

OmniUPF 架构图

简介

1. 概述
2. 5G 网络架构
3. UPF 功能
4. PCF 与 SMF 交互
5. 网络切片
6. 策略控制
7. 计费
8. 安全

架构

OmniUPF 基于 eBPF 技术实现 5G/LTE 网络切片 QoS 策略控制，支持 Linux eBPF 技术实现 OmniUPF 5G SA 5G NSA 及 LTE 网络切片。

功能特性

OmniUPF 支持 3GPP 5G 及 LTE 网络切片。

- 支持网络切片
- 支持 QoS 策略控制
- 支持策略控制
- 支持计费
- 支持安全
- 支持策略控制

OmniUPF 支持 3GPP TS 23.501 5G TS 23.401 LTE 网络切片 UPF 策略控制，支持 Linux eBPF 技术实现。

OmniUPF 架构图

架构图

- 支持 3GPP 标准
- 支持 eBPF 可编程
- GTP-U 支持
- 支持 IPv4 和 IPv6
- XDP 支持
- 支持

QoS 支持

- 支持 QoS 策略
- 支持 PDR
- 支持 FAR
- 支持 SDF
- 支持 URR

支持

- PFCP 支持 SMF/PGW-C
- RESTful API
- 支持
- eBPF 支持
- 支持 Web 支持

支持

- 支持 eBPF 支持
- 支持
- 支持
- 支持
- 支持

支持 Web UI 支持

OmniUPF

OmniUPF 5G SA 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

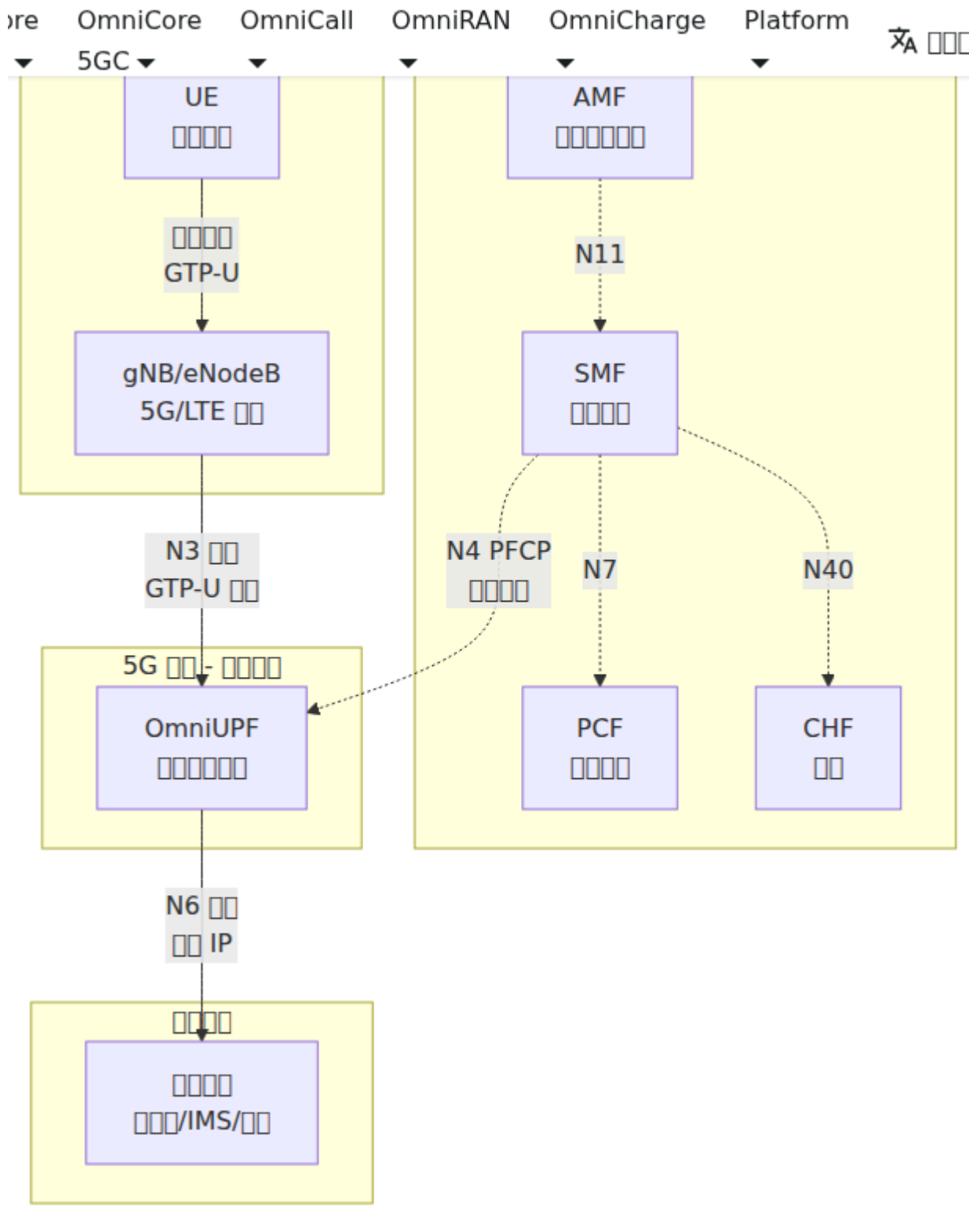
OmniUPF

- **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

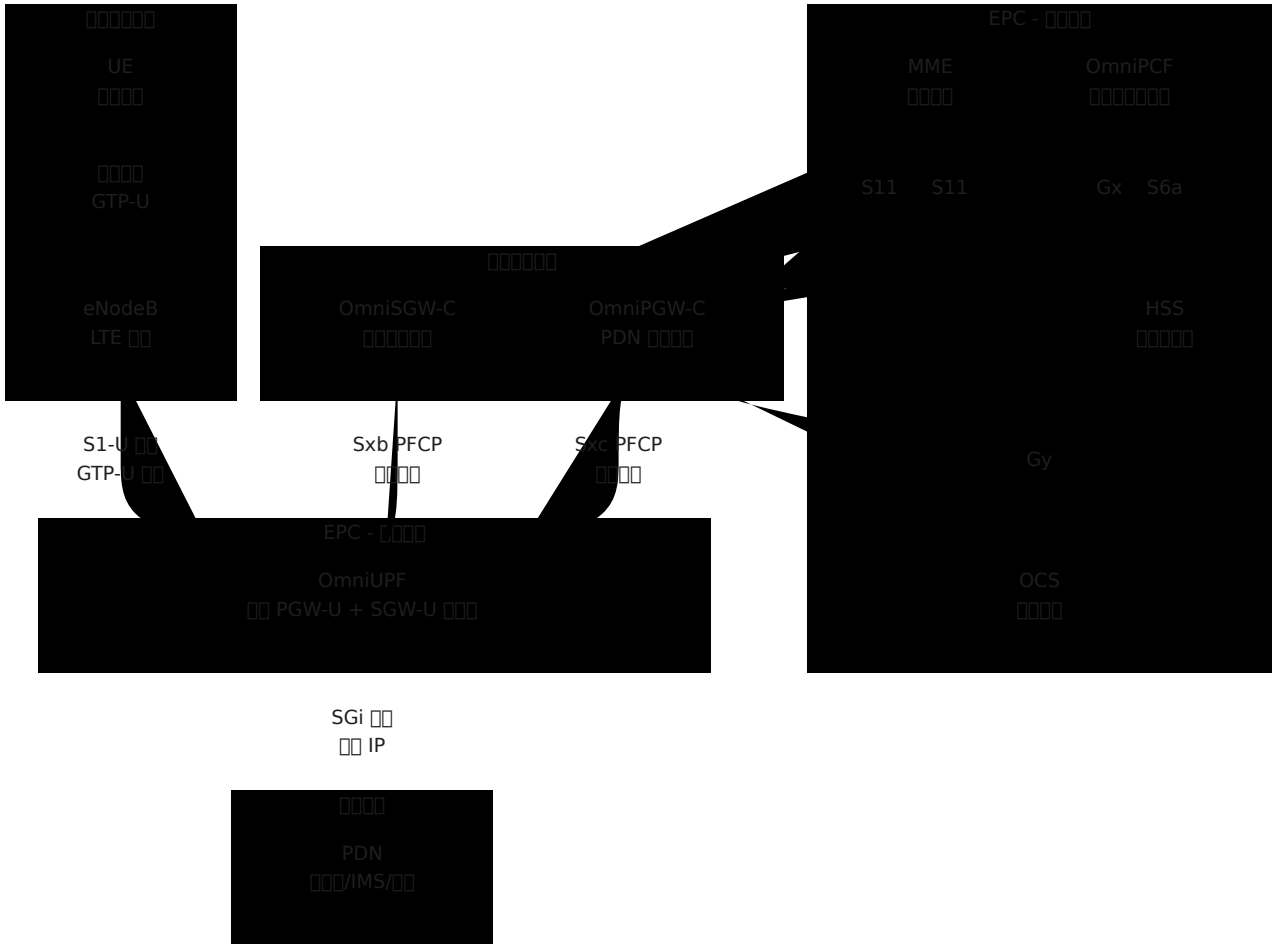


4G LTE/EPC

OmniUPF is a 4G LTE EPC component that maps to OmniPGW-U and OmniSGW-U in the 4G LTE network.

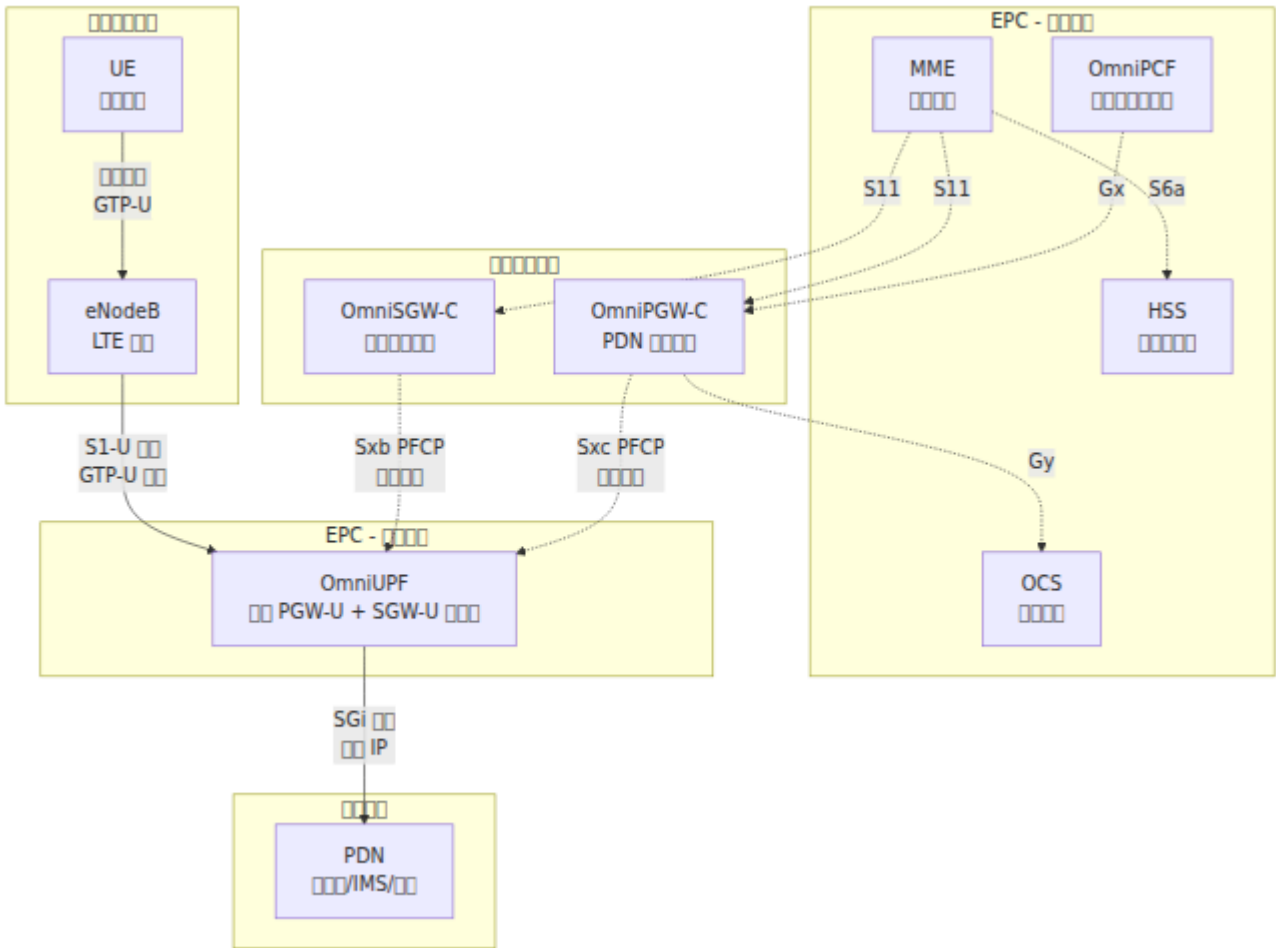
PGW-U/SGW-U 4G

OmniUPF SGW-U PGW-U



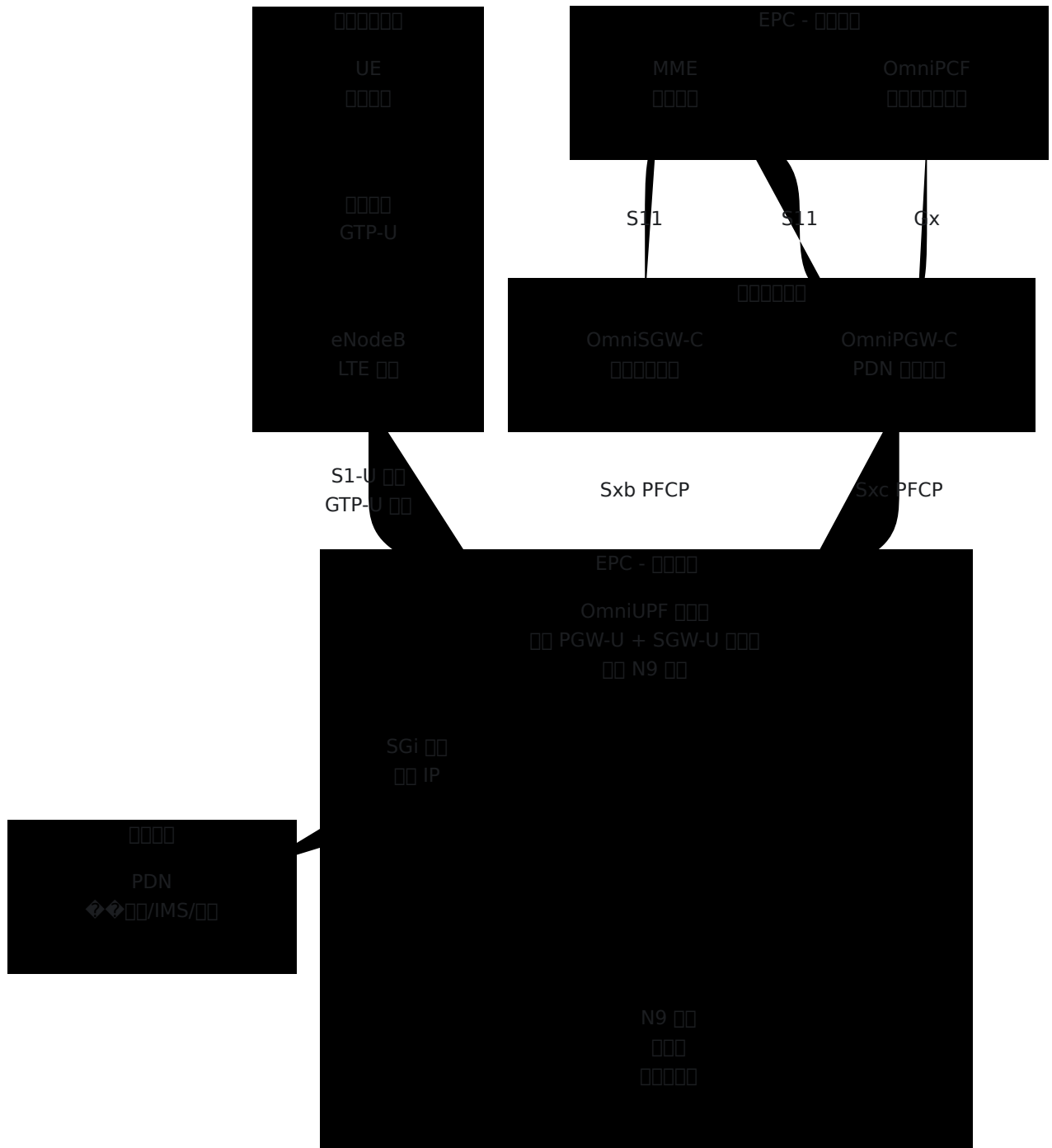
SGW-U PGW-U

OmniUPF - SGW-U PGW-U



N9 SGWU+PGWU

OmniUPF SGWU PGWU N9 eBPF



Challenges

- New N9 interface - requires eBPF for packet processing
- ~40-50% CPU usage - requires XDP for packet processing
- New interface - requires new hardware
- New interface - `n3_address = n9_address` for connectivity
- New interface 3GPP interface - requires PFCP and GTP-U

Summary

```
# /etc/omniupf/runtime.exs
xdp_interfaces = "eth0"
n3_address = "10.0.1.10"      # S1-U IP
n9_address = n3_address      # IP N9
pfcf_address = "10.0.1.10"   # SGWU-C PGWU-C
pfcf_port = 8805
```

Configuration

- Configuration
- Configuration
- Config/Configuration
- Configuration < 100K

Configuration

- Configuration SGWU PGWU
- Configuration
- Configuration > 1M

Configuration N9

Configuration

Configuration OmniUPF OmniPGW-U OmniSGW-U

1. Configuration

- 5G OmniSMF N4 OmniUPF PFCP
- 4G OmniPGW-C OmniSGW-C Sxb/Sxc OmniPGW-U/OmniSGW-U PFCP
- UE PDU 5G PDP 4G PFCP
- PFCP PDR FAR QER URR
- eBPF

2. Configuration UE → Configuration

- **5G** N3 gNB GTP-U
- **4G** S1-U SGW-U S5/S8 PGW-U eNodeB GTP-U
- TEID PDR
- eBPF QER
- FAR
- GTP-U N6 5G SGi 4G
- URR

3. → UE

- **5G** N6 IP
- **4G** SGi IP
- UE IP PDR
- SDF
- FAR GTP-U
- GTP-U TEID
- **5G** N3 gNB
- **4G** S1-U SGW-U S5/S8 PGW-U eNodeB

4.

- **5G** OmniSMF PDR/FAR
- **4G** OmniSGW-C/OmniPGW-C eNodeB TAU
-
-

4G 5G

OmniUPF 3GPP 5G 4G

5G

Interface	Direction	Protocol	3GPP Reference
N4	OmniSMF ↔ OmniUPF	PFCP	TS 29.244
N3	gNB → OmniUPF	RAN GTP-U	TS 29.281
N6	OmniUPF → DN	IP	TS 23.501
N9	OmniUPF ↔ OmniUPF	UPF	TS 23.501

4G/EPC Interface

Interface	Direction	Protocol	3GPP Reference
Sxb	OmniSGW-C ↔ OmniUPF/SGW-U	PFCP	TS 29.244
Sxc	OmniPGW-C ↔ OmniUPF/PGW-U	PDN PFCP	TS 29.244
S1-U	eNodeB → OmniUPF/SGW-U	RAN GTP-U	TS 29.281
S5/S8	OmniUPF/SGW-U ↔ OmniUPF/PGW-U	GTP-U	TS 29.281
SGi	OmniUPF/PGW-U → PDN	IP	TS 23.401

PFCP N4/Sxb/Sxc TS 29.244 PFCP

UPF

eBPF

eBPF Linux

- **GTP-U**
- TEID UE IP SDF
- **QoS** QER
- FAR
- URR

eBPF eBPF

<code>uplink_pdr_map</code>	PDR	TEID 32	PDR FAR ID QER ID URR IDs
<code>downlink_pdr_map</code>	PDR IPv4	UE IP	PDR
<code>downlink_pdr_map_ip6</code>	PDR IPv6	UE IPv6	PDR
<code>far_map</code>		FAR ID	
<code>qer_map</code>	QoS	QER ID	QoS MBR GBR
<code>urr_map</code>		URR ID	
<code>sdf_filter_map</code>	SDF	PDR ID	

- **PF** **PF**
- **XDP** **PF**
- **PF** CPU **PF** CPU **PF**
- **PF** eBPF **PF** PDRs/FARs **PF**

PF **PF**

PF **PF**

PF **PF** 3GPP TS 29.244 **PF** **PF** PGW-C **PF**

PF

- **PF** PF **PF**
- **PF** **PF** PF **PF**
- **PF** PF **PF** eBPF **PF**
- **PF** SMF **PF**

PF **PF**

PF	PF	PF
PF	SMF → UPF	PF PF PF
PF	SMF → UPF	PF PF PF
PF	PF	PF
PF	SMF → UPF	PF PDU PF PDR/FAR/QER/URR
PF	SMF → UPF	PF QoS PF
PF	SMF → UPF	PF
PF	UPF → SMF	PF

□□□□□□□□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 □□□□□□□□□□□□

□□□□□

- □□□□□□□□□□ PFCP □□□□□
- □□□□□□□□ PDR□FAR□QER□URR □□
- □□□□□□□□□□□□□□□□□□□□XDP □□
- □□□□□□□□□□□□□□□□
- □□□□□□□□ eBPF □□□□□□□□□□□□

API □□□□□ 34 □□□□□



API	Endpoint	Functionality
Health	/health	Health check
Configuration	/config	UPF configuration
PCF	/pfcpsessions, /pfcpsessions	PCF sessions/associations
PDRs	/uplink_pdr_map, /downlink_pdr_map, /downlink_pdr_map_ip6, /uplink_pdr_map_ip6	PDR configuration
FARs	/far_map	FAR configuration
QERs	/qer_map	QoS configuration
URRs	/urr_map	URR configuration
Buffer	/buffer	Buffer configuration
Stats	/packet_stats, /route_stats, /xdp_stats, /n3n6_stats	Statistics
eBPF	/map_info	eBPF configuration
Dataplane	/dataplane_config	N3/N9 configuration

API endpoints are listed in the table above.

Web

Web interface for UPF configuration and monitoring.

□□□

- □□□□□□□□ PFCP □□□□ UE IP□TEID □□□□□
- □□□□□□□□□□□□□□ PDRs□FARs□QERs □ URRs
- □□□□□□□□□□□□□ FAR □□□□
- □□□□□□□□□□□□□□ XDP □ N3/N6 □□□□
- □□□□□ eBPF □□□□□□□□□□□□□□□□□□□□
- □□□□□□□ UPF □□□□□□□□□
- □□□□□□□□□□□□□□□□□□

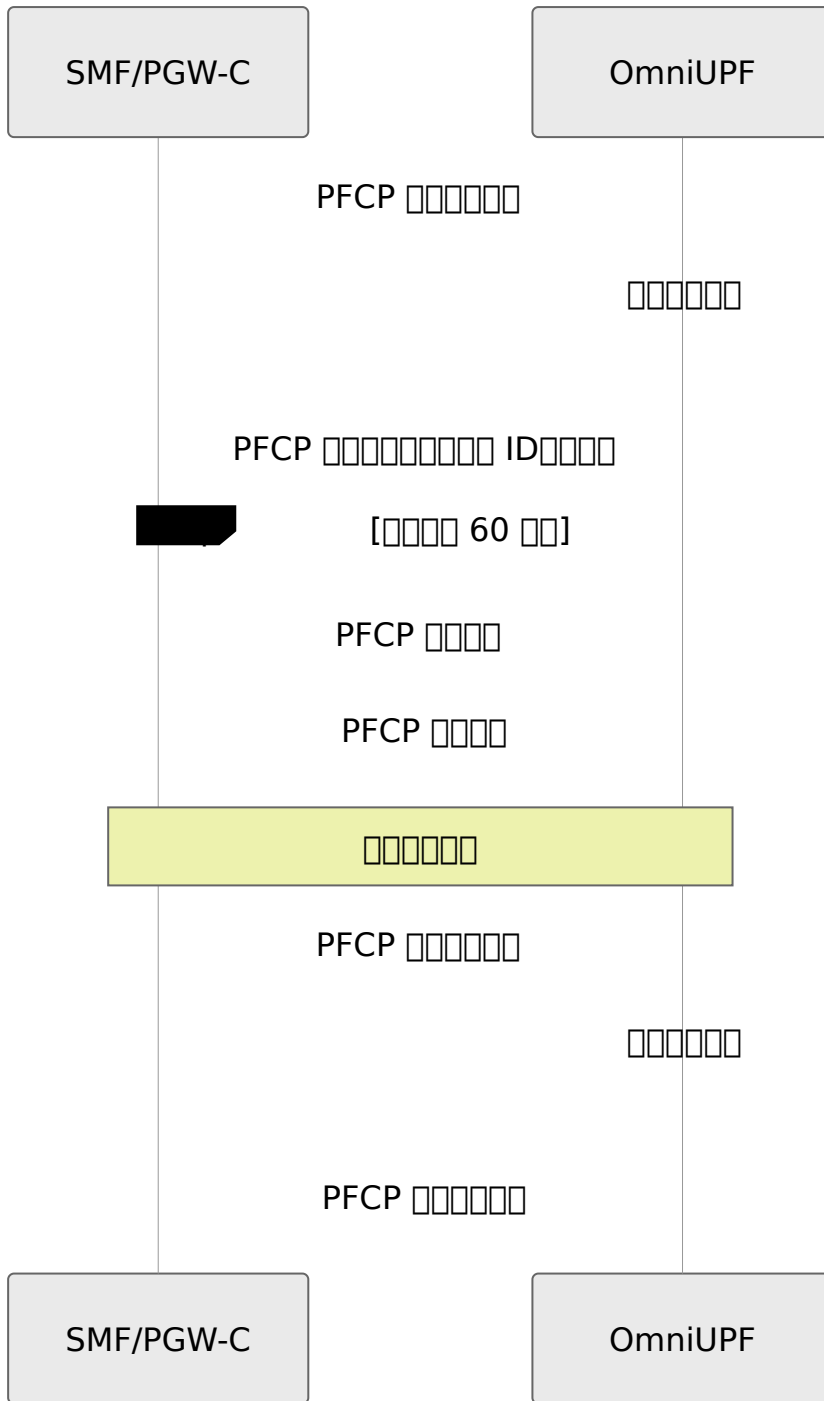
□□□□□ UI □□□□□□ **Web UI** □□□□□

PFCP □□□ SMF □□

PFCP □□

□□□□□□□□ SMF □□□ UPF □□ PFCP □□□

□□□□□□□



[]

- SMF [] UPF []
- UPF [] ID [] FQDN [] IP []
- []
- []

[] []

SMF หน้าที่

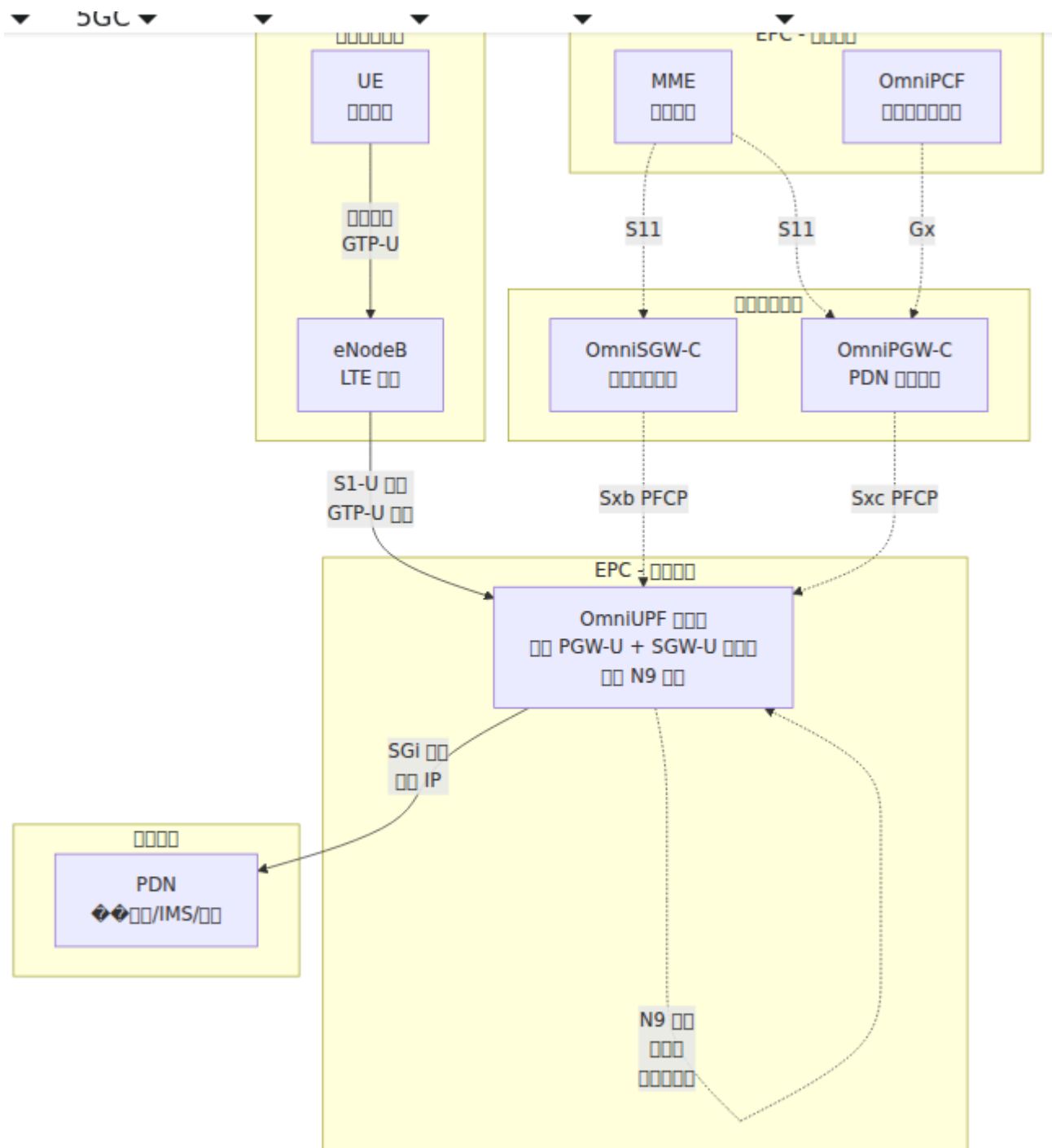
OmniUPF หน้าที่ SMF 3GPP TS 29.244 หน้าที่

หน้าที่

SMF หน้าที่ PCFP หน้าที่ OmniUPF หน้าที่ SMF หน้าที่

1. SMF หน้าที่
2. SMF หน้าที่ UPF หน้าที่ PCFP หน้าที่
3. SMF หน้าที่ หน้าที่
4. UPF หน้าที่ = SMF หน้าที่
5. UPF หน้าที่ หน้าที่ SMF หน้าที่
6. SMF หน้าที่

หน้าที่



□□□□

□ SMF □□□□□□□□

```
WARN: [ ] NodeID: smf-1 [ ]: 192.168.1.10 [ ]
WARN: SMF [ ]2025-01-15T10:00:00Z[ ]2025-01-15T10:30:15Z[ ]
- SMF [ ] 245 [ ]
INFO: [ ] SMF [ ] 2[LocalSEID[ ]
INFO: [ ] SMF [ ] 3[LocalSEID[ ]
...
INFO: [ ] SMF [ ] 246[LocalSEID[ ]
```

[]

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 []

[]

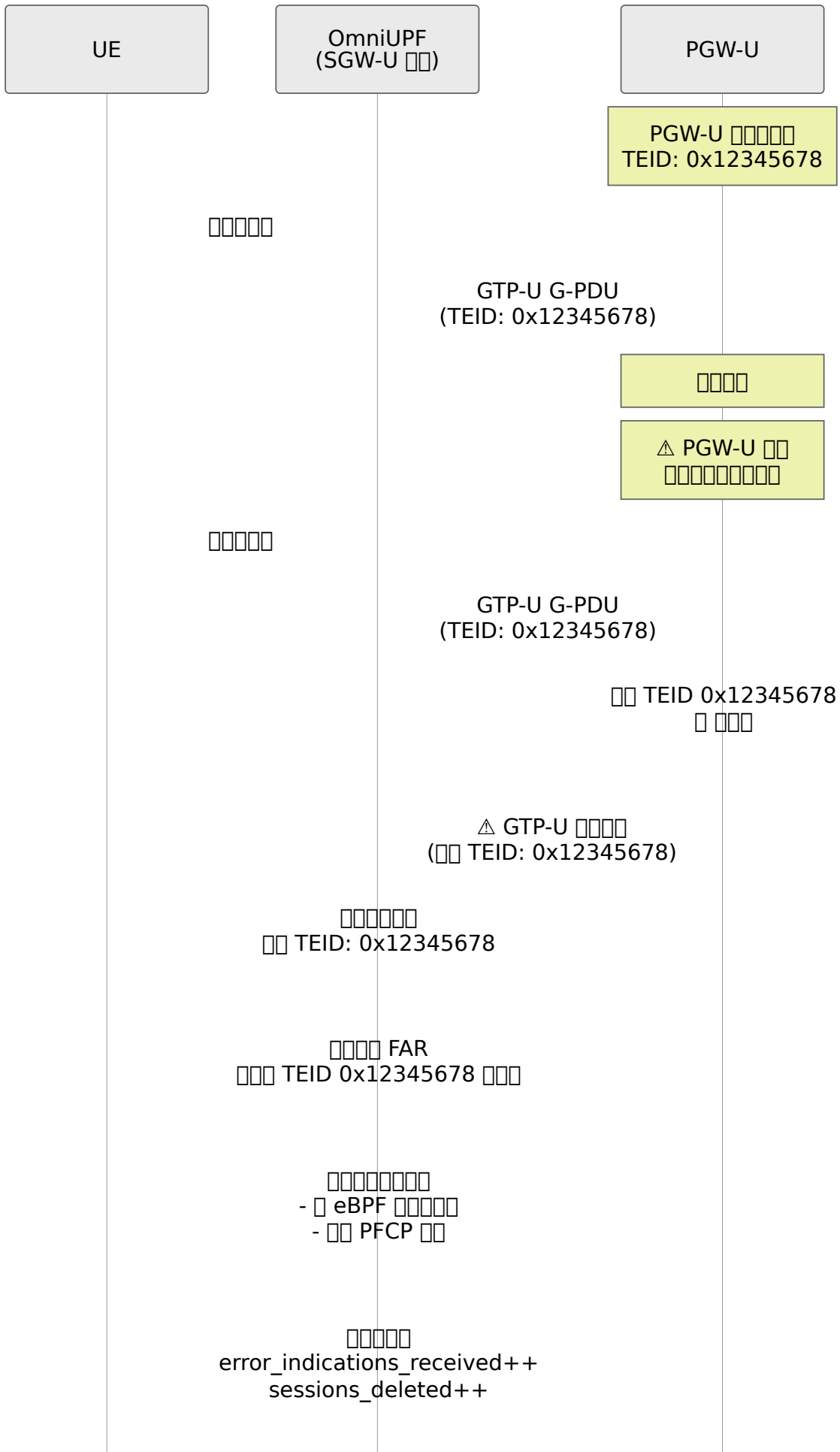
[] OmniUPF [] GTP-U [] SGW-U [] PGW-U [] TEID []
[]

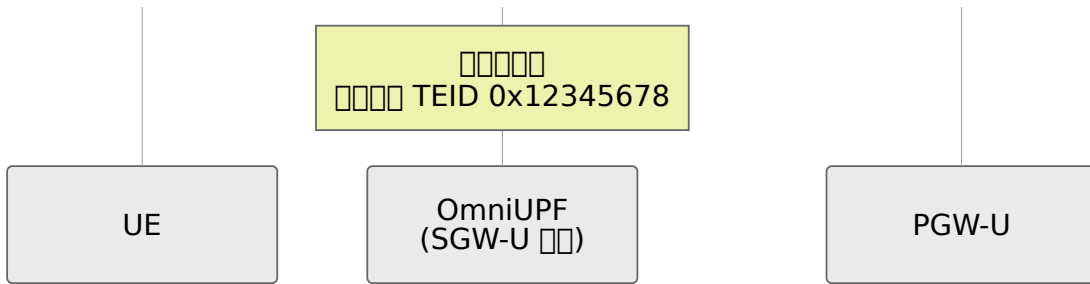
- []
- []
- []

[]

1. **UPF** [] → [] TEID X [] GTP-U [] 2152[]

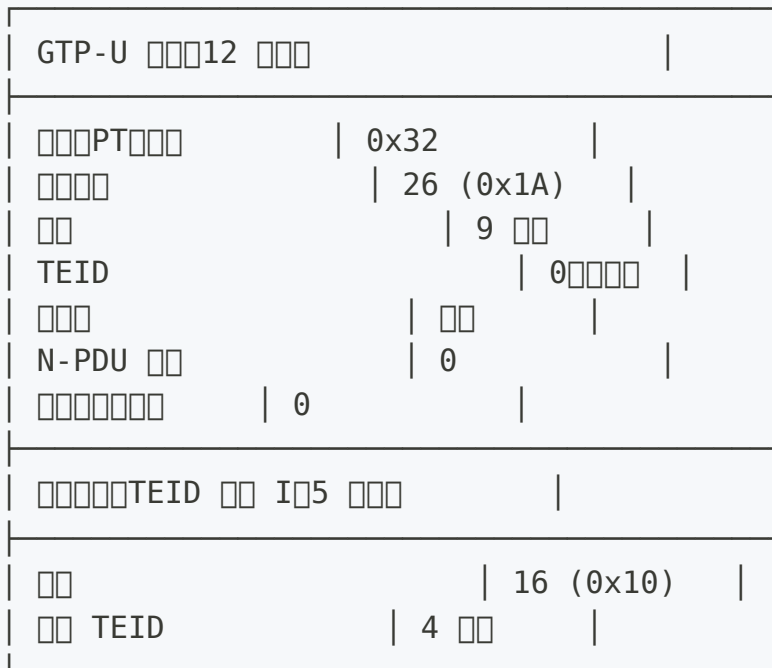
2. **TEID X** → TEID
3. → GTP-U 26 TEID
4. **UPF** → TEID X
5. **UPF** → TEID X FAR
6. **UPF** → eBPF PFCP
7. **UPF** → Prometheus





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
- UPF-1
- UPF-1

```

WARN: 192.168.50.10:2152 GTP-U TEID 0x12345678 - TEID
WARN: LocalSEID=42 FAR GlobalId=1 TEID 0x12345678
192.168.50.10
INFO: GTP-U LocalSEID=42 TEID 0x12345678
192.168.50.10
WARN: GTP-U 1 TEID 0x12345678 192.168.50.10

```

Prometheus

```

#
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"}

```

- `node_id` PFCP ID
- `peer_address` IP

□□□□

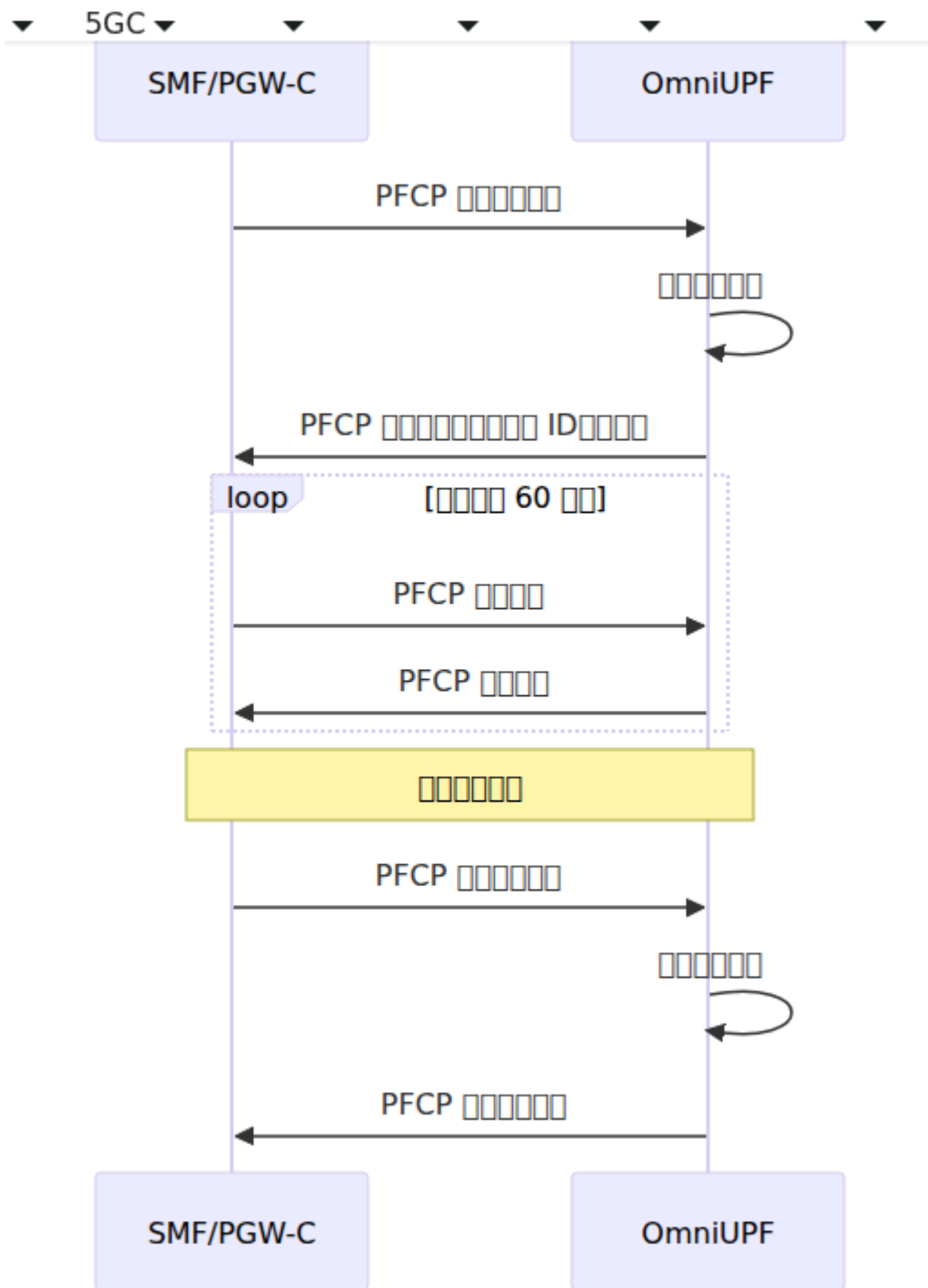
1. □□□□□□□□□□ - □□□□□□
2. **TEID** □□□□□□□□□□ □□□□ □□
3. □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
4. □□□□□□□□□□□□□□□□ □□□□□□□□□□ □□□□□□□□
5. □□□□□□□□□□
 - □□□□□□□□□□ = □□□□□□□□□□□□□□□□□□□□□□□□□□
 - □□□□□□□□□□ = □□□□□□□□□□□□□□□□□□□□□□□□□□
6. □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

□□□□□□□□□□□□□□□□ **GTP-U** □□□□□□□□□□

PFCP □□□□

□ UE □□ PDU □□□5G□□ PDP □□□□LTE□□□□□SMF □□ UPF □□ PFCP □□□□

□□□□□□□□□□



[]

- [] **PDR** [] N3 TEID [] FAR [] N6
- [] **PDR** [] UE IP [] FAR [] N3 [] GTP-U []
- **FAR** []
- **QER** [] QoS [] MBR [] GBR [] QFI []
- **URR** []

PFCP

SMF QoS

1. N2

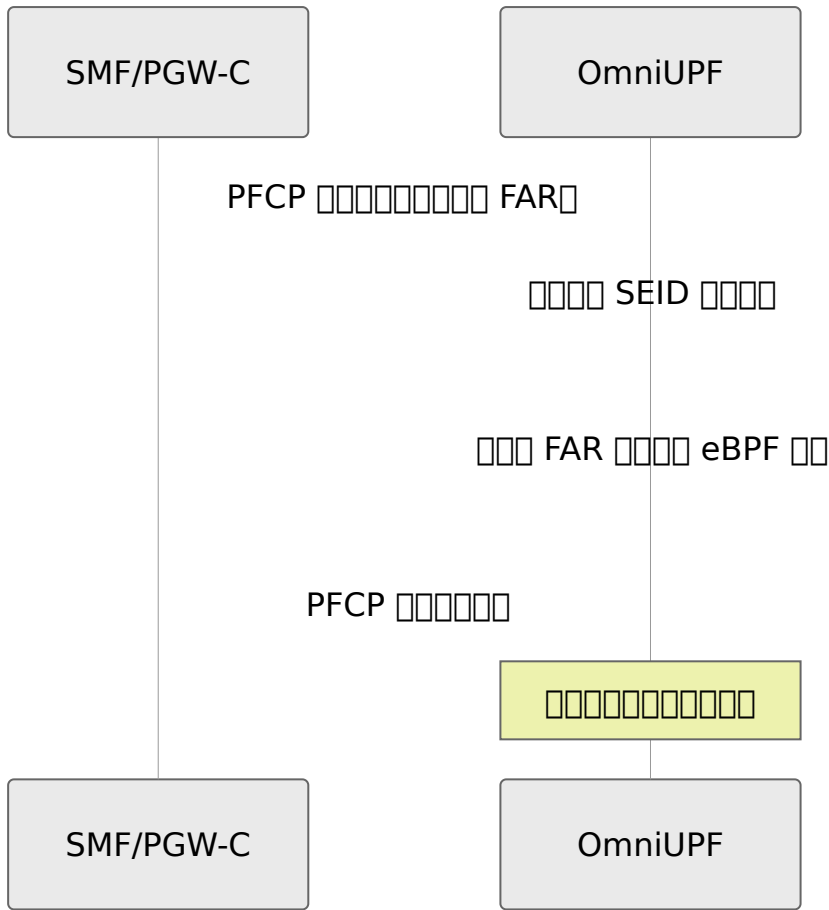
- gNB F-TEID FAR
-
-

2. QoS

- MBR/GBR QER
- PDR SDF QoS

3.

- PDR
- FAR

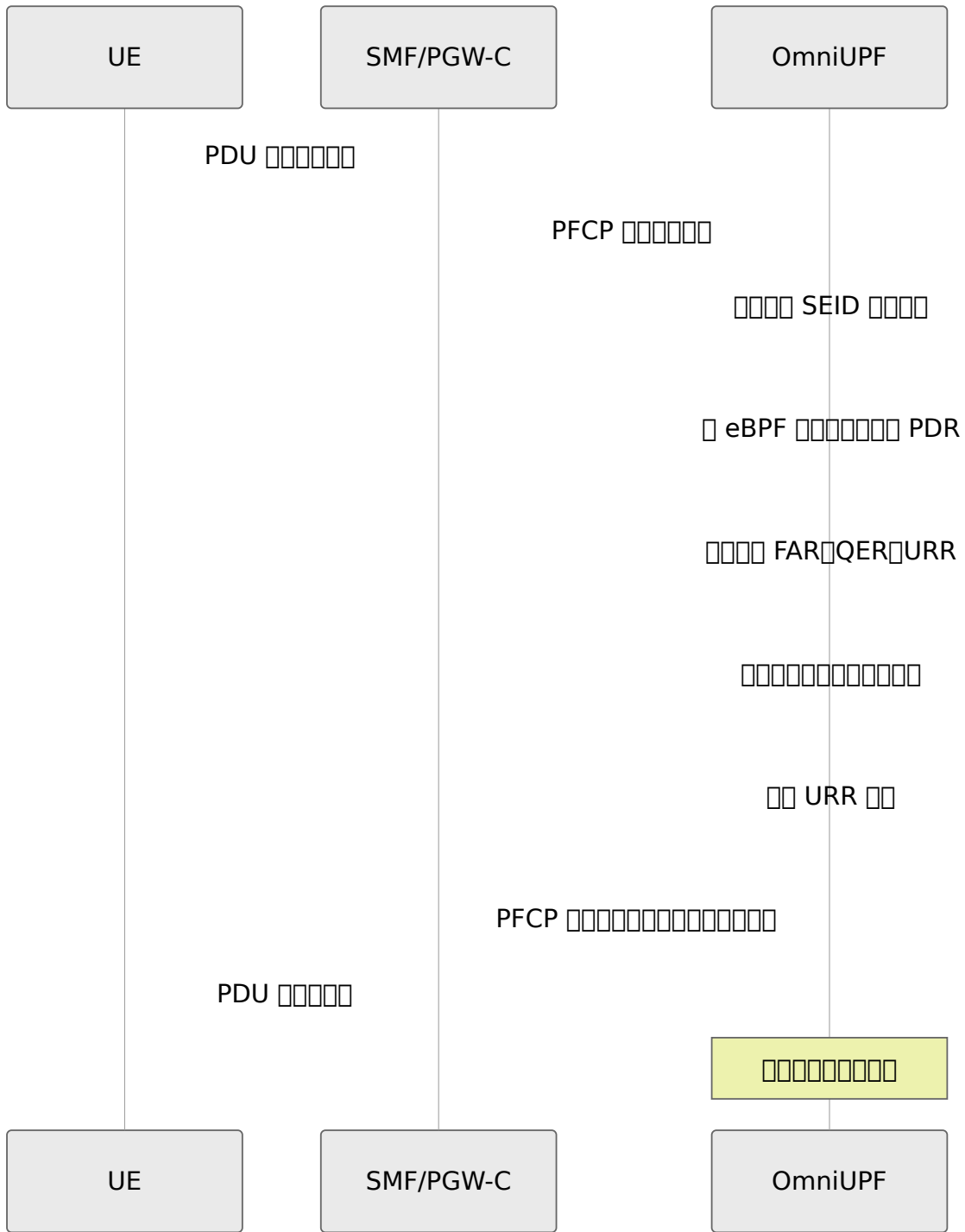


□□□□□□□□□□ □□□□□□

PFCP □□□□

□ PDU □□□□□□□□ SMF □□ UPF □ PFCP □□□□

□□□□□□□□



□□□□□□

- □□□❓❓ PDR□□□□□□□□
- □□□□ FAR□QER□URR
- □□□□□□
- □ SMF □□□□□□□□□□□□□□

Table of Contents

OmniUPF supports Web based REST API

Table of Contents

PFCP

PFCP supports UE PDU 5G PDP LTE

- SEID
- PDR
- FAR
- QoS QER
- URR

Table of Contents

- UE IP TEID
- IP TEID
- PDR/FAR/QER/URR
- PFCP

Table of Contents

Table of Contents

PDR

PDR

- PDR N3 TEID
- PDR UE IP IPv4 IPv6
- SDF
- PDR

QoS Parameters FAR

FAR (Forwarding Action Rule) is used to...

- FAR is used to...
- ...
- FAR is used to...
- ...

QoS Parameters QER

QER (Quality of Effort Rule) is used to...

- QoS parameters MBR, GBR...
- QER is used to...
- 5G QoS parameters QFI

QoS Parameters URR

URR (Usage Reporting Rule) is used to...

- URR is used to...
- ...
- URR is used to...

QoS Parameters URR (continued)

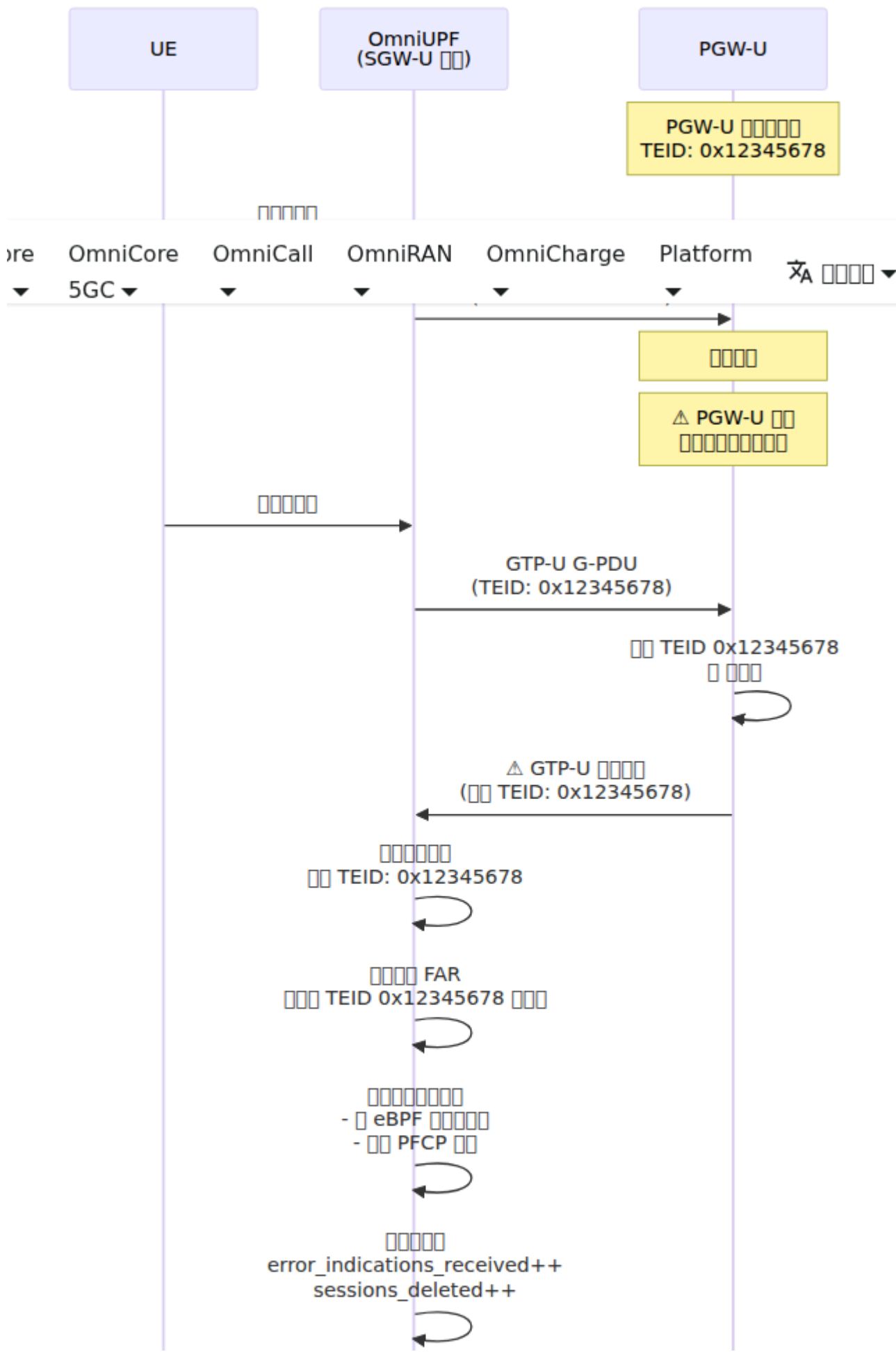
UPF

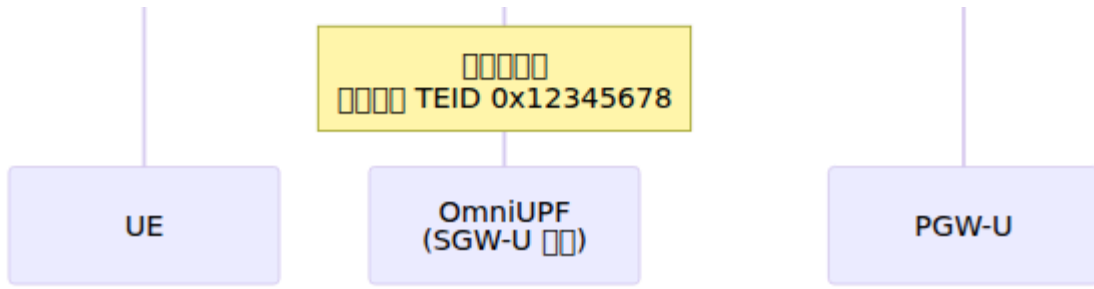
UPF (User Plane Function) is used to...

UPF is used to...

UPF (continued)

UPF (continued)





OmniUPF 支持多种业务类型

- **TCP** 业务类型
- 业务类型 Zoom/Teams/WhatsApp
- 业务类型
- **VoIP** 业务类型
- 业务类型

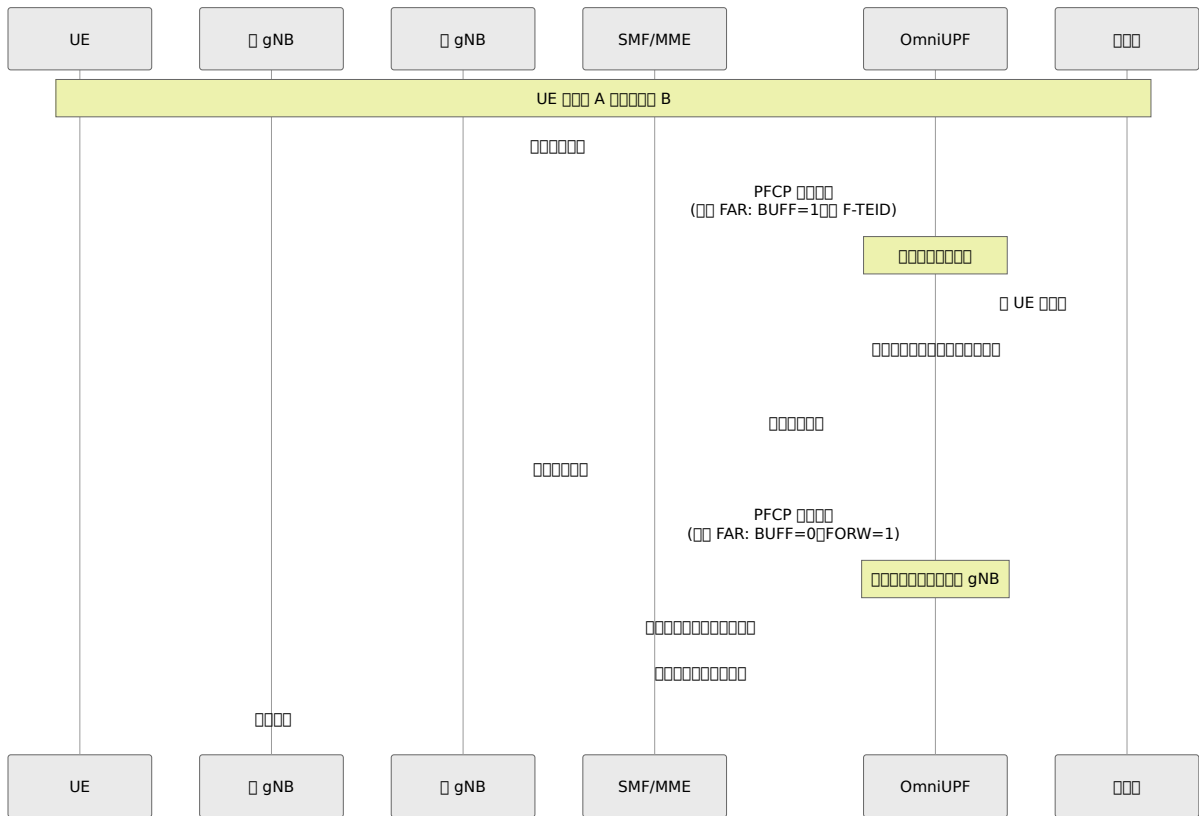
OmniUPF 支持多种业务类型

业务类型

OmniUPF 支持多种业务类型

1. N2 5G/ X2 4G

UE 业务类型



40ms

- **T+0ms** SMF/MME → OmniUPF
- **T+10ms** SMF/MME → OmniUPF (PFCP session establishment)
- **T+10-50ms** SMF/MME → gNB (QoS parameters)
- **T+50ms** SMF/MME → OmniUPF (PFCP session establishment)
- **T+50ms+** OmniUPF → gNB (QoS parameters)

40ms ~ 40ms (Total delay)

2. QoS Parameters

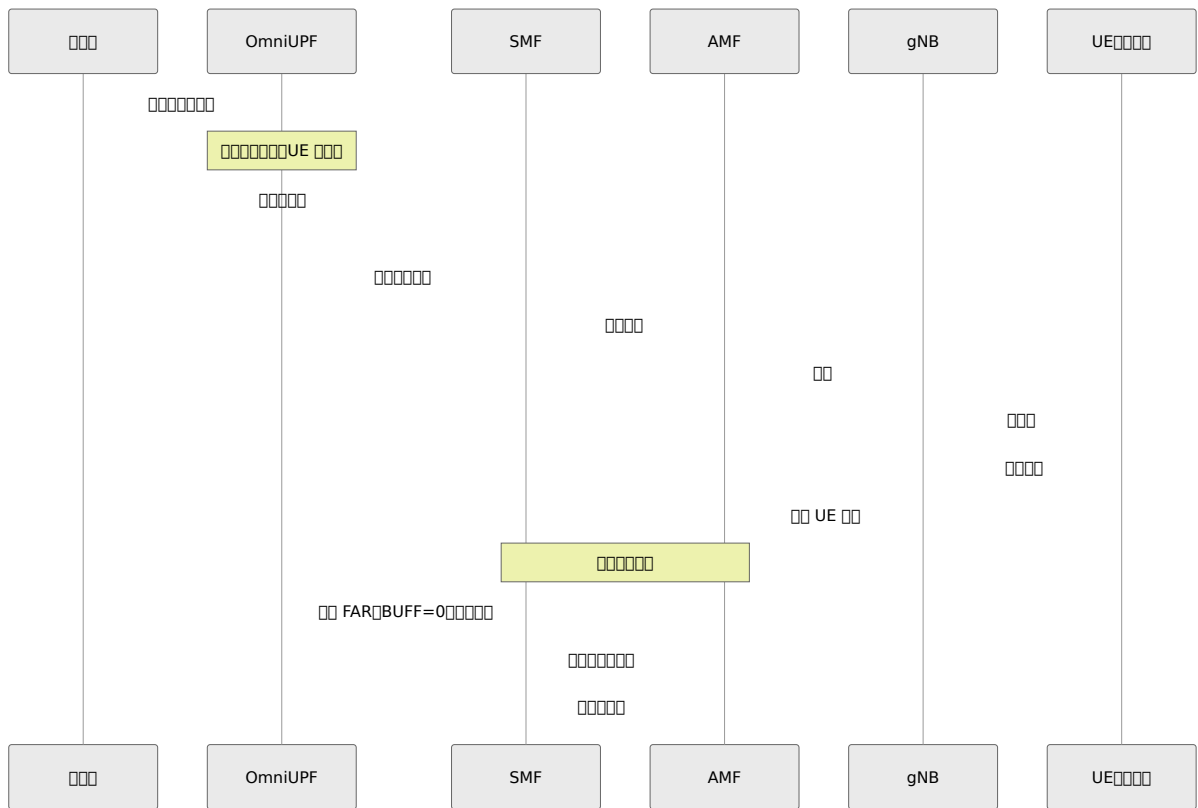
QoS Parameters

- **QoS** parameters for 4G and 5G NSA
- QoS parameters for 5G SA
- QoS parameters for UPF and ULCL

QoS parameters for 4G and 5G NSA

3. 5G Core Network Architecture

UE registration and session establishment



Registration and session establishment

4. RAT 4G ↔ 5G

UE 4G ↔ 5G

- eNodeB ↔ gNB
- TEID
- RAT

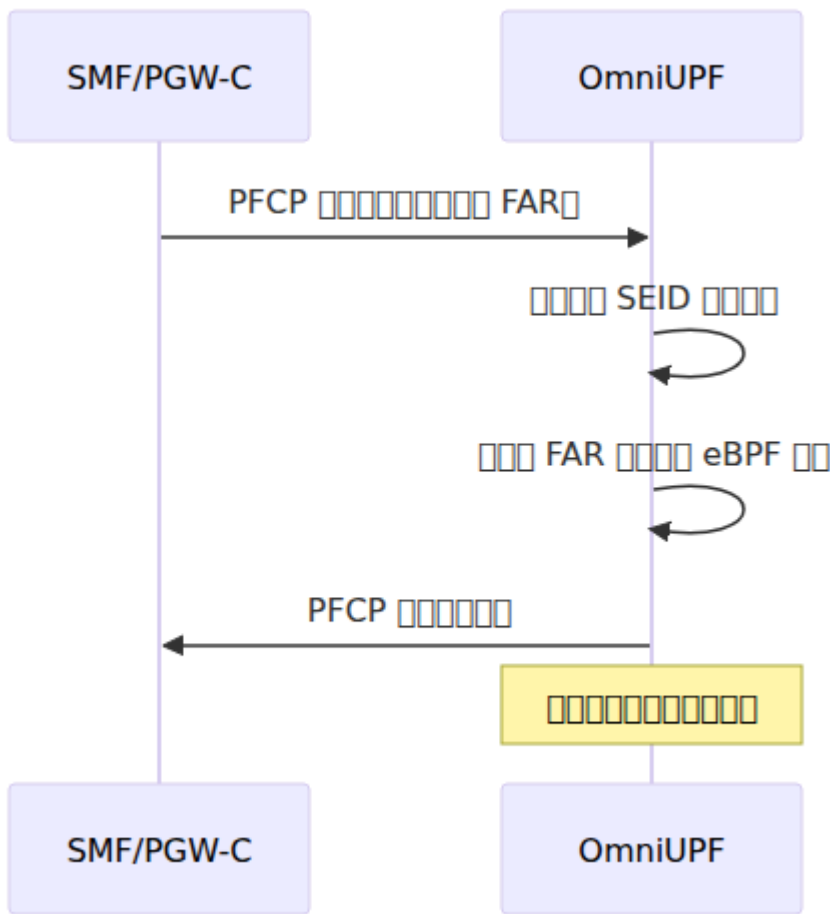
OmniUPF

OmniUPF

1. eBPF FAR

2. 22152 eBPF

22152



22152

- 22152 UDP 22152 eBPF
- GTP-U FAR ID TEID
- FAR ID
- - FAR 10,000
 - FAR 100,000
 - TTL 30 - TTL
- 60

22152

1. SMF PFCP FAR BUFF=1 2
2. eBPF BUFF 22152

3. 設定可能なFAR ID
 4. SMF設定可能なFAR 設定FORW=1、BUFF=0
 5. 設定可能なFAR
 6. 設定可能な
-

設定可能な

設定可能な

項目	設定可能な	設定可能な
設定可能な	設定可能な 1-2 設定可能な	設定可能な
設定可能な	設定可能な	設定可能な
設定可能な	設定可能な	設定可能な
設定可能な VoIP 設定	設定可能な	設定可能な
設定可能な	設定可能な	設定可能な
設定可能な	SSH 設定可能な	設定可能な

設定可能な

- 設定可能な**CDR**設定可能な KPI
 - 設定可能な
 - 設定可能な
 - 設定可能な“設定可能な”
-

設定可能な

設定可能な Web UI 設定可能な API 設定可能な

設定可能な

- **FAR ID** `00000000000000000000000000000000`
- `00000000000000000000000000000000` FAR `000000`
- `00000000000000000000000000000000`
- `00000000000000000000000000000000` > TTL `0000`

000000

- `00000000000000000000000000000000`
- `00000000000000000000000000000000`
- `00` **TTL** `0000000000000000`
- `0000000000` FAR `0000000000`

000000

- `0000000000` SMF `000000000000` FAR `00`
- `00000000000000000000000000000000`
- `0000000000` TTL `00000000` FAR `000000`
- `00000000000000000000000000000000` SMF `00`

00000000000000000000000000000000

000000

`/etc/omniupf/runtime.exs` `0000000000`

```
# 0000
buffer_port = 22152 # 00000000 UDP 00000000
```

0000

- `00000000000000000000000000000000` `buffer_max_packets` `0000` 20,000+
- `00000000000000000000000000000000` `buffer_packet_ttl` `0000` 15 `00`
- `0000000000` `buffer_packet_ttl` `0000` 10 `0000000000000000`
- `00000000000000000000000000000000`

00000000000000000000000000000000

□□□□□

□□□□□

□□□□□□□□□□

- □□□ **RX** □□□□□□□□□□□□
- □□□ **TX** □□□□□□□□□□□□
- □□□□□□□□□□□□□□□□
- **GTP-U** □□□□□□□□□□

□□□□□

□□□□□□□□□□

- □□□□□□□□□□□□
- □□□□□□□□□□□□/□□
- □□□□□□□ TEID□□□□ UE IP

XDP □□□

□□□□□□□□□□

- **XDP** □□□□ XDP □□□□□□□□
- **XDP** □□□□□□□□□□□□□□
- **XDP** □□□□ XDP □□□□□□□□
- **XDP** □□□□□□□□

N3/N6 □□□□□

□□□□□□□□□□

- **N3 RX/TX**□□□ RAN□gNB/eNodeB□□□□□
- **N6 RX/TX**□□□□□□□□□□□□
- □□□□□□□□□□□□□□

□□□□□□□□□□□ □□□□□

□□□□

eBPF □□□□□□

UPF □□□□ eBPF □□□□□□□□□□

- □□□□□□□□□□□□□□□□
- □□□□ **eBPF** □□□□□□□□
- □□□□□□□□
 - □□□ < 50% □□□□
 - □□□ 50-70% □□□□
 - □□□□ 70-90% □□□□
 - □□□ > 90% □□□□

□□□□□□□□□□

- `uplink_pdr_map` □□□□□□□□
- `downlink_pdr_map` □□□ IPv4 □□□□
- `far_map` □□□□□□
- `qer_map` □□ QoS □□
- `urr_map` □□□□□□

□□□□□□

- □□ PDR □□□□□□□□□□□□ + □□□□
- □□□□□□ UPF □□□□□□□□□□□□
- □□□□□□□□□□□□□□

□□□□□□□□□□ □□□□□□

□□□□

UPF □□□

□□□□□□ UPF □□□□□□

- **N3** 網元 RAN 網元 IP 網元GTP-U
- **N6** 網元網元網元網元 IP 網元
- **N9** 網元網元 UPF 網元網元 IP 網元網元
- **PCF** 網元 SMF 網元 IP 網元
- **API** 網元REST API 網元
- 網元網元Prometheus 網元

網元網元

網元 eBPF 網元網元

- 網元 **N3** 網元網元 N3 網元
- 網元 **N9** 網元網元 N9 網元網元網元

網元網元網元網元 網元

網元網元

網元網元網元網元網元網元

網元網元

網元PCF 網元網元UE 網元網元

網元網元

1. PCF 網元

- 網元 SMF 網元網元 UPF PCF 網元網元 8805
- 網元網元網元 PCF 網元
- 網元 ID 網元 SMF 網元 UPF 網元網元

2. eBPF 網元

- 網元網元網元網元網元>90%網元網元
- 網元 UPF 網元 eBPF 網元
- 網元網元網元網元

3. 詳細 PDR/FAR 詳細

- 詳細 UE IP 詳細
- 詳細 TEID 詳細
- 詳細 FAR 詳細

4. 詳細

- 詳細 N3 詳細 IP 詳細 gNB 詳細
- 詳細 N6 詳細
- 詳細 GTP-U 詳細

詳細詳細詳細詳細詳細

詳細詳細詳細詳細

詳細UE 詳細詳細詳細詳細

詳細

1. PDR 詳細

- 詳細 PDR TEID 詳細 gNB 詳細 TEID 詳細
- 詳細 PDR UE IP 詳細 IP 詳細
- 詳細 SDF 詳細

2. FAR 詳細

- 詳細 FAR 詳細
- 詳細
- 詳細

3. QoS 詳細

- 詳細 QER MBR 詳細
- 詳細 GBR 詳細
- 詳細

4. MTU

- GTP-U 40-50
 - N3/N6 MTU
 - ICMP
-

1.

- FAR 2
- SMF
-

2. TTL

-
- TTL
-

3.

- FAR
-
- max_per_far max_total

1. 100ms

- eBPF 100ms 64 100ms
- 100ms
- 100ms URR 100ms

2. 100ms

- 100ms eBPF 100ms
- 100ms eBPF 100ms
- 100ms XDP 100ms

3. 100ms

- 100ms N3/N6 100ms
- 100ms eBPF 100ms
- 100ms XDP 100ms

100ms

100ms CPU 100ms

100ms

1. 100ms **XDP** 100ms XDP 100ms
2. 100ms **eBPF** 100ms
3. 100ms **CPU** 100ms eBPF 100ms
4. 100ms NIC 100ms XDP 100ms

100ms

- **XDP** 100ms 10M+ 100ms
- **PDR** 100ms PDR 100ms
- 100ms UPF 100ms
- 100ms NIC 100ms

100ms 100ms

□□□□

□□□□□□□□

□□□□ UPF □□□□□□□□□□□□

□□□□

□□□□□□□□□□

- □□□□YAML□□□□□□CLI□
- □□□□UPF/PGW-U/SGW-U□
- XDP □□□□□□
- □□□□□□□□Proxmox□VMware□KVM□Hyper-V□VirtualBox□
- NIC □□□□ XDP □□□□□□
- □□□□□□□□□□
- □□□□□□□□□□

XDP □□□□

□□□ XDP □□□□□□□□□□

- XDP □□□□□□□□□□/□□/□□□□
- □□□□□□□□□□
- Proxmox VE □□ XDP □□□□□□□□
-

OmniUPF 介紹

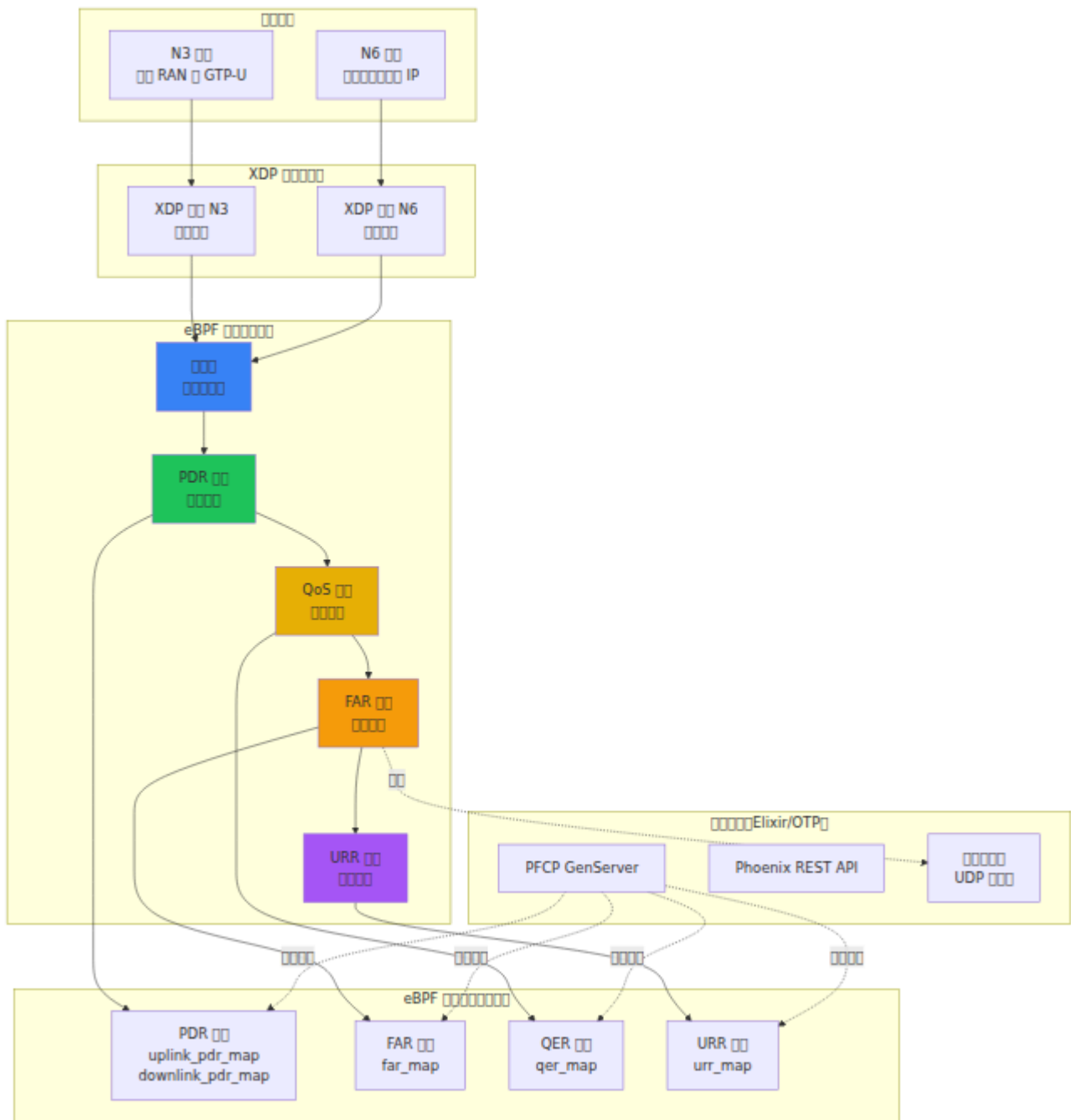
目錄

1. 介紹
2. eBPF 介紹
3. XDP 介紹
4. 網路架構
5. eBPF 應用
6. 安全
7. QoS 介紹
8. 性能
9. 部署與測試

簡介

OmniUPF 是一個基於 eBPF 和 XDP 的 5G/LTE 網路核心網元。它利用 Elixir/OTP 和 GenServer 實現了 PFCP 協議，並在 Linux 系統中利用 eBPF 實現了高性能的網路處理。OmniUPF 是一個基於 eBPF 和 XDP 的 5G/LTE 網路核心網元。它利用 Elixir/OTP 和 GenServer 實現了 PFCP 協議，並在 Linux 系統中利用 eBPF 實現了高性能的網路處理。

eBPF 依賴於 `ipentrypoint_bpf.o` C 庫和 `ebpf/xdp/` 目錄下的 NIF 庫。此外，還需要 `libbpf` 庫和 Elixir 的 `libbpf` 庫。



□□□□□□

□□□□□□

- □□□□□□□□□□□□
- □□□□□□□□□□□□□□
- □□ XDP □□□□□□□□

概要

- eBPF による CPU 最適化
- ネットワーク最適化
- システム/アプリケーション

導入

- XDP による SmartNIC 対応
- ネットワーク XDP 最適化
- システム最適化

eBPF 入門

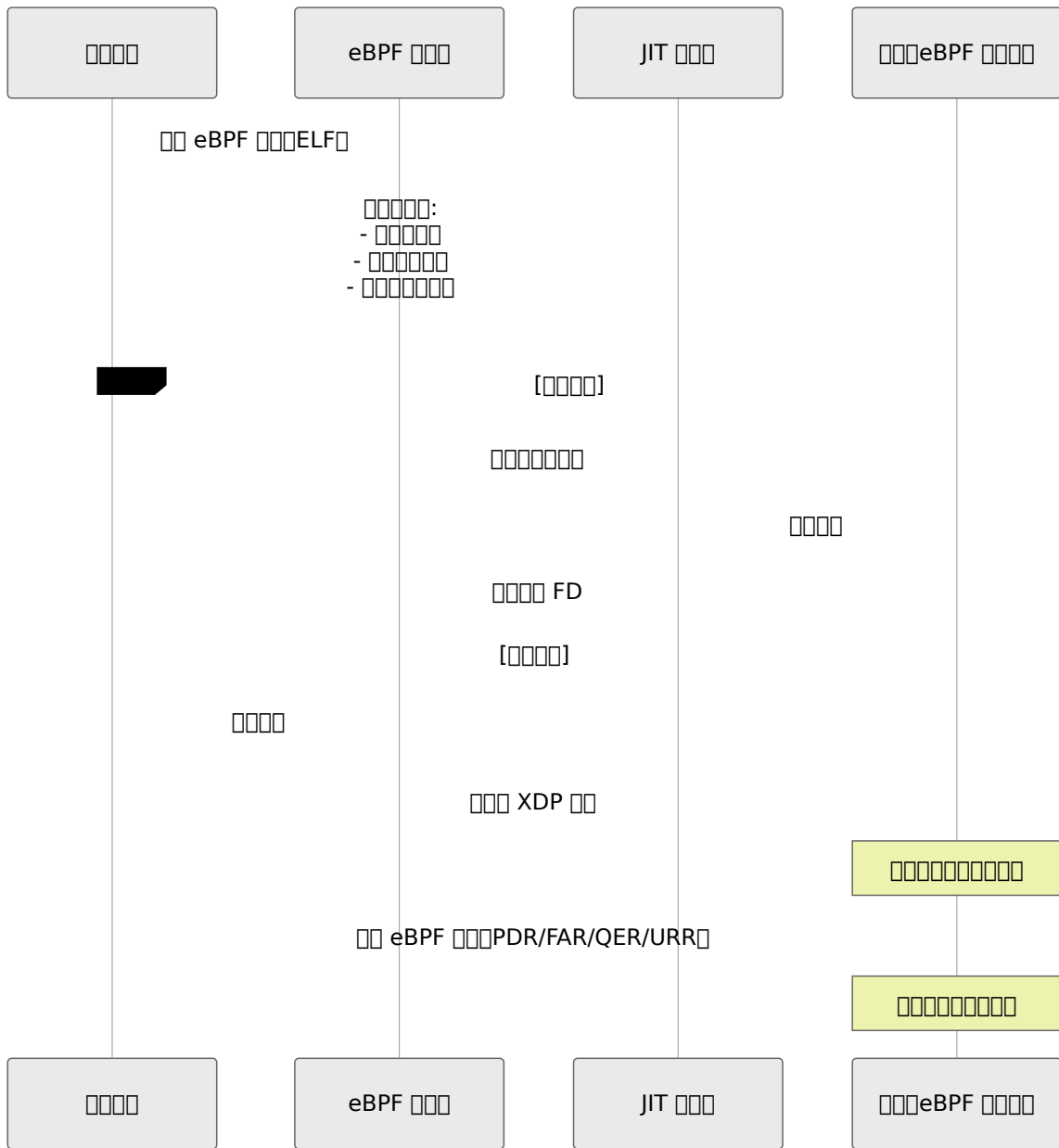
eBPF 入門

eBPFはLinuxカーネルの拡張機能で、Linuxカーネルの様々な部分で動作するプログラムを簡単に実装できる。これは、ネットワーク、システム、アプリケーションの最適化に役立つ。

概要

- eBPFプログラムの実行環境
- eBPFプログラムの実装方法
- eBPFプログラムのデバッグ方法
- eBPFプログラムの最適化方法

eBPF 架构图



eBPF 钩子

eBPF 钩子用于在内核态中插入自定义的 eBPF 程序

OmniUPF 架构图

名称	键	值
BPF_MAP_TYPE_HASH	键值对	TEID UE IP PDR
BPF_MAP_TYPE_ARRAY	数组	ID QER FAR URR
BPF_MAP_TYPE_PERCPU_HASH	每CPU键值对	PDR
BPF_MAP_TYPE_LRU_HASH	LRU键值对	

特点

- O(1) 查找
- 支持多种键值对
- 支持数组
- 支持每CPU键值对

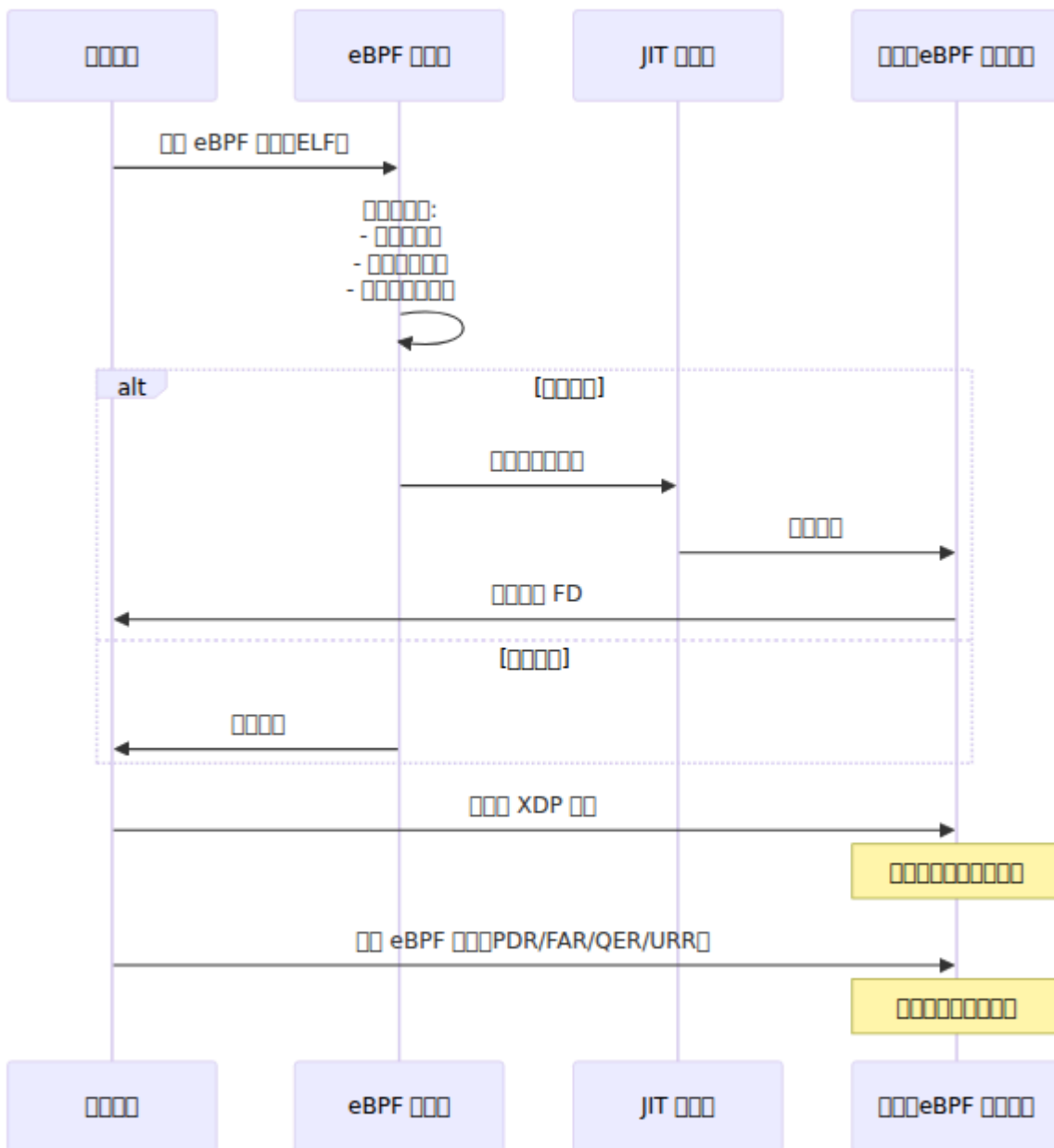
XDP 简介

XDP 是什么

XDP 是 Linux 内核中 eBPF 的一种应用，用于在用户态或内核态对网络数据包进行快速处理。

XDP 的应用

OmniUPF 利用 XDP 实现高性能的流量转发和策略控制。



1. XDP

SmartNIC

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

runtime.exs

```
xdp_attach_mode = "offload"
```

□□□

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

2. XDP □□□□□□□□□□□□□□

□□□□□□□□□□□□

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

□□□□ runtime.exe □□□

```
xdp_attach_mode = "native"
```

□□□

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

□□□□□□□□□

- Intel: i40e, ice, ixgbe, igb
 - Mellanox: mlx4, mlx5
 - Broadcom: bnxt
 - Amazon: ena
 - □□□ 10G+ □□□
-

3. XDP 简介

简介

- eBPF 处理 SKB 数据
- XDP 简介
- 性能提升
- 应用场景

配置 `runtime.exs` 文件

```
xdp_attach_mode = "generic"
```

优点

- 性能提升
- 支持 SR-IOV 虚拟化
- 灵活配置
- 易于集成

支持 1-5 Gbps 网络接口/网卡

XDP 进阶

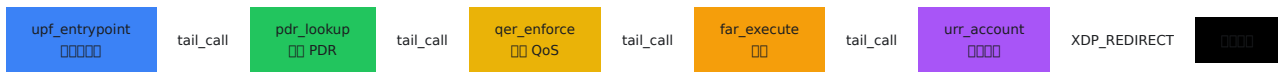
eBPF 与 XDP 的结合使用

状态	原因	OmniUPF 处理
XDP_PASS	数据包通过	数据包通过 ICMP 数据包
XDP_DROP	数据包被丢弃	数据包被丢弃
XDP_TX	数据包被发送	数据包
XDP_REDIRECT	数据包被重定向	数据包被重定向 N3 ↔ N6
XDP_ABORTED	数据包被中止	eBPF 中止

数据包

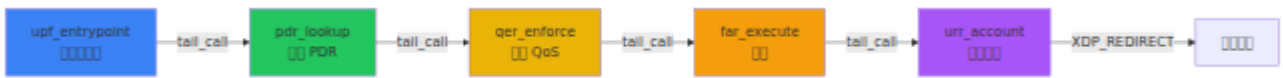
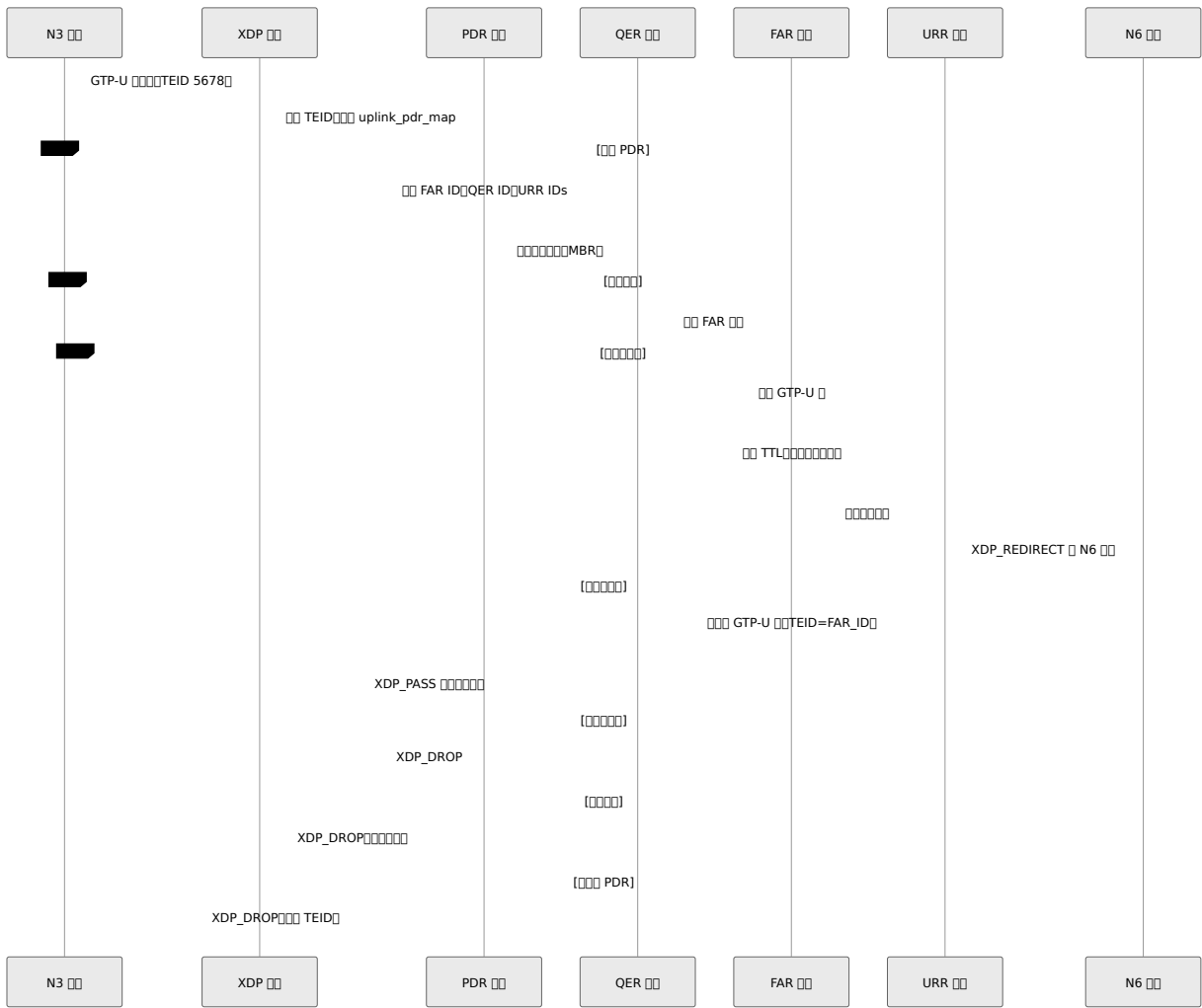
数据包

OmniUPF 使用 eBPF 处理数据包



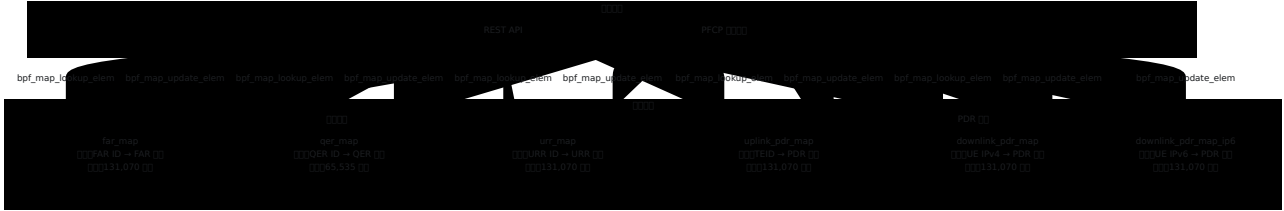
数据包

- 使用 eBPF 处理数据包 eBPF 处理
- 数据包被丢弃
- 数据包被发送
- 使用 33 个数据包



eBPF

□□□□□□



□□□□

OmniUPF `max_sessions` □□□□□□□□□□□□

```

PDR □□ = 2 × max_sessions (□□ + □□)
FAR □□ = 2 × max_sessions (□□ + □□)
QER □□ = 1 × max_sessions (□□□□□)
URR □□ = 3 × max_sessions (□□□□□ URR)
    
```

□□□max_sessions = 65,535□□

- PDR □□□□□ 131,070 □□
- FAR □□□131,070 □□
- QER □□□65,535 □□
- URR □□□131,070 □□

□□□□

```

PDR □□□3 × 131,070 × 212 B = ~83 MB
FAR □□□ 131,070 × 20 B = ~2.6 MB
QER □□□ 65,535 × 36 B = ~2.3 MB
URR □□□ 131,070 × 20 B = ~2.6 MB
□□□~91 MB □□□□
    
```

□□□□

□□□□

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

□□□□

Parse error on line 10: ...□□□□
□□□□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; // 0000000000
ip->daddr = local_upf_ip;       // 000000000000 IP
ip->protocol = IPPROTO_UDP;
ip->ttl = 64;

```

4. UDP

```

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

```

5. GTP-U

```

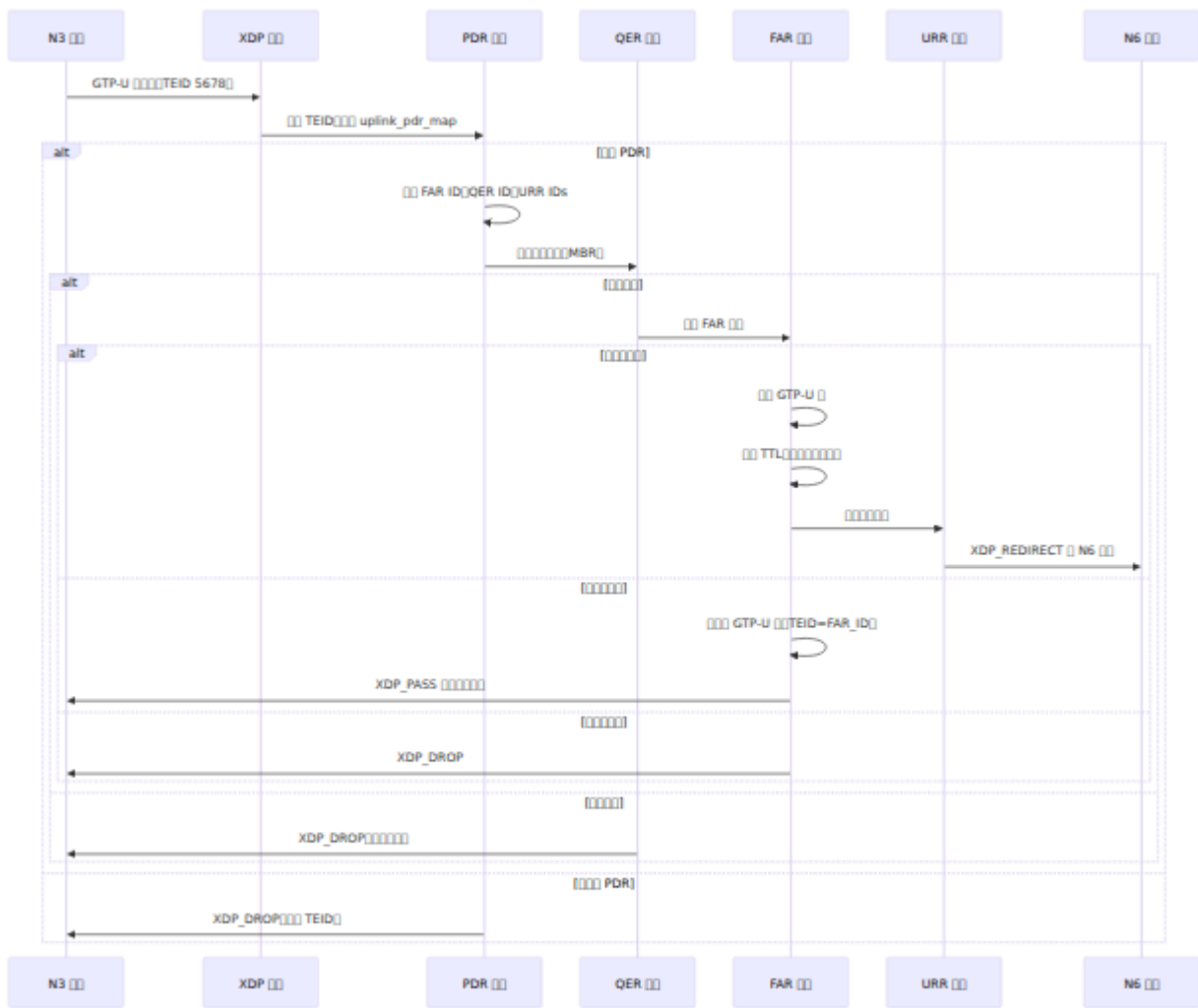
gtp->version = 1;
gtp->message_type = GTPU_G_PDU;
gtp->teid = htonl(far_id | (direction << 24)); // FAR ID
gtp->message_length = htons(orig_packet_len);

```

6. XDP_PASS

- UDP 22152
-

SMF FAR



□□□□□□

□□	□□□	□□
□□ FAR □□□	10,000 □□□	□□ FAR □□□□□□□□
□□□□	100,000 □□□	□□□□□□□□□□
□□□ TTL	30 □	□□□□□□□□□□
□□□□	22152	□□□□□□□ UDP □□
□□□□□□	60 □	□□□□□□□□□□

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;

    // calculate tx_time
    __u64 tx_time = packet_size * 8 * (NSEC_PER_SEC / rate);
    __u64 now = bpf_ktime_get_ns();

    // calculate start
    __u64 start = *window_start;
    if (start + tx_time > now)
        return XDP_DROP; // rate limit

    // update window_start
    if (start + window_size < now) {
        *window_start = now - window_size + tx_time;
        return XDP_PASS;
    }

    // update window_start
    *window_start = start + tx_time;
    return XDP_PASS;
}

```

□□□□

- □□□□5ms□5,000,000 □□□
- □□□□□□□□□□□□□□□□
- □□□□□□□□□□□□□□□□
- **MBR = 0**□□□□□□□□

QoS 计算

链路 MBR = 100 Mbps 帧大小 = 1500 字节

1. 帧大小

$$\begin{aligned} \text{tx_time} &= 1500 \text{ 字节} \times 8 \text{ 位/字节} \times (1,000,000,000 \text{ ns/sec} \div 100,000,000 \text{ bps}) \\ \text{tx_time} &= 1500 \times 8 \times 10 = 120,000 \text{ ns} = 120 \mu\text{s} \end{aligned}$$

2. 帧间隔

- 帧间隔 $t=0$ 帧间隔 $t=120\mu\text{s}$ 帧
- 帧间隔 $t=100\mu\text{s}$ 帧
- 帧间隔 $t=150\mu\text{s}$ 帧

3. 最大帧速率

$$\begin{aligned} \text{Max PPS} &= (100 \text{ Mbps} \div 8) \div 1500 \text{ 字节} = 8,333 \text{ 帧/秒} \\ \text{帧间隔} &= 120 \mu\text{s} \end{aligned}$$

性能

性能

配置	速率	帧速率	延迟
XDP 智能NIC	100 Gbps	148 Mpps	< 1 μs
XDP 10G NIC	10 Gbps	8 Mpps	2-5 μs
XDP 10G NIC 4 核	40 Gbps	32 Mpps	2-5 μs
XDP 普通	1-5 Gbps	0.8-4 Mpps	50-100 μs

遅延

ネットワーク遅延とXDP遅延

処理	遅延	遅延
NIC RX	0.5 μ s	0.5 μ s
XDP 遅延	0.1 μ s	0.6 μ s
PDR 遅延	0.3 μ s	0.9 μ s
QER 遅延	0.1 μ s	1.0 μ s
FAR 遅延	0.5 μ s	1.5 μ s
URR 遅延	0.2 μ s	1.7 μ s
GTP-U 遅延	0.8 μ s	2.5 μ s
XDP_REDIRECT	0.5 μ s	3.0 μ s
NIC TX	0.5 μ s	3.5 μ s

ネットワーク遅延 ~3.5 μ s XDP 遅延 10G NIC

CPU 遅延

遅延

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

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 Mpps \times (1500 B + 3 \times 64 B) \approx 160 Gbps

UPF

Setting SMF as parent of SMF would create a cycle

- SMF \rightarrow UPF
- UPF \rightarrow UE
- UPF \rightarrow SMF

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

```
# eBPF 前提条件
ulimit -l unlimited

# XDP 前提条件 IRQ 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: $(20 \text{ Gbps} \div 10 \text{ Gbps}) \times 1.5 = 3-4$
- $(1 \text{M} \times 212 \text{ B} \times 3) + 100 \text{ MB} \approx 750 \text{ MB}$
- $(20 \text{ Gbps} \times 2) + 10 \text{ Gbps} = 50 \text{ Gbps}$

□□□□

- **UPF** □□□□ - □□ UPF □□□□
- □□□□□□ - PDR□FAR□QER□URR □□□□
- □□□□ - □□□□□□□□
- **Web UI** □□□□ - □□□□□□
- □□□□□□ - □□□□□□□□
- □□□□□□ - □□ BUFF □□□□□□□□□□□□□□□□□□□□

OmniUPF 架构图

简介

- 简介
- 架构图
- XDP 架构图
- 架构图
- 架构图
- 架构图
- 架构图
- 架构图
- 架构图
- 架构图

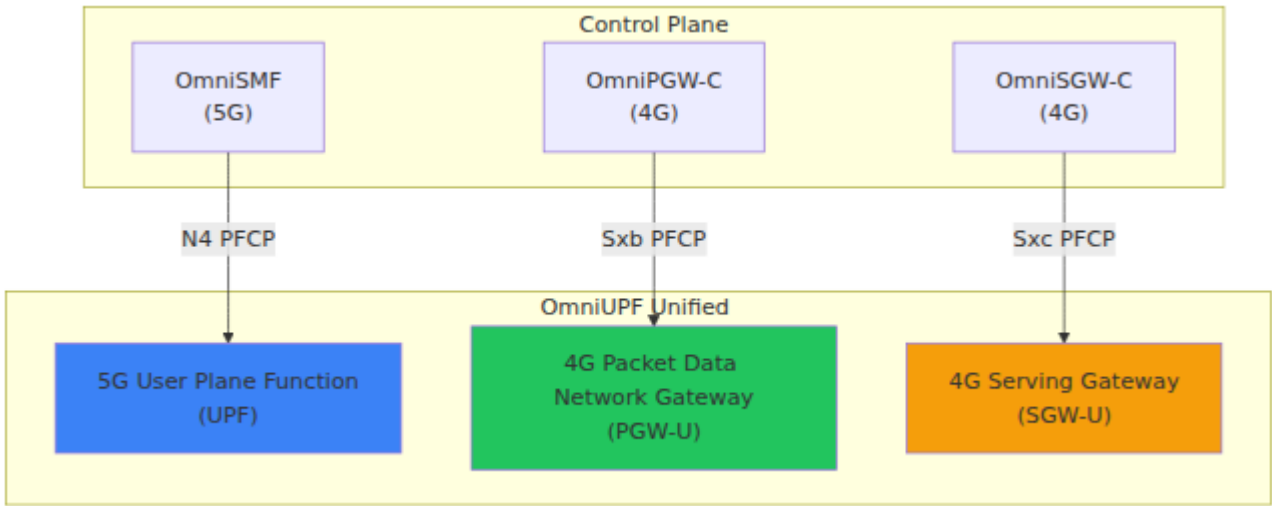
部署

OmniUPF 部署在 4G (EPC) 和 5G 网络中，通过 Elixir/OTP 部署 Elixir 进程 (runtime.exs)。

部署命令：`./etc/omniupf/runtime.exs` 部署 UPF。

配置

OmniUPF 配置在 `config.exs` 文件中。



목적

이 레퍼토리에서는 (SMF, PGW-C, SGW-C)와 연결되는 OmniUPF의 PCFP 구성 방법을 설명합니다.
 이 레퍼토리에서는 OmniUPF를 구성합니다.

구성

- OmniUPF 구성을 위한 PCFP 구성
- OmniUPF 구성을 위한 UPF, PGW-U, SGW-U 구성
- 필요한 구성 요소

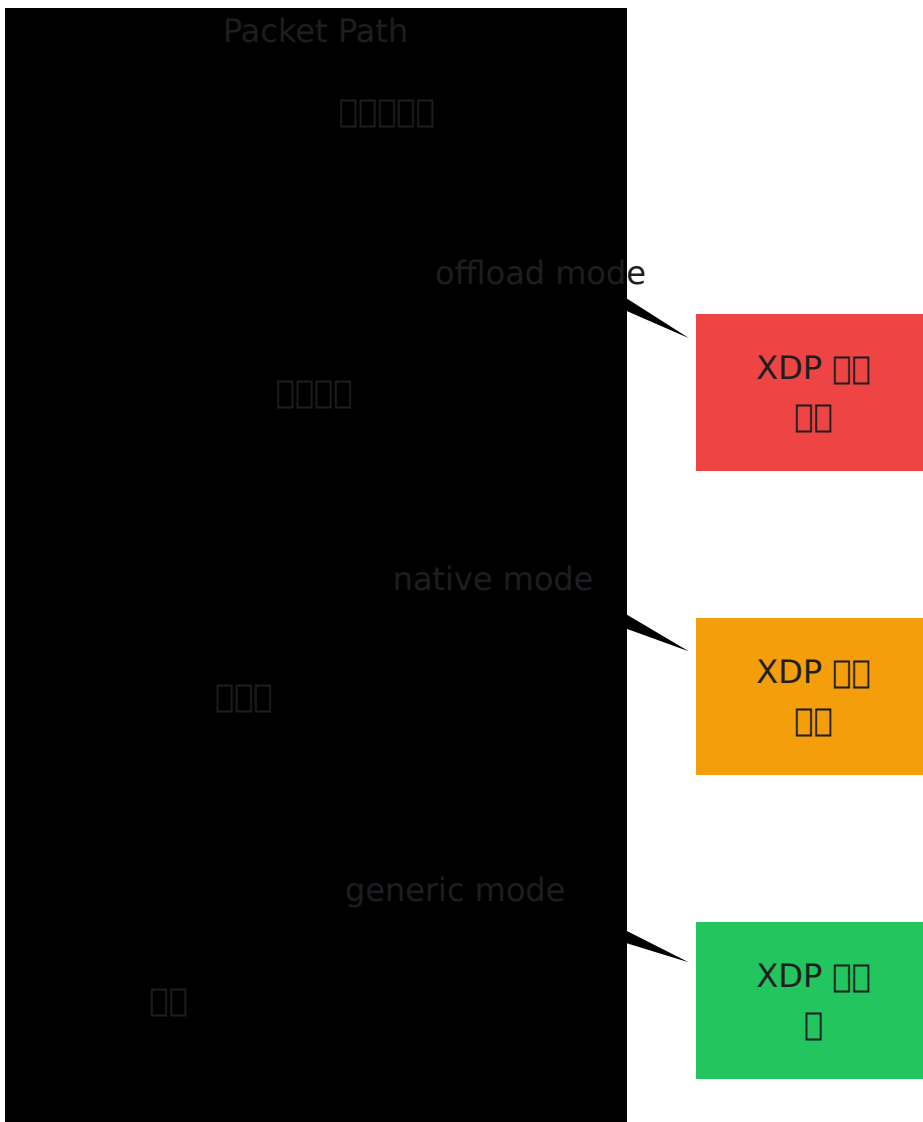
XDP 레퍼토리

OmniUPF의 XDP (eXpress Data Path) 구성 방법을 설명합니다.

이 레퍼토리는 XDP를 구성하는 Proxmox를 구성하는 XDP 레퍼토리입니다.

网卡

网卡	网卡	性能	技术	NIC 技术
网卡	网卡 (网卡)	~1-2 Mpps	网卡	网卡 NIC
网卡	网卡 (网卡)	~5-10 Mpps	网卡 (网卡 SR-IOV 网卡)	网卡 XDP 网卡
网卡	NIC 网卡 (SmartNIC)	~10-40 Mpps	网卡	网卡 XDP 网卡 SmartNIC



网卡 (NIC)

网卡XDP 性能提升

网卡

- 网卡 性能 提升
- 网卡性能提升
- 网卡性能
- 网卡性能提升

网卡

- 网卡 (~1-2 Mpps 性能)
- 网卡 XDP 性能提升

网卡 (runtime.exs)

```
xdp_attach_mode = "generic"
```

网卡

- 网卡 SR-IOV 性能
- 网卡性能
- 网卡 XDP 性能 NIC
- Proxmox/VMware/VirtualBox 性能提升

网卡 (NIC)

网卡XDP 性能提升

网卡

- 网卡 (~5-10 Mpps 性能)
- 网卡性能提升
- 网卡性能提升

- SR-IOV 支持

支持

- XDP 支持
- NIC/支持 XDP

支持 (runtime.exs 支持)

```
xdp_attach_mode = "native"
```

支持

- 支持
- SR-IOV 支持
- XDP 支持 NIC (Intel/Mellanox)

支持

- XDP 支持 (支持)
- Linux 5.15+ 支持 XDP 支持

支持 (支持)

支持 XDP 支持 SmartNIC 支持

支持

- 支持 (~10-40 Mpps)
- 支持 CPU 支持
- 支持
- CPU 支持

支持

- 支持 SmartNIC 支持
- SmartNIC 支持

- `runtime.exe`

`runtime.exe`

```
xdp_attach_mode = "offload"
```

`runtime.exe`

- `runtime.exe`
- `runtime.exe`
- CPU `runtime.exe`

`runtime.exe`

- `runtime.exe` XDP `runtime.exe` SmartNIC (Netronome Agilio CX/Mellanox BlueField)
- `runtime.exe`

`runtime.exe`

`runtime.exe`

Key	Description	Type	Value
<code>xdp_interfaces</code>	N3/N6/N9 <code>runtime.exe</code> (N3/N6/N9 <code>runtime.exe</code>)	String	"auto"
<code>n3_address</code>	N3 <code>runtime.exe</code> IPv4 <code>runtime.exe</code> (N3 RAN <code>runtime.exe</code> GTP-U)	IP	"127.0.0.1"
<code>n9_address</code>	N9 <code>runtime.exe</code> IPv4 <code>runtime.exe</code> (UPF <code>runtime.exe</code> UPF <code>runtime.exe</code> ULCL)	IP	<code>runtime.exe</code> <code>runtime.exe</code> <code>runtime.exe</code>

`runtime.exe`

```

xdp_interfaces = "eth0,eth1"
n3_address = "10.100.50.233"
n9_address = "10.100.50.234"

```

PFCP

Parameter	Description	Unit	Value
<code>pfcp_address</code>	PFCP Controller Address (N4/Sxb/Sxc)	IP Address	"127.0.0.1"
<code>pfcp_port</code>	PFCP Port	Port	8805
<code>node_id</code>	PFCP Controller ID	IP Address	"127.0.0.1"
<code>heartbeat_interval_ms</code>	PFCP Heartbeat Interval (ms)	ms	5000
<code>heartbeat_timeout_ms</code>	PFCP Heartbeat Timeout (ms)	ms	5000
<code>heartbeat_retries</code>	Heartbeat Retries	Count	3

Example (`runtime.exs`):

```

pfcp_address = "10.100.50.241"
pfcp_port = 8805
node_id = "10.100.50.241"
heartbeat_interval_ms = 10_000
heartbeat_retries = 5

```

API 設定

項目	説明	型	デフォルト値
<code>api_port</code>	REST API 接続先 (Phoenix)	整数	8080
<code>log_level</code>	ログレベル (:debug, :info, :warning, :error)	文字列	:info

設定 (`runtime.exs`)

```
api_port = 8080
log_level = :debug
```

GTP 設定

項目	説明	型	デフォルト値
<code>gtpu_port</code>	GTP-U 接続先	整数	2152
<code>gtp_echo_interval</code>	エコー間隔	整数	10
<code>gtp_echo_retries</code>	エコーリトライ回数	整数	3
<code>gtp_peers</code>	GTP 接続先リスト	リスト	[]

設定 (`runtime.exs`)

```
gtp_echo_interval = 15
gtp_echo_retries = 3
gtp_peers = [
  %{address: parse_ip("10.100.50.50"), echo: true, echo_interval:
  10},
  %{address: parse_ip("10.100.50.60"), echo: false},
]
```

eBPF 参数

参数	描述	单位	默认值
<code>max_sessions</code>	会话数限制	个	<code>1_000_000</code>
<code>far_map_size</code>	FAR eBPF 表大小	个	<code>131_070</code>
<code>qer_map_size</code>	QER eBPF 表大小	个	<code>65_535</code>
<code>urr_map_size</code>	URR eBPF 表大小	个	<code>65_535</code>

配置 (在 `runtime.exs` 中)

```
max_sessions = 100_000
far_map_size = 131_070
qer_map_size = 65_535
urr_map_size = 131_070
```

端口

参数	描述	单位	默认值
<code>buffer_port</code>	eBPF 缓冲区 UDP 端口	个	<code>22152</code>

配置 (在 `runtime.exs` 中)

```
buffer_port = 22152
```

Configuration

Parameter	Description	Default	Value
<code>feature_ueip</code>	OmniUPF UE IP		<code>false</code>
<code>ueip_pool</code>	UE IP pool (CIDR) (requires <code>feature_ueip</code>)	CIDR	<code>"10.60.0.0/24"</code>
<code>feature_ftup</code>	OmniUPF F-TEID		<code>true</code>
<code>teid_pool_start</code>	TEID start		<code>1</code>
<code>teid_pool_end</code>	TEID end		<code>1_000_000</code>

Example (UE IP) (`runtime.exs`)

```
feature_ueip = true
ueip_pool = "10.45.0.0/16"
```

Example (F-TEID) (`runtime.exs`)

```
feature_ftup = true
teid_pool_start = 1
teid_pool_end = 1_000_000
```

Free Range Routing

Free Range Routing (FRR) allows UE to route traffic through the network.

名前	説明	タイプ	デフォルト値
<code>route_manager_enabled</code>	UE のルートマネージャを有効にする	ブール値	<code>true</code>
<code>route_manager_type</code>	ルートマネージャの種類 ("frr" または "static")	文字列	<code>"frr"</code>

例 (`runtime.exs`)

```
route_manager_enabled = true
route_manager_type = "frr"
```

影響

- 有効化された UPF
 - OSPF または BGP ルーティング
 - FRRouting モジュール
-

eBPF / XDP

Key	Description	Type	Default Value
<code>xdp_interfaces</code>	Which XDP interfaces to load on. "auto" means all interfaces.	String	"auto"
<code>xdp_attach_mode</code>	XDP attach mode. "generic" is the default, "native" is for high performance, "offload" is for hardware offload.	String	"generic"
<code>ebpf_pin_path</code>	Path to the BPF pin directory.	String	"/sys/fs/bpf/upf_pipeline"
<code>xdp_obj_path</code>	Path to the eBPF object file.	String	"/etc/omniupf/ipentrypoint_bpf.o"

Example configuration in `runtime.exs`:

```

xdp_interfaces = "auto" # Load on "eth0,eth1" interfaces
xdp_attach_mode = "native"
ebpf_pin_path = "/sys/fs/bpf/upf_pipeline"
xdp_obj_path = "/etc/omniupf/ipentrypoint_bpf.o"

```

Next steps:

Elixir (runtime.exs)

File path: `/etc/omniupf/runtime.exs`


```

import Config

parse_ip = fn str ->
  {:ok, addr} = :inet.parse_address(String.to_charlist(str))
  addr
end

#
=====
# PFCP (N4/Sx) []
#
=====
pfcip_address = "10.100.50.241"      # [] PFCP [] IP []
pfcip_port = 8805                    # PFCP [] ([]: 8805)
node_id = "10.100.50.241"          # [] ID

#
=====
# GTP-U []
#
=====
n3_address = "10.100.50.233"        # N3 [] IP (GTP-U [] RAN)
n9_address = n3_address              # N9 [] IP ([] [] N3)
gtpu_port = 2152                     # GTP-U [] ([]: 2152)
buffer_port = 22152                  # []

#
=====
# eBPF / XDP []
#
=====
xdp_interfaces = "auto"              # "auto" []
xdp_attach_mode = "native"          # XDP []: "generic"|"native" []
ebpf_pin_path = "/sys/fs/bpf/upf_pipeline"
xdp_obj_path = "/etc/omniupf/ipentrypoint_bpf.o"

#
=====
# []
#
=====
max_sessions = 100_000
teid_pool_start = 1

```

```
teid_pool_end = 10_000_000
far_map_size = 131_070
qer_map_size = 65_535
urr_map_size = 65_535

#
=====
# 
#
=====
feature_ueip = true
feature_ftup = true
ueip_pool = "10.45.0.0/16"

#
=====
# 
#
=====
route_manager_enabled = true
route_manager_type = "frr"

#
=====
# 
#
=====
heartbeat_interval_ms = 10_000
heartbeat_timeout_ms = 5_000
heartbeat_retries = 5

#
=====
# GTP-U 
#
=====
gtp_echo_interval = 10
gtp_echo_retries = 3
gtp_peers = [
    %{address: parse_ip("10.100.50.50"), echo: true, echo_interval: 10
]

#
=====
```

```
# HTTP API
#
=====
api_port = 8080
log_level = :info

#
=====
# ( )
#
=====
config :upf_ex,
  pfcf_address: parse_ip.(pfcf_address),
  pfcf_port: pfcf_port,
  n3_address: parse_ip.(n3_address),
  n9_address: parse_ip.(n9_address),
  node_id: parse_ip.(node_id),
  api_port: api_port,
  ebpf_pin_path: ebpf_pin_path,
  xdp_obj_path: xdp_obj_path,
  interface_names: String.split(xdp_interfaces, ","),
  xdp_attach_mode: xdp_attach_mode,
  feature_ueip: feature_ueip,
  feature_ftup: feature_ftup,
  ueip_pool: ueip_pool,
  teid_pool_start: teid_pool_start,
  teid_pool_end: teid_pool_end,
  far_map_size: far_map_size,
  qer_map_size: qer_map_size,
  urr_map_size: urr_map_size,
  max_sessions: max_sessions,
  route_manager_enabled: route_manager_enabled,
  route_manager_type: route_manager_type,
  buffer_port: buffer_port,
  gtpu_port: gtpu_port,
  heartbeat_interval_ms: heartbeat_interval_ms,
  heartbeat_timeout_ms: heartbeat_timeout_ms,
  heartbeat_retries: heartbeat_retries,
  gtp_echo_interval: gtp_echo_interval,
  gtp_echo_retries: gtp_echo_retries,
  gtp_peers: gtp_peers

config :logger, level: log_level
```

命令

```
# UPF 启动
sudo systemctl start omniupf

# UPF 停止
sudo systemctl stop omniupf

# 重启
sudo systemctl restart omniupf

# 查看状态
sudo systemctl status omniupf

# 查看日志
sudo journalctl -u omniupf -f

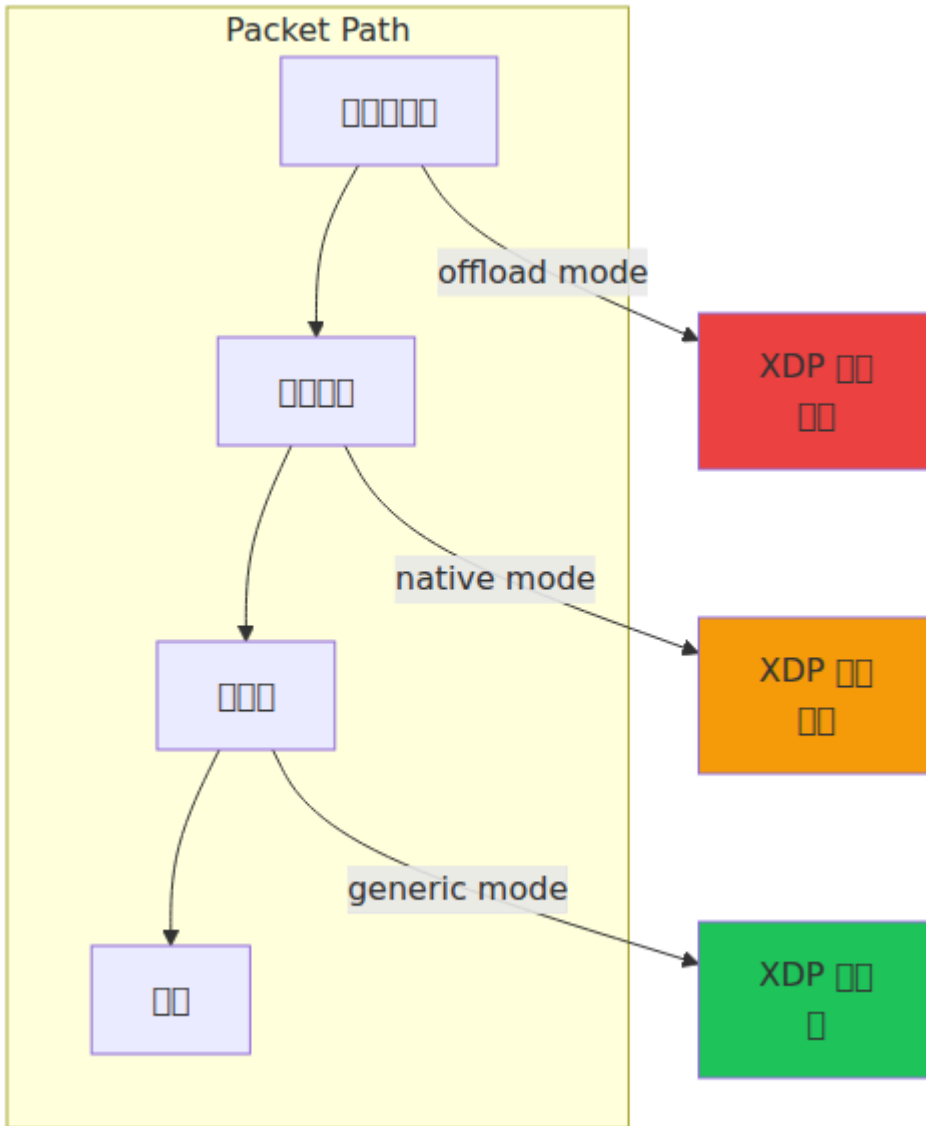
# 手动启动/停止/守护进程
sudo /opt/omniupf/bin/upf_ex start
sudo /opt/omniupf/bin/upf_ex stop
sudo /opt/omniupf/bin/upf_ex daemon # 守护进程
```

安装与配置

安装

OmniUPF 安装与配置 XDP 配置

在 Proxmox 中安装 XDP 配置



Proxmox VE

□□□□□□

1. □□□□ (□□ XDP)

□□□□□□□□□□

□□□

- □□□□□ VirtIO □ E1000
- XDP □□□ generic
- □□□ ~1-2 Mpps

Proxmox 設定

```
iface: net0  
type: VirtIO (virtio)  
bridge: vbr0
```

OmniUPF 設定 (runtime.exs)

```
xdp_interfaces = "eth0"  
xdp_attach_mode = "generic"
```

2. SR-IOV 設定 (XDP)

概要

特徴

- SR-IOV 対応 NIC
- XDP モード native
- 処理速度 ~5-10 Mpps

要件

- SR-IOV 対応 NIC (Intel X710/Mellanox ConnectX-5)
- BIOS で SR-IOV を有効にする
- IOMMU (intel_iommu=on / amd_iommu=on / GRUB)

Proxmox での SR-IOV 設定

```

# 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

# 配置 NIC 使用 VFs (例如: eth0 使用 4 个 VFs)
echo 4 > /sys/class/net/eth0/device/sriov_numvfs

# 设置开机脚本
echo "echo 4 > /sys/class/net/eth0/device/sriov_numvfs" >>
/etc/rc.local
chmod +x /etc/rc.local

```

Proxmox 配置

```

CPU -> CPU -> PCI 配置
CPU: SR-IOV 配置
CPU: 配置
CPU GPU: 配置
PCI-Express: 配置 (配置)

```

OmniUPF 配置 (在 `runtime.exs` 中)

```

xdp_interfaces = "ens1f0"      # SR-IOV VF 配置
xdp_attach_mode = "native"

```

3. PCI 配置 (配置 XDP)

配置 NIC

配置

- 配置 NIC 配置

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

Proxmox `PCI`

```

NIC -> PCI NIC
NIC: PCI NIC (MAC: 0000:01:00.0)
NIC: PCI
NIC GPU: PCI
PCI-Express: PCI

```

OmniUPF `runtime.exs`

```

xdp_interfaces = "ens1f0"
xdp_attach_mode = "native" # "offload" SmartNIC

```

KVM/QEMU

`virt-install`

```

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

```

SR-IOV `XML`

```

<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
 - 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
-

1 Mpps packets ↔ packets GTP-U packets

$$1536 \text{ packets} \times 1,000,000 \text{ pps} \times 8 \text{ bytes/packet} = 12,288 \text{ Mbps} \sim 12.3 \text{ Gbps}$$

IP packets (N6 packets)

N6 (packets) packets IP packets GTP-U

N6 packets

- IP packets 20 packets
- UDP packets 8 packets
- packets 1 packets
- packets 29 packets

1 Mpps packets packets N6 packets

$$29 \text{ packets} \times 1,000,000 \text{ pps} \times 8 \text{ bytes/packet} = 232 \text{ Mbps}$$

N6 packets (1500 MTU)


- IP MTU packets 1500 packets
- packets 1500 packets

1 Mpps packets packets N6 packets

$$1500 \text{ packets} \times 1,000,000 \text{ pps} \times 8 \text{ bytes/packet} = 12,000 \text{ Mbps} = 12 \text{ Gbps}$$

packets

Intel X710 NIC (N3 packets 10 Mpps packets GTP-U)

□□□□	□□□□□□□□	GTP-U □□	□ 10 Mpps □ □□□□	□□□□
VoIP □□ (N3)	65-150 □□	101-186 □ □	0.8-1.5 Gbps	AMR-WB □□□ G.711
□□□□□ (N3)	400-600 □ □	436-636 □ □	3.5-5.1 Gbps	HTTP/HTTPS□□□
□□□□ (N3)	1200 □□	1236 □□	9.9 Gbps	□□ 2024 □□□□
□□  (N3)	1400-1450 □□	1436-1486 □□	11.5-11.9 Gbps	HD/4K □□□
□□ MTU (N3)	1500 □□	1536 □□	12.3 Gbps	□□ TCP □□

□ **N6** □□ (□□ IP□□ **GTP-U**)□

□□□□	□□□□□□	□ 10 Mpps □□□□□□	□□□□
VoIP □□□□	65-150 □□	0.5-1.2 Gbps	□□ RTP □
□□□□□□	400-600 □□	3.2-4.8 Gbps	HTTP □□
□□□□	1200 □□	9.6 Gbps	□□ 2024 □□
□□□	1400-1450 □□	11.2-11.6 Gbps	□□□□
□□ MTU	1500 □□	12.0 Gbps	□□□□□□

□ **10 Mpps** □□□□□□□□□□ (1200 □□□□)□□□□ ~10 Gbps □□□ □ N3 □ N6 □□□□

□□□□□□□□

□ 1200 □□□□□□□□□□ (□□□□□□□□□□□□□□□)□

NIC Mpps	N3 (GTP-U)	N6 (IP)	
1 Mpps	~1.0 Gbps	~1.0 Gbps	
5 Mpps	~4.9 Gbps	~4.8 Gbps	
10 Mpps	~9.9 Gbps	~9.6 Gbps	
20 Mpps	~19.7 Gbps	~19.2 Gbps	
40 Mpps	~39.4 Gbps	~38.4 Gbps	

XDP

OmniUPF uses XDP to process packets in the NIC

Intel NICs

		XDP		
Intel X710	i40e			~10 Mpps
Intel XL710	i40e			~10 Mpps
Intel E810	ice			~15 Mpps
Intel 82599ES	ixgbe			~8 Mpps
Intel I350	igb			~1 Mpps
Intel E1000	e1000			~1 Mpps

Mellanox/NVIDIA NICs

网卡	驱动	XDP 支持	性能	吞吐量
Mellanox ConnectX-5	mlx5	支持	支持	~12 Mpps
Mellanox ConnectX-6	mlx5	支持	支持	~20 Mpps
Mellanox BlueField	mlx5	支持	支持 + 支持	~40 Mpps
Mellanox ConnectX-4	mlx4	支持	支持	~2 Mpps

Broadcom NICs

网卡	驱动	XDP 支持	性能	吞吐量
Broadcom BCM57xxx	bnxt_en	支持	支持	~8 Mpps
Broadcom NetXtreme II	bnx2x	支持	支持	~1 Mpps

其他

网卡	驱动	XDP 支持	性能	吞吐量
Netronome Agilio CX	nfp	支持	支持	~30 Mpps
Amazon ENA	ena	支持	支持	~5 Mpps
Solarflare SFC9xxx	sfc	支持	支持	~8 Mpps
VirtIO	virtio_net	支持	支持	~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
```

如何如何如何 NIC

如何 1200 如何如何如何 (如何如何) 如何

OS	NIC	OS	Mpps	Speed (N3)	Notes
Linux	VirtIO NIC (VirtIO E1000)	OS	1-2 Mpps	1-2 Gbps	Standard performance
Linux	Intel X710/Mellanox CX-5	OS	5-10 Mpps	5-10 Gbps	Good performance
Linux/Windows	Intel E810/Mellanox CX-6	OS	10-20 Mpps	10-20 Gbps	High performance
Linux	Mellanox CX-6/Intel E810 (SR-IOV)	OS	20-40 Mpps	20-40 Gbps	Very high performance
Linux	Mellanox BlueField/Netronome Agilio	OS	40+ Mpps	40+ Gbps	Best performance
Proxmox VE (SR-IOV)	VirtIO	OS	1-2 Mpps	1-2 Gbps	Standard performance
Proxmox VE (SR-IOV)	Intel X710/E810 VF/Mellanox CX-5 VF	OS	5-10 Mpps	5-10 Gbps	High performance

Network

OS XDP

- XDP
- XDP

NIC

- Cilium XDP

- IO Visor XDP

1

1 SR-IOV (OmniUPF)

SR-IOV OmniUPF

```
# (/etc/omniupf/runtime.exs)
xdp_interfaces = "eth0"
xdp_attach_mode = "generic"
api_port = 8080
pfcip_address = "127.0.0.1"
pfcip_port = 8805
node_id = "127.0.0.1"
n3_address = "127.0.0.1"
log_level = :debug
max_sessions = 1_000
```

2 Intel X710 NIC (UPF)

Intel X710 NIC UPF

```
# Proxmox (/etc/omniupf/runtime.exs)
xdp_interfaces = "ens1f0,ens1f1"    # N3  ens1f0N6  ens1f1
xdp_attach_mode = "native"
api_port = 8080
pfcip_address = "10.100.50.241"
pfcip_port = 8805
node_id = "10.100.50.241"
n3_address = "10.100.50.233"
n9_address = "10.100.50.234"
log_level = :info
max_sessions = 500_000
gtp_echo_interval = 30
gtp_peers = [
  %{address: parse_ip("10.100.50.10"), echo: true, echo_interval:
30},
  %{address: parse_ip("10.100.50.11"), echo: true, echo_interval:
30},
]
heartbeat_interval_ms = 10_000
feature_ueip = true
ueip_pool = "10.45.0.0/16"
```

Proxmox SR-IOV (UPF)

Proxmox SR-IOV UPF SR-IOV

```
# Proxmox SR-IOV ( /etc/omniupf/runtime.exs)
xdp_interfaces = "ens1f0"           # SR-IOV VF
xdp_attach_mode = "native"
api_port = 8080
pfcf_address = "192.168.100.10"
pfcf_port = 8805
node_id = "192.168.100.10"
n3_address = "192.168.100.10"
log_level = :info
max_sessions = 100_000
gtp_echo_interval = 15
gtp_peers = [
  %{address: parse_ip("192.168.100.50"), echo: true},
]
```

4 PGW-U (4G EPC)

OmniUPF 4G EPC PGW-U

```
# PGW-U ( /etc/omniupf/runtime.exs)
xdp_interfaces = "eth0"
xdp_attach_mode = "native"
api_port = 8080
pfcf_address = "10.200.1.10"
pfcf_port = 8805
node_id = "10.200.1.10"
n3_address = "10.200.1.10"           # S5/S8 (GTP-U)
log_level = :info
max_sessions = 200_000
gtp_echo_interval = 20
gtp_peers = [
  %{address: parse_ip("10.200.1.50"), echo: true, echo_interval:
20},
]
heartbeat_interval_ms = 5_000
```

00 50000 (UPF + PGW-U 00)

000OmniUPF 000 5G 0 4G 000000

```
# 00000 (/etc/omniupf/runtime.exs)
xdp_interfaces = "eth0,eth1"
xdp_attach_mode = "native"
api_port = 8080
pfcf_address = "10.50.1.100"
pfcf_port = 8805
node_id = "10.50.1.100"
n3_address = "10.50.1.100"
n9_address = "10.50.1.101"
log_level = :info
max_sessions = 300_000
gtp_echo_interval = 15
gtp_peers = [
  %{address: parse_ip("10.50.2.10"), echo: true, echo_interval:
15},
  %{address: parse_ip("10.50.2.20"), echo: true, echo_interval:
15},
]
heartbeat_interval_ms = 10_000
feature_ueip = true
ueip_pool = "10.60.0.0/16"
```

00 60SmartNIC 0000

00000 Netronome Agilio CX SmartNIC 00000000

```
# SmartNIC 配置 (/etc/omniupf/runtime.exs)
xdp_interfaces = "enpls0np0"          # SmartNIC 配置
xdp_attach_mode = "offload"
api_port = 8080
pfcip_address = "10.10.1.50"
pfcip_port = 8805
node_id = "10.10.1.50"
n3_address = "10.10.1.50"
log_level = :warning
max_sessions = 1_000_000
far_map_size = 2_000_000
qer_map_size = 1_000_000
gtp_echo_interval = 30
gtp_peers = [
  %{address: parse_ip("10.10.2.10"), echo: true},
  %{address: parse_ip("10.10.2.20"), echo: true},
  %{address: parse_ip("10.10.2.30"), echo: true},
]
heartbeat_interval_ms = 15_000
```

配置项

配置项

配置项 `max_sessions` 配置项

```
max_sessions = 100_000
far_map_size = 131_070
qer_map_size = 65_535
urr_map_size = 65_535
```

配置项~91 MB 配置 100K 配置

□□□□

□□□□□□□□

```
□□□□□ = min(  
    pdr_map_size / 2,  
    far_map_size / 2,  
    qer_map_size  
)
```

□□□

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

□□□□□ = min(100,000, 100,000, 100,000) = **100,000**

□□□□

□□□□□□□□

- PDR: 2 x 212 B = 424 B
- FAR: 2 x 20 B = 40 B
- QER: 1 x 36 B = 36 B
- URR: 2 x 20 B = 40 B
- □□: ~540 B □□□

□□ **100K** □□□~52 MB □□□□

□□□□□□□□□□□□□□ 2x □□□□□

```
# 内核参数  
ulimit -l
```

```
# 内核参数 (eBPF 支持)  
ulimit -l unlimited
```

功能

- [内核参数](#) - eBPF/XDP 支持
- [内核参数](#) - PDR/FAR/QER/URR 支持
- [内核参数](#) - 支持
- [内核参数](#) - 支持 Prometheus 支持
- **Web UI** 支持 - 支持
- [内核参数](#) - UPF 支持
- [内核参数](#) - 支持



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	설명	참고
upf_pfcpx_rx	message_name, peer_address	받은 PFCP 메시지	
upf_pfcpx_tx	message_name, peer_address	보낸 PFCP 메시지	
upf_pfcpx_rx_errors	message_name, cause_code, peer_address	받은 PFCP 메시지 오류	
upf_pfcpx_rx_latency	message_type, peer_address	받은 PFCP 메시지 지연률 (p50, p90, p99)	

이벤트 이름은 PFCP 메시지의 종류와 관련이 있습니다.

XDP 이벤트

XDP 이벤트는 XDP 프레임의 처리 결과를 나타냅니다.

이벤트	필드	설명	참고
upf_xdp_aborted	XDP_ABORTED	XDP 프레임이 중단됨	
upf_xdp_drop	XDP_DROP	XDP 프레임이 삭제됨	
upf_xdp_pass	XDP_PASS	XDP 프레임이 통과됨	
upf_xdp_tx	XDP_TX	XDP 프레임이 전송됨	
upf_xdp_redirect	XDP_REDIRECT	XDP 프레임이 리디렉션됨	

Packet Type

Packet Type (packet_type) List

Table Name	Column Name	Column Type	Description
upf_rx	packet_type	enum	Packet Type
upf_route	packet_type	enum	Packet Type

upf_rx packet_type List

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

upf_route packet_type List

- ip4-cache - IPv4 Cache
- ip4-ok - IPv4 FIB OK
- ip4-error-drop - IPv4 FIB Error Drop
- ip4-error-pass - IPv4 FIB Error Pass
- ip6-cache - IPv6 Cache
- ip6-ok - IPv6 FIB OK
- ip6-error-drop - IPv6 FIB Error Drop
- ip6-error-pass - IPv6 FIB Error Pass

PFCP

UPF PFCP

Table Name	Columns	Indexes	Description
upf_pfcpc_sessions	PK		PFCP sessions
upf_pfcpc_associations	PK		PFCP associations
upf_pfcpc_association_status	PK	node_id, address	PFCP association status (1=active, 0=inactive)
upf_pfcpc_sessions_per_node	PK	node_id, address	PFCP sessions per node

URR

PFCP URR

Table Name	Columns	Indexes	Description
upf_urr_uplink_volume_bytes	PK	peer_address	Uplink volume bytes
upf_urr_downlink_volume_bytes	PK	peer_address	Downlink volume bytes
upf_urr_total_volume_bytes	PK	peer_address	Total volume bytes (uplink + downlink)

PFCP URR REST API `/api/v1/urr_map`

□□□□□□

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

名称	数据类型	单位	说明
upf_buffer_packets_total	uint64_t	个	UPF 缓冲区的总包数
upf_buffer_packets_dropped	uint64_t	个	UPF 缓冲区中丢弃的包数，原因见 reason
upf_buffer_packets_flushed	uint64_t	个	UPF 缓冲区中已刷新的包数
upf_buffer_packets_current	uint64_t	个	UPF 缓冲区中当前的包数
upf_buffer_bytes_total	uint64_t	字节	UPF 缓冲区的总字节数
upf_buffer_bytes_current	uint64_t	字节	UPF 缓冲区中当前的字节数
upf_buffer_fars_active	uint64_t	个	UPF 缓冲区中活跃的 FAR 数

计数器	单位	范围	备注
upf_buffer_listener_packets_received_total	无	无	eBPF 计数器 计数器 计数器 计数器
upf_buffer_listener_packets_buffered_total	无	无	计数器 计数器 计数器 无
upf_buffer_listener_errors_total	无	type	计数器 计数器 计数器 无
upf_buffer_listener_error_indications_sent_total	无	remote_peer	计数器 TEID GTP-U 计数器 无
upf_buffer_flush_success_total	无	无	计数器 计数器 计数器
upf_buffer_flush_errors_total	无	reason	计数器 计数器 无

计数器	单位	范围	类型
upf_buffer_flush_packets_sent_total	无	无	计数器 计数器 计数器 计数器

upf_buffer_packets_dropped reason 原因

- `expired` - 会话 TTL 过期
- `global_limit` - 全局限制
- `far_limit` - 会话限制 FAR 限制
- `cleared` - 清除

upf_buffer_listener_errors_total type 类型

- `read_error` - 读取错误
- `too_small` - 缓冲区太小 GTP 消息
- `invalid_gtp_type` - 无效的 G-PDU GTP 消息
- `unknown_teid` - 未知的 TEID 消息 PDR/FAR
- `not_buffering_far` - FAR 不在缓冲区
- `truncated_ext` - GTP 消息截断
- `no_payload` - GTP 消息无负载
- `buffer_full` - 缓冲区满

upf_buffer_flush_errors_total reason 原因

- `far_lookup_failed` - FAR 查找失败 eBPF 限制 FAR 消息
- `no_forw_action` - FAR 没有转发动作
- `connection_failed` - 连接失败 UDP 消息

计数器

PFPCP 计数器 UE

名称	数据类型	单位	描述
upf_dldr_sent_total	无符号整数	个	SMF 向 DLDR 发送的总消息数
upf_dldr_send_errors	无符号整数	个	SMF 向 DLDR 发送消息时发生的错误数
upf_dldr_active_notifications	无符号整数	个	DLDR 向 SMF 发送的活跃通知数
upf_far_index_size	无符号整数	个	FarIndex 索引的大小
upf_far_index_registrations_total	无符号整数	个	FarIndex 索引中的总注册数
upf_far_index_unregistrations_total	无符号整数	个	FarIndex 索引中的总注销数
upf_buffer_notify_to_flush_duration_seconds	无符号浮点数	秒	DLDR 向 SMF 发送通知到刷新缓冲区的持续时间

upf_buffer_notify_to_flush_duration_seconds:

- 取值范围：0.01, 0.05, 0.1, 0.5, 1.0, 2.0, 5.0, 10.0, 30.0, 60.0
- 默认值：10.100.50.241

- 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 5G Core SMF TEID 5G Core SMF 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 5G Core SMF 5G Core SMF	node_id, peer_address
upf_buffer_listener_error_indications_received_total	5G Core SMF 5G Core SMF 5G Core SMF	node_id, peer_address
upf_buffer_listener_error_indication_sessions_deleted_total	5G Core SMF 5G Core SMF 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

#
sum by (node_id, peer_address) (upf_buffer_listener_error_indications
```

eBPF

eBPF

名称	类型	配置	描述
upf_ebpf_map_capacity	整数	map_name	eBPF 规则数量
upf_ebpf_map_used	整数	map_name	eBPF 规则使用数量

map_name 配置

- pdr_map - 策略数据规则
- far_map - 转发策略规则
- qer_map - QoS 策略规则
- session_map - 会话策略规则
- teid_map - TEID 策略规则
- ue_ip_map - UE IP 策略规则

🔍 Prometheus 配置

配置

配置地址 `metrics_address` 为 `/metrics` 端口 `:9090`

```
# 配置 curl 命令
curl http://localhost:9090/metrics

# 配置 Prometheus 数据
upf_pfcps_sessions 42
upf_pfcps_associations 2
upf_urr_total_volume_bytes{peer_address="10.100.50.241"}
1048576000
```

Prometheus 配置

OmniUPF 配置 `prometheus.yml`

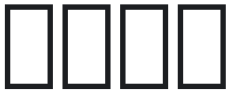
```
scrape_configs:
  - job_name: 'omniupf'
    static_configs:
      - targets: ['localhost:9090']
```

Grafana

Grafana

-
- PFCP
-
-
- eBPF

-
- `metrics_address` UPF
- **Web UI**
- eBPF
- PDR, FAR, QER, URR
-



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



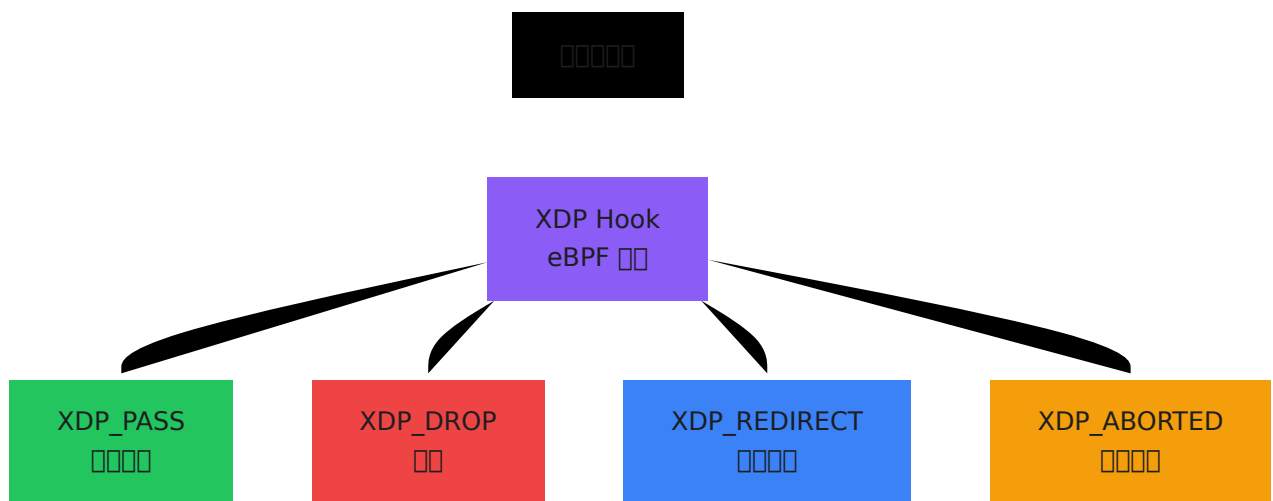
OmniUPF OmniUPF Web UI REST API



□□	□□	□□□□	□□□□
□□□□□	□□□□□□□□□□□□	□□	RX/TX □□□□□□□□□□
□□□□	□□ N3/N6 □□□□	□□	N3 RX/TX, N6 RX/TX
XDP □□	□□□□□□□□□□	□□	XDP □□□□□□□□□□
□□□□	□□□□□□□□□□	□□	FIB □□□□□□□□/□□□□
eBPF □□□□	□□□□□□□□	□ 10 □	□□□□□□□□□□□□□□
□□□□□□	□□□□□□□□□□□□	□ 5 □	□□□□□□□□□□□□□□□□ FAR □□

XDP

XDP (eXpress Data Path)



•

- XDP

- **XDP** **TX** **XDP**
- **XDP**
- **XDP**
- **TX** **XDP**

TX

- **TX > 0** **eBPF**
- **TX > 0**
- **TX**
- **TX**

TX

TX

TX

- **RX ARP**
- **RX GTP ECHO** **GTP-U**
- **RX GTP OTHER** **GTP**
- **RX GTP PDU** **GTP-U**
- **RX GTP UNEXP** **GTP**
- **RX ICMP** **ping**
- **RX ICMP6** **ICMPv6**
- **RX IP4** **IPv4**
- **RX IP6** **IPv6**
- **RX OTHER**
- **RX TCP**
- **RX UDP**

TX

- **GTP-U PDU**
- **ICMP**

- **TCP** **UDP**
 -
-

FIB ()

IPv4 FIB

-
- **OK**

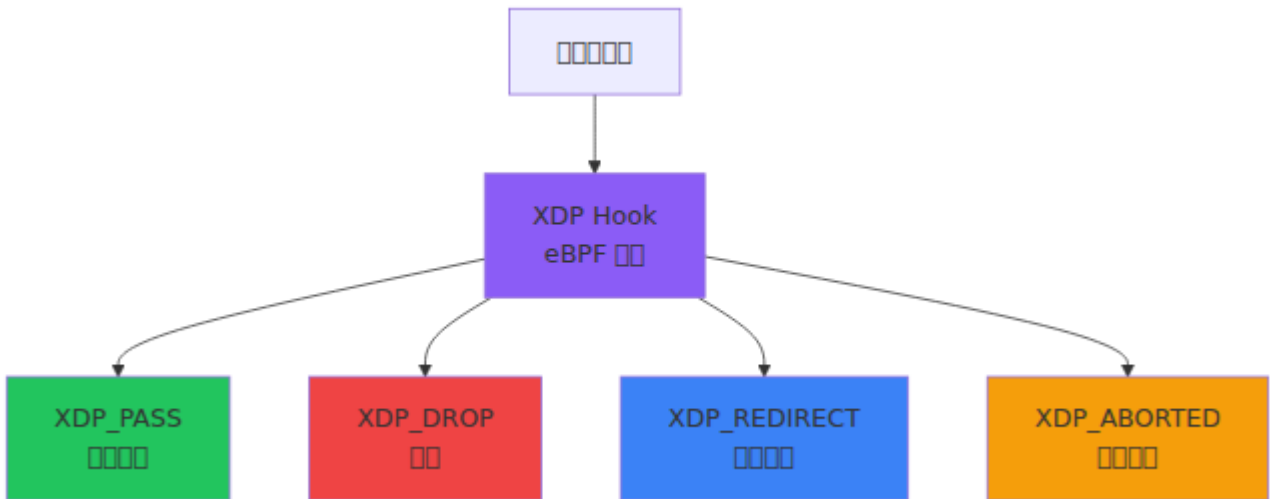
IPv6 FIB

- IPv6
- **OK** IPv6

- - **OK**
 -
-

eBPF

eBPF



eBPF

far_map (远端映射)

- 131,070 个
- 4 B (FAR ID)
- 16 B (远端地址)
- ~2.6 MB
- 远端地址 - 远端地址

pdr_map_downlin (下行 PDRs - IPv4)

- 131,070 个
- 4 B (UE IPv4 地址)
- 208 B (PDR 信息)
- ~27 MB
- 远端地址 - 远端地址

pdr_map_downlin_ip6 (下行 PDRs - IPv6)

- 131,070 个
- 16 B (UE IPv6 地址)
- 208 B (PDR 信息)
- ~29 MB
- 远端地址 - IPv6 远端地址

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
- -

□□□□

□□	□□□□□□□□
0-50% (□□)	□□□□ - □□□□□□
50-70% (□□)	□□ - □□□□□□□□□□□□□□
70-90% (□□)	□□ - □ 1 □□□□□□□□
90-100% (□□)	□□ - □□□□□□□□□□□□□□

□□□□□□

□□□□□□□□

1. □□□□□□□□
2. □□□□□□□□

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 Prometheus 設定

設定

設定

$$\text{設定 (pps)} = (\text{設定}) / (\text{設定})$$

設定

- RX 7,000
- 10 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
- CPU
-
-

Throughput

Throughput

$$\text{Throughput (Mbps)} = (\text{Bytes} \times 8) / (\text{Time} \times 1,000,000)$$

Example

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

Latency

-
- N3/N6
- 2

Drop Rate

Drop Rate

$$\text{Drop Rate (\%)} = (\text{Dropped Bytes} / \text{RX Bytes}) \times 100$$

Drop Rate

- < 0.1%
- 0.1% - 1%

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

- QER MBR
 - eBPF
 - TEID UE IP
 -
-

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

1

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

- eBPF > 50%
-
- PFCP
- URR

□□□□

□□□□□□□□□□□□

1. **Prometheus** □□□□□□□□□□□□□□□□ □□□□ □□□□□□□□
 2. □□□□□□□□ OmniUPF □□□□□□□□□□
 3. **REST API** □□□□□□□□ /map_info □/packet_stats □□
 4. **Web UI** □□□□□□□□□□□□□□□□□□□□
-

□□□□

□□□□□□□□

□□□□□□□□

```
□□□□□ = min(  
  PDR □□□□ / 2, # 000□ + □□ PDR □□□  
  FAR □□□□ / 2, # □□ + □□ FAR □□□  
  QER □□□□      # □□□□□□□□ QER  
)
```

□□□

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

□□□□□ = min(131,070 / 2, 131,070 / 2, 65,535) = **65,535** □□

□□□□

□□□ **eBPF** □□□□□

```
□□ = ∑ (□□□□ × (□□□ + □□□))
```

□□□□

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

□□□□□□□□

- □□□□□□□□□□ `ulimit -l`
- □□□□□□ 2 □□□□□□
- □□□□□□□□

□□□□

□□□□□□□□□□

1. □□□□□□□□□□

- □□□□~5 Mbps
- □□□□~1 Mbps
- VoIP~0.1 Mbps

2. □□□□□□□□

$$\square\square\square\square = \square\square\square \times \square\square\square\square\square\square$$

3. □□□□□□

$$\square\square\square\square = \square\square\square\square \times 2 \quad \# \text{ 100\% } \square\square$$

□□□□

- 10,000 □□□□□□
- □□□□□□ 2 Mbps

- 20 Gbps
- 40 Gbps (N3 + N6)

10000

10000

1. 1000000000
2. 100000000
3. 10000000

1000000

$$\text{Percentage} = (\text{Value} - \text{Target}) / (\text{Target})$$

1000

- 30,000
- 65,535
- 2,000
- $(65,535 - 30,000) / 2,000 = 17.8$

12 5

1000000000

10000

Percentage > 1%

1000

1. 1000 → 10000
2. 1000000000
3. XDP XDP

□□□□

- **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 □□□□□
 - □□□□□□□□□□□□□□□□
-

□□□□

□□□□□□□□□□□□□□□□ 100%

□□□

1. □□□□□□□
2. □□□□□□□□ 100%
3. □□□□□□□□□□□□□□□□

□□□□□

1. □□□□□□□□□□□□□□□□
2. □□ SMF □□□□□
3. □□□□□□□□ FAR □□

□□□□□□□

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

□□□□

□□□□□□□□□□□□□□□□□□□□ CPU □□

□□□

1. □□□□□□□□□□□□□□□□
2. □□ XDP □□□□□□□□□□
3. □□ UPF □□□□□ CPU □□□□
4. □□ N3/N6 □□□□□□

□□□□□

- □□□□ UPF □□
- □□□□□□□ CPU □□□□□

- 网络策略
- eBPF 网络策略

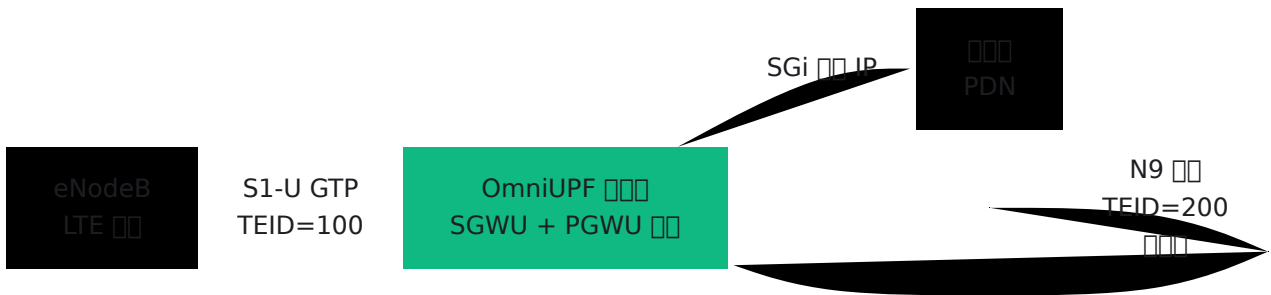
网络策略

- 网络策略 UPF 网络策略
 - CPU 网络 RSS 网络策略
 - 网络策略
 - 网络 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

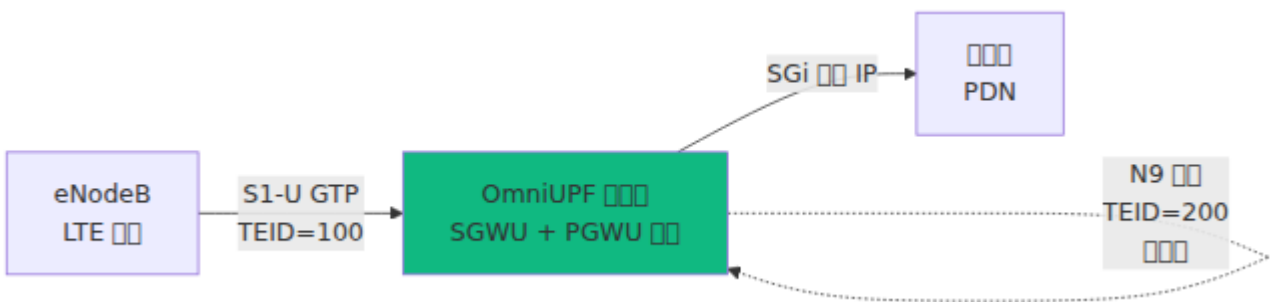
N9



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 →

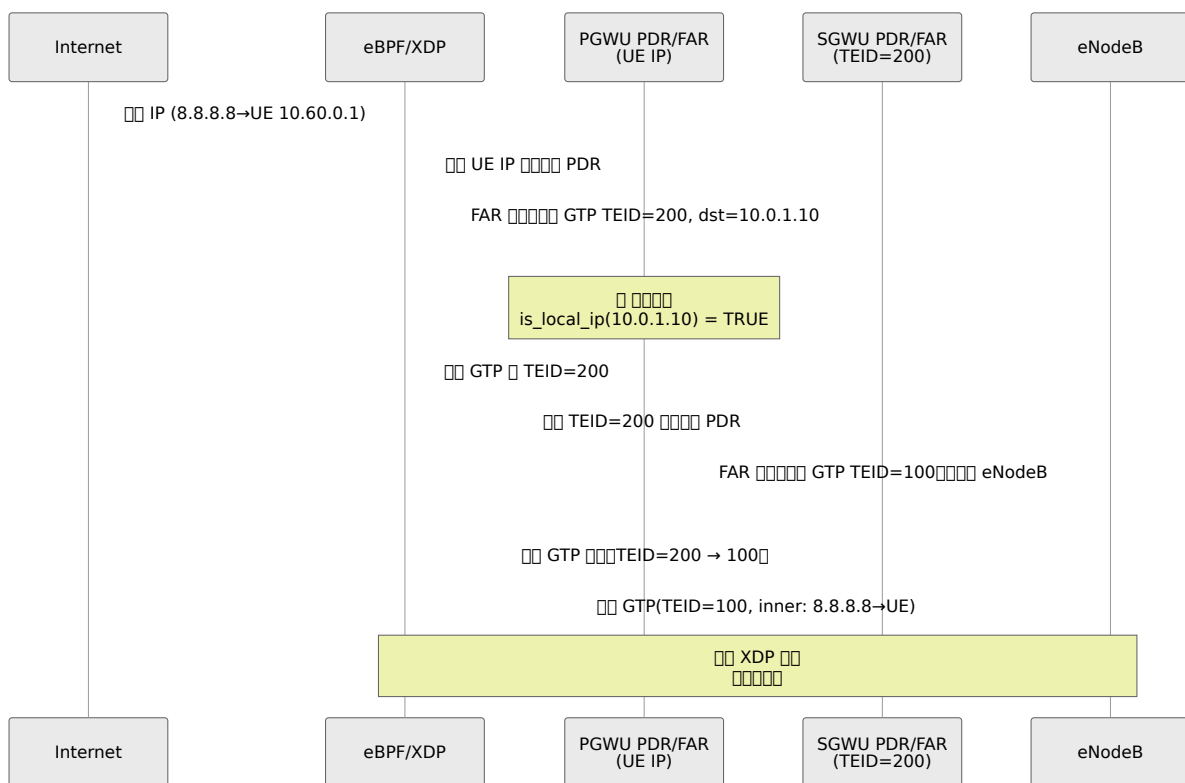


eBPF `ebpf/xdp/n3n6_entrypoint.c` 349-403

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 []

Internet → PGWU → SGWU → eNodeB



eBPF [] [] `ebpf/xdp/n3n6_entrypoint.c` [] 137-194 [] IPv4 [] 265-322 [] IPv6 []

[][] []

1. [] [] [] [] UE [] IP [] [] 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. PDR TEID=200 SGWU
 7. FAR GTP eNodeB TEID=100
 8. GTP S1-U eNodeB
-

□□

□□

□□□□

- SGWU-C OmniUPF PCF 192.168.1.10:8805
- PGWU-C OmniUPF PCF

□□□

- N3 N9 IP
 - SGWU-C PGWU-C IP
-

OmniUPF □□

/etc/omniupf/runtime.exs:

```

# 配置
xdp_interfaces = "eth0"                # S1-U 与 N9 配置
xdp_attach_mode = "native"            # 配置

# PCF 配置
pfcf_address = "192.168.1.10"         # OmniUPF 与 PCF 配置
pfcf_port = 8805                      # PCF 配置
node_id = "192.168.1.10"             # OmniUPF 与 PCF 配置 ID

# 配置
n3_address = "10.0.1.10"              # S1-U/N3 配置 IP
n9_address = n3_address                # N9 配置 IP 与 N3 配置

# 配置
feature_ueip = true                    # UE IP 配置
ueip_pool = "10.60.0.0/16"
feature_ftup = true
teid_pool_start = 1
teid_pool_end = 65_535

# 配置
max_sessions = 100_000                # 配置 UE 配置

# API
api_port = 8080

```

配置

- `n3_address` 与 `n9_address` 配置
- 配置 PCF 配置
- 配置 SGWU + PGWU 配置 `max_sessions`

配置

SGWU-C 配置

```
# OmniUPF PFCP
upf_pfcpc_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_pfcpc_address: "192.168.1.10:8805"

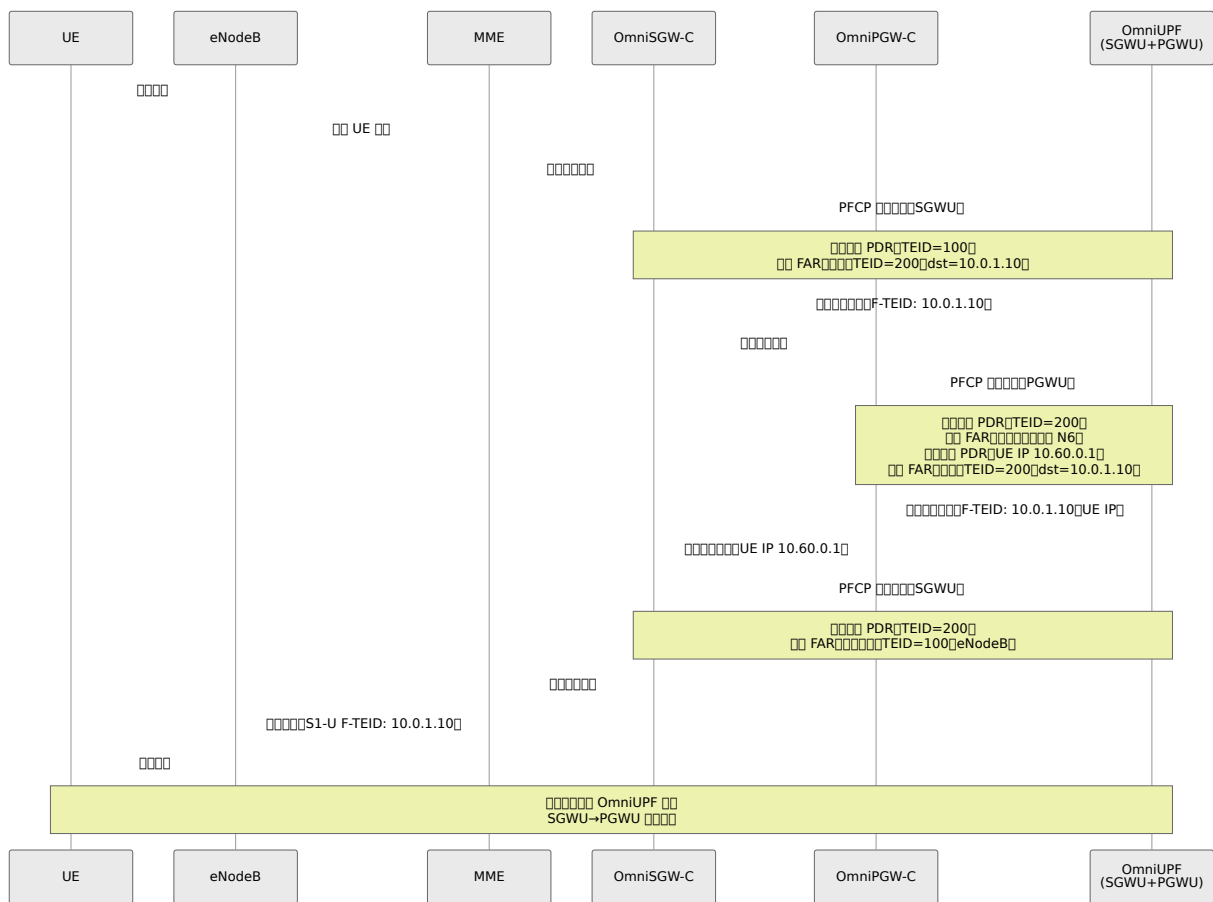
# N9 SGWU
pgwu_n9_address: "10.0.1.10"

# SGi
pgwu_sgi_address: "192.168.100.1"
```

- PFCP :8805
- OmniUPF SGWU-C PGWU-C PFCP
- ID

UE PDU

UE



PCP

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 N6
- PDR UE IP=10.60.0.1 → FAR TEID=200 dst=10.0.1.10

N9

XDP

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

```
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
```

REST API

PCFP

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

```
{
  "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

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

curl

```
{
  "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

□□□□□□ runtime.exs □□□

```
# n3
n3_address = "10.0.1.10"
n9_address = "10.0.1.20" # n3 IP

# n9
n3_address = "10.0.1.10"
n9_address = n3_address # n3 IP
```

curl

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

output

```
{
  "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

steps

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

```
# runtime.exs
buffer_port = 22152
```

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

N9 性能指标

性能

指标	配置	性能指标 N9	目标
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 线程 + 线程	1x XDP 线程	约 40-50%
内存使用	内存使用	内存使用	约

特性

- 支持 OmniUPF 部署
- 支持 IP 地址
- 支持
- 支持
- 支持 eBPF 性能

性能

性能

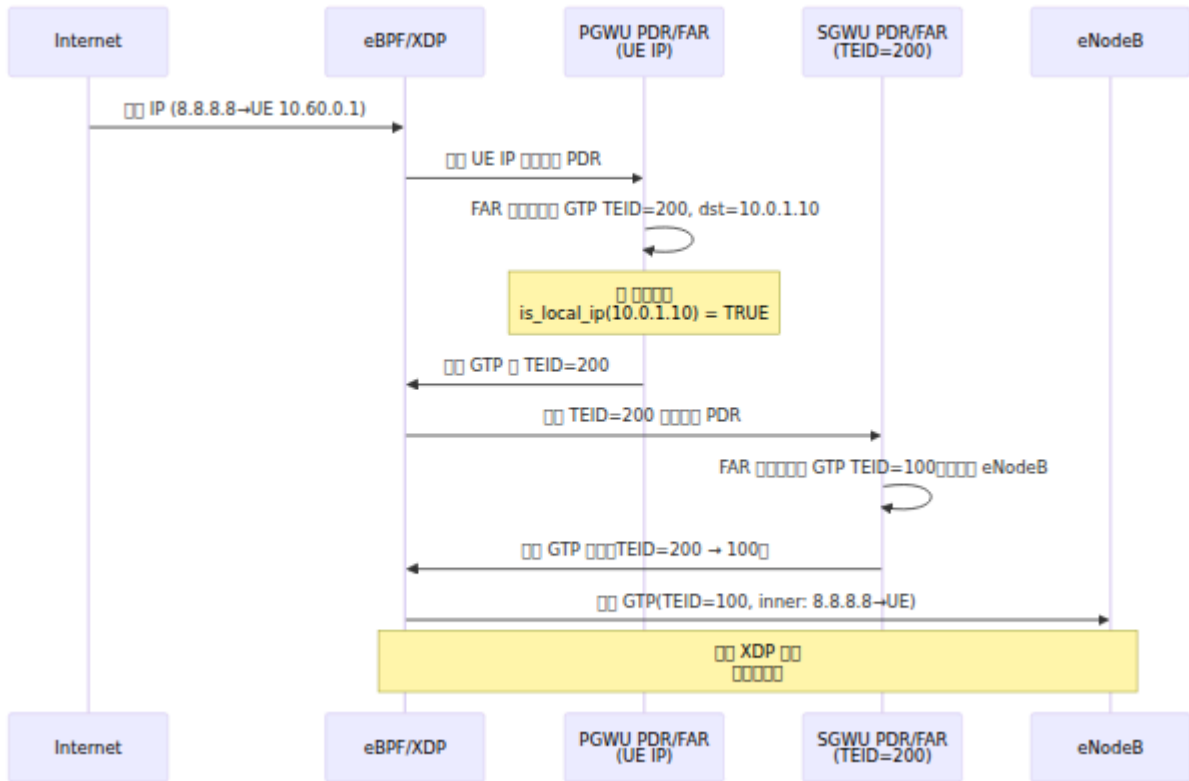
- 性能
- 性能 < 100K
- 性能 EPC
- 性能

□□□□□□

- □ □□□□□ SGWU □ PGWU □□□□□□□□
- □ □□□□ > 1M □□□□□□□□□□
- □ □□□□□ □□□□□ SGW □ PGW

□□□□□□□□□□□□

□□□□□□ **N9** □□□□□□□□□□



□□

N9 □□□ □□□ **OmniUPF** □□□□□□□□ **4G EPC** □□□ □□□□□□□□□□ eBPF □□□□
SGWU→PGWU □□□□□□□□□□□□□□□□

- □ □□□□□□ □□□□□□□□
- □ **40-50% CPU** □□ □□□□□□□□

- [n3_address == n9_address](#) - [n3_address == n9_address](#)
- [n3_address == n9_address](#) - [n3_address == n9_address](#)
- [n3_address == n9_address](#) **3GPP** [n3_address == n9_address](#) - [n3_address == n9_address](#) PFCP/GTP-U [n3_address == n9_address](#)

[n3_address == n9_address](#) [n3_address == n9_address](#) - [n3_address == n9_address](#) OmniUPF [n3_address == n9_address](#) eBPF [n3_address == n9_address](#) [n3_address == n9_address](#)

[n3_address == n9_address](#)

- [n3_address == n9_address](#) [CONFIGURATION.md](#)
- [n3_address == n9_address](#) [ARCHITECTURE.md](#)
- [n3_address == n9_address](#) [METRICS.md](#)
- [n3_address == n9_address](#) [MONITORING.md](#)
- [n3_address == n9_address](#) [OPERATIONS.md](#)
- [n3_address == n9_address](#) [TROUBLESHOOTING.md](#)

PFCP 問題集

概要

PFCPは、3GPP TS 28.413で定義された、Omnium UPFとUEとの間で、PFCPセッションを管理するためのプロトコル。

この問題集は、3GPP TS 129.244で定義されたPFCPの動作を説明する。

監視

Omnium UPFは、Prometheusを使用してPFCPの動作を監視する。

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

エラー

- メッセージの送信に失敗した
- メッセージの受信に失敗した
- メッセージの処理に失敗した

エラーメッセージは、PFCPメッセージで送信される。

メッセージ

RequestAccepted

メッセージID	メッセージ名	メッセージ内容
1	RequestAccepted	UEからUPFへのPFCPセッションの確立を通知する。

0000/00000000

00	00	0000
74	PFCPEntityInCongestion	UPF 000000000000000000000000
75	NoResourcesAvailable	0000000000000000eBPF 0000000000000000 TEID 0000
77	SystemFailure	0000000000000000eBPF 0000000000000000 000000

0000000000

00	00	0000
68	InvalidLength	000000000000000000000000 OmniUPF 000000
70	InvalidForwardingPolicy	UPF 0000000000000000 OmniUPF 000000
71	InvalidFTEIDAllocationOption	0000 F-TEID 0000000000000000 OmniUPF 00 0000
76	ServiceNotSupported	000000000000000000000000 OmniUPF 000000
78	RedirectionRequested	UPF 0000000000000000 UPF 00000000 OmniUPF 000000

0000000000

00000000

000000 **NodeID**

SMF → UPF: `NodeID`
UPF → SMF: `MandatoryIEMissing`

SMF `NodeID` 消息

NodeID 消息

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

`NodeID` 消息 FQDN 或 IPv4/IPv6 消息

消息

SMF → UPF: `RecoveryTimeStamp`
UPF → SMF: `MandatoryIEMissing`

`RecoveryTimeStamp` 消息

消息

消息

SMF → UPF: `NoEstablishedPFCPSession`
UPF → SMF: `NoEstablishedPFCPSession`

消息 PFCP 消息

消息

SMF → UPF: `RuleCreationModificationFailure`
UPF 消息 `FAR`/`QER`/`URR`
UPF 消息 `PDR`/`eBPF` 消息
UPF → SMF: `RuleCreationModificationFailure`

□□□□

- □□ eBPF □□□□□□ □□□□
- □ UPF □□□□□□□□
- □□□□□□□□

□□□□ **F-SEID**

```
SMF → UPF: □□□□□□□□ CP F-SEID□  
UPF → SMF: □□□□□□□□□MandatoryIEMissing□
```

□□□□□□□□□□□□□□ CP F-SEID□

□□□□□□

□□□□ **SEID**

```
SMF → UPF: □□□□□□□SEID=12345□  
UPF □□ SEID 12345 □□□  
UPF → SMF: □□□□□□□□□SessionContextNotFound□
```

□□□□

- □□ SEID □□□□□□□□□□□□□□
- □□□□□□□□□□
- □□□□□□□ UPF □□□N9 □□□□□□□□

□□□□□□

□□□□ **SEID**

```
SMF → UPF: □□□□□□□SEID=67890□  
UPF □□ SEID 67890 □□□  
UPF → SMF: □□□□□□□□□SessionContextNotFound□
```

□□□□SEID □□□□□□□□□□□□

□□□□□□□□□□□□□□

□□ Prometheus □□

□□ Prometheus □□□□□□□□

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

# □□□□□□□
topk(5, sum by (cause_code) (upf_pfcpx_errors))

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

# □□□□□□□
upf_pfcpx_errors{message_name="SessionEstablishmentRequest",
cause_code!="RequestAccepted"}
```

□□ Web □□

□□□ □□ □□□□□□

- □□□□□□□□□□□□□□□□
- □□□□□□□/□□□□
- □□□□□□□□

□□□ □□ □□□□□□□□

- eBPF □□□□□□□RuleCreationModificationFailure □□□□□□
- □□□□□□□

□□□ Web UI □□ □□□□□□□□□□

□□□□□□□□

□ **MandatoryIEMissing** □□

1. SMF
2. PCF
3. SMF

RuleCreationModificationFailure

1. eBPF GET /api/v1/map_info
2. upf_ebpf_map_used / upf_ebpf_map_capacity
3. > 70%
- 4.

NoEstablishedPCFAssociation

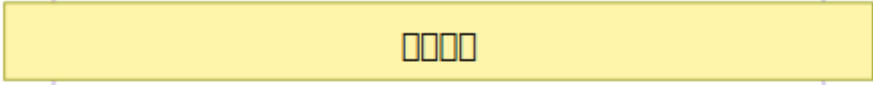
1. GET /api/v1/pfcp_associations
- 2.
- 3.
4. SMF UPF

SessionContextNotFound

1. SEID
- 2.
3. N9 UPF
4. GET /api/v1/pfcp_sessions

□□□□□□□□

□□□□□



Request

alt

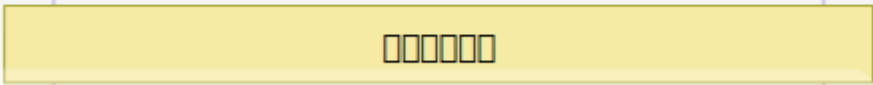
[NodeID]

Core 5GC | OmniCore | OmniCall | OmniRAN | OmniCharge | Platform |

MandatoryIEMissing

[NodeID]

MandatoryIEIncorrect



Request

Sizing your network?

Get a deployment estimate in 60 seconds. Pick a workload size, see the topology, and get pricing — no sales call needed.

alt

[NodeID]

NoEstablishedPFCPAssociation

[eBPF]

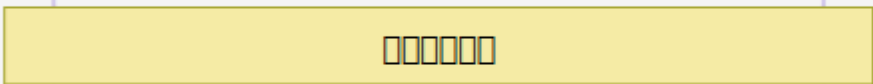
Size my network

RuleCreationModificationFailure

Talk to engineer

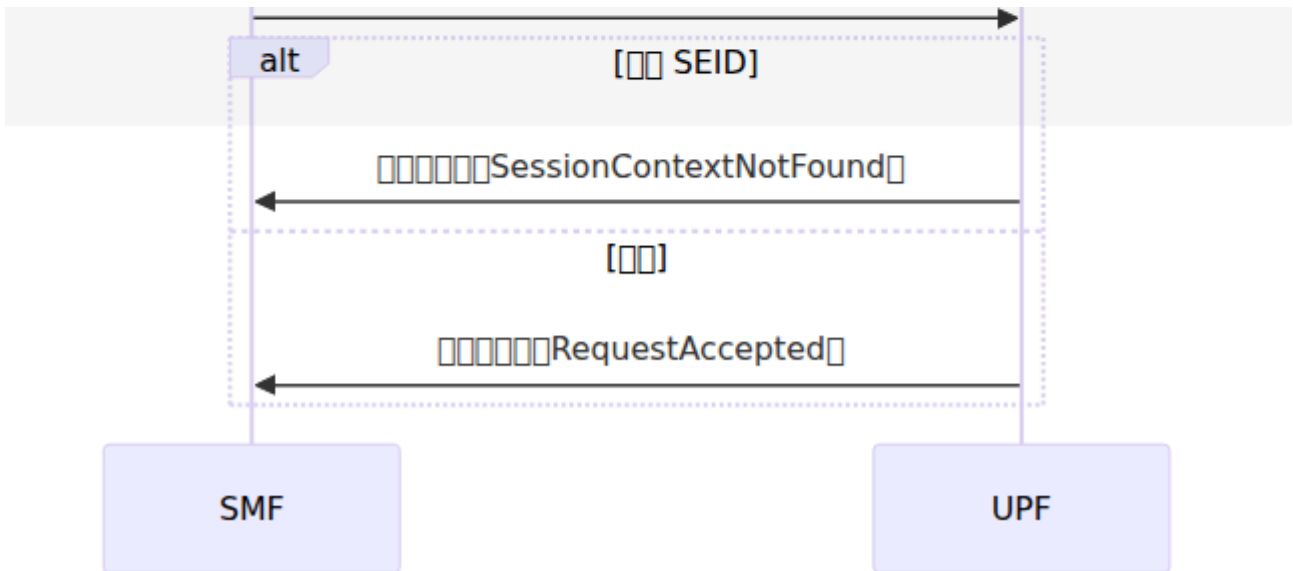
RequestAccepted

Response



Request

Response



□□□□□

□□□□□

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

# □□□□□□□
- alert: PfcpruleCreationFailures
  expr: |

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

# □□□□□□□
- alert: PfcppnoAssociation
  expr: |

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

3GPP 4G-LTE

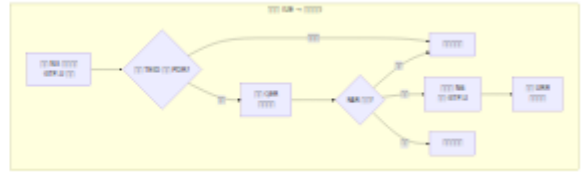
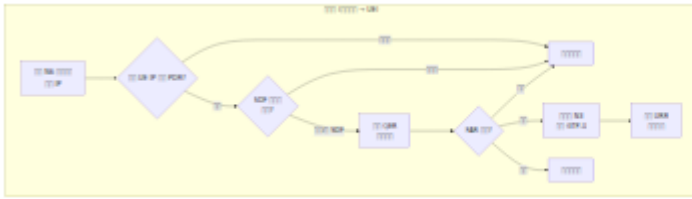
OmniUPF 4G-LTE

- **3GPP TS 129.244 v16.4.0** - PCRF 4G-LTE
- **8.2.1** - 4G-LTE
- **8.19** - 4G-LTE

4G-LTE

- **PCRF 4G-LTE** - PCRF 4G-LTE
- **4G-LTE** - upf_pfcpx_rx_errors 4G-LTE
- **4G-LTE** - 4G-LTE
- **4G-LTE** - PCRF 4G-LTE
- **Web UI 4G-LTE** - 4G-LTE

□□□□□□

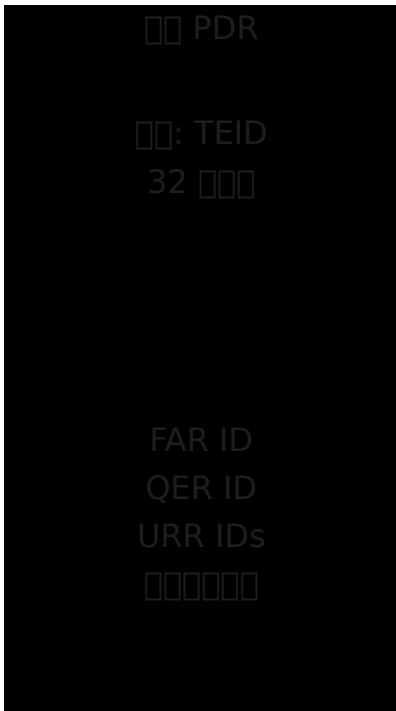


□□□□□□□ (PDR)

□□

PDR □□□□□□□□□□□□ UPF □□□□□□□□□□□□

PDR



PDR

PDR RAN N3

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
- 阻止其他流量

端口:

- 端口: TCP/UDP/ICMP
- 端口: 端口范围 HTTP 443/SIP 5060
- IP 地址: 地址范围
- 端口: 3GPP 地址范围

规则 SDF 规则:

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

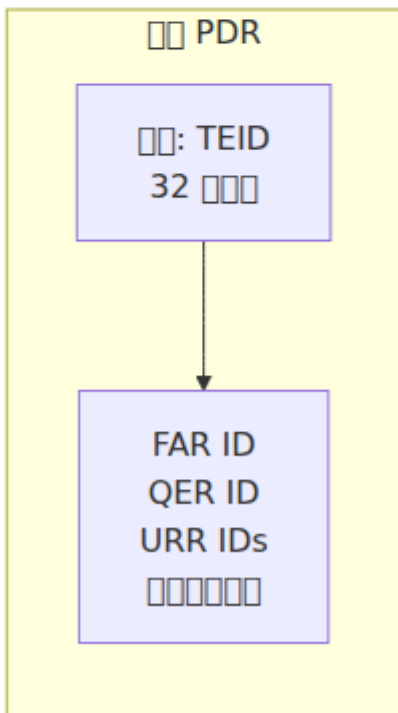
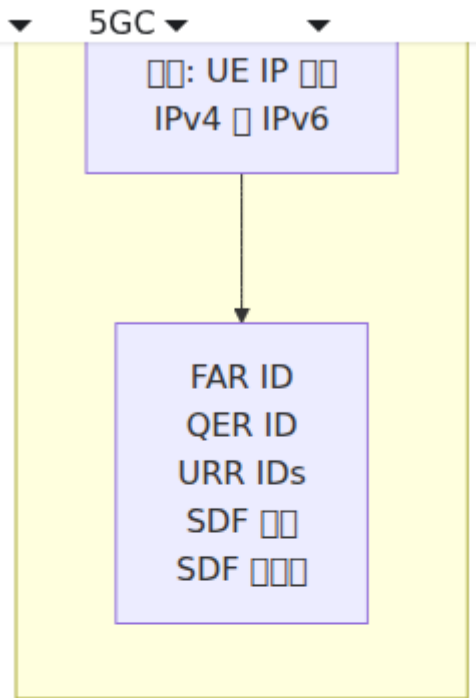
流量过滤器 (FAR)

规则

FAR 规则指定 PDR 规则匹配的流量在 GTP-U 隧道中的处理

FAR

Core OmniCore OmniCall



□□□□

FAR □□□□□□□□□□□□

操作	1	2	操作
FORWARD	1	2	操作
BUFFER	2	4	操作
DROP	0	1	操作
NOTIFY	3	8	操作
DUPLICATE	4	16	操作

操作:

- 操作: 2 (FORWARD) - 操作
- 操作: 6 (FORWARD + BUFFER) - 操作
- 操作: 4 (BUFFER) - 操作
- 操作: 1 (DROP) - 操作

操作

BUFFER 操作 2操作操作操作操作操作操作操作操作操作 UPF 操作操作操作操作操作 UE 操作操作操作操作操作

操作操作

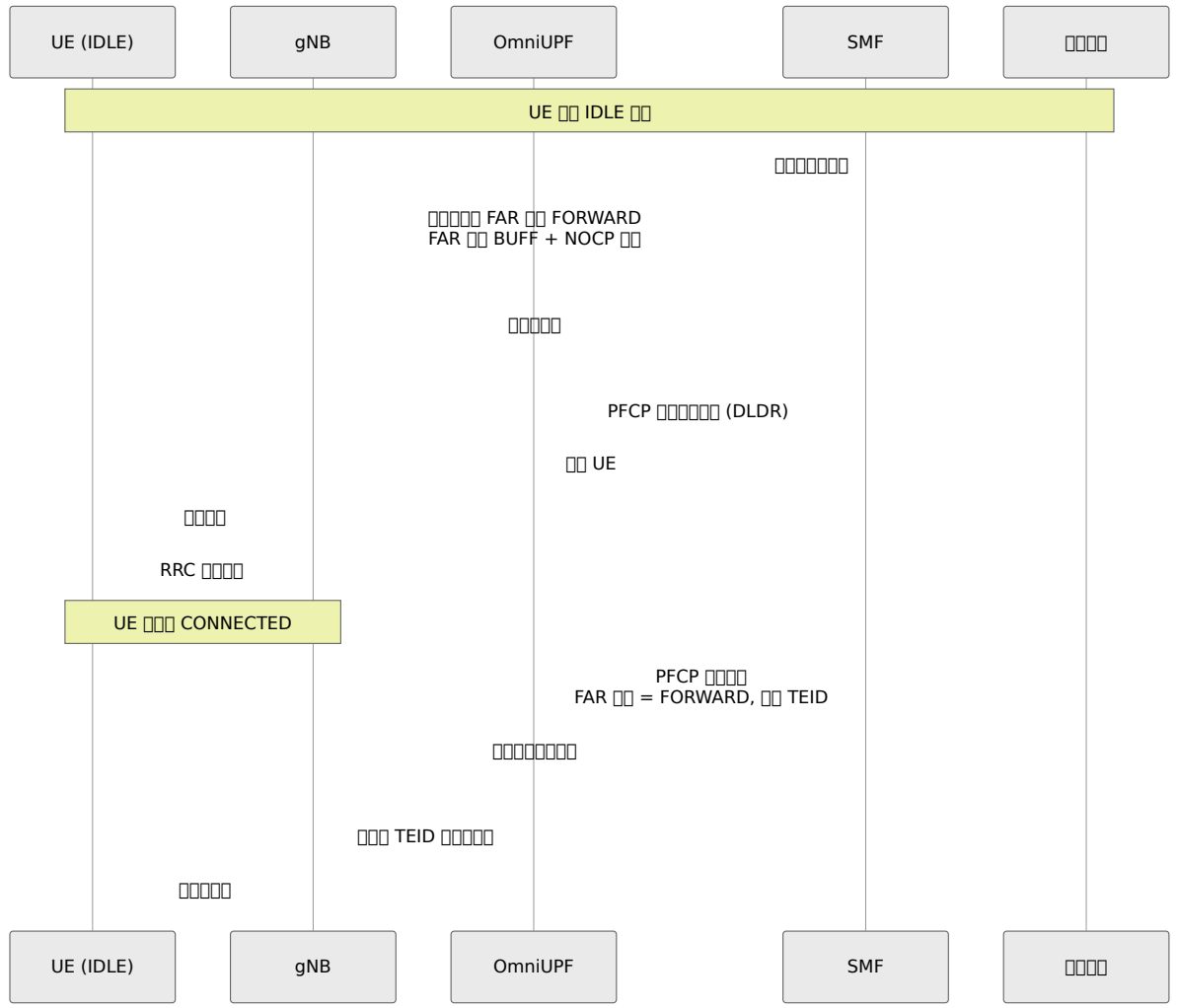
操作操作操作: 操作操作操作 IDLE 操作操作 gNB 操作 UE 操作UPF

- 操作
- 操作 SMF 操作操作 (DLDR)
- SMF 操作 AMF 操作 UE 操作操作操作
- 操作操作SMF 操作 FAR 操作 FORWARD 操作
- UPF 操作操作操作操作 UE

操作操作操作: 操作 gNB 操作 gNB 操作操作UPF 操作操作操作操作操作

- 操作 gNB 操作
- SMF 操作 FAR 操作操作 BUFFER

3. []
4. UE [] gNB
5. SMF [] FAR [] TEID [] FORWARD []
6. UPF [] gNB



[]

[]:

- []: 100,000 []
- []: []
- **TTL** ([]): 60 []
- [] **TTL** [] []: []

[] **FAR** []:

- [] **FAR** []: 10,000 []

- 原因: 原因 FAR 原因

原因:

- 原因 FAR 原因
- 原因 reason="global_limit" 原因 reason="far_limit"
- 原因 TTL 原因

原因 (DLDR)

原因 UPF 原因 IDLE 原因 UE 原因 SMF 原因 PCFP 原因

DLDR 原因:

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

SMF 原因 DLDR 原因:

1. 原因 AMF → gNB 原因 UE
2. 原因 UE 原因 RRC 原因
3. 原因 PCFP 原因 FAR
4. FAR 原因 BUFF+NOCP 原因 FORW
5. UPF 原因

DLDR 原因:

- upf_dldr_sent_total: 原因 DLDR 原因
- upf_dldr_send_errors: 原因 DLDR
- upf_buffer_notify_to_flush_duration_seconds: 原因 DLDR 原因

原因 原因 原因

原因

原因 BUFF 原因:

- FAR 原因 |= 0x04 原因 2原因

- `len: 2 (FORW)` → `len: 6 (FORW+BUFF)`
- `len: 6 (FORW+BUFF)`

len: 6 (FORW+BUFF) → len: 2 (FORW):

- FAR `len = 0x04` len BUFF
- `len: 6 (FORW+BUFF)`
- len IDLE UE len

len: 6 (FORW+BUFF) → len: 2 (FORW):

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

len:

- len FAR len
- len TEID/len
- len
- FAR len FORW len

len:

- len
- len
- len `reason="cleared"`

len

len Web UI: len len len

- len
- len
- len FAR len
- len FAR len
- len
- len/len FAR len

- 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 PDR0000 FAR
3. FAR 00 BUFF 00 → 0000000
4. UPF 0 SMF 00 DLDR
5. SMF 00 UE
6. UE 0000 gNB
7. SMF 00 FAR: 00 = 0x02 (FORW)
8. UPF 0000 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 □□□□□□□□□□□□□□

□□□□□□

□□□□□□ 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 □□□□

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 101

QFI (QoS 101):

- 6 101 5G QoS 101
- 1-9 101 QFI 9 = 101
- 101 5GC 101

101:

- 101 (0): 101
- 101 (101): 101

101 (MBR):

- 101
- 101 kbps 101
- **MBR = 0:** 101
- 101 MBR 101

101 (GBR):

- 101
- 101 kbps 101
- **GBR = 0:** 101
- **GBR > 0:** 101

QoS 101

101 (GBR = 0):

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

□□□ (GBR > 0):

QER ID: 2

QFI: 1

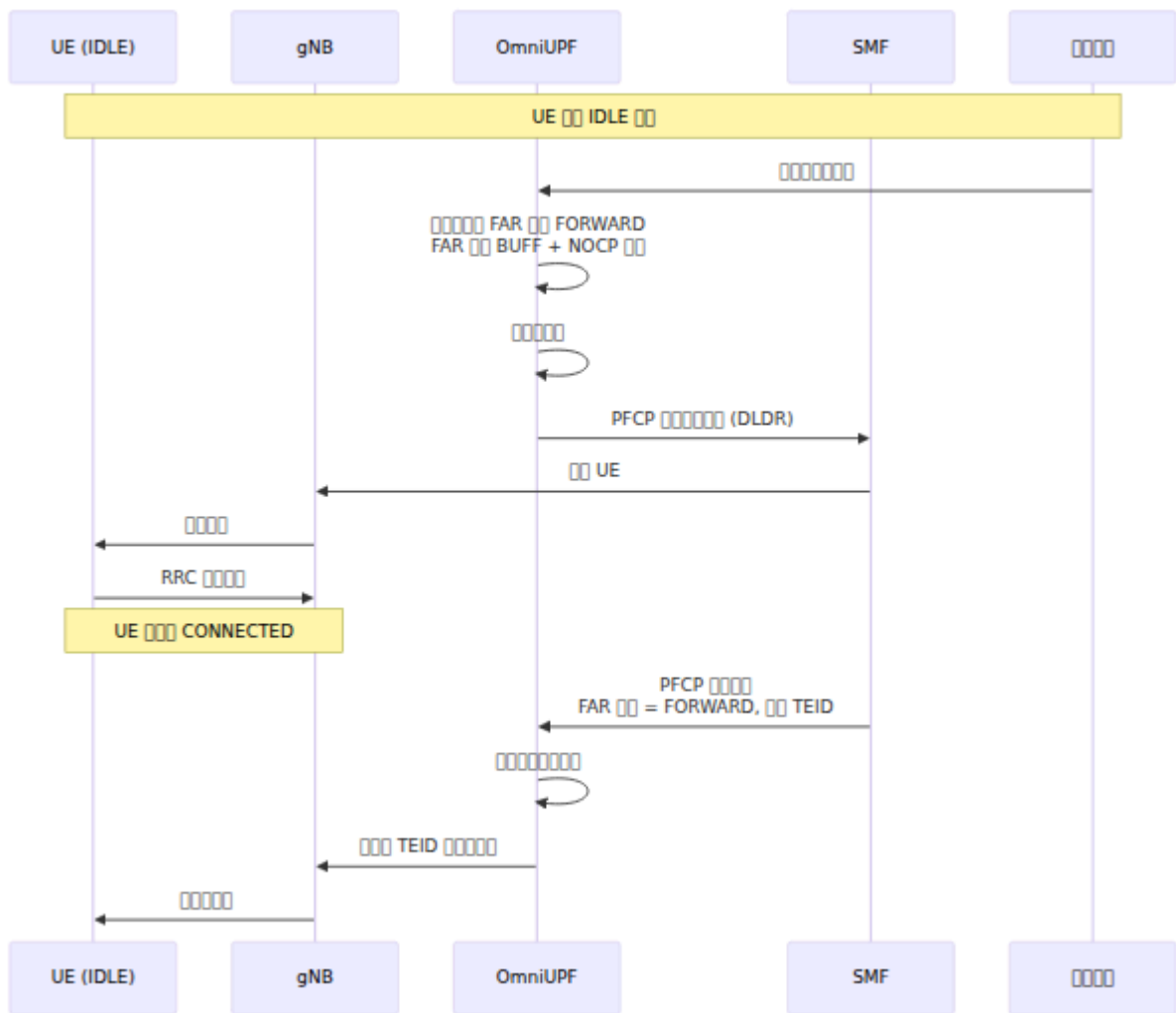
□□ MBR: 10000 kbps (10 Mbps)

□□ MBR: 10000 kbps (10 Mbps)

□□ GBR: 5000 kbps (5 Mbps)

□□ GBR: 5000 kbps (5 Mbps)

QoS



MBR

OmniUPF uses eBPF to implement MBR XDP

MBR

MBR: MBR = 0

UPF MBR

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

$$\text{tx_time} = (\text{packet_size_bytes} \times 8) \times (1,000,000,000 \text{ ns/sec}) / \text{MBR_kbps}$$

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

5. `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`

$$\begin{aligned} \text{tx_time} &= (1500 \text{ } \times 8 \text{ } / \text{ }) \times (1,000,000,000 \text{ ns/sec}) / \\ &100,000,000 \text{ bps} \\ &= 12,000,000,000 / 100,000,000 \\ &= 120 \text{ ns} \end{aligned}$$

`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
- MBR
-

:

- MBR `qer->ul_start`
- MBR `qer->dl_start`
-

MBR

MBR :

- : GTP-U/UDP/IP ~50-60
- : =
- :
- : 5ms MBR

:

MBR: 100 Mbps
 : ~95-98 Mbps GTP-U/UDP/IP

:

1. URR : `upf_urr*_volume_bytes`
2. : `(volume_delta_bytes × 8) / time_delta_seconds / 1000 = kbps`
3. QER MBR

GBR ()

: OmniUPF GBR GBR QER

GBR :

- GBR PFCP SMF
- GBR QER API

- GBR 流量
- GBR 流量

流量:

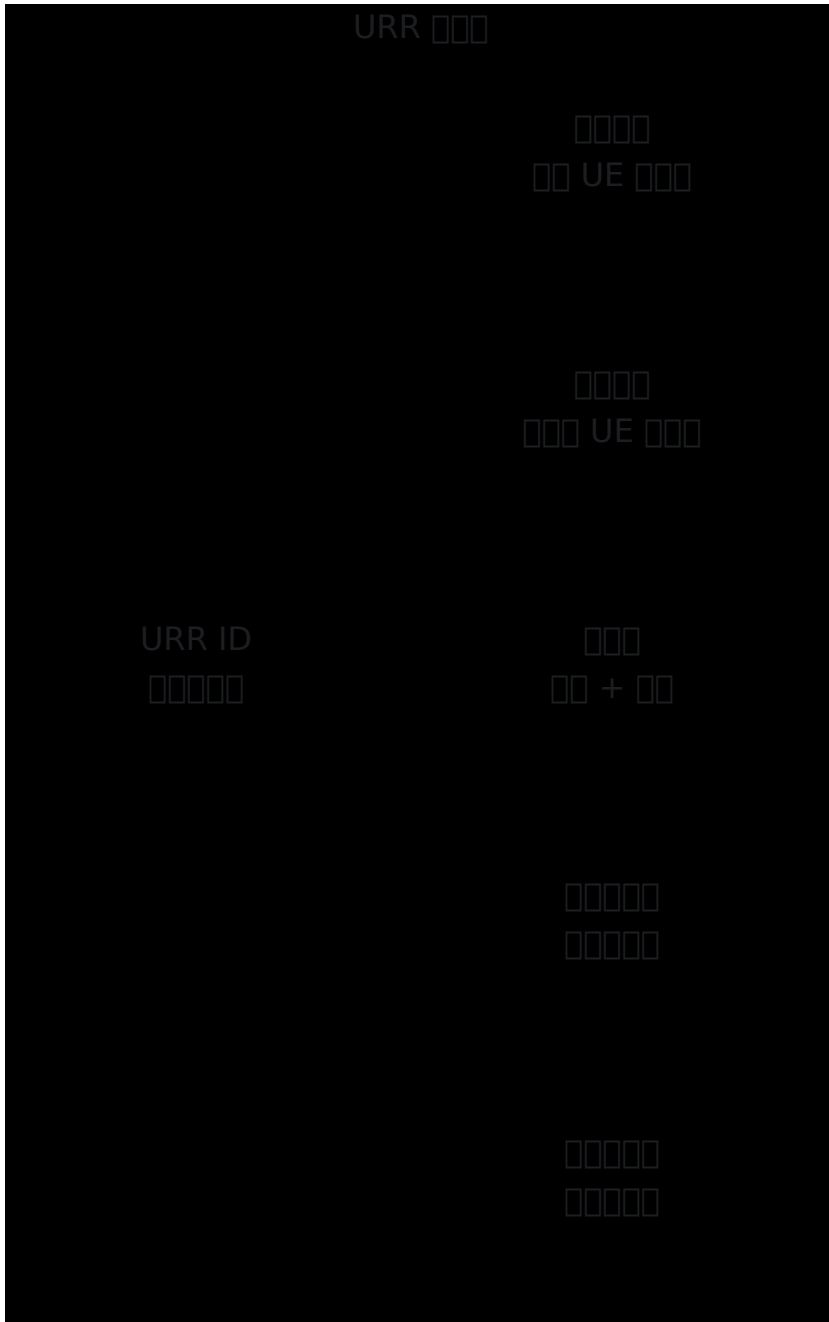
- GBR 流量
- 流量 eBPF QoS 流量

流量 (URR)

流量

URR 流量 SMF 流量

URR 00



000000

000000:

- UE 000000000000
- GTP-U 00000000
- IP 00000000

□□□□:

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

□□□:

- □□□□□□□□□□
- □□□□□□□

□□□□□□□□

URR □□□□□□□□□□□□□□

□□□□:

- □□□□□□□□□□□□□□
- □□: □ 1 GB □□□□□□□

□□□□:

- □□□□□□□□
- □□: □ 5 □□□□□□□

□□□□:

- □□□□□□□□
- □ QoS □□□□□□
- □□□□□□□

□□□□□□□□

Web UI □□□□□□□□□□□□□□□□

□□	□□
0 - 1023	B (□□)
1024 - 1048575	KB (□□□)
1048576 - 1073741823	MB (□□□)
1073741824 - 1099511627775	GB (□□□□)
1099511627776+	TB (□□□)

□□:

URR ID: 0

□□□□: 12.3 KB

□□□□: 9.0 KB

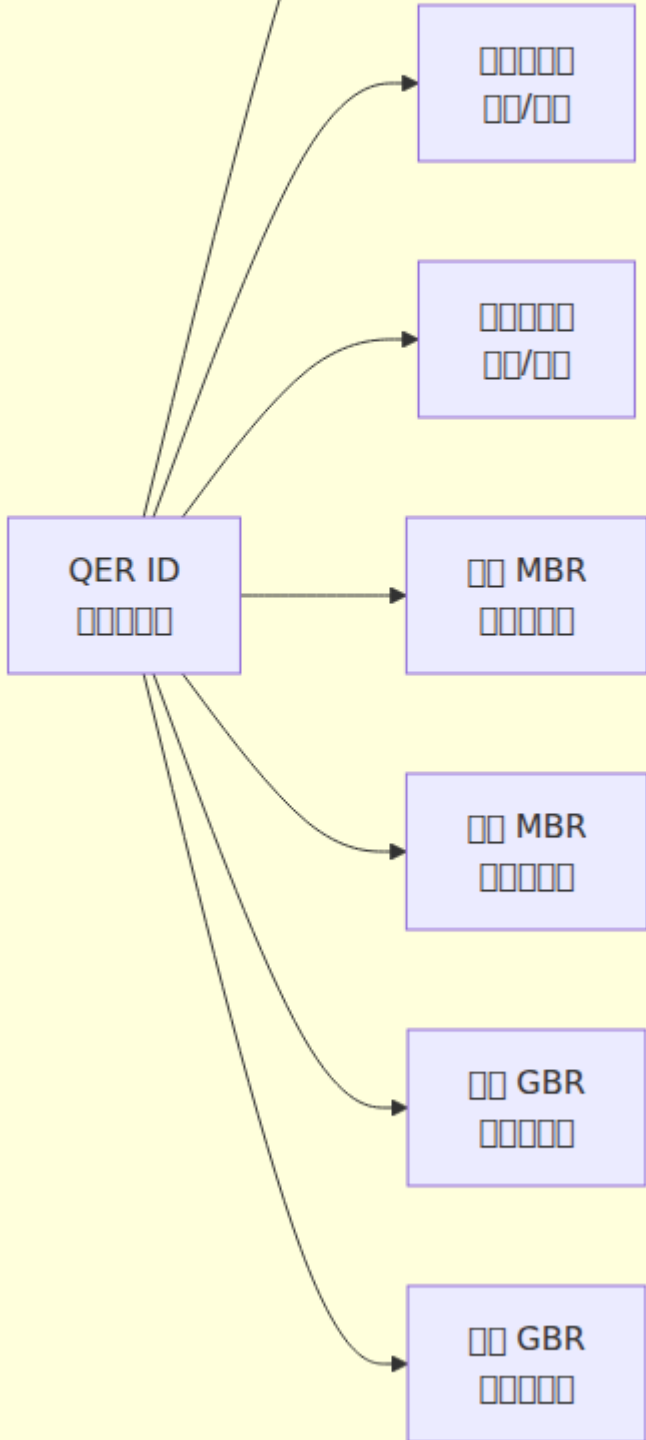
□□□: 21.3 KB

URR □□□□

QER ID

QER ID

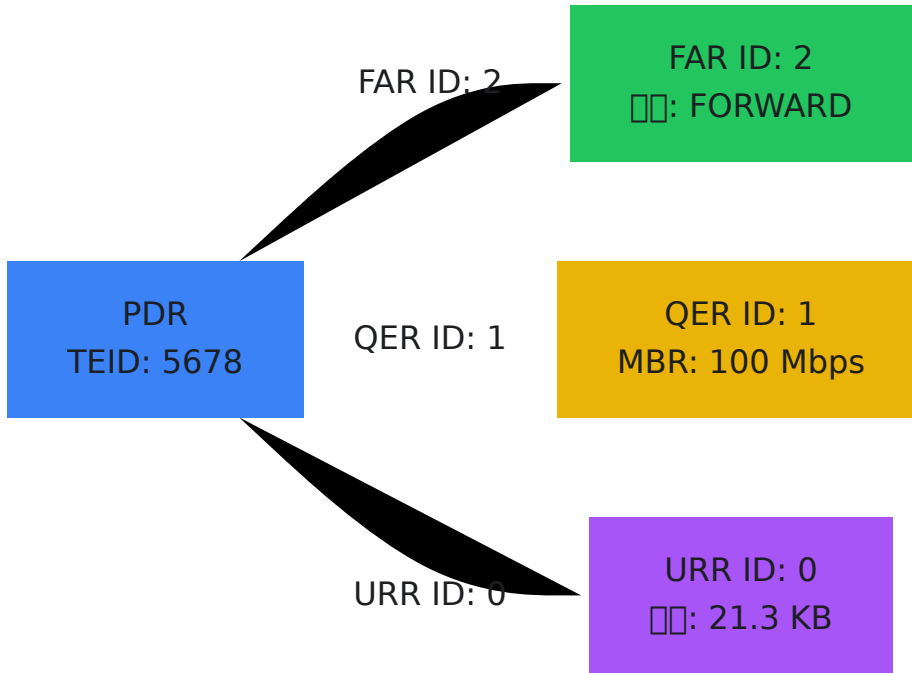
Core 5GC OmniCore OmniCall OmniRAN OmniCha



□□□□

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 []: No SDF

FAR ID 1 ([]):

[]: 2 (FORWARD)
[]: True
[] IP: 200.198.5.10
TEID: 5678

FAR ID 2 ([]):

[]: 2 (FORWARD)
[]: False

QER ID 1:

QFI: 9
[] MBR: 100000 kbps
[] MBR: 100000 kbps
[] GBR: 0 kbps
[] GBR: 0 kbps

URR ID 0:

[]: 12.3 KB
[]: 9.0 KB
[]: 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 → QERs
2. UE QoS QER ID
3. MBR MBR
4. URR

QoS:

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

QoS MBR

QoS

QoS URR:

1. QoS → URRs
- 2.
- 3.
- 4.

QoS:

-
-
-

QoS

QoS

QoS PDR:

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

FAR:

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

QER:

- 1.
2. MBR

QER:

1. → QERs
2. MBR
3. URR

FAR:

1. → FARs
2. FORWARD DROP
3. BUFFER

:

- 1.
- 2.
3. > 30
- 4.

5. FAR 設定

設定:

1. 初期値を 100,000 に設定
2. FAR 設定値を FAR 設定値 10,000 に設定
3. 初期値
4. 初期値

URR 設定

設定:

1. PDR 設定 URR ID
2. 初期値 PDR
3. FAR 設定値
4. URR ID 設定 URR 設定

初期値 **SMF**:

1. PCFP 設定
2. URR 設定/初期値
3. PCFP 設定

初期値

- **UPF 初期値** - OmniUPF 初期値
- **Web UI 初期値** - 初期値
- **初期値** - 初期値
- **初期値** - 初期値

OmniUPF 部署架构图

部署

1. 部署
2. 部署
3. 部署
4. 部署
5. PFCP 部署
6. 部署
7. XDP 与 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. Check OmniUPF status
systemctl status omniupf

# 2. Check PFCP status
curl http://localhost:8080/api/v1/upf_pipeline

# 3. Check eBPF status
ls /sys/fs/bpf/

# 4. Check XDP status
ip link show | grep -i xdp

# 5. Check logs
dmesg | tail -50
journalctl -u omniupf -n 50
```

□□□□

OmniUPF REST API

□□ **UPF** □□□

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

□□ **PFCP** □□□

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

□□□□□□□

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

□□ **eBPF** □□□□□

```
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
```

查看 XDP 设备

```
bpftool net list
```

查看 XDP 程序列表

```
bpftool prog show
```

查看 XDP 程序

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

查看

查看 N4 的 PFCP 设备

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

查看 N3 的 GTP-U 设备

```
# UPF kernel tcpdump kernel XDP kernel
# XDP kernel GTP-U

#
# 1. gNB to UPF kernel TAP
# 2. kernel/SPAN to N3
# 3. kernel VM

# kernel/UPF
# tcpdump -i <mirror_interface> -n udp port 2152 -w
/tmp/n3_capture.pcap

# kernel API
curl http://localhost:8080/api/v1/packet_stats
curl http://localhost:8080/api/v1/n3n6_stats
```

kernel

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

kernel

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

ARP

```
ip neigh show
```

kernel

kernel “eBPF kernel”

kernel

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

安装 eBPF 依赖

安装

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

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

# 验证
mount | grep bpf
```

安装

安装

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

安装 Linux 内核

安装

```
# 確認
uname -r

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

# 確認
uname -r # 5.15.0 >= 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

```
# Ubuntu/Debian
sudo apt update
sudo apt install libbpf-dev

# 確認
ldconfig -p | grep libbpf
```

確認

確認

確認

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

XXXXXXXXXX YAML XXXX

XXXXXX

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

```
# XXXXX
# - XXXXXXXXXXXXX
# - XXXXXXX
# - XXXXXXXXXXXXXXX
# - XXXXXXXXX
```

```
# XX YAML XXXX
cat > config.yml <<EOF
interface_name: [eth0]
xdp_attach_mode: generic
api_address: :8080
pfc_p_address: :8805
EOF
```

XXXXXXXXXXXXXXXX

XXX

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

XXXXXXXXXXXXXXXX

XXXXXX

```
# 確認ネットワークインターフェースの状態
ip link show

# IPアドレスを確認
ip addr show eth0

# config.yamlのinterface_nameを確認
interface_name: [ens1f0] # 確認

# /sys/class/net/ディレクトリを確認
ls /sys/class/net/
```

確認

エラー

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

確認 8080 8805 9090

確認

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

# 確認
sudo kill <PID>

# OmniUPF の確認
api_address: :8081
pfcg_address: :8806
metrics_address: :9091
```

PFCP Node ID

Issue

```
ERR0[0000] invalid pfc_node_id: must be valid IPv4 address
```

PFCP Node ID must be a valid IPv4 address

Example

```
# Example IP address
pfc_node_id: 10.100.50.241

# Localhost
# pfc_node_id: localhost
# pfc_node_id: upf.example.com
```

PFCP Node

PFCP Node

Issue

- Web UI “Node”
- SMF “PFCP Node”

Issue

```
# 1. Check PFCP connections
sudo netstat -ulpn | grep 8805

# 2. Check iptables and ufw status
sudo iptables -L -n | grep 8805
sudo ufw status

# 3. Check PFCP traffic
tcpdump -i any -n udp port 8805 -vv

# 4. Check API endpoint for PFCP pipeline
curl http://localhost:8080/api/v1/upf_pipeline
```

Check connections

Check **PFCP**

Check

```
# Allow PFCP UDP 8805
sudo ufw allow 8805/udp
sudo iptables -A INPUT -p udp --dport 8805 -j ACCEPT
```

Check **PFCP** ID

Check

```
# PFCP ID configuration N4 IP
pfcpc_node_id: 10.100.50.241 # N4 IP
```

Check **SMF**

Check

```
# 测试 SMF 连通性
ping <SMF_IP>

# 测试 SMF 路由
ip route get <SMF_IP>

# 配置路由
sudo ip route add <SMF_NETWORK>/24 via <GATEWAY>
```

SMF 配置 UPF IP

配置

- 配置 SMF 与 UPF 连接
- 配置 SMF 与 UPF 的 pfcpc_node_id IP
- 配置 SMF 与 UPF 的 N4 接口

配置 PFCP 配置

配置

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

配置

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

# 查看日志
journalctl -u omniupf -f | grep heartbeat
```

配置

配置

□□□□

```
# □□□ SMF □□□□□□  
ping -c 100 <SMF_IP> | grep loss
```

```
# □□□□□□□□□□□□  
# - □□□□□□  
# - □□□□□□/□□□□□□  
# - □□□□□□
```

□□□□□□□□

□□□□□

```
# □□□□□□  
heartbeat_interval: 30 # □ 5 □□□ 30 □  
heartbeat_retries: 5 # □□□□□□  
heartbeat_timeout: 10 # □□□□□□
```

□□□□□□□□

□□□□□□□□□□**RX/TX** □□□ **0**□

□□□

- □□□□□□ 0 RX/TX □□□
- UE □□□□□□□□

□□□

```
# 1. XDP
ip link show eth0 | grep xdp

# 2. UP
ip link show eth0

# 3. XDP
# tcpdump XDP GTP-U
curl http://localhost:8080/api/v1/packet_stats
```

XDP

```
# OmniUPF XDP
sudo systemctl restart omniupf

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

```
#
sudo ip link set eth0 up

#
ethtool eth0 | grep "Link detected"

#
```

```
# config.yml
interface_name: [ens1f0] # 'ip link show'
```

- RX TX
- > 1%

```
#
curl http://localhost:8080/api/v1/xdp_stats | jq '.drop'

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

#
watch -n 1 'curl -s http://localhost:8080/api/v1/packet_stats | jq ".total_rx, .total_tx, .total_drop"'
```

PDR TEID UE IP

```
# 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

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

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

2. QER

```
# QER
curl http://localhost:8080/api/v1/qer_map | jq '.data[] |
{qer_id, ul_gate: .ul_gate_status, dl_gate: .dl_gate_status}'
```

3. URR

```
# URR
curl http://localhost:8080/api/v1/urr_map | jq '.data[] |
select(.urr_id == 0)'
```

```
#
# throughput_kbps = (volume_delta_bytes × 8) /
time_delta_seconds / 1000
```

4. MBR

- MBR 95-98%
- MBR
- MBR

- MBR SMF PFCP QER MBR
- SMF
- SMF QoS

MBR

OmniUPF `iptables` eBPF MBR `iptables`
`iptables -M MBR`

`iptables`

- **VoIP** MBR `iptables` G.711 = ~80 kbps
- MBR > `iptables` + `iptables` 1080p = ~5-10 Mbps
- `iptables` 5ms `iptables`

`iptables`

`iptables`

- RX N3 `iptables` TX N3 `iptables`
- RX N6 `iptables` TX N6 `iptables`

`iptables`

```
# iptables N3/N6 iptables XDP iptables  
curl http://localhost:8080/api/v1/n3n6_stats  
curl http://localhost:8080/api/v1/packet_stats  
  
# iptables tcpdump iptables XDP iptables GTP-U iptables  
# iptables API iptables xdpdump iptables  
# iptables "XDP iptables" iptables
```

`iptables` **RX N3** `iptables` **TX N6** `iptables`

`iptables` FAR `iptables` N6 `iptables`

`iptables`

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

```
# UE N6 IPADDRESS
ip route get 8.8.8.8 # IPADDRESS
```

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

IPADDRESS RX N6 TX N3

IPADDRESS PDR IP GTP

IPADDRESS

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

```
# UE FAR ACTION OUTER_HEADER_CREATION
curl http://localhost:8080/api/v1/sessions | jq '.[].fars[] |
.outer_header_creation'
```

```
# UE gNB IPADDRESS
ping <GNB_N3_IP>
```

XDP eBPF

IPADDRESS XDP IPADDRESS XDP

IPADDRESS XDP IPADDRESS

IPADDRESS

```
ERROR[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. 取得セッション
curl http://localhost:8080/api/v1/sessions | jq '.*[] | {seid,
uplink_teid, created_at}'

# 2. SMF 取得
# SMF 取得 API

# 3. 監視
watch -n 5 'curl -s http://localhost:8080/api/v1/map_info | jq ".
[] | select(.map_name==\"pdr_map_downlin\") | .usage_percent"'
```

設定

```
# config.yml 設定
max_sessions: 200000 # 100000

# 設定
pdr_map_size: 400000
far_map_size: 400000
qer_map_size: 200000
```

OmniUPF 設定

□□□□

□□□□□□□□□□□□□□

□□□

- □□□ < 1 Gbps □□ NIC □□□□
- CPU □□□□

□□□

```
# □□□□□□□□  
curl http://localhost:8080/api/v1/packet_stats | jq '.total_rx,  
.total_tx'  
  
# □□ NIC □□  
ethtool -S eth0 | grep -i drop  
  
# □□ XDP □□  
ip link show eth0 | grep xdp
```

□□□□□

□□□□ **XDP** □□

□□□□□

```
# □□□□□□□□□□□□□□□□  
xdp_attach_mode: native # □□□□ XDP □ NIC/□□□□
```

□□□□

□□□□□

```
# 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'
```

□□□□□□□□

□□□□□

```
# □□□□□□□□□□□□  
curl http://localhost:8080/api/v1/upf_buffer_info | jq '.buffers[]  
| {far_id, packet_count, direction}'  
  
# □□□□□□□□□□□□  
# □□□ OmniUPF □□□ PFCP □□□□□□□ FAR□
```

FIB □□□□

□□□□□

```
# □□□□□□□□□□□□□□□□  
# □□ BPF_ENABLE_ROUTE_CACHE=1 □□  
  
# □□□□□  
# □□□□□□□□□□□□□□□□□□□□
```

□□□□□□□□□□□□□□

□□□

- □□□□□□□□□□
- NIC □□ RX □□

□□□

```
# 检查 NIC 错误
ethtool -S eth0 | grep -E "drop|error|miss"

# 查看网卡统计
ethtool -g eth0

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

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
```

网络性能优化

网络性能优化 XDP 技术

Proxmox VM 网络 XDP 配置

简介

- 网络性能优化 XDP 技术
- 网络性能优化

网络性能优化 SR-IOV

配置

配置 1 网络性能优化

```
xdp_attach_mode: generic
```

SR-IOV

```
# 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

网卡

- 网卡 < 100 Mbps

VirtualBox 网卡 SR-IOV 网卡 XDP

网卡

```
# 网卡配置
xdp_attach_mode: generic

# VirtualBox 网卡
# - VirtIO-Net 网卡
# - "网卡"网卡
# - CPU 网卡
# - NAT

# KVM/Proxmox 网卡
```

NIC

NIC 网卡 XDP

网卡

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

网卡

```
# 检查 NIC 驱动
ethtool -i eth0 | grep driver

# 检查 XDP 支持
modinfo <driver_name> | grep -i xdp

# 检查 XDP 是否加载
ip link show | grep -B 1 "xdpgeneric\|xdpdrv\|xdpoffload"
```

检查

检查 **1**

```
xdp_attach_mode: generic
```

检查 **2** **NIC**

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

# 检查
# 检查
# 访问 https://downloadcenter.intel.com/
```

检查 **3** **NIC**

```
# 检查 XDP 支持的 NIC
# - Intel X710/E810
# - Mellanox ConnectX-5/ConnectX-6
# - Broadcom BCM57xxx/bnxt_en
```

检查

检查

- XDP
- NIC

```
#  
dmesg | tail -100  
  
#  
journalctl -k | grep -E "BUG:|panic:"
```

```
# 1.  
sudo apt update  
sudo apt upgrade  
sudo reboot  
  
# 2. XDP  
xdp_attach_mode: native  
  
# 3.  
xdp_attach_mode: generic  
  
# 4. NIC Linux
```

- SMF
- UE PDU

PFCP

□□□

```
# □□ OmniUPF □□□□□□□□  
journalctl -u omniupf | grep -i "session establishment"  
  
# □□ PCFP □□□□  
curl http://localhost:8080/api/v1/sessions | jq 'length'  
  
# □□□□□□□□□□ PCFP □□  
tcpdump -i any -n udp port 8805 -w /tmp/pfcp_session.pcap
```

□□□□□

□□□□□□

□□□□□

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

□□□ **PDR/FAR** □□

□□□□□

```
# □□ OmniUPF □□□□□□□□  
journalctl -u omniupf | grep -E "invalid|error" | tail -20  
  
# □□□□□  
# - □□□ UE IP □□□0.0.0.0 □□□□  
# - □□□ TEID□□□ □□□□  
# - PDR □□ FAR  
# - □□□ FAR □□  
  
# □□ SMF □□□□□□□□
```

□□□□□□□□ **UEIP/FTUP** □

□□□□

```
# □□□□□□□□□□□□  
feature_ueip: true # □ UPF □□ UE IP  
ueip_pool: 10.60.0.0/16  
  
feature_ftup: true # □ UPF □□ F-TEID  
teid_pool: 100000
```

□□□□

□□□□□□□□□□□□

□□□

- □□□□□□□□□□
- □□□□□□□□□□

□□□

```
# □□□□□□  
curl http://localhost:8080/api/v1/upf_buffer_info  
  
# □□□□ FAR □□□  
curl http://localhost:8080/api/v1/upf_buffer_info | jq '.buffers[]  
| {far_id, packet_count, oldest_packet_ms}'  
  
# □□□□□□□□  
watch -n 5 'curl -s http://localhost:8080/api/v1/upf_buffer_info |  
jq ".total_packets_buffered"'
```

□□□□□□□□

FAR □□□□ FORWARD

□□□SMF □□□□ PFCP □□□□□□□□ FAR

□□□□

```
# □□ FAR □□  
curl http://localhost:8080/api/v1/sessions | jq '.[].fars[] |  
{far_id, applied_action}'  
  
# □□ BUFF = 1□□□□  
# □□ FORW = 2□□□□  
  
# □□□□ BUFF □□□□□□ SMF□  
# - □□ PFCP □□□□□□  
# - □□ FAR □□□ FORW □□
```

□□ **TTL** □□

□□□□□□ FAR □□□□□□

□□□□

```
# □□□□ TTL  
buffer_packet_ttl: 60 # □ 30 □□□ 60 □
```

□□□□

□□□□ FAR □□□□□□□□

□□□□

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

□□□□

□□□□□□

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

```

~~~~~

~~~~~

```

# 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**~~~~~

```

# Linux tcpdump VM
# hypervisor UPF N3
# VM
tcpdump -i eth1 -n udp port 2152 -w /tmp/n3_virtual.pcap

```

## □□□□□□□□

- XDP □□□□□□□□□□
- □□□□ NIC □□□□□□□□□□
- □□□□ tcpdump □ XDP □□□□□□□□□□□□□□
- □□□□□□□□□□□□□□ UPF □□□□

## □□□□ **UPF** □□□□□□□□□□

- □ PFCP □□□□UDP 8805□ - □□□□□□□□ XDP □□
- □ API □□□□□□
- □ GTP-U □□□□UDP 2152□ - □□□□□□□□ XDP □□

---

## □□□□□

### □□□□□□□□□□□□□□□□□□

#### 1. □□□□□□□□

```
# □□□□
uname -a
cat /etc/os-release

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

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

# □□□□
ip addr > /tmp/network.txt
ip route >> /tmp/network.txt
ethtool eth0 >> /tmp/network.txt
```

## 2. 网络架构

- OmniUPF 部署
  - Linux 网络
  - 网络策略
  - 网络性能优化
  - 网络故障排查
  - 网络升级
- 

## 网络策略

- 网络策略 - 网络策略
- 网络策略 - eBPF/XDP 网络策略
- 网络策略 - 网络策略
- 网络策略 - 网络策略 Prometheus 部署
- **PFCP** 网络策略 - PFCP 网络策略
- 网络策略 - PDR/FAR/QER/URR 部署
- 网络策略 - UPF 网络策略

# OmniUPF 安裝 / 設定說明

## 目錄

1. 簡介
2. 需求
3. PFCP 設定
4. 網路設定
5. 安裝
6. 設定
7. 設定 URL
8. API
9. Prometheus 設定
10. 測試

## 安裝

本說明書將介紹如何安裝 OmniUPF。UPF 安裝需要 MikroTik 的 mangle 規則和 DNAT 規則。

安裝 OmniUPF 需要 SMF 的 `redirect_information` 和 PFCP 設定。OmniUPF 安裝後，需要設定以下項目：

- **DNS** 設定：需要設定 Apple、Android、Windows 的 DNS 伺服器。
- **IP** 設定：需要設定 CAPTCHA 的 IP 地址。
- 設定其他相關參數。

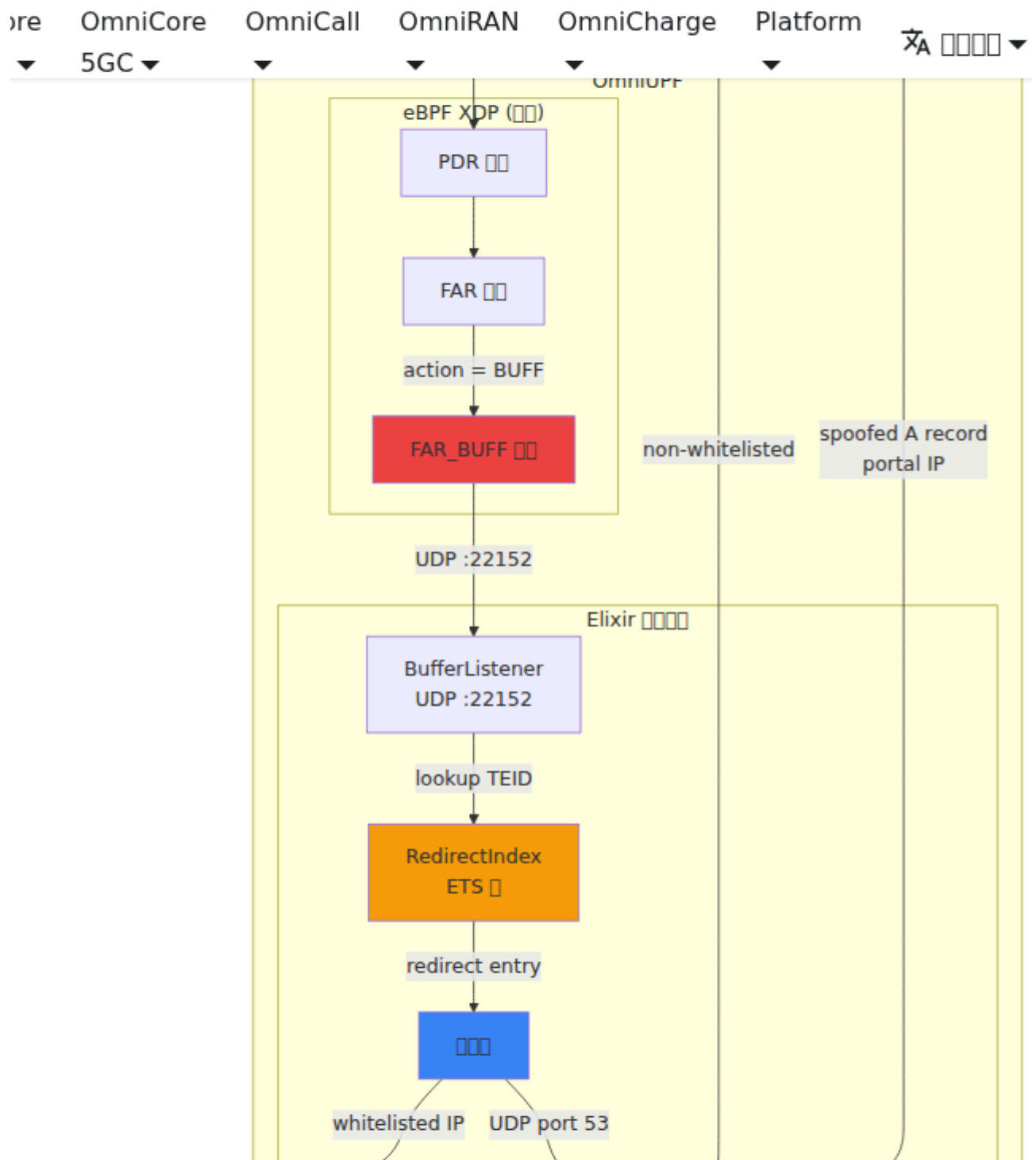
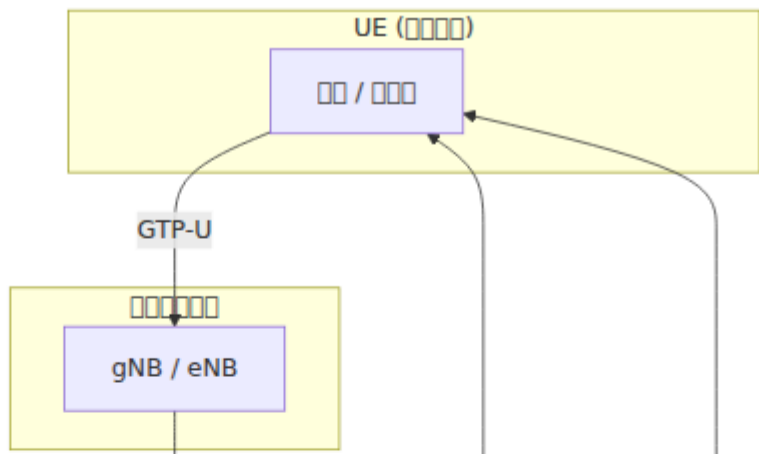
安裝 OmniUPF 後，需要設定 SMF 的 FAR 規則和 FORW 規則。

## 設定

- 設定 **eBPF** 規則：需要設定 `FAR_BUFFER` 規則。

- 000000 -- 000000000000 IP 0000 URL00 SMF 0 redirect\_information IE 00
  - 000000 **UPF** -- SMF 000“000000”0UPF 0000000000
  - 000000 -- 00 FAR 000 FORW000000000000 GTP 000000 UE
-





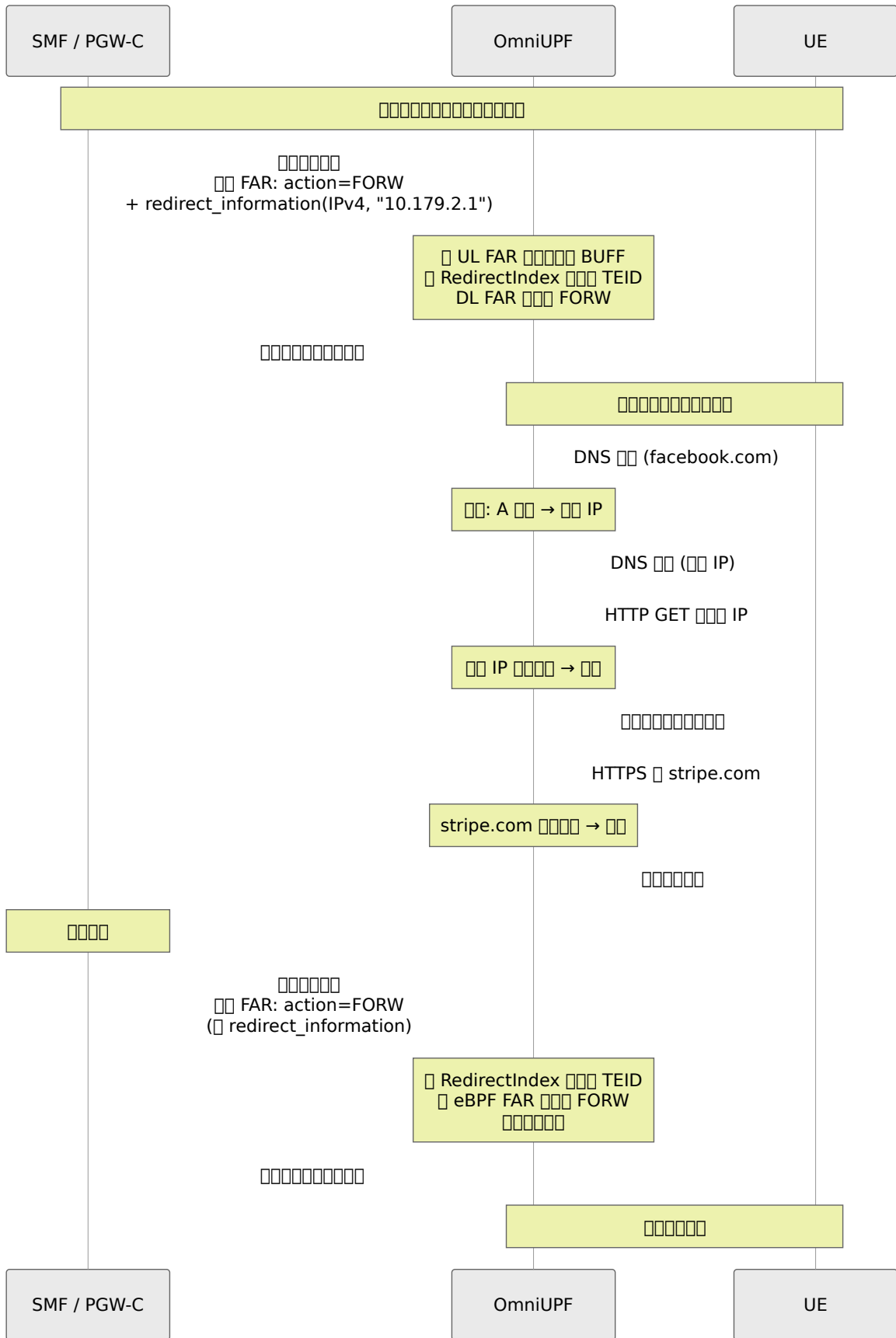


2. PFCP SGW FAR `teid` PGW UL PDR TEIDs `outer_header_creation` FAR PGW N9 FAR
3. PGW UL FAR SGW gNB UL FAR eBPF `BUFF` eBPF eNB SGW TEID PGW TEID
4. **SGW** **gNB** **UL PDRs** `RedirectIndex` PGW UL PDRs SGW UL PDR TEID eBPF eNB
5. UE **SGW** **DL FAR** eNB FAR `remoteip != n3_address` DL TEID — eNB TEID

**UPF** SGW SMF UPF UL FAR TEID — SGW UL PDRs

---

# PFCP □□□□



## PFCEP IEs

SMF FAR forwarding\_parameters redirect\_information

| IE                    |                                               |
|-----------------------|-----------------------------------------------|
| apply_action          | SMF FORWUPF BUFF                              |
| redirect_information  |                                               |
| forwarding_parameters | redirect_information<br>outer_header_creation |

3GPP TS 29.244 8.2.20-1

|         |   | UPF                         |
|---------|---|-----------------------------|
| IPv4    | 0 | IPv4 IP                     |
| IPv6    | 1 | IPv6 IP                     |
| URL     | 2 | URL IP IP URL UPF — API Web |
| SIP URI | 3 |                             |

## 

DNS IP

# IP

| OS                       | Domain                                     | Response                                                           |
|--------------------------|--------------------------------------------|--------------------------------------------------------------------|
| <b>Apple (iOS/macOS)</b> | <code>captive.apple.com</code>             | HTTP 200<br><HTML><HEAD><TITLE></TITLE></HEAD><BODY></BODY></HTML> |
| <b>Android</b>           | <code>connectivitycheck.gstatic.com</code> | HTTP 204                                                           |
| <b>Windows</b>           | <code>www.msftconnecttest.com</code>       | HTTP 200<br>Microsoft Connect Test                                 |
| <b>Samsung</b>           | <code>connectivitycheck.samsung.com</code> | HTTP 200                                                           |

IP address list

- iOS/Android
- Windows

IP address list URL

## UPF

UPF configuration files: `.deb`, `/etc/omniupf/runtime.exs`, `rel/env.sh`, `RELEASE_CONFIG_DIR=/etc/omniupf`, Erlang `config/runtime.exs`, UPF

```
#
=====
#  /
#
=====

#
walled_garden_enabled = true

# IP /
walled_garden_portal_ip = "10.179.2.1"

# DNS
walled_garden_dns_resolver = "8.8.8.8"

#
# "*" "api.stripe.com"
walled_garden_whitelist = [
    "stripe.com",
    "*.stripe.com",
    "js.stripe.com",
    "hcaptcha.com",
    "*.hcaptcha.com",
    "newassets.hcaptcha.com",
]
```



## DNS

| Host         | Hosts                                                   | Hosts                           |
|--------------|---------------------------------------------------------|---------------------------------|
| stripe.com   | stripe.com<br>api.stripe.com<br>js.stripe.com           | evilstripe.com<br>notstripe.com |
| *.stripe.com | api.stripe.com<br>js.stripe.com<br>dashboard.stripe.com | stripe.com<br>evilstripe.com    |
| hcaptcha.com | hcaptcha.com<br>newassets.hcaptcha.com                  | evihcaptcha.com                 |

stripe.com foo.stripe.com evilstripe.com

## IP

DNS IP

- api.stripe.com
- DNS 104.18.7.25
- 104.18.7.25 IP
- HTTP/HTTPS 104.18.7.25

IP

## hCaptcha

Stripe hCaptcha

```
walled_garden_whitelist = [  
  # Stripe  
  "stripe.com",  
  "*.stripe.com",  
  
  # CAPTCHA  
  "hcaptcha.com",  
  "*.hcaptcha.com",  
  
  # Google Fonts  
  "fonts.googleapis.com",  
  "fonts.gstatic.com",  
  
  # Example CDN  
  "cdn.example.com",  
]
```

---

## URLs

PFCP SMF FAR `redirect_information` IE

- **IPv4** IP
- **IPv6** IPv6 IP
- **URL** FAR DNS IP UPF URL — API

MVNO

| <input type="checkbox"/><br><input type="checkbox"/>   | SMF redirect_information                                                              | IP                                                                                 |                                                    |
|--------------------------------------------------------|---------------------------------------------------------------------------------------|------------------------------------------------------------------------------------|----------------------------------------------------|
| <input type="checkbox"/><br><input type="checkbox"/> A | IPv4: 10.179.2.1                                                                      | 10.179.2.1                                                                         | 10.179.2.1                                         |
| <input type="checkbox"/><br><input type="checkbox"/> B | IPv6: 2001:db8::1                                                                     | 2001:db8::1                                                                        | 2001:db8::1                                        |
| <input type="checkbox"/><br><input type="checkbox"/> C | URL:<br><a href="https://topup.mvno.com/recharge">https://topup.mvno.com/recharge</a> | <a href="https://topup.mvno.com">topup.mvno.com</a><br><input type="checkbox"/> IP | <a href="https://topup.mvno.com">https://topup</a> |

UPF  IP  URL  API

# API

/v1/walled\_garden  /api  api\_port   
 8080  Phoenix HTTP

## GET /v1/walled\_garden

IP  CIDR

```
{
  "redirect_count": 2,
  "redirects": [
    {
      "teid": "0x4000",
      "session_seid": 1,
      "portal_ip": "10.179.2.1",
      "redirect_url": null,
      "ue_ip": "10.60.0.1",
      "gnb_ip": "10.179.1.21",
      "dl_teid": "0x5000",
      "far_global_id": 42
    },
    {
      "teid": "0x4001",
      "session_seid": 2,
      "portal_ip": "10.179.2.2",
      "redirect_url": "https://topup.mvno.com",
      "ue_ip": "10.60.0.2",
      "gnb_ip": "10.179.1.21",
      "dl_teid": "0x5001",
      "far_global_id": 43
    }
  ],
  "whitelisted_ips": [
    {"ip": "10.179.2.1", "type": "portal"},
    {"ip": "104.18.7.25", "type": "resolved"},
    {"ip": "104.18.6.25", "type": "resolved"}
  ],
  "whitelisted_cidrs": ["192.168.0.0/24"]
}
```

□□□□

| 필드                                     | 설명                                  |
|----------------------------------------|-------------------------------------|
| <code>redirect_count</code>            | 리다이렉트 횟수                            |
| <code>redirects[].teid</code>          | 리다이렉트 TEID                          |
| <code>redirects[].session_seid</code>  | PFCP 세션 SEID                        |
| <code>redirects[].portal_ip</code>     | 포털 IP                               |
| <code>redirects[].redirect_url</code>  | SMF 리다이렉트 URL (리다이렉트 URL이 null인 경우) |
| <code>redirects[].ue_ip</code>         | UE IP                               |
| <code>redirects[].gnb_ip</code>        | GTP-U gNB IP                        |
| <code>redirects[].dl_teid</code>       | GTP-U 다운링크 TEID                     |
| <code>redirects[].far_global_id</code> | FAR ID                              |
| <code>whitelisted_ips</code>           | 리다이렉트 허용 IP + DNS                   |
| <code>whitelisted_cidrs</code>         | 리다이렉트 허용 API CIDR                   |

## POST /v1/walled\_garden

리다이렉트 SEID 및 PFCP 세션 ID를 사용하여 API를 호출합니다. — 리다이렉트 PFCP 세션 ID 및 FAR ID를 사용하여 `redirect_information` API를 호출하여 리다이렉트 SMF 세션 ID를 반환합니다.

요청 본문

```
{
  "seid": 1,
  "url": "http://10.179.2.1/"
}
```



## DELETE /v1/walled\_garden/:seid

刪除由 N9 遠端 SGW 遠端 eBPF 遠端  
遠端 `action=FORW` 遠端 TEID 遠端

遠端 `:seid` — 遠端 SEID

遠端 **200 OK**

```
{"status": "redirect removed", "info": {"seid": 1}}
```

遠端

|     |    |
|-----|----|
| 遠端  | 遠端 |
| 404 | 遠端 |

## GET /v1/walled\_garden/whitelist

遠端 IP 遠端 IP 遠端 DNS 遠端 IP 遠端 API 遠端 CIDR 遠端

遠端

```
{  
  "ips": [  
    {"ip": "10.179.2.1", "type": "portal"},  
    {"ip": "104.18.7.25", "type": "resolved"}  
  ],  
  "cidrs": ["192.168.100.0/24"]  
}
```

## POST /v1/walled\_garden/whitelist

遠端 IP 遠端 CIDR 遠端 UPF 遠端 — 遠端  
遠端 `runtime.exs` 遠端 `walled_garden_whitelist`

IP

```
{"ip": "203.0.113.10"}
```

CIDR

```
{"cidr": "192.168.100.0/24"}
```

ip cidr CIDR IP IP DNS

200 OK — IP

```
{"status": "added", "ip": "203.0.113.10"}
```

200 OK — CIDR

```
{"status": "added", "cidr": "192.168.100.0/24"}
```

| 400 | IP CIDR |
|-----|---------|

# Prometheus

## Gauges

upf\_walled\_garden\_active\_redirects Gauge:

```
# upf_walled_garden_active_redirects
```

## Counters

`upf_walled_garden_packets_intercepted_total`: Counter: Total number of intercepted packets

`upf_walled_garden_packets_dropped_total`: Counter: Total number of dropped DNS packets

`upf_walled_garden_packets_forwarded_total`: Counter:

- `dst_ip` - Destination IP: Total number of forwarded packets per destination IP

`upf_walled_garden_bytes_forwarded_total`: Counter:

- `dst_ip` - Destination IP: Total number of forwarded bytes per destination IP

`upf_walled_garden_dns_spoofed_total`: Counter:

- `domain` - Domain: Total number of spoofed DNS responses per domain

`upf_walled_garden_dns_forwarded_total`: Counter:

- `domain` - Domain: Total number of forwarded DNS responses per domain

□□□□

```
# □□□□□□□□□□
upf_walled_garden_active_redirects

# □□□□□□□□/□□
rate(upf_walled_garden_packets_intercepted_total[5m])

# □□□□□□□□□□□□□□□□
rate(upf_walled_garden_packets_dropped_total[5m])

# □□□□□□□□□□□□□□/□□
sum by (dst_ip)
(rate(upf_walled_garden_bytes_forwarded_total[5m]))

# □□□□□□□□□□ 5 □□□□□□□□
topk(5, sum by (dst_ip)
(rate(upf_walled_garden_bytes_forwarded_total[5m])))

# □□□□□□□□□□
topk(10, sum by (domain)
(rate(upf_walled_garden_dns_spoofed_total[5m])))

# □□□□□□□□□□
sum by (domain) (rate(upf_walled_garden_dns_forwarded_total[5m]))

# □□□□□□□□□□
sum(rate(upf_walled_garden_packets_dropped_total[5m]))
/ sum(rate(upf_walled_garden_packets_intercepted_total[5m]))
```

---

□□□□

□□□□□□□□□□□□

□□: □□□□□□□□□□ API □□□□□□□□□□□□□□□□□□□

□□□□:

- 配置 IP 地址
- 配置 URL
- DNS 配置 UE GTP-U 配置

配置:

1. 配置 curl http://<portal\_ip>/
2. 配置 URL Apple GET /hotspot-detect.html
3. GET /v1/walled\_garden 配置
4. Prometheus upf\_walled\_garden\_dns\_spoofed\_total 配置
5. GTP-U DL FAR 配置 FORWARD

## 配置

配置: 配置 Stripe 配置

配置:

- 配置
- 配置 CDN IP 配置
- 配置

配置:

1. GET /v1/walled\_garden — whitelisted\_ips 配置 IP
2. Prometheus: upf\_walled\_garden\_dns\_forwarded\_total{domain="stripe.com"} 配置
3. js.stripe.com m.stripe.network
4. upf\_walled\_garden\_bytes\_forwarded\_total dst\_ip 配置

## 配置

配置: SMF 配置 redirect\_information 配置

配置:

- walled\_garden\_enabled false

- `redirect_information` IE `0x0000 FAR 0x0000 0x FAR 0x`
- FAR `0x00000000` BUFF

0x0000:

1. `walled_garden_enabled = true`
2. `GET /v1/pfcp_sessions` — `0x0000 FAR 0x0000 FAR 0x0000 0x04` BUFF
3. `GET /v1/walled_garden` — `0x0000 TEID 0x000000000000`
4. `UPF 0x000000000000 redirect_info` 0x

0x000000000000

0x: 0x0000000000000000

0x0000:

- SMF `0x0000000000000000`
- BUFF->FORW `0x00000000`

0x0000:

1. `GET /v1/walled_garden` — `0x000000000000`
2. `GET /v1/pfcp_sessions` — `0x0000 FAR 0x0000 0x02` FORW
3. `SMF 0x0000000000000000`
4. `UPF 0x0000000000000000`

0x0000 **TEID** 0x000000000000

0x: DNS `0x000000000000` UE GTP-U `0x00000000` gNB IP `0x0000` eNB `0x00000000` TEID

0x: `4G/SGW+PGW 0x0000` SGW DL FAR `0x0000` eNB TEID `0x0000000000000000` TEID `0x000000000000` eNB `0x0000` TEID `0x0000000000000000` SGW `0x0000` DL FAR `0x0000` — `0x0000000000000000` TEID

0x0000: `0x00000000` DL TEID `0x0000` SGW `0x0000` DL FAR `0x0000`

`outer_header_creation` `remoteip != n3_address` FAR `0x0000000000000000` `0x0000000000000000` TEID `0x0000` DNS `0x0000000000000000` TEID

1. GET /v1/walled\_garden — dl\_teid
2. GET /v1/pfcp\_sessions — SGW FAR gNB FAR teid remoteip
3. SGW RedirectIndex DELETE /v1/walled\_garden/:seid SMF

## 304

304: HTTP 304 Not Modified

304: HTTP 304 Not Modified Web /hotspot-detect.html/generate\_204 If-Modified-Since If-None-Match 304 304

304:

1. UPF HTTP Web
2. 200 304
3. Cache-Control: no-store ETag/Last-Modified
4. curl -v -H "If-None-Match: foo" http://<portal\_ip>/hotspot-detect.html — 200 304

## IP

IP IPPROTO\_RAW 255 Erlang :socket IP eBPF

GTP-U IP UE IP IPPROTO\_RAW IP UPF N6

IP

| ❌                                                  | 🔍                                    | 📄                                              |
|----------------------------------------------------|--------------------------------------|------------------------------------------------|
| 🔍                                                  | EPERM — 📄<br>📄 root 📄<br>CAP_NET_RAW | 📄 OmniUPF 📄 root 📄<br>CAP_NET_RAW 📄            |
| 📄📄📄📄📄<br>📄 UE                                      | N6 📄📄📄📄                              | 📄 UPF 📄📄📄📄📄📄                                   |
| 📄📄📄 Walled<br>garden: raw<br>socket open<br>failed | 📄📄📄📄📄                                | 📄 systemd 📄<br>AmbientCapabilities=CAP_NET_RAW |
| 📄📄📄 UE 📄📄<br>📄📄 IP                                 | N6 📄 NAT 📄📄                          | 📄📄📄 UE IP 📄📄📄 UPF 📄📄                           |

📄 UPF 📄📄 Walled garden forward failed 📄 raw socket open failed 📄📄  
 📄 upf\_walled\_garden\_packets\_forwarded\_total 📄  
 upf\_walled\_garden\_bytes\_forwarded\_total Prometheus 📄📄📄📄📄📄

## Prometheus 📄📄📄📄

📄: 📄📄📄📄📄 dst\_ip 📄 domain 📄📄📄📄📄📄📄📄📄📄📄📄📄📄📄📄📄📄📄📄📄📄📄📄  
 📄📄📄📄

```
# 📄📄📄 /24 📄📄📄📄📄 IP
sum by (dst_subnet) (
  label_replace(
    rate(upf_walled_garden_bytes_forwarded_total[5m]),
    "dst_subnet", "$1.0/24", "dst_ip", "(\\d+\\.\\d+\\.\\d+)\\.\\d+"
  )
)
```

# Web UI 開發

## 目錄

1. 簡介
2. 專案環境建置
3. 專案結構
4. 專案建置
5. 專案測試
6. 專案部署
7. 專案維護
8. 專案更新
9. 專案安全
10. XDP 專案
11. 專案總結

## 簡介

OmniUPF Web UI 專案旨在提供一個基於 Phoenix LiveView 的 Web 管理介面，用於監控和管理 5G Core Network 的 UPF 設備。

- 支援管理 PFCP 連線 PDU 封包
- 支援管理 PDR、FAR、QER 及 URR 規則
- 支援管理 網路切片
- 支援管理 網路切片
- 支援管理 eBPF 規則
- 支援管理 網路切片

## 安裝

專案依賴 REST API 與 OmniUPF 專案進行通訊。

- PFCP
- 
- 
- 
- eBPF

## API

### API

HTTPS OmniUPF

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

443 HTTPS

### API

`config/config.exe` OmniUPF

UPF

`upf_hosts` UI OmniUPF

### API

- `/sessions` - PFCP
- `/rules` - PDR, FAR, QER, URR
- `/buffers` -
- `/statistics` - XDP
- `/capacity` - eBPF
- `/upf_config` - UPF

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

**URL** `/sessions`

OmniUPF `PFCP`

**PFCP**

`PFCP` `SMF/PGW-C`

| <b>ID</b> | SMF <code>PGW-C</code> <code>FQDN</code> <code>IP</code> |
|-----------|----------------------------------------------------------|
|           | SMF/PGW-C <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          |
| SMF 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 のパケットフローを識別するルール     |
| FARs     | パケットフローの転送先を指定するルール     |
| QERs     | パケットフローの QoS を指定するルール   |
| URRs     | パケットフローの帯域幅を制限するルール     |
| その他      | UE の状態や位置に関する情報         |

### 送信

- UE IP アドレスを UE IP とする
- TEID を GTP-U ID とする
- PDR/FAR/QER/URR JSON を送信
- 送信間隔 10 秒

### 受信

受信データ“”を解析

- 受信データ (**PDRs**) から TEID、UE IP、FAR ID、QER ID、SDF を抽出し JSON
  - **PDR ID** が 0 - 有効な PDR を示す
  - PDR の TEID ≠ 0 である PDR
  - PDR の IPv4 アドレスが PDR
  - PDR の IPv6 アドレスが IPv6 の PDR
- 受信データ (**FARs**) を解析

- QoS 策略 (QERs) MBR GBR QFI 等 QoS 策略
- 策略 (URRs) 策略

策略 PDRs FARs QERs

策略

策略 UE 策略

1. 策略
2. 策略 UE IP 策略
3. 策略 TEID 策略
4. 策略 PDR/FAR 策略

策略

- 策略
- 策略 UPF 策略
- 策略



- 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 10000

### IPv6 00 PDRs

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

### FAR 000 - 00000000

0000 FAR 000000000000

000

- 0000000 FAR ID 00000000 FAR 0000
- 0000000 PDR 00000000 FAR ID 00000000
- 0000000FAR 000000000000

| 0             | 00                                           |
|---------------|----------------------------------------------|
| <b>FAR ID</b> | 000000000000                                 |
| 00            | 00000000FORWARD\DROP\BUFFER\DUPLICATE\NOTIFY |
| 00            | 000000000000/0000                            |
| 00            | 000000000000TEID\IP 000                      |

### FAR 000000

- **FORWARD (1)**000000000000

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

packets

- “**Packet**” “**Packet**”
- 
- eBPF packets FAR

## QER - QoS

QoS

- PDR QER ID QER
- PDR QER
- QER 100 1000

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

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 000000000000 URR
- 00000000 PDR 000000000000 URR 000000000000
- 000000000000 URR 00000000 1000000 1000000

| 0             | 00                            |
|---------------|-------------------------------|
| <b>URR ID</b> | 0000000000000000 PDR 00000000 |
| 0000          | 0 UE 0000000000000000         |
| 0000          | 0000000000 UE 000000          |
| 0000          | 0000000000                    |
| 00            | 0000000000 URR 00000000       |

000000

- 00000000B 00KB 00MB 00GB 00TB
- 0000000000000000
- 0000000000

0000

- 0000000000 URR
- 00000000000 0 00000 URR 00000000

□□

□□□□□□□□

1. □□□□□ → PDR □□□
2. □□□□ TEID □ UE IP
3. □□ PDR □□□□ FAR □ QER □□

□□□□□□□□□□

1. □□□□□ → FAR □□□
2. □□□ PDR □□□ FAR ID
3. □□□□□ FORWARD□□□□ DROP □ BUFFER□
4. □□□□□□□□□□

□□ **QoS** □□□

1. □□□□□ → QER □□□
2. □□ MBR □ GBR □□□□□□□
3. □□ 5G □□ QFI □□

□□□□□□□□

1. □□□□□ → URR □□□
2. □□□□◀◀□□□□□□□□
3. □□□□□□□□□□□
4. □□□□□□□

□□□□□□

**URL**□/buffers

□□

□□□□□□□ UPF □□□□□□□□□□□□□□□□□□□□□□□□□□

□□□

□□□□□□□□□□□□□□□□

- □□□□□□ FAR □□□□□□□□
- □□□□□□□□□□□□
- □ **FARs**□□□□□□□□ FAR □□
- □□ **FAR** □□□□□□□□ FAR □□□□□□□□□□
- □□□□□□□□□□□□□□
- □□□ **TTL**□□□□□□□□□□□□□□

□ **FAR** □□□□

□□□□□□□□□□ FAR □□□

| □             | □□               |
|---------------|------------------|
| <b>FAR ID</b> | □□□□□□□□□□       |
| □□□□□□        | □ FAR □□□□□□□□□□ |
| □□□□□□        | □ FAR □□□□□□□□   |
| □□□□□□        | □□□□□□□□□□□□     |
| □□□□□□        | □□□□□□□□□□□□     |
| □□            | □□□□□□□□□□□□     |

□□□□□□□□

□□□□□□□□□□□□ FAR□□□□□□□□□□□□

□□□□□□

- □□□□□□□□□□□□ FAR □□□□□□ FAR □□□□□□
- □□□□□□□□□□□□□□□□ FAR □□□□

□□□□□□

- 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

□□□□

□□□□□ 5 □□□□□□□□□□□□□□□□

□□□□□

**URL** □ /statistics

□□

□□□□□□□ OmniUPF □□□□□□□□□□□□ Prometheus □□□□□□□□□□ □□□□□

□□□□□

□□□□□□□□□□□□

- □□□□□□□□□□□□□□□□□□
- □□□□□□□□□□□□□□□□□□
- □□□□□□□□□□□□□□□□□□
- **GTP-U** □□□□□□ GTP-U □□□□□□□□□□

□□□□□□□ UPF □□□□□□□□□□□□

□□□□□

□□□□□□□□□□□□□□□□

- □□□□□□□□□□□□□□□□□□
- □□□□□□□□□□□□□□□□□□
- □□□□□□□□□□□□□□□□□□

□□□□□□□□□□□□□□□□

**XDP** □□

eXpress Data Path □□□□□

- **XDP** □□□□□ XDP □□□□□□□□□□

- **XDP** 内核态数据包处理
- **XDP** 用户态 XDP 数据包处理
- **XDP** 内核态 XDP 数据包处理

内核态 XDP 数据包处理

### **XDP** 用户态

- 用户态 XDP
- eBPF 用户态
- 用户态 XDP
- 用户态

### **N3/N6** 用户态

用户态 XDP 数据包处理

### **N3** 用户态 RAN 用户态

- 用户 **N3** 用户态 gNB/eNodeB 用户态
- 用户 **N3** 用户态 gNB/eNodeB 用户态

### **N6** 用户态 XDP 数据包处理

- 用户 **N6** 用户态 XDP 数据包处理/IMS 用户态 XDP 数据包处理
- 用户 **N6** 用户态 XDP 数据包处理

用户态 XDP 数据包处理

用户态 XDP 数据包处理

用户

用户态 XDP

1. 用户态 XDP/用户态
2. 用户态 XDP 数据包处理
3. 用户 **N3** 用户 **N6** 用户态 XDP 数据包处理

□□□□□□□□

1. □□□□□□□□□□
2. □□ XDP □□□□□
3. □□□□□□□□□□□□□□

□□□□□

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

□□□□□

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

□□□□□

□□□□□ 5 □□□□□□

□□□□□

**URL**□/capacity

□□

□□□□□□ UPF □□□□□□□□□□ eBPF □□□□□□□□□□

**eBPF** □□□□□

□□ eBPF □□□□□□□□□□□□

| 項目    | 説明                                 |
|-------|------------------------------------|
| 概要    | eBPF を用いて uplink_pdr_map / far_map |
| 目的    | パケットの転送先を動的に変更する                   |
| 対象    | 特定の IP アドレスとポート番号                  |
| 動作    | パケットの転送先を動的に変更する                   |
| メリット  | パケットの転送先を動的に変更できる                  |
| デメリット | パケットの転送先を動的に変更する必要がある              |
| 注意事項  | パケットの転送先を動的に変更する必要がある              |

概要

パケットの転送先を動的に変更する

- 転送先 <50% のパケットを動的に変更する
- 転送先 50-70% のパケットを動的に変更する
- 転送先 70-90% のパケットを動的に変更する
- 転送先 >90% のパケットを動的に変更する

動作

**uplink\_pdr\_map**

- 特定の TEID のパケットを PDR
- 転送先を動的に変更する
- パケットの転送先を動的に変更する

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

- 特定の UE IP のパケットを PDR
- 特定の UE IPv4/IPv6 のパケットを PDR
- パケットの転送先を動的に変更する

## 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. 10,000 - 100,000
3. 100,000 - 1,000,000

## UPF

eBPF UPF UPF

- 10,000 - 100,000
- 100,000 - 1,000,000
- 1,000,000+

## UPF

$$\text{UPF} = (\text{PDR} + \text{PDR}) \times \text{PDR}$$

UPF 100 64 PDR 64 MB

## UPF

UPF 10

## UPF

URL /upf\_config

## UPF

UPF

## UPF

UPF

- **PFCP** SMF/PGW-C IP
- **N3** RAN/gNB/eNodeB IP
- **N6** IP

- **N9** 通过UPF 连接到 IP 网络
- **API** 通过REST API 连接
- 通过OmniUPF 连接

### 通过eBPF

通过eBPF

- 通过 **N3** 连接到 N3 网络
- 通过 **N9** 连接到 N9 网络

通过eBPF 连接到 eBPF 网络

通过

通过 **UPF** 连接

1. 通过 N3 通过 IP 连接到 gNB 网络
2. 通过 N6 连接到网络
3. 通过 PCFP 连接到 SMF 网络

通过

1. 通过网络
2. 通过网络
3. 通过网络

通过

1. 通过 UPF 连接到网络
2. 通过网络
3. 通过网络

通过

**URL** `/routes`

□□

□□□□□□□□□□□□□□□□ UE IP □□□□□□□□□□□□ OSPF □ BGP□□□□□□□□

□□□□□□

□□□□□□□□□□□□□□□□

- □□□□□□□□□□□□
- □□□□□ UE IP □□□□□□
- □□□□□□□□□□□□□□□□
- □□□□□□□□□□□□□□□□

□□ **UE IP** □□

□□□□□□□□□□□□ IP □□□□□□

| □               | □□                       |
|-----------------|--------------------------|
| □□              | □□□□□□                   |
| <b>UE IP</b> □□ | □□□□ UE □ IPv4 □ IPv6 □□ |

□□□

- □□□□□□□□□□□□ UE IP □□
- □□□□□□□□□□□□□□□□
- □□□□□□□□□□

**OSPF** □□

OSPF□□□□□□□□□□□□□□□□□□□□□□□□□□□□

| 項目      | 内容      |
|---------|---------|
| OSPF ID | OSPF ID |
| OSPF IP | OSPF IP |
| OSPF    | OSPF    |
| OSPF    | OSPF    |
| OSPF    | OSPF    |
| OSPF    | OSPF    |
| OSPF    | OSPF    |
| OSPF    | OSPF    |

**OSPF**

- 
- 

**BGP**

BGP

| 項目    | 説明        |
|-------|-----------|
| IP    | BGP 宛先 IP |
| ASN   | 自治システム番号  |
| 経路    | BGP 経路    |
| 優先度   | 優先度       |
| メトリック | メトリック     |
| 状態    | 状態        |
| 宛先    | 宛先 BGP    |
| 経路    | 経路 BGP    |

### BGP 経路

- 経路 BGP 宛先
- 経路 BGP 宛先

経路 BGP ID 宛 ASN 宛 BGP 宛

### OSPF 経路

経路 UE 宛 OSPF 宛 LSA 宛

| 項目     | 説明                |
|--------|-------------------|
| リンク ID | LSA のリンク ID       |
| リンク    | リンク ID            |
| リンク ID | リンク ID            |
| リンク    | OSPF のリンク E1 と E2 |
| リンク    | リンク OSPF のリンク     |
| リンク    | LSA のリンク ID       |
| リンク    | LSA のリンク ID       |

### リンク

- リンク UE のリンク ID OSPF
- リンク ID
- リンク LSA のリンク ID

### リンク ID

#### リンク ID

- リンク ID FRR のリンク ID
- リンク ID UE のリンク ID
- リンク ID

#### リンク ID

- リンク ID
- リンク OSPF のリンク BGP のリンク ID

□□

□□□□□□□□

1. □□□□□□
2. □□ OSPF □□□□□□“□□”□
3. □□ BGP □□□□“□□□□”
4. □□□□□□□□/□□□□□□

□□ **UE** □□□□□□

1. □□□□ UE IP □□□□□□□□ UE
2. □□□ OSPF □□□□□□□□
3. □□ UE □□□□□□□□□□ LSA □
4. □□□□□□□□□□□□ UPF □□

□□□□□□□□□□□□

1. □□□□□□□□□□□□□□□□□□
2. □□□□□□□□□□□□□□□□□□
3. □□□□□□□□□□□□□□□□□□
4. □□□□□□□□□□ OSPF/BGP □□□□□□

□□□ **UPF** □□□□

1. □□□□□□□□□□□□□□□□□□ UPF □□
2. □□□□□□□□□□□□□□□□□□
3. □□ OSPF □□□□□□□□□□
4. □□ BGP □□□□□□□□

□□□□□□□□

1. □□ UE □□□□□□□□□□□□□□
2. □□□□□□□□□□□□□□□□□□
3. □□ OSPF LSA □□□□□□□□
4. □□ BGP □□□□□□□□□□

□□□□

□□□□ 10 □□□□□□□□□□□□□□□□□□ UE □□□

□□□□

□□□□□ UPF □□□□ FRR□□□□□□□□□□□□

- **OSPF**□□□□□□□□ 2 LSA □□□□
- **BGP**□□□□□□□□□□ BGP □□□
- □□□□□ REST API □□□□ vtysh □□□□□ FRR

# XDP □□□□

URL □ /xdp\_capabilities

□□

XDP □□□□□ eXpress Data Path (XDP) □□□□□□□□□□ UPF □□□□□□□□□□□□

□□□□

□□□□□□□□□□□□□□□□□□

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

## XDP

XDP

### XDP\_DRV

- ~5-10 Mpps
- XDP
- XDP NIC i40e ixgbe mlx5
- XDP
- X

### XDP\_SKB

- ~1-2 Mpps
- 
- 
- 
- 

- XDP
- 

- “”
- XDP

*XDP*  *Mpps*

- "✓  XDP\_DRV
- 

- "△  XDP\_DRV
- 
- "△  XDP\_DRV"
- 

- XDP

# Mpps 計算

計算 Mpps から Gbps

計算

計算 Mpps

- 0.1 - 100 Mpps
- XDP Mpps
- 

計算

- 64 - 9000
- 1200 GTP
- GTP

計算

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

計算

計算 Gbps

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

計算 Gbps

-

- 1000 ~50 1000 GTP 10000
- 1000 Gbps = Mpps × (Packet\_Size - 50) / 1000

1000000

- 10 Mpps 1000000000000000
- 10000 Mpps = 10,000,000 100000

100000

- 1000000000
- 1000 10 Mpps × 1200 100 × 8 1000 ÷ 1000 = 96 Gbps

## 10 Mpps

1000000000000000

1000 Mpps

- 1000000000
- 1000000000000000
- 1000000000

1000000000

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

GTP 100000

- 100000014 100
- IP 10020 1000IPv4100 40 1000IPv6100
- UDP 1008 100
- GTP 1008 100000000
- 1000000000000000 ~50 100

□□

□□ **XDP** □□□

1. □□□ XDP □□□□
2. □□□□ XDP □□□□□ DRV □□□□□□□□
3. □□ Mpps □□□□
4. □□□□□□

□□□□□□□□

1. □□□□□□□□□□□□ Mpps□
2. □□□□□□□□□□□□□□□□
3. □□□□□□□□□□ Gbps□
4. □□□□□□□□□□□□□□

□□ **XDP** □□□

1. □□□□□□□□□□ XDP\_DRV □□
2. □□□□□□□□□□□□
3. □□□□□□□□□□□□□□□□□□
4. □□□□□□□□□□ CPU □□□□

□□□□□

1. □□□□□□□□□□□□□□□□ Mpps
2. □□□ XDP □□□□□□□□
3. □□□□□□□□□□
4. □□□□□□□□□□□□□□□□

□□□□□□□□

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

## □□□□□□

### □□□□□ XDP\_DRV □□

- □□□□□ XDP □□□ NIC □ Intel i40e/ixgbe □ Mellanox mlx5 □
- □□ NIC □□□□□□□□□□
- □□□□□□ RSS □□□□□□□□□□
- □□ NIC □□□□□□□□

### □□□□□ XDP\_SKB □□

- □□□□□□□□
- □□□□□□□□□□□□
- □□□□□□□□□□□□□□

### □□□□□□

- □□□□□□ CPU □□□□□□□□□□
- □□□□□□□□□□□□□□
- □□ RSS □□□□□□□□□□□□□□

## □□□□

XDP □□□□□ 30 □□□□□□□□□□□□□□□□□□□□

## □□□□□□

URL □ [/logs](#)

## □□

□□□□ OmniUPF □□□□□□□□

## □□□

- □□ Phoenix LiveView □□□□□□□□
- □□□□□□□□□□

- 00000000
- 00000000000000

0000

OmniUPF 000000 Elixir Logger 000

- **DEBUG**00000000
- **INFO**000000000000
- **WARNING**000000000000
- **ERROR**00000000

00

0000000000

1. 000000
2. 0 SMF 000000
3. 00 PFCP 0000000000

00 **PFCP** 000

1. 00 PFCP 000000
2. 000000/00/00
3. 000000

00000000

1. 0000000000
2. 00 eBPF 000000
3. 00 FAR/PDR 0000

# □□□□

## □□□□

### □□□

- □□□□□□□□□□□□□□
- □□□□□□□□□□□□□□
- □□□□□□□□□□
- □□ XDP □□□□

### □□□□□□

- □□□□□□□□□□
- □□□□□□□ TTL□□□□□□□□□□
- □□□□□□□□□□
- □□“□□”□□□□“□□”□□□□□□□□

### □□□□□

- □□□□□□□□□□ UE □□
- □□□□□□□□□□
- □□□ UPF □□□□□□□□
- □□□□□□□□□□□□□□

### □□□□□

- □□□□□□□□□□
- □□□□□□□□□□ UE □□□□
- □□□□□□□□□□□□
- □□□□□□□□□□□□□□□□□□

### □□

- □□□□□□□□□□□ 5-10 □□□□□□□□□□
- □□□□□□□□□□□□□□

- 5G Core Network (5GC) URR (Usage Reporting Rule) 5GC Core Network
- 5G Core Network UPF (User Plane Function) 5GC Core Network

## 5G Core Network

- **5GC Core Network** - PDR (Policy Data Rule) FAR (Forwarding Action Rule) QER (QoS Enforcement Rule) URR (Usage Reporting Rule) 5GC Core Network
- **5GC Core Network** - 5GC Core Network
- **5GC Core Network** - Prometheus 5GC Core Network
- **PFCP** (Policy and Charging Control Function) - PFCP 5GC Core Network
- **API** (Application Programming Interface) - REST API 5GC Core Network
- **5GC Core Network** - UE (User Equipment) FRR (Fast Reroute) 5GC Core Network
- **XDP** (eXpress Data Path) - XDP 5GC Core Network eBPF (extended Berkeley Packet Filter) 5GC Core Network
- **5GC Core Network** - 5GC Core Network
- **UPF** (User Plane Function) - UPF 5GC Core Network

# OmniUPF 与 XDP 教程

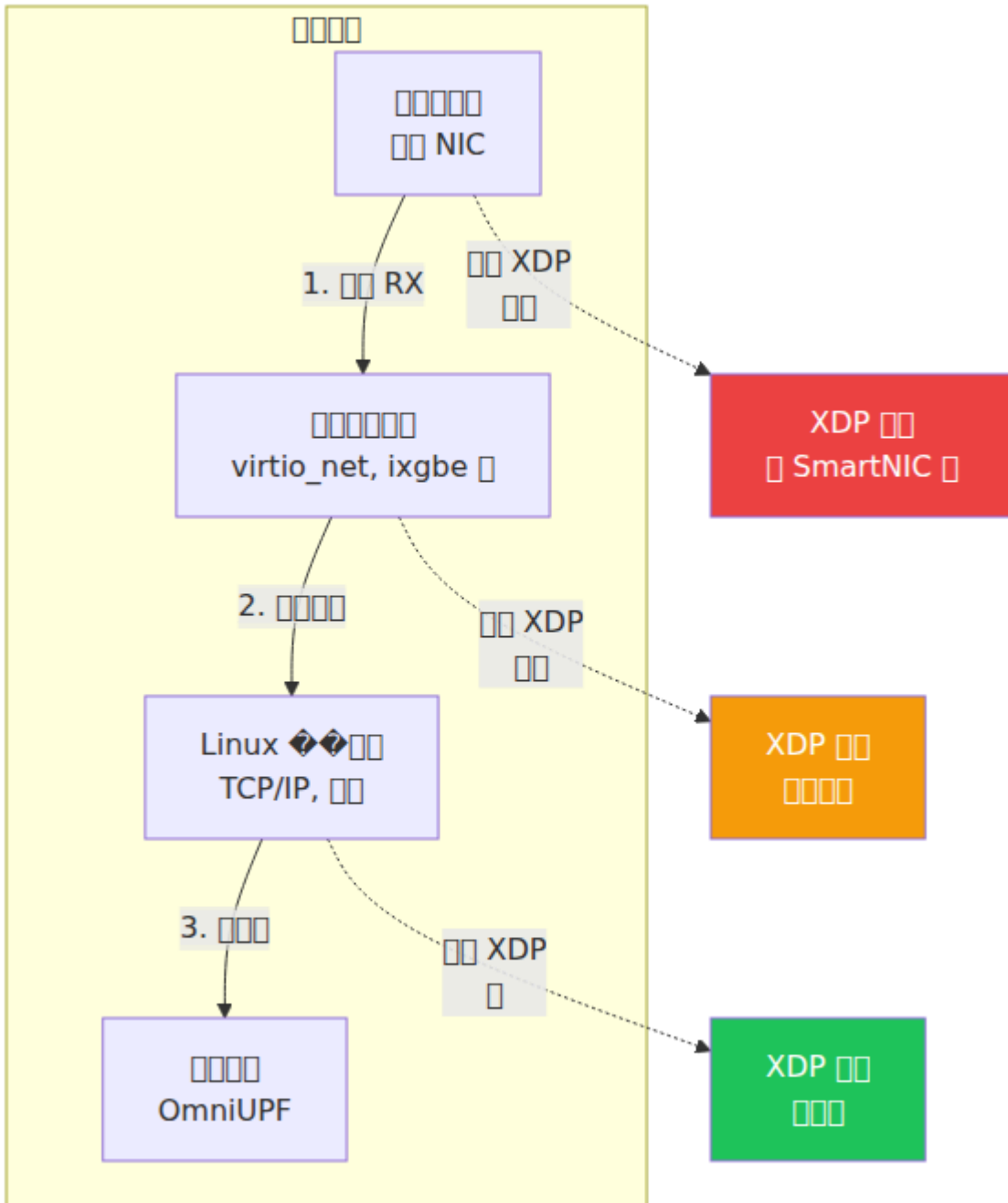
## 目录

1. 简介
2. XDP 概述
3. 网络架构
4. 配置环境
5. 安装 SmartNIC
6. Proxmox VE 部署 XDP
7. 配置 XDP 程序
8. XDP 性能测试
9. XDP 进阶

## 简介

OmniUPF 与 **XDP (eXpress Data Path)** 是 Linux 内核中用于高性能网络数据包处理的新技术。XDP 允许在网卡驱动层直接处理数据包，绕过传统的内核网络栈，从而显著降低延迟并提高吞吐量。eBPF 则提供了一种安全且灵活的方式来编写和部署 XDP 程序。

XDP 程序通过 **eBPF** 技术进行部署和管理。



XDP OmniUPF

---

# XDP 对比

| 对比项    | Linux 通用                              | 通用                                   | 专用                                    |
|--------|---------------------------------------|--------------------------------------|---------------------------------------|
| 平台     | Linux 通用                              | 通用                                   | NIC 专用                                |
| 性能     | ~1-2 Mpps                             | ~5-10 Mpps                           | ~10-40 Mpps                           |
| 延迟     | ~100 $\mu$ s                          | ~10 $\mu$ s                          | ~1 $\mu$ s                            |
| CPU 消耗 | 高                                     | 中                                    | 低                                     |
| NIC 支持 | 普通 NIC                                | 支持 XDP 的 NIC                         | 支持 XDP 的 SmartNIC                     |
| 部署位置   | 操作系统内核                                | 操作系统内核                               | 网卡 PCI 设备                             |
| 配置     | 简单                                    | 简单                                   | 简单                                    |
| 配置项    | <code>xdp_attach_mode: generic</code> | <code>xdp_attach_mode: native</code> | <code>xdp_attach_mode: offload</code> |

对比项 对比项

对比项

对比项

对比 XDP 对比项 对比 Linux 通用 eBPF 对比 XDP 对比项

## 概要

- 1000~1-2 Gbps (Mpps)
- 1000000 ~100 個
- **CPU** を使わず XDP を使います

## 前提

- Linux 5.10 以上
- 10Gbps ネットワークカード
- 100MB 以上のメモリ

## 設定

```
# config.yaml
interface_name: [eth0]
xdp_attach_mode: generic # 10000
```

10Gbps ネットワークカードを接続し、CPU を使わず

---

## 性能比較

### 概要

XDP を使わずに eBPF を使った Linux ネットワークスタックの性能比較

## 結果

- 1000000 ~5-10 Gbps (Mpps)
- 1000000 ~10 個
- **CPU** を使わず
- ネットワークカード CPU を使わず NIC を使います

## 网卡

- 网卡驱动
- 网卡固件
- 网卡配置
- 网卡性能

## NIC 网卡

网卡 XDP 驱动 XDP 驱动程序网卡 XDP

### 网卡 NIC 驱动

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

### 网卡 NIC 虚拟化

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

VirtualBox 网卡 XDP 驱动程序

## 配置

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

网卡驱动程序网卡 Proxmox 网卡

---

# SmartNIC

概要

XDP NIC SmartNIC eBPF CPU

特徴

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

適用

- UPF 10G+
- 
- CPU

製品

Netronome Agilio SmartNIC XDP

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

PCI -

設定

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

---

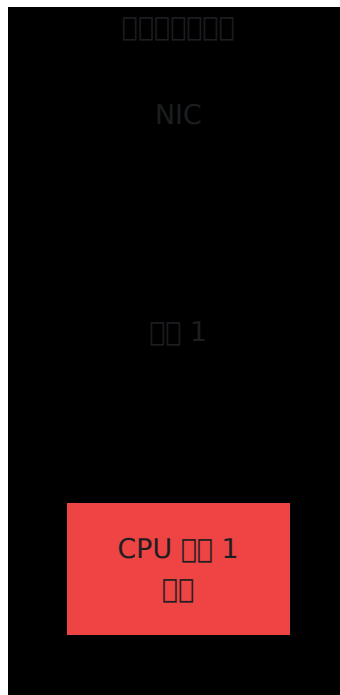
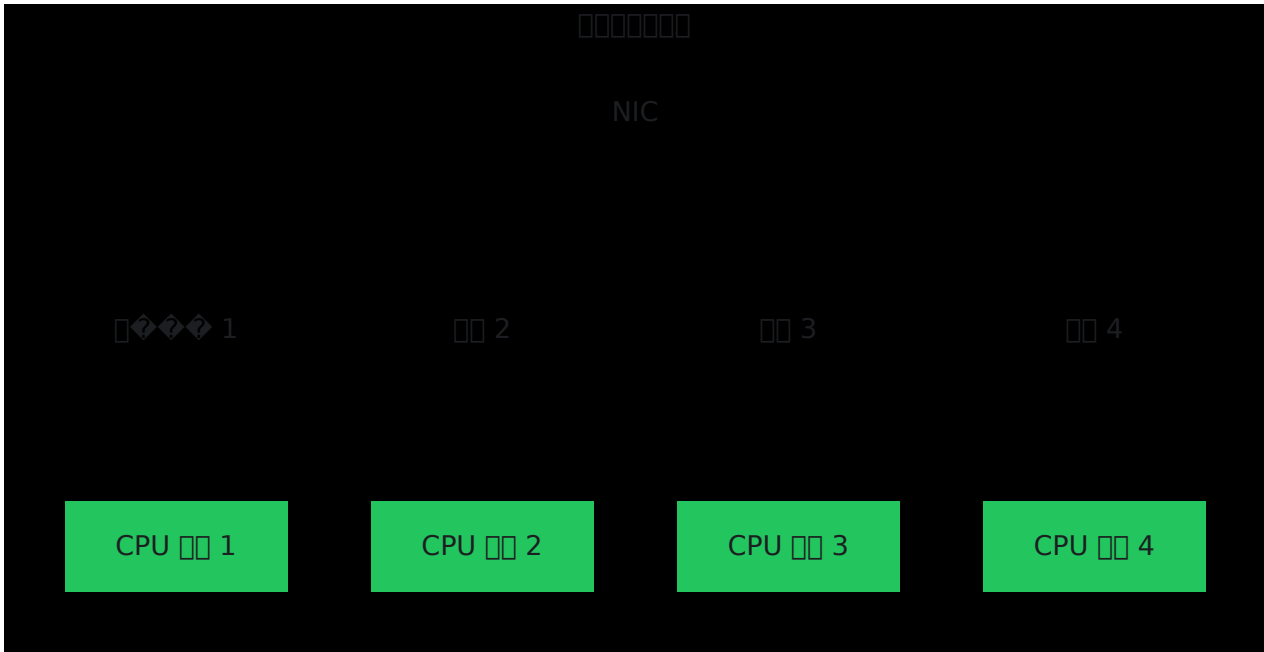
# Proxmox VE 网络 XDP

Proxmox VE 使用 **VirtIO** 网络接口 `virtio_net` 支持 XDP 功能

使用 **1** 个 CPU

配置

- 使用 CPU 0 → 1
- 使用 CPU 1 → 0



## 2. Proxmox

### Proxmox Web UI

1. Proxmox Web UI
  - Proxmox Web UI
  - Proxmox Web UI

## 2. 設定

- 設定 設定 設定
- 設定設定設定設定 net0
- 設定 設定

## 3. 設定

- 設定 "設定" 設定
- 設定 8 設定 vCPU 設定 16
- 設定 設定

## 4. 設定

- 設定 設定

## 設定 Proxmox 設定

```
# SSH 設定 Proxmox 設定

# 設定 ID
qm list

# 設定 XXX 設定 ID
qm set XXX -net0 virtio=XX:XX:XX:XX:XX:XX,bridge=vmbr0,queues=8

# 設定 191 MAC 設定 BC:24:11:1D:BA:00
qm set 191 -net0 virtio=BC:24:11:1D:BA:00,bridge=vmbr0,queues=8

# 設定
qm shutdown XXX

# 設定
qm start XXX
```

## 設定

- 4 設定 2-4 vCPU 設定
- 8 設定 4-8 vCPU 設定
- 16 設定 8+ vCPU 設定

## 3. 8個のキューを設定する

SSHで接続して確認する

```
# 確認
ethtool -l eth0

# 確認
# eth0 の設定
# Combined:      8          <-- 8個のキュー

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

# 確認 8個のキュー
```

## 4. OmniUPF を XDP に変更する

OmniUPF を XDP に変更する

```
# 確認
sudo nano /config.yaml
```

XDP に変更する

```
# XDP
xdp_attach_mode: generic

# XDP
xdp_attach_mode: native
```

OmniUPF を再起動する

```
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`

XXXXXXXXXXXXXXXXXXXX

XXXXXX

- CPU XXXXXXXXXXXXXXX
  - `top` - CPU XXXXXXXXXXXXXXX
  - XDP XXXXXXX `curl http://localhost:8080/api/v1/xdp_stats`
- 

## XXXXXXXXXXXXXXXXXXXX **XDP**

### VMware ESXi / vSphere

VMware `vmxnet3` XXXXXXXXXXXXXXX XDP

XXXX

- ESXi 6.7 XXXXXXX
- `vmxnet3` XXXXXXX 1.4.16 XXX
- XXXXXXX 14 XXX

XXXXXXXX

1. XXXXXXX
2. XXXXXXXXXXXXXXX
  - XXXXXXXXXXXXXXX → XXXXXXX
  - XXXXXXX → XXX
  - `vmxnet3` XXXXXXX XXX
3. `.vmx` XXXXXXXXXXXXXXXXXXXXXXX

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

4. XXXXXXXXXXXXXXX

```
ethtool -l ens192 # 00000000
```

## OmniUPF

```
interface_name: [ens192] # VMware 0000 ens192  
xdp_attach_mode: native
```

## KVM / libvirt

### virsh

```
# 00000000  
virsh edit your-vm-name
```

### 000000000000

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

### 000000000000

```
ethtool -l eth0
```

## Microsoft Hyper-V

Hyper-V 00 hv\_netvsc 0000000000 XDP

### 000

- Windows Server 2016 000000
- 0000 Linux 0000 4.3 0000

- `netsh`

`netsh`

`netsh Hyper-V network interface PowerShell`

```
# netsh VMQ interface - Hyper-V netsh  
Set-VMNetworkAdapter -VMName "YourVM" -VrssEnabled $true -  
VmmqEnabled $true
```

`netsh OmniUPF`

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

## VirtualBox

`netsh VirtualBox interface XDP`

`netsh VirtualBox interface e1000 virtio-net interface XDP netsh`

`netsh interface xdp`

```
xdp_attach_mode: generic # VirtualBox netsh
```

---

## netsh XDP netsh

`netsh XDP interface netsh`

## 1. 检查 OmniUPF 配置

```
# 检查配置
journalctl -u omniupf --since "5 minutes ago" | grep -i xdp

# 输出
# ✓ "xdp_attach_mode:native"
# ✓ "启用 XDP 支持"
# ✗ "配置" 与 "配置"
```

## 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,          # 0 个
  "xdp_drop": 1234,         # 1234 个
  "xdp_pass": 5678,         # 5678 个
  "xdp_redirect": 9012,    # 9012 个
  "xdp_tx": 3456           # 3456 个
}
```

## 4. 网卡驱动

```
# 查看网卡驱动 XDP
ethtool -i eth0 | grep driver

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

## 5. 网络性能

网络性能测试

```
# 网络性能测试
watch -n 1 'curl -s http://localhost:8080/api/v1/packet_stats | jq
.rx_packets'

# 性能 ~1-2 Mpps
# 性能 ~5-10 Mpps 5-10 秒
```

---

## 网卡 XDP 配置

配置网卡 "网卡 XDP 配置"

配置

```
配置 XDP 配置 eth0
```

配置

1. 配置

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

```
# virtio_net/vmxnet3/hv_netvsc XDP
```

## 2. 检查内核版本

```
uname -r
```

```
# 内核版本 ≥ 5.15 支持 XDP
```

## 3. 检查并禁用 XDP 驱动

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

```
# 检查 XDP 是否启用
```

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

检查

- 内核版本 ≥ 5.15+
- 检查 virtio\_net 驱动 `modprobe virtio_net`
- 检查 XDP 是否启用

检查 XDP 是否启用

检查

```
检查 XDP 是否
```

检查

```
使用 dmesg 检查
```

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

□□□□

### 1. □□□□□□□□ XDP

- VirtualBox □□□□□□□□ XDP
- □□□ NIC □□□□

### 2. □□□□□□

- □□□ `ethtool -l eth0`
- □□□ > 1 □□□□

### 3. □□ XDP □□□□□□

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

# □□□□
# CONFIG_XDP_SOCKETS=y
# CONFIG_BPF=y
```

□□□□

- □□□□□□□□ Proxmox □□□
- □□□□□□□□
- □□□□□□□□□□□□ XDP

□□□□□□□□□□□□□□□□

□□□□□□□□□□□□□□□□□□

□□□

### 1. □□□□□□□□

```
# 检查网络接口
ethtool -S eth0 | grep rx_queue

# 检查网络接口
```

## 2. 检查 CPU 使用率

```
# 检查 CPU 使用率
mpstat -P ALL 1

# 检查 CPU 使用率
```

## 3. 检查 XDP 配置

```
# 检查 XDP 配置
sudo bpftool net list

# 检查 XDP 配置
```

检查

- 检查 CPU 使用率 8-16 秒
- 检查 CPU 使用率
- 检查 CPU 使用率

---

检查 XDP 配置 `xdp_aborted > 0`

检查

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

检查

## XDP 与 eBPF 入门

### 1. 检查 eBPF 是否安装

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

### 2. 安装 eBPF 工具

```
# 安装 eBPF 工具  
curl http://localhost:8080/api/v1/map_info  
  
# 安装成功 100% 完成
```

## 总结

- 了解 eBPF 是什么
- 了解 eBPF 的用途
- 了解 Linux 内核 eBPF 子系统

---

## 在 Proxmox 中配置 eBPF

使用 `ethtool -l eth0` 查看网卡 1 的队列配置

## 配置

### 1. 在 Proxmox 中配置

```
# 在 Proxmox 中配置  
grep net0 /etc/pve/qemu-server/YOUR_VM_ID.conf  
  
# 配置 queues=8
```

### 2. 验证配置

```
# Proxmox
qm status YOUR_VM_ID

# Check "status: stopped"

```

Check

```
# Proxmox
# Shutdown VM
qm shutdown YOUR_VM_ID
sleep 10
qm start YOUR_VM_ID

# Check network
ethtool -l eth0

```

Check network status

## Check XDP

Check

```
Check XDP

```

Check

XDP requires `CAP_NET_ADMIN` and `CAP_SYS_ADMIN`

Check

1. `root` restart **OmniUPF**

```
sudo systemctl restart omniupf

```

2. Check **systemd**

```
# /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 실행 --privileged 실행

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

## 테스트 결과

OmniUPF 테스트 결과

| 구분               | 일반         | 개선         | 비교      |
|------------------|------------|------------|---------|
| 처리량              | 1.5 Mpps   | 8.2 Mpps   | 약 5.5 배 |
| 지연               | 95 μs      | 12 μs      | 약 8 배   |
| CPU 사용량 (1 Gbps) | 85% (1 코어) | 15% (4 코어) | 약 5 배   |
| 처리 용량            | ~1.2 Gbps  | ~10 Gbps   | 약 8 배   |

이러한 성능 향상을 위해서는 XDP를 활용해야 합니다.

## XDP 테스트

△ 테스트 결과: Omnitouch 테스트 결과 100% 달성

### 테스트 방법: XDP vs NIC

NIC vs OmniUPF vs XDP 비교

Intel NIC

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

Mellanox/NVIDIA NIC

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

Broadcom NIC

| Model           | Speed       | Driver  | XDP Support | Notes                      |
|-----------------|-------------|---------|-------------|----------------------------|
| <b>BCM57xxx</b> | 10/25/50GbE | bnxt_en | Yes ✓       | Supports SR-IOV on Dell/HP |

Other NIC

| OS                 | NIC Driver | Kernel Driver | XDP Support | Kernel           | Notes                |
|--------------------|------------|---------------|-------------|------------------|----------------------|
| <b>Proxmox/KVM</b> | VirtIO     | virtio_net    | Yes ✓       | Linux<br>Windows | Standard             |
| <b>VMware ESXi</b> | vmxnet3    | vmxnet3       | Yes ✓       | Linux            | ESXi 6.7+            |
| <b>Hyper-V</b>     | NIC        | hv_netvsc     | Yes ✓       | Linux            | Windows Server 2016+ |
| <b>AWS</b>         | ENA        | ena           | Yes ✓       | Linux            | EC2 Standard         |
| <b>VirtualBox</b>  | NIC        | NIC           | Yes         | Linux            | Standard             |

## NIC

XDP eBPF NIC

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

NIC XDP

OmniUPF

1-10 Gbps

- **NIC** Intel X520 10GbE 4000
- 支持 XDP
- 支持 UPF 约 ~8-10 Gbps
- 约 \$100-200/个

### 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 个
- 支持 XDP
- 支持 UPF 约 ~5-10 Gbps
- 支持

### 

支持 OmniUPF 个

| NIC/OS       | OS        | NIC            |
|--------------|-----------|----------------|
| Realtek NICs | Linux XDP | Intel i350     |
| VirtualBox   | XDP       | Proxmox/KVM    |
| NICs         |           | Intel/Mellanox |
| NICs < 2014  | XDP       | Intel X520     |

## Checklist

### Kernel

- Kernel Linux XDP

```
# modinfo <driver_name> | grep -i xdp
```

- Kernel  $\geq 5.15$  XDP

```
uname -r
```

- NIC RSS/VMDq

- PCI PCIe

- 10GbE PCIe 2.0 x4
- 40GbE PCIe 3.0 x8
- 100GbE PCIe 3.0 x16 PCIe 4.0 x8

- OS

- NIC
- VirtIO SR-IOV
- NIC

## 📄

- [CONFIGURATION.md](#) - [CONFIGURATION](#)
  - [TROUBLESHOOTING.md](#) - [TROUBLESHOOTING](#)
  - [ARCHITECTURE.md](#) - eBPF & XDP [ARCHITECTURE](#)
  - [MONITORING.md](#) - [MONITORING](#)
- 

## 📄

### Proxmox [XDP](#) [TL;DR](#)

```
# 📄 Proxmox XDP
qm set <VM_ID> -net0 virtio=<MAC>,bridge=vbr0,queues=8
qm shutdown <VM_ID> && sleep 10 && qm start <VM_ID>

# 📄 XDP
ethtool -l eth0 # 📄 8 XDP
sudo nano /etc/omniupf/config.yaml # 📄 xdp_attach_mode: native
sudo systemctl restart omniupf
journalctl -u omniupf --since "1 min ago" | grep xdp # 📄 XDP
```

### [XDP](#) [XDP](#)

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

# 📄 XDP
curl -s http://localhost:8080/api/v1/xdp_stats | jq

# 📄 XDP
ethtool -l eth0
```

# OmniUPF API

OmniUPF API RESTful eBPF API UPF

## API

- **PFCP** UE IP TEID
- **PFCP**

- **PDR** (IPv4/IPv6)
- **FAR**
- **QoS** (**QER**) QoS
- **URR**

- FAR (GET /buffer, GET /buffer/:far\_id)
- (POST /buffer/:far\_id/flush, DELETE /buffer/:far\_id, DELETE /buffer)
- (POST /buffer/:far\_id/notify)
- DLDR (GET /buffer/notifications)

- (GTP, IP, TCP, UDP, ICMP, ARP)
- **XDP**
- **N3/N6** RAN
- **FIB**

API

- **UE** API UE IP gNB API (GET /routes)
- **FRR** API Free Range Routing API (POST /routes/sync)
- API (GET /routing/sessions)
- **OSPF** API OSPF API (GET /ospf/database/external)

API

- **UPF** API (GET /config, POST /config)
- API (GET /dataplane\_config)
- **XDP** API XDP API (GET /xdp\_capabilities)
- **eBPF** API (GET /map\_info)

## Web UI

OmniUPF Web UI API API Web UI API

## Swagger API

API **OpenAPI 3.0 (Swagger)** Swagger UI

- API
- API
- API
- HTTP

Swagger UI 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
```



```

    &#123;
    "data": [
        &#123; /* session object */ &#125;,
        &#123; /* session object */ &#125;,
        ...
    ],
    "pagination": &#123;
        "total": 5432,
        "page": 2,
        "page_size": 50,
        "total_pages": 109
    &#125;
&#125;

```

## 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 PDR mapping
- `/api/v1/downlink_pdr_map` - Downlink PDR IPv4 mapping
- `/api/v1/downlink_pdr_map/full` - Downlink SDF PDR IPv4 mapping
- `/api/v1/downlink_pdr_map_ip6` - Downlink PDR IPv6 mapping
- `/api/v1/downlink_pdr_map_ip6/full` - Downlink SDF PDR IPv6 mapping
- `/api/v1/far_map` - FAR mapping
- `/api/v1/qer_map` - QoS mapping
- `/api/v1/urr_map` - URRC 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` - 清除缓冲区
- `POST /api/v1/buffer/:far_id/notify` - 通知 DLDR 更新

配置

- `GET /api/v1/config` - 获取 UPF 配置
- `POST /api/v1/config` - 更新 UPF 配置
- `GET /api/v1/dataplane_config` - 获取数据面配置

路由

- `GET /api/v1/routes` - 获取 UE 路由
- `POST /api/v1/routes/sync` - 同步 FRR 路由
- `GET /api/v1/routing/sessions` - 获取路由会话
- `GET /api/v1/ospf/database/external` - 获取 OSPF 外部 LSA

分页

- 通过 Web UI 设置 `page_size=100`
- 通过 API 设置 `page_size=1000`
- 通过 `pagination.total_pages` 获取总页数
- 通过 `page_size` 设置 API 返回的每页条数

## CORS 配置

配置 CORS 以允许 API 和 Web UI 访问 API

## Prometheus 配置

REST API 和 OmniUPF 通过 `/metrics` 暴露 Prometheus 指标

配置

- 配置 PFCP 计数器
- 配置 DLDR 计数器
- XDP 计数器

- 0000
- eBPF 00000000
- URR 0000

000 0000 00000000

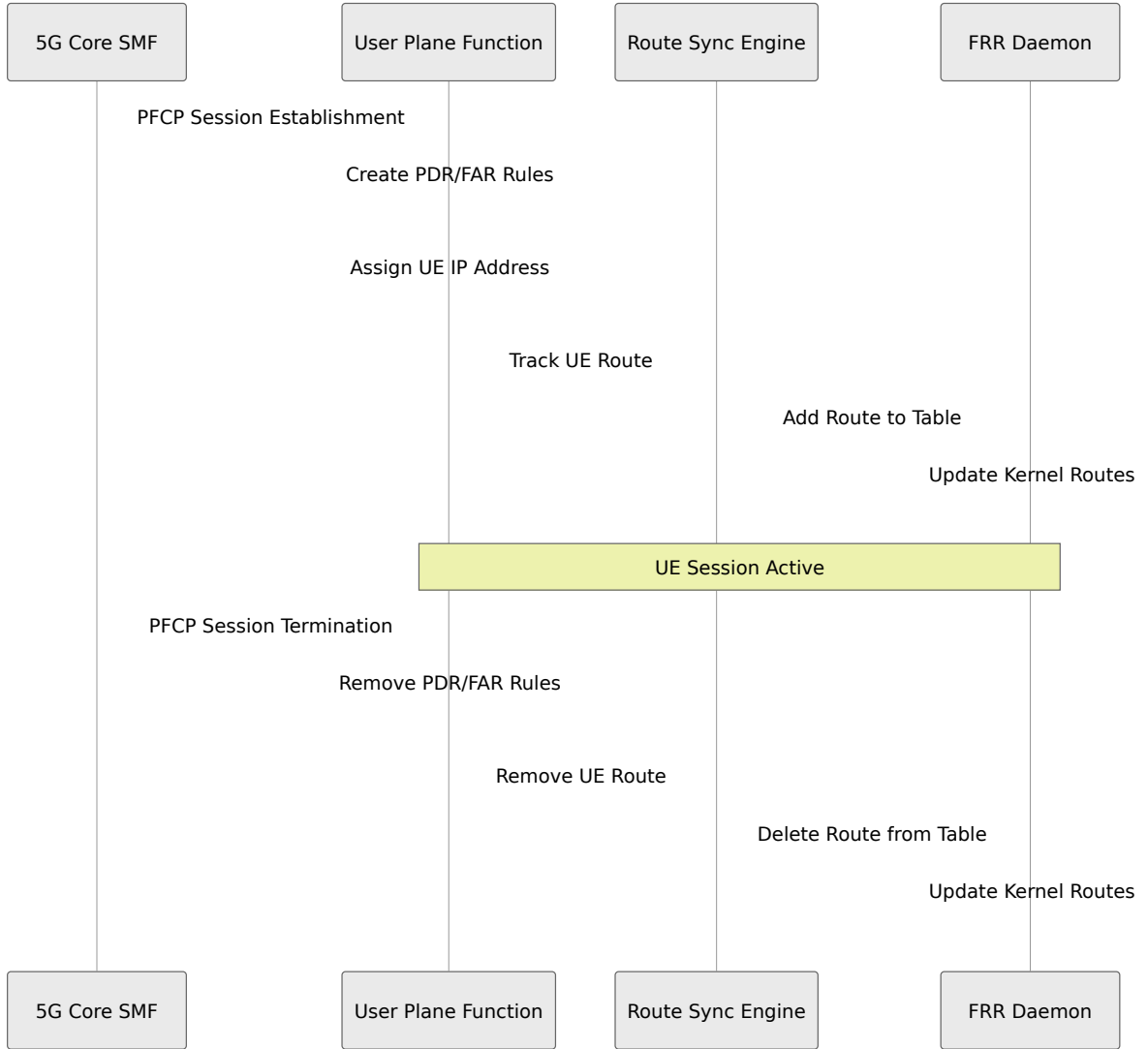
0000

- **Web UI** 00 - 000 API 0000000000
- 0000 - Prometheus 0000
- **PFCP** 0000 - PFCP 0000000000
- 000000 - PDR\FAR\QER\URR 00
- 000000 - FRR 000 UE 00
- 0000 - 0000000000
- 0000 - UPF 0000
- **Swagger UI** - 000 API 0000 localhost 00000 UPF 000



□□□□□□□□

□□□□□□



□□□□

UPF □□□□□□ UE IP □□□□□□□□□□□□□□□□□□□□

1. □□ **UE** □□□□□□□□□□ PFCP □□□□□□□□ UE IP □□
2. □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
3. □□□ **FRR**□□□□□□ API □□□□□□□□□□□□ FRR □□□□□
4. □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

# FRR 配置

配置

FRR 配置 **Ansible** 配置 配置 Ansible 配置 FRR 配置 **Jinja2** 配置 Ansible 配置 UPF 配置

配置 FRR Jinja2 配置

```
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
```

配置

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

4. Ansible 透過 IP 地址 ID 透過 UPF 與 FRR 連接

### Jinja2 透過 Ansible

- **OSPF** 透過 ID 透過 Ansible 配置
- **BGP** 透過 ASN 透過 Ansible 配置
- 透過 Ansible 配置 `redistribute static` 與 UE 連接
- 透過 Ansible 配置 `redistribute static` 與 UE 連接
- 透過 Ansible 配置 OSPF/BGP 透過 Ansible

**UPF** 透過 FRR 透過 Ansible 配置 UPF 透過 PFCP 透過 FRR vtysh 透過 UE IP 透過 IPv4 或 /32 IPv6 或 /128 透過 Ansible

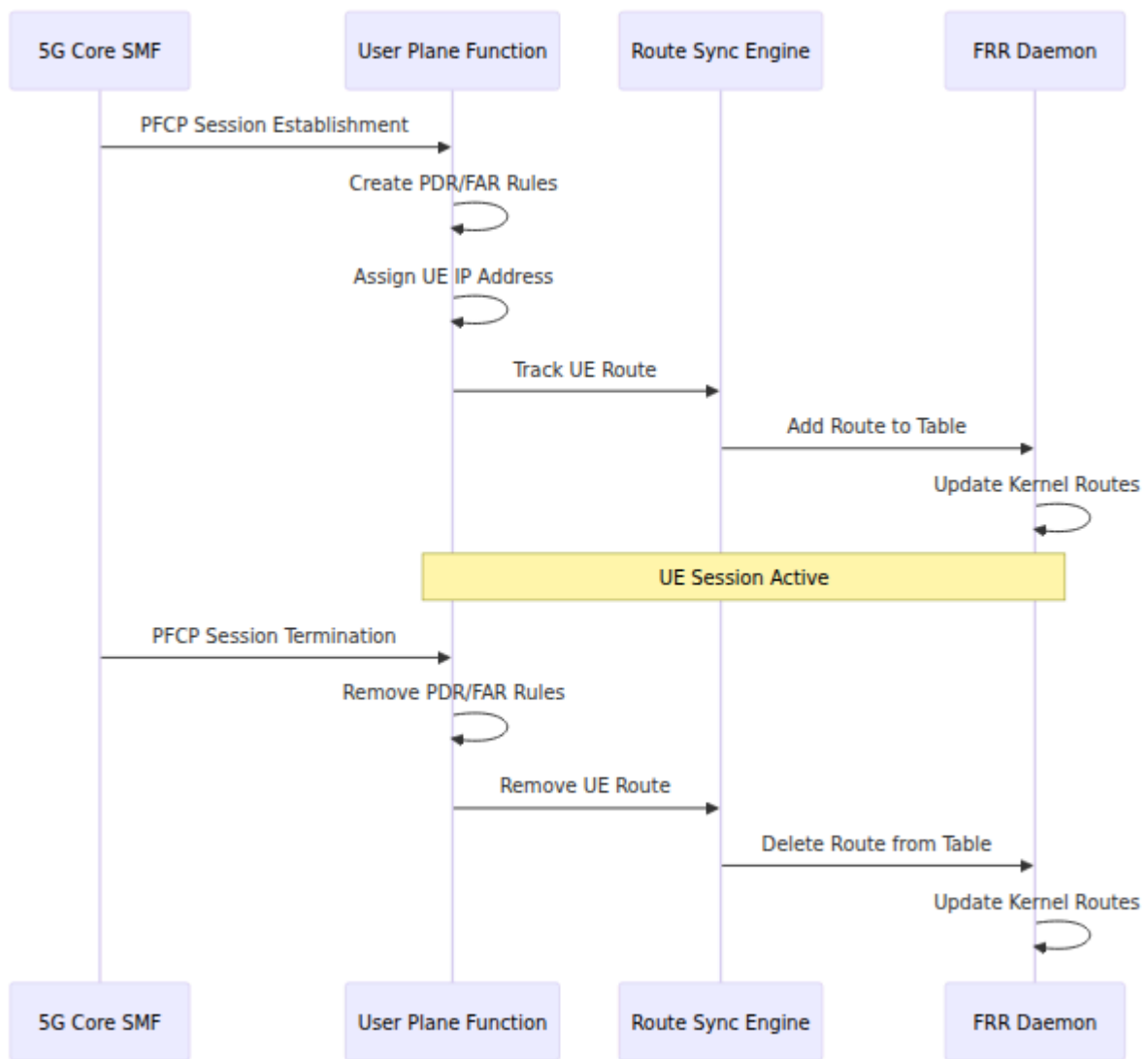
1. 透過 **FRR** 透過 Ansible 配置 UPF 透過 vtysh
2. 透過 `**` 透過 `redistribute static` 透過 FRR 配置
3. 透過 **FRR** 透過 Ansible 配置 OSPF/BGP 透過 Ansible
4. 透過 Ansible 配置 UE 透過 Ansible 配置 UPF 透過 Ansible

透過 UPF 透過 FRR 透過 vtysh 透過 Ansible 配置 FRR 透過 Ansible 配置 OSPF/BGP 透過 Ansible `redistribute static` 透過 Ansible `redistribute kernel` 透過 Ansible

透過 Ansible

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

□□□□



□□□□□

## Web UI □□

UPF □□□□□□□□ □□ □□□□□□

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

- **BGP** 詳細 BGP 詳細説明
- **OSPF** 詳細 UE 詳細説明 LSA 詳細説明

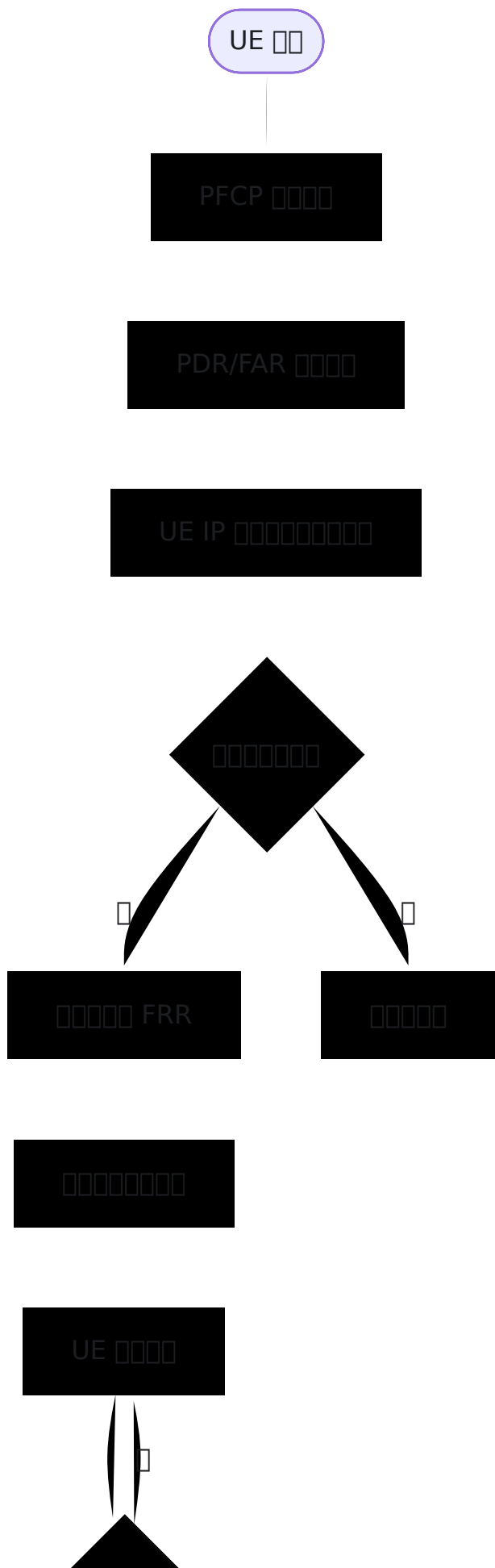
詳細説明 UE 詳細説明

詳細説明

詳細説明 Web UI 詳細説明

1. 詳細 UPF 詳細説明 UE 詳細説明
2. 詳細 FRR 詳細説明
3. 詳細説明
4. 詳細説明
5. 詳細説明







□



□□

- API
- API UE
- API UE
- API
- API 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

配置 IPv4 地址 `100.64.18.5` 并配置静态路由

- 配置 IP
- 配置掩码
- 配置地址
- 配置静态路由

## IPv6 配置

配置 IPv4 和 IPv6 UE 配置

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

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

FRR OSPF UE OSPF LSA 5 OSPF

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

1. UPF UE IP
2. FRR
3. FRR OSPF
4. OSPF