1
```

API 测试

测试

OmniUPF API 测试脚本用于测试PDR 和 URR 接口

测试脚本

1. **测试PDR接口**
 - `page` 测试 1 次
 - `page\_size` 测试100和1000

2. **测试URR接口**
 - `offset` 测试100
 - `limit` 测试1000

测试脚本

```bash

# 测试PDR接口 50 次

GET /api/v1/pfcp\_sessions?page=2&page\_size=50

# 测试URR接口 100 次 50 次

GET /api/v1/pfcp\_sessions?offset=100&limit=50

# 测试PDR接口 100 次

GET /api/v1/pfcp\_sessions

测试脚本

```

 {
 "data": [
 { /* session object */ },
 { /* session object */ },
 ...
],
 "pagination": {
 "total": 5432,
 "page": 2,
 "page_size": 50,
 "total_pages": 109
 }
}

```

## API Endpoints

- `/api/v1/pfcp_sessions` - PFCP Sessions
- `/api/v1/pfcp_associations` - PFCP Associations
- `/api/v1/routes` - UE IP Routes
- `/api/v1/uplink_pdr_map` - Uplink PDR Mapping
- `/api/v1/uplink_pdr_map/full` - Uplink SDF Mapping PDR
- `/api/v1/downlink_pdr_map` - Downlink PDR IPv4 Mapping
- `/api/v1/downlink_pdr_map/full` - Downlink SDF Mapping PDR IPv4
- `/api/v1/downlink_pdr_map_ip6` - Downlink PDR IPv6 Mapping
- `/api/v1/downlink_pdr_map_ip6/full` - Downlink SDF Mapping PDR IPv6
- `/api/v1/far_map` - FAR Mapping
- `/api/v1/qer_map` - QoS Mapping
- `/api/v1/urr_map` - URR Mapping

## API Operations

- `GET /api/v1/buffer` - Get FAR Buffer
- `GET /api/v1/buffer/:far_id` - Get FAR Buffer by ID
- `GET /api/v1/buffer/notifications` - Get DLDR Notifications
- `DELETE /api/v1/buffer` - Delete FAR Buffer
- `DELETE /api/v1/buffer/:far_id` - Delete FAR Buffer by ID

- `POST /api/v1/buffer/:far_id/flush` - 清除緩衝區
- `POST /api/v1/buffer/:far_id/notify` - 通知 DLDR 完成

查詢

- `GET /api/v1/config` - 查詢 UPF 配置
- `POST /api/v1/config` - 更新 UPF 配置
- `GET /api/v1/dataplane_config` - 查詢數據平面配置

路由管理

- `GET /api/v1/routes` - 查詢 UE 路由
- `POST /api/v1/routes/sync` - 同步 FRR 路由
- `GET /api/v1/routing/sessions` - 查詢路由會話
- `GET /api/v1/ospf/database/external` - 查詢 OSPF 外部 LSA

其他

- 查詢 Web UI 頁面大小 `page_size=100`
- 查詢頁面大小 `page_size=1000`
- 查詢 `pagination.total_pages`
- 查詢 `page_size` API 返回大小

## CORS 配置

CORS (CORS) 配置 API 返回 Web UI 返回 API

## Prometheus 配置

REST API 返回 OmniUPF 返回 `/metrics` 返回 `:9090` Prometheus 返回

配置

- 配置 PFCP 配置
- 配置
- XDP 配置

- **Network**
- **eBPF** **Network**
- **URR** **Network**

**Network** **Network** **Network**

## Network

- **Web UI** **Network** - **Network** API **Network**
- **Network** - Prometheus **Network**
- **PFCP** **Network** - PFCP **Network**
- **Network** - PDR/FAR/QER/URR **Network**
- **Network** - FRR **Network** UE **Network**
- **Network** - **Network**
- **Network** - UPF **Network**
- **Swagger UI** - **Network** API **Network** localhost **Network** UPF **Network**

# OmniUPF 白皮书

## 目录

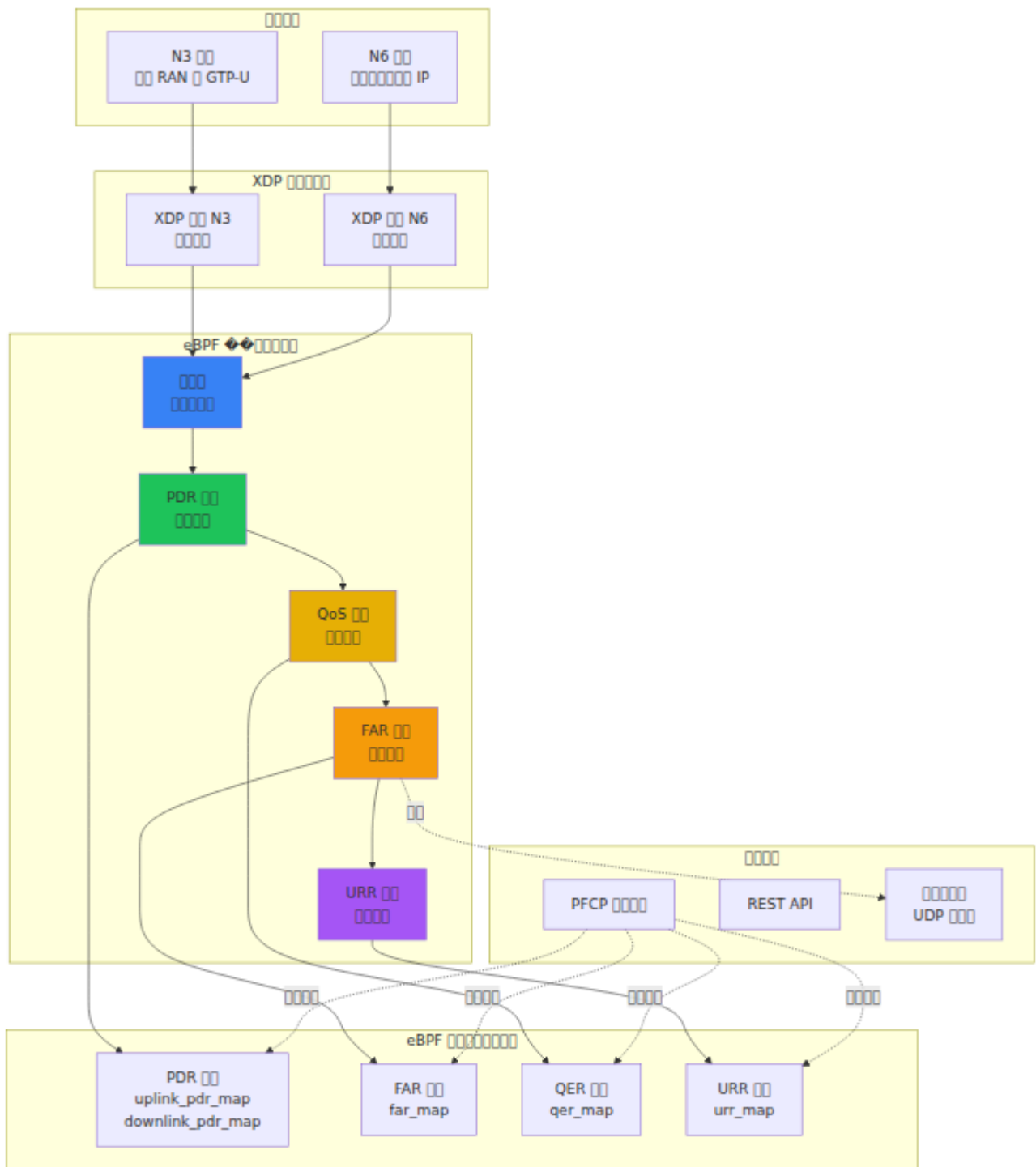
1. 简介
2. eBPF 概述
3. XDP 概述
4. 网络架构
5. eBPF 应用
6. 性能
7. QoS 支持
8. 安全
9. 部署与集成

## 简介

OmniUPF 是一款基于 eBPF 和 XDP 的 5G/LTE 网络功能实现，旨在提供高性能、低延迟的网络服务。它运行在 Linux 内核之上，通过 eBPF 和 XDP 技术实现了对网络流量的精细控制和管理。



111



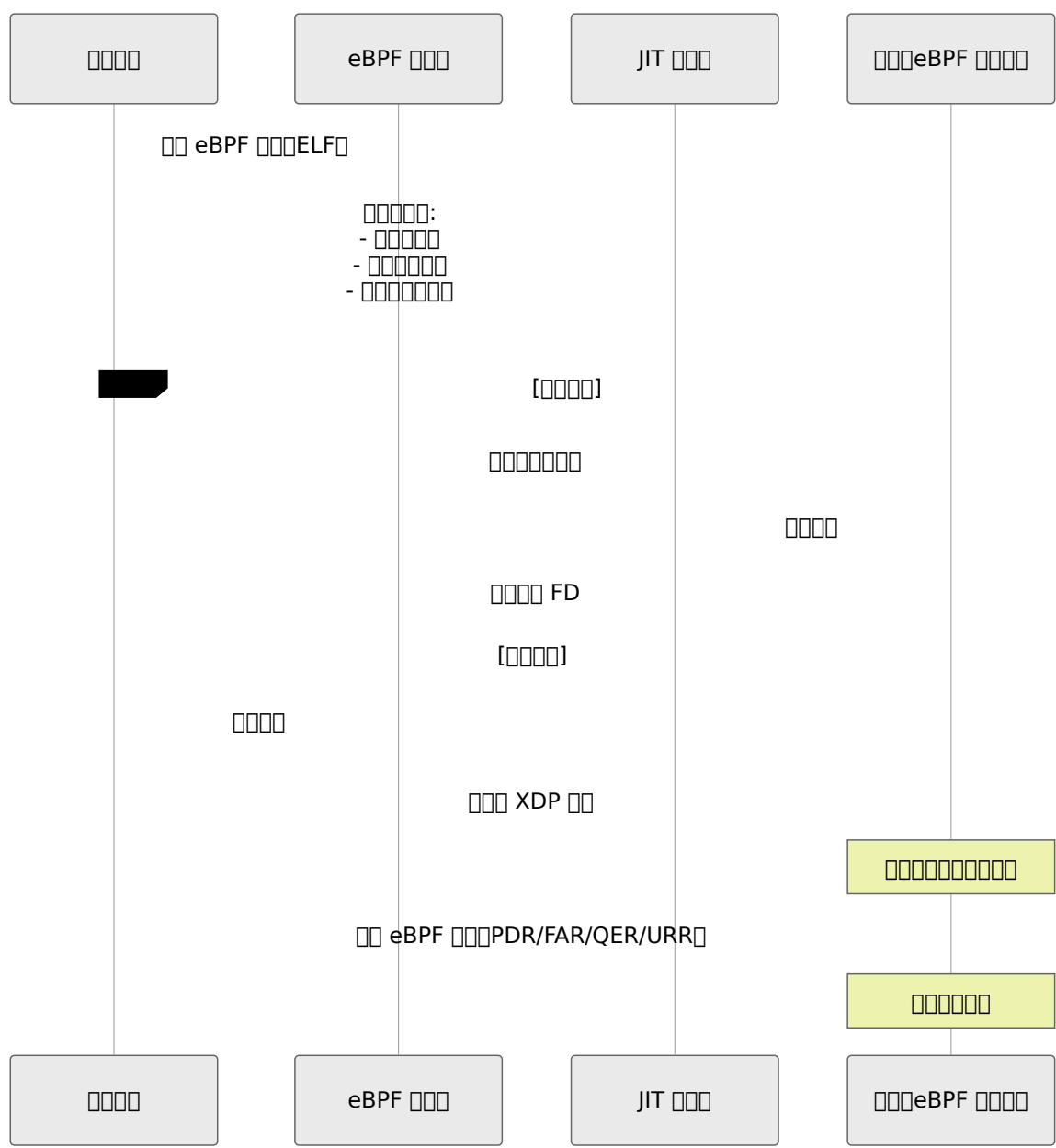
□ □ □ □ □ □

□ □ □ □ □ □

- 
- 



eBPF 架构图



eBPF 应用

eBPF 应用 eBPF 应用

OmniUPF 应用

| 名称                       | 用途      | 备注             |
|--------------------------|---------|----------------|
| BPF_MAP_TYPE_HASH        | 哈希表     | TEID UE IP PDR |
| BPF_MAP_TYPE_ARRAY       | 数组      | ID QER FAR URR |
| BPF_MAP_TYPE_PERCPU_HASH | 每CPU哈希表 | PDR            |
| BPF_MAP_TYPE_LRU_HASH    | LRU哈希表  |                |

特点

- $O(1)$  查找
- 支持多种数据类型
- 支持多种操作
- 支持多种映射类型

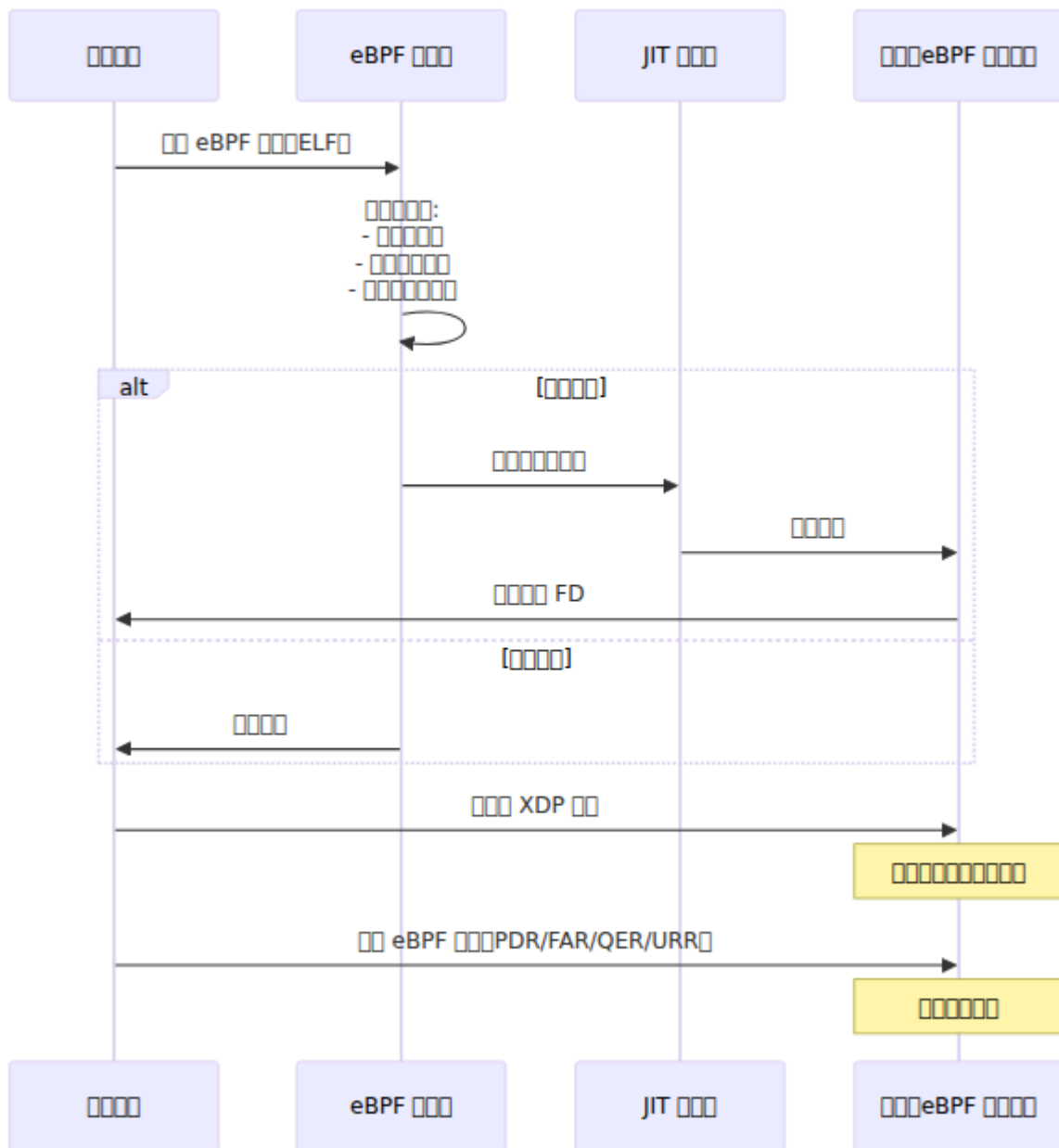
# XDP 简介

## XDP 是什么

XDP 是 Linux 内核中 eBPF 的一种应用，用于在用户态和内核态之间进行数据交换。

## XDP 的应用

OmniUPF 使用 XDP 实现高性能的数据包处理。



## 1. XDP

SmartNIC

- eBPF SmartNIC
- NIC CPU
- 100 Gbps+
- SmartNIC Netronome Mellanox ConnectX-6

SmartNIC

`xdp_attach_mode: offload`

□□□

- □□□□ SmartNIC □□
- □□ eBPF □□□□□
- □□□□ eBPF □□□□□□□□□□

---

## 2. XDP □□□□□□□□□□□□□□□□

□□□□□□□□□□□

- eBPF □□□□□□□□□□□□□□□□
- □□□□ SKB□□□□□□□□□□□□□□□□
- □□□□□□ 10-40 Gbps
- □□□□ XDP □□□□□□□□□□□□□□□□

□□□

`xdp_attach_mode: native`

□□□

- □□□□□□□□□□□□□□□□
- □□□□□□□□□□
- □□□ eBPF □□□□

□□□□□□□□□□

- □□□□i40e□ice□ixgbe□igb
  - Mellanox□mlx4□mlx5
  - □□□bnxt
  - □□□□ena
  - □□□ 10G+ □□□□
-

### 3. XDP

- eBPF SKB
- XDP
- 
- 

`xdp_attach_mode`: generic

- 
- SR-IOV VM
- 
- 

1-5 Gbps/

---

## XDP

eBPF XDP

| 動作           | 処理              | OmniUPF 動作      |
|--------------|-----------------|-----------------|
| XDP_PASS     | パケットを通過させる      | パケットを通過しICMPを返す |
| XDP_DROP     | パケットを丢弃         | パケットを通過しパケットを丢弃 |
| XDP_TX       | パケットを送信         | パケットを送信         |
| XDP_REDIRECT | パケットをリダイレクト     | パケットをN3 ↔ N6    |
| XDP_ABORTED  | パケットを通過しパケットを丢弃 | eBPF 動作         |

パケットを通過させる

パケットを

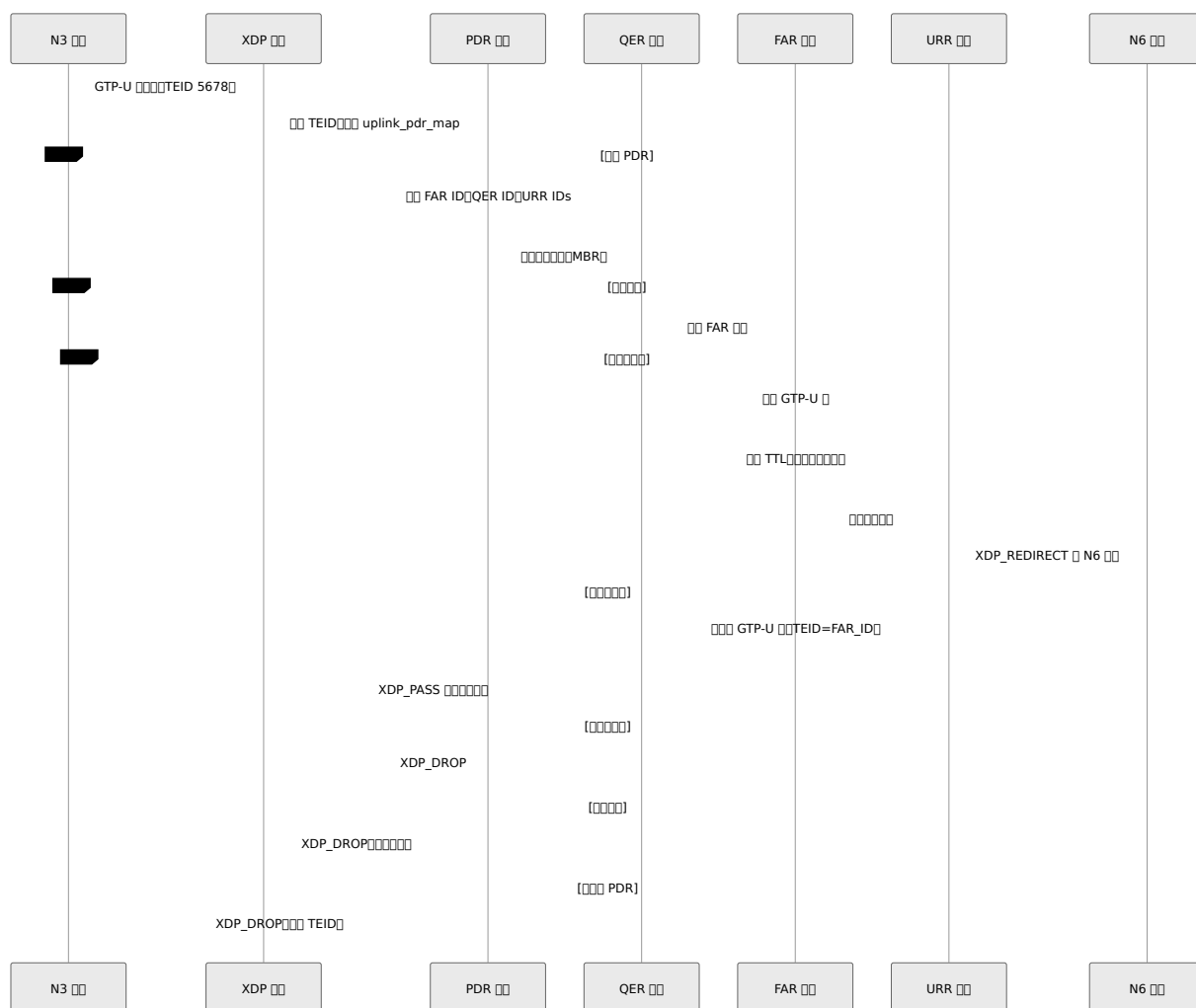
OmniUPF は eBPF を利用してパケットを処理する



パケット

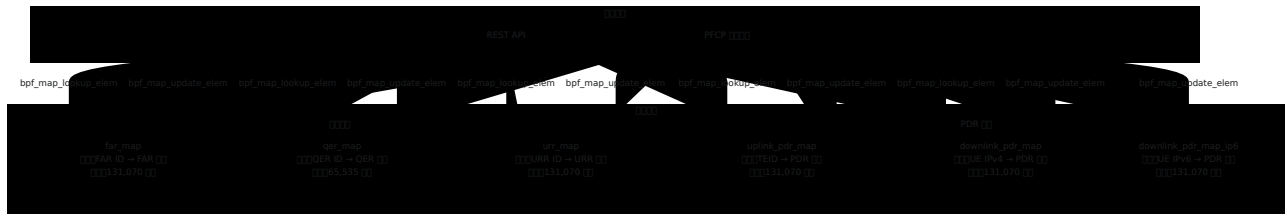
- 1. eBPF を利用して eBPF を
- 2. パケットをパケットを
- 3. パケットをパケット
- 4. 33 パケット





# eBPF いろいろ

## いろいろいろいろ



## いろいろ

OmniUPF いろいろ `max_sessions` いろいろいろいろ

PDR いろいろ =  $2 \times \text{max\_sessions}$  (いろいろ + いろいろ)  
FAR いろいろ =  $2 \times \text{max\_sessions}$  (いろいろ + いろいろ)  
QER いろいろ =  $1 \times \text{max\_sessions}$  (いろいろ)  
URR いろいろ =  $3 \times \text{max\_sessions}$  (いろいろ URR)

いろいろ `max_sessions` = 65,535 いろいろ

- PDR いろいろ 131,070 いろいろ
- FAR いろいろ 131,070 いろいろ
- QER いろいろ 65,535 いろいろ
- URR いろいろ 131,070 いろいろ

## いろいろ

PDR いろいろ  $3 \times 131,070 \times 212 \text{ B} = \sim 83 \text{ MB}$   
FAR いろいろ  $131,070 \times 20 \text{ B} = \sim 2.6 \text{ MB}$   
QER いろいろ  $65,535 \times 36 \text{ B} = \sim 2.3 \text{ MB}$   
URR いろいろ  $131,070 \times 20 \text{ B} = \sim 2.6 \text{ MB}$   
いろいろ  $\sim 91 \text{ MB}$  いろいろ

□□□□

□□□□

OmniUPF □□□□□□□□□□□□□□□□ GTP-U □□□□□□□□□□ UDP □□□□□□□□□□□□

□□□□

Parse error on line 10: ...□□□<br/>□□□FAR\_ID → [□□□□] end -----^  
Expecting 'SQE', 'DOUBLECIRCLEEND', 'PE', '-)', 'STADIUMEND',  
'SUBROUTINEEND', 'PIPE', 'CYLINDEREND', 'DIAMOND\_STOP', 'TAGEND',  
'TRAPEND', 'INVTRAPEND', 'UNICODE\_TEXT', 'TEXT', 'TAGSTART', got 'SQS'

□□

□□□□□□

□□□□□□FAR □□□ 2 □□□□eBPF □□□

1. □□□□□□□□□□

```
orig_packet_len = ntohs(ip->tot_len); // □ IP □□□□
```

2. □□□□□□□□

```
// □□□□ IP + UDP + GTP-U □□□□
gtp_encap_size = sizeof(struct iphdr) + sizeof(struct udphdr) +
sizeof(struct gtpuhdr);
bpf_xdp_adjust_head(ctx, -gtp_encap_size);
```

3. □□□□ IP □□

```
ip->saddr = original_sender_ip; // 源IP地址
ip->daddr = local_upf_ip; // 本地UPF IP
ip->protocol = IPPROTO_UDP;
ip->ttl = 64;
```

#### 4. 填充 UDP 首部

```
udp->source = htons(22152); // BUFFER_UDP_PORT
udp->dest = htons(22152);
udp->len = htons(sizeof(udphdr) + sizeof(gtpuhdr) +
orig_packet_len);
```

#### 5. 填充 GTP-U 首部

```
gtp->version = 1;
gtp->message_type = GTPU_G_PDU;
gtp->teid = htonl(far_id | (direction << 24)); // 填充 FAR ID 和方向
gtp->message_length = htons(orig_packet_len);
```

#### 6. 填充 XDP\_PASS 首部

- 填充 XDP\_PASS 首部
- 填充 XDP\_PASS 首部

填充 XDP\_PASS 首部

填充 XDP\_PASS 首部 SMF 和 FAR 的 IP 地址和端口号



# QoS

~~~~~

OmniUPF ~~~~ QoS

Parse error on line 5: ...= packet\_size × 8 × (NSEC\_PER\_SEC / rate -----  
-----^ Expecting 'SQE', 'DOUBLECIRCLEEND', 'PE', '-)', 'STADIUMEND',  
'SUBROUTINEEND', 'PIPE', 'CYLINDEREND', 'DIAMOND\_STOP', 'TAGEND',  
'TRAPEND', 'INVTRAPEND', 'UNICODE\_TEXT', 'TEXT', 'TAGSTART', got 'PS'

~~

~~~~~

~~~~~ qer.h ~~~

```

static __always_inline enum xdp_action limit_rate_sliding_window(
 const __u64 packet_size,
 volatile __u64 *window_start,
 const __u64 rate)
{
 static const __u64 NSEC_PER_SEC = 1000000000ULL;
 static const __u64 window_size = 5000000ULL; // 5ms

 // rate = 0
 if (rate == 0)
 return XDP_PASS;

 // 计算窗口大小
 __u64 tx_time = packet_size * 8 * (NSEC_PER_SEC / rate);
 __u64 now = bpf_ktime_get_ns();

 // 计算窗口起始位置
 __u64 start = *window_start;
 if (start + tx_time > now)
 return XDP_DROP; // 窗口已满

 // 更新窗口起始位置
 if (start + window_size < now) {
 *window_start = now - window_size + tx_time;
 return XDP_PASS;
 }

 // 更新窗口起始位置
 *window_start = start + tx_time;
 return XDP_PASS;
}

```

总结

- 窗口大小 5ms = 5,000,000 ns
- 窗口起始位置
- 窗口结束位置
- **MBR = 0**

# QoS 計算

100 MBR = 100 Mbps / 1500 bytes = 1500 packets

## 1. 送信時間

$$tx\_time = 1500 \text{ bytes} \times 8 \text{ bits/byte} \times (1,000,000,000 \text{ ns/sec} \div 100,000,000 \text{ bps})$$
$$tx\_time = 1500 \times 8 \times 10 = 120,000 \text{ ns} = 120 \mu\text{s}$$

## 2. 受信時間

- 送信開始時刻  $t=0$  から受信開始時刻  $t=120\mu\text{s}$  まで
- 受信開始時刻  $t=100\mu\text{s}$  から送信開始時刻まで
- 受信開始時刻  $t=150\mu\text{s}$  から送信開始時刻まで

## 3. 最大PPS

$$\text{Max PPS} = (100 \text{ Mbps} \div 8) \div 1500 \text{ bytes} = 8,333 \text{ packets/sec}$$
$$\text{送信時間} = 120 \mu\text{s}$$

# まとめ

## 参考

| 技術                  | 帯域       | PPS        | 遅延                   |
|---------------------|----------|------------|----------------------|
| XDP 搭載 SmartNIC     | 100 Gbps | 148 Mpps   | < 1 $\mu\text{s}$    |
| XDP 搭載 10G NIC      | 10 Gbps  | 8 Mpps     | 2-5 $\mu\text{s}$    |
| XDP 搭載 10G NIC 4ポート | 40 Gbps  | 32 Mpps    | 2-5 $\mu\text{s}$    |
| XDP 無し              | 1-5 Gbps | 0.8-4 Mpps | 50-100 $\mu\text{s}$ |



# 遅延

遅延の構成要素XDP 遅延

| 項目           | 遅延          | 遅延          |
|--------------|-------------|-------------|
| NIC RX       | 0.5 $\mu$ s | 0.5 $\mu$ s |
| XDP 遅延       | 0.1 $\mu$ s | 0.6 $\mu$ s |
| PDR 遅延       | 0.3 $\mu$ s | 0.9 $\mu$ s |
| QER 遅延       | 0.1 $\mu$ s | 1.0 $\mu$ s |
| FAR 遅延       | 0.5 $\mu$ s | 1.5 $\mu$ s |
| URR 遅延       | 0.2 $\mu$ s | 1.7 $\mu$ s |
| GTP-U 遅延     | 0.8 $\mu$ s | 2.5 $\mu$ s |
| XDP_REDIRECT | 0.5 $\mu$ s | 3.0 $\mu$ s |
| NIC TX       | 0.5 $\mu$ s | 3.5 $\mu$ s |

遅延の構成要素 ~3.5  $\mu$ sXDP 遅延10G NIC

## CPU 遅延

遅延の構成要素

- 遅延8-10 MppsXDP 遅延
- 遅延遅延12-15 Mpps
- 遅延遅延遅延遅延 8

遅延遅延 CPU 遅延

$$\text{CPU \%} \approx (\text{bytes} / 10,000,000) \times 100\%$$

2 Mpps ~20%

**eBPF**

- ~100 ns
- ~300 ns
- ~50 ns

$$\text{bytes} = \text{pps} \times (\text{header} + \text{payload} \times 64)$$

$$10 \text{ Mpps} \times (1500 \text{ B} + 3 \times 64 \text{ B}) \approx 160 \text{ Gbps}$$

**UPF**

Setting SMF as parent of SMF would create a cycle

- SMF UPF
- UPF UE
- UPF

## 前提条件

### CPU 前提条件

1. XDP 前提条件 CPU 前提条件
2. RSS 前提条件 RX 前提条件
3. eBPF 前提条件

### NIC 前提条件

1. RX 前提条件
2. NIC RSS 前提条件
3. 前提条件

## 前提条件

```
XDP eBPF 前提条件
ulimit -l unlimited

IRQ 前提条件 XDP 前提条件
systemctl stop irqbalance

CPU 前提条件
cpupower frequency-set -g performance

前提条件
sysctl -w net.core.rmem_max=134217728
sysctl -w net.core.wmem_max=134217728
```

## 前提条件

### 前提条件

```
CPU 前提条件 = (PPS ÷ 10,000,000) × 1.5 (50% 前提条件)
前提条件 = (前提条件 × 212 B × 3) + 100 MB (eBPF 前提条件 + 前提条件)
前提条件 = (前提条件 × 2) + 10 Gbps (前提条件)
```

前提条件100 前提条件20 Gbps 前提条件

- CPU:  $(20 \text{ Gbps} \div 10 \text{ Gbps}) \times 1.5 = 3\text{-}4$
- 内存:  $(1\text{M} \times 212 \text{ B} \times 3) + 100 \text{ MB} \approx 750 \text{ MB}$
- 带宽:  $(20 \text{ Gbps} \times 2) + 10 \text{ Gbps} = 50 \text{ Gbps}$

## 网络架构

- **UPF** 网络 - 核心 UPF 网络
- 网络切片 - PDR, FAR, QER, URR 网络
- 网络切片 - 网络切片
- **Web UI** 网络 - 网络切片
- 网络切片 - 网络切片

# OmniUPF 部署

## 前提条件

- 网络
- 容器
- XDP 支持
- 内核
- 网络
- 网络
- NIC 支持
- 网络
- 网络

## 部署

OmniUPF 部署需要配置 4G (EPC) 和 5G 网络。YAML 文件

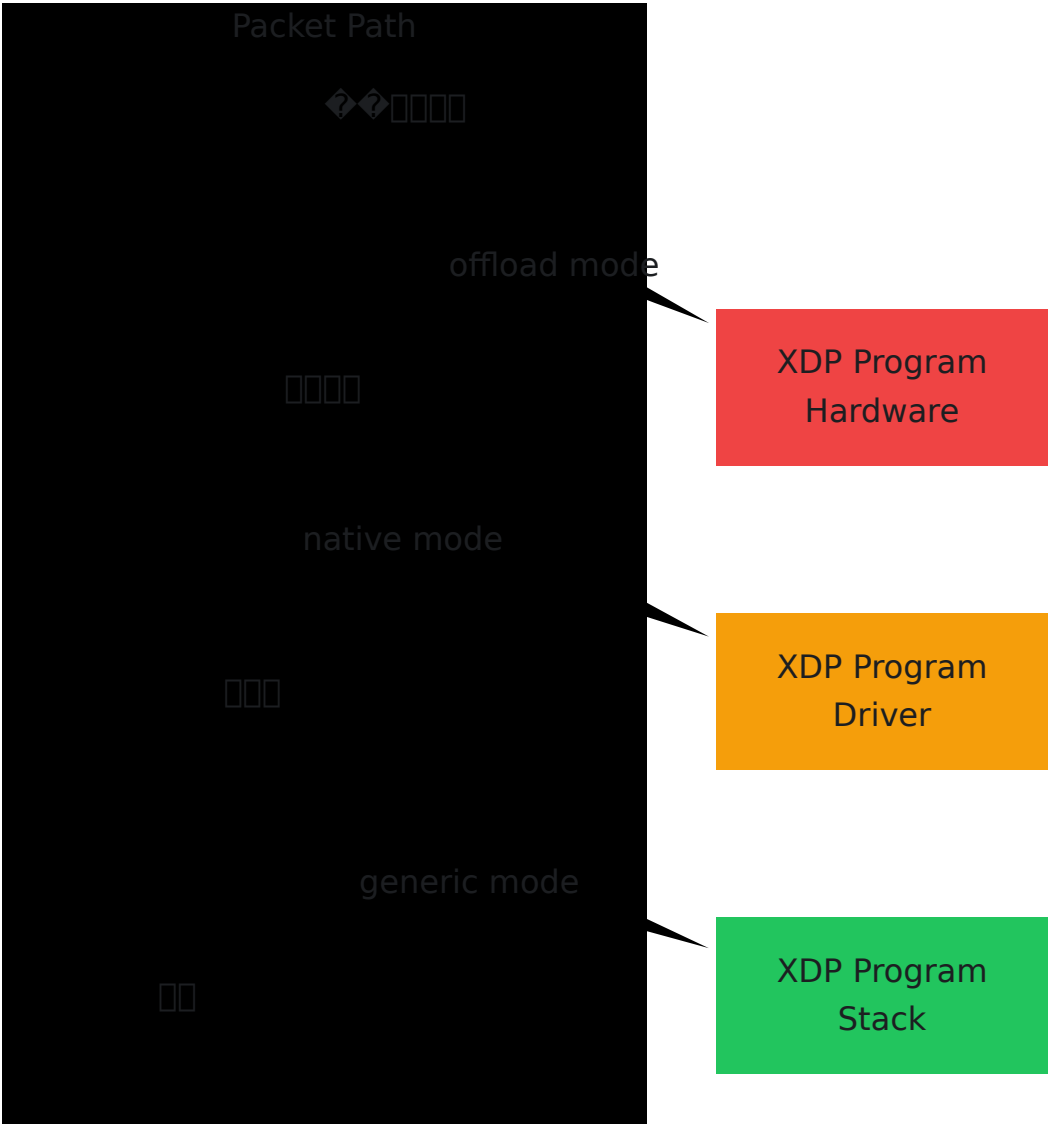
## 部署

OmniUPF 部署需要配置



网络性能

| 网络接口 | 网络类型              | 性能          | 网络接口                | NIC 类型             |
|------|-------------------|-------------|---------------------|--------------------|
| 网卡   | 网卡 (NIC)          | ~1-2 Mpps   | 网卡                  | 网卡 NIC             |
| 网卡   | 网卡 (NIC)          | ~5-10 Mpps  | 网卡 (支持 SR-IOV 的 VM) | 网卡 XDP 网卡          |
| 网卡   | NIC 网卡 (SmartNIC) | ~10-40 Mpps | 网卡                  | 网卡 XDP 网卡 SmartNIC |



## NIC (NIC)

NIC XDP 対応 NIC

NIC

- NIC 対応 NIC
- 対応 NIC
- 対応 NIC
- 対応 NIC

NIC

- NIC (~1-2 Mpps 対応)
- NIC XDP 対応 NIC

NIC

`xdp_attach_mode: generic`

NIC

- NIC SR-IOV 対応
- 対応 NIC
- NIC XDP 対応 NIC
- Proxmox VMware VirtualBox 対応 NIC

## NIC (NIC)

NIC XDP 対応 NIC

NIC

- NIC (~5-10 Mpps 対応)
- 対応 NIC
- 対応 NIC



- 直接 SR-IOV 方式

特徴

- 遅延が低い XDP 方式
- 専用 NIC/ソフトウェア XDP

注意

```
xdp_attach_mode: native
```

前提条件

- 専用 NIC
- SR-IOV 方式
- XDP 対応 NIC (Intel/Mellanox)

要件

- XDP 対応 (NIC 対応)
- Linux 5.15+ 対応 XDP

---

## SmartNIC (NIC)

XDP 対応 SmartNIC 方式

特徴

- 遅延 (~10-40 Mpps)
- 専用 CPU 方式
- 専用 NIC
- CPU 方式

注意

- SmartNIC 方式
- SmartNIC 方式

- 配置項目

項目

xdp\_attach\_mode: offload

項目

- 配置項目
- 配置項目
- CPU 項目

項目

- XDP 項目 SmartNIC (Netronome Agilio CXMellanox BlueField)
- 項目

項目

項目

| 項目             | 項目                                 | 項目 | 項目               |
|----------------|------------------------------------|----|------------------|
| interface_name | N3/N6/N9 項目 (XDP 項目)               | 項目 | [lo]             |
| n3_address     | N3 項目 IPv4 項目 (項目 RAN 項目 GTP-U)    | IP | 127.0.0.1        |
| n9_address     | N9 項目 IPv4 項目 (UPF 項目 UPF 項目 ULCL) | IP | 項目 n3_address 項目 |

項目

```
interface_name: [eth0, eth1]
n3_address: 10.100.50.233
n9_address: 10.100.50.234
```

## PFCP

| Item                      | Description                        | Unit | Value     |
|---------------------------|------------------------------------|------|-----------|
| pfcp_address              | PFCP address (N4/Sxb/Sxc)          | Port | :8805     |
| pfcp_node_id              | PFCP node ID                       | IP   | 127.0.0.1 |
| pfcp_remote_node          | Remote PFCP node (SMF/PGW-C/SGW-C) | Port | []        |
| association_setup_timeout | Association setup timeout (s)      | sec  | 5         |
| heartbeat_retries         | Heartbeat retries                  | sec  | 3         |
| heartbeat_interval        | PFCP heartbeat interval (s)        | sec  | 5         |
| heartbeat_timeout         | PFCP heartbeat timeout (s)         | sec  | 5         |

Example

```
pfcp_address: :8805
pfcp_node_id: 10.100.50.241
pfcp_remote_node:
 - 10.100.50.10 # OmniSMF
 - 10.100.60.20 # OmniPGW-C
heartbeat_interval: 10
heartbeat_retries: 5
```

# API 配置

| 項目              | 説明                                      | 単位     | 値     |
|-----------------|-----------------------------------------|--------|-------|
| api_address     | REST API 接続先                            | IP:ポート | :8080 |
| metrics_address | Prometheus 接続先 (IP:ポート)                 | IP:ポート | :9090 |
| logging_level   | ログレベル (trace, debug, info, warn, error) | レベル    | info  |

例

```
api_address: :8080
metrics_address: :9090
logging_level: debug
```

# GTP 配置

| 項目                | 説明            | 単位 | 値   |
|-------------------|---------------|----|-----|
| gtp_peer          | GTP 接続先       | IP | [ ] |
| gtp_echo_interval | GTP エコー間隔 (秒) | 秒  | 10  |

例

```
gtp_peer:
 - 10.100.50.50:2152 # gNB
 - 10.100.50.60:2152 # UPF N9
gtp_echo_interval: 15
```

# eBPF 設定

| 項目           | 説明              | 単位 | 値     | 計算式              |
|--------------|-----------------|----|-------|------------------|
| max_sessions | セッション数          | 個  | 65535 | セッション数           |
| pdr_map_size | PDR eBPF マップサイズ | 個  | 0     | max_sessions × 2 |
| far_map_size | FAR eBPF マップサイズ | 個  | 0     | max_sessions × 2 |
| qer_map_size | QER eBPF マップサイズ | 個  | 0     | max_sessions     |
| urr_map_size | URR eBPF マップサイズ | 個  | 0     | max_sessions × 2 |

セッション数 0 (個) 単位 max\_sessions セッション数

単位

```
max_sessions: 100000
セッション数0000
PDR: 200,000 個
FAR: 200,000 個
QER: 100,000 個
URR: 200,000 個
```

セッション数

```
max_sessions: 50000
pdr_map_size: 131070 # セッション
far_map_size: 131070
qer_map_size: 65535
urr_map_size: 131070
```

配置

| 項目                      | 説明                  | 単位  | 値      |
|-------------------------|---------------------|-----|--------|
| buffer_port             | eBPF ユーザー空間 UDP ポート | ポート | 22152  |
| buffer_max_packets      | FAR 最大パケット数         | 個   | 10000  |
| buffer_max_total        | 最大総容量 (0=無制限)       | バイト | 100000 |
| buffer_packet_ttl       | パケット TTL (0=無制限)    | 秒   | 30     |
| buffer_cleanup_interval | クリーンアップ間隔 (0=無制限)   | 秒   | 60     |

出力

```
buffer_port: 22152
buffer_max_packets: 20000
buffer_max_total: 200000
buffer_packet_ttl: 60
buffer_cleanup_interval: 30
```

設定

| 項目           | 説明                                   | 単位   | 値            |
|--------------|--------------------------------------|------|--------------|
| feature_ueip | OmniUPF ユーザー IP 機能                   | ブール値 | false        |
| ueip_pool    | UE IP 範囲 IP 範囲 (feature_ueip 有効時)    | CIDR | 10.60.0.0/24 |
| feature_ftup | OmniUPF F-TEID 機能                    | ブール値 | false        |
| teid_pool    | F-TEID 範囲 TEID 範囲 (feature_ftup 有効時) | ポート  | 65535        |

UE IP

```
feature_ueip: true
ueip_pool: 10.45.0.0/16 # UE IP
```

F-TEID

```
feature_ftup: true
teid_pool: 1000000 # 1M TEID
```

FRR (Free Range Routing) UE

| route_manager_enabled    | UE     |    | false          |
|--------------------------|--------|----|----------------|
| route_manager_type       | (frr ) |    | frr            |
| route_manager_vtysh_path | vtysh  |    | /usr/bin/vtysh |
| route_manager_nexthop    | UE IP  | IP | `` ( )         |

```
route_manager_enabled: true
route_manager_type: frr
route_manager_vtysh_path: /usr/bin/vtysh
route_manager_nexthop: 10.0.1.1 # UE
```

- 配置する UPF 等
  - OSPF と BGP の設定
  - FRRouting の設定
- 

設定

**YAML** 形式 (例)

例: `config.yml`



```
配置
interface_name: [eth0]
n3_address: 10.100.50.233
n9_address: 10.100.50.233
xdp_attach_mode: native

PFCP 配置
pfcip_address: :8805
pfcip_node_id: 10.100.50.241
pfcip_remote_node:
 - 10.100.50.10

API 配置
api_address: :8080
metrics_address: :9090
logging_level: info

配置
max_sessions: 100000

GTP 配置
gtp_peer:
 - 10.100.50.50:2152
gtp_echo_interval: 10

配置
feature_ueip: true
ueip_pool: 10.45.0.0/16
feature_ftup: false

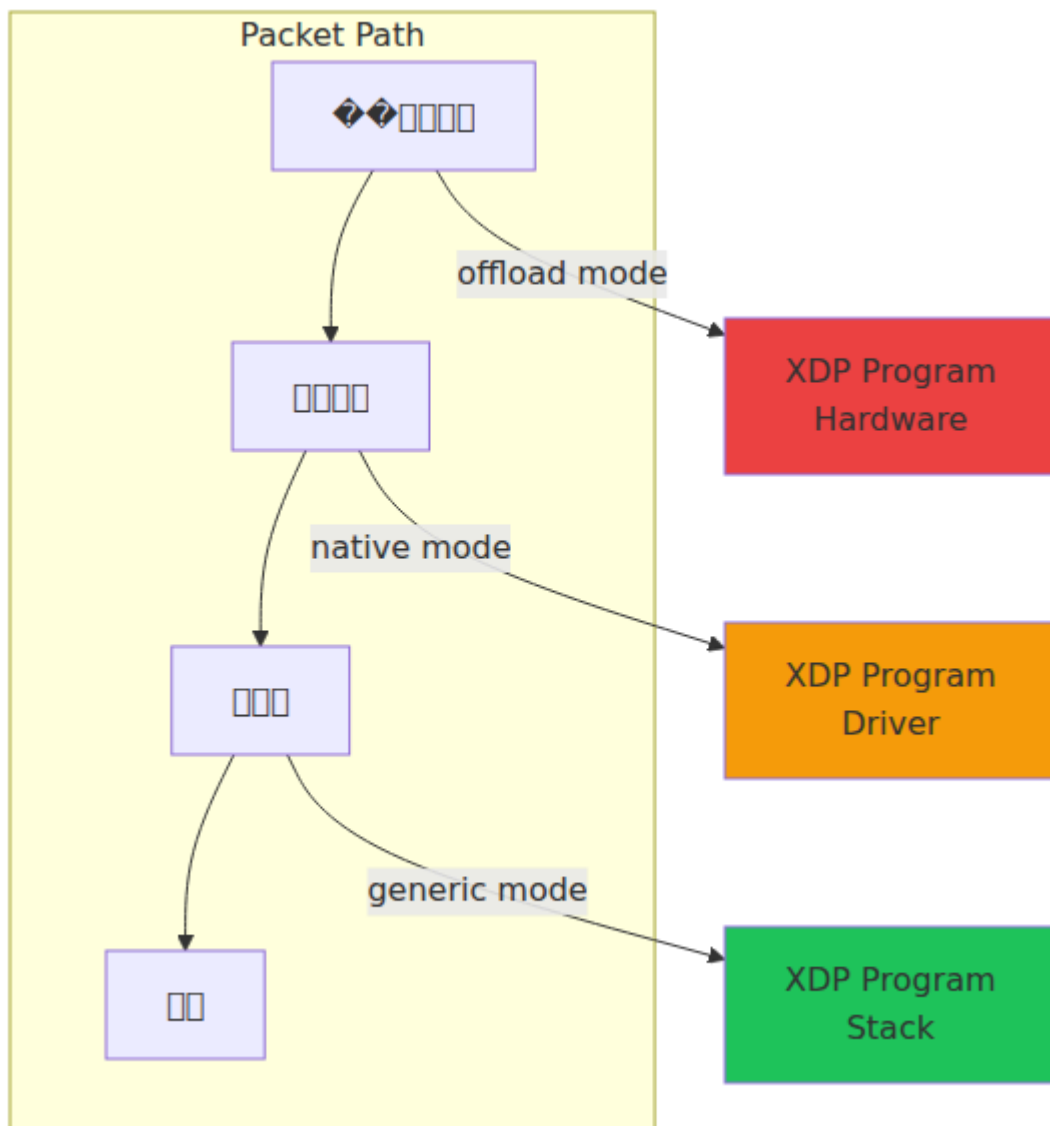
配置
buffer_max_packets: 15000
buffer_packet_ttl: 45
```

配置

配置

OmniUPF 配置 XDP 配置

Proxmox XDP



## Proxmox VE

Proxmox VE

### 1. XDP (XDP)

Proxmox VE VM

Config

- VirtIO E1000
- XDP generic

- 1000~1-2 Mpps

## Proxmox VM 設定

```
net0: net0
ide0: VirtIO (ide0)
ide1: vmbr0
```

## OmniUPF 設定

```
interface_name: [eth0]
xdp_attach_mode: generic
```

---

## 2. SR-IOV 設定 (XDP)

前提条件

確認

- SR-IOV 対応 NIC
- XDP モード `native`
- 1000~5-10 Mpps

設定

- SR-IOV 対応 NIC (例 Intel X710/Mellanox ConnectX-5)
- BIOS で SR-IOV 有効
- IOMMU (`intel_iommu=on` 或 `amd_iommu=on` 及び GRUB 設定)

## Proxmox VM SR-IOV

```
编辑 GRUB 配置
nano /etc/default/grub

设置 GRUB_CMDLINE_LINUX_DEFAULT:
intel_iommu=on iommu=pt

更新 GRUB 配置
update-grub
reboot

为 NIC 创建 VFs (例如 eth0 为 4 个 VFs)
echo 4 > /sys/class/net/eth0/device/sriov_numvfs

设置持久化
echo "echo 4 > /sys/class/net/eth0/device/sriov_numvfs" >>
/etc/rc.local
chmod +x /etc/rc.local
```

## Proxmox VM 配置

```
网络 → 桥 → PCI 桥
桥: SR-IOV 桥
桥: 桥
GPU: 桥
PCI-Express: 桥 (桥)
```

## OmniUPF 配置

```
interface_name: [ens1f0] # SR-IOV VF 桥
xdp_attach_mode: native
```

## 3. PCI 桥 (桥 XDP)

创建 VM 桥 NIC

桥

- 桥 NIC 桥 VM

- XDP `native` `offload` (SmartNIC)
- `~5-40 Mpps` (NIC)

## Proxmox VM

```

NIC → PCI NIC
NIC: NIC (00000000:01:00.0)
NIC:
GPU:
PCI-Express:

```

## OmniUPF

```

interface_name: [ens1f0]
xdp_attach_mode: native # 'offload' SmartNIC

```

# KVM/QEMU

## OmniUPF

```

virt-install \
 --name omniupf \
 --network bridge=br0,model=virtio \
 --disk path=/var/lib/libvirt/images/omniupf.qcow2 \
 ...

```

## SR-IOV

```

<interface type='hostdev' managed='yes'>
 <source>
 <address type='pci' domain='0x0000' bus='0x01' slot='0x10'
function='0x1' />
 </source>
</interface>

```

# VMware ESXi

## vSwitch (XDP)

- VMXNET3
- XDP generic

## SR-IOV (XDP)

- ESXi SR-IOV
  - SR-IOV VM
  - XDP native
- 

# Microsoft Hyper-V

## (XDP)

- 
- XDP generic

## SR-IOV (XDP)

- Hyper-V SR-IOV
  - SR-IOV
  - XDP native
- 

# VirtualBox

## NAT/ (XDP)

- VirtIO-Net Intel PRO/1000
  - XDP generic
  - VirtualBox SR-IOV
-

# NIC 計算

## 100 Mpps 計算

100pps (Mpps) 100pps (Gbps) 計算 - 100pps (Mpps) 100pps (Gbps) 計算 - 100pps (Mpps) 100pps (Gbps) 計算  
100 VoIP 100pps (Mpps) 100pps (Gbps) 計算

100pps (Mpps) 100pps (Gbps) 計算

100pps (Mpps) 100pps (Gbps) 計算 N3 100pps (Mpps) 100pps (Gbps) 計算 N6 100pps (Mpps) 100pps (Gbps) 計算 IP 100pps

### GTP-U 100pps (N3 100pps)

- 100 IPv4 100pps 100pps
- 100 UDP 100pps 100pps
- 100 GTP-U 100pps 100pps
- 100 GTP-U 100pps 36 100pps

### 100 GTP-U 100pps (N3) 100pps

- 100 IP 100pps 20 100pps (IPv4)
- 100 UDP 100pps 8 100pps
- 100pps 1 100pps
- 100pps 29 100pps
- 100 GTP-U 100pps 36 100pps
- 100pps 65 100pps

### 100 1 Mpps 100pps (N3) 100pps GTP-U 100pps

$$65 \text{ 100pps} \times 1,000,000 \text{ pps} \times 8 \text{ 100pps} = 520 \text{ Mbps}$$

### 100 GTP-U 100pps (N3 1500 MTU) 100pps

- 100 IP MTU 1500 100pps (100pps IP 100pps)
- 100 GTP-U 100pps 36 100pps
- 100pps 1536 100pps

1 Mpps の GTP-U

$$1536 \text{ 字节} \times 1,000,000 \text{ pps} \times 8 \text{ 位/字节} = 12,288 \text{ Mbps} \approx 12.3 \text{ Gbps}$$

IP (N6)

N6 (IPv6) の IP の GTP-U

N6

- IP 20 字节
- UDP 8 字节
- 1 字节
- 29 字节

1 Mpps の N6

$$29 \text{ 字节} \times 1,000,000 \text{ pps} \times 8 \text{ 位/字节} = 232 \text{ Mbps}$$

N6 (1500 MTU)

- IP MTU 1500 字节
- 1500 字节

1 Mpps の N6

$$1500 \text{ 字节} \times 1,000,000 \text{ pps} \times 8 \text{ 位/字节} = 12,000 \text{ Mbps} = 12 \text{ Gbps}$$

Intel X710 NIC (N3 10 Mpps)



| 項目              | 標準値           | GTP-U 値       | 10 Mpps 値             | 備考                |
|-----------------|---------------|---------------|-----------------------|-------------------|
| VoIP 値<br>(N3)  | 65-150 値      | 101-186 値     | <b>0.8-1.5 Gbps</b>   | AMR-WB 値<br>G.711 |
| データ 値<br>(N3)   | 400-600 値     | 436-636 値     | <b>3.5-5.1 Gbps</b>   | HTTP/HTTPS 値      |
| データ<br>(N3)     | <b>1200 値</b> | <b>1236 値</b> | <b>9.9 Gbps</b>       | 値 <b>2024</b> 値   |
| データ (N3)        | 1400-1450 値   | 1436-1486 値   | <b>11.5-11.9 Gbps</b> | HD/4K 値           |
| データ MTU<br>(N3) | 1500 値        | 1536 値        | <b>12.3 Gbps</b>      | 値 TCP 値           |

■ N6 値 (値 IP 値 GTP-U)■

| 項目      | 標準値           | 10 Mpps 値             | 備考              |
|---------|---------------|-----------------------|-----------------|
| VoIP 値  | 65-150 値      | <b>0.5-1.2 Gbps</b>   | 値 RTP 値         |
| データ 値   | 400-600 値     | <b>3.2-4.8 Gbps</b>   | HTTP 値          |
| データ 値   | <b>1200 値</b> | <b>9.6 Gbps</b>       | 値 <b>2024</b> 値 |
| データ 値   | 1400-1450 値   | <b>11.2-11.6 Gbps</b> | 値               |
| データ MTU | 1500 値        | <b>12.0 Gbps</b>      | 値               |

■ 10 Mpps 値 (1200 値) 値 ~10 Gbps 値 N3 値 N6 値

値

値 GTP-U 値 (36 値) 値

## 帯域幅 (帯域幅)

- **VoIP (AMR-WB 帯域幅)** 65-80 kHz → GTP-U: 101-116 kHz
- 帯域幅 50-200 kHz → GTP-U: 86-236 kHz
- 帯域幅 (**HTTP/3**) 400-800 kHz → GTP-U: 436-836 kHz
- 帯域幅 1200-1450 kHz → GTP-U: 1236-1486 kHz
- 帯域幅 1500 kHz → GTP-U: 1536 kHz

## GTP-U 帯域幅

- 帯域幅 (< 200 kHz) ~**35-70%** Mpps 帯域幅
- 帯域幅 (200-800 kHz) ~**5-20%** Mpps - 帯域幅
- 帯域幅 (> 1200 kHz) ~**3%** Mpps - 帯域幅

## 帯域幅

### 帯域幅 **10 Mpps** NIC N3 帯域幅

- **VoIP** 帯域幅 (100 帯域幅) ~1.0 Gbps (GTP-U 帯域幅)
- 帯域幅 (1200 帯域幅) ~9.9 Gbps
- 帯域幅 (1400 帯域幅) ~11.5 Gbps
- 帯域幅 (1500 帯域幅) ~12.3 Gbps

### N6 帯域幅 (GTP-U 帯域幅)

- 帯域幅 (1200 帯域幅) ~9.6 Gbps 10 Mpps
- 帯域幅 (1500 帯域幅) ~12.0 Gbps 10 Mpps

## ?? UPF 帯域幅

- 帯域幅 (VoIP 帯域幅) Mpps 帯域幅 - 帯域幅 10 Mpps 1-2 Gbps
- 帯域幅 (1200 帯域幅) 帯域幅 10 Mpps 帯域幅 9-10 Gbps
- 帯域幅 (帯域幅) 帯域幅 10 Mpps 帯域幅 10-12 Gbps
- 帯域幅 **N3** N6 - N3 GTP-U 帯域幅 N6 帯域幅

## 帯域幅

### 1200 帯域幅 (帯域幅)



| 网卡                         | 驱动   | XDP 支持 | 模式       | 性能       |
|----------------------------|------|--------|----------|----------|
| <b>Mellanox ConnectX-5</b> | mlx5 | 是      | 内核       | ~12 Mpps |
| <b>Mellanox ConnectX-6</b> | mlx5 | 是      | 内核       | ~20 Mpps |
| <b>Mellanox BlueField</b>  | mlx5 | 是      | 内核 + 用户态 | ~40 Mpps |
| <b>Mellanox ConnectX-4</b> | mlx4 | 否      | 内核       | ~2 Mpps  |

### Broadcom NICs

| 网卡                       | 驱动      | XDP 支持 | 模式  | 性能      |
|--------------------------|---------|--------|-----|---------|
| <b>Broadcom BCM57xxx</b> | bnxt_en | 是      | 内核  | ~8 Mpps |
| Broadcom NetXtreme II    | bnx2x   | 否      | 用户态 | ~1 Mpps |

### 其他

| 网卡                         | 驱动         | XDP 支持 | 模式 | 性能       |
|----------------------------|------------|--------|----|----------|
| <b>Netronome Agilio CX</b> | nfp        | 是      | 内核 | ~30 Mpps |
| <b>Amazon ENA</b>          | ena        | 是      | 内核 | ~5 Mpps  |
| <b>Solarflare SFC9xxx</b>  | sfc        | 是      | 内核 | ~8 Mpps  |
| VirtIO                     | virtio_net | 否      | 内核 | ~2 Mpps  |

## 如何 NIC XDP 如何

### 如何如何如何 XDP 如何

```
如何 NIC 如何
ethtool -i eth0 | grep driver

如何如何 XDP 如何
modinfo <driver_name> | grep -i xdp

如何 Intel i40e
modinfo i40e | grep -i xdp
```

### 如何 XDP 如何如何

```
如何 XDP 如何如何
ip link show eth0 | grep -i xdp

如何 (XDP 如何):
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 xdp qdisc mq
```

---

## 如何如何 NIC

如何 1200 如何如何如何 (如何如何) 如何

| 項目                  | NIC 種類                               | ポート数   | Mpps (10G)    | 速度 (N3)       | 備考        |
|---------------------|--------------------------------------|--------|---------------|---------------|-----------|
| 標準/基本               | 標準 NIC (VirtIO, E1000)               | 1<br>1 | 1-2<br>Mpps   | 1-2<br>Gbps   | 標準<br>PoC |
| 標準                  | Intel X710, Mellanox CX-5            | 1<br>1 | 5-10<br>Mpps  | 5-10<br>Gbps  | 標準<br>10G |
| 標準/標準               | Intel E810, Mellanox CX-6            | 1<br>1 | 10-20<br>Mpps | 10-20<br>Gbps | 標準<br>10G |
| 標準                  | Mellanox CX-6, Intel E810 (標準)       | 1<br>1 | 20-40<br>Mpps | 20-40<br>Gbps | 標準<br>10G |
| 標準                  | Mellanox BlueField, Netronome Agilio | 1<br>1 | 40+<br>Mpps   | 40+<br>Gbps   | 標準        |
| Proxmox VM (標準)     | VirtIO                               | 1<br>1 | 1-2<br>Mpps   | 1-2<br>Gbps   | 標準        |
| Proxmox VM (SR-IOV) | Intel X710/E810 VF, Mellanox CX-5 VF | 1<br>1 | 5-10<br>Mpps  | 5-10<br>Gbps  | 標準 VM     |

標準

- 標準 1200 標準 GTP-U 標準 (N3 標準 1236 標準)
- N6 標準 (~9.6 Gbps 標準 10 Mpps) 標準 GTP-U 標準
- 標準 - VoIP 標準

標準

標準 XDP 標準

- XDP 標準
- 標準 XDP 標準

NIC 

- Cilium XDP 〇〇〇〇
- IO Visor XDP 〇〇

|  |  |  |  |
|--|--|--|--|
|  |  |  |  |
|--|--|--|--|

□□ **1**□□□□□ (□□□□)

SR-IOV OmniUPF

```
interface_name: [eth0]
xdp_attach_mode: generic
api_address: :8080
pfc_p_address: :8805
pfc_node_id: 127.0.0.1
n3_address: 127.0.0.1
metrics_address: :9090
logging_level: debug
max_sessions: 1000
```

□□ **2□□□□□□ (□□□□)**

□□□□ Intel X710 NIC □□□□□□□□□□ UPF

```
配置参数
interface_name: [ens1f0, ens1f1] # N3 为 ens1f0 N6 为 ens1f1
xdp_attach_mode: native
api_address: :8080
pfcip_address: 10.100.50.241:8805
pfcip_node_id: 10.100.50.241
n3_address: 10.100.50.233
n9_address: 10.100.50.234
metrics_address: :9090
logging_level: info
max_sessions: 500000
gtp_peer:
 - 10.100.50.10:2152 # gNB 1
 - 10.100.50.11:2152 # gNB 2
gtp_echo_interval: 30
pfcip_remote_node:
 - 10.100.50.50 # OmniSMF
heartbeat_interval: 10
feature_ueip: true
ueip_pool: 10.45.0.0/16
buffer_max_packets: 50000
buffer_packet_ttl: 60
```

---

## 3 Proxmox VM 中 SR-IOV (配置)

配置 Proxmox VM 中 SR-IOV 配置 UPF



```
Proxmox SR-IOV []
interface_name: [ens1f0] # SR-IOV VF
xdp_attach_mode: native
api_address: :8080
pfcf_address: 192.168.100.10:8805
pfcf_node_id: 192.168.100.10
n3_address: 192.168.100.10
metrics_address: :9090
logging_level: info
max_sessions: 100000
gtp_peer:
 - 192.168.100.50:2152
gtp_echo_interval: 15
pfcf_remote_node:
 - 192.168.100.20 # SMF
```

---

## [] 4PGW-U [] (4G EPC)

[][] OmniUPF [] 4G EPC [] PGW-U

```
PGW-U []
interface_name: [eth0]
xdp_attach_mode: native
api_address: :8080
pfcf_address: 10.200.1.10:8805
pfcf_node_id: 10.200.1.10
n3_address: 10.200.1.10 # S5/S8 [] (GTP-U)
metrics_address: :9090
logging_level: info
max_sessions: 200000
gtp_peer:
 - 10.200.1.50:2152 # SGW-U
gtp_echo_interval: 20
pfcf_remote_node:
 - 10.200.2.10 # OmniPGW-C (Sxb [])
heartbeat_interval: 5
```

---

## 第5章 (UPF + PGW-U)

OmniUPF 5G 4G

```
#
interface_name: [eth0, eth1]
xdp_attach_mode: native
api_address: :8080
pfcf_address: :8805
pfcf_node_id: 10.50.1.100
n3_address: 10.50.1.100
n9_address: 10.50.1.101
metrics_address: :9090
logging_level: info
max_sessions: 300000
gtp_peer:
 - 10.50.2.10:2152 # 5G gNB
 - 10.50.2.20:2152 # 4G eNodeB (SGW-U)
gtp_echo_interval: 15
pfcf_remote_node:
 - 10.50.3.10 # OmniSMF (5G)
 - 10.50.3.20 # OmniPGW-C (4G)
heartbeat_interval: 10
feature_ueip: true
ueip_pool: 10.60.0.0/16
```

## 第6章 SmartNIC

Netronome Agilio CX SmartNIC

```

SmartNIC
interface_name: [enpls0np0] # SmartNIC
xdp_attach_mode: offload
api_address: :8080
pfc_p_address: 10.10.1.50:8805
pfc_p_node_id: 10.10.1.50
n3_address: 10.10.1.50
metrics_address: :9090
logging_level: warn #
max_sessions: 1000000
pdr_map_size: 2000000
far_map_size: 2000000
qer_map_size: 1000000
gtp_peer:
 - 10.10.2.10:2152
 - 10.10.2.20:2152
 - 10.10.2.30:2152
gtp_echo_interval: 30
pfc_p_remote_node:
 - 10.10.3.10
heartbeat_interval: 15
buffer_max_packets: 100000
buffer_max_total: 1000000

```

max\_sessions OmniUPF

```

max_sessions: 100000
#
PDR: 200,000 (2 × max_sessions)
FAR: 200,000 (2 × max_sessions)
QER: 100,000 (1 × max_sessions)
URR: 200,000 (2 × max_sessions)

```

セッション~91 MB 約 100K 程度

---

セッション数

セッション数制限

```
max_sessions: 100000
pdr_map_size: 300000 # セッション PDR
far_map_size: 200000
qer_map_size: 150000 # セッション QER
urr_map_size: 200000
```

---

セッション

セッション数

```
Max Sessions = min(
 pdr_map_size / 2,
 far_map_size / 2,
 qer_map_size
)
```

セッション

- PDR 約200,000
- FAR 約200,000
- QER 約100,000

Max Sessions = min(100,000, 100,000, 100,000) = **100,000**

---

セッション

セッション数

- PDR:  $2 \times 212 \text{ B} = 424 \text{ B}$
- FAR:  $2 \times 20 \text{ B} = 40 \text{ B}$
- QER:  $1 \times 36 \text{ B} = 36 \text{ B}$
- URR:  $2 \times 20 \text{ B} = 40 \text{ B}$
- **合計**:  $\sim 540 \text{ B}$  **必要**

**合計 100K** **メモリ**  $\sim 52 \text{ MB}$  **必要**

**メモリ** **制限**  $2 \times$  **必要**

```
メモリ制限
ulimit -l

メモリ制限 (eBPF 用)
ulimit -l unlimited
```

## 機能

- **機能** - eBPF/XDP **機能**
- **機能** - PDR/FAR/QER/URR **機能**
- **機能** - **機能**
- **機能** - **機能** Prometheus **機能**
- **Web UI** **機能** - **機能**
- **機能** - UPF **機能**

# 📄📄📄📄

📄📄📄📄 OmniUPF 📄 /metrics 📄📄📄📄 Prometheus 📄📄

## 📄📄📄📄

1. **PFCP** 📄📄📄 - 📄📄📄📄📄📄📄📄📄📄📄📄📄📄
2. **XDP** 📄📄📄 - 📄📄📄📄📄📄📄📄📄📄📄📄📄📄
3. 📄📄📄📄 - 📄📄📄📄📄📄📄📄📄📄
4. **PFCP** 📄📄📄📄📄 - 📄📄📄📄📄📄📄📄📄
5. **URR** 📄📄 - 📄 PFCP 📄📄📄📄📄📄📄📄📄
6. 📄📄📄📄📄 - 📄📄📄📄📄📄📄📄📄📄
7. 📄📄📄📄📄📄📄📄 - PFCP 📄📄📄📄📄📄 FAR 📄📄📄
8. **eBPF** 📄📄📄📄 - eBPF 📄📄📄📄📄📄

## 📄📄📄📄

### **PFCP** 📄📄📄📄

📄📄📄 UPF 📄📄📄📄📄📄📄 PFCP 📄📄📄📄📄📄

| 이벤트                  | 필드                                           | 내용                         | 설명 |
|----------------------|----------------------------------------------|----------------------------|----|
| upf_pfcpx_rx         | message_name,<br>peer_address                | PFCP 수신                    |    |
| upf_pfcpx_tx         | message_name,<br>peer_address                | PFCP 전송                    |    |
| upf_pfcpx_rx_errors  | message_name,<br>cause_code,<br>peer_address | PFCP 수신 오류                 |    |
| upf_pfcpx_rx_latency | message_type,<br>peer_address                | PFCP 수신 지연 (p50, p90, p99) |    |

이벤트 이름은 PFCP 관련 이벤트를 나타냅니다.

XDP 이벤트

XDP 이벤트는 XDP 관련 이벤트를 나타냅니다.

| 이벤트              | 필드           | 내용           |
|------------------|--------------|--------------|
| upf_xdp_aborted  | peer_address | XDP_ABORTED  |
| upf_xdp_drop     | peer_address | XDP_DROP     |
| upf_xdp_pass     | peer_address | XDP_PASS     |
| upf_xdp_tx       | peer_address | XDP_TX       |
| upf_xdp_redirect | peer_address | XDP_REDIRECT |

# 

packet\_type

| upf_rx    |  | packet_type |  |
|-----------|--|-------------|--|
| upf_route |  | packet_type |  |

## upf\_rx packet\_type

- arp - ARP
- icmp - ICMP
- icmp6 - ICMPv6
- ip4 - IPv4
- ip6 - IPv6
- tcp - TCP
- udp - UDP
- other -
- gtp-echo - GTP /
- gtp-pdu - GTP-U PDU
- gtp-other - GTP
- gtp-unexp - / GTP

## upf\_route packet\_type

- ip4-cache - IPv4
- ip4-ok - IPv4 FIB
- ip4-error-drop - IPv4 FIB
- ip4-error-pass - IPv4 FIB
- ip6-cache - IPv6
- ip6-ok - IPv6 FIB
- ip6-error-drop - IPv6 FIB
- ip6-error-pass - IPv6 FIB



# PFCP 表

表は UPF によって管理される PFCP セッションを表す

| 名前                                | 型       | キー                  | 説明                                |
|-----------------------------------|---------|---------------------|-----------------------------------|
| upf_pfcpsessions                  | 整数<br>0 |                     | 現在の PFCP セッションの数                  |
| upf_pfcpsessionsassociations      | 整数<br>0 |                     | 現在の PFCP セッションの数の合計               |
| upf_pfcpsessionsassociationstatus | 整数<br>0 | node_id,<br>address | 現在の PFCP セッションのステータス<br>1=成功 0=失敗 |
| upf_pfcpsessionspernode           | 整数<br>0 | node_id,<br>address | 現在の PFCP セッションの数                  |

# URR 表

表は PFCP セッションによって管理される URR の情報を表す

| 名前                         | 型       | キー           | 説明                         |
|----------------------------|---------|--------------|----------------------------|
| upf_urruplinkvolumebytes   | 整数<br>0 | peer_address | アップリンクの URR のバイト数          |
| upf_urrdownlinkvolumebytes | 整数<br>0 | peer_address | ダウンリンクの URR のバイト数          |
| upf_urrtotalvolumebytes    | 整数<br>0 | peer_address | アップリンク + ダウンリンクの URR のバイト数 |

表は PFCP セッションによって管理される URR の情報を表す REST API の `/api/v1/urr_map` 表

□□□□□□

□□□□□□□□□□□□□□□□□□ UE □□□□□□□□□□ UPF □□□□□□□□□□□□□□ UE □□□□□□□□□□□□

| 변수명                        | 타입       | 유니트    | 설명                             |
|----------------------------|----------|--------|--------------------------------|
| upf_buffer_packets_total   | uint32_t | 개      | UPF 버퍼에 저장된 패킷의 총 개수           |
| upf_buffer_packets_dropped | uint32_t | reason | UPF 버퍼에서 드롭된 패킷의 개수            |
| upf_buffer_packets_flushed | uint32_t | 개      | UPF 버퍼에서 플러시된 패킷의 개수           |
| upf_buffer_packets_current | uint32_t | 개      | UPF 버퍼에 현재 저장된 패킷의 개수          |
| upf_buffer_bytes_total     | uint64_t | 바이트    | UPF 버퍼에 저장된 패킷의 총 바이트 수        |
| upf_buffer_bytes_current   | uint64_t | 바이트    | UPF 버퍼에 현재 저장된 패킷의 바이트 수       |
| upf_buffer_fars_active     | uint32_t | 개      | UPF 버퍼에 현재 저장된 패킷의 개수 (FAR 포함) |

| 计数器                                              | 单位  | 范围          | 说明                                                            |
|--------------------------------------------------|-----|-------------|---------------------------------------------------------------|
| upf_buffer_listener_packets_received_total       | 无单位 | 无限制         | 接收到的 eBPF 数据包总数<br>包括接收到的数据包<br>接收到的数据包<br>接收到的数据包<br>接收到的数据包 |
| upf_buffer_listener_packets_buffered_total       | 无单位 | 无限制         | 缓冲的数据包总数<br>缓冲的数据包<br>缓冲的数据包<br>缓冲的数据包                        |
| upf_buffer_listener_errors_total                 | 无单位 | type        | 错误总数<br>错误总数<br>错误总数<br>错误总数                                  |
| upf_buffer_listener_error_indications_sent_total | 无单位 | remote_peer | 发送的错误指示总数<br>TEID<br>GTP-U<br>GTP-U<br>GTP-U                  |
| upf_buffer_flush_success_total                   | 无单位 | 无限制         | 成功刷新的总数<br>成功刷新的总数<br>成功刷新的总数                                 |
| upf_buffer_flush_errors_total                    | 无单位 | reason      | 刷新错误的总数<br>刷新错误的总数<br>刷新错误的总数                                 |

| 項目名                                 | 単位 | 説明          | 初期値 |
|-------------------------------------|----|-------------|-----|
| upf_buffer_flush_packets_sent_total | 個  | パケットを送信した総数 | 0   |

**upf\_buffer\_packets\_dropped reason**

- expired - TTL 超過
- global\_limit - 総数制限
- far\_limit - FAR 制限
- cleared - クリア

**upf\_buffer\_listener\_errors\_total type**

- read\_error - 読み取りエラー
- too\_small - GTP 太小
- invalid\_gtp\_type - G-PDU GTP 無効
- unknown\_teid - TEID 不明
- not\_buffering\_far - FAR バッファリング
- truncated\_ext - GTP 拡張部切断
- no\_payload - GTP ペイロードなし
- buffer\_full - バッファ満

**upf\_buffer\_flush\_errors\_total reason**

- far\_lookup\_failed - eBPF による FAR 検索失敗
- no\_forw\_action - FAR による転送アクションなし
- connection\_failed - UDP 接続失敗

パケット送信エラー

PFCEP によるパケット送信エラー

| 이름                                          | 유형                    | 범위                     | 설명                               |
|---------------------------------------------|-----------------------|------------------------|----------------------------------|
| upf_dldr_sent_total                         | Integer<br>32비트<br>대수 | 0 ~ 2 <sup>32</sup> -1 | 이 SMF가 DLDR를 보낸 횟수               |
| upf_dldr_send_errors                        | Integer<br>32비트<br>대수 | 0 ~ 2 <sup>32</sup> -1 | DLDR 전송 실패 횟수                    |
| upf_dldr_active_notifications               | Integer<br>32비트<br>대수 | 0 ~ 2 <sup>32</sup> -1 | DLDR를 통해 FAR에 전송된 알림 수           |
| upf_far_index_size                          | Integer<br>32비트<br>대수 | 0 ~ 2 <sup>32</sup> -1 | FarIndex의 크기 (FAR에 저장된 항목 수)     |
| upf_far_index_registrations_total           | Integer<br>32비트<br>대수 | 0 ~ 2 <sup>32</sup> -1 | FarIndex에 등록된 FAR의 수             |
| upf_far_index_unregistrations_total         | Integer<br>32비트<br>대수 | 0 ~ 2 <sup>32</sup> -1 | FarIndex에서 등록 해제된 FAR의 수         |
| upf_buffer_notify_to_flush_duration_seconds | Integer<br>32비트<br>대수 | pfcp_peer              | DLDR가 버퍼를 flush할 때까지 대기하는 시간 (초) |

**upf\_buffer\_notify\_to\_flush\_duration\_seconds:**

- 이 값은 0.01, 0.05, 0.1, 0.5, 1.0, 2.0, 5.0, 10.0, 30.0, 60.0 중 하나여야 합니다.
- 이 값은 pfcp\_peer가 SMF/PGW-C에 전송되는 10.100.50.241에 의해 설정됩니다.

- 5G UPF 5G Core SMF 5G Core SMF 5G Core SMF
- 5G Core SMF 5G Core SMF 5G Core SMF

GTP-U 5G Core SMF

5G Core SMF GTP-U 5G Core SMF TEID 5G Core SMF 5G Core SMF

| 5G Core SMF                                                 | 5G Core SMF | 5G Core SMF           |
|-------------------------------------------------------------|-------------|-----------------------|
| upf_buffer_listener_error_indications_sent_total            | 5G Core SMF | node_id, peer_address |
| upf_buffer_listener_error_indications_received_total        | 5G Core SMF | node_id, peer_address |
| upf_buffer_listener_error_indication_sessions_deleted_total | 5G Core SMF | node_id, peer_address |

- `node_id` PFCP ID `"pgw-u-1"` `"smf-1"` PFCP `"unknown"`
- `peer_address` IP `"192.168.50.10"`

UPF

- UPF TEID GTP-U TEID UPF
- eNodeB/gNodeB UPF/
- UPF

UPF

- UPF GTP-U PGW-U SGW-U UPF TEID
- TEID
- UPF

UPF

- 
- TEID
- 
- 

PromQL

```
#
rate(upf_buffer_listener_error_indications_received_total[5m])

#
upf_buffer_listener_error_indication_sessions_deleted_total{peer_addr

UPF TEID
sum by (node_id, peer_address) (upf_buffer_listener_error_indications
```

## eBPF

eBPF



| 項目名                   | 型        | 単位       | 説明         |
|-----------------------|----------|----------|------------|
| upf_ebpf_map_capacity | uint32_t | map_name | eBPF 最大容量  |
| upf_ebpf_map_used     | uint32_t | map_name | eBPF 現在使用量 |

map\_name の種類

- pdr\_map - PDR 登録情報
- far\_map - FQDN 登録情報
- qer\_map - QoS 登録情報
- session\_map - セッション登録情報
- teid\_map - TEID 登録情報
- ue\_ip\_map - UE IP 登録情報

## ❓❓ Prometheus

概要

メトリクスを `metrics_address` に `/metrics` 経由で `:9090` ポートで取得

```
curl
curl http://localhost:9090/metrics

出力
upf_pfcp_sessions 42
upf_pfcp_associations 2
upf_urr_total_volume_bytes{peer_address="10.100.50.241"}
1048576000
```

## Prometheus

OmniUPF の Prometheus 設定は `prometheus.yml`

```
scrape_configs:
 - job_name: 'omniupf'
 static_configs:
 - targets: ['localhost:9090']
```

## Grafana 部署

部署 Grafana 部署

- 部署 Grafana
- 部署 PFCP 部署
- 部署 Grafana
- 部署 Grafana
- eBPF 部署

### 部署

- 部署 - 部署 Grafana
- 部署 - 部署 metrics\_address 部署 UPF 部署
- **Web UI** 部署 - 部署 Grafana
- 部署 - eBPF 部署
- 部署 - 部署 PDR/FAR/QR/URR 部署
- 部署 - 部署 Grafana

# 

## 

- 1.
- 2.
- 3.
- 4.
- 5.
- 6.
- 7.

## 

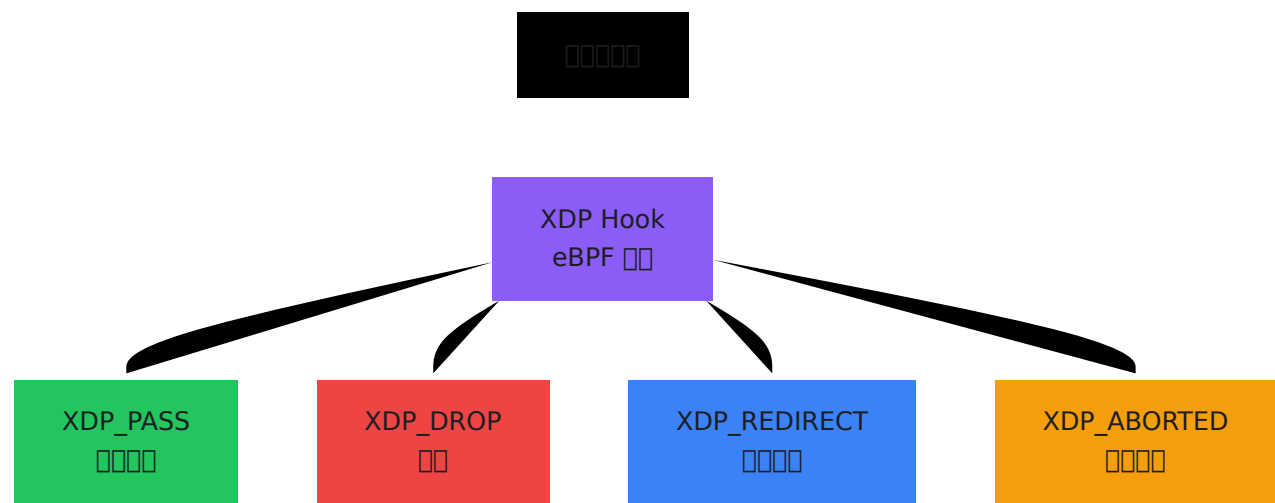
## 

|             |       |    | RX/TX              |
|-------------|-------|----|--------------------|
|             | N3/N6 |    | N3 RX/TX, N6 RX/TX |
| <b>XDP</b>  |       |    | XDP                |
|             |       |    | FIB                |
| <b>eBPF</b> |       | 10 |                    |
|             |       | 5  | FAR                |

- **RX N3  $\approx$  TX N6**  RAN 
- **RX N6  $\approx$  TX N3**  RAN
- 
  -  >> 
  - 
  - 

# XDP 简介

XDP (eXpress Data Path) 是一种高性能的数据包处理技术



特点

- XDP 位于网络栈的最底层

- **TX** XDP 000000000000
- 00000000000000000000000000000000
- 00000000000000000000000000000000
- **TX** XDP 00000000

0000

- 00 > 00 eBPF 00000000000000000000000000000000
- 00 > 00 00000000000000000000000000000000
- 00000000000000000000000000000000
- 00000000000000000000000000000000

00000000

00000000000000000000000000000000

00000000

- **RX ARP** 000000000000
- **RX GTP ECHO** GTP-U 0000/0000000000
- **RX GTP OTHER** 00 00 GTP 0000
- **RX GTP PDU** GTP-U 0000000000000000
- **RX GTP UNEXP** 0000 GTP 000000
- **RX ICMP** 000000000000 ping 000000
- **RX ICMP6** ICMPv6 0000
- **RX IP4** IPv4 0000
- **RX IP6** IPv6 0000
- **RX OTHER** 000000
- **RX TCP** 000000000000
- **RX UDP** 000000000000

0000

- 00 **GTP-U PDU** 000000000000
- 00 **ICMP** 0000000000000000

- **TCP** 及 **UDP** 的封包處理
- 網路封包的處理

## 網路

FIB (Forwarding Information Base) 的運作原理

### IPv4 FIB 運作

- 透過路由表進行封包轉送
- **OK** 表示封包成功轉送

### IPv6 FIB 運作

- 透過路由表進行 IPv6 封包轉送
- **OK** 表示 IPv6 封包成功轉送

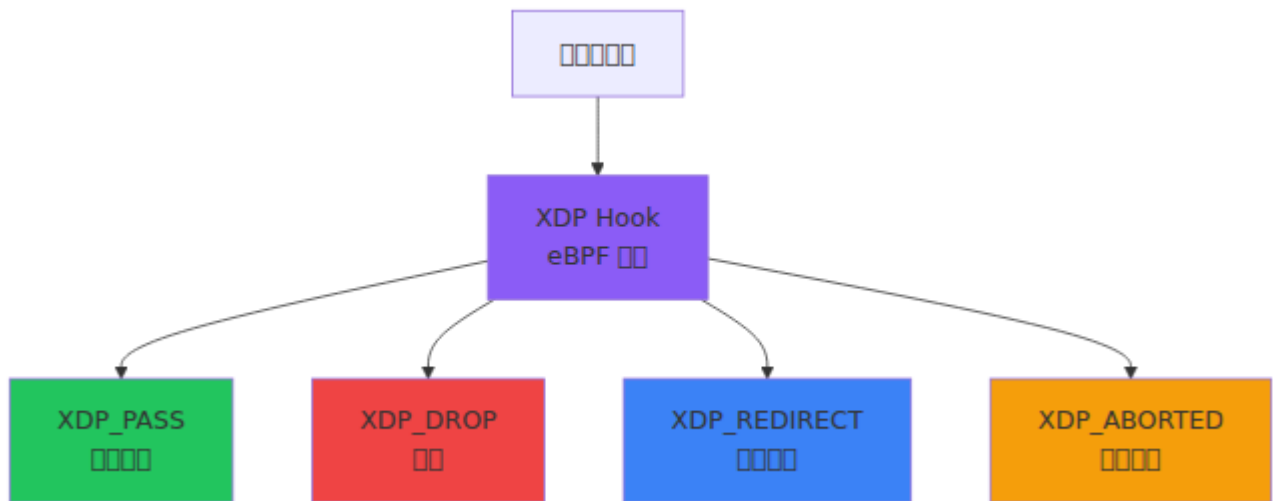
## 網路

- 透過路由表進行封包轉送
- **OK** 表示封包成功轉送
- 透過路由表進行封包轉送

## 網路

### eBPF 運作

eBPF 的運作原理



## eBPF

### far\_map ( )

- 131,070
- 4 B (FAR ID)
- 16 B ( )
- ~2.6 MB
- -

### pdr\_map\_downlin ( PDRs - IPv4)

- 131,070
- 4 B (UE IPv4 )
- 208 B (PDR )
- ~27 MB
- -

### pdr\_map\_downlin\_ip6 ( PDRs - IPv6)

- 131,070
- 16 B (UE IPv6 )
- 208 B (PDR )
- ~29 MB
- - IPv6



### **pdr\_map\_teid\_ip** (PDRs)

- 131,070
- 4 B (TEID)
- 208 B (PDR)
- ~27 MB
- -

### **qer\_map** (QoS)

- 65,535
- 4 B (QER ID)
- 32 B (QoS)
- ~2.3 MB
- - QoS

### **urr\_map** (URRs)

- 131,070
- 4 B (URR ID)
- 16 B (URR)
- ~2.6 MB
- -

表 1

| 項目                  | 評価基準      |
|---------------------|-----------|
| <b>0-50% (低)</b>    | 計画 - 計画   |
| <b>50-70% (中)</b>   | 計画 - 計画   |
| <b>70-90% (高)</b>   | 計画 - 計画 1 |
| <b>90-100% (最高)</b> | 計画 - 計画   |

表 2

表 3

1. 計画
2. 計画

### 3. 設定

設定

1. OmniUPF 設定
2. Prometheus 設定
3. OmniUPF 設定
4. Prometheus 設定
5. Prometheus 設定

設定 eBPF 設定 Prometheus 設定 UPF 設定

設定

OmniUPF 設定 Prometheus 設定

設定

設定

設定 (pps) = (設定) / (設定)

設定

- RX 設定7,000
- 10 設定17,000
- 設定 = (17,000 - 7,000) / 10 = 1,000 pps

設定

- UPF 10,000 - 100,000 pps
- UPF 100,000 - 1,000,000 pps
- UPF 1,000,000 - 10,000,000 pps

設定

- XDP 実装状況
  - CPU 実装状況
  - ネットワーク実装状況
  - その他
- 

## 計算式

計算式

$$\text{Mbps (Mbps)} = (\text{バイト数} \times 8) / (\text{時間} \times 1,000,000)$$

計算式

- RX 300 MB
- 60 秒 800 MB
- $\text{Mbps} = (300 \text{ MB} \times 8) / (60 \times 1,000,000) = 40 \text{ Mbps}$

計算式

- ネットワーク実装状況
  - ネットワークN3/N6 実装状況
  - 2 台 ネットワーク実装状況
- 

## 計算式

計算式

$$\text{割合 (\%)} = (\text{バイト数} / \text{RX バイト}) \times 100$$

計算式

- **< 0.1%** ネットワーク実装状況
- **0.1% - 1%** ネットワーク実装状況

- **1% - 5%** 100% QoS 100%
- **> 5%** 100% 100%

## 100%

- QER 100% MBR 100%
  - eBPF 100%
  - 100% TEID 100% UE IP
  - 100%
- 

## 100%

## 100%

## 100% 100%

- eBPF 100% > 90%
- XDP 100% > 0
- 100% > 5%
- UPF 100%

## 100% 1 100%

- eBPF 100% > 70%
- 100% > 1%
- 100% 100%
- 100% TTL 100% 30 100%

## 100% 100%

- eBPF 100% > 50%
- 100% 100%
- 100% PFCP 100%/100%
- URR 100%

## 目標

実現可能な範囲で

1. **Prometheus** 監視システム構築可能な **クラウド** 監視システム
2. 監視対象 OmniUPF 監視システム
3. **REST API** 監視対象 `/map_info` `/packet_stats` 監視
4. **Web UI** 監視システム構築可能な

## 概要

### 監視対象

監視対象

```
監視対象 = min(
 PDR 監視 / 2, # 000 + 監視 PDR 監視
 FAR 監視 / 2, # 監視 + 監視 FAR 監視
 QER 監視 # 監視監視監視 QER
)
```

監視

- PDR 監視131,070
- FAR 監視131,070
- QER 監視65,535

監視 =  $\min(131,070 / 2, 131,070 / 2, 65,535) = \mathbf{65,535}$  監視

### 監視

監視 **eBPF** 監視

```
監視 = $\sum (\text{監視} \times (\text{監視} + \text{監視}))$
```

## 計算

- PDR 計算  $3 \times 131,070 \times 212 \text{ B} = 83.3 \text{ MB}$
- FAR 計算  $131,070 \times 20 \text{ B} = 2.6 \text{ MB}$
- QER 計算  $65,535 \times 36 \text{ B} = 2.3 \text{ MB}$
- URR 計算  $131,070 \times 20 \text{ B} = 2.6 \text{ MB}$
- 合計 ~91 MB 必要

## 環境構築

- 環境構築 `ulimit -l`
- 環境構築 2 環境構築
- 環境構築

## 設定

### 環境構築

#### 1. 環境構築

- 環境構築 ~5 Mbps
- 環境構築 ~1 Mbps
- VoIP ~0.1 Mbps

#### 2. 環境構築

$$\text{環境構築} = \text{環境構築} \times \text{環境構築}$$

#### 3. 環境構築

$$\text{環境構築} = \text{環境構築} \times 2 \quad \# 100\% \text{ 環境構築}$$

## 結果

- 10,000 環境構築
- 環境構築 2 Mbps

- 20 Gbps
- 40 Gbps (N3 + N6 100G)

100G

100G

1. 100Gbps
2. 100Gbps
3. 100Gbps

100Gbps

$$\text{Throughput} = (\text{Tx} - \text{Rx}) / (\text{Latency})$$

100G

- 30,000
- 65,535 100G
- 2,000 100G
- $(65,535 - 30,000) / 2,000 = \mathbf{17.8}$  100G

100Gbps 12 100Gbps 5 100Gbps

100Gbps

100Gbps

100Gbps > 1% 100Gbps

100Gbps

1. 100Gbps → 100Gbps
2. 100Gbps
3. 100 XDP 100Gbps XDP 100Gbps



□□□□

- **QER** □□□□□□ QER MBR □□□□□□
- □□ **TEID**□□□□□ PDR TEID □□□ gNB □□□□
- □□ **UE IP**□□□□□ PDR □□□□□ UE IP
- □□□□□□□□□□□□□□

□□□□

- □□□□□□□□□□□□□□ QER MBR
- □□ SMF □□□□□□□□□□ PDR
- □□□□□□□□□□□□□□

---

# XDP □□□□

□□□XDP □□ > 0

□□□

1. □□□□□ → XDP □□
2. □□□□□□
3. □□ OmniUPF □□□□□ eBPF □□

□□□□

- eBPF □□□□□□
- □□□□□□□□
- eBPF □□□□□□
- □□□□

□□□□

- □□□□ OmniUPF □□
  - □□□□□□□□□□□□□□□□Linux 5.4+□
  - □□ eBPF □□□□
  - □□□□□□□□□□□□□□
-

## 目標

ネットワークの効率性を 100%

目標

1. ネットワーク
2. ネットワーク 100%
3. ネットワークの効率性

目標

1. ネットワークの効率性
2. ネットワーク SMF ネットワーク
3. ネットワーク FAR ネットワーク

目標

1. ネットワーク eBPF ネットワーク
2. ネットワーク UPF ネットワークの効率性
3. ネットワークの効率性

## 目標

ネットワークの効率性を CPU ネットワーク

目標

1. ネットワークの効率性
2. ネットワーク XDP ネットワークの効率性
3. ネットワーク UPF ネットワーク CPU ネットワーク
4. ネットワーク N3/N6 ネットワーク

目標

- ネットワーク UPF ネットワーク
- ネットワーク CPU ネットワーク

- 网络策略
- eBPF 网络策略

## 网络策略

- 网络策略 UPF 网络策略
  - CPU 网络 RSS 网络策略
  - 网络策略网络策略
  - 网络 eBPF 网络策略
- 

## 网络策略

- 网络策略 - 网络 Prometheus 网络策略
- **UPF** 网络策略 - 网络 UPF 网络策略
- 网络策略 - PDR、FAR、QER、URR 网络策略
- **Web UI** 网络策略 - 网络策略
- 网络策略 - 网络策略
- 网络策略 - eBPF 网络策略

# N9 Loopback

## SGWU & PGWU

概要

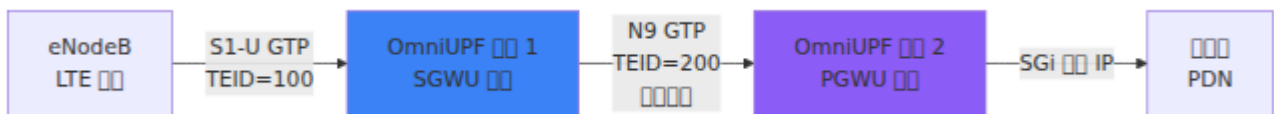
OmniUPF において、SGWU (Service Gateway-U) と PGWU (PDN Gateway-U) を N9 で接続する。

- 4G EPC - UPF
- 5G Core - UPF
- 5G Core - EPC
- 5G Core - EPC

N3 と N9 の IP 接続。OmniUPF において SGWU と PGWU を eBPF で接続する。

接続

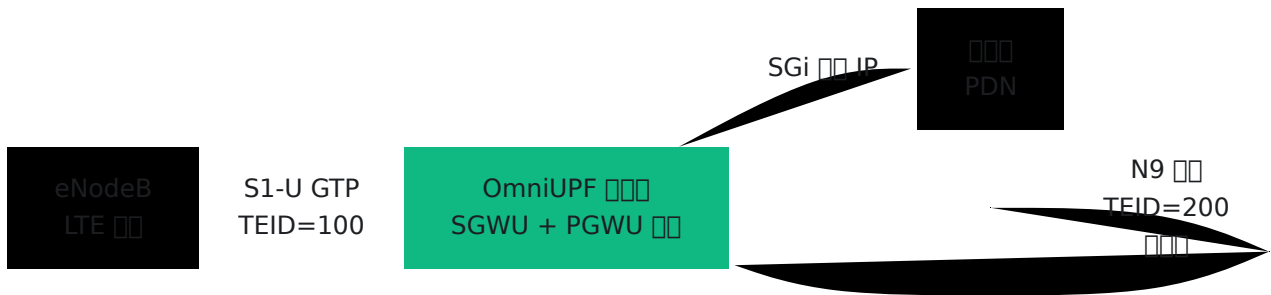
接続図



手順

- eNodeB → SGWU (GTP TEID=100) S1-U
- SGWU PDR GTP (TEID=200)
- N9 PGWU
- PGWU GTP (TEID=200)
- 2 XDP + 1

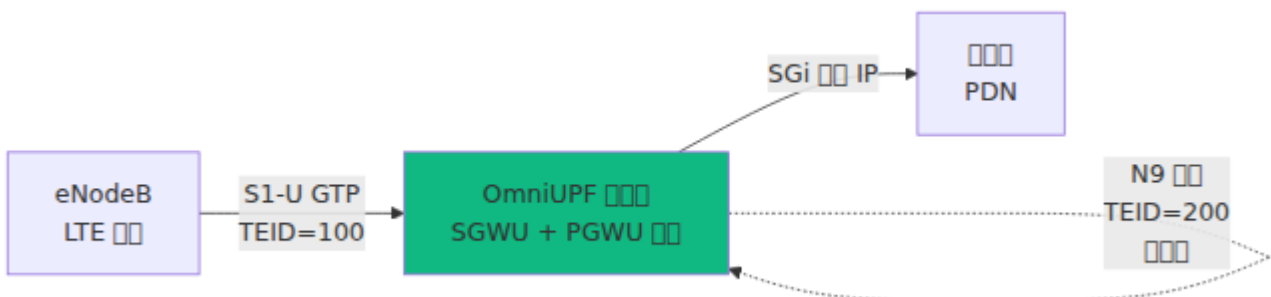
## N9



## N9

1. eNodeB → SGWU GTP (TEID=100) S1-U
2. SGWU PDR
3. IP = IP (10.0.1.10)
4. GTP TEID 200 (PGWU)
5. PGWU
6. **1** **XDP**

**eNodeB → SGWU → PGWU →**

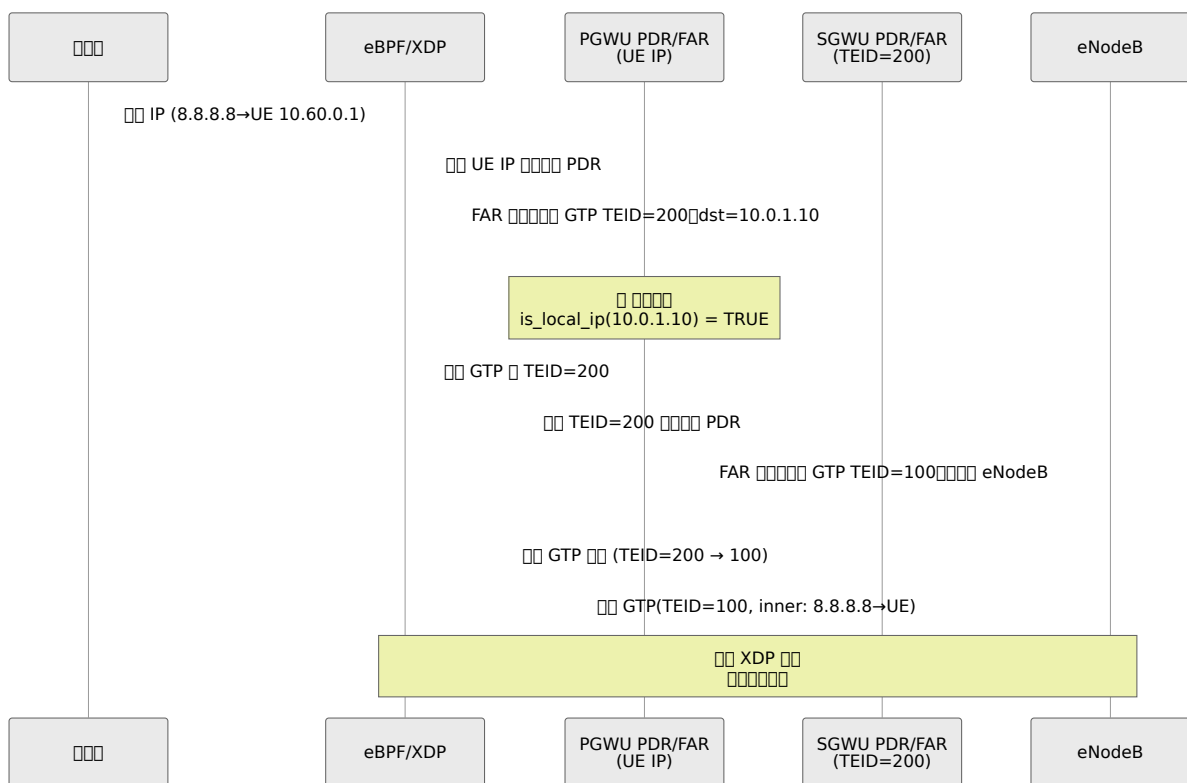


`cmd/ebpf/xdp/n3n6_entrypoint.c` 349-403

1. eNodeB GTP TEID=100
2. **PDR** SGWU PDR (TEID=100)

3. **FAR** [] [] TEID=200 [] GTP[] 10.0.1.10
4. [] [] `is_local_ip(10.0.1.10)` [] TRUE
5. [] **TEID** [] `ctx->gtp->teid` [] 100 [] 200 ([]) []
6. [] [] TEID=200 [] PDR (PGWU [])
7. **FAR** [] [] GTP [] [] [] []
8. [] [] IP [] [] N6 []

## [] [] [] [] [] → PGWU → SGWU → eNodeB



[] [] [] `cmd/ebpf/xdp/n3n6_entrpoint.c` [] 137-194 [] (IPv4) [] 265-322 [] (IPv6)

[] [] [] []

1. [] [] [] [] [] IP [] [] [] [] UE (10.60.0.1)
2. **PDR** [] [] [] UE IP [] [] [] PDR (PGWU [])
3. **FAR** [] [] [] TEID=200 [] GTP[] 10.0.1.10
4. [] [] [] `is_local_ip(10.0.1.10)` [] TRUE
5. [] **GTP** [] [] TEID=200 [] [] []
6. [] [] [] [] TEID=200 [] PDR (SGWU [])

7. **FAR** 配置 GTP 配置 eNodeB TEID=100

8. 配置 GTP 配置 S1-U 配置 (eNodeB)

---

配置

配置

配置

- **SGWU-C**配置 OmniUPF PCF 配置 (配置 192.168.1.10:8805)
- **PGWU-C**配置 配置 OmniUPF PCF 配置

配置

- **N3** 配置 **N9** 配置 **IP** 配置
  - **SGWU-C** 配置 **PGWU-C** 配置 **IP** 配置
- 

## OmniUPF 配置

config.yml:

```

配置
interface_name: [eth0] # S1-U 与 N9 接口
xdp_attach_mode: native # 是否启用 xdp 模式

PCF 配置
pfcf_address: ":8805" # 监听地址 8805
pfcf_node_id: "192.168.1.10" # OmniUPF 的 PCF 节点 ID

网络配置
n3_address: "10.0.1.10" # S1-U/N3 接口 IP
n9_address: "10.0.1.10" # N9 接口 IP 与 N3 接口

APIs
api_address: ":8080" # REST API
metrics_address: ":9090" # Prometheus 监控接口

配置
ueip_pool: "10.60.0.0/16" # UE IP 地址池
teid_pool: 65535 # TEID 地址池

其他
max_sessions: 100000 # 最大 UE 数

```

配置

- `n3_address` 与 `n9_address` 必须一致
- 是否启用 PCF 模式
- 是否启用 SGWU + PGWU 模式 `max_sessions` 值

配置

SGWU-C 配置



```
[] OmniUPF PFCP []
upf_pfcip_address: "192.168.1.10:8805"

S1-U [] [] OmniUPF n3_address []
sgwu_s1u_address: "10.0.1.10"

N9 [] [] PGWU[] OmniUPF []
sgwu_n9_address: "10.0.1.10"
```

## PGWU-C []

```
[] [] OmniUPF PFCP []
upf_pfcip_address: "192.168.1.10:8805"

N9 [] [] SGWU []
pgwu_n9_address: "10.0.1.10"

SGi []
pgwu_sgi_address: "192.168.100.1"
```

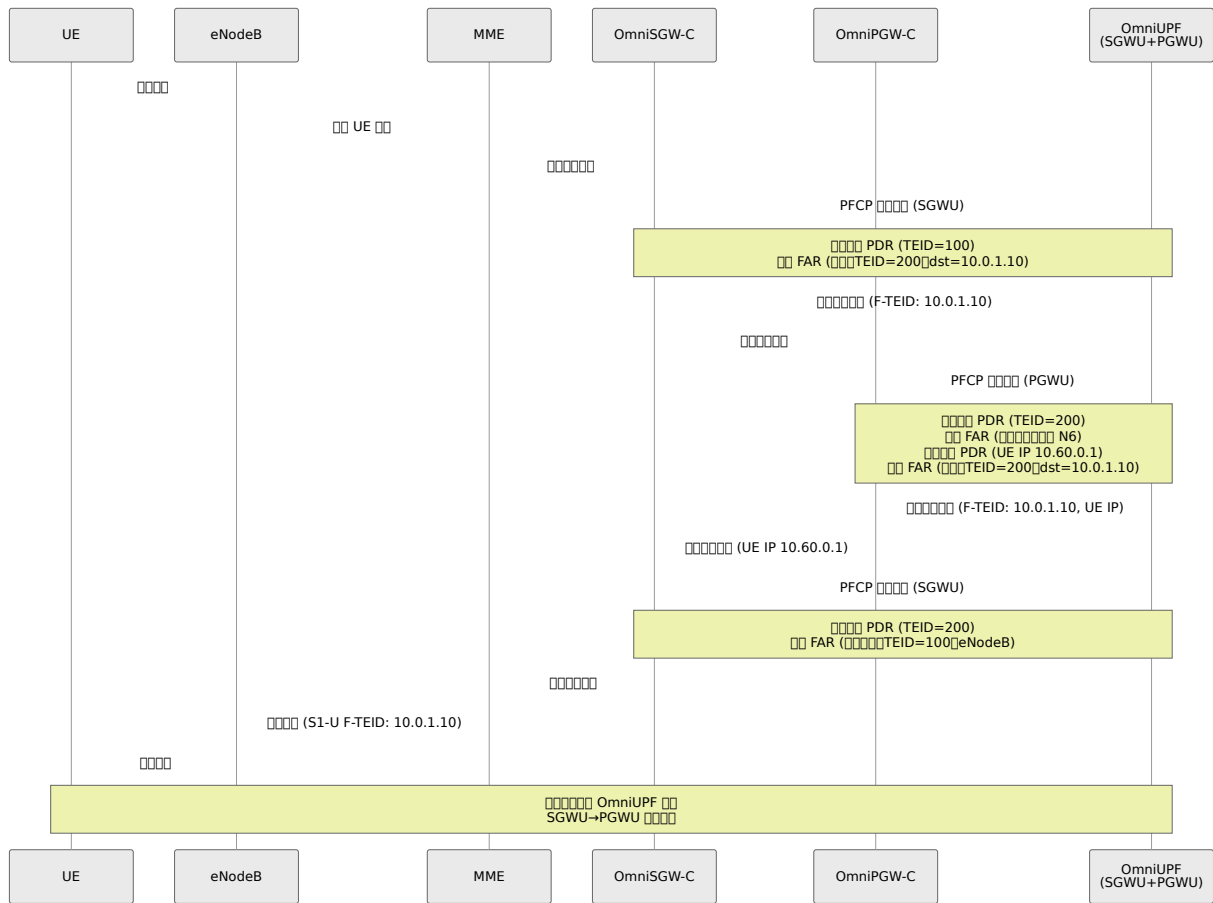
[]

- [] [] **PFCP** [] (:8805)
- OmniUPF [] SGWU-C [] PGWU-C [] [] **PFCP** []
- [] ID []

[][][]

**UE** [] **PDU** []

[] UE []



PFCP

SGWU OmniSGW-C

- PDR TEID=100 eNodeB → FAR TEID=200 dst=10.0.1.10
- PDR TEID=200 PGWU → FAR TEID=100 eNodeB

PGWU OmniPGW-C

- PDR TEID=200 SGWU → FAR
- PDR UE IP=10.60.0.1 → FAR TEID=200 dst=10.0.1.10

N9

XDP

```
查看 eBPF 数据
```

```
sudo cat /sys/kernel/debug/tracing/trace_pipe | grep loopback
```

输出结果

```
upf: [n3] session for teid:100 -> 200 remote:10.0.1.10
upf: [n9-loopback] self-forwarding detected, processing inline
TEID:200
upf: [n9-loopback] decapsulated, routing to N6

upf: [n6] use mapping 10.60.0.1 -> teid:200
upf: [n6-loopback] downlink self-forwarding detected, processing
inline TEID:200
upf: [n6-loopback] SGWU updating GTP tunnel to eNodeB TEID:100
upf: [n6-loopback] forwarding to eNodeB
```

---

## 配置 REST API 数据

配置 PFCP 数据

```
curl http://localhost:8080/api/v1/upf_pipeline | jq
```

输出结果

```
{
 "associations": [
 {
 "node_id": "sgwc.example.com",
 "address": "192.168.1.20:8805",
 "sessions": 1000
 },
 {
 "node_id": "pgwc.example.com",
 "address": "192.168.1.21:8805",
 "sessions": 1000
 }
],
 "total_sessions": 2000
}
```

SGWU-C PGWU-C

---

```
curl http://localhost:8080/api/v1/sessions | jq '.sessions[] |
{local_seid, ue_ip, uplink_teid}'
```

```
{
 "local_seid": 12345,
 "ue_ip": "10.60.0.1",
 "uplink_teid": 100
}
{
 "local_seid": 67890,
 "ue_ip": "10.60.0.1",
 "uplink_teid": 200
}
```

UE

- SGWU-C TEID=100 S1-U
- PGWU-C TEID=200 N9

```
curl http://localhost:8080/api/v1/xdp_stats | jq
```

- xdp\_processed eBPF
- xdp\_pass
- xdp\_redirect XDP
- xdp\_tx

**N9**

- xdp\_pass
- xdp\_tx xdp\_redirect

**N9**

n3\_address ≠ n9\_address

```
n3
n3_address: "10.0.1.10"
n9_address: "10.0.1.20" # n3 IP address

n9
n3_address: "10.0.1.10"
n9_address: "10.0.1.10" # n9 IP address
```

curl

```
curl http://localhost:8080/api/v1/dataplane_config | jq
```

jq

```
{
 "n3_ipv4_address": "10.0.1.10",
 "n9_ipv4_address": "10.0.1.10"
}
```

---

## PGWU PDR

PGWU [n9-loopback] no PDR for destination TEID

PGWU PGWU TEID

1.

1. PGWU PFCP

```
curl http://localhost:8080/api/v1/sessions | jq '.sessions[] |
select(.uplink_teid == 200)'
```

2. PGWU FAR

```
curl http://localhost:8080/api/v1/far_map | jq '.[[] |
select(.teid == 200)'
```

PGWU-C SGWU-C N9 TEID

## CPU

CPU

eBPF

```
eBPF
sudo bpftool map dump name pdr_map_teid_ip4 | wc -l
sudo bpftool map dump name far_map | wc -l
```

- max\_sessions
- QER
- 

eNodeB

```
buffer_port: 22152
buffer_max_packets: 20000 #
buffer_max_total: 100000
buffer_packet_ttl: 30 #
```

📄

```
curl http://localhost:8080/api/v1/upf_buffer_info | jq
```

## N9 性能指标

📊

| 指标      | 配置             | 性能 (N9 指标) | 单位          |
|---------|----------------|------------|-------------|
| 吞吐量     | 1-5 Gbps       | < 1 Gbps   | 约 1000 Gbps |
| 延迟      | 低延迟            | 约 CPU/网络   | 约 2-3 ms    |
| CPU 使用率 | 2x XDP 处理 + 其他 | 1x XDP 处理  | 约 40-50%    |
| 内存使用    | 内存池化           | 内存池化       | 约           |

📋

- 性能指标 OmniUPF 性能指标
- 性能指标 性能指标IP 指标
- 性能指标 性能指标
- 性能指标 性能指标
- 性能指标 性能指标 eBPF 性能指标

📋

性能指标

- 性能指标 性能指标
- 性能指标/性能指标 < 100K 性能指标
- 性能指标/性能指标 性能指标 VM 性能指标 EPC 性能指标



- **IPsec Tunnel** SGWU ↔ PGWU **IPsec Tunnel**
- **IPsec Tunnel** > 1M **IPsec Tunnel**
- **IPsec Tunnel** **IPsec Tunnel** SGW ↔ PGW

- 降低网络延迟
- 降低 **40-50%** 的 **CPU** 资源消耗
- 降低功耗 - 提升能效比
- 降低延迟 - 提升用户体验
- 支持 3GPP 标准 - 支持 PCF/GTP-U 等

通过 `n3_address == n9_address` 实现地址匹配 - 实现 OmniUPF 的 eBPF 性能提升

性能提升

- 查看 [CONFIGURATION.md](#)
- 查看 [ARCHITECTURE.md](#)
- 查看 [METRICS.md](#)
- 查看 [MONITORING.md](#)
- 查看 [OPERATIONS.md](#)
- 查看 [TROUBLESHOOTING.md](#)

# PFCP 配置指南

## 简介

PFCP 是 3GPP TS 129.244 中定义的一种协议，用于在 UPF 和 PCF 之间传输策略控制信息。OmniUPF 支持 PFCP 功能，并提供了相关的配置和监控接口。

本文档将介绍 OmniUPF 中 PFCP 的配置和监控方法。

## 配置

OmniUPF 通过 Prometheus 暴露 PFCP 相关的指标，用于监控 PFCP 的运行状态。

```
upf_pfcp_rx_errors{message_name="...", cause_code="...",
peer_address="..."}
```

以下指标用于监控 PFCP 的运行状态：

- upf\_pfcp\_rx\_errors: PFCP 接收到的错误消息数量。
- upf\_pfcp\_tx\_errors: PFCP 发送的错误消息数量。
- upf\_pfcp\_sessions: 当前活跃的 PFCP 会话数量。

您可以通过 Prometheus 查询这些指标，以了解 PFCP 的运行情况。

## 监控

您可以通过以下方式监控 PFCP 的运行状态：

| 序号 | 消息名称                   | 描述                       |
|----|------------------------|--------------------------|
| 1  | <b>RequestAccepted</b> | 表示请求被接受，IE 包含会话 ID/地址/标识 |

错误码

| 错误码 | 错误名称                                   | 错误描述                                    |
|-----|----------------------------------------|-----------------------------------------|
| 64  | <b>RequestRejected</b>                 | 请求被拒绝                                   |
| 65  | <b>SessionContextNotFound</b>          | SEID 会话上下文在 UPF 中不存在                    |
| 66  | <b>MandatoryIEMissing</b>              | NodeID 或 F-SEID 或 RecoveryTimeStamp 缺失  |
| 67  | <b>ConditionalIEMissing</b>            | 条件 IE 缺失                                |
| 69  | <b>MandatoryIEIncorrect</b>            | NodeID 或 RecoveryTimeStamp 或 F-SEID 不正确 |
| 72  | <b>NoEstablishedPFCPAssociation</b>    | 没有建立 PFCP 关联                            |
| 73  | <b>RuleCreationModificationFailure</b> | PDR/FAR/QER 或 URR 创建或修改失败               |

□□□/□□□□□□

| 코드 | 이름                            | 설명                                      |
|----|-------------------------------|-----------------------------------------|
| 74 | <b>PFCPEntityInCongestion</b> | UPF 엔티티가 혼잡 상태에 있습니다.                   |
| 75 | <b>NoResourcesAvailable</b>   | 엔티티가 eBPF 리소스를 사용할 수 없습니다. TEID: [TEID] |
| 77 | <b>SystemFailure</b>          | 엔티티가 eBPF 시스템에 실패했습니다. [디버그 정보]         |

□ □ □ □ □ □ □ □

| 項目 | 項目名                                 | 説明                                              |
|----|-------------------------------------|-------------------------------------------------|
| 68 | <b>InvalidLength</b>                | IE 長さが OmniUPF によってサポートされていない                   |
| 70 | <b>InvalidForwardingPolicy</b>      | UPF によってサポートされていない OmniUPF 転送ポリシー               |
| 71 | <b>InvalidFTEIDAllocationOption</b> | 転送 F-TEID によってサポートされていない OmniUPF 転送オプション        |
| 76 | <b>ServiceNotSupported</b>          | 転送サービスによってサポートされていない OmniUPF 転送サービス             |
| 78 | <b>RedirectionRequested</b>         | UPF によってサポートされていない UPF 転送サービス<br>OmniUPF 転送サービス |

|  |  |  |  |  |  |  |
|--|--|--|--|--|--|--|
|  |  |  |  |  |  |  |
|--|--|--|--|--|--|--|

□ □ □ □ □ □

NodeID

SMF → UPF: NodeID  
UPF → SMF: MandatoryIEMissing

SMF NodeID IE

NodeID

SMF → UPF: NodeID="invalid"  
UPF → SMF: MandatoryIEIncorrect

NodeID FQDN IPv4/IPv6

SMF → UPF: RecoveryTimeStamp  
UPF → SMF: MandatoryIEMissing

RecoveryTimeStamp

SMF → UPF:  
UPF → SMF: NoEstablishedPFCPAssociation

PFCP

SMF → UPF:  
UPF FARQERURR  
UPF PDReBPF  
UPF → SMF: RuleCreationModificationFailure

00000

- 00 eBPF 00000000 000000
- 0 UPF 0000000000
- 0000000000

00000 **F-SEID**

SMF → UPF: 00000000 CP F-SEID0  
UPF → SMF: 0000000000MandatoryIEMissing0

0000000000000000 CP F-SEID0

00000000

00000 **SEID**

SMF → UPF: 00000000SEID=123450  
UPF 00 SEID 12345 000  
UPF → SMF: 0000000000SessionContextNotFound0

00000

- 00 SEID 0000000000000000
- 000000000000
- 00000000 UPF 000N9 000000

00000000

00000 **SEID**

SMF → UPF: 00000000SEID=678900  
UPF 00 SEID 67890 000  
UPF → SMF: 0000000000SessionContextNotFound0

00000SEID 0000000000000000

# 🔍 故障調査と監視

## 📊 Prometheus 📊

📊 Prometheus 📊

```
📊 Prometheus 📊
rate(upf_pfcpx_errors{cause_code!="RequestAccepted"}[5m])

📊 Prometheus 📊
topk(5, sum by (cause_code) (upf_pfcpx_errors))

📊 SMF 📊
sum by (peer_address, cause_code)
(upf_pfcpx_errors{cause_code!="RequestAccepted"})

📊 Prometheus 📊
upf_pfcpx_errors{message_name="SessionEstablishmentRequest",
cause_code!="RequestAccepted"}
```

## 📊 Web UI

📊 📊 📊

- 📊 📊 📊 📊 📊
- 📊 📊 📊 / 📊
- 📊 📊 📊

📊 📊 📊

- eBPF 📊 📊 RuleCreationModificationFailure 📊
- 📊 📊

📊 Web UI 📊 📊 📊

📊 📊 📊

📊 MandatoryMissing 📊



1. SMF 消息 IE
2. PCF 消息
3. SMF 消息 IE 消息

### 消息 **RuleCreationModificationFailure**

1. eBPF 消息 GET /api/v1/map\_info
2. 消息 upf\_ebpf\_map\_used / upf\_ebpf\_map\_capacity
3. 消息 70% 消息
4. 消息 消息

### **NoEstablishedPCFPAssociation** 消息

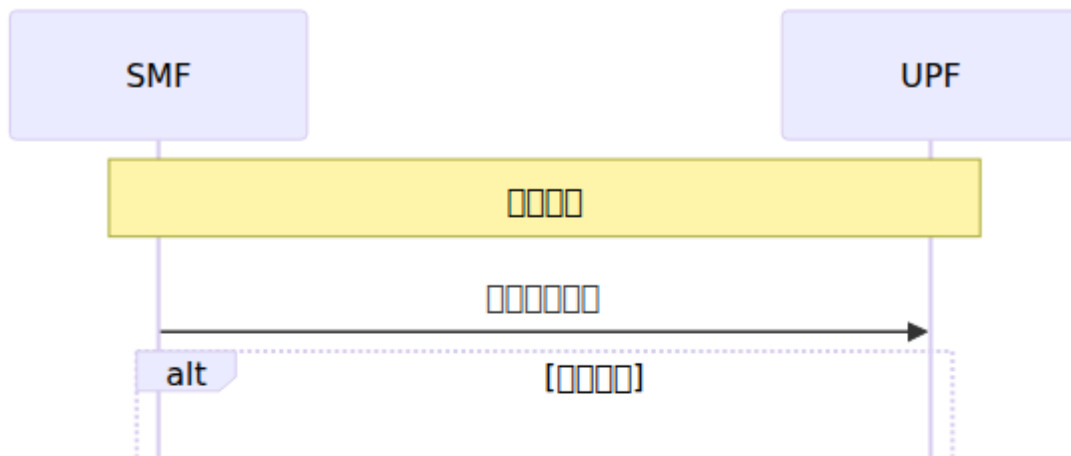
1. 消息 GET /api/v1/pfcp\_associations
2. 消息
3. 消息
4. SMF 与 UPF 消息

### 消息 **SessionContextNotFound**

1. 消息 SEID
2. 消息
3. N9 消息 UPF 消息
4. 消息 GET /api/v1/pfcp\_sessions

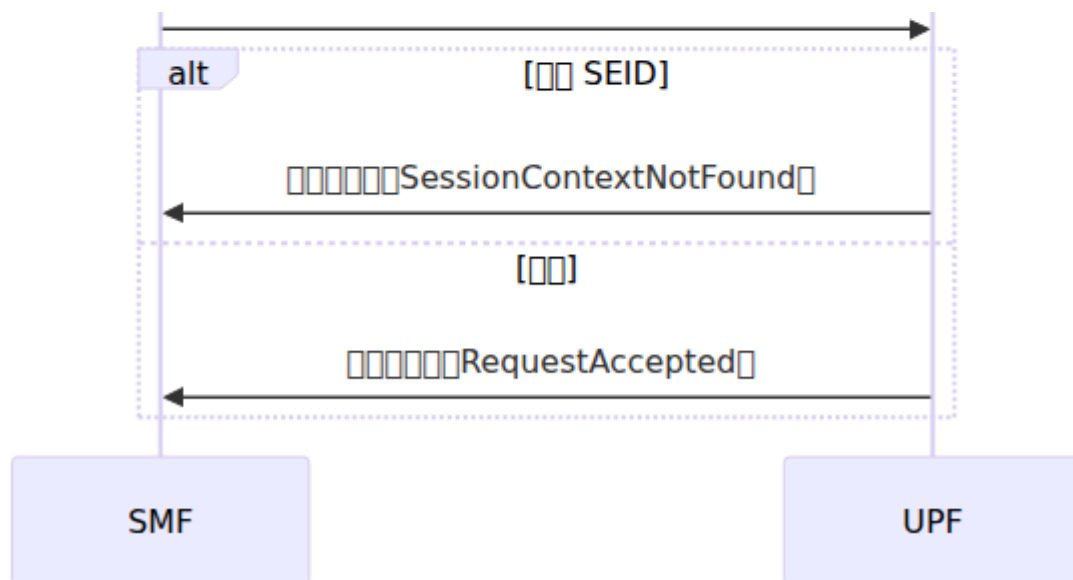
□□□□□□□□□□

□□□□□□



OmniCharge   OmniRAN   Downloads     Omnitouch Website





[] [] [] []

[] [] [] []

```

[] [] [] [] []
- alert: PfcphighRejectionRate
 expr: |
 rate(upf_pfcph_rx_errors{cause_code!="RequestAccepted"}[5m]) > 0.1
 annotations:
 summary: "[] PFCP [] [] [] [{ $value }] /s"

[] [] [] [] []
- alert: PfcpruleCreationFailures
 expr: |

rate(upf_pfcph_rx_errors{cause_code="RuleCreationModificationFailure"}
[5m]) > 0
 annotations:
 summary: "[] [] PFCP [] [] [] []"

[] [] [] [] []
- alert: PfcphNoAssociation
 expr: |

rate(upf_pfcph_rx_errors{cause_code="NoEstablishedPFCPAssociation"}
[5m]) > 0
 annotations:
 summary: "[] [] [] [] [] [] [] [] PFCP []"

```

# 3GPP 4G/5G

OmniUPF 4G/5G 网络

- **3GPP TS 129.244 v16.4.0** - PCF 规范
- **8.2.1** - 5G IE 规范
- **8.19** - 5G 规范

## 4G/5G

- **PCF** 规范 - PCF 规范
- **upf\_pfcpx\_errors** 规范
- **PCF** - 5G 规范
- **PCF** - PCF 规范
- **Web UI** 规范 - 5G 规范

**UE** ☐ ☐ ☐ ☐

□□□□□

- API - 外部サービスと連携する API
- Web UI - Web UI を提供する

11

UPF 00000000 **FRR** 00000000 000000000000UE IP 00000000 UE 000000000000  
000000000000000000

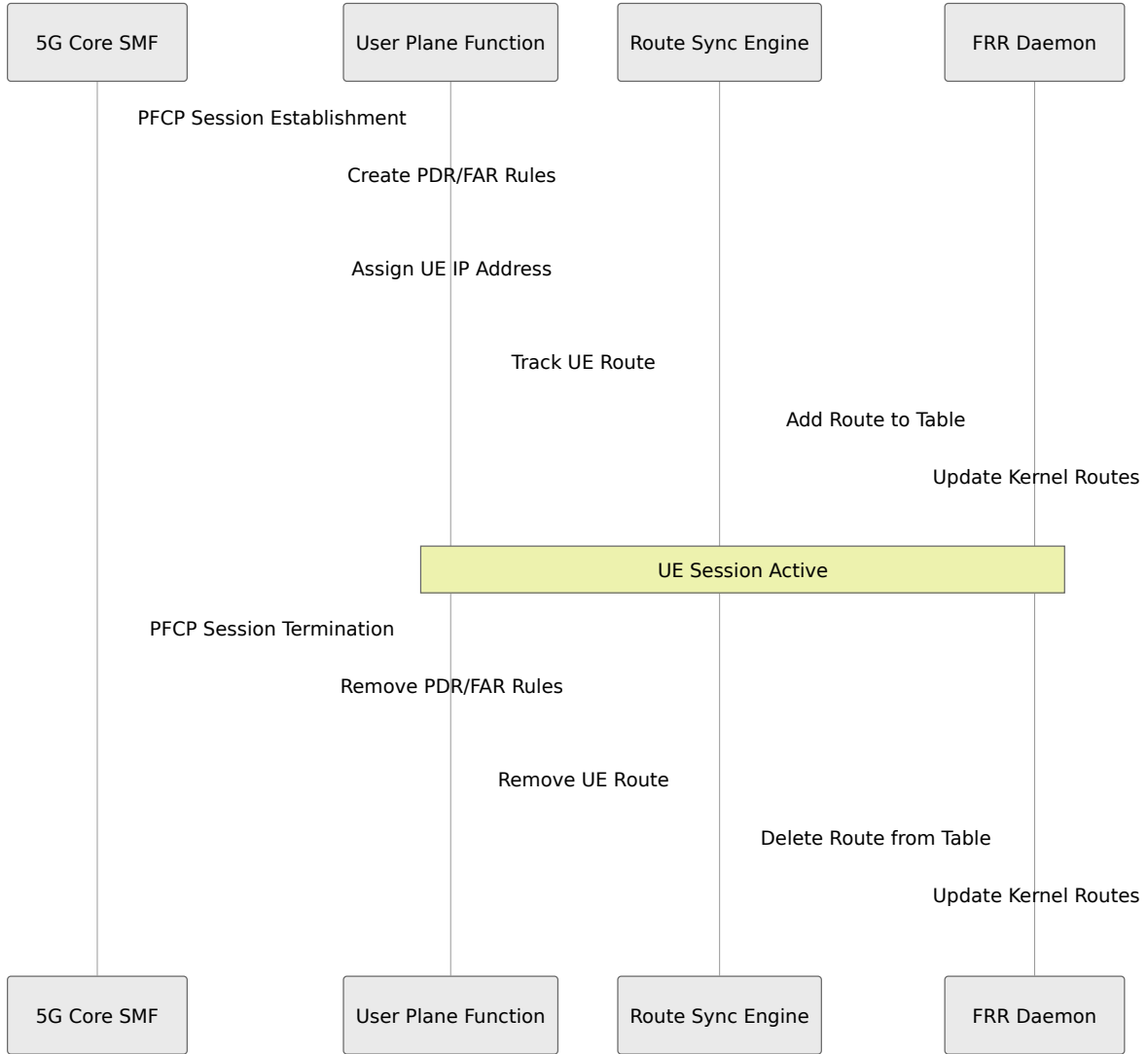
**□□□ FRR□**

**FRR** (Free Range Router) は、Linux 上で動作する BGP、OSPF、RIP の  
 多プロトコルルーティングソフトウェアです。



# 5G Core Network

## Network Architecture



## Flowchart

UPF 接收 UE IP 并配置路由规则

1. 接收 **UE** 通过 PFCP 接收 UE IP
2. 配置路由规则
3. 通过 **FRR** API 配置 FRR 路由
4. 配置路由规则/配置路由规则

# FRR

## 

FRR 用 **Ansible** 用 設定テンプレート Ansible 用 FRR 用 **Jinja2** 用 Ansible 用 UPF 用

用 FRR Jinja2 用

```
frr version 7.2.1
frr defaults traditional
hostname pgw02
log syslog informational
service integrated-vtysh-config
!
ip route {{ hostvars[inventory_hostname]['ansible_default_ipv4']
['gateway'] }}/32 {{ ansible_default_ipv4['interface'] }}
!
interface {{ ansible_default_ipv4['interface'] }}
 ip address ospf router-id {{hostvars[inventory_hostname]
['ansible_host']}}
 ip ospf authentication null
!
router ospf
 ospf router-id {{hostvars[inventory_hostname]['ansible_host']}}
 redistribute static
 network {{ hostvars[inventory_hostname]['ansible_default_ipv4']
['network'] }}/{{ mask_cidr }} area 0
 area 0 authentication message-digest
!
line vty
!
end
```

## 

1. 用 Ansible 用 FRR Jinja2 用  
roles/frr/templates/frr.conf.j2
2. 用 UPF 用 Ansible 用
3. 用 Ansible 用 FRR 用 UPF 用



4. `ansible` `ansible` `IP` `ID` `UPF` `FRR`

## Jinja2

- **OSPF** `ID`
- **BGP** `ASN`
- `redistribute static` `UE`
- `redistribute static`
- `OSPF/BGP`

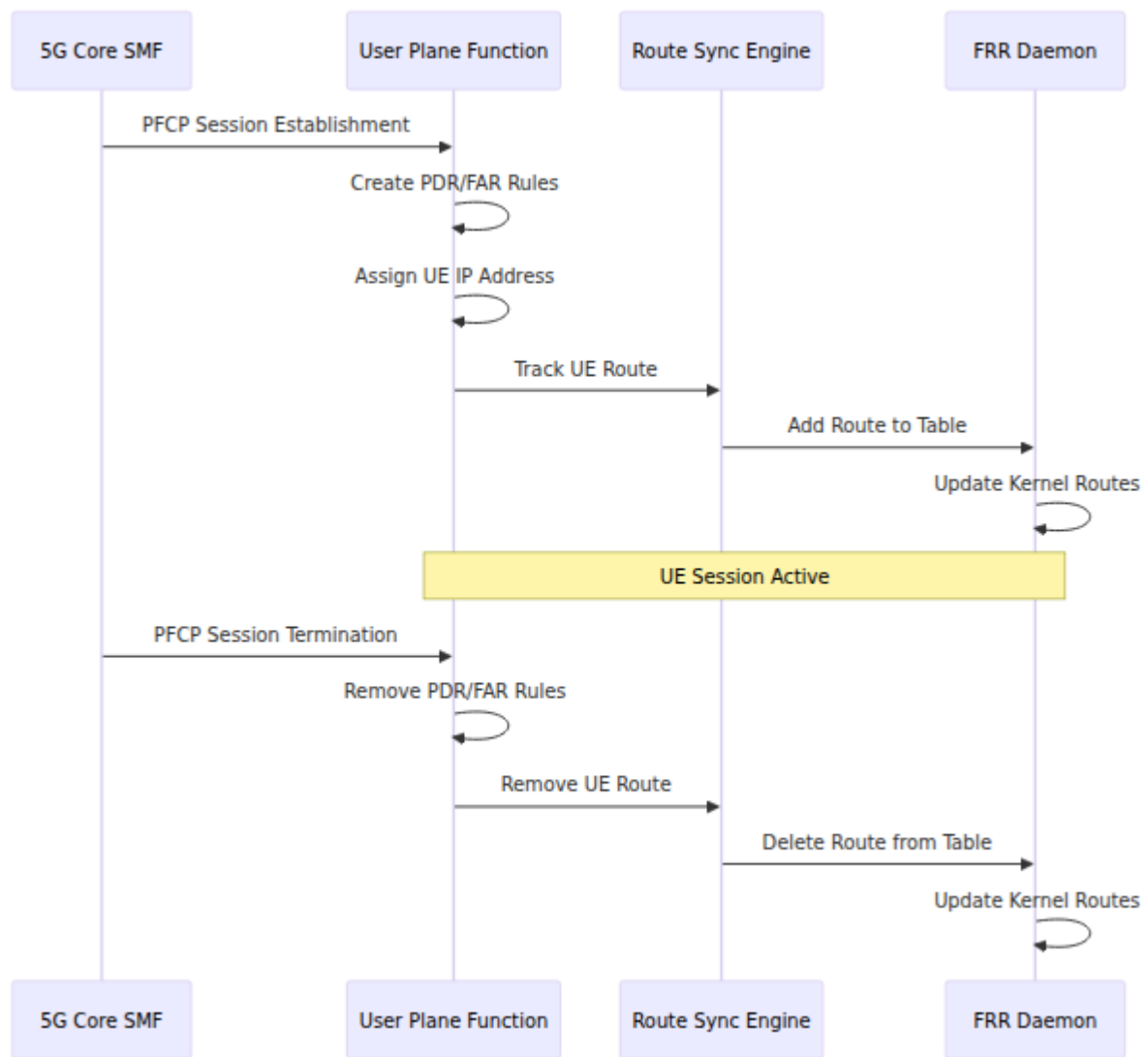
**UPF** `FRR` `UPF` `PFCP` `FRR vtysh` `UE IP` `IPv4` `/32` `IPv6` `/128`

1. `FRR` `UPF` `vtysh`
2. `redistribute static` `FRR`
3. `FRR` `OSPF` `BGP`
4. `UE` `UPF`

`UPF` `FRR` `vtysh` `FRR` `OSPF/BGP` `redistribute static` `redistribute kernel`

- `Ansible` `FRR Jinja2` `UPF`
- **Ansible** `Jinja2` `OSPF` `BGP`
- **UPF** `UPF` `PFCP` `UE IP /32`
- `UPF` `FRR` `UE`
- `Ansible` `UPF`

□□□□



□□□□□

## Web UI □□

UPF □□□□□□□□ □□ □□□□□□

- □□□□□□□□□□□□□□□□
- □□□□□□□□□□ UE IP □□□□
- □□□□□□□□□□□□□□□□□□
- □□□□□□□□□□□□□□□□□□ UE IP □□□□□□□□
- **OSPF** □□□□□□□□□□□□ OSPF □□□□

- **BGP** 通過 BGP 實現跨網段互連
- **OSPF** 通過 OSPF LSA 實現跨網段 UE 互連

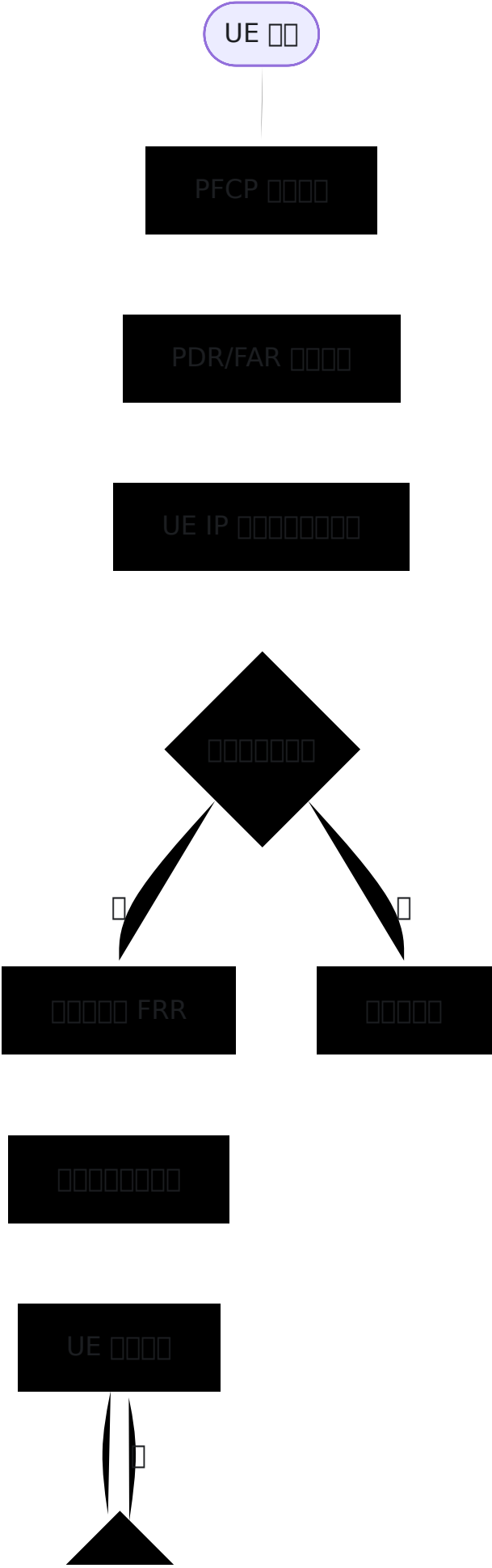
通過 UE 實現跨網段互連

實現互連

通過 Web UI 實現互連

1. 通過 UPF 實現 UE 互連
2. 通過 FRR 實現互連
3. 實現互連
4. 實現互連
5. 實現互連







□



□□

- □□□□□□□□□□□□□□□□
- □□□□□□□□□□□□ UE □□□□□□□
- □□□□□□□□□□□□ UE □□
- □□□□□□□□□□□□□□□□
- □□□□□□ Web UI □□□□□□□□

□□□□

## API □□

UPF □□□□□□□□□□

- GET /api/v1/routes - □□□□□□□ UE □□□□□□□□
- POST /api/v1/routes/sync - □□□□□□ FRR □□□□□□□□
- GET /api/v1/route\_stats - □□□□□□□□□□
- GET /api/v1/routing/sessions - □□□□□□□□□□ OSPF □□□ BGP □□□□
- GET /api/v1/ospf/database/external - □□ OSPF AS-External LSA □□□□□□□□ □□□□□

□□□ API □□ - □□□□ □□□□□□□□□□□□□□□□

## 配置

IPv4 地址 100.64.18.5 配置为静态路由

- 配置静态路由
- 配置路由
- 配置
- 配置

## IPv6 配置

配置 IPv4 和 IPv6 UE 配置

| 配置   | 配置   | 配置              |
|------|------|-----------------|
| IPv4 | /32  | 100.64.18.5/32  |
| IPv6 | /128 | 2001:db8::1/128 |

配置 IPv6 静态路由 FRR 配置 OSPFv3 和 BGP IPv6 配置

```
router ospf6
 redistribute static
```

配置 BGP

```
router bgp <asn>
 address-family ipv6 unicast
 redistribute static
```

# FRR

## OSPF LSA

OSPF FRR OSPF UE OSPF LSA 5

FRR OSPF LSA UE 100.64.18.5/32 E2 2

- LSA (10.98.0.20) UPF
- LSA (192.168.1.1) OSPF
- LSA UE 100.64.18.5 E2 2 OSPF

- UPF UE IP
- FRR
- FRR OSPF
- OSPF



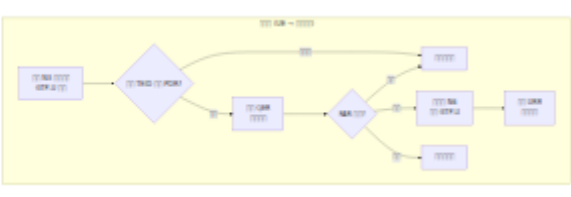
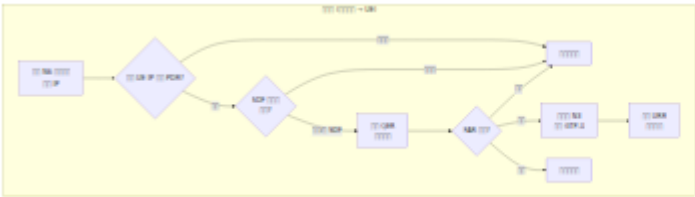
11

- 11

|  |  |  |  |
|--|--|--|--|
|  |  |  |  |
|--|--|--|--|

| 項目名                               | 単位                   | 説明           | 備考                |
|-----------------------------------|----------------------|--------------|-------------------|
| <b>PDR</b> (Policy Data Report)   | Policy Data Report   | TEID と UE IP | SMF から PCF へ送信/受信 |
| <b>FAR</b> (Flow Admission Rule)  | Flow Admission Rule  | FAR ID       | SMF から PCF へ送信/受信 |
| <b>QER</b> (QoS Enforcement Rule) | QoS Enforcement Rule | QER ID       | SMF から PCF へ送信/受信 |
| <b>URR</b> (Usage Reporting Rule) | Usage Reporting Rule | URR ID       | SMF から PCF へ送信/受信 |

□□□□□□

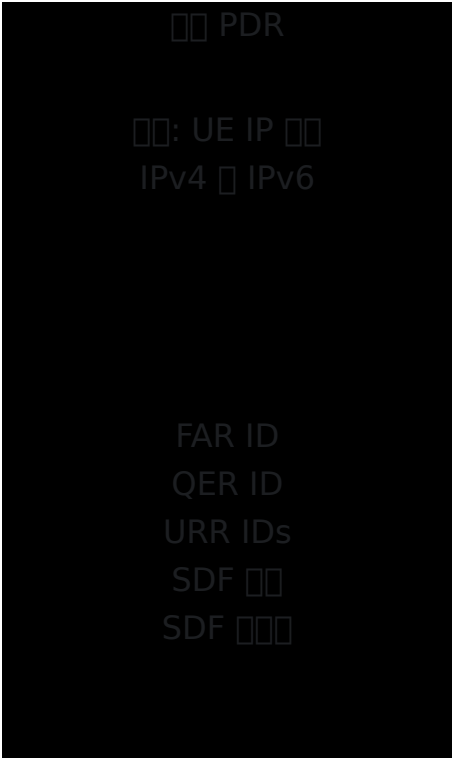


# □□□□□□□ (PDR)

□□

PDR □□□□□□□□□□□□□□ UPF □□□□□□□□□□□□□□

# PDR



# PDR

PDR RAN N3

TEID (TEID)

- 32
- SMF gNB
- UE

:

- **FAR ID:**
- **QER ID:** QoS
- **URR IDs:**
- : GTP-U

:

1. GTP-U TEID
2. `uplink_pdr_map` eBPF
3. FAR ID QER ID URR IDs
- 4.

:

TEID: 5678  
FAR ID: 2  
QER ID: 1  
 : False  
SDF : No SDF

## 📄 PDR

📄 PDR 📄📄📄📄📄 N6 📄📄📄📄📄📄

📄📄📄: UE IP 📄

- IPv4 📄 (32 📄) 📄 IPv6 📄 (128 📄)
- 📄 PDU 📄📄📄📄 SMF 📄
- 📄 UE 📄

📄📄📄:

- **FAR ID:** 📄📄📄📄📄
- **QER ID:** QoS 📄📄📄📄📄📄
- **URR IDs:** 📄📄📄📄📄📄📄
- **SDF 📄:** 📄📄📄📄📄
  - **No SDF:** 📄📄📄📄📄📄
  - **SDF Only:** 📄📄 SDF 📄📄📄

- **SDF + Default**: SDF ☐ FAR ☐

- **SDF** ☐: ☐ IP ☐

☐:

1. ☐ IP
2. ☐ **downlink\_pdr\_map** (IPv4) ☐ **downlink\_pdr\_map\_ip6** (IPv6) ☐
3. ☐ SDF ☐
4. ☐ FAR ID ☐ QER ID ☐ URR IDs
5. ☐

☐:

```
UE IP: 10.45.0.1
FAR ID: 1
QER ID: 1
☐: False
SDF ☐: No SDF
```

# SDF 规则 (SDF Rules)

SDF 规则用于指定流量过滤规则

规则:

- 允许 YouTube 流量
- 为 VoIP 流量提供高 QoS
- 阻止非法流量

规则:

- 协议: TCP/UDP/ICMP
- 端口: 允许 HTTPS 443/SIP 5060
- IP 地址: 允许
- 规则: 3GPP 规则

规则 SDF 规则:

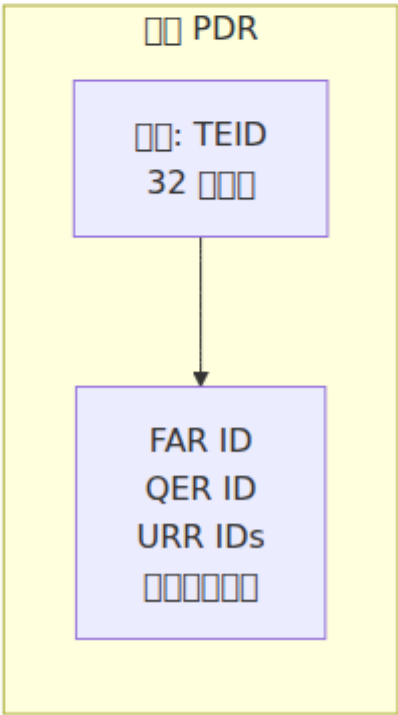
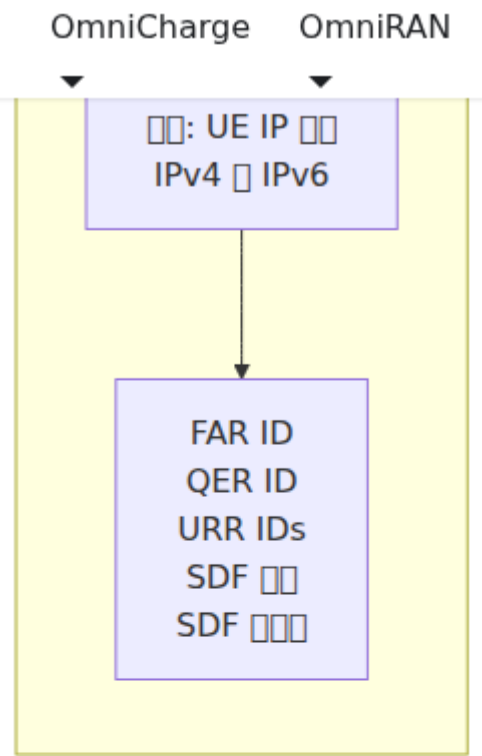
PDR ID: 10  
UE IP: 10.45.0.1  
SDF 规则: SDF Only  
SDF 规则:  
- 协议: UDP, 端口: 5060-5061 → FAR ID 5 (VoIP FAR)  
- 协议: TCP, 端口: 443 → FAR ID 1 (Web FAR)

# 流量过滤规则 (FAR)

规则

FAR 规则 PDR 规则 GTP-U 规则

FAR



FAR



| 動作               | 1 | 2  | 動作           |
|------------------|---|----|--------------|
| <b>FORWARD</b>   | 1 | 2  | パケットを送信      |
| <b>BUFFER</b>    | 2 | 4  | パケットをバッファに格納 |
| <b>DROP</b>      | 0 | 1  | パケットを丢弃      |
| <b>NOTIFY</b>    | 3 | 8  | パケットを送信      |
| <b>DUPLICATE</b> | 4 | 16 | パケットを複製      |

動作:

- 動作: 2 (FORWARD) - パケットを送信
- 動作: 6 (FORWARD + BUFFER) - パケットを送信
- 動作: 4 (BUFFER) - パケットをバッファに格納
- 動作: 1 (DROP) - パケットを丢弃

動作

BUFFER 動作 2パケットを送信する前に UPF 動作 4パケットを送信する前に UE 動作 4パケットを送信する前に

動作

動作: 動作 4パケットを送信する前に IDLE 動作 4パケットを送信する前に gNB 動作 4パケットを送信する前に UE 動作 4パケットを送信する前に

- 動作
- 動作 SMF 動作 (DLDR)
- 動作 SMF 動作 AMF 動作 UE 動作
- 動作 SMF 動作 FAR 動作 FORWARD 動作
- 動作 UPF 動作 4パケットを送信する前に UE

動作: 動作 gNB 動作 gNB 動作 4パケットを送信する前に UPF 動作 4パケットを送信する前に

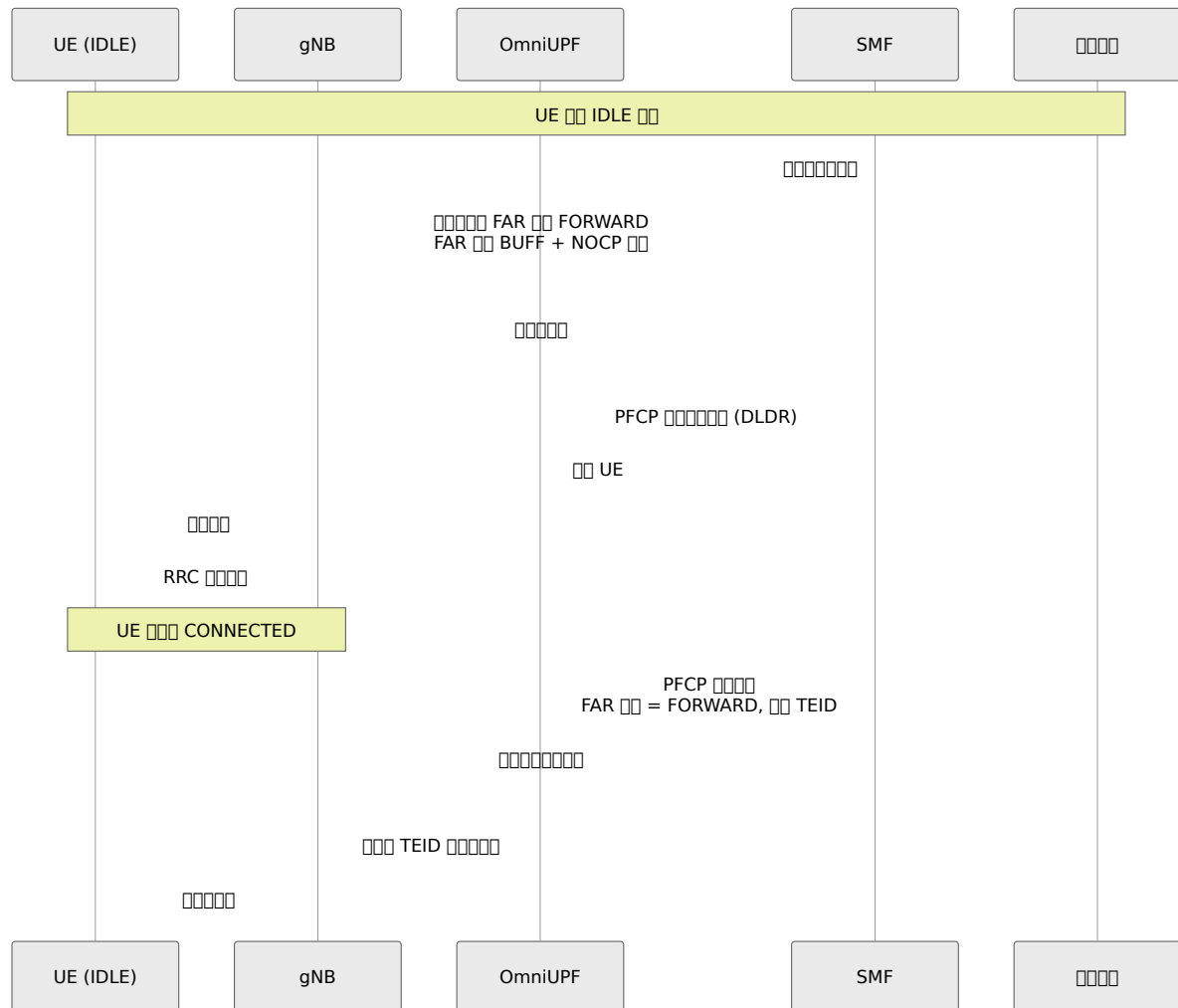
- 動作 gNB 動作
- 動作 SMF 動作 FAR 動作 4パケットを送信する前に BUFFER

3.

#### 4. UE 与 gNB

## 5. SMF → FAR → TEID → FORWARD →

## 6. UPF □□□□□□□□ gNB



□ □ □ □ □ □ □

□□□□□□:

- 最大接続数: 100,000接続
- 接続数: 10000
- **TTL (秒数):** 60 秒
- 最大 TTL 差: 1000

**□□ FAR □□:**

- **FAR** 10,000

- 原因: 达到 FAR 限制

原因:

- 达到 FAR 限制
- 原因: `reason="global_limit"` 或 `reason="far_limit"`
- 达到 TTL 限制

DLDR (DLDR)

UPF 在 IDLE 态 UE 通过 SMF 和 PCF 交互

DLDR 原因:

- 原因: DLDR (DLDR)
- **FAR ID:** 达到 FAR
- 原因: 达到 QFI 限制

SMF 和 DLDR 原因:

1. 通过 AMF → gNB 和 UE
2. 通过 UE 和 RRC 交互
3. 通过 PCF 和 FAR
4. FAR 原因: `BUFF+NOCP` 或 `FORW`
5. UPF 原因

DLDR 原因:

- `upf_dldr_sent_total`: 发送 DLDR 次数
- `upf_dldr_send_errors`: 发送 DLDR 错误
- `upf_buffer_notify_to_flush_duration_seconds`: DLDR 缓冲时间

原因 原因 原因

原因

原因 BUFF 原因:

- FAR 原因 `|= 0x04` 原因 2

- `forw: 2 (FORW) → 00: 6 (FORW+BUFF)`
- `00000000`

`00000000` BUFF `forw`:

- FAR `00 = 0x04` BUFF
- `0000000000`
- `00` IDLE UE `00000000`

`00000000` BUFF `00`:

- FAR `00 &= ~0x04` `0000` 2
- `00: 00: 6 (FORW+BUFF) → 00: 2 (FORW)`
- `000000000000000000000000`

`0000`:

- `00 00` FAR `0000000000000000`
- `0000000000` TEID/`00000000`
- `0000000000`
- FAR `0000` FORW `00`

`0000`:

- `0000000000000000`
- `0000000000000000`
- `0000` `reason="cleared"`

`00000000`

`000000` Web UI: `000 00 0000`

- `00000000`
- `000000`
- `0000000000` FAR `00`
- `00` FAR `00000000`
- `0000000000`
- `00/0000` FAR `0000`

- 00000000

0000:

- 000 > 10 0: 00000000
- 000 > 30 0: 00000000000000
- 0000000: 0000000000000000

**Prometheus** 00:

- upf\_buffer\_packets\_current: 00000000
- upf\_buffer\_bytes\_current: 00000000
- upf\_buffer\_fars\_active: 00000000 FAR
- upf\_buffer\_packets\_dropped{reason}: 00000000

000 0000 000000000000

000000

**00 1: IDLE UE** 0000

0000:

- UE 00 IDLE 000000 gNB 000
- FAR 00: 0x04 (0 BUFF)

0000:

1. DN 00000000
2. UPF 00 PDR000 FAR
3. FAR 00 BUFF 00 → 000000
4. UPF 0 SMF 00 DLDR
5. SMF 00 UE
6. UE 000 gNB
7. SMF 00 FAR: 00 = 0x02 (FORW)
8. UPF 000 TEID 00000000

**00 2:** 0000

□□□□:

- UE □□□ gNB-1 (TEID 1234)
- FAR □□: 0x02 (FORW)

□□□□:

1. SMF □□ FAR: □□ = 0x06 (FORW+BUFF)
2. □□□□□□ gNB-1 □□□
3. UE □□□ gNB-2
4. SMF □□ FAR: TEID = 5678, □□ = 0x02 (FORW)
5. UPF □□□□□□□□□□□□ gNB-2□□□□ TEID
6. □□□□□□□□□□□□

### □□ 3: □□□□

□□□□:

- UE □□□□□□□□

□□□□:

1. SMF □□ FAR: □□ = 0x04 (□ BUFF)
2. □□□□□□□□□□□□□□□□
3. □□□□□□□□
4. SMF □□ FAR: □□ = 0x02 (FORW)□□□□
5. UPF □□□□□□□□□□□□□□

## Network

Network GTP-U

**FAR** (N3 → N6):

- : False
- : GTP-U IP

**FAR** (N6 → N3):

- : True
- IP: gNB IP 200.198.5.10
- TEID: UE ID
- : GTP-U gNB

## Web UI FAR

ID FAR

□□:

1. どの 何 → FARs 何
2. 何何何何何何 FAR ID
3. 何 "何" 何 FAR 何何

□□□□□:

- FAR ID
- 000000 + 000000
- 00000000/0000
- 00000000
- 00 IP 000000000000
- TEID
- 000000

## QoS □□□□ (QER)

11

QER



**QER** □□

QER 00

QFI  
QoS 0000

00000  
00/00

00000  
00/00

QER ID  
00000

00 MBR  
00000

00 MBR  
00000

00 GBR  
00000

00 GBR  
00000

## QoS 設定

### QFI (QoS 識別子):

- 6 種類の 5G QoS
- 1-9 の QFI 9 =
- 5GC

設定:

- (0):
- (1):

### MBR (MBR):

- 
- kbps
- **MBR = 0:**
- MBR

### GBR (GBR):

- 
- kbps
- **GBR = 0:**
- **GBR > 0:**

## QoS

(GBR = 0):

```
QER ID: 1
QFI: 9
MBR: 100000 kbps (100 Mbps)
MBR: 100000 kbps (100 Mbps)
GBR: 0 kbps
GBR: 0 kbps
```

□□□ (GBR > 0):

QER ID: 2

QFI: 1

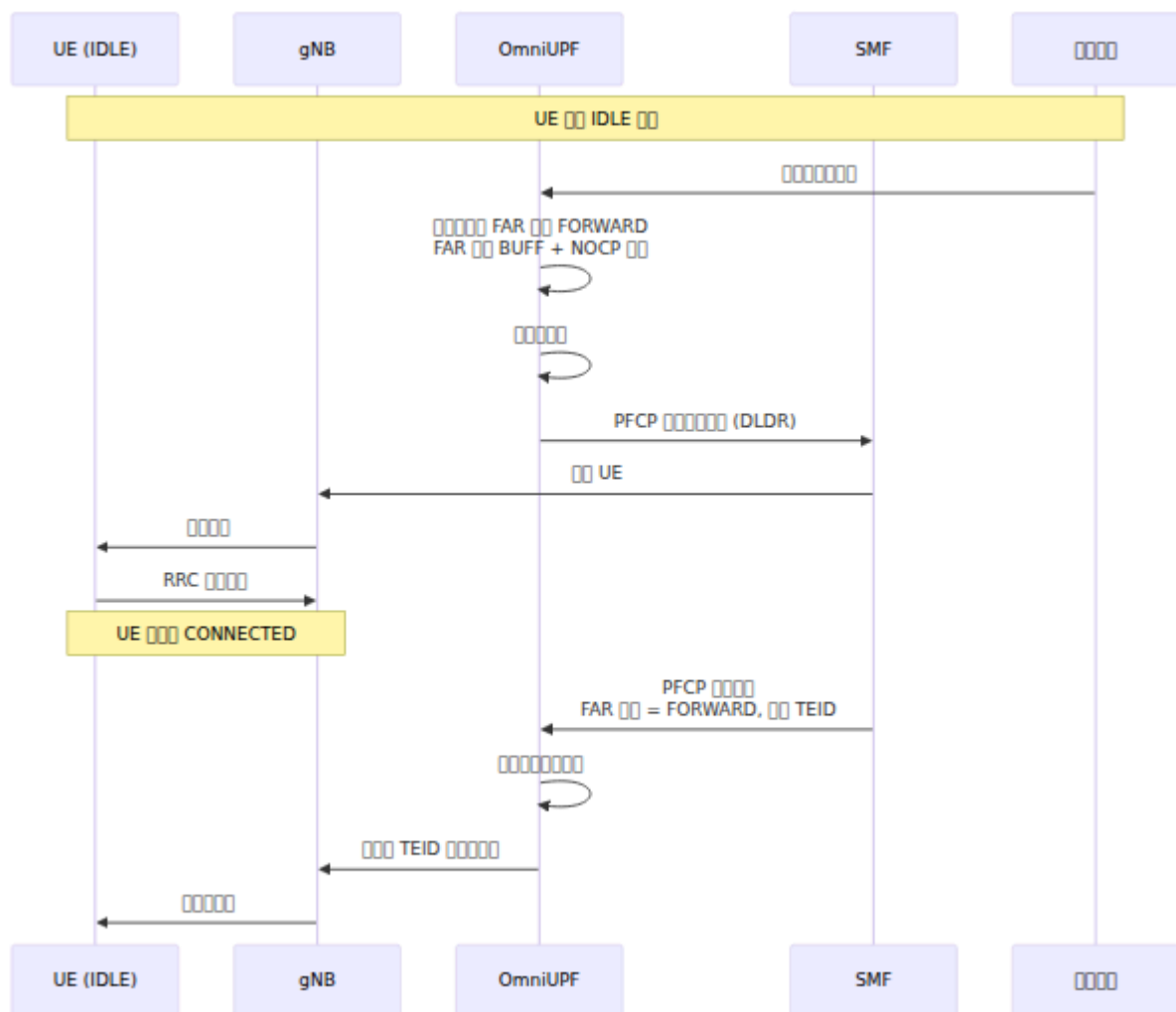
□□ MBR: 10000 kbps (10 Mbps)

□□ MBR: 10000 kbps (10 Mbps)

□□ GBR: 5000 kbps (5 Mbps)

□□ GBR: 5000 kbps (5 Mbps)

## QoS



## MBR

OmniUPF uses eBPF to implement MBR. XDP

:

UPF

1. : CLOSED
2. **MBR** : MBR = 0
3. :

```
tx_time = (packet_size_bytes × 8) × (1,000,000,000 ns/sec) /
MBR_kbps
```

4. `tx_time`: `tx_time` 5ms `tx_time` `tx_time`

5. `tx_time`: `tx_time` `tx_time` `tx_time`

`tx_time`:

`tx_time`:

- MBR = 100,000 kbps (100 Mbps)
- `tx_time` = 1500 `tx_time`
- `tx_time` = 5,000,000 ns (5 ms)

`tx_time` 1: `tx_time` 100 Mbps `tx_time`

```
tx_time = (1500 tx_time × 8 tx_time) × (1,000,000,000 ns/sec) /
100,000,000 bps
 = 12,000,000,000 / 100,000,000
 = 120 ns
```

`tx_time` 2: `tx_time`

```
current_time = 1000000000 ns
window_start = 999990000 ns
if (window_start + tx_time > current_time):
 tx_time
```

`tx_time` 3: `tx_time`

```
window_start = window_start + 120 ns
tx_time
```

`tx_time`

**5ms** `tx_time`:

- `tx_time` 5 `tx_time`
- `tx_time` 5 `tx_time`
- `tx_time` `tx_time`

`tx_time`:

- 5ms 時間間隔
- 送信側 MBR 時間間隔
- 受信側時間間隔

送信側時間間隔:

- 送信 MBR 時間 `qer->ul_start` 時間
- 送信 MBR 時間 `qer->dl_start` 時間
- 時間間隔時間間隔

## MBR 時間間隔

時間間隔時間間隔 MBR 時間:

- 時間間隔: GTP-U/UDP/IP 時間間隔時間間隔 ~50-60 時間
- 時間間隔時間間隔: 時間間隔時間間隔 = 時間間隔時間間隔
- 時間間隔時間間隔: 時間間隔時間間隔時間間隔時間間隔
- 時間間隔時間間隔: 5ms 時間間隔時間間隔時間間隔 MBR

時間:

送信 MBR: 100 Mbps  
時間間隔時間間隔: ~95-98 Mbps時間間隔 GTP-U/UDP/IP 時間間隔

時間間隔時間間隔:

1. 送信 URR 時間間隔時間間隔: `upf_urr_*_volume_bytes`
2. 時間間隔時間間隔: `(volume_delta_bytes × 8) / time_delta_seconds / 1000 = kbps`
3. 送信 QER 時間間隔 MBR 時間間隔

## GBR (時間間隔)

時間: OmniUPF 時間間隔時間間隔 GBR 時間間隔GBR 時間間隔 QER 時間間隔時間間隔時間間隔時間間隔

GBR 時間:

- GBR 時間間隔 PFCP 時間間隔 SMF 時間間隔
- GBR 時間間隔 QER 時間間隔時間間隔 API 時間間隔

- 5G Core Network GBR QoS
- GBR QoS

QoS:

- GBR QoS
- 5G Core Network eBPF QoS

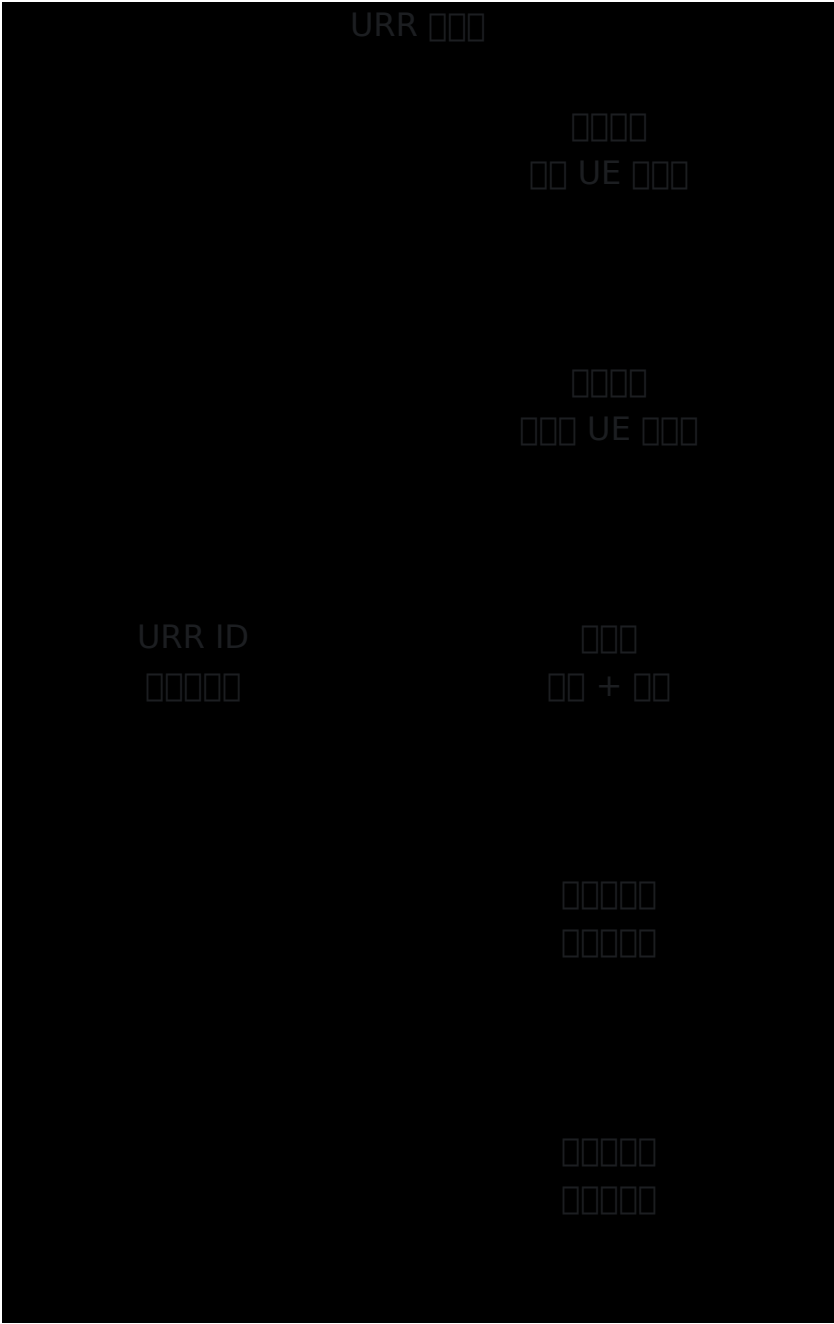
## 5G Core Network (URR)

QoS

URR QoS SMF QoS



# URR 表



表例

表例:

- UE 192.168.1.1
- GTP-U 192.168.1.2
- IP 192.168.1.3

□□□□:

- □□□□□□□□ UE □□□
- □ GTP-U □□□□□
- □□ IP □□□□□□□

□□□:

- □□□□□□□□□□
- □□□□□□□

□□□□□□□□

URR □□□□□□□□□□□□□□

□□□□:

- □□□□□□□□□□□□□□
- □□: □ 1 GB □□□□□□□

□□□□:

- □□□□□□□□
- □□: □ 5 □□□□□□□

□□□□:

- □□□□□□□□□
- □ QoS □□□□□□
- □□□□□□□

□□□□□□□□

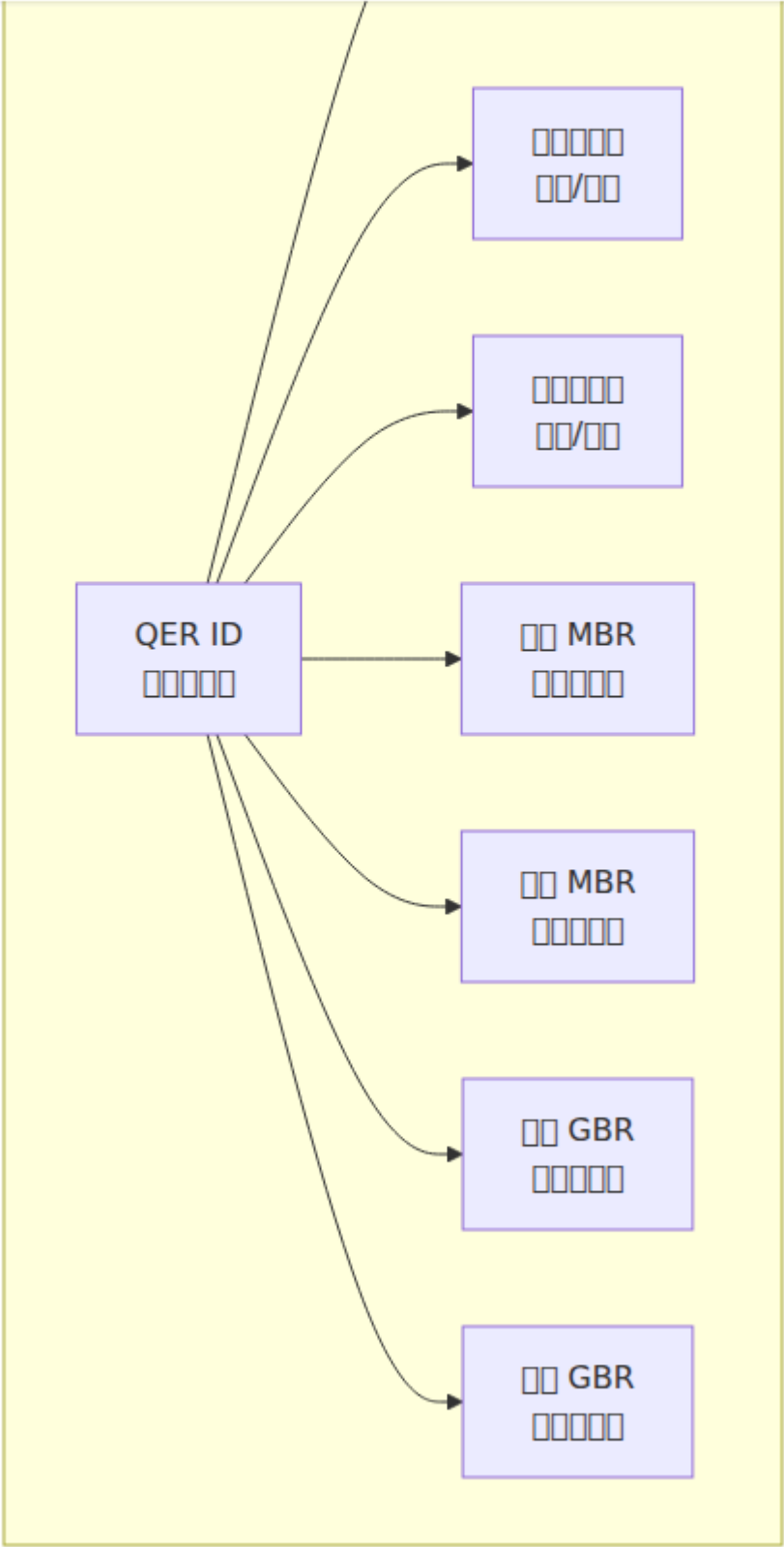
Web UI □□□□□□□□□□□□□□□□

| 범위                         | 단위         |
|----------------------------|------------|
| 0 - 1023                   | B (바이트)    |
| 1024 - 1048575             | KB (킬바이트)  |
| 1048576 - 1073741823       | MB (메가바이트) |
| 1073741824 - 1099511627775 | GB (기가바이트) |
| 1099511627776+             | TB (테라바이트) |

총용량:

URR ID: 0  
파일용량: 12.3 KB  
메타데이터: 9.0 KB  
인덱스: 21.3 KB

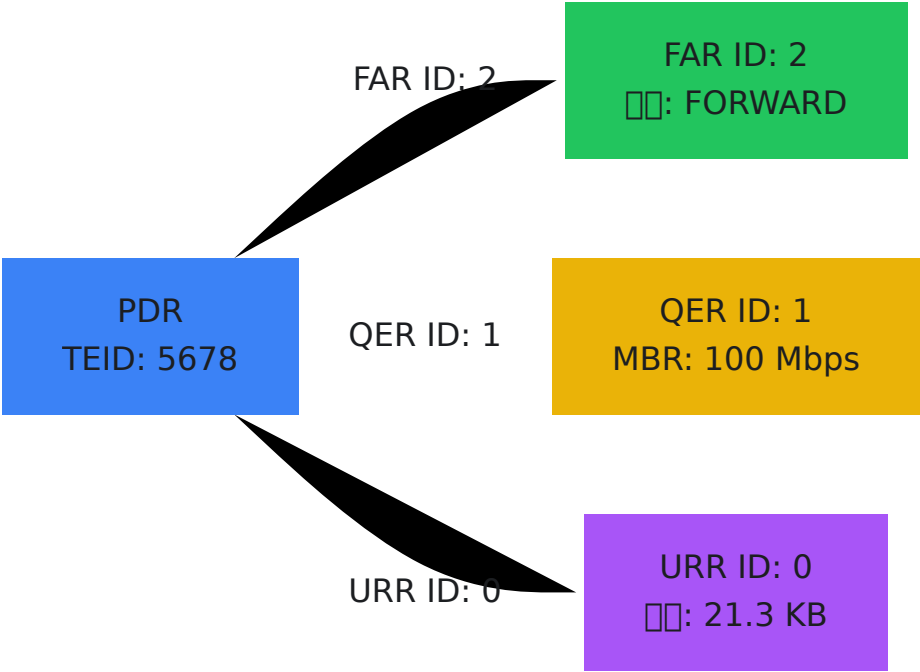
**URR** □□□□



□□□□

**PDR → FAR → QER → URR** □

□□ PDR □□□□ FAR□FAR □□□□□□ QER □□□□□□ URR□



□□□□□□

□□ **PDR:**

TEID: 5678  
FAR ID: 2  
QER ID: 1  
URR IDs: [0]  
□□□□□□: False

□□ **PDR:**

UE IP: 10.45.0.1  
FAR ID: 1  
QER ID: 1  
URR IDs: [0]  
SDF 00: No SDF

#### **FAR ID 1 (00):**

00: 2 (FORWARD)  
000000: True  
00 IP: 200.198.5.10  
TEID: 5678

#### **FAR ID 2 (00):**

00: 2 (FORWARD)  
000000: False

#### **QER ID 1:**

QFI: 9  
00 MBR: 100000 kbps  
00 MBR: 100000 kbps  
00 GBR: 0 kbps  
00 GBR: 0 kbps

#### **URR ID 0:**

0000: 12.3 KB  
0000: 9.0 KB  
000: 21.3 KB

## □□□□

### □□□□□□□

□□□□□:

1. □□□□□
2. □□ IP □ TEID □□ UE
3. □□ "□□" □□□□□ (PDR, FAR, QER, URR)

□□□□□:

1. □□□□□
2. □ PDR □□□□□ TEID□□□□□ UE IP□□□□□□
3. □□ FAR ID□QER ID□URR IDs
4. □□□ FAR/QER/URR □□□□□□□□□

### □□/□□□□

□□: □□□□□□□□□□□□□□

□□:

1. □□□ □□ → FARs
2. □□□□□□□□ FAR ID
3. □□ "□□"
4. □□□□□□□□□□ "□□□□"
5. □□ FAR □□□ 2 □□□□□□□□□ 4□

□□□□□□□□□□:

1. □□□□□
2. □□□□□□□ FAR
3. □□□□□□□□ "□□□□"



## QoS

QoS:

1. QoS → QoS
2. UE QoS ID
3. MBR MBR
4. URR

QoS:

$$\text{QoS (kbps)} = (\text{QoS} \times 8) / (\text{QoS} \times 1000)$$

QoS MBR

QoS

QoS:

1. QoS → URRs
2. QoS
3. QoS
4. QoS

QoS:

- QoS
- QoS
- QoS

QoS

QoS

QoS:

1. TEID UE IP PDR
2. FAR ID
3. SDF

**□□ FAR:**

1. FAR FORWARD DROP BUFFER
- 2.
3. IP TEID

**QR:**

1. 0000000000000000
2. MBR 000000

|  |  |  |  |  |  |
|--|--|--|--|--|--|
|  |  |  |  |  |  |
|--|--|--|--|--|--|

☐☐ **QER** ☐☐☐☐:

1. 1000 00 → QERs
2. 00 MBR 000000000000
3. 00 URR 0000000000000000

**FAR**:

1. 0000 00 → FARS
2. 00000000 FORWARD0000 DROP
3. 000000000000 BUFFER 000

□ □ □ □

□□□□□□□□ :

1. 00000000
2. 000000000000
3. 00 > 30 00000000
4. 0000000000

## 5. 1 FAR 10000

10000:

1. 100,000
2. 10,000 FAR 10,000
3. 10,000
4. 10,000

## URR 1000

1000000:

1. 100 PDR 1000 URR ID
2. 1000000 PDR
3. 100 FAR 1000000000000
4. 100 URR ID 10000 URR 1000

1000000 **SMF**:

1. 100 PFCP 1000000
2. 100 URR 10000000/1000000
3. 100 PFCP 1000000000

## 100000

- **UPF 10000** - OmniUPF 10000000
- **Web UI 10000** - 1000000000000
- **10000** - 100000000
- **1000000** - 100000000

# OmniUPF 部署架构图

## 部署

1. 部署
2. 部署
3. 部署
4. 部署
5. PCF 部署
6. 部署
7. XDP 及 eBPF 部署
8. 部署
9. 部署
10. NIC 部署
11. 部署
12. 部署

## 部署

部署 OmniUPF 部署

## 部署

部署

```
1. 检查 OmniUPF 是否安装
systemctl status omniupf

2. 检查 PFCP 是否安装
curl http://localhost:8080/api/v1/upf_pipeline

3. 检查 eBPF 是否安装
ls /sys/fs/bpf/

4. 检查 XDP 是否安装
ip link show | grep -i xdp

5. 检查日志
dmesg | tail -50
journalctl -u omniupf -n 50
```

检查

## OmniUPF REST API

检查 **UPF** 是否安装

```
curl http://localhost:8080/api/v1/upf_status
```

检查 **PFCP** 是否安装

```
curl http://localhost:8080/api/v1/upf_pipeline
```

检查 eBPF 是否安装

```
curl http://localhost:8080/api/v1/sessions | jq 'length'
```

检查 **eBPF** 是否安装

```
curl http://localhost:8080/api/v1/map_info
```

□□□□□□□□

```
curl http://localhost:8080/api/v1/packet_stats
```

□□ **XDP** □□□

```
curl http://localhost:8080/api/v1/xdp_stats
```

---

## eBPF □□□□

□□□□ **eBPF** □□□

```
ls -lh /sys/fs/bpf/
bpftool map list
```

□□□□□□□□□□

```
bpftool map show
bpftool map dump name pdr_map_downlin
```

□□□□□□□□□□

```
bpftool map dump name far_map | grep -c "key:"
```

---

## **XDP** □□□□

□□ **XDP** □□□□□□□□

```
ip link show eth0 | grep xdp
```

看看 **XDP** 看看

```
bpftool net list
```

看看 **XDP** 看看看看

```
bpftool prog show
```

看看 **XDP** 看看

```
bpftool prog dump xlated name xdp_upf_func
```

---

看看

看看 **N4** 看看 **PFCP** 看看看看

```
PFCP 看看 XDP 看看tcpdump 看看
tcpdump -i eth0 -n udp port 8805 -w /tmp/pfcp_traffic.pcap
```

看看 **N3** 看看 **GTP-U** 看看看看

```
0000UPF 0000000 tcpdump 0000 XDP 00000000
XDP 0000000000000000 GTP-U

0000000000
1. gNB 0 UPF 000000 TAP
2. 00000000/SPAN 00 N3 00
3. 00000000000000 VM

0000/000000000000 UPF00
tcpdump -i <mirror_interface> -n udp port 2152 -w
/tmp/n3_capture.pcap

000000 API 0000000000
curl http://localhost:8080/api/v1/packet_stats
curl http://localhost:8080/api/v1/n3n6_stats
```

0000000000

```
watch -n 1 'ip -s link show eth0'
```

00000000

```
ip route show
ip route get 10.45.0.100 # 00 UE IP 000
```

00 **ARP** 00

```
ip neigh show
```

000000

0000“**eBPF** 00000000”

000



```
ERROR[0000] failed to load eBPF objects: mount bpf filesystem at /sys/fs/bpf
```

安装eBPF 依赖包

安装

```
安装 eBPF 依赖包
sudo mount bpffs /sys/fs/bpf -t bpf

安装 bpffs 到 /etc/fstab
echo "bpffs /sys/fs/bpf bpf defaults 0 0" | sudo tee -a /etc/fstab

挂载
mount | grep bpf
```

安装内核头文件

安装

```
ERROR[0000] kernel version 5.4.0 is too old, minimum required is 5.15.0
```

安装Linux 内核头文件

安装

```
確認
uname -r

libbpfUbuntu/Debian
sudo apt update
sudo apt install linux-generic-hwe-22.04
sudo reboot

確認
uname -r # 5.15.0
```

---

## libbpf 確認

確認

```
error while loading shared libraries: libbpf.so.0: cannot open
shared object file
```

libbpf 確認

確認

```
libbpfUbuntu/Debian
sudo apt update
sudo apt install libbpf-dev

確認
ldconfig -p | grep libbpf
```

---

確認

確認

確認

```
ERR0[0000] unable to read config file: unmarshal errors
```

配置文件中 YAML 格式

配置

```
检查 YAML 格式
cat config.yml | python3 -c "import yaml, sys;
yaml.safe_load(sys.stdin)"
```

```
配置项
- 配置项名称
- 配置项值
- 配置项类型
- 配置项默认值
```

```
检查 YAML 格式
cat > config.yml <<EOF
interface_name: [eth0]
xdp_attach_mode: generic
api_address: :8080
pfcp_address: :8805
EOF
```

配置文件中配置项

配置

```
ERR0[0000] interface eth0 not found
```

配置文件中配置项

配置

```
確認
ip link show

確認
ip addr show eth0

設定ファイル config.yml
interface_name: [ens1f0] # 確認

確認
ls /sys/class/net/
```

---

確認

確認

```
ERR0[0000] failed to start API server: address already in use
```

確認 8080 8805 9090 確認

確認

```
確認
sudo lsof -i :8080
sudo netstat -tulpn | grep :8080

確認
sudo kill <PID>

設定 OmniUPF
api_address: :8081
pfcf_address: :8806
metrics_address: :9091
```

---

# 配置 PFCP 节点 ID

错误

ERR0[0000] invalid pfcf\_node\_id: must be valid IPv4 address

配置 PFCP 节点 ID 时必须为 IPv4 地址

配置示例

```
配置 IP 地址
pfcf_node_id: 10.100.50.241

配置
pfcf_node_id: localhost
pfcf_node_id: upf.example.com
```

## PFCP 配置

### 配置 PFCP 节点

配置

- Web UI 配置“配置”
- SMF 配置“PFCP 配置”

配置

```
1. 检查 PFCP 进程是否运行
sudo netstat -ulpn | grep 8805

2. 检查 iptables 和 ufw 配置
sudo iptables -L -n | grep 8805
sudo ufw status

3. 检查 PFCP 流量
tcpdump -i any -n udp port 8805 -vv

4. 检查 API 是否返回 PFCP 配置
curl http://localhost:8080/api/v1/upf_pipeline
```

检查配置

检查 **PFCP**

检查

```
检查 PFCP 是否监听 UDP 8805
sudo ufw allow 8805/udp
sudo iptables -A INPUT -p udp --dport 8805 -j ACCEPT
```

检查 **PFCP** 的 ID

检查

```
检查 PFCP 的 ID 是否与 N4 的 IP 匹配
pfcpc_node_id: 10.100.50.241 # 检查 N4 的 IP 是否
```

检查 **SMF**

检查

```
ping SMF
ping <SMF_IP>

show SMF route
ip route get <SMF_IP>

add route
sudo ip route add <SMF_NETWORK>/24 via <GATEWAY>
```

## SMF connects UPF IP

Steps

- SMF connects UPF IP
- SMF sends UPF IP `pfcpc_node_id` IP
- SMF connects UPF IP N4 IP

## PFPCP connects

Logs

```
WARN[0030] PFCP heartbeat timeout for association 10.100.50.10
```

Logs

```
curl PFPCP
curl http://localhost:8080/api/v1/upf_pipeline | jq
'.associations[] | {remote_id, uplink_teid_count}'

check logs
journalctl -u omniupf -f | grep heartbeat
```

PFPCP connects

PFPCP connects

000000

```
000 SMF 0000000
ping -c 100 <SMF_IP> | grep loss
```

```
0000000000000000
- 0000000
- 000000/000000
- 0000000
```

0000000000

000000

```
0000000
heartbeat_interval: 30 # 5 000 30 0
heartbeat_retries: 5 # 0000000
heartbeat_timeout: 10 # 0000000
```

0000000000

000000000000**RX/TX** 000 **00**

000

- 0000000 0 RX/TX 000
- UE 0000000000

000



```
1. 检查 XDP 是否安装
ip link show eth0 | grep xdp

2. 检查网卡 UP
ip link show eth0

3. 检查是否安装 XDP 驱动
使用 tcpdump 检查 XDP 是否 GTP-U 数据
curl http://localhost:8080/api/v1/packet_stats
```

检查

## **XDP 安装**

检查

```
安装 OmniUPF 检查 XDP
sudo systemctl restart omniupf

检查
ip link show eth0 | grep xdp
bpftool net list
```

检查是否安装

检查

```
检查
sudo ip link set eth0 up

检查
ethtool eth0 | grep "Link detected"

检查是否安装 XDP 驱动
```

检查是否安装

检查

```
在 config.yml 中配置接口
interface_name: [ens1f0] # 使用 'ip link show' 查看接口名称
```

配置完成后，使用以下命令查看接口状态：

命令：

- RX 和 TX 流量
- 丢包率 > 1%

命令：

```
查看丢包率
curl http://localhost:8080/api/v1/xdp_stats | jq '.drop'

查看路由统计
curl http://localhost:8080/api/v1/packet_stats | jq '.route_stats'

实时监控丢包率
watch -n 1 'curl -s http://localhost:8080/api/v1/packet_stats | jq
".total_rx, .total_tx, .total_drop"'
```

配置完成

配置 PDR 和 TEID 的 UE IP

配置完成

```
00000000
curl http://localhost:8080/api/v1/sessions

000000000000
- PFCP 00000
- SMF 00000
- 000000

00 PDR 0000
bpftool map dump name pdr_map_teid_ip | grep -c key
bpftool map dump name pdr_map_downlin | grep -c key
```

0000

00000

```
00 FIB 0000
curl http://localhost:8080/api/v1/packet_stats | jq '.route_stats'

00 UE IP 000
ip route get 10.45.0.100

00000000
sudo ip route add 10.45.0.0/16 dev eth1 # 0 UE 0000 N6
```

## QER 0000

000

- 00000000
- 0000000000
- URR 0000000000
- 00000000 XDP 000000

000

1. 00000 **MBR** 000

```
確認 QER ID
curl http://localhost:8080/api/v1/pfcp_sessions | jq '.data[] |
select(.ue_ip == "10.45.0.1")'

取得 QER 情報
curl http://localhost:8080/api/v1/qer_map | jq '.data[] |
select(.qer_id == 1)'
```

## 2. 確認

```
確認 0 状態のゲート
curl http://localhost:8080/api/v1/qer_map | jq '.data[] |
{qer_id, ul_gate: .ul_gate_status, dl_gate: .dl_gate_status}'
```

## 3. URRC 確認

```
確認 URRC 情報
curl http://localhost:8080/api/v1/urrc_map | jq '.data[] |
select(.urrc_id == 0)'
```

# 計算式

$$\text{throughput\_kbps} = (\text{volume\_delta\_bytes} \times 8) / \text{time\_delta\_seconds} / 1000$$

## 4. MBR 確認

- 確認  $\approx$  MBR の 95-98% 程度
- 確認 MBR 状態
- 確認 MBR 状態

確認

- MBR 状態 SMF 側 PFCP 状態 QER 状態 MBR
- 確認 SMF 状態
- 確認 SMF 状態 QoS 状態

MBR 状態

OmniUPF    eBPF    MBR

MBR - MBR

5/5

- **VoIP** 音声データ MBR 音声コーデックG.711 = ~80 kbps
- 音声データ MBR > 音声データ + 音声1080p = ~5-10 Mbps
- 音声データ 5ms 音声データ

□ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □

111

- RX N3 □□□□□□ TX N3 □□□□□□□□
- RX N6 □□□□□□ TX N6 □□□□□□□□

111

```
N3/N6 XDP
curl http://localhost:8080/api/v1/n3n6_stats
curl http://localhost:8080/api/v1/packet_stats
```

```
tcpdump XDP GTP-U
API xdpdump
"XDP "XXXXXXXXXX
```

RX N3 TX N6

□□□□□ FAR □□□□ N6 □□□□□

□□□□□

```
00 FAR 0000 FORWARD 00
curl http://localhost:8080/api/v1/sessions | jq '.[].fars[] |
select(.applied_action == 2)'

00 N6 000000
ip route get 8.8.8.8 # 0000000000

00000000000000
sudo ip route add default via <N6_GATEWAY> dev eth1
```

000000**RX N6**00 **TX N3**00

00000000 PDR 000 GTP 00

000000

```
00 UE IP 000 PDR 0000
curl http://localhost:8080/api/v1/sessions | jq '.[].pdrs[] |
select(.pdi.ue_ip_address)'

00 FAR 0000 OUTER_HEADER_CREATION
curl http://localhost:8080/api/v1/sessions | jq '.[].fars[] |
.outer_header_creation'

00 gNB 000
ping <GNB_N3_IP>
```

---

## XDP 0 eBPF 00

000000 **XDP** 000000000000000000 **XDP** 000000

000**XDP** 00000000

000

```
ERR0[0000] failed to load XDP program: invalid argument
```

❏❏❏

```
❏❏❏❏ XDP ❏❏
grep XDP /boot/config-$(uname -r)

❏❏❏❏
CONFIG_XDP_SOCKETS=y
CONFIG_BPF=y
CONFIG_BPF_SYSCALL=y

❏❏ dmesg ❏❏❏❏❏❏❏
dmesg | grep -i bpf
```

❏❏❏❏❏❏❏❏

❏❏❏❏ **XDP** ❏❏

❏❏❏❏❏

```
❏❏❏❏❏❏❏❏❏ XDP ❏❏❏❏❏❏❏❏❏
Ubuntu 22.04+ ❏❏❏❏ XDP
sudo apt install linux-generic-hwe-22.04
sudo reboot
```

**XDP** ❏❏❏❏❏❏

❏❏❏❏❏

```
❏❏ OmniUPF ❏❏❏❏❏❏❏❏❏
journalctl -u omniupf | grep verifier

❏❏❏❏❏
- eBPF ❏❏❏❏❏❏❏❏❏❏❏❏❏❏❏
- ❏❏❏❏❏❏❏❏eBPF ❏❏❏❏❏❏❏

❏❏ eBPF ❏❏❏❏❏❏❏❏❏❏❏❏❏
sudo sysctl kernel.bpf_stats_enabled=1
```

## 🔍 XDP 🔍

🔍

- XDP 🔍 aborted > 0
- 🔍

🔍

```
🔍 XDP 🔍
curl http://localhost:8080/api/v1/xdp_stats | jq '.aborted'

🔍 XDP 🔍
watch -n 1 'curl -s http://localhost:8080/api/v1/xdp_stats'
```

## 🔍 eBPF 🔍

🔍

```
🔍 eBPF 🔍
dmesg | grep -i bpf

🔍 OmniUPF 🔍 eBPF 🔍
sudo systemctl restart omniupf

🔍 eBPF 🔍
🔍 BPF_ENABLE_LOG=1 🔍 OmniUPF
```

---

## 🔍 eBPF 🔍

🔍

- 🔍
- 🔍 100%

🔍



```
取得全データ
curl http://localhost:8080/api/v1/map_info | jq '.[[]] | {map_name, capacity, used, usage_percent}'

取得高負荷データ
curl http://localhost:8080/api/v1/map_info | jq '.[[]] | select(.usage_percent > 90)'
```

確認

```
1. 取得全データ
curl http://localhost:8080/api/v1/sessions | jq '.[[]] | {seid, uplink_teid, created_at}'

2. 取得 SMF 取得全データ
取得 SMF 取得全データ API

3. 取得高負荷データ
watch -n 5 'curl -s http://localhost:8080/api/v1/map_info | jq "[[]] | select(.map_name==\"pdr_map_downlin\") | .usage_percent\"'
```

確認

```
config.yml 取得全データ
max_sessions: 200000 # 100000 取得全データ

取得高負荷データ
pdr_map_size: 400000
far_map_size: 400000
qer_map_size: 200000
```

取得全データ OmniUPF 取得全データ

---

## 前提条件

### ネットワーク環境

環境

- 速度 < 1 Gbps の NIC を使用する
- CPU を使用する

コマンド

```
ネットワーク統計を確認
curl http://localhost:8080/api/v1/packet_stats | jq '.total_rx,
.total_tx'

NIC の状態を確認
ethtool -S eth0 | grep -i drop

XDP の状態を確認
ip link show eth0 | grep xdp
```

確認

確認 **XDP** の状態

確認

```
ネットワーク統計を確認
xdp_attach_mode: native # XDP を NIC/ソフトウェアで実行
```

確認

確認

```
NIC RSS
ethtool -L eth0 combined 4 # 4 RX/TX

RSS
ethtool -l eth0

CPU
/proc/interrupts irqbalance
```

```
#
buffer_max_packets: 5000
buffer_packet_ttl: 15
```

- Ping > 50ms
- 

```
UE
ping -c 100 <UE_IP> | grep avg

#
curl http://localhost:8080/api/v1/upf_buffer_info | jq '.total_packets_buffered'

#
curl http://localhost:8080/api/v1/packet_stats | jq '.route_stats'
```

□□□□□□□□

□□□□□

```
000000000000
curl http://localhost:8080/api/v1/upf_buffer_info | jq '.buffers[]
| {far_id, packet_count, direction}'

000000000000
000 OmniUPF 000 PCF 00000000 FAR0
```

**FIB** □□□□

□□□□□

[illegible]

□ □ □ □ □ □ □ □ □ □

000

- 00000000
- NIC 00 RX 00

111

```
网卡丢包
ethtool -S eth0 | grep -E "drop|error|miss"

网卡统计信息
ethtool -g eth0

实时监控
watch -n 1 'ethtool -S eth0 | grep -E "drop|miss"'
```

网卡配置

```
设置 RX 队列大小
ethtool -G eth0 rx 4096

设置 TX 队列大小
ethtool -G eth0 tx 4096

查看配置
ethtool -g eth0
```

网卡性能优化

网卡性能优化 XDP 加速

## Proxmox VM 网卡 XDP 加速

简介

- 网卡性能优化 XDP 加速
- 网卡性能优化

网卡性能优化 SR-IOV

网卡性能

网卡 1 网卡性能优化

```
xdp_attach_mode: generic
```

## 2. SR-IOV

```
1. Proxmox
1. IOMMU
nano /etc/default/grub
intel_iommu=on iommu=pt
update-grub
reboot

2. VF
echo 4 > /sys/class/net/eth0/device/sriov_numvfs

3. Proxmox UI VF VM
→ PCI → VF

VM
interface_name: [ens1f0] # SR-IOV VF
xdp_attach_mode: native
```

---

## VMware

- OmniUPF

vSwitch MAC

```
vSphere vSwitch
1. vSwitch →
2. →
3. → MAC
4. →
```

# VirtualBox

- < 100 Mbps

VirtualBox SR-IOV XDP

```
#
xdp_attach_mode: generic

VirtualBox
- VirtIO-Net
- "Net"
- CPU
- NAT

KVM/Proxmox
```

## NIC

NIC XDP

```
ERR0[0000] failed to attach XDP program: operation not supported
```

```
查看 NIC 名称
ethtool -i eth0 | grep driver

查看驱动程序 XDP
modinfo <driver_name> | grep -i xdp

查看 XDP 模式
ip link show | grep -B 1 "xdpgeneric\|xdpdrv\|xdpoffload"
```

输出

第 1 个驱动程序

```
xdp_attach_mode: generic
```

第 2 个 NIC 名称

```
在 Ubuntu 上
sudo apt update
sudo apt install linux-modules-extra-$(uname -r)

查看驱动程序
查看模式
查看 https://downloadcenter.intel.com/ 页面
```

第 3 个 NIC

```
查看 XDP 在 NIC 上
- Intel X710/E810
- Mellanox ConnectX-5/ConnectX-6
- Broadcom BCM57xxx/bnxt_en 驱动程序
```

---

驱动程序名称

模式



- 内核 XDP 驱动程序
- NIC 驱动程序

操作

```
查看内核消息
dmesg | tail -100

查看内核日志
journalctl -k | grep -E "BUG:|panic:"
```

故障排除

```
1. 更新内核
sudo apt update
sudo apt upgrade
sudo reboot

2. 设置 XDP 驱动程序为 native
xdp_attach_mode: native

3. 设置 XDP 驱动程序为 generic
xdp_attach_mode: generic

4. 检查 NIC 是否支持 Linux 驱动程序
```

## 网络接口

### 网络接口配置

操作

- SMF 服务配置
- UE 网络 PDU 会话

网络 PFCP 驱动程序 驱动程序驱动程序驱动程序

000

```
00 OmniUPF 00000000
journalctl -u omniupf | grep -i "session establishment"

00 PCFP 0000
curl http://localhost:8080/api/v1/sessions | jq 'length'

0000000000 PCFP 00
tcpdump -i any -n udp port 8805 -w /tmp/pfcp_session.pcap
```

00000

000000

00000

```
00000000
curl http://localhost:8080/api/v1/map_info | jq '.[0] |
select(.usage_percent > 90)'

00000000 eBPF 00000000
```

000 **PDR/FAR** 00

00000

```
00 OmniUPF 00000000
journalctl -u omniupf | grep -E "invalid|error" | tail -20

00000
- 000 UE IP 0000.0.0.0 0000
- 000 TEID00 0000
- PDR 00 FAR
- 000 FAR 00

00 SMF 00000000
```

0000000**UEIP/FTUP**

配置

```
配置 UPF 的 UE IP
feature_ueip: true # 是否 UPF 的 UE IP
ueip_pool: 10.60.0.0/16

feature_ftup: true # 是否 UPF 的 F-TEID
teid_pool: 100000
```

配置

配置

配置

- 配置
- 配置

配置

```
配置
curl http://localhost:8080/api/v1/upf_buffer_info

配置 FAR 配置
curl http://localhost:8080/api/v1/upf_buffer_info | jq '.buffers[] | {far_id, packet_count, oldest_packet_ms}'

配置
watch -n 5 'curl -s http://localhost:8080/api/v1/upf_buffer_info | jq ".total_packets_buffered"'
```

配置

**FAR 配置 FORWARD**

配置 SMF 配置 PCFP 配置 FAR

配置

```
配置 FAR 配置
curl http://localhost:8080/api/v1/sessions | jq '.[].fars[] |
{far_id, applied_action}'

配置 BUFF = 100000
配置 FORW = 200000

配置 BUFF 配置 SMF
- 配置 PFCP 配置
- 配置 FAR 配置 FORW 配置
```

配置 **TTL** 配置

配置 FAR 配置

配置

```
配置 TTL
buffer_packet_ttl: 60 # 配置 30 配置 60 配置
```

配置

配置 FAR 配置

配置

```
配置
buffer_max_packets: 20000 # 配置 FAR
buffer_max_total: 200000 # 配置
```

---

□□□□

□□□□□□

```
logging_level: debug # trace | debug | info | warn | error
```

```
□□□□□□□□ OmniUPF
sudo systemctl restart omniupf

□□□□□□
journalctl -u omniupf -f --output cat
```

---

## eBPF □□□□

```
□□ eBPF □□□□□□□□ bpftrace□
sudo bpftrace -e 'tracepoint:xdp:* { @[probe] = count(); }'

□□□□□□
sudo bpftrace -e 'tracepoint:bpf:bpf_map_lookup_elem {
printf("%s\n", str(args->map_name)); }'
```

---

## □□ **XDP** □□□□□□

□□ **XDP** □□□□□□□□

XDP □□□□□□ □□ □□□□□□□□□□ **tcpdump** □□□□ **XDP** □□□□□□□□N3 □□ GTP-U □□□□□□□□UDP  
2152□□ XDP □□□□□□□□□□□□ UPF □□□□□□ **tcpdump** □□

□□□□□□□□□□□□

```
1. API
curl http://localhost:8080/api/v1/xdp_stats
curl http://localhost:8080/api/v1/packet_stats | jq
curl http://localhost:8080/api/v1/n3n6_stats

2. PFCP
tcpdump -i any -n udp port 8805 -w /tmp/pfcp.pcap

3. GTP-U
TAP
#
- gNB
- UPF
- SPAN/N3
- hypervisor
#
UPF
tcpdump -i <mirror_interface> -n udp port 2152 -w /tmp/n3_mirror.pcap
```

```
TAP
Cisco SPAN
(config)# monitor session 1 source interface Gi1/0/1
(config)# monitor session 1 destination interface Gi1/0/24

Gi1/0/24
tcpdump -i eth0 -n udp port 2152 -w /tmp/n3_capture.pcap
```

**VMware KVM**

```
UPF VM
Linux tcpdump VM
hypervisor UPF N3

VM
tcpdump -i eth1 -n udp port 2152 -w /tmp/n3_virtual.pcap
```

## 環境構築

- XDP 環境構築
- 仮想 NIC 環境構築
- 仮想 tcpdump を XDP 環境で実行
- 仮想 UPF 環境構築

## 仮想 UPF 環境構築

- 仮想 PFCP 仮想 UDP 8805 - 仮想 XDP 環境
- 仮想 API 環境
- 仮想 GTP-U 仮想 UDP 2152 - 仮想 XDP 環境

## 確認

### 環境構築確認

#### 1. 環境構築

```
確認
uname -a
cat /etc/os-release

OmniUPF 確認
curl http://localhost:8080/api/v1/upf_status
curl http://localhost:8080/api/v1/map_info
curl http://localhost:8080/api/v1/packet_stats

ログ
journalctl -u omniupf --since "1 hour ago" > /tmp/omniupf.log
dmesg > /tmp/dmesg.log

ネットワーク
ip addr > /tmp/network.txt
ip route >> /tmp/network.txt
ethtool eth0 >> /tmp/network.txt
```

## 2. 网络层

- OmniUPF 网络
- Linux 网络
- 网络层
- 网络层
- 网络层
- 网络层

---

## 网络层

- 网络层 - 网络层
- 网络层 - eBPF/XDP 网络层
- 网络层 - 网络层
- 网络层 - 网络层 Prometheus 网络
- **PFCP** 网络层 - PFCP 网络层
- 网络层 - PDR/FAR/QER/URR 网络
- 网络层 - UPF 网络层



# Web UI 開發

## 目錄

1. 簡介
2. 專案環境
3. 網路架構
4. 網路設備
5. 網路協議
6. 網路安全
7. 網路優化
8. 網路監控
9. 網路故障排除
10. XDP 網路設備
11. 網路安全

## 目錄

OmniUPF Web UI 開發環境與工具  
Phoenix LiveView 開發環境

- 網路設備 PFCP 網路設備 PDU 開發
- 網路設備 PDR、FAR、QER 及 URR 開發
- 網路設備 網路設備開發
- 網路設備 網路設備開發
- 網路設備 eBPF 網路設備開發
- 網路設備 網路設備開發

## 目錄

網路設備 REST API 開發 OmniUPF 網路設備開發

- 配置 PCF 代理
- 配置策略引擎
- 配置策略引擎 
- 配置策略引擎
- 配置 eBPF 代理

## 配置策略引擎

### 配置策略引擎

配置策略引擎 HTTPS 代理 OmniUPF 代理

```
https://<upf-server>:443/
```

配置策略引擎 443 代理 HTTPS

### 配置策略引擎

配置策略引擎 `config/config.exs` 代理 OmniUPF 代理

配置策略引擎 UPF 代理

`upf_hosts` 配置策略引擎 UI 代理 OmniUPF 代理

### 配置策略引擎

配置策略引擎

- 配置 - `/sessions` - PCF 代理
- 配置 - `/rules` - PDR/FAR/QER/URR 代理
- 配置 - `/buffers` - 配置策略引擎
- 配置 - `/statistics` - 配置策略引擎 XDP 代理
- 配置 - `/capacity` - eBPF 代理
- 配置 - `/upf_config` - UPF 代理

- `/routes` - UE `OSPF`/`BGP`
- **XDP** `/xdp_capabilities` - XDP
- `/logs` -

**URL**`/sessions`

OmniUPF `PFCP`

**PFCP**

`PFCP` `SMF/PGW-C`

| <b>ID</b> | <code>SMF</code> <code>PGW-C</code> <code>FQDN</code> <code>IP</code> |
|-----------|-----------------------------------------------------------------------|
|           | <code>SMF/PGW-C</code> <code>PFCP</code> <code>IP</code>              |
| <b>ID</b> | <code>PFCP</code> <code>ID</code>                                     |

- `SMF` `UPF`
- 
- `ID`

`PFCP` `UE PDU`

| 項目      | 説明                      |
|---------|-------------------------|
| UE SEID | UPF での UE の識別子          |
| SM SEID | SMF での UE の識別子          |
| UE IP   | UE の IPv4 または IPv6 アドレス |
| TEID    | UE と GTP-U ネットワーク間の識別子  |
| PDRs    | UE のパケットデータレコーズ         |
| FARs    | UE のフィルタリングルール          |
| QERs    | UE の QoS 設定             |
| URRs    | UE のユーティリティレコーズ         |
| その他     | UE のその他の設定              |

UE

- UE IP アドレスを UE IP とする
- TEID を識別子 ID とする
- PDR/FAR/QER/URR JSON を取得
- 10 秒間隔

UE のパケットデータレコーズ

UE のパケットデータレコーズ (PDRs) を取得

- UE のパケットデータレコーズ (PDRs) を取得
  - PDR ID を取得 - UE のパケットデータレコーズ ID を取得
  - UE の PDR ID が 0 でないことを確認
  - UE の PDR ID が IPv4 のパケットデータレコーズ ID
  - UE の PDR ID が IPv6 のパケットデータレコーズ ID
- UE のパケットデータレコーズ (FARs) を取得

- **QoS** **profiles (QERs)** MBR GBR QFI **maps** QoS
- **URRs** **rules**

*Network elements store PDRs, FARs, and QERs*

UE

UE **UE** **context**

1. **UE context**
2. **UE IP**
3. **TEID**
4. **PDR/FAR**

**Network**

- **Network**
- **UPF** **context**
- **Network**

UE Information

- UE IP & TEID
- QoS parameters
- FAR rules
- QER QoS rules

UPF

UPF 10 QoS rules

- UPF QoS rules
- UPF QoS rules
- QoS rules

UPF

URL /rules

UPF QoS rules

**PDR** rules - QoS rules

UPF PDR rules

**PDRs** (N3 → N6):

- TEID QoS rules PDR rules
- **TEID** gNB & GTP-U ID rules - QoS rules
- **FAR ID** QoS rules - FAR rules
- **QER ID** QoS rules - QER rules
- **URR IDs** QoS rules - URR rules
- GTP-U rules
- **SDF** QoS rules

**PDRs** (N6 → N3):

- 000000 UE IPv4 000000000000 PDR 0000
- **UE IP**000000 IPv4 000000000000
- **FAR ID**0000000000000000 - 000 FAR 0000
- **QER ID**0000 QoS 0000000000 - 000 QER 0000
- **URR IDs**0000000000000000 - 000 URR 0000
- **SDF** 0000000000000000 sdf\sdf + 000
- 000000000000 PDR00000 100000 1000

### IPv6 00 PDRs0

- API 00 IPv6 00 PDR 000
- 000 IPv4 00000 IPv6 0000
- 000000000000 UI 000

## FAR 000 - 0000000

0000 FAR 000000000000

000

- 000000 FAR ID 00000000 FAR 0000
- 000000 PDR 00000000 FAR ID 00000000
- 000000FAR 000000000000

| 0             | 00                                            |
|---------------|-----------------------------------------------|
| <b>FAR ID</b> | 000000000000                                  |
| 00            | 00000000FORWARD0DROP0BUFFER0DUPLICATE0NOTIFY0 |
| 00            | 0000000000/000                                |
| 00            | 000000000TEID0IP 000                          |

### FAR 00000

- **FORWARD (1)**000000000000

- **DROP (2)** packets
- **BUFFER (4)** packets
- **NOTIFY (8)** packets
- **DUPLICATE (16)** packets

1.

- 1. “**QoS**” “**QoS**”
- 2. **QoS**
- 3. **eBPF** **FAR**

**QoS - QoS**

1. QoS

1.

- 1. **PDR** **QoS ID** **QoS**
- 2. **PDR** **QoS**
- 3. **QoS** **100** **1000**

| QoS           | QoS                       |
|---------------|---------------------------|
| <b>QoS ID</b> | QoS <b>QoS</b> <b>PDR</b> |
| <b>MBR</b>    | kbps                      |
| <b>MBR</b>    | kbps                      |
| <b>GBR</b>    | kbps                      |
| <b>GBR</b>    | kbps                      |
| <b>QFI</b>    | QoS <b>5G</b>             |

**QoS**



- **MBR = 0** 00000000
- **GBR = 0** 0000000000000000
- **GBR > 0** 0000000000000000

## URR 0000 - 00000000

0000000000000000

0000

- 00000000 URR ID 0000000000000000 URR
- 00000000 PDR 00000000 URR ID 0000000000 URR
- 00000000 PDR 0000000000000000URR 0000000000
- 00000000000000 URR000000 100000 10000

| 0             | 00                            |
|---------------|-------------------------------|
| <b>URR ID</b> | 0000000000000000 PDR 00000000 |
| 00000         | 0 UE 00000000000000           |
| 00000         | 0000000000 UE 00000           |
| 0000          | 0000000000                    |
| 00            | 0000000000 URR 00000          |

000000

- 00000000B0KB0MB0GB0TB0
- 0000000000000000
- 0000000000

0000

- 0000000000 URR
- 00000000000 0 00000 URR00000000





- 000000000000 FAR 000000000000
- 0000000000000000000000000000

00000000

- 000000“0000”00
- 0000 FAR 0000
- 0000

00

00000000

1. 00000000000000000000
2. 00 FAR 000000000000
3. 000000000000

000000

1. 0000000000“00”0000000000
2. 00000000000000
3. 00“0000”000000

0000000000

1. 000000000000 FAR0000000000
2. 00“00”0000000000
3. 000“0000”00000000

0000000000

1. 00000000000000000000
2. 0000000000 FAR
3. 00 SMF 0000000000000000
4. 00 SMF 000000000000

# 概要

バージョン 5 からサポートされています。

## インストール

URL `/statistics`

### 前提

インストールする OmniUPF は Prometheus をインストールする必要があります。

### インストール

インストール方法

- インストール方法
- インストール方法
- インストール方法
- **GTP-U** インストール GTP-U インストール

インストール UPF インストール

### インストール

インストール方法

- インストール方法
- インストール方法
- インストール方法

インストール方法

## XDP

eXpress Data Path

- **XDP** インストール XDP インストール

- **XDP** 可編程數據平面
- **XDP** 可編程 XDP 數據平面
- **XDP** 可編程 XDP 數據平面

可編程 XDP 數據平面

## XDP 可編程

- 可編程數據平面
- eBPF 可編程數據平面
- 可編程數據平面
- 可編程數據平面

## N3/N6 可編程

可編程數據平面

## N3 可編程 RAN 可編程

- 可編程 **N3** 可編程 gNB/eNodeB 可編程數據平面
- 可編程 **N3** 可編程 gNB/eNodeB 可編程數據平面

## N6 可編程數據平面

- 可編程 **N6** 可編程數據平面/IMS 可編程數據平面
- 可編程 **N6** 可編程數據平面

可編程數據平面

可編程數據平面

可編程

可編程數據平面

1. 可編程數據平面/可編程
2. 可編程數據平面
3. 可編程 **N3** 可編程 **N6** 可編程數據平面

□□□□□□□□

1. 0000000000000000
2. 00 XDP 00000000
3. 0000000000000000

□ □ □ □ □

1. ☐ XDP ☐☐☐☐☐☐☐☐
2. ☐ XDP ☐☐☐☐☐☐☐☐
3. ☐ N3/N6 ☐☐☐☐☐

□□□□□

1. □□□□□□□□
2. □ UPF □□□□□□
3. □□□□□□□□

□ □ □ □

□□□□□ 5 □□□□□□

**URL** `/capacity`

11

□□□□□ UPF □□□□□□□□□ eBPF □□□□□□□□□

**eBPF** ☐ ☐ ☐ ☐ ☐

□□ eBPF □□□□□□□□□□

| 項目   | 説明                                    |
|------|---------------------------------------|
| 概要   | eBPF を利用して uplink_pdr_map と far_map を |
| 目的   | パケットの転送先を動的に変更する                      |
| 対象   | パケットの転送先を変更する                         |
| 前提条件 | パケットの転送先を変更する                         |
| 手順   | パケットの転送先を変更する                         |
| 結果   | パケットの転送先を変更する                         |

パケットの転送先

パケットの転送先を変更する

- パケット転送率 <50% のパケットを転送する
- パケット転送率 50-70% のパケットを転送する
- パケット転送率 70-90% のパケットを転送する
- パケット転送率 >90% のパケットを転送する

パケットの転送先を変更する

uplink\_pdr\_map

- パケット TEID をパケット PDR
- パケット転送率
- パケット転送率

downlink\_pdr\_map / downlink\_pdr\_map\_ip6

- パケット UE IP をパケット PDR
- パケット UE IPv4/IPv6 をパケット
- パケット転送率



## far\_map

- FAR ID
- PDR
- 

## qer\_map

- QER ID QoS
- QoS

## urr\_map

- URR ID
- 

- 1.
- 2.
- 3.

1. PDR
- 2.
- 3.

- 1.
2. PDR >90%
- 3.

- 1.

2. ــــــــــــــــــــــــــــــــــــــــ
3. ــــــــــــــــــــــــــــــــــــــــــــــــــــــــ

## ــــــــــــــــ

eBPF ــــــــــــــــ UPF ــــــــــــــــ UPF ــــــــــــــــــــــــــــــــــــــــــــــــ

- ــــــــــــــــــــــــــــــــ 10,000 - 100,000 ــــــــ
- ــــــــــــــــــــــــــــــــ 100,000 - 1,000,000 ــــــــ
- ــــــــــــــــــــــــــــــــ 1,000,000+ ــــــــ

ــــــــــــــــ

$$\text{ــــــــــــــــ} = (\text{ــــــــ} + \text{ــــــــ}) \times \text{ــــــــ}$$

ــــــــــــــــ 100 ــــــــ 64 ــــــــ PDR ــــــــــــــــ 64 MB ــــــــــــــــ

## ــــــــــــــــ

ــــــــــــــــ 10 ــــــــــــــــ

## ــــــــــــــــ

**URL** ــــــــ /upf\_config

ــــــــ

ــــــــــــــــ UPF ــــــــــــــــــــــــــــــــــــــــ

**UPF** ــــــــ

ــــــــ UPF ــــــــ

- **PFCP** ــــــــSMF/PGW-C ــــــــ IP ــــــــــــــــ
- **N3** ــــــــRANgNB/eNodeBــــــــ IP ــــــــ
- **N6** ــــــــــــــــــــــــ IP ــــــــ

- **N9** 通過UPF 通過IP 通過
- **API** 通過REST API 通過
- 通過OmniUPF 通過

通過**eBPF**通過

通過通過通過通過通過

- 通過 **N3** 通過通過 N3 通過
- 通過 **N9** 通過通過 N9 通過通過通過

通過通過通過 eBPF 通過通過通過通過通過通過通過通過

通過

通過 **UPF** 通過

1. 通過 N3 通過 IP 通過 gNB 通過
2. 通過 N6 通過通過通過通過通過
3. 通過 PCFP 通過通過 SMF 通過

通過通過通過

1. 通過通過通過通過通過通過
2. 通過通過通過
3. 通過通過

通過通過

1. 通過 UPF 通過通過
2. 通過通過通過通過
3. 通過

通過通過

**URL** `/routes`

概要

本ドキュメントは、UEのIPアドレスをOSPFとBGPで管理する方法について説明します。

前提条件

本ドキュメントを実行するには、以下の条件を満たす必要があります。

- ネットワーク環境が正常に構築されていること。
- UEのIPアドレスが設定されていること。
- OSPFとBGPの構成が正しいこと。
- ネットワークの接続が正常に確立されていること。

UE IP の設定

UEのIPアドレスを設定するための手順は以下の通りです。

| 項目        | 設定値                          |
|-----------|------------------------------|
| IP アドレス   | 192.168.1.100                |
| UE IP の設定 | UE の IP を IPv4 と IPv6 の両方に設定 |

確認

- UEのIPアドレスが正しく設定されていることを確認します。
- OSPFとBGPの構成が正しいことを確認します。
- ネットワークの接続が正常に確立されていることを確認します。

OSPF の設定

OSPFを設定するための手順は以下の通りです。

| 項目             | 説明              |
|----------------|-----------------|
| Router ID      | OSPF 識別子        |
| Interface      | OSPF 対応 IP アドレス |
| Area           | OSPF 領域         |
| Cost           | OSPF 経路コスト      |
| Priority       | OSPF 優先度        |
| Authentication | OSPF 認証         |
| Timers         | OSPF タイマー       |

**OSPF 設定**

- Router ID の設定
- Interface の OSPF 設定

**BGP 設定**

BGP 設定

| 項目  | 説明             |
|-----|----------------|
| IP  | BGP の宛先 IP     |
| ASN | 自治システム番号       |
| 経路  | BGP の経路        |
| 入/出 | 入/出の方向         |
| 宛先  | 宛先の IP         |
| 宛先  | 宛先の IP BGP の宛先 |
| 宛先  | 宛先の IP BGP の宛先 |

**BGP の**

- BGP の宛先 IP
- BGP の宛先 IP

BGP の ID は ASN の BGP の

**OSPF の**

OSPF の LSA の

| 項目     | 説明                       |
|--------|--------------------------|
| ルータ ID | LSA の識別子                 |
| タイプ    | LSA の種類                  |
| リンク ID | リンクの識別子                  |
| リンクタイプ | OSPF のリンクタイプ (E1 または E2) |
| コスト    | リンクの OSPF コスト            |
| ネイバー   | LSA の隣接ルータ               |
| フラグ    | LSA のフラグ                 |

機能

- ルータ UE が OSPF を実行
- ルータがリンクを形成
- ルータ LSA を生成

動作

動作

- ルータが FRR を実行
- ルータが UE を実行
- ルータがリンクを形成

動作

- ルータが OSPF を実行
- ルータが BGP を実行

11

|  |  |  |  |  |  |  |  |  |
|--|--|--|--|--|--|--|--|--|
|  |  |  |  |  |  |  |  |  |
|--|--|--|--|--|--|--|--|--|

1. 配置OSPF
2. 配置BGP
3. 配置OSPF和BGP联动
4. 配置OSPF和BGP联动的高级应用

UE

1. 四字节 UE IP 四字节四字节 UE
2. 四字节 OSPF 四字节四字节
3. 四字节 UE 四字节四字节 LSA 四
4. 四字节四字节四字节 UPF 四

□ □ □ □ □ □ □ □ □ □

1. 配置 OSPF 进程
2. 配置 OSPF 网络
3. 配置 OSPF 接口
4. 配置 OSPF/BGP 路由

□□□ **UPF** □□□

1. 四連環 UPF 四
2. 四連環四連環
3. 四 OSPF 四連環
4. 四 BGP 四連環

□ □ □ □ □ □ □

1. UE 注册失败的原因
2. 鉴权失败的原因
3. UE OSPF LSA 问题
4. UE BGP 问题



環境

環境 10 環境環境環境環境環境環境 UE 環境

環境

環境環境 UPF 環境 FRR環境環境環境環境

- **OSPF**環境環境環境 2 LSA 環境
- **BGP**環境環境環境 BGP 環境
- 環境環境REST API 環境 vtysch 環境 FRR

XDP 環境

URL環境/xdp\_capabilities

環境

XDP 環境環境 eXpress Data Path (XDP) 環境環境環境 UPF 環境環境環境環境

環境

環境環境環境環境環境

| 環境   | 環境                                |
|------|-----------------------------------|
| 環境   | 環境 XDP 環境環境環境eth0環境ens1f0環境       |
| 環境   | 環境環境環境環境i40e環境ixgbe環境virtio_net環境 |
| 環境環境 | 環境環境環境                            |
| 環境   | 環境 XDP 環境DRV環境SKB 環境 NONE環境       |
| 環境   | 環境環境 NIC 環境                       |

## XDP 概要

概要として XDP を簡単に説明します

### XDP\_DRV 概要

- 100~5-10 Mpps 対応
- 一般的な XDP 対応 NIC
- XDP 対応 NIC 例: i40e, ixgbe, mlx5
- XDP 対応 NIC 例: i40e, ixgbe, mlx5
- XDP 対応 NIC 例: i40e, ixgbe, mlx5

### XDP\_SKB 概要

- 100~1-2 Mpps
- XDP 対応 NIC
- XDP 対応 NIC
- XDP 対応 NIC
- XDP 対応 NIC

概要として XDP を簡単に説明します

- XDP 対応 NIC
- XDP 対応 NIC

概要として XDP を簡単に説明します

- XDP 対応 NIC
- XDP 対応 NIC

*XDP* *XXXXXXXXXXXXXXXXXXXX Mpps XXXXXX*

XX

XXXXXXXXXXXXXXXXXXXX

XXXXXXXXXX

- "✓ XXXXX XDP\_DRV XXXXXXXXXXXXXXX"
- XXXXXXXXXXXXXXX

XXXXXXXXXX

- "△ XXXXX XDP\_DRV XXXXXXXXXXXXXXX"
- XXXXXXXXXXXXXXXXXXXXXXX
- "△ XXXXXXXXXXXXXXX XDP\_DRV"
- XXXXXXXXXXXXXXX

XXXXXXXXXX

- XX XDP XXXXXXXX

# Mpps 計算式

計算式は、Mpps と Gbps の関係を示す。

計算式

計算式 Mpps

- 0.1 - 100 Mpps
- XDP Mpps
- 

計算式

- 64 - 9000
- 1200 GTP
- GTP

計算式

- 64B
- 128B
- 256B
- 512B
- 1024B
- 1518B

計算式

計算式 Gbps

- 
- $Gbps = Mpps \times Packet\_Size \times 8 / 1000$
- GTP UDP IP

計算式 Gbps

-

- 1000 ~50 1000 GTP 10000
- 1000  $\text{Gbps} = \text{Mpps} \times (\text{Packet\_Size} - 50) / 1000$

1000000

- 100 Mpps 1000000000000000
- 1000 10 Mpps = 10,000,000 100000

100000

- 1000000000
- 1000  $10 \text{ Mpps} \times 1200 \text{ 10} \times 8 \text{ 10/10} \div 1000 = 96 \text{ Gbps}$

## 100 Mpps

1000000000000000

1000 Mpps

- 100000000
- 1000000000000000
- 1000000000

100000000

- 1000000000000000 Mpps = 1000 Gbps
- 1000000000000000 Mpps = 1000 Gbps
- 1000000000000000

**GTP** 100000

- 100000 14 10
- IP 1000 20 1000 IPv4 1000 40 1000 IPv6
- UDP 1000 8 10
- GTP 1000 8 1000000
- 1000000000000000 ~50 10

11

# XX XDP XXX

1. 测试 XDP 性能
2. 测试 XDP 性能与 DRV 性能对比
3. 测试 Mpps 性能
4. 测试延迟

□□□□□□□□

1. 1000000000Mbps
2. 1000000000000000000
3. 1000000000Gbps
4. 10000000000000000

## ❏ XDP ❏❏❏

1. 0x00000000 XDP\_DRV 0
2. 0x00000000
3. 0x0000000000000000
4. 0x00000000 CPU 0

□□□□□

1. 1000000000000 Mpps
2. 100 XDP 100000000
3. 1000000000
4. 1000000000000000

□ □ □ □ □ □ □ □ □

1. XDP DRV SKB
- 2.
- 3.
- 4.

# 

## 

- 支持 XDP 的 NIC (Intel i40e/ixgbe, Mellanox mlx5)
- 支持 NIC 的 XDP 驱动
- 支持 RSS 负载均衡
- 支持 NIC 的 XDP 驱动

## 

- 支持 XDP 驱动
- 支持 XDP 驱动
- 支持 XDP 驱动

## 

- 支持 CPU 的 XDP 驱动
- 支持 XDP 驱动
- 支持 RSS 负载均衡

# 

XDP 支持 30 个 XDP 驱动

# 

URL: /logs

# 

支持 OmniUPF 支持

# 

- 支持 Phoenix LiveView 支持
- 支持 XDP 驱动

- 設定項目
- 設定項目

## 設定

OmniUPF 設定 Elixir Logger 設定

- **DEBUG** 設定
- **INFO** 設定
- **WARNING** 設定
- **ERROR** 設定

## 設定

設定項目

1. 設定
2. 設定 SMF 設定
3. 設定 PCF 設定

設定 **PCF** 設定

1. 設定 PCF 設定
2. 設定 PCF/PCF/PCF
3. 設定

設定項目

1. 設定項目
2. 設定 eBPF 設定
3. 設定 FAR/PDR 設定



# 网络层

## 网络层

### 网络层

- 网络层的主要功能
- 网络层的主要协议
- 网络层的主要设备
- 网络层 XDP 技术

### 网络层

- 网络层的主要功能
- 网络层的主要协议 TTL 和分片
- 网络层的主要设备
- 网络“层”和“网”的概念

### 网络层

- 网络层的主要功能 UE 层
- 网络层的主要设备
- 网络层 UPF 设备
- 网络层的主要协议

### 网络层

- 网络层的主要功能
- 网络层的主要设备 UE 层
- 网络层的主要设备
- 网络层的主要协议

## 网络

- 网络层的主要设备 5-10 层
- 网络层的主要设备

- 5G Core Network (5GC) URR (User Rate Restriction) 5G Core Network
- 5G Core Network UPF (User Plane Function) 5G Core Network

## 5G Core Network

- **5G Core Network** - PDR (Policy Data Rule) FAR (Forwarding Action Rule) QER (QoS Enforcement Rule) URR (User Rate Restriction) 5G Core Network
- **5G Core Network** - 5G Core Network
- **5G Core Network** - Prometheus 5G Core Network
- **PFCP** (Protocol Function Control Plane) - PFCP (Protocol Function Control Plane) 5G Core Network
- **API** (Application Programming Interface) - REST API 5G Core Network
- **5G Core Network** - UE (User Equipment) FRR (Fast Reroute) 5G Core Network
- **XDP** (eXpress Data Path) - XDP (eXpress Data Path) eBPF (eBPF) 5G Core Network
- **5G Core Network** - 5G Core Network
- **UPF** (User Plane Function) - UPF (User Plane Function) 5G Core Network

# OmniUPF 与 XDP 部署指南

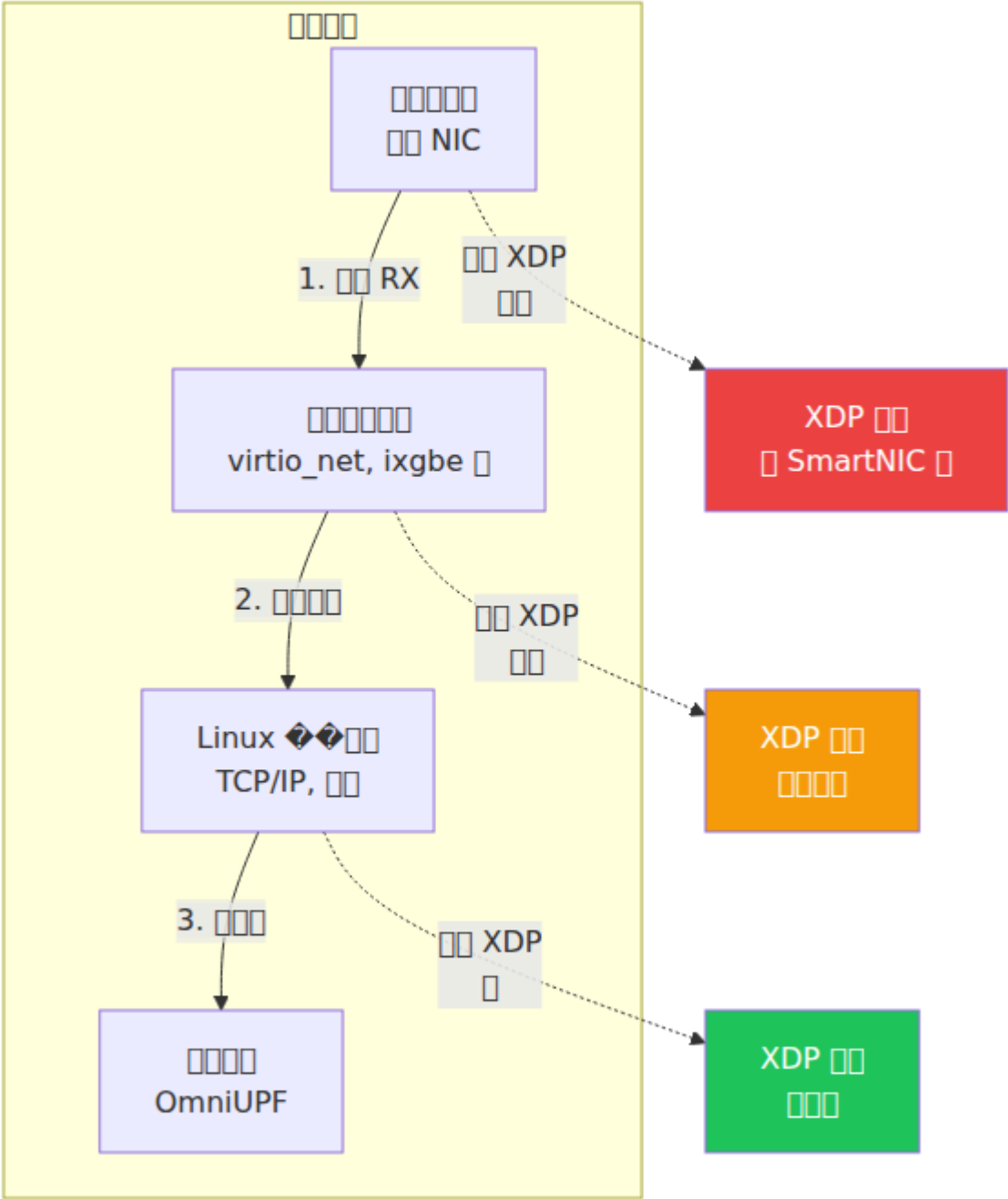
## 目录

- 简介
- XDP 概述
- 部署环境
- 网络拓扑
- 安装 SmartNIC
- 在 Proxmox VE 上部署 XDP
- 配置 XDP 程序
- 验证 XDP 功能
- 性能测试

## 简介

OmniUPF 利用 **XDP (eXpress Data Path)** 技术，在 Linux 内核中实现高性能的网络数据包处理。XDP 是 Linux 内核中一种新的网络数据包处理机制，它允许在网卡驱动层直接处理数据包，从而避免了用户空间和内核空间之间的上下文切换，显著提高了网络性能。

XDP 与 **eBPF** 技术相结合，可以实现更灵活的网络策略配置。



XDP SmartNIC OmniUPF

# XDP

| 項目     | Linux 標準                 | 標準                      | 標準                       |
|--------|--------------------------|-------------------------|--------------------------|
| 性能     | Linux 標準                 | 標準                      | NIC 標準                   |
| 速度     | ~1-2 Mpps                | ~5-10 Mpps              | ~10-40 Mpps              |
| 遅延     | ~100 μs                  | ~10 μs                  | ~1 μs                    |
| CPU 消費 | 高                        | 低                       | 低                        |
| NIC 消費 | 標準 NIC                   | 標準 XDP 標準               | 標準 XDP 標準 SmartNIC       |
| 拡張性    | 標準                       | 標準                      | 標準 PCI 標準                |
| コスト    | 標準                       | 標準                      | 標準                       |
| 設定     | xdp_attach_mode: generic | xdp_attach_mode: native | xdp_attach_mode: offload |

Two rows of empty boxes for writing answers. Each row contains 15 boxes, with a gap between the 10th and 11th boxes in each row.



11

XX XDP XXXXXXXXXXXX X Linux XXXXXXXX eBPF XXXXXXXXXXXX XDP XXXXXXXXXXXXXXXXXXXX

|  |  |  |  |
|--|--|--|--|
|  |  |  |  |
|--|--|--|--|

## 目標

- 環境構築
- 設定確認
- 動作確認
- トラブルシューティング

## NIC 種類

NIC XDP 対応 NIC 対応 XDP NIC 対応 XDP

### NIC 種類

- ixgbe 10G i40e 40G ice 100G
- bnxt\_en
- mlx4\_en mlx5\_core
- Netronome nfp
- Marvell mvneta mvpp2

### NIC 種類

- VirtIO virtio\_net KVM Proxmox OpenStack ✓
- VMware vmxnet3 ✓
- hv\_netvsc Hyper-V ✓
- ena AWS ✓
- SR-IOV ixgbevf i40evf PCI NIC ✓

VirtualBox NIC XDP 対応

## 設定

```
config.yaml
interface_name: [eth0]
xdp_attach_mode: native
```

Proxmox NIC 設定

---

# SmartNIC

XDP NIC SmartNIC eBPF CPU

- ~10-40 (Mpps)
- ~1
- CPU** NIC

- UPF 10G+
- 
- CPU

Netronome Agilio SmartNIC XDP

- Netronome Agilio CX 10G/25G/40G/100G

PCI -

```
config.yaml
interface_name: [eth0]
xdp_attach_mode: offload
```

---



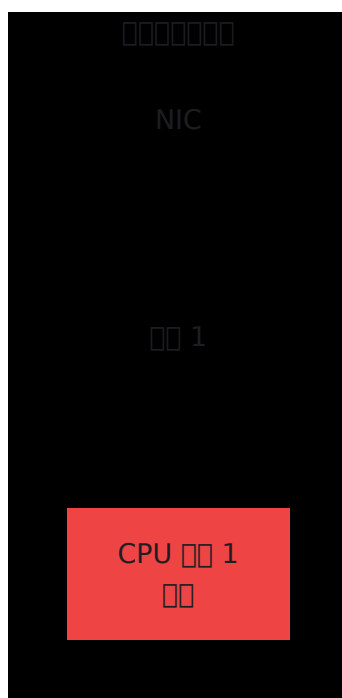
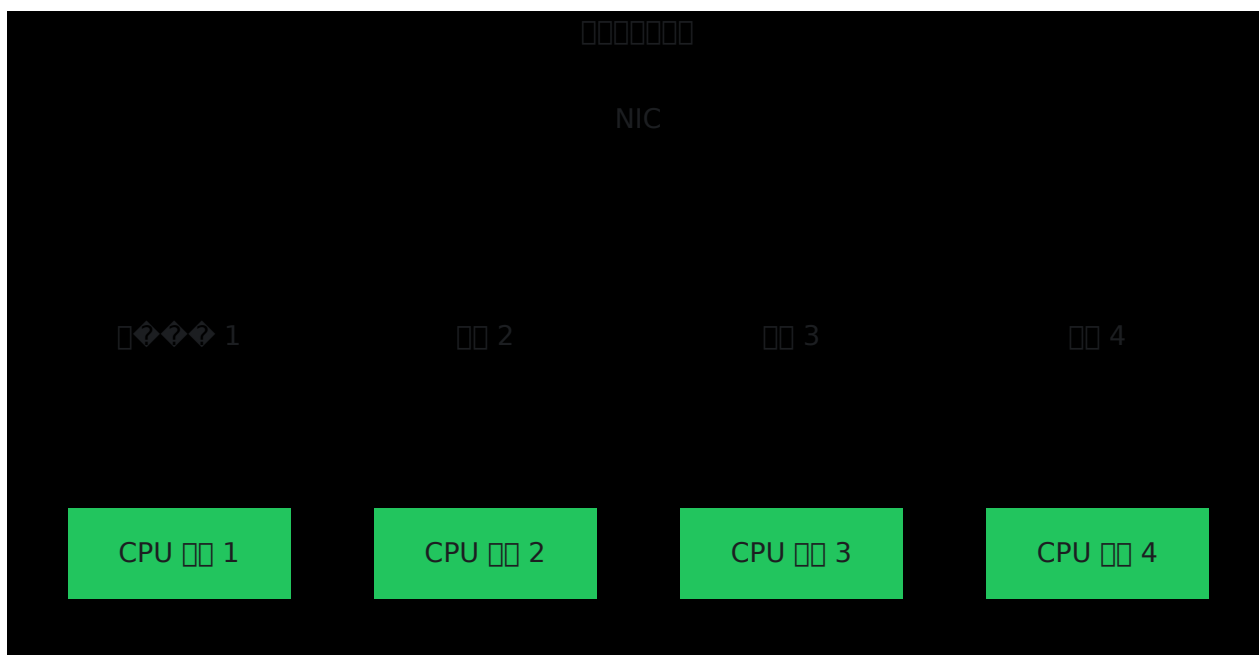
# Proxmox VE 仮想マシン XDP

Proxmox VE 上で **VirtIO** 仮想ネットワークカード `virtio_net` を利用して XDP を実装する方法について説明します。

## 1. 環境構築

仮想マシンを起動し、ネットワークカードを VirtIO 型に設定します。

- 仮想マシンの設定画面で、ネットワークカードのタイプを `VirtIO` に変更します。
- 仮想マシンの設定画面で、ネットワークカードのモデルを `virtio_net` に変更します。



## 第 2 步 Proxmox 安装

### 安装 A 步骤 Proxmox Web UI

#### 1. 安装 Proxmox Web UI

- 安装 Proxmox Web UI
- 安装 Proxmox

## 2. 設定

- 名前空間を設定
- 仮想ネットワークインターフェース名 `net0` を設定
- 名前空間を設定

## 3. 設定

- 名前空間名を設定
- 仮想 CPU 数 **8** を設定 vCPU 数を **16** に設定
- 名前空間を設定

## 4. 設定

- 名前空間を設定

## Proxmox での操作

```
SSH で Proxmox に接続

仮想マシン ID を取得
qm list

仮想マシン XXX の仮想ネットワーク ID を取得
qm set XXX -net0 virtio=XX:XX:XX:XX:XX:XX,bridge=vmbr0,queues=8

仮想マシン 191 の MAC アドレスを BC:24:11:1D:BA:00 に設定
qm set 191 -net0 virtio=BC:24:11:1D:BA:00,bridge=vmbr0,queues=8

仮想マシン XXX をシャットダウン
qm shutdown XXX

仮想マシン XXX を再起動
qm start XXX
```

## 注意事項

- **4** vCPU 以下の仮想マシンは 2-4 vCPU を推奨
- **8** vCPU 以上の仮想マシンは 4-8 vCPU を推奨
- **16** vCPU 以上の仮想マシンは 8+ vCPU を推奨

## 📌 3 📌 📌 📌 📌 📌 📌 📌 📌

📌 📌 📌 📌 SSH 📌 📌 📌 📌 📌

```
📌 📌 📌 📌
ethtool -l eth0

📌 📌 📌 📌
eth0 📌 📌 📌 📌
Combined: 8 <-- 📌 📌 📌 📌 📌 📌

📌 📌 📌 📌
ls -ld /sys/class/net/eth0/queues/rx-* | wc -l
ls -ld /sys/class/net/eth0/queues/tx-* | wc -l

📌 📌 📌 📌 8 📌 📌 📌 📌 📌 📌
```

## 📌 4 📌 **OmniUPF** 📌 📌 📌 **XDP**

📌 OmniUPF 📌 📌

```
📌 📌 📌 📌
sudo nano /config.yaml
```

📌 XDP 📌 📌

```
📌 📌
xdp_attach_mode: generic

📌 📌
xdp_attach_mode: native
```

📌 OmniUPF 📌

```
sudo systemctl restart omniupf
```

## 📄 5📄📄📄📄 XDP 📄📄📄📄📄📄

📄📄📄📄

```
📄📄📄📄
journalctl -u omniupf --since "1 minute ago" | grep -i
"xdp\|attach"

📄📄📄📄
xdp_attach_mode:native
XDPAttachMode:native
📄📄 XDP 📄📄📄📄 "eth0"📄📄 2📄
```

📄📄 API 📄📄

```
📄📄📄
curl -s http://localhost:8080/api/v1/config | grep xdp_attach_mode

📄📄📄📄
"xdp_attach_mode": "native",
```

## 📄📄 Proxmox 📄📄

📄📄📄📄📄 XDP 📄📄

📄📄📄📄

- 📄📄📄📄📄📄📄`ethtool -l eth0`
- 📄📄📄📄📄`uname -r`📄📄  $\geq 5.15$
- 📄📄📄 VirtIO 📄📄📄`lsmod | grep virtio_net`

📄📄📄📄📄📄 1 📄📄

📄📄📄📄

- 📄📄📄📄 📄📄📄📄📄📄📄📄📄📄📄📄
- 📄📄 `qm shutdown XXX && sleep 5 && qm start XXX`
- 📄 Proxmox 📄📄📄📄`grep net0 /etc/pve/qemu-server/XXX.conf`

VMware ESXi / vSphere

VMware

- CPU 使用率を確認
- `top` - CPU 使用率を確認
- XDP 確認 `curl http://localhost:8080/api/v1/xdp_stats`

## VMware ESXi / vSphere XDP

### VMware ESXi / vSphere

VMware `vmxnet3` ネットワークアダプタ XDP

要件

- ESXi 6.7 以降
- `vmxnet3` ネットワークアダプタ 1.4.16 以降
- ESXi 6.7 以降 14 以降

手順

1. ネットワークアダプタを確認
2. ネットワークアダプタのバージョンを確認
  - ネットワークアダプタ → ネットワークアダプタ
  - ネットワークアダプタ → ネットワークアダプタ
  - ネットワークアダプタのバージョンを確認
3. ネットワークアダプタの `.vmx` ファイルを確認

```
ethernet0.pnicFeatures = "4"
ethernet0.multiqueue = "8"
```

4. ネットワークアダプタを確認

```
ethtool -l ens192 # 查看网卡
```

## OmniUPF

```
interface_name: [ens192] # VMware 网卡 ens192
xdp_attach_mode: native
```

## KVM / libvirt

### virsh 查看虚拟机

```
查看虚拟机
virsh edit your-vm-name
```

### 配置网络

```
<interface type='network'>
 <source network='default' />
 <model type='virtio' />
 <driver name='vhost' queues='8' />
</interface>
```

### 查看网卡

```
ethtool -l eth0
```

## Microsoft Hyper-V

Hyper-V 使用 `hv_netvsc` 驱动支持 XDP

### 支持

- Windows Server 2016 及以上
- Linux 内核 4.3 及以上

- 配置

配置

Hyper-V 配置 PowerShell

```
VMQ 配置 - Hyper-V 配置
Set-VMNetworkAdapter -VMName "YourVM" -VrssEnabled $true -
VmqEnabled $true
```

OmniUPF

```
interface_name: [eth0]
xdp_attach_mode: native
```

## VirtualBox

VirtualBox 配置 XDP

VirtualBox 配置 e1000 virtio-net XDP

配置

```
xdp_attach_mode: generic # VirtualBox 配置
```

## XDP

XDP 配置



## 1. 检查 OmniUPF 日志

```
检查日志
journalctl -u omniupf --since "5 minutes ago" | grep -i xdp

输出
✓ "xdp_attach_mode:native"
✓ "成功 XDP 初始化"
✗ "失败" 在 "失败"
```

## 2. 检查 API 配置

```
检查配置
curl -s http://localhost:8080/api/v1/config | jq .xdp_attach_mode

输出
"native"
```

## 3. 检查 XDP 统计

```
检查 XDP 统计
curl -s http://localhost:8080/api/v1/xdp_stats | jq

输出
{
 "xdp_aborted": 0, # 成功 0 次
 "xdp_drop": 1234, # 失败
 "xdp_pass": 5678, # 成功
 "xdp_redirect": 9012, # 成功
 "xdp_tx": 3456 # 成功
}
```

## 4. 確認

```
確認XDP
ethtool -i eth0 | grep driver

Proxmox/KVM "virtio_net"
VMware "vmxnet3"
Hyper-V "hv_netvsc"
```

## 5. 確認

確認

```
確認
watch -n 1 'curl -s http://localhost:8080/api/v1/packet_stats | jq .rx_packets'

~1-2 Mpps
~5-10 Mpps 5-10
```

## XDP

"XDP"

```
XDP eth0
```

1.

```
ethtool -i eth0 | grep driver
```

```
確認 virtio_net/vmxnet3/hv_netvsc 対応 XDP 対応
```

## 2. 確認

```
uname -r
```

```
5.15 以上 対応 XDP
```

## 3. 確認 XDP 対応

```
ip link show eth0 | grep xdp
```

```
対応 XDP 対応
```

```
ip link set dev eth0 xdp off
```

確認

- 5.15+
- virtio\_net 対応 `modprobe virtio_net`
- XDP 対応

確認

確認

```
確認 XDP 対応
```

確認

```
確認 dmesg 確認
```

```
dmesg | grep -i xdp | tail -20
```

環境構築

1. 仮想マシンを XDP

- VirtualBox をインストール XDP
- 仮想 NIC を設定

2. ネットワーク

- 仮想 `ethtool -l eth0`
- 仮想 > 1 を設定

3. 仮想 XDP をインストール

```
仮想マシン XDP
grep XDP /boot/config-$(uname -r)

仮想
CONFIG_XDP_SOCKETS=y
CONFIG_BPF=y
```

環境構築

- 仮想マシン Proxmox をインストール
- 仮想マシン
- 仮想マシン XDP

仮想マシンをインストール

仮想マシンをインストール

仮想

1. 仮想マシン

```
查看接收队列
ethtool -S eth0 | grep rx_queue

查看接收队列大小
```

## 2. 查看 CPU 使用情况

```
查看系统 CPU 使用率
mpstat -P ALL 1

查看指定 CPU 的使用率
```

## 3. 查看 XDP 配置和统计

```
查看 bpf 程序
sudo bpftool net list

查看 XDP 统计
```

注意事项

- 确保内核版本 8-16 及以上
- 查看 CPU 使用率是否过高
- 确保系统有足够的 CPU 资源

---

## 查看 XDP 统计值 `xdp_aborted > 0`

示例

```
curl http://localhost:8080/api/v1/xdp_stats
{
 "xdp_aborted": 1234, # 统计值
 ...
}
```

总结

## XDP 和 eBPF 的入门教程

### 1. 安装 eBPF 依赖

```
dmesg | grep -i bpf | tail -20
```

### 2. 验证安装

```
eBPF 测试
curl http://localhost:8080/api/v1/map_info

测试 100% 成功
```

总结

- 了解 eBPF 的基本概念
- 掌握 eBPF 的部署和验证方法
- 了解 Linux 内核 eBPF 的架构

---

## Proxmox 网络配置

使用 `ethtool -l eth0` 查看 1 号网卡的配置

配置

### 1. Proxmox 配置

```
Proxmox 配置
grep net0 /etc/pve/qemu-server/YOUR_VM_ID.conf

queues=8
```

### 2. 验证配置

```
在 Proxmox 中
qm status YOUR_VM_ID

等待 "status: stopped" 出现
```

等待

```
在 Proxmox 中
等待 VM 停止
qm shutdown YOUR_VM_ID
sleep 10
qm start YOUR_VM_ID

检查网络接口
ethtool -l eth0
```

等待 VM 启动并检查网络接口

## 配置 XDP 规则

配置

配置 XDP 规则

配置

XDP 规则 CAP\_NET\_ADMIN 和 CAP\_SYS\_ADMIN 权限

配置

1. 以 **root** 身份运行 **OmniUPF** 配置

```
sudo systemctl restart omniupf
```

2. 配置 **systemd** 服务

```
/lib/systemd/system/omniupf.service
[Service]
CapabilityBoundingSet=CAP_NET_ADMIN CAP_SYS_ADMIN CAP_NET_RAW
AmbientCapabilities=CAP_NET_ADMIN CAP_SYS_ADMIN CAP_NET_RAW
```

3. `docker` `--privileged`

```
docker run --privileged -v /sys/fs/bpf:/sys/fs/bpf ...
```

## Table

OmniUPF

|            | 1.5 Mpps   | 8.2 Mpps   | 5.5 |
|------------|------------|------------|-----|
|            | 95 $\mu$ s | 12 $\mu$ s | 8   |
| CPU 1 Gbps | 85% 1      | 15%        | 5   |
|            | ~1.2 Gbps  | ~10 Gbps   | 8   |

## XDP

Omnitouch 100%

## XDP NIC

NIC OmniUPF XDP



Intel NIC

| Model      | Speed    | Driver | XDP Support        | Notes           |
|------------|----------|--------|--------------------|-----------------|
| Intel X520 | 10GbE    | ixgbe  | Yes                | Supports SR-IOV |
| Intel X710 | 10/40GbE | i40e   | Yes                | Supports SR-IOV |
| Intel E810 | 100GbE   | ice    | Yes                | Supports SR-IOV |
| Intel i350 | 1GbE     | igb    | Yes (Kernel 5.10+) | Supports SR-IOV |

Intel/NVIDIA NIC

| Model       | Speed         | Driver | XDP Support | Notes                     |
|-------------|---------------|--------|-------------|---------------------------|
| ConnectX-4  | 25/50/100GbE  | mlx5   | Yes         | Supports SR-IOV           |
| ConnectX-5  | 25/50/100GbE  | mlx5   | Yes         | Supports SR-IOV           |
| ConnectX-6  | 50/100/200GbE | mlx5   | Yes         | Supports SR-IOV           |
| BlueField-2 | 100/200GbE    | mlx5   | Yes         | Supports DPU and SmartNIC |

Intel NIC

| Model    | Speed       | Driver  | XDP Support | Notes           |
|----------|-------------|---------|-------------|-----------------|
| BCM57xxx | 10/25/50GbE | bnxt_en | Yes         | Supports SR-IOV |

Intel NIC

| OS          | NIC Driver | Driver     | XDP Support | Notes         | OS                   |
|-------------|------------|------------|-------------|---------------|----------------------|
| Proxmox/KVM | VirtIO     | virtio_net | Yes ✓       | Kernel module | Linux                |
| VMware ESXi | vmxnet3    | vmxnet3    | Yes ✓       | None          | Linux ESXi 6.7+      |
| Hyper-V     | VMNIC      | hv_netvsc  | Yes ✓       | None          | Windows Server 2016+ |
| AWS         | ENA        | ena        | Yes ✓       | None          | EC2 Linux            |
| VirtualBox  | VMNIC      | VMNIC      | Yes ✓       | None          | Linux                |

## Network Interface Card (NIC)

Supports XDP, eBPF, and NIC offload

| Model     | Port           | Speed  | Features                           |
|-----------|----------------|--------|------------------------------------|
| Netronome | Agilio CX 10G  | 10GbE  | Supports XDP, eBPF                 |
| Netronome | Agilio CX 25G  | 25GbE  | Supports XDP, eBPF                 |
| Netronome | Agilio CX 40G  | 40GbE  | Supports XDP, eBPF, ~\$2,500-5,000 |
| Netronome | Agilio CX 100G | 100GbE | Supports XDP, eBPF                 |

Netronome NIC supports XDP, eBPF, and NIC offload

## Performance

Netronome OmniUPF supports

Supports 1-10 Gbps

- **NIC** Intel X520 10GbE 40nm
- 支持 XDP
- 支持 UPF 约 ~8-10 Gbps
- 约 \$100-200/卡

## 10-50 Gbps

- **NIC** Intel X710 40GbE / Mellanox ConnectX-4 25GbE
- 支持 XDP
- 支持 UPF 约 ~25-40 Gbps
- 约 \$300-800

## 50-100+ Gbps

- **NIC** Mellanox ConnectX-5/6 100GbE
- 支持 XDP
- 支持 UPF 约 ~80-100 Gbps
- 约 \$1,000-2,500

## Proxmox/KVM

- **NIC** VirtIO 8-16 核
- 支持 XDP
- 支持 UPF 约 ~5-10 Gbps
- 支持 SR-IOV

## 总结

支持 OmniUPF 支持

| NIC/网卡        | 网卡                 | 网卡                |
|---------------|--------------------|-------------------|
| Realtek NICs  | 不支持 XDP 的 Linux 内核 | Intel i350 网卡     |
| VirtualBox    | 不支持 XDP 的          | 不支持 Proxmox/KVM   |
| 网卡 NICs       | 网卡                 | 网卡 Intel/Mellanox |
| 网卡 NICs <2014 | 不支持 XDP 的          | Intel X520 网卡     |

## 网卡

### 网卡

#### 1. 网卡 Linux 内核 XDP

```
网卡
modinfo <driver_name> | grep -i xdp
```

#### 2. 网卡 ≥ 5.15 内核 XDP

```
uname -r
```

#### 3. 网卡 NIC 网卡RSS/VMDq

#### 4. 网卡 PCI 网卡 PCIe 网卡

- 10GbE PCIe 2.0 x4 网卡
- 40GbE PCIe 3.0 x8 网卡
- 100GbE PCIe 3.0 x16 或 PCIe 4.0 x8

#### 5. 网卡

- 网卡 NIC
- 网卡 VirtIO 或 SR-IOV 网卡
- 网卡 NIC 网卡

## 目次

- [CONFIGURATION.md](#) - 基本設定
  - [TROUBLESHOOTING.md](#) - トラブルシューティング
  - [ARCHITECTURE.md](#) - eBPF と XDP の仕組み
  - [MONITORING.md](#) - 監視とログ
- 

## インストール

### Proxmox 上で XDP を実行する

```
Proxmox 上で実行
qm set <VM_ID> -net0 virtio=<MAC>,bridge=vbr0,queues=8
qm shutdown <VM_ID> && sleep 10 && qm start <VM_ID>

設定
ethtool -l eth0 # 8 個の queue
sudo nano /etc/omniupf/config.yaml # xdp_attach_mode: native
sudo systemctl restart omniupf
journalctl -u omniupf --since "1 min ago" | grep xdp # ログを確認
```

### XDP の実行を確認する

```
確認
curl -s http://localhost:8080/api/v1/config | grep xdp_attach_mode

統計
curl -s http://localhost:8080/api/v1/xdp_stats | jq

確認
ethtool -l eth0
```

# OmniUPF 開箱

## 簡介

1. 簡介
2. 支援 5G 網路功能
3. UPF 功能
4. PCF 支援 SMF 功能
5. 網路功能
6. 網路功能
7. 網路功能
8. 網路功能

## 安裝

OmniUPF 支援 eBPF 網路功能，支援 5G/LTE 網路功能，支援 QoS 網路功能，支援 Linux eBPF 網路功能，支援 OmniUPF 支援 5G SA/5G NSA 及 LTE 網路功能。

## 網路功能

OmniUPF 支援 3GPP 網路功能，支援 5G 及 LTE 網路功能。

- 支援 3GPP 網路功能
- 支援 QoS 網路功能
- 支援 3GPP 網路功能
- 支援 3GPP 網路功能
- 支援 3GPP 網路功能
- 支援 3GPP 網路功能

OmniUPF 支援 3GPP TS 23.501 5G 及 TS 23.401 LTE 網路功能，UPF 支援 Linux 及 eBPF 網路功能。

# OmniUPF 特性

## 特徴

- 3GPP 準拠
- eBPF による柔軟な制御
- GTP-U による柔軟な制御
- IPv4 と IPv6 のサポート
- XDP による柔軟な制御
- 柔軟な制御

## QoS 機能

- QoS 機能 (QER)
- PDR
- FAR
- SDF
- URR

## 特徴

- PCF による柔軟な制御 (SMF/PGW-C)
- RESTful API
- 柔軟な制御
- eBPF による柔軟な制御
- Web による柔軟な制御

## 特徴

- eBPF による柔軟な制御
- 柔軟な制御
- 柔軟な制御
- 柔軟な制御
- 柔軟な制御

柔軟な制御 Web UI による

## OmniUPF

OmniUPF 5G SA 5G NSA 4G LTE/EPC

OmniUPF

- **UPF** - 5G/NSA N4/PFCP OmniSMF
- **PGW-U** **PDN** - 4G EPC Sxc/PFCP OmniPGW-C
- **SGW-U** - 4G EPC Sxb/PFCP OmniSGW-C

OmniUPF

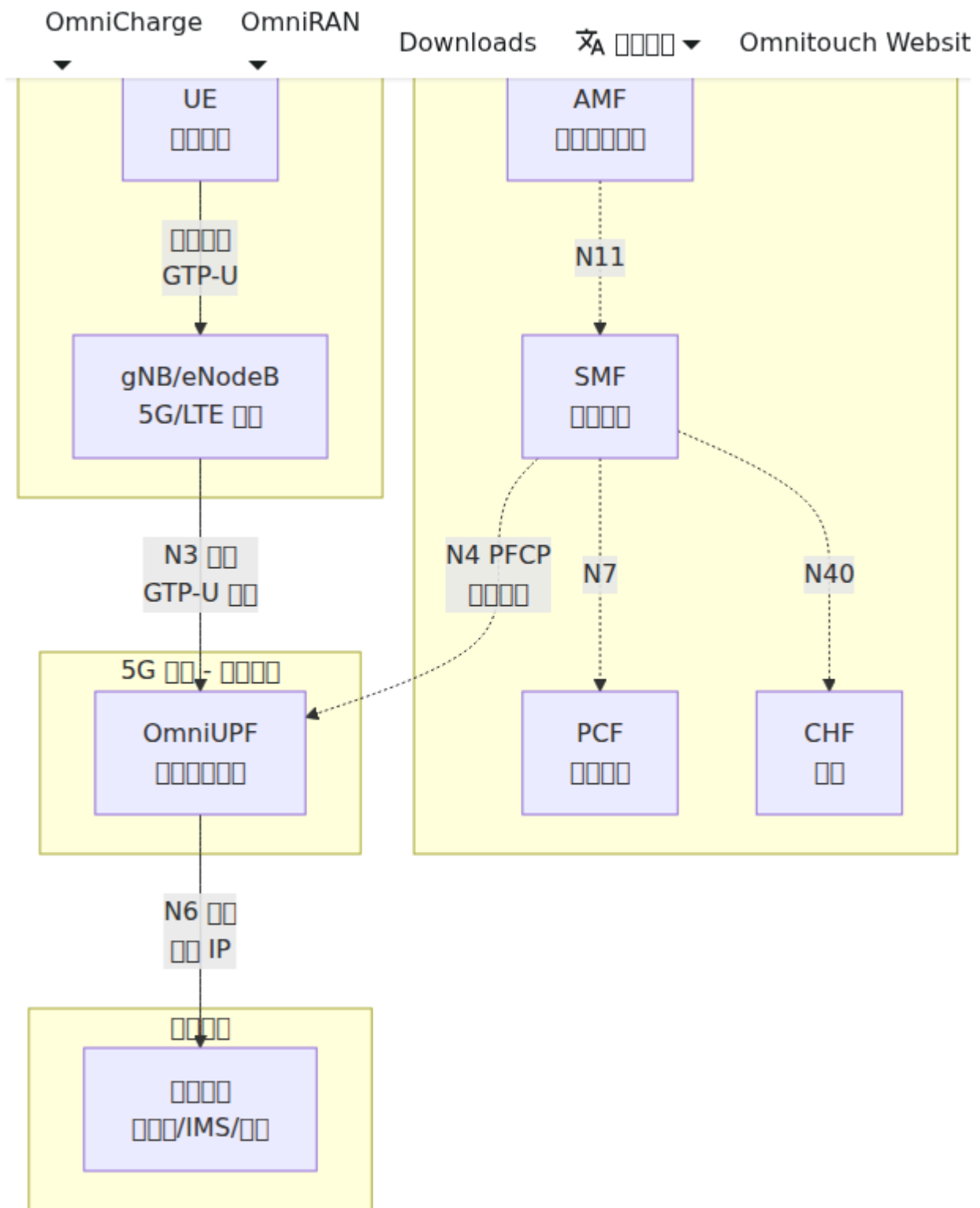
- **UPF** 5G
- **PGW-U + SGW-U** 4G EPC
- **UPF + PGW-U + SGW-U** 4G 5G

eBPF PFCP UPF PGW-U SGW-U

## 5G SA

OmniUPF 5G



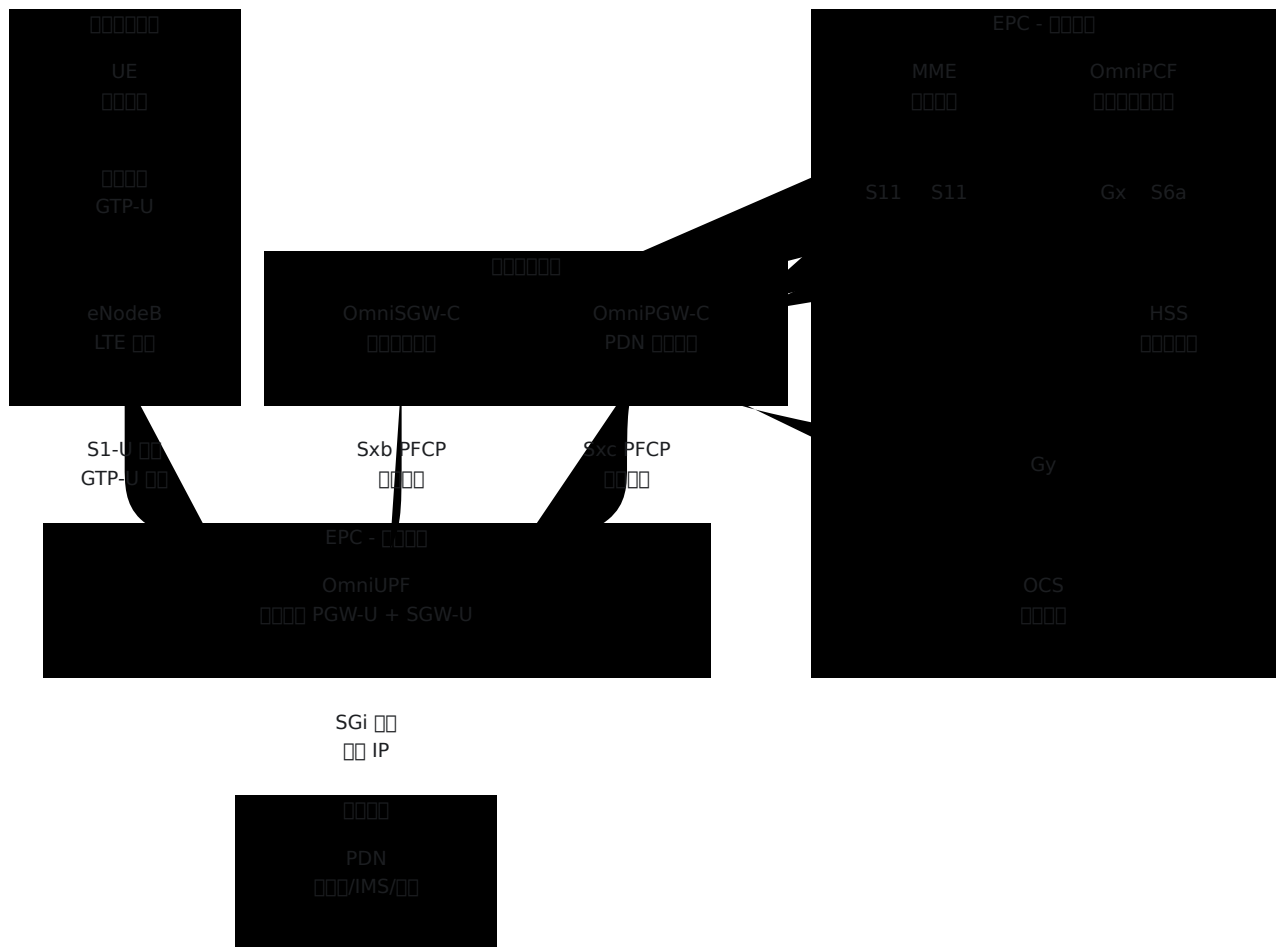


## 4G LTE/EPC

OmniUPF 4G LTE EPC OmniPGW-U OmniSGW-U

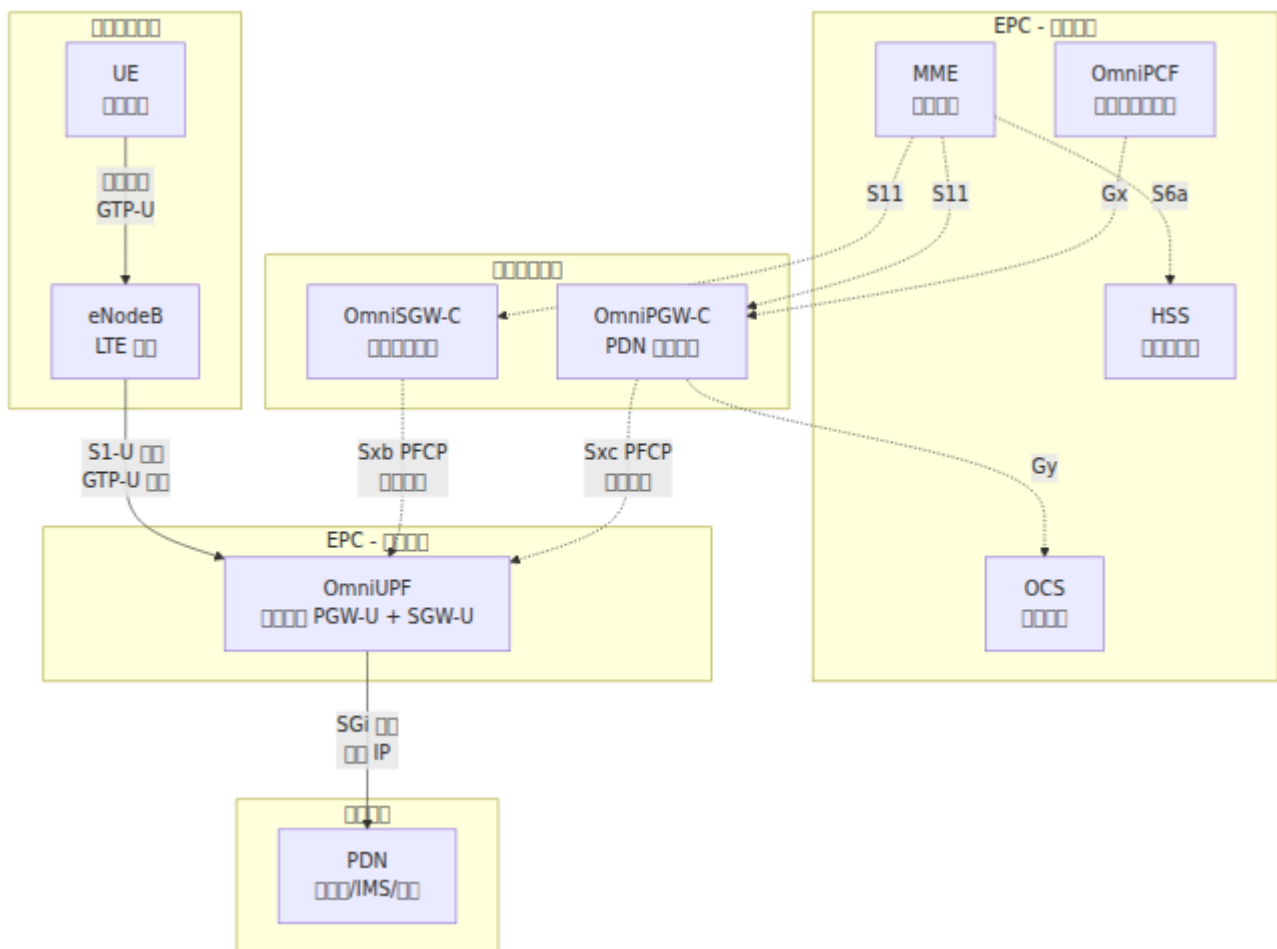
## PGW-U/SGW-U 4G

OmniUPF SGW-U PGW-U



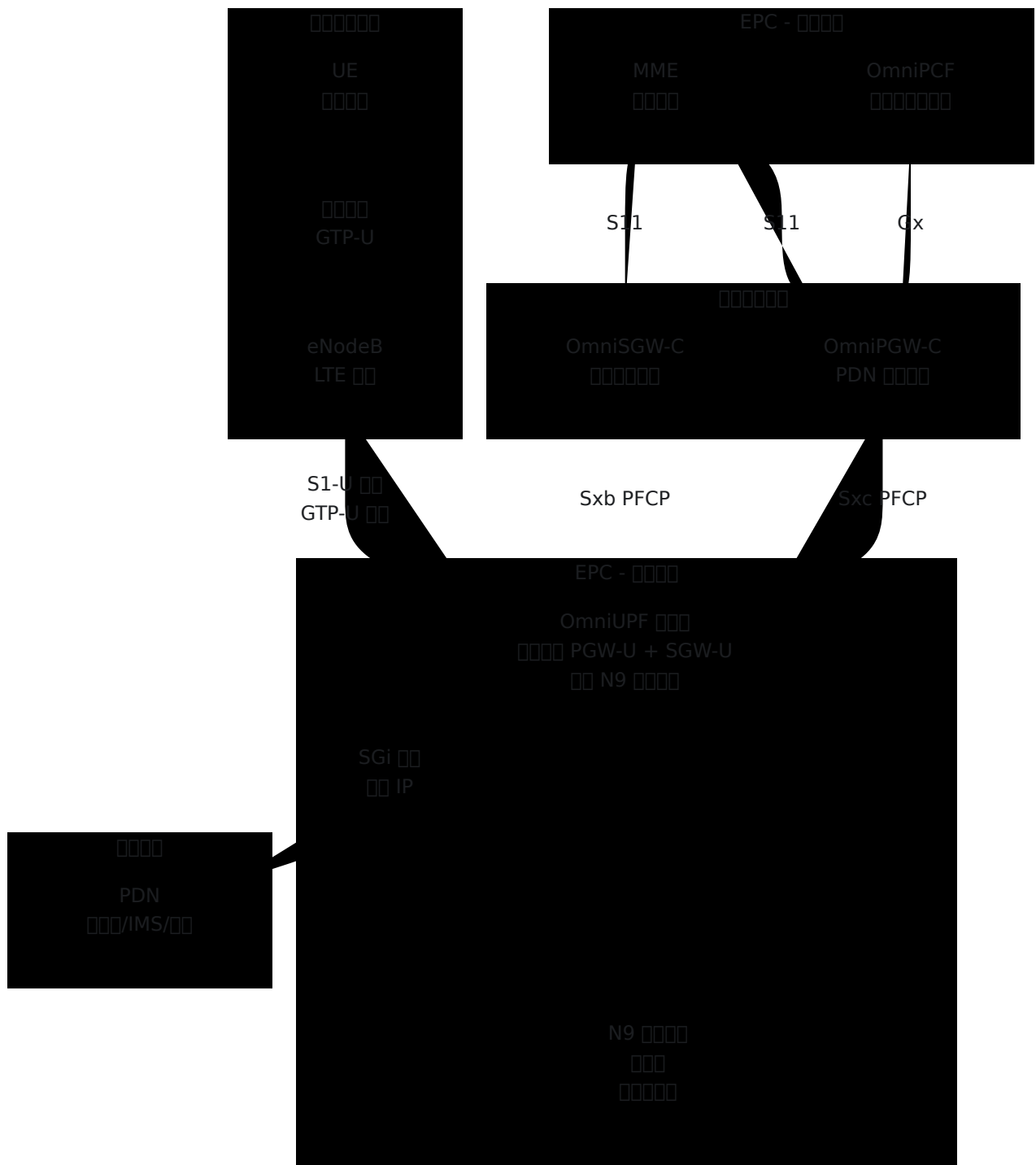
## SGW-U PGW-U

OmniUPF - SGW-U PGW-U



## N9 SGWU+PGWU

OmniUPF SGWU PGWU eBPF N9



## Network

- **N9** - eBPF
- **40-50% CPU** - XDP
- -
- - `n3_address = n9_address`
- **3GPP** - PFCP & GTP-U

## Network

```
OmniUPF config.yml
interface_name: [eth0]
n3_address: "10.0.1.10"
n9_address: "10.0.1.10"
pfcf_address: ":8805"
S1-U IP
IP N9
SGWU-C PGWU-C
```

- 
- 
- 
- 

- 
- 
- 

1.

- 
- 
- 
- 
- 

2.

-

- **4G** ネットワーク S1-U ネットワーク SGW-U ネットワーク S5/S8 ネットワーク PGW-U ネットワーク eNodeB ネットワーク GTP-U ネットワーク
- ネットワーク TEID ネットワーク PDR
- eBPF ネットワーク QER ネットワーク
- FAR ネットワーク
- ネットワーク GTP-U ネットワーク N6 ネットワーク 5G ネットワーク SGi ネットワーク 4G ネットワーク
- URR ネットワーク

### 3. ネットワーク → UE

- **5G** ネットワーク N6 ネットワーク IP ネットワーク
- **4G** ネットワーク SGi ネットワーク IP ネットワーク
- ネットワーク UE IP ネットワーク PDR
- SDF ネットワーク
- FAR ネットワーク GTP-U ネットワーク
- ネットワーク TEID ネットワーク GTP-U ネットワーク
- **5G** ネットワーク N3 ネットワーク gNB
- **4G** ネットワーク S1-U ネットワーク SGW-U ネットワーク S5/S8 ネットワーク PGW-U ネットワーク eNodeB

### 4. ネットワーク

- **5G** ネットワーク OmniSMF ネットワーク PDR/FAR ネットワーク
- **4G** ネットワーク OmniSGW-C/OmniPGW-C ネットワーク eNodeB ネットワーク TAU ネットワーク
- ネットワーク
- ネットワーク

## 4G と 5G

OmniUPF ネットワーク 3GPP ネットワーク 5G ネットワーク 4G ネットワーク

### 5G ネットワーク

| 面         | 面 → 面             | 面            | 3GPP 面    |
|-----------|-------------------|--------------|-----------|
| <b>N4</b> | OmniSMF ↔ OmniUPF | PFCP 面       | TS 29.244 |
| <b>N3</b> | gNB → OmniUPF     | 面 RAN 面GTP-U | TS 29.281 |
| <b>N6</b> | OmniUPF → 面       | 面 DN 面 IP    | TS 23.501 |
| <b>N9</b> | OmniUPF ↔ OmniUPF | 面/面 UPF 面    | TS 23.501 |

#### 4G/EPC 面

| 面            | 面 → 面                           | 面            | 3GPP 面    |
|--------------|---------------------------------|--------------|-----------|
| <b>Sxb</b>   | OmniSGW-C ↔ OmniUPF面SGW-U 面     | 面 PFCP 面     | TS 29.244 |
| <b>Sxc</b>   | OmniPGW-C ↔ OmniUPF面PGW-U 面     | PDN 面 PFCP 面 | TS 29.244 |
| <b>S1-U</b>  | eNodeB → OmniUPF面SGW-U 面        | 面 RAN 面GTP-U | TS 29.281 |
| <b>S5/S8</b> | OmniUPF面SGW-U面 ↔ OmniUPF面PGW-U面 | 面GTP-U       | TS 29.281 |
| <b>SGi</b>   | OmniUPF面PGW-U 面 → PDN           | 面 IP         | TS 23.401 |

面 PFCP 面N4面Sxb面Sxc面 TS 29.244 面 PFCP 面

# UPF

## eBPF

eBPF Linux

- **GTP-U** GTP-U
- TEID UE IP SDF
- **QoS** QER
- FAR
- URR

eBPF eBPF

| uplink_pdr_map       | PDR      | TEID32  | PDR FAR ID QER ID URR IDs |
|----------------------|----------|---------|---------------------------|
| downlink_pdr_map     | PDR IPv4 | UE IP   | PDR                       |
| downlink_pdr_map_ip6 | PDR IPv6 | UE IPv6 | PDR                       |
| far_map              |          | FAR ID  |                           |
| qer_map              | QoS      | QER ID  | QoS MBR GBR               |
| urr_map              |          | URR ID  |                           |
| sdf_filter_map       | SDF      | PDR ID  |                           |



- 可移植なフレームワーク
- **XDP** による高性能なパケット処理
- 高効率 CPU 利用による CPU 削減
- 可移植 eBPF による PDR/FAR 処理

可移植なフレームワーク

## PFCP 概要

**PFCP** は 3GPP TS 29.244 に定義された SMF と PGW-C 間のプロトコル

概要

- 可移植な PFCP 処理の実装
- 可移植なパケット処理による PFCP 処理
- 可移植な PFCP 処理による eBPF 処理
- 可移植な SMF 処理の実装

可移植 **PFCP** 処理

| メッセージ      | 方向        | 内容                              |
|------------|-----------|---------------------------------|
| Request    | SMF → UPF | PFCP 処理の開始                      |
| Response   | SMF → UPF | PFCP 処理の完了                      |
| Keep-Alive | UPF → SMF | Keep-Alive 処理                   |
| Report     | SMF → UPF | Report PDU による PDR/FAR/QRER/URR |
| Report     | SMF → UPF | Report QoS による QoS              |
| Report     | SMF → UPF | Report による Report               |
| Report     | UPF → SMF | Report による Report               |

□□□□□□□**IE**□□

- PDR FAR QER URR
- PDR FAR QER URR
- PDR FAR QER URR
- UE IP F-TEID SDF
- 
- QoS MBR GBR QFI
- 

## REST API ☐☐☐

## REST API ☐ UPF ☐

□ □ □ □ □

- 内核态用户态 PFCP 交互
- 内核态用户态 PDR/FAR/QER/URR 交互
- 内核态用户态 XDP 交互
- 内核态用户态 其他
- 内核态用户态 eBPF 交互

**API** □□□□ 34 □□□□

| API  | API                                                                            | API       |
|------|--------------------------------------------------------------------------------|-----------|
| API  | /health                                                                        | API       |
| API  | /config                                                                        | UPF API   |
| API  | /pfcpsessions, /pfcpassociations                                               | PFCP API/ |
| PDRs | /uplink_pdr_map, /downlink_pdr_map, /downlink_pdr_map_ip6, /uplink_pdr_map_ip6 | API       |
| FARs | /far_map                                                                       | API       |
| QERs | /qer_map                                                                       | QoS API   |
| URRs | /urr_map                                                                       | API       |
| API  | /buffer                                                                        | API       |
| API  | /packet_stats, /route_stats, /xdp_stats, /n3n6_stats                           | API       |
| API  | /map_info                                                                      | eBPF API  |
| API  | /dataplane_config                                                              | N3/N9 API |

API API

## Web

Web UPF

□□□

- □□□□□□□□ PFCP □□□□ UE IP□TEID □□□□□
- □□□□□□□□□□□□□□ PDR□FAR□QER □ URR
- □□□□□□□□□□□□□□ FAR □□□□
- □□□□□□□□□□□□□□XDP □ N3/N6 □□□□
- □□□□□eBPF □□□□□□□□□□□□□□□□□□□□
- □□□□□□□ UPF □□□□□□□□□□
- □□□□□□□□□□□□□□□□□□□□□□

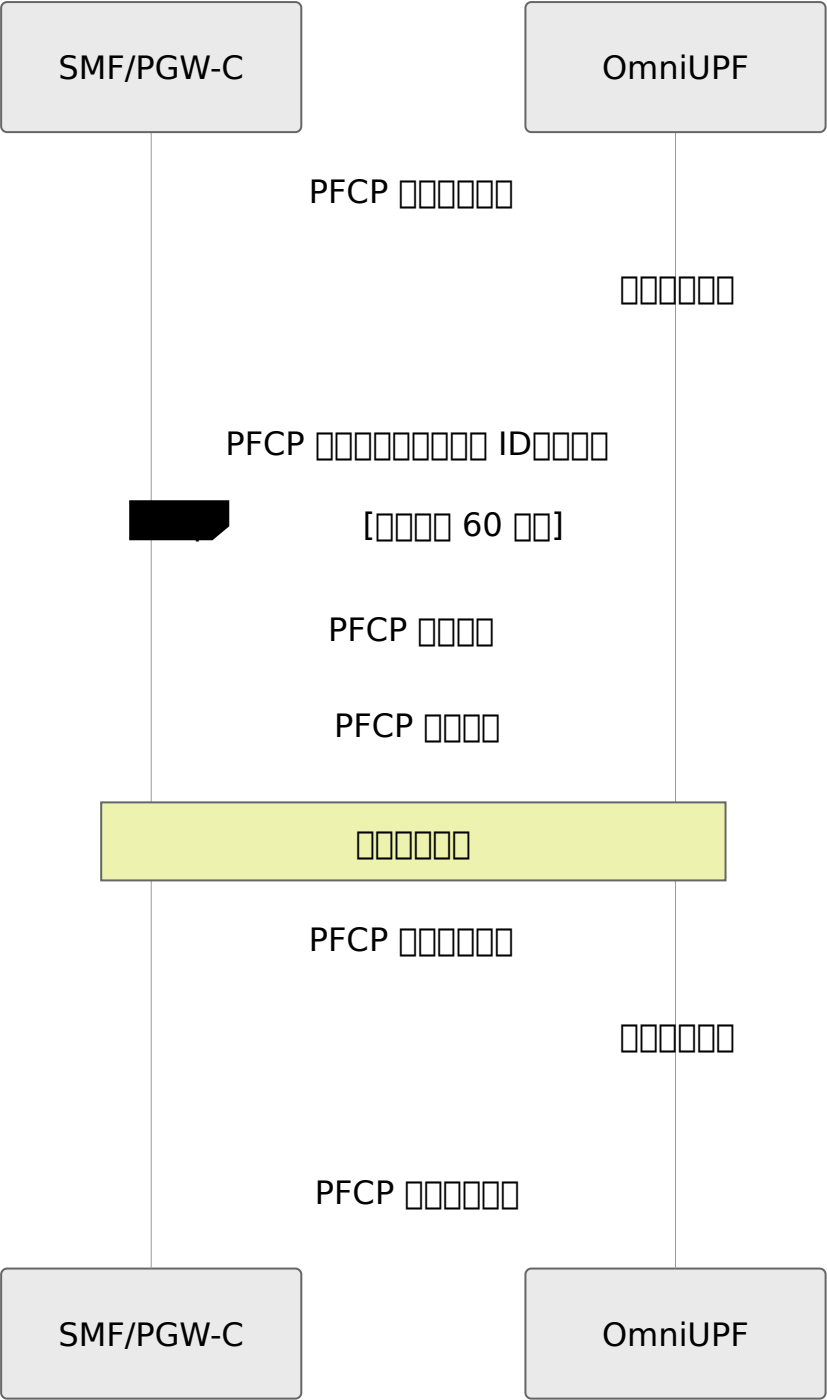
□□□□□ UI □□□□□ **Web UI** □□□□□

# PFCP □□□ SMF □□

## PFCP □□

□□□□□□□□SMF □□□ UPF □□ PFCP □□□

□□□□□□□□



SMF

- SMF and UPF interaction
- UPF ID, FQDN, IP address
- N4 interface
- PFCP messages

SMF and UPF interaction

# SMF 功能

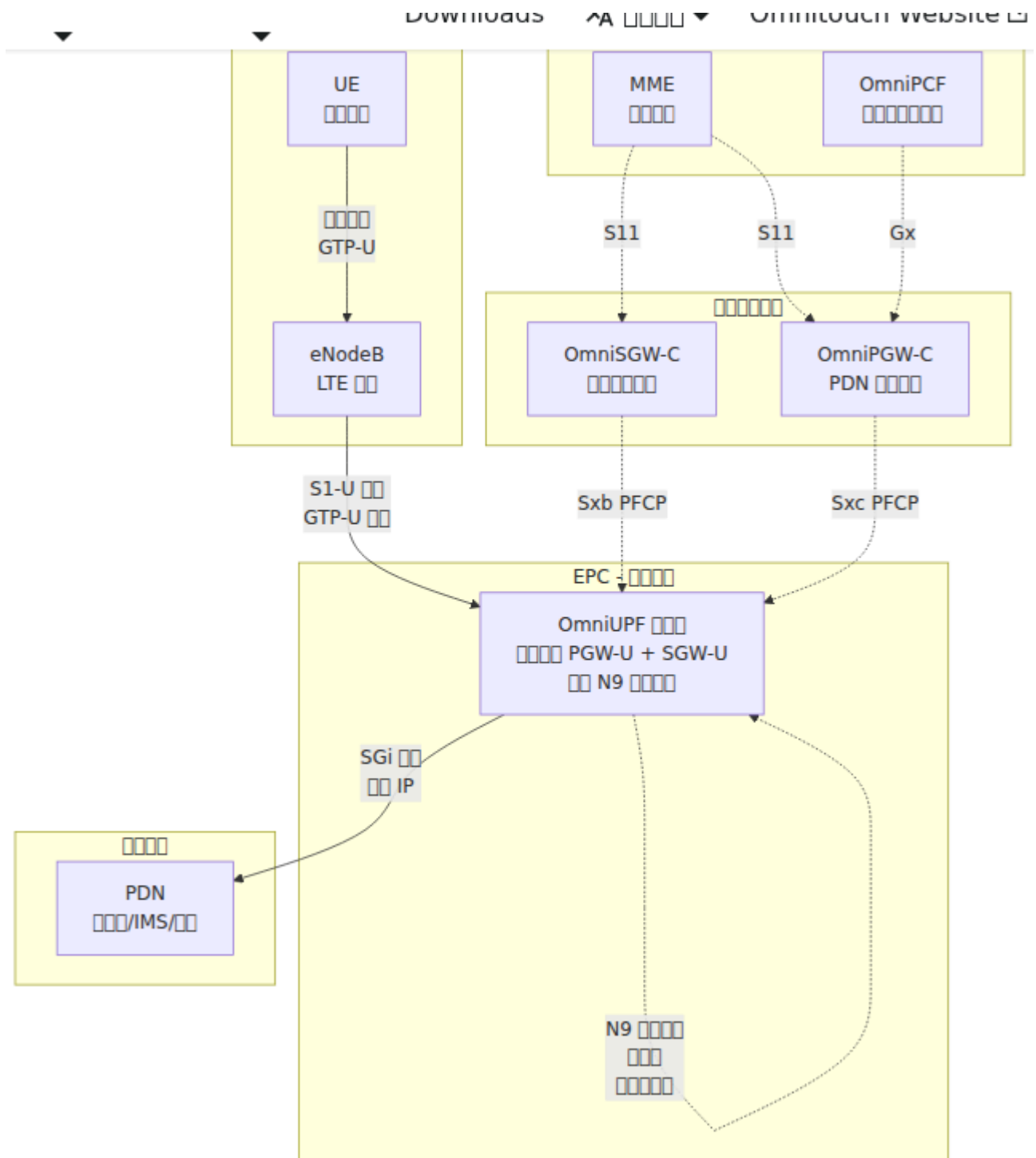
OmniUPF 实现 SMF 功能 3GPP TS 29.244 定义

功能

SMF 和 PCFP 功能 OmniUPF 实现 SMF 功能

1. SMF 功能
2. SMF 和 UPF 和 PCFP 功能
3. SMF 功能
4. UPF 功能 = SMF 功能
5. UPF 功能 SMF 功能
6. SMF 功能

功能



□□□□□

□ SMF □□□□□□□□□□

```
WARN: NodeID: smf-1 : 192.168.1.10
WARN: SMF 2025-01-15T10:00:00Z2025-01-15T10:30:15Z
- SMF 245
INFO: 2LocalSEID SMF
INFO: 3LocalSEID SMF
...
INFO: 246LocalSEID SMF
```

1. SMF SMF
2. SMF
3. **3GPP** 3GPP TS 29.244 5.22.2

“CP UP CP CP PFCP”

## GTP-U

OmniUPF 3GPP TS 29.281 PGW-U SGW-U eNodeB gNodeB  
GTP-U

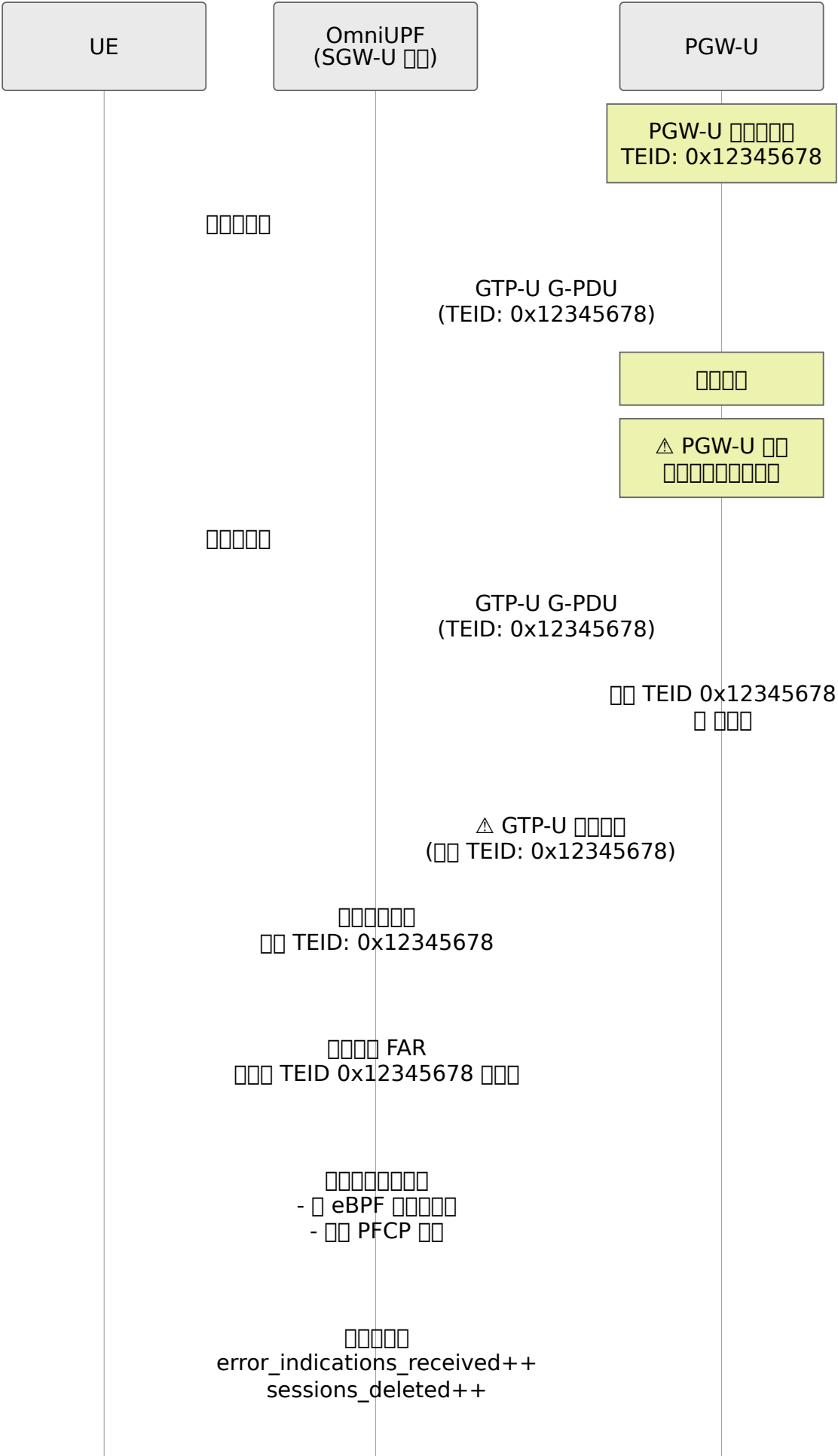
OmniUPF GTP-U SGW-U PGW-U TEID

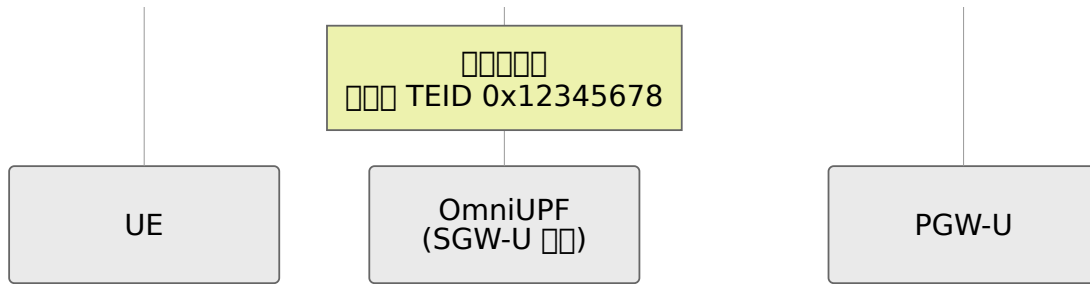
- 
- 
- 

1. **UPF** → TEID X GTP-U 2152



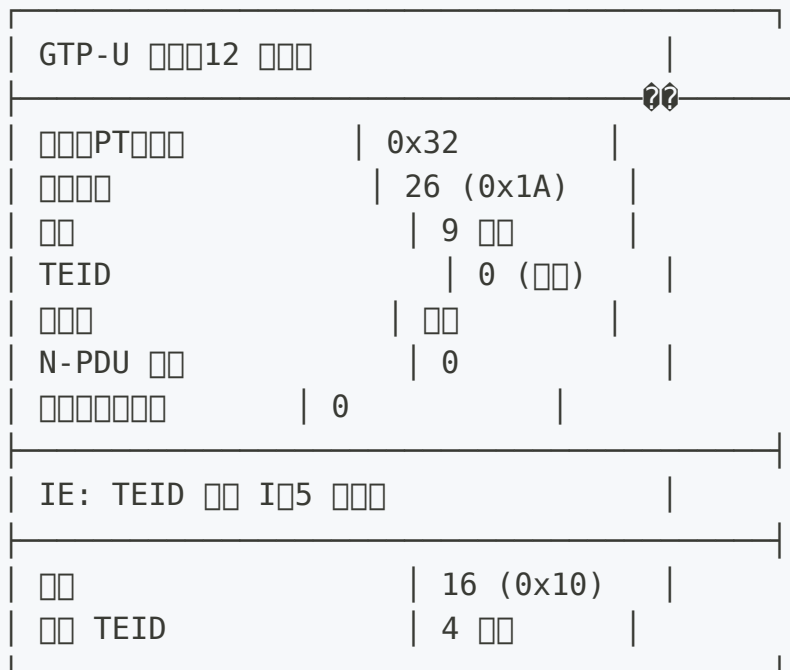
2. **TEID X** → **TEID**
3. → GTP-U 26 **TEID** IE
4. **UPF** → **TEID X**
5. **UPF** → **TEID X** FAR
6. **UPF** → eBPF PFCP
7. **UPF** → Prometheus





3GPP TS 29.281 7.3.1

GTP-U



1 PGW-U S5/S8 GTP

- SGW-U OmniUPF S5/S8 PGW-U
- PGW-U S5/S8
- SGW-U TEID
- PGW-U
- SGW-U

2 UPF N9

- UPF-1 OmniUPF N9 UPF-2

- UPF-2 配置
- UPF-1 配置
- UPF-1 配置

配置

配置

```
WARN: [192.168.50.10:2152] GTP-U [TEID 0x12345678 - [TEID
[TEID
WARN: [LocalSEID=42][FAR GlobalId=1 [TEID 0x12345678][
192.168.50.10
INFO: [LocalSEID=42][192.168.50.10 [TEID 0x12345678 [GTP-
U [
WARN: [1 [192.168.50.10 [TEID 0x12345678 [GTP-U [
```

## Prometheus 配置

配置

```
[
upf_buffer_listener_error_indications_received_total{node_id="pgw-u-
1",peer_address="192.168.50.10"}

[
upf_buffer_listener_error_indication_sessions_deleted_total{node_id='
u-1',peer_address="192.168.50.10"}

[TEID
upf_buffer_listener_error_indications_sent_total{node_id="enodeb-
1",peer_address="10.60.0.1"}
```

配置

- `node_id` 配置 PFCP 节点 ID 配置“”
- `peer_address` 配置 IP 地址

配置

□□□□□

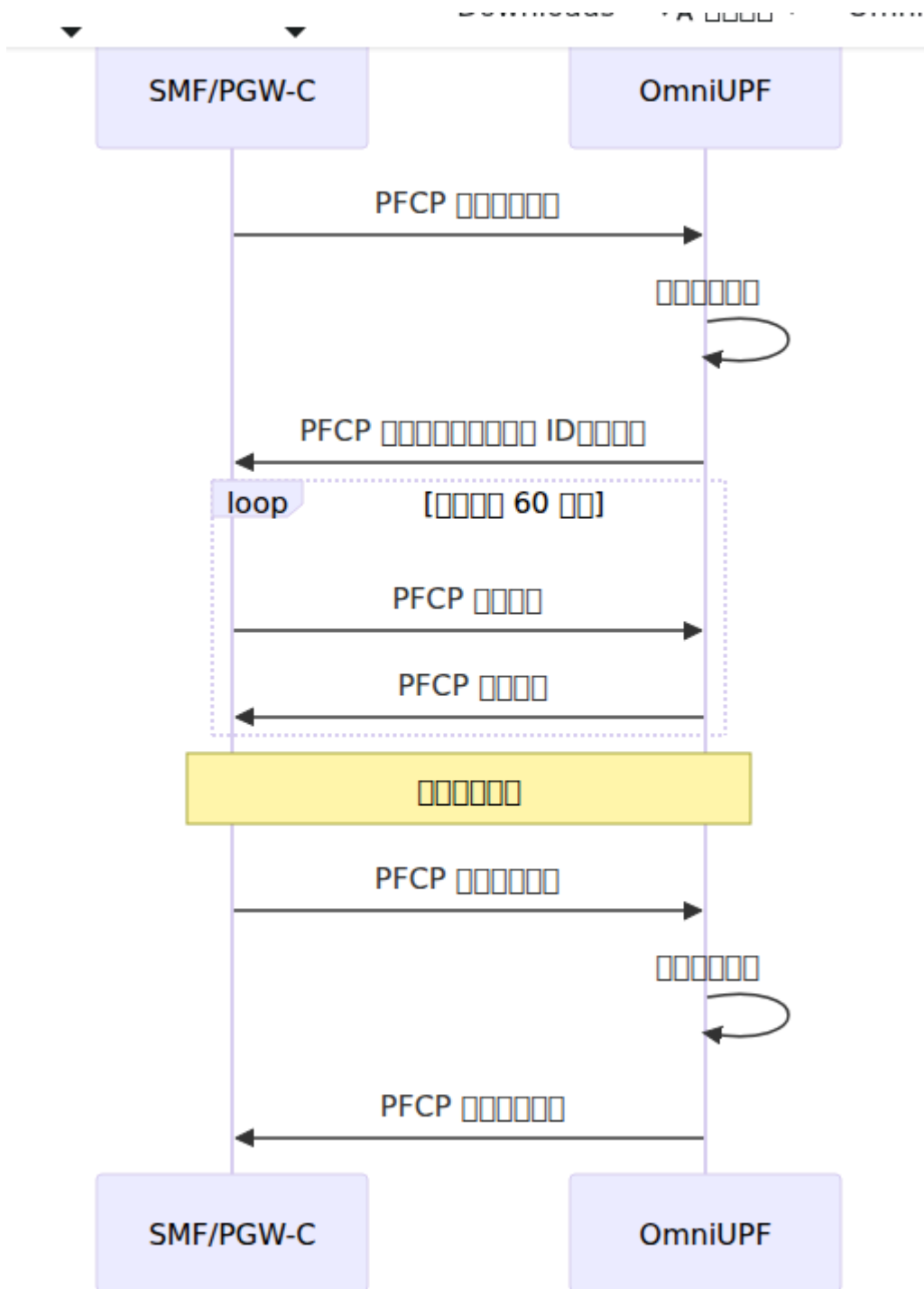
1.  $\text{TEID} \oplus \text{TEID} = 0$
2. **TEID**  $\oplus \text{TEID} = 0$
3.  $\text{TEID} \oplus \text{TEID} = 0$
4.  $\text{TEID} \oplus \text{TEID} = 0$
5.  $\text{TEID} \oplus \text{TEID} = 0$ 
  - $\text{TEID} \oplus \text{TEID} = 0$
  - $\text{TEID} \oplus \text{TEID} = 0$
6.  $\text{TEID} \oplus \text{TEID} = 0$

□□□□□□□□□□□□□□□□ GTP-U □□□□□□□

**PFCP** ☐ ☐ ☐ ☐

UE PDU 5G PDP LTE SMF UPF PCF

□□□□□□



PFCP session establishment

- **PDR** (Policy Data Rule) N3 TEID (Tunnel Endpoint ID) FAR (Forwarding Action Rule) N6
- **PDR** (Policy Data Rule) UE IP (User Equipment IP) FAR (Forwarding Action Rule) N3 GTP-U (GPRS Tunneling Protocol - User Plane)
- **FAR** (Forwarding Action Rule) (Forwarding Action Rule)
- **QER** (QoS Enforcement Rule) QoS (Quality of Service) MBR (Maximum Bit Rate) GBR (Guaranteed Bit Rate) QFI (QoS Flow Identifier)
- **URR** (Usage Reporting Rule) (Usage Reporting Rule)

# PFCP 訊息

SMF 透過 PFCP 與 UPF 交換 QoS 相關訊息

訊息流程

## 1. 透過 N2 介面

- SMF 透過 gNB 向 UPF 發送 F-TEID 及 FAR
- UPF 向 SMF 發送 PDR
- SMF 向 UPF 發送 SDF 及 QoS

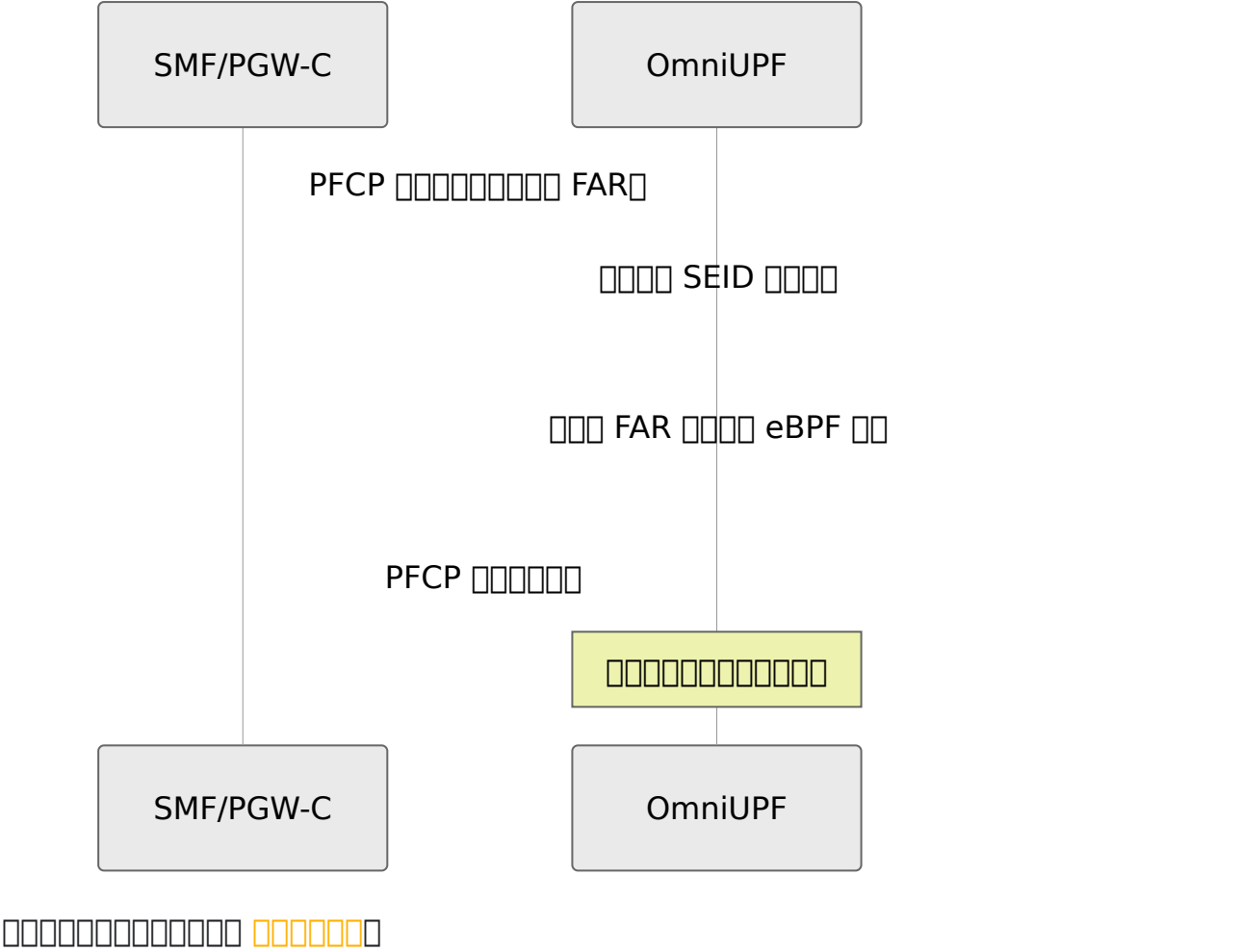
## 2. QoS 設定

- SMF 向 UPF 發送 MBR/GBR 及 QER
- UPF 向 SMF 發送 PDR 及 SDF 以設定 QoS

## 3. 訊息

- SMF 向 UPF 發送 PDR
- UPF 向 SMF 發送 FAR 及 QoS

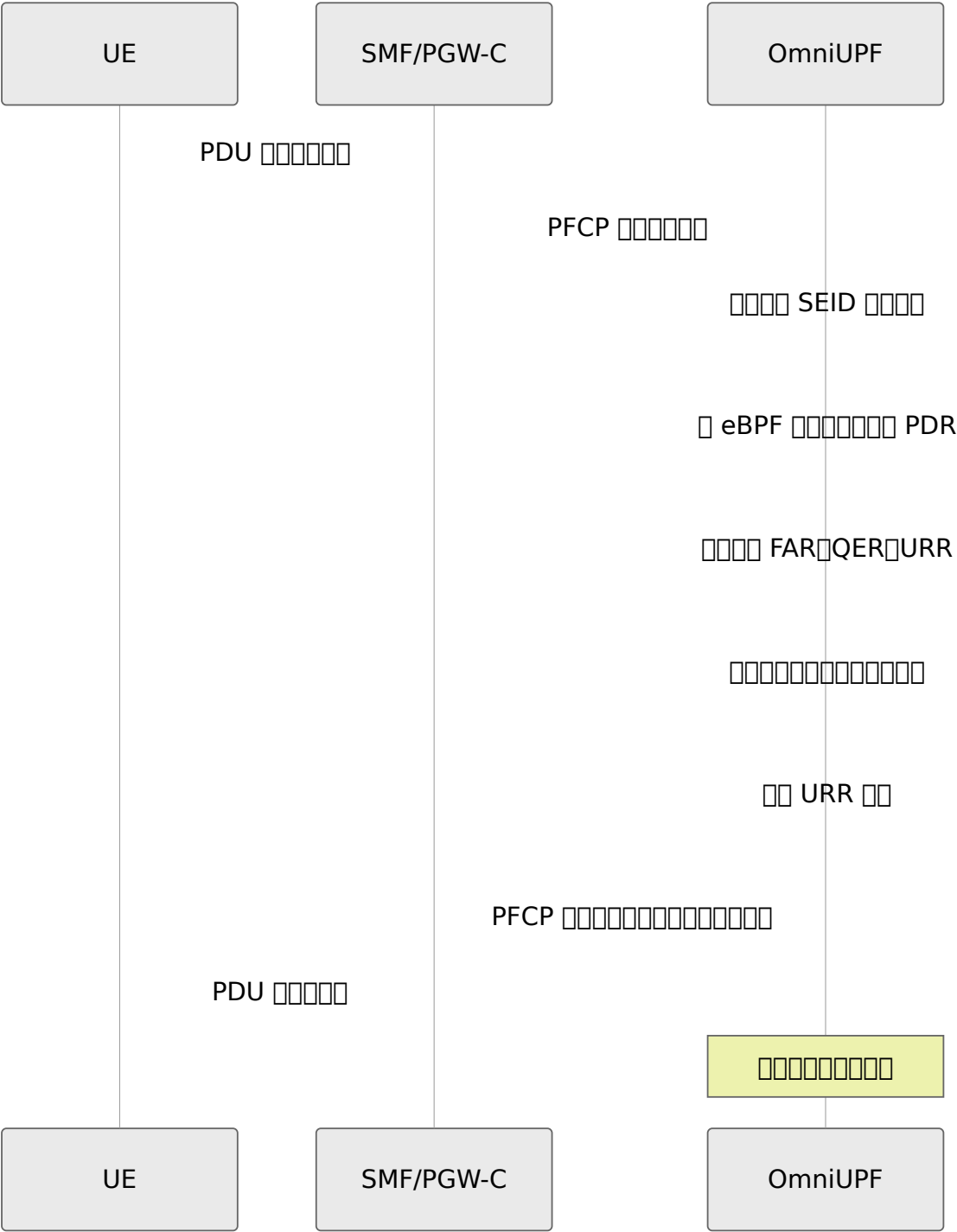
訊息流程



**PFCP**

PDU SMF UPF PFCP





000000

- 0000 PDR00000000
- 0000 FAR0QER0URR
- 00000000
- 0000000000 SMF 000000

# 概要

OmniUPF は Web を経由して REST API を利用して管理・制御される

## 機能

### 1. PFCP 機能

PFCP を通じて UE PDU セッション 5G PDP セッション LTE を管理

- セッション SEID を管理
- セッション PDR
- セッション FAR
- QoS を管理 QER
- セッション URR

### 2. 管理機能

- セッション UE IP を TEID を管理
- IP を TEID を管理
- セッション PDR/FAR/QER/URR を管理
- セッション PFCP を管理

管理・制御は REST API を利用

## 構成

### 1. PDR 機能

PDR は N3 を通じて UE PDU セッション 5G PDP セッション LTE を管理

- セッション PDR を N3 を通じて TEID を管理
- セッション PDR を UE IP を IPv4 または IPv6 を管理
- セッション SDF を管理
- セッション PDR を管理

## FAR

FAR 

- **FAR** 0000000000000000
- 0000000000000000
- 0000 **FAR** 00000
- 00000000000000 FAR

**QoS** □□□□□□ **QER**□□

QER

- **QoS** MBR GBR
- **QER**
- **QFI** 5G QoS

## URR

URR □□□□□□□□□□□□□□

- □□□□□□□□□□□□□□□□
- □□□□□□□□□□
- □□□□□□□□ **URR**

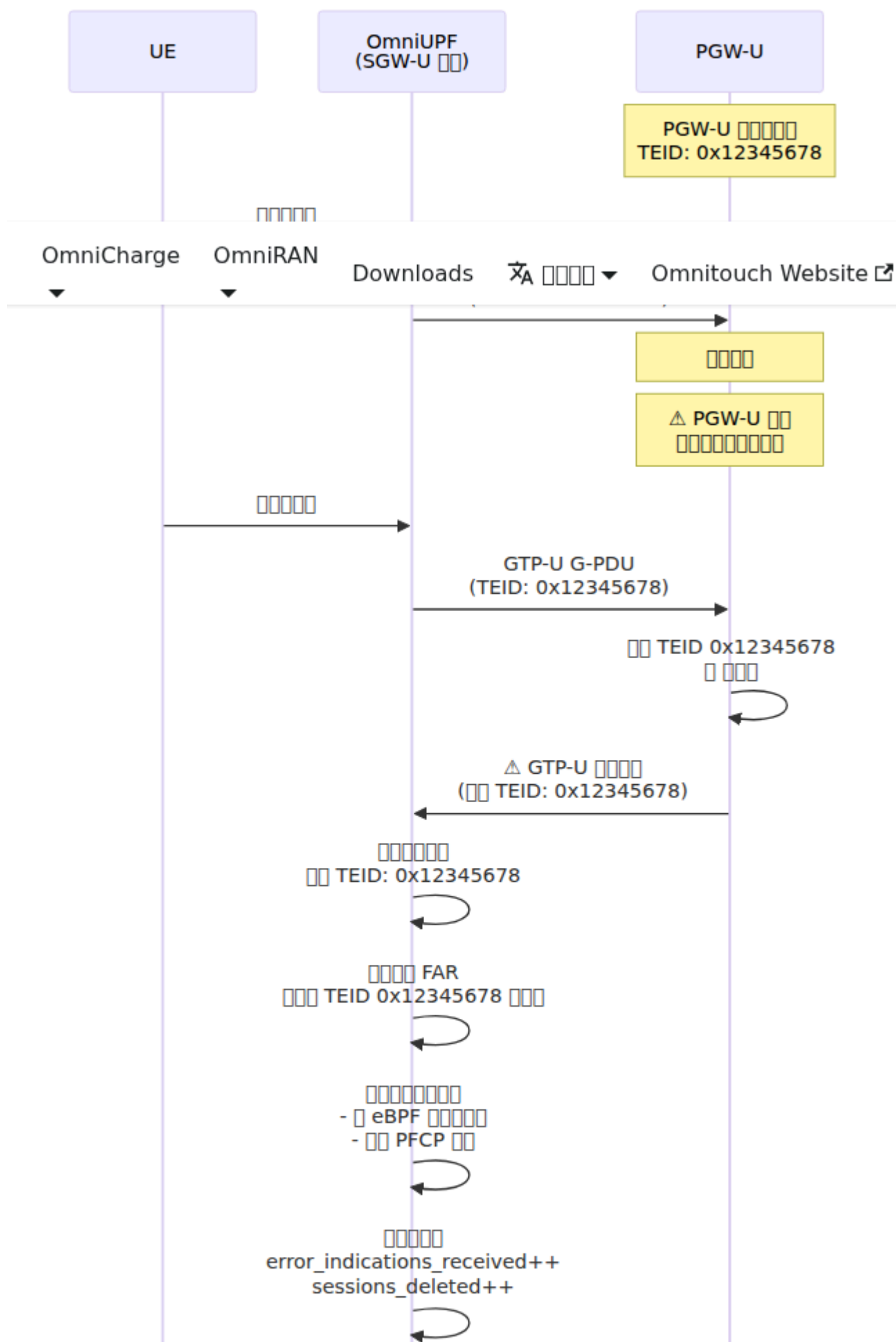
□ □ □ □ □

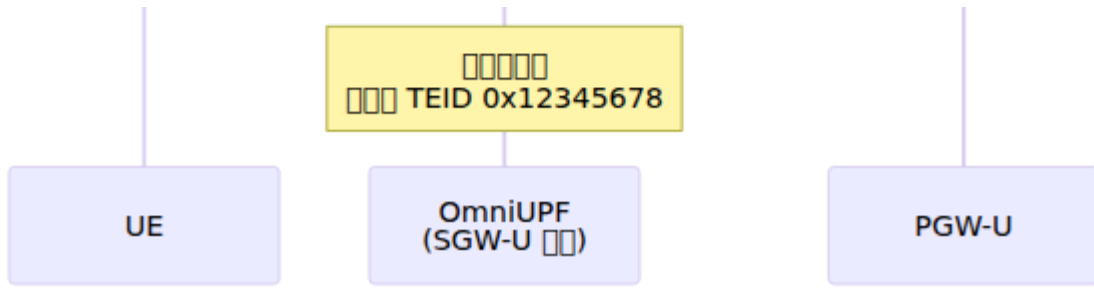
□□□□□ **UPF** □□□□

UPF

1 2 3 4 5 6 7 8 9 10 11 12

[illegible]





~40ms

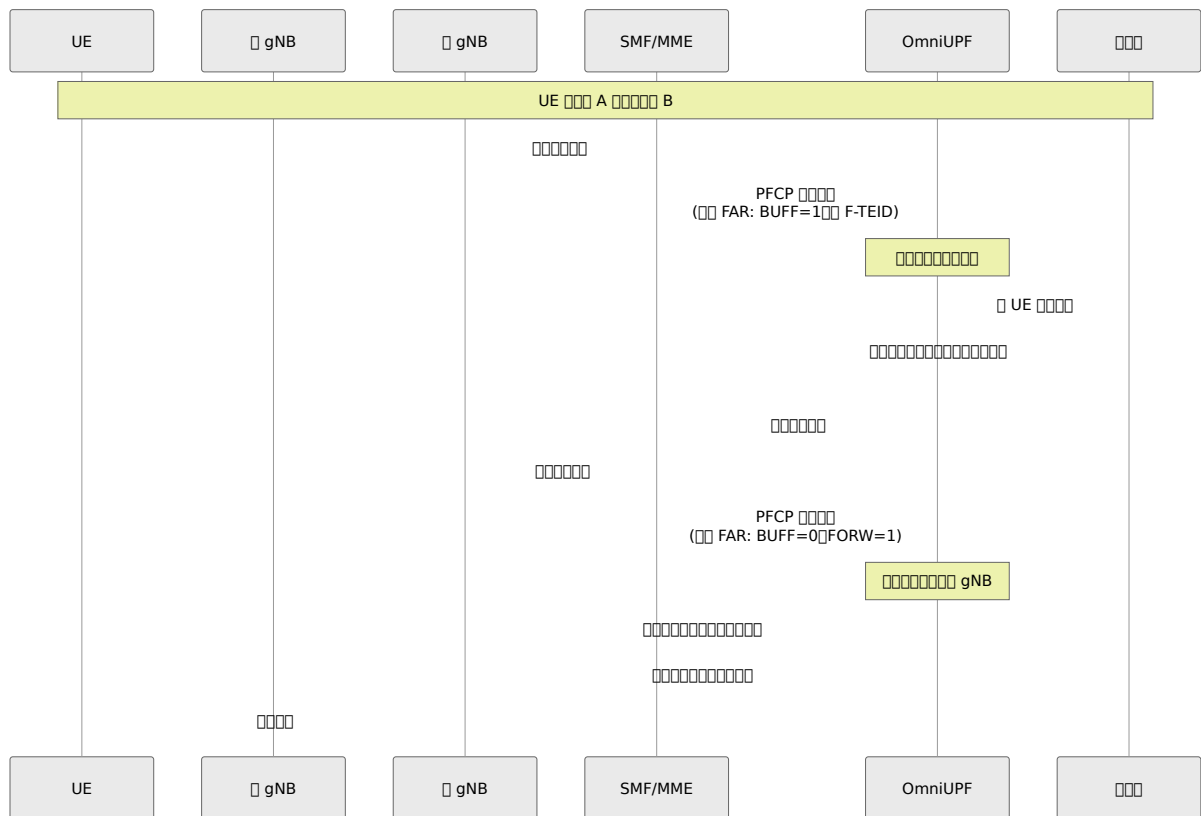
- **TCP**
- Zoom/Teams/WhatsApp
- VoIP
- 

OmniUPF

OmniUPF

1. N2 5G/ X2 4G

UE



Handover

- **T+0ms** Handover request
- **T+10ms** SMF sends UPF handover request
- **T+10-50ms** Handover request - Handover request
- **T+50ms** Handover request SMF sends UPF
- **T+50ms+** UPF handover request

Handover ~40ms Handover request Handover request

## 2. QoS Handover

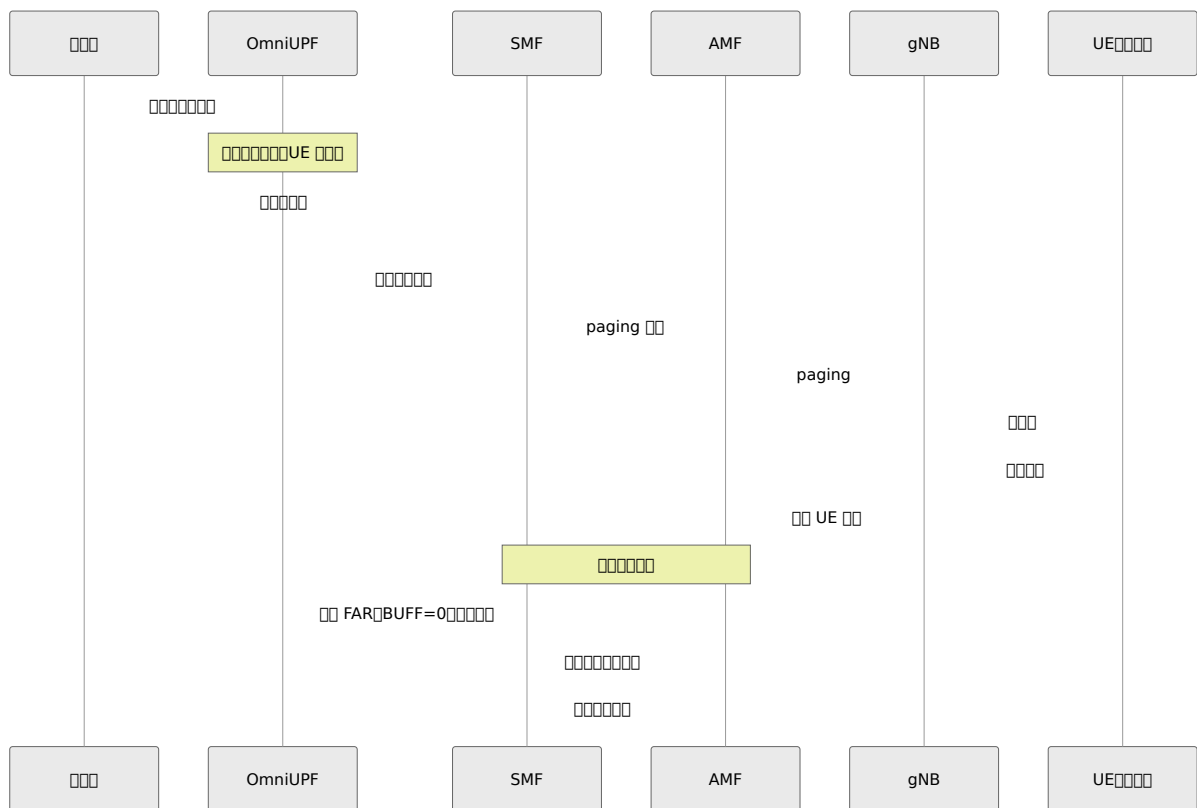
Handover

- **QoS** Handover/Handover 4G Handover 5G Handover NSA Handover
- Handover Handover Handover Handover Handover Handover
- Handover Handover Handover Handover Handover Handover UPF Handover ULCL Handover

Handover Handover Handover Handover Handover Handover Handover/Handover Handover

### 3. 3GPP 5G Core Network

UE 3GPP 5G Core Network



3GPP 5G Core Network 3GPP 5G Core Network 3GPP 5G Core Network UE 3GPP 5G Core Network UE 3GPP 5G Core Network

### 4. RAT 4G ↔ 5G

UE 4G 5G 3GPP 5G Core Network

- 3GPP eNodeB ↔ gNB
- 3GPP TEID 3GPP
- 3GPP RAT 3GPP 5G Core Network

### OmniUPF 3GPP 5G Core Network

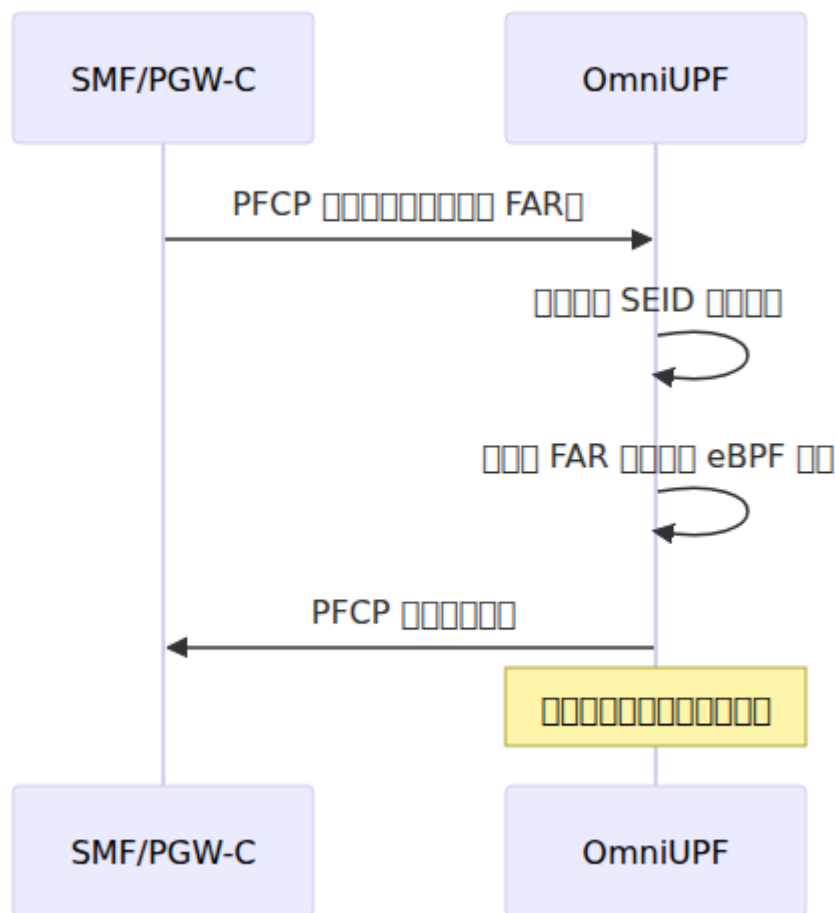
3GPP 5G Core Network

OmniUPF 3GPP 5G Core Network

#### 1. eBPF 3GPP 5G Core Network FAR 3GPP 5G Core Network

## 2. 3GPP 5G Core Network Architecture

Sequence



Details

- eBPF rules are stored in a buffer of size 22152
- GTP-U messages contain FAR ID and TEID
- FARs are mapped to eBPF rules
- Details
  - FAR buffer size: 10,000
  - eBPF rules buffer size: 100,000
  - TTL: 30 seconds - TTL timer
- eBPF rules are updated every 60 seconds

Implementation

1. SMF sends PFCP message with FARs to OmniUPF
2. OmniUPF maps FARs to eBPF rules and updates the buffer



- 3. 0000000000000000 FAR ID0000000000000000
- 4. 00000SMF 0000000000 FAR 00 FORW=10BUFF=0
- 5. 00000000000000000000000000000000 FAR 0000000000
- 6. 000000000000000000000000

000000000000

00000000

| 00                  | 0000             | 000       |
|---------------------|------------------|-----------|
| 0000000000          | 0000 1-2 000000  | 000000    |
| 0000000000          | 000000000000     | 00000000  |
| 000000000           | 0000000000       | 000000000 |
| 0000 <b>VoIP</b> 00 | 000000000000     | 000000000 |
| 00000000            | 0000000000       | 0000      |
| 000000000000        | SSH 000000000000 | 00000000  |

0000000000

- 00000000**CDR**0000 KPI
- 0000000000000000000000
- 00000000000000000000
- 00000000“00000000”

0000000

000000000000 Web UI 00 API 0000000000

000

- 0000 **FAR ID** 0000000000000000
- 0000000000000000 FAR0000
- 000000000000000000
- 00000000000000 > TTL 000

00000

- 000000000000000000
- 0000000000000000000000
- 00 **TTL**0000000000
- 0000000000 FAR 00000000

00000

- 00000000 SMF 00000 FAR 00000000
- 0000000000000000000000
- 0000000000TTL 00000000 FAR 0000
- 000000000000000000 SMF 00

0000000000000000 00000000

00000

0 config.yml 0000000000

```
0000
buffer_port: 22152 # 00000000 UDP 00000000
buffer_max_packets: 10000 # 00 FAR 0000000000000000
buffer_max_total: 100000 # 00 FAR 00000000
buffer_packet_ttl: 30 # TTL00000000000000
buffer_cleanup_interval: 60 # 00000000
```

0000

- 0000000000000000000000 buffer\_max\_packets 000 20,000+
- 0000000000000000000000 buffer\_packet\_ttl 000 15 0
- 00000000 buffer\_packet\_ttl 000 10 0000000000

- 5G Core Network Architecture

5G Core Network Architecture

## 5G Core Network

5G Core Network

5G Core Network

- 5G Core Network Architecture
- 5G Core Network Architecture
- 5G Core Network Architecture
- **GTP-U** Transport

5G Core Network

5G Core Network

- 5G Core Network Architecture
- 5G Core Network Architecture/UE
- 5G Core Network TEID UE IP

## XDP

5G Core Network

- **XDP** Transport XDP Transport
- **XDP** Transport XDP Transport
- **XDP** Transport XDP Transport
- **XDP** Transport XDP Transport

## N3/N6

5G Core Network

- **N3 RX/TX** RAN gNB/eNodeB
- **N6 RX/TX** Transport

- 网络性能指标

网络性能指标 网络性能

---

网络性能

**eBPF** 网络性能

UPF 网络性能 eBPF 网络性能

- 网络性能指标
- 网络性能 **eBPF** 网络性能
- 网络性能
  - 网络性能<50%网络性能
  - 网络性能50-70%网络性能
  - 网络性能70-90%网络性能
  - 网络性能>90%网络性能

网络性能

- `uplink_pdr_map` 网络性能
- `downlink_pdr_map` 网络性能 IPv4 网络性能
- `far_map` 网络性能
- `qer_map` 网络性能 QoS 网络性能
- `urr_map` 网络性能

网络性能

- 网络性能 PDR 网络性能 + 网络性能
- 网络性能 UPF 网络性能
- 网络性能

网络性能 网络性能

---

## 📌

### UPF 📌

📌 UPF 📌

- **N3** 📌 RAN 📌 IP 📌GTP-U
- **N6** 📌📌📌📌📌 IP 📌
- **N9** 📌📌 UPF 📌📌 IP 📌📌📌
- **PCF** 📌📌 SMF 📌 IP 📌
- **API** 📌📌REST API 📌📌
- 📌📌📌Prometheus 📌📌

📌📌📌📌

📌 eBPF 📌📌📌📌

- 📌 **N3** 📌📌📌 N3 📌📌
- 📌 **N9** 📌📌📌 N9 📌📌📌📌📌

📌📌📌📌📌📌 📌📌📌

## 📌

📌📌📌📌📌📌📌📌📌

## 📌📌📌📌

📌PCF 📌📌📌📌UE 📌📌📌📌

📌📌📌📌

### 1. PCF 📌📌

- 📌 SMF 📌📌📌 UPF PCF 📌📌 8805
- 📌📌📌📌 PCF 📌📌
- 📌📌 ID 📌 SMF 📌 UPF 📌📌📌

## 2. eBPF 實現

- 實現效率提升 >90%
- 與 UPF 集成 eBPF 實現
- 實現效率提升

## 3. 實現 PDR/FAR 實現

- 實現 UE IP 實現
- 實現 TEID 實現
- 實現 FAR 實現

## 4. 實現

- 實現 N3 實現 IP 實現 gNB 實現
- 實現 N6 實現
- 實現 GTP-U 實現

實現效率提升 100%

實現效率提升

實現 UE 實現

實現

## 1. PDR 實現

- 實現 PDR TEID 實現 gNB 實現 TEID 實現
- 實現 PDR UE IP 實現 IP 實現
- 實現 SDF 實現

## 2. FAR 實現

- 實現 FAR 實現
- 實現
- 實現

### 3. QoS 設定

- 設定 QER MBR 設定
- 設定 GBR 設定
- 設定 設定

### 4. 設定 MTU 設定

- 設定 GTP-U 設定 40-50 設定
- 設定 N3/N6 設定 MTU 設定
- 設定 ICMP 設定

## 設定

設定

設定

### 1. 設定

- 設定 FAR 設定 2 設定
- 設定 SMF 設定
- 設定

### 2. 設定 TTL 設定

- 設定
- 設定 TTL 設定
- 設定

### 3. 設定

- 設定 FAR 設定
- 設定
- 設定 max\_per\_far 設定 max\_total 設定

設定

## 目標

ネットワークの最適化

ネットワーク

### 1. ネットワーク

- eBPF を利用して 64 ビット幅の整数を扱う
- ネットワークの最適化
- ネットワークの URR を利用して

### 2. ネットワーク

- ネットワーク eBPF を利用して
- ネットワークの最適化 eBPF を利用して
- ネットワーク XDP を利用して

### 3. ネットワーク

- ネットワーク N3/N6 を利用して
- ネットワーク eBPF を利用して
- ネットワーク XDP を利用して

## 目標

ネットワークの最適化 CPU を利用して

ネットワーク

1. ネットワーク XDP を利用して XDP を利用して
2. ネットワーク eBPF を利用して ネットワーク
3. ネットワーク CPU を利用して eBPF を利用して
4. ネットワーク NIC を利用して XDP を利用して

ネットワーク

- ネットワーク XDP を利用して 10M+ を利用して



- **PDR** ☐ ☐ ☐ PDR ☐ ☐ ☐ ☐
- ☐ ☐ ☐ UPF ☐ ☐ ☐ ☐
- ☐ ☐ ☐ NIC ☐ ☐ ☐ ☐ ☐ ☐

☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐

☐ ☐ ☐ ☐

☐ ☐ ☐ ☐ ☐ ☐

☐ ☐ UPF ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐

☐ ☐ ☐

☐ ☐ ☐ ☐ ☐ ☐

- ☐ ☐ ☐YAML☐ ☐ ☐ ☐CLI☐
- ☐ ☐ ☐UPF/PGW-U/SGW-U☐
- XDP ☐ ☐ ☐
- ☐ ☐ ☐Proxmox☐VMware☐KVM☐Hyper-V☐VirtualBox☐
- NIC ☐ ☐ ☐ XDP ☐ ☐ ☐
- ☐ ☐ ☐ ☐ ☐
- ☐ ☐ ☐ ☐ ☐

**XDP** ☐ ☐ ☐

☐ ☐ XDP ☐ ☐ ☐ ☐ ☐

- XDP ☐ ☐ ☐ ☐ ☐ ☐ ☐
- ☐ ☐ ☐ ☐ ☐
- Proxmox VE ☐ ☐ XDP ☐ ☐ ☐ ☐
- ☐ ☐ ☐ ☐ ☐ ☐
- VMware ESXi☐KVM ☐ Hyper-V ☐ XDP ☐ ☐