

OmniUPF 開箱

簡介

1. 背景
2. 為何 5G 需要 UPF
3. UPF 簡介
4. PCF 與 SMF 簡介
5. 網路架構
6. 網路功能
7. 網路性能
8. 網路安全

背景

OmniUPF 採用 eBPF 技術實現 5G/LTE 網路功能，並提供 QoS 控制。OmniUPF 基於 Linux eBPF 技術實現，OmniUPF 支援 5G SA/5G NSA 及 LTE 網路功能。

OmniUPF 支援 5G/LTE 網路功能

OmniUPF 支援 3GPP 5G 及 LTE 網路功能。

- 支援 5G/LTE 網路功能
- 支援 QoS 控制
- 支援 5G/LTE 網路功能
- 支援 5G/LTE 網路功能
- 支援 5G/LTE 網路功能
- 支援 5G/LTE 網路功能

OmniUPF 支援 3GPP TS 23.501 5G 及 TS 23.401 LTE 網路功能。OmniUPF 基於 Linux 及 eBPF 技術實現。

OmniUPF 特徴

特徴

- 3GPP 準拠
- eBPF による柔軟な制御
- GTP-U/GPRS による柔軟な制御
- IPv4 / IPv6 対応
- XDPによる高速パケット処理
- 柔軟な拡張性

QoS 機能

- QoS による QER
- PDR
- FAR
- SDF
- URR

特徴

- PCF による柔軟な制御 SMF/PGW-C
- RESTful API
- 柔軟な拡張性
- eBPF による柔軟な制御
- Web による管理

特徴

- eBPF による柔軟な制御
- 柔軟な拡張性
- 柔軟な拡張性
- 柔軟な拡張性
- 柔軟な拡張性

柔軟な拡張性 Web UI 対応

UPF

OmniUPF 5G SA 5G NSA 4G LTE/EPC

OmniUPF

- **UPF** - 5G/NSA N4/PFCP OmniSMF
- **PGW-U** **PDN** - 4G EPC Sxc/PFCP OmniPGW-C
- **SGW-U** - 4G EPC Sxb/PFCP OmniSGW-C

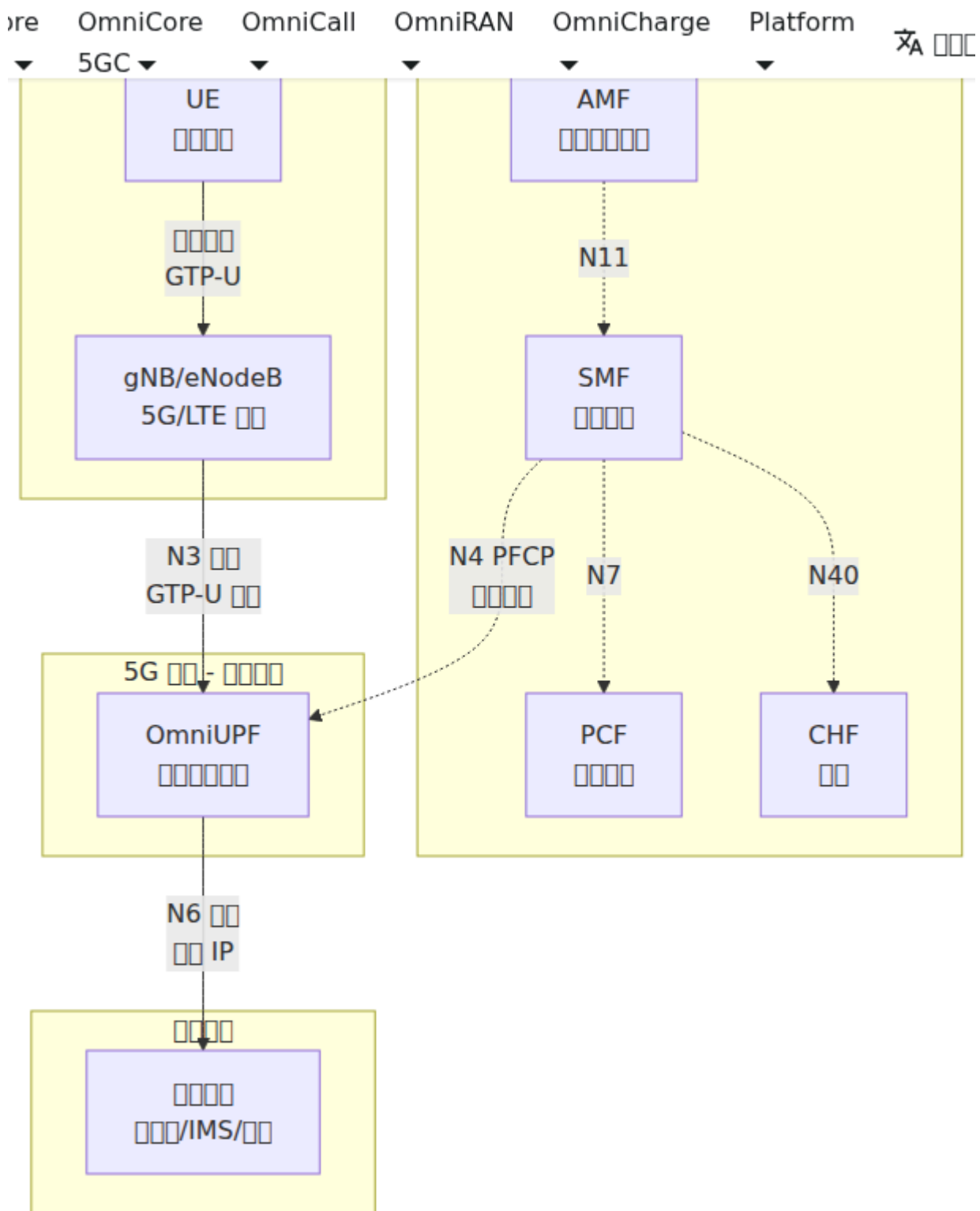
OmniUPF

- **UPF** 5G
- **PGW-U + SGW-U** 4G EPC
- **UPF + PGW-U + SGW-U** 4G 5G

eBPF PFCP UPF PGW-U SGW-U

5G SA

OmniUPF 5G

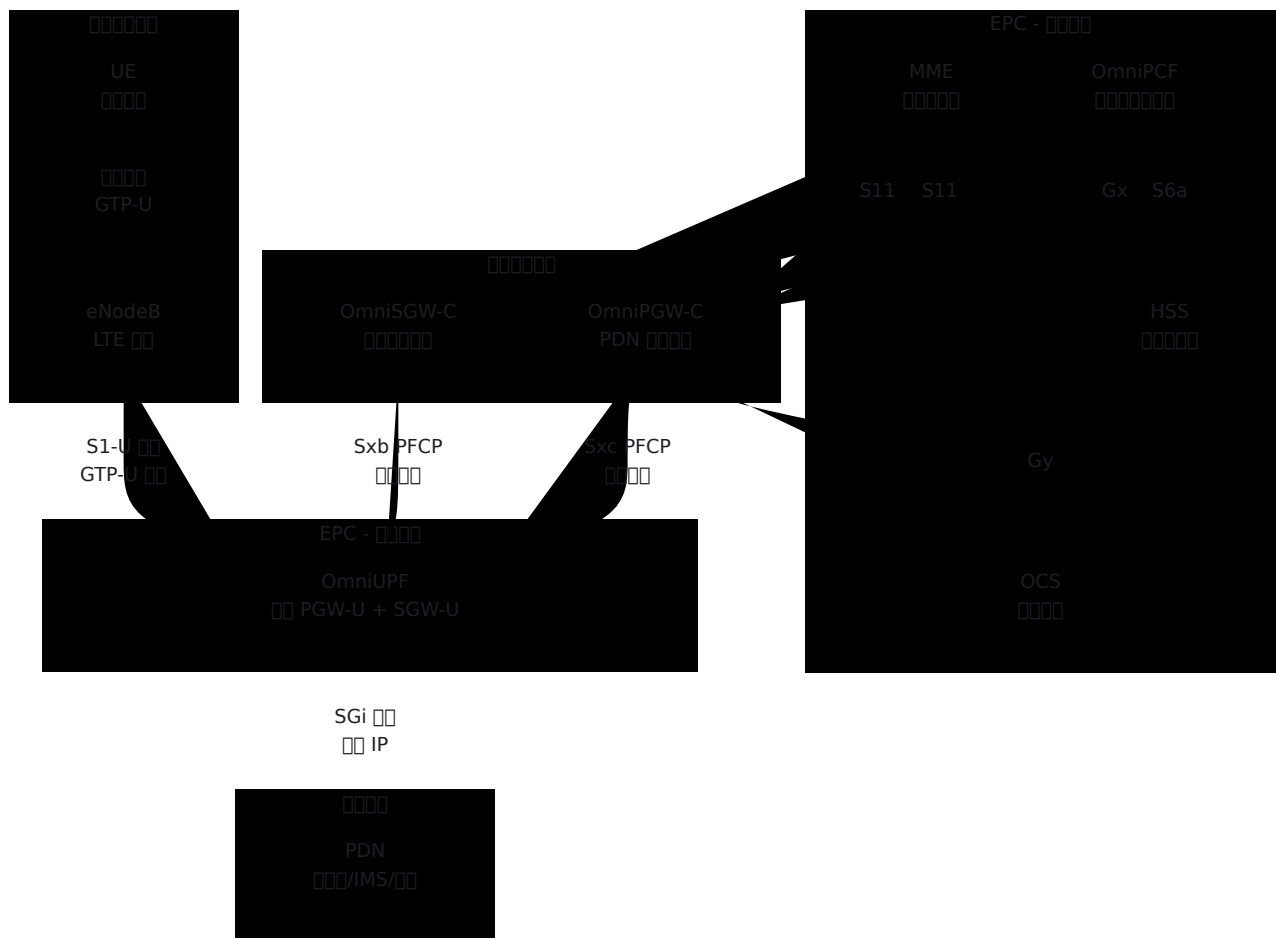


4G LTE/EPC

OmniUPF 4G LTE EPC OmniPGW-U OmniSGW-U

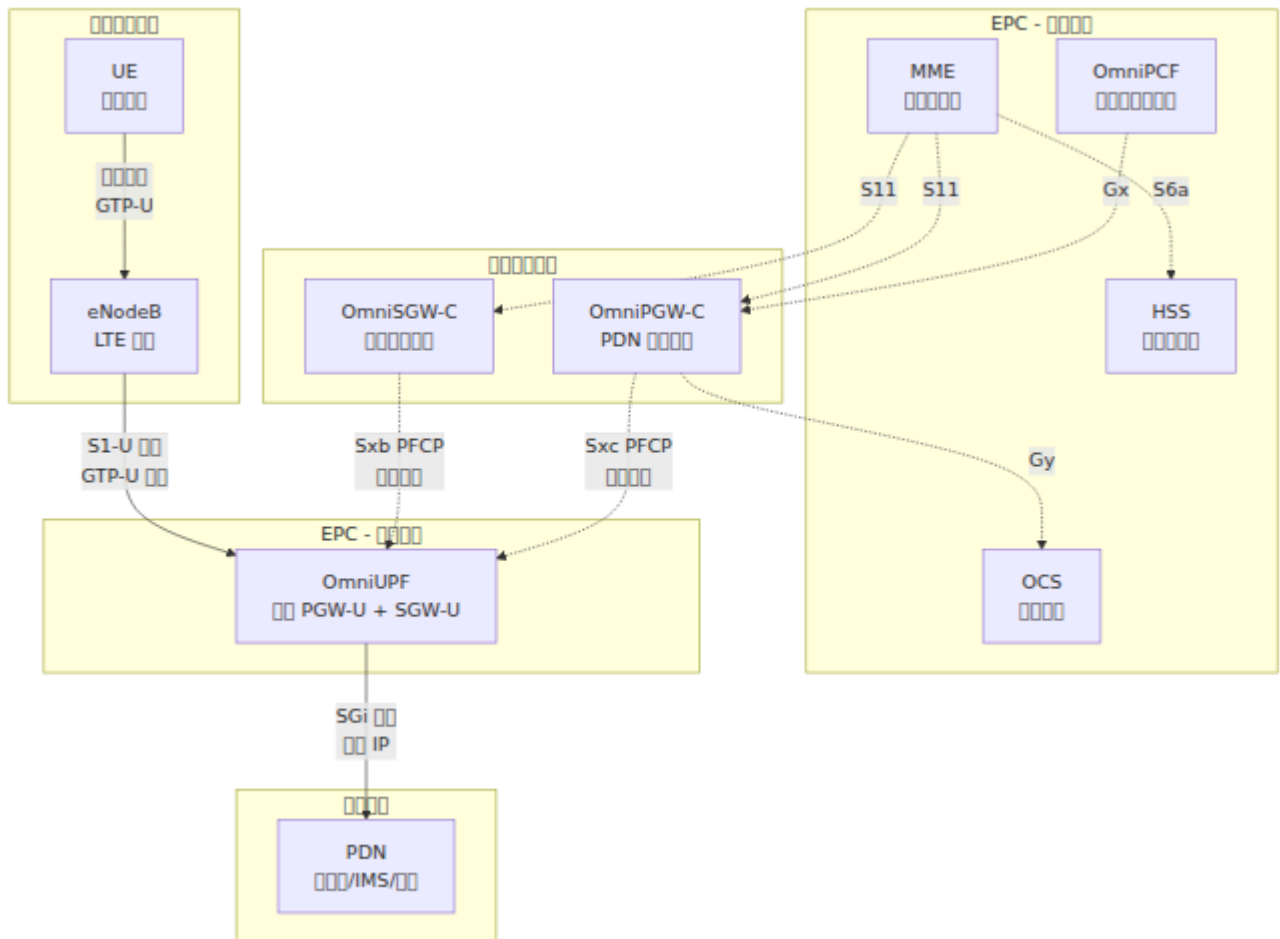
PGW-U/SGW-U 4G

OmniUPF SGW-U PGW-U



SGW-U PGW-U 00000/0000

OmniUPF - SGW-U PGW-U



4G/5G EPC Integration

4G/5G EPC Integration: OmniUPF (PGW-U + SGW-U) and OmniSGW-C / OmniPGW-C PDN

1. 4G/5G

- **5G** OmniSMF (N4) and OmniUPF (PGW-U) PFCP
- **4G** OmniPGW-C and OmniSGW-C Sxb/Sxc and OmniPGW-U/OmniSGW-U PFCP
- 4G/5G UE PDU 5G PDP 4G PFCP
- PFCP PDR, FAR, QER, URR
- eBPF (Diamonds)

2. 4G/5G UE → 4G/5G

- **5G** N3 gNB GTP-U
- **4G** S1-U SGW-U S5/S8 PGW-U eNodeB GTP-U

- トラフィック TEID を通知する PDR あり
- eBPF を利用し QERを通知する
- FAR を通知する
- ネット GTP-U を通知する N6 5G ネット SGi 4G ネット
- URR を通知する

3. ネットワーク → UE

- **5G** ネット N6 ネット IP ネット
- **4G** ネット SGi ネット IP ネット
- ネット UE IP ネット PDR あり
- SDF を通知する
- FAR ネット GTP-U ネット
- ネット TEID ネット GTP-U ネット
- **5G** ネット N3 ネット gNB
- **4G** ネット S1-U ネット SGW-U ネット S5/S8 ネット PGW-U ネット eNodeB

4. ネットワーク

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

ネットワーク 4G ネット 5G ネット

OmniUPF ネット 3GPP ネット 5G ネット 4G ネット

5G ネット

번호	방향	프로토콜	3GPP 사양
N4	OmniSMF ↔ OmniUPF	PFCP 프로토콜	TS 29.244
N3	gNB → OmniUPF	5G RAN 프로토콜 GTP-U	TS 29.281
N6	OmniUPF → 인터넷	인터넷 DN 프로토콜 IP	TS 23.501
N9	OmniUPF ↔ OmniUPF	5G/4G/LTE UPF 프로토콜	TS 23.501

4G/EPC

번호	방향	프로토콜	3GPP 사양
Sxb	OmniSGW-C ↔ OmniUPF/SGW-U	4G 프로토콜 PFCP	TS 29.244
Sxc	OmniPGW-C ↔ OmniUPF/PGW-U	PDN 프로토콜 PFCP	TS 29.244
S1-U	eNodeB → OmniUPF/SGW-U	4G RAN 프로토콜 GTP-U	TS 29.281
S5/S8	OmniUPF/SGW-U ↔ OmniUPF/PGW-U	4G 프로토콜 GTP-U	TS 29.281
SGi	OmniUPF/PGW-U → PDN	인터넷 프로토콜 IP	TS 23.401

4G 프로토콜 PFCP는 N4, Sxb, Sxc와 TS 29.244에 정의되며, 5G 프로토콜 PFCP는 N4, Sxb, Sxc와 TS 29.244에 정의됩니다.

UPF

eBPF

eBPF Linux

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

eBPF eBPF

uplink_pdr_map	PDR	TEID32	PDR FAR ID QER ID URR IDs
downlink_pdr_map	PDR IPv4	UE IP	PDR
downlink_pdr_map_ip6	PDR IPv6	UE IPv6	PDR
far_map		FAR ID	
qer_map	QoS	QER ID	QoS MBR GBR
urr_map		URR ID	
sdf_filter_map	SDF	PDR ID	

- 可移植なフレームワーク
- **XDP** による高性能なパケット処理
- 単一 CPU ではなく複数 CPU 対応
- 従来の eBPF よりも PDR/FAR などの処理が楽

可移植なフレームワーク

PFCP の概要

PFCP は 3GPP TS 29.244 で定義された SMF と PGW-C 間の

プロトコル

- 単一の PFCP セッション/ルール
- 複数のネットワーク要素間で PFCP ルール
- 単一の PFCP セッションで eBPF を利用
- 単一の SMF が複数のネットワーク要素

PFCP の機能

方向	送信元	受信先
送信	SMF → UPF	PFCP セッション
送信	SMF → UPF	PFCP ルール
受信	UPF	セッション
送信	SMF → UPF	パケット PDU を含む PDR/FAR/QER/URR
送信	SMF → UPF	ネットワーク QoS 設定
送信	SMF → UPF	パケットの転送
送信	UPF → SMF	パケットの転送

3GPP TS 23.501 5G Core Network Architecture

- PDR, FAR, QER, URR
- PDR, FAR, QER, URR
- PDR, FAR, QER, URR
- UE IP, FQDN, SDF
- QoS, MBR, GBR, QFI
- eBPF

REST API

REST API, UPF

3GPP TS 23.501

- PCF, PCF
- PDR, FAR, QER, URR
- XDP
- eBPF

API, 34

API	API	API
健康检查	/health	健康检查
配置	/config	UPF 配置
PCF	/pfcpsessions, /pfcpsessions	PCF 会话/关联 管理
PDRs	/uplinkpdrmap, /downlinkpdrmap, /downlinkpdrmapip6, /uplinkpdrmapip6	策略数据规则
FARs	/far_map	转发策略规则
QERs	/qer_map	QoS 策略规则 管理
URRs	/urr_map	计费策略规则
缓冲	/buffer	缓冲策略规则 管理
统计	/packetstats, /route_stats, /xdp_stats, /n3n6_stats	统计
映射	/map_info	eBPF 映射 管理
数据面配置	/dataplane_config	N3/N9 数据面 配置

API 列表

Web 管理

Web 管理 UPF 管理

111

- 5G Core Network PFCP 5G Core Network UE IP TEID 5G Core Network
- 5G Core Network PDR FAR QER 5G Core Network URR
- 5G Core Network FAR 5G Core Network
- 5G Core Network XDP 5G Core Network N3/N6 5G Core Network
- 5G Core Network eBPF 5G Core Network 5G Core Network
- 5G Core Network UPF 5G Core Network
- 5G Core Network 5G Core Network

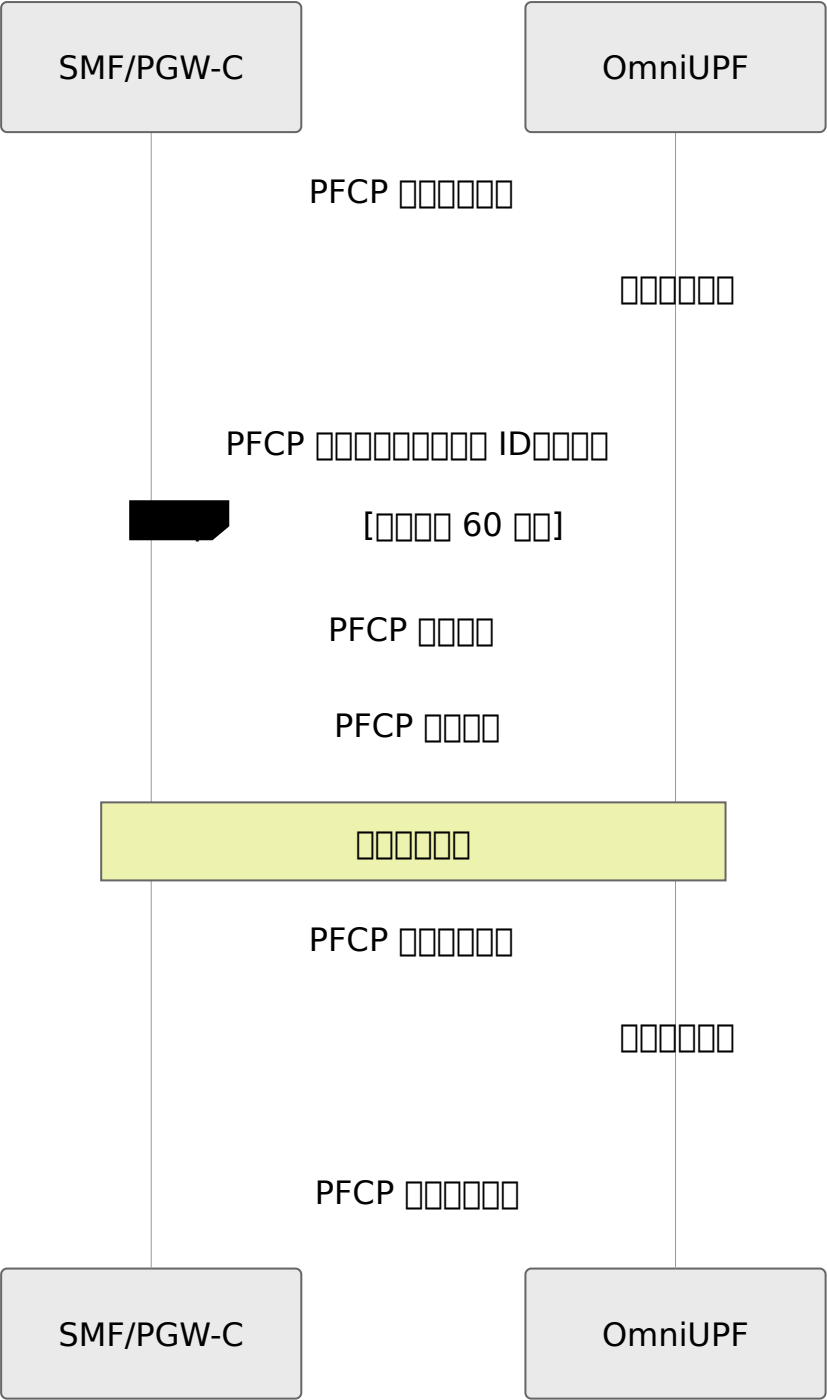
□□□□ UI □□□□□ **Web UI** □□□□

PFCP **SMF**

PFCP ☐ ☐

□□□□□□□□SMF □□□ UPF □□ PFCP □□□

□ □ □ □ □ □ □



Network

- SMF and UPF are connected via N4 interface
- UPF sends ID/FQDN and IP address to SMF
- SMF sends PFCP messages to UPF
- SMF sends PFCP messages to UPF

Network and PFCP interface

SMF 功能

OmniUPF 实现 SMF 功能 3GPP TS 29.244 规范

功能

SMF 与 PCFP 交互实现 OmniUPF 功能 SMF 功能

1. SMF 功能
2. SMF 与 UPF 交互 PCFP 功能
3. SMF 与 交互
4. UPF 功能 = SMF 功能
5. UPF 功能 SMF 功能
6. SMF 功能

功能



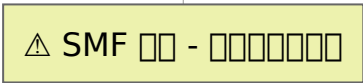
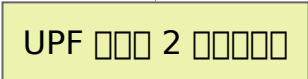
5G Core
(Timestamp: 2025-01-15 10:00:00)

5G Core

5G Core

5G Core UE 1

5G Core UE 2

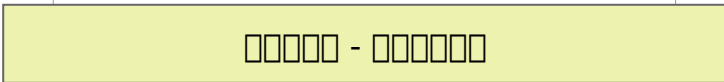


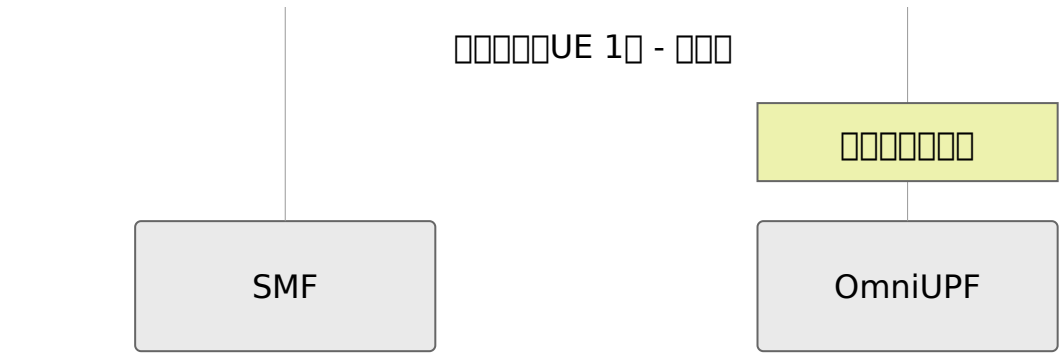
5G Core
(Timestamp: 2025-01-15 10:30:15)

5G Core:
- 10:00:00
- 10:30:15
- SMF 2

5G Core:
- UE 1
- UE 2
- eBPF
- 5G Core

5G Core





Network

SMF configuration

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

Network

- SMF configuration SMF configuration
- SMF configuration SMF configuration
- 3GPP** 3GPP TS 29.244 5.22.2

“CP configuration UP configuration CP configuration PFCP”

Network configuration

GTP-U

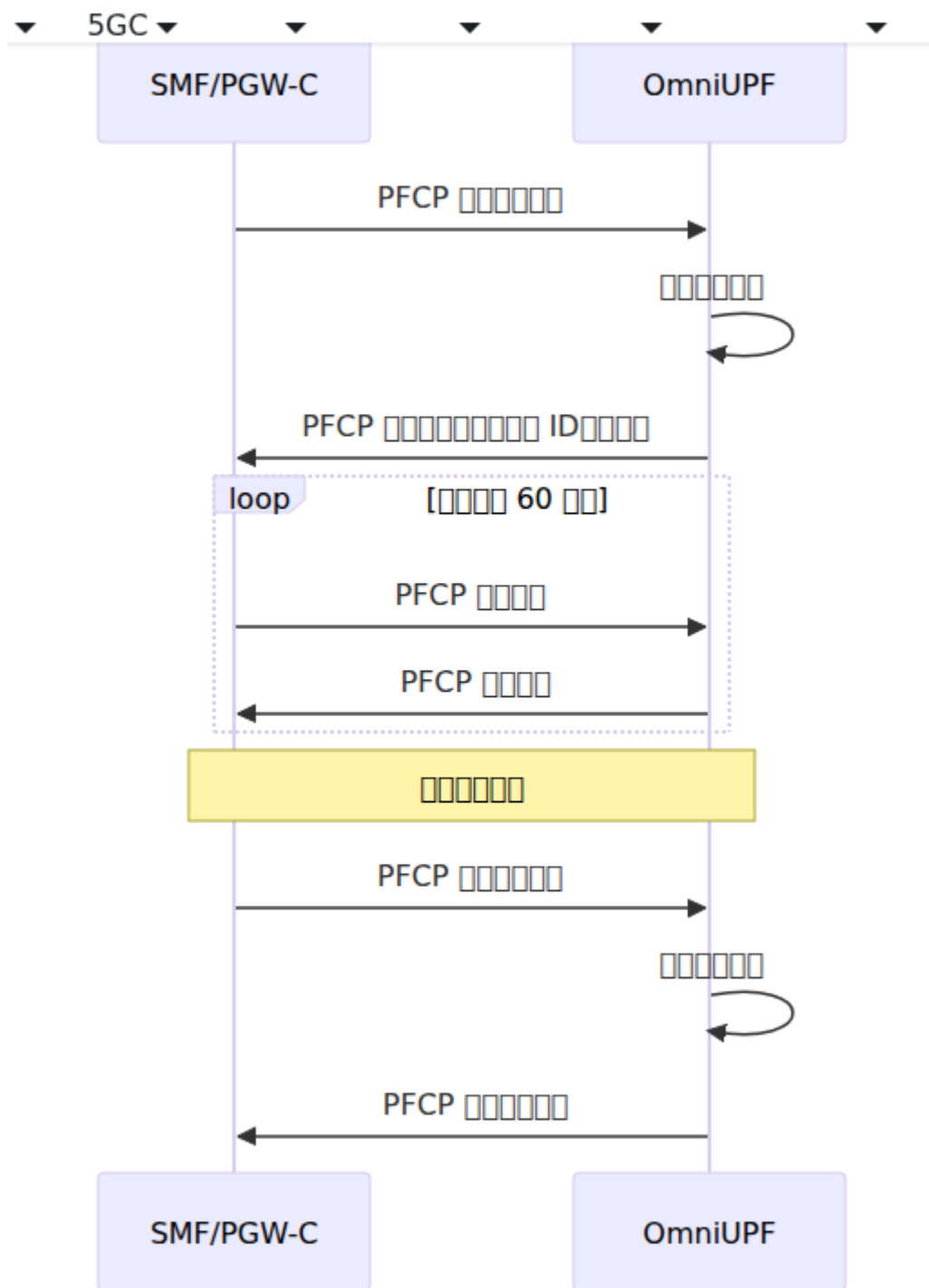
OmniUPF 3GPP TS 29.281 PGW-U SGW-U eNodeB gNodeB GTP-U

Network

OmniUPF GTP-U SGW-U PGW-U TEID

-
-
-

- UPF → TEID X GTP-U 2152
- TEID X → TEID
- GTP-U 26 IE TEID
- UPF → TEID X
- UPF → TEID X FAR
- UPF → eBPF PFCP
- UPF → Prometheus



3GPP TS 29.281 7.3.1

GTP-U

GTP-U12			
PT	0x32		
	26 (0x1A)		
	9		
TEID	0 (00)		
N-PDU	0		
	0		
IE: TEIDI5			
	16 (0x10)		
TEID	4		

1 S5/S8 GTP PGW-U

- SGW-UOmniUPFS5/S8PGW-U
- PGW-US5/S8
- SGW-U TEID
- PGW-U
- SGW-U

2 N9 UPF

- UPF-1OmniUPFN9UPF-2
- UPF-2
- UPF-1
- UPF-1


```
WARN: 192.168.50.10:2152GTP-UTEID 0x12345678 - 
TEID
WARN: LocalSEID=42FAR GlobalId=1TEID 0x12345678
192.168.50.10
INFO: LocalSEID=42192.168.50.10TEID 0x12345678GTP-U
WARN: 192.168.50.10TEID 0x12345678GTP-U1
```

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"}

# 
upf_buffer_listener_error_indications_sent_total{node_id="enodeb-1",peer_address="10.60.0.1"}
```

- node_idPFCEID“”
- peer_addressIP

-
- TEID**TEID
-
- TEID

5. □□□□□□□□

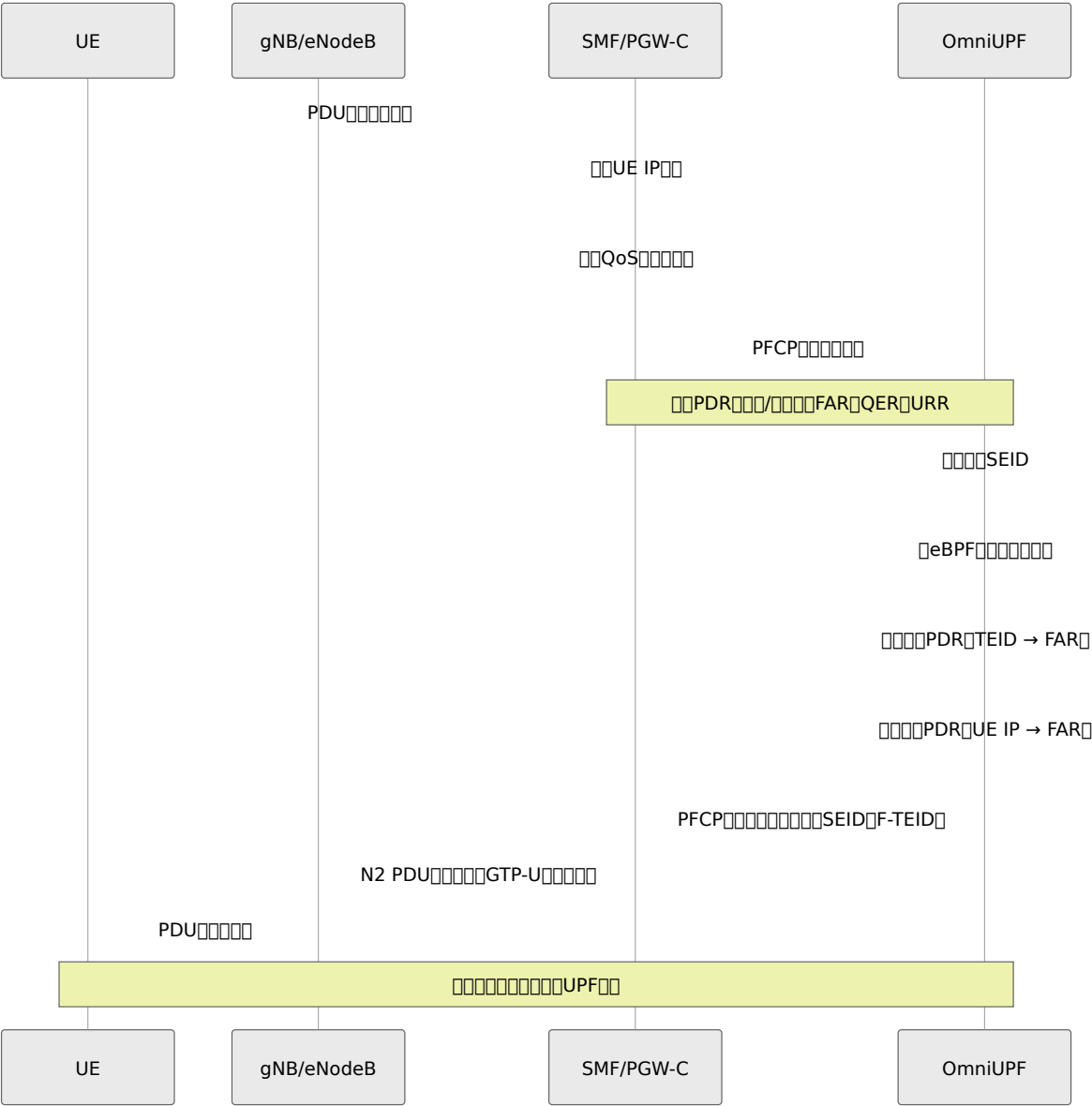
- □□□□□□ = □□□□□□□□□□□□□□
- □□□□□□ = □□□□□□□□□□□□□□

6.

PFCP□□□□

UE PDU 5G PDP LTE SMF UPF PCF

□ □ □ □ □ □ □



Network

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

PFCP

SMF QoS

Network

1. 網路N2

- 網路gNB網路F-TEID網路FAR
- 網路網路網路網路
- 網路網路網路網路

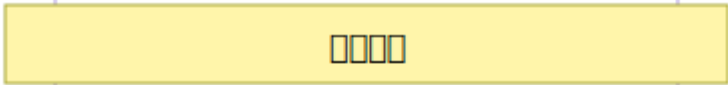
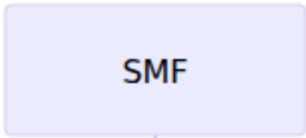
2. QoS

- 網路MBR/GBR網路QER
- 網路PDR網路/網路SDF網路網路網路QoS

3. 網路

- 網路網路網路PDR
- 網路FAR網路網路

網路網路



SMF - OmniUPF

SMF - OmniUPF

(Timestamp: 2025-01-15 10:00:00)



SMF - OmniUPF

SMF - OmniUPF UE 1

SMF - OmniUPF UE 2

UPF - SMF 2

△ SMF - SMF

SMF - OmniUPF

(Timestamp: 2025-01-15 10:30:15)

SMF - OmniUPF

Timestamp: 10:00:00

Timestamp: 10:30:15

SMF - SMF



SMF - OmniUPF

- SMF UE 1

- SMF UE 2

- SMF eBPF

- SMF - SMF

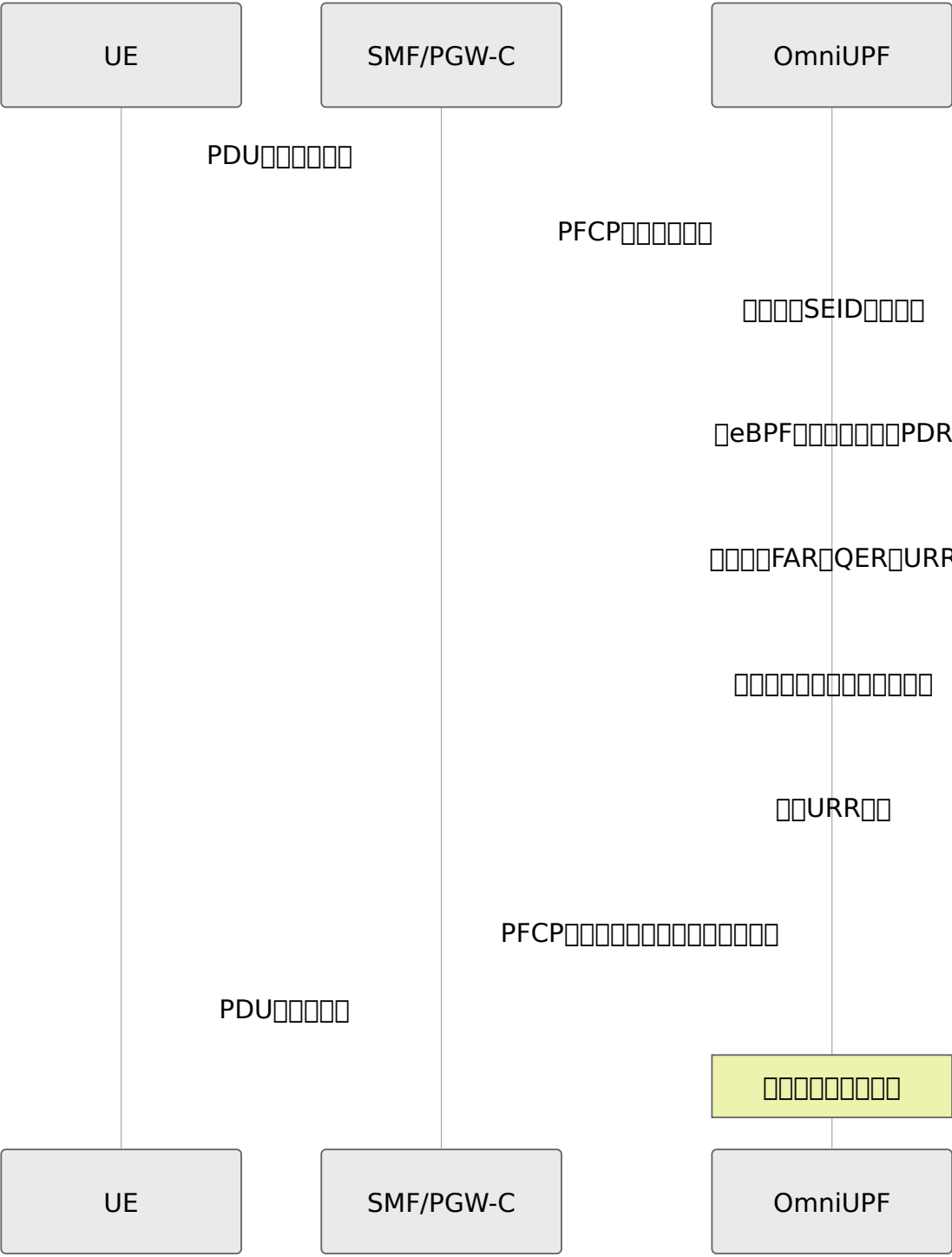


SMF - OmniUPF

SMF - SMF



□□□□□□



Sequence

- PDR
- FAR, QER, URR
-
- SMF

--	--	--	--

OmniUPF Web REST API

--	--	--	--

PFCP

PFCP 5G PDP LTE

- 國際SEID標準
- 國際PDR
- 國際FAR
- 國際QoS/QER
- 國際URR

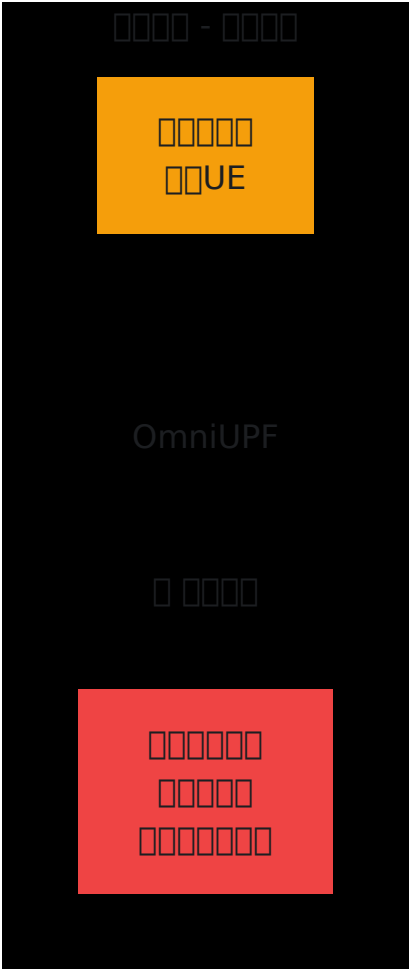
□□□□□□

- 〇〇〇〇〇〇〇〇UE IP〇〇〇TEID〇〇〇〇〇
- 〇**IP**〇〇〇**TEID**〇〇〇〇
- 〇〇〇〇〇〇〇〇〇〇〇〇〇〇〇PDR/FAR/QER/URR〇〇
- 〇〇〇〇**PFCP**〇〇〇〇〇〇〇〇

□□□□□□□□**PDR**□□

PDR□□□□□□□□□□□□□□□□

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





以下為測試環境配置說明

- **TCP** 連接埠配置
- 支援的應用程式: Zoom, Teams, WhatsApp
- 網路拓撲結構
- ****VoIP 測試**** 配置
- 安全策略配置

OmniUPF 測試環境配置說明

測試環境

OmniUPF 測試環境配置

1. N2 接口 5G / X2 接口 4G

UE 測試環境配置

```

sequenceDiagram
    participant UE
    participant gNB as gNB/eNodeB
    participant SMF as SMF/PGW-C
    participant OmniUPF

    SMF->>SMF: UE IP
    SMF->>SMF: QoS
    SMF->>OmniUPF: PFCP
    Note over SMF,OmniUPF: PDR, FAR, QER, URR
    OmniUPF->>OmniUPF: SEID
    OmniUPF->>OmniUPF: eBPF
    OmniUPF->>OmniUPF: PDR, TEID -> FAR
    OmniUPF->>OmniUPF: PDR, UE IP -> FAR
    OmniUPF->>SMF: PFCP, SEID, F-TEID
    SMF->>gNB: N2 PDU, GTP-U
    gNB->>UE: PDU
    Note over SMF,OmniUPF: UPF
  
```

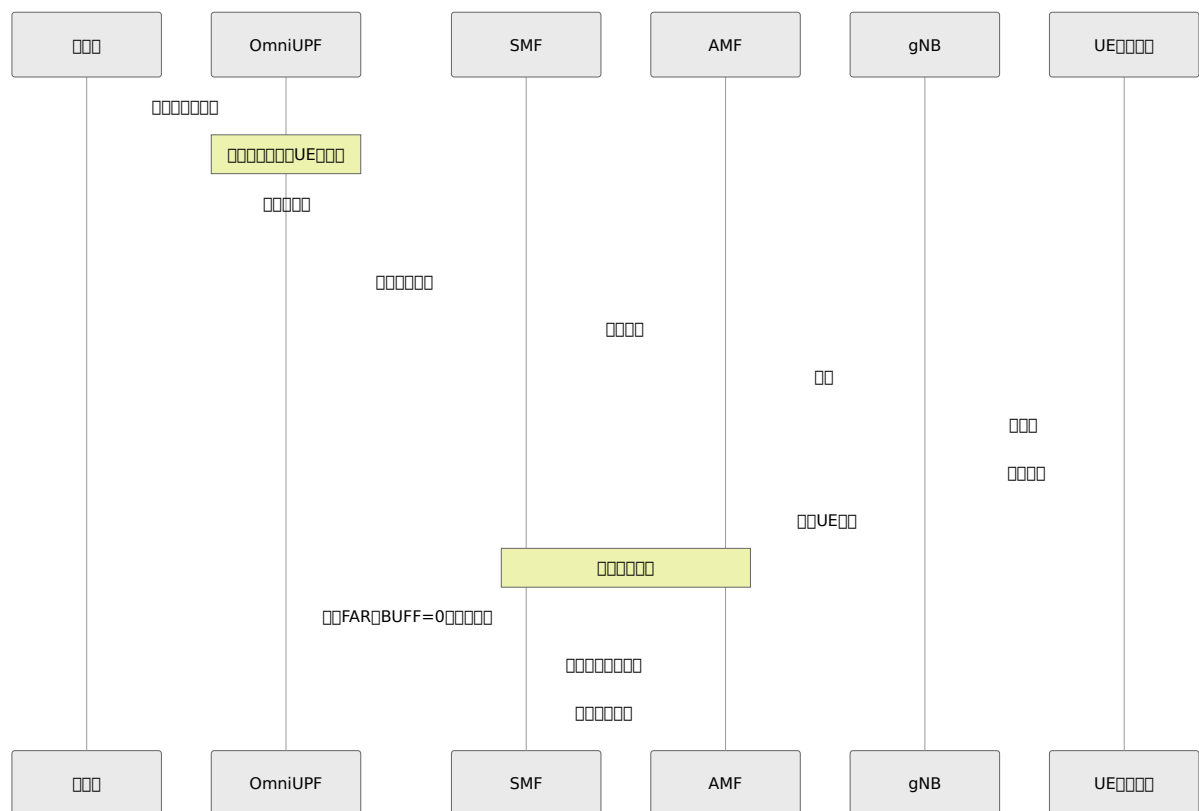
The diagram illustrates the PFCP session establishment process between the SMF/PGW-C and the OmniUPF. The process begins with the SMF/PGW-C performing local actions: it receives the UE IP and QoS parameters. It then sends a PFCP message to the OmniUPF. This message triggers the OmniUPF to perform several local actions: it generates a SEID, configures the eBPF, and sets up PDRs (Policy Data Rules) with TEID and UE IP, associating them with FARs (Forwarding Action Rules). The OmniUPF then sends a PFCP message back to the SMF/PGW-C, providing the SEID and F-TEID. The SMF/PGW-C then sends an N2 PDU and GTP-U to the gNB/eNodeB, which finally delivers the PDU to the UE. A yellow bar at the bottom indicates the UPF (User Plane Function) context.

- **QoS** / **4G** / **5G** / **NSA**
- **ULCL**
- **UPF**

[illegible]

3. 数据字典

QUE

[illegible]

4. **RAT** 4G ↔ 5G

UE 4G 5G

- eNodeB ↔ gNB
- TEID
- RAT

OmniUPF□□□□□□□□

□□□□□

OmniUPF□□□□□□□□□□

- 1. **eBPF**□□□□□□□□□□FAR□□□□□□□□□□□□□□
- 2. □□□□□□□□□□□□□□□□□□□□□□□□

□□□□□

5G Core
N6

PDR

FAR

- PDR

BUFF=1

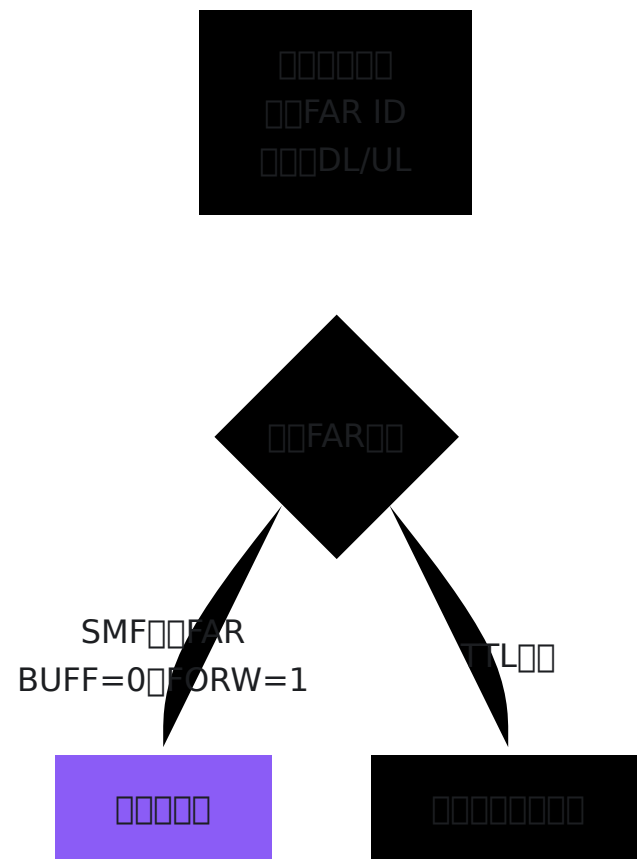
FORW=1

=1

eBPF: GTP-U

gNB

localhost:22152



SMF FAR ID

SMF FAR ID

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

SMF FAR ID

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

3. 0000000000000000FAR ID0000000000000000
4. 00000SMF000000000000FAR00FORW=10BUFF=0
5. 0000000000000000FAR0000000000000000000000
6. 0000000000000000000000000000000000000000

□□□□□□□□□□

□□□□□□□

項目	概要	詳細
ネットワーク構成	概要1-2ネットワーク構成	詳細説明
セキュリティ対策	セキュリティ対策概要	詳細説明
運用管理	運用管理概要	詳細説明
バックアップ・復旧計画	バックアップ・復旧計画概要	詳細説明
災害対策	災害対策概要	詳細説明
その他	その他概要	詳細説明

□ □ □ □ □ □ □ □ □

- 國際標準化組織 (ISO) 國際標準
- 國際標準化組織 (ISO) 國際標準
- 國際標準化組織 (ISO) 國際標準
- 國際標準化組織 (ISO) 國際標準

□□□□□□

□□□□□□ Web UI □ API □□□□□□□□

103

- 00000000 **FAR ID** 00000000000000000000
- 00000000000000000000 FAR000000
- 00000000000000000000
- 00000000000000000000 > TTL 000

□□□□□

- 000000000000000000000000
- 00000000000000000000000000000000
- 00 **TTL**000000000000
- 000000000000 FAR 00000000

□□□□□

- 送信元アドレス SMF 宛先アドレス FAR 送信元ポート
- 送信元アドレス送信元ポート宛先アドレス宛先ポート
- 送信元アドレスTTL 送信元ポート FAR 宛先ポート
- 送信元アドレス送信元ポート宛先アドレス SMF 宛先ポート

□□□□

```
□ /etc/omniupf/runtime.exs □□□□□□□□
```

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

111

- `buffer_max_packets` 20,000+

- 設定可能なパラメータ `buffer_packet_ttl` の値 15
- 設定可能な `buffer_packet_ttl` の値 10 未満
- 設定可能なパラメータの範囲を制限する

設定可能なパラメータ 0000

設定可能な

設定可能な

設定可能なパラメータ

- **RX** 設定可能なパラメータ
- **TX** 設定可能なパラメータ
- 設定可能なパラメータの範囲を制限する
- **GTP-U** 設定可能なパラメータ

設定可能な

設定可能なパラメータ

- 設定可能なパラメータ
- 設定可能なパラメータ/パラメータ
- 設定可能なパラメータ TEID/UE IP

XDP 設定

eXpress 設定可能なパラメータ

- **XDP** 設定可能な XDP 設定可能なパラメータ
- **XDP** 設定可能なパラメータ
- **XDP** 設定可能な XDP 設定可能なパラメータ
- **XDP** 設定可能なパラメータ

N3/N6 設定可能な

設定可能なパラメータ

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

UPF

UPF

eBPF

UPF eBPF

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

UPF

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

UPF

- PDR +
- UPF
-

UPF

□□□□

UPF □□□

□□□□□ UPF □□□□□

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

□□□□□□□

□□□ eBPF □□□□□□□

- □□ **N3** □□□□□□ N3 □□□□
- □□ **N9** □□□□□□ N9 □□□□□□□□□□

□□□□□□□□□□ □□□□□

□□□□

□□□□□□□□□□□□□□□□□□

□□□□□□

□□□PCF □□□□□□□□□□□□□□□□□□

□□□□□□□

1. PCF □□□□□

- □□ SMF □□□□□□□ UPF PCF □□□□□ 8805□
- □□□□□□□□□□ PCF □□□□□
- □□ SMF □ UPF □□□□□ ID □□□□□□□

2. eBPF ☐☐☐☐☐☐

- 透水性係數 $>90\%$ 透水性良好
- 高 UPF 係數高 eBPF 係數高
- 透水性良好透水性良好

3. ☐ ☐ ☐ **PDR/FAR** ☐

- UE IP 10.10.10.10
- TEID 10000000
- FAR 10.10.10.10

4.

- 細胞 N3 細胞 IP 細胞細胞 gNB 細胞
- 細胞細胞細胞 N6 細胞細胞細胞
- 細胞 GTP-U 細胞細胞細胞細胞

[illegible]

UE 0000000000000000

□□□□□□

1. PDR □□□□

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

2. FAR □□□□

- FAR FORWARD DROP BUFFER
- GTP-U
-

3. QoS 設定

- 設定 QER MBRを適用する
- 設定 GBRを適用する
- 設定を適用する

4. 設定 MTU 設定

- 設定 GTP-U 設定40-50 設定を適用する
- 設定 N3/N6 設定 MTU 設定
- 設定 ICMP 設定

設定

設定を適用する

設定

1. 設定

- 設定 FAR 設定 2
- 設定 SMF 設定を適用する
- 設定を適用する

2. 設定 TTL 設定

- 設定
- 設定 TTL 設定を適用する
- 設定

3. 設定

- 設定 FAR 設定
- 設定を適用する
- 設定 max_per_far 設定 max_total 設定

設定を適用する 設定

目標

ネットワークの最適化

ネットワーク

1. ネットワーク

- eBPF を使って 64 ビット幅の整数
- ネットワークの最適化
- ネット URRL を最適化する

2. ネットワーク

- ネット eBPF を最適化する
- ネットワークの最適化 eBPF を
- ネット XDP を最適化する

3. ネットワーク

- ネット N3/N6 を最適化する
- ネットワークの最適化 eBPF を最適化する
- ネットワークの最適化 XDP を最適化する

目標

ネットワークの最適化 CPU

ネットワーク

1. ネット **XDP** を最適化する XDP を最適化する
2. ネット **eBPF** を最適化する eBPF を最適化する
3. ネット **CPU** を最適化する eBPF を最適化する
4. ネットワークの最適化 NIC を最適化する XDP を最適化する

ネットワーク

- **XDP** を最適化する 10M+ を最適化する

- **PDR** [PDR](#) [PDR](#)
- [UPF](#) [UPF](#)
- [NIC](#) [NIC](#)

[PDR](#) [UPF](#)

[PDR](#)

[UPF](#)

[UPF](#) [UPF](#)

[UPF](#)

[UPF](#)

- [YAML](#) [CLI](#)
- [UPF/PGW-U/SGW-U](#)
- [XDP](#) [XDP](#)
- [Proxmox](#) [VMware](#) [KVM](#) [Hyper-V](#) [VirtualBox](#)
- [NIC](#) [XDP](#) [XDP](#)
- [XDP](#)
- [XDP](#)

XDP [XDP](#)

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

- [XDP](#) [XDP](#) [XDP](#)
- [XDP](#)
- [Proxmox VE](#) [XDP](#) [XDP](#)
- [XDP](#)
- [VMware ESXi](#) [KVM](#) [Hyper-V](#) [XDP](#) [XDP](#)
- [XDP](#) [XDP](#)
- [XDP](#) [XDP](#)

📌

📌

- eBPF 📌
- 📌 XDP 📌
- PFCP 📌
- 📌GTP-U 📌 22152📌
- QoS 📌5ms 📌
- 📌3.5μs 📌10 Mpps/core📌

📌

PFCP 📌

- 📌 (PDR) - 📌
- 📌 (FAR) - 📌
- QoS 📌 (QER) - 📌 (MBR/GBR)
- 📌 (URR) - 📌
- 📌
- 📌

📌

📌

- N3/N6 📌
- XDP 📌/📌/📌/📌
- 📌 eBPF 📌
- 📌
- 📌
- 📌

Web UI 📌

📌

- 📌

- 3GPP TS 23.501/23.502
- 3GPP TS 23.503 PDR, FAR, QER, URR
- 3GPP TS 23.504
- 3GPP TS 23.505
- eBPF
- 3GPP TS 23.506

API

REST API

- OpenAPI/Swagger
- API
- PCF
- PDR - IPv4, IPv6
- FAR
- QoS (QER)
- URR
-
-
- FRR
- eBPF
-
-
- API

Prometheus

- PCF
- XDP
-
- PCF
- URR PCF

- 网络性能优化和故障排除
- 网络性能优化DLDR 模型
- eBPF 网络性能优化
- Prometheus 监控
- Grafana 可视化

PFCP 网络性能

PFCP 网络性能

- 网络性能 3GPP 网络TS 129.244
- 网络性能优化DLDR 模型
- 网络性能优化
- 网络 Prometheus 监控
- 网络性能优化
- 网络性能优化
- 网络性能优化

UE 网络性能

FRR 网络性能

- FRR网络性能优化
- UE 网络性能
- 网络性能优化
- 网络 OSPF 和 BGP 网络性能
- OSPF 网络性能
- OSPF 网络 LSA 网络性能
- BGP 网络性能
- Web UI 网络性能
- 网络性能优化
- 网络性能优化 Mermaid 图

IPv6 / 网络性能

网络 IPv6 网络 IPv6 和 IPv4v6 PDN 网络

- 配置 IPv6 地址
- UE IPv6 地址
- UE /128 地址 OSPFv3 配置
- IPv6 路由/转发配置
- 配置 IPv6 邻居配置

配置

配置

- 配置
- 配置
- PFCP 配置
- 配置
- XDP 与 eBPF 配置
- 配置
- 配置 Proxmox/VMware/VirtualBox
- NIC 配置
- 配置

配置

配置 OmniUPF

1. 配置
2. 配置 配置
3. Web UI 配置 配置

配置 Proxmox 配置

1. XDP 配置 - Proxmox 配置 XDP 配置 - 配置
2. 配置 - 配置
3. 配置 - Proxmox SR-IOV 配置
4. 配置 - Proxmox 配置

□□□□

1. **XDP** □□□□ - □□□□ **XDP** □□□ **5-10** □□□□□□
2. □□□□ - □□□□
3. □□□□ - **XDP** □□
4. □□□□ - □□□□
5. □□□□ - □□□□

□□□□□□□

1. □□□□ - □□□□□□□□
2. □□□□□□
3. □□□□ - □□□□

□□□□

1. □□□□ - □□□□
2. □□□□ - □□□□
3. □□□□ - □□□□□□

□□ **UE** □□□ **FRR** □□

1. **UE** □□□□□□ - □□□□□□□□□□
2. **API** □□ - □□□□ - □□ **API** □□
3. **Web UI** □□ - □□□□□□
4. **UE** □□□□ - **FRR** □□ - OSPF LSA □□

□□ **REST API**

1. **API** □□ - □□□ **API** □□
2. **API** □□ - **Swagger UI** - □□□ **API** □□□
3. **API** □□ - **API** □□ - **API** □□□□
4. **Web UI** □□ - Web □□□□ **API** □□□□□

□□□□□□

1. □□□□□□ - □□□□□□
2. □□□□ - □□□□□□□□□□

3. Web UI 测试 - 接口测试

测试

测试 **API** 测试

OmniUPF 接口测试 REST API

```
# 测试
GET http://localhost:8080/api/v1/upf_status

# PFCP 测试
GET http://localhost:8080/api/v1/upf_pipeline

# 测试
GET http://localhost:8080/api/v1/sessions

# 测试
GET http://localhost:8080/api/v1/packet_stats
GET http://localhost:8080/api/v1/xdp_stats

# 测试
GET http://localhost:8080/api/v1/map_info

# 测试
GET http://localhost:8080/api/v1/upf_buffer_info
```

测试 API 接口 Swagger UI <http://<upf-ip>:8080/swagger/index.html>

测试

```

# /etc/omniupf/runtime.exs
xdp_interfaces = "eth0"
xdp_attach_mode = "native"
"offload"
n3_address = "10.100.50.233"
pfcf_address = "10.100.50.241"
pfcf_port = 8805
node_id = "10.100.50.241"

#
max_sessions = 100_000

# API
api_port = 8080

# N3/N6/N9
# "generic" | "native" |
# N3 IP
# PFCF
# PFCF
# PFCF ID

#
#

# REST API

```

- **eBPF** < **70%**
- **eBPF** **70-90%** 1
- **eBPF** > **90%** -
- < **0.1%**
- **0.1-1%** -
- > **5%** -
- **XDP** > **0**eBPF

3GPP

OmniUPF 3GPP

TS	TS	TS
TS 23.501	5G (5GS) (GPRS)	5G UPF (PGW-U)
TS 23.401	E-UTRAN (GPRS) (GTPv1-U)	LTE UPF (PGW-U)
TS 29.244	(PCF) (N4 PCF)	
TS 29.281	(GPRS) (GTPv1-U)	GTP-U
TS 23.503	5G (5GS) (QoS)	QoS
TS 29.212	(PCC) (QoS)	QoS

5G

- 3GPP -
- AMF - 5G
- CHF - 5G
- DN - IMS
- eNodeB - LTE
- F-TEID - IP GTP-U ID
- gNB - 5G
- GTP-U -
- MBR - QoS
- GBR - QoS
- N3 - RAN UPF
- N4 - SMF UPF PCF
- N6 - UPF
- N9 - UPF UPF
- PCF - 5G

- **PDU** - 5G
- **PGW-C** - PDN - LTE SMF
- **PGW-U** - PDN - LTE UPF
- **QFI** - QoS - 5G QoS
- **QoS** -
- **RAN** - gNB/eNodeB
- **SEID** - PCF ID
- **SMF** - 5G
- **TEID** - GTP-U ID
- **UE** -
- **UPF** - 5G

PCF

- **Association:** SMF - UPF
- **FAR:** -
- **IE:** - PCF
- **Node ID:** UPF - SMF (FQDN - IP)
- **PDR:** -
- **PCF:** - N4
- **QER:** QoS -
- **SDF:** -
- **Session:** UE PDU PDP PCF
- **URR:** -

eBPF - Linux

- **BPF:** -
- **eBPF:** BPF -
- **Hash Map:** eBPF
- **XDP:** eXpress -
- **Verifier:** eBPF
- **Map:** eBPF

- **Zero-copy:** 零拷贝技术

OmniUPF 架构

- **OmniUPF:** 基于 eBPF 的通用用户平面功能
- **Datapath:** 基于 eBPF 的数据平面
- **Control Plane:** PFCP 控制平面
- **REST API:** 基于 HTTP 的 API
- **Web UI:** 基于 Web 的界面

OmniUPF

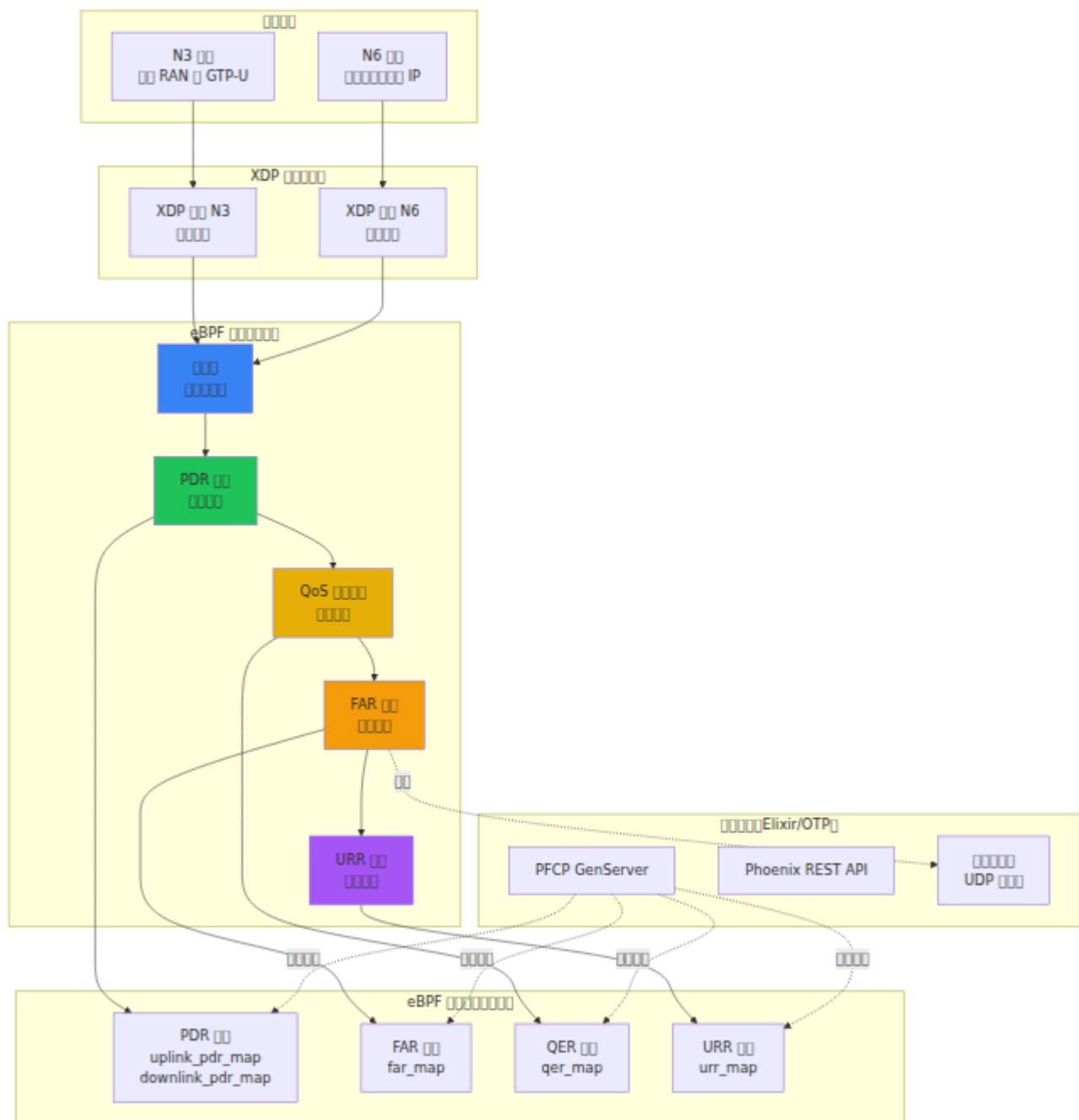
目次

1. 概要
2. eBPF 概要
3. XDP 概要
4. 環境構築
5. eBPF 概要
6. 概要
7. QoS 概要
8. 概要
9. 環境構築

概要

OmniUPF は eBPF/XDP を利用して 5G/LTE ネットワークを構築するための Elixir/OTP による GenServer を PFCP を利用して Linux 上で eBPF を実行する環境を構築するためのライブラリです。

eBPF を利用して `ipentrypoint_bpf.o` という C コードを `ebpf/xdp/` ディレクトリに NIF を `libbpf` を利用して Elixir で呼び出す。



□□□□□□

□□□□□□

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

優點

- eBPF 不需要 CPU 支援
- 簡單易用
- 靈活性/可擴展性

缺點

- XDP 需要 SmartNIC 支援
- 需要 XDP 支援
- 需要大量記憶體

eBPF 簡介

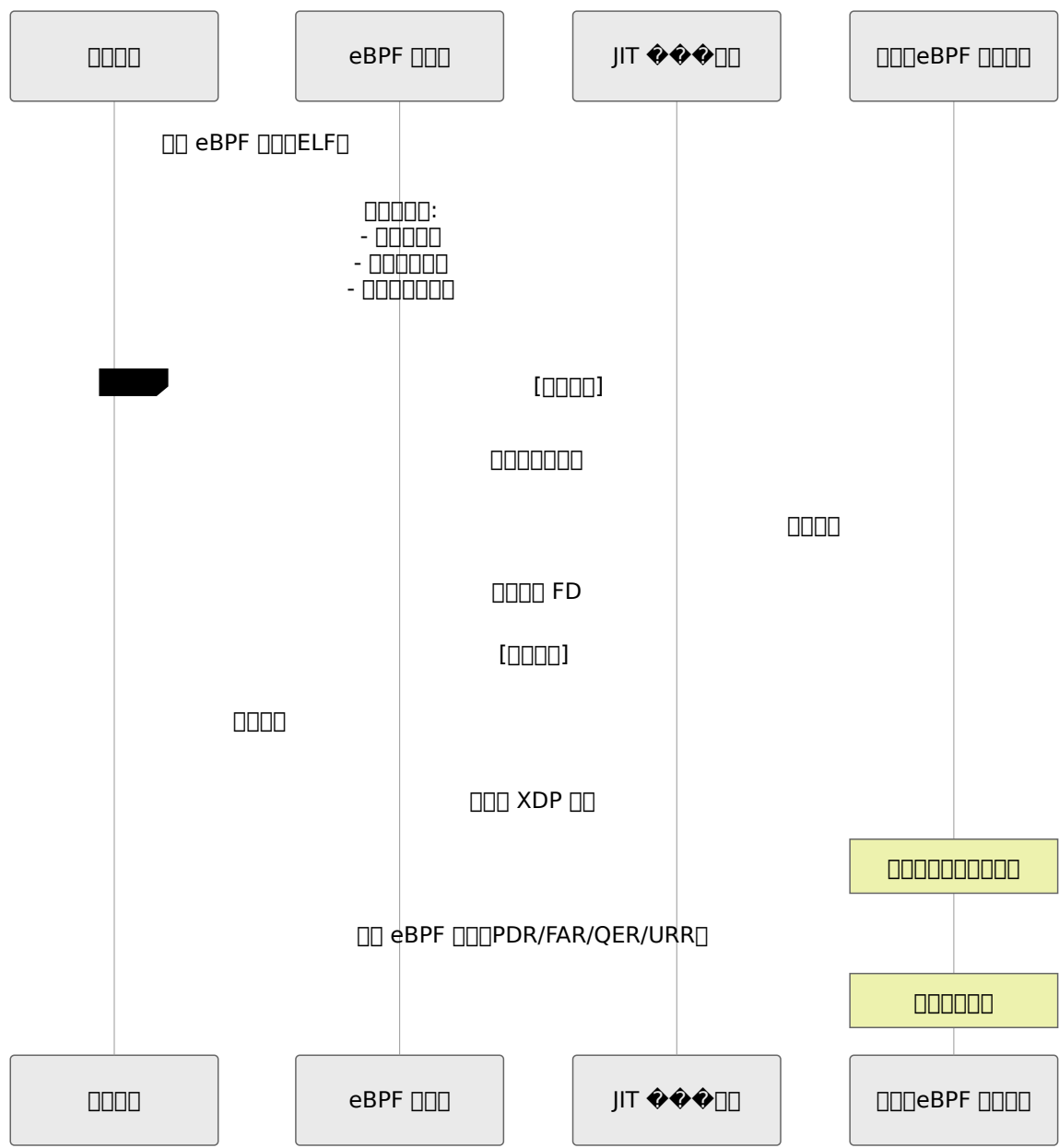
什麼是 eBPF

eBPF 是一個基於 Linux 的開源專案，它允許用戶在 Linux 系統上運行用戶定義的代碼，而無需修改系統代碼或安裝額外的驅動程序。

優點

- 簡單易用
- 靈活性/可擴展性
- 不需要 eBPF 支援
- 需要大量記憶體

eBPF 架构图



eBPF 用户空间

eBPF 用户空间 eBPF 用户空间

OmniUPF 用户空间

種類	項目	説明
BPF_MAP_TYPE_HASH	ハッシュマップ	TEID UE IP PDR
BPF_MAP_TYPE_ARRAY	配列マップ	ID QER FAR URR
BPF_MAP_TYPE_PERCPU_HASH	CPU別ハッシュマップ	PDR
BPF_MAP_TYPE_LRU_HASH	LRUハッシュマップ	

特徴

- $O(1)$ 検索
- 動的サイズ
- 動的型
- 動的型

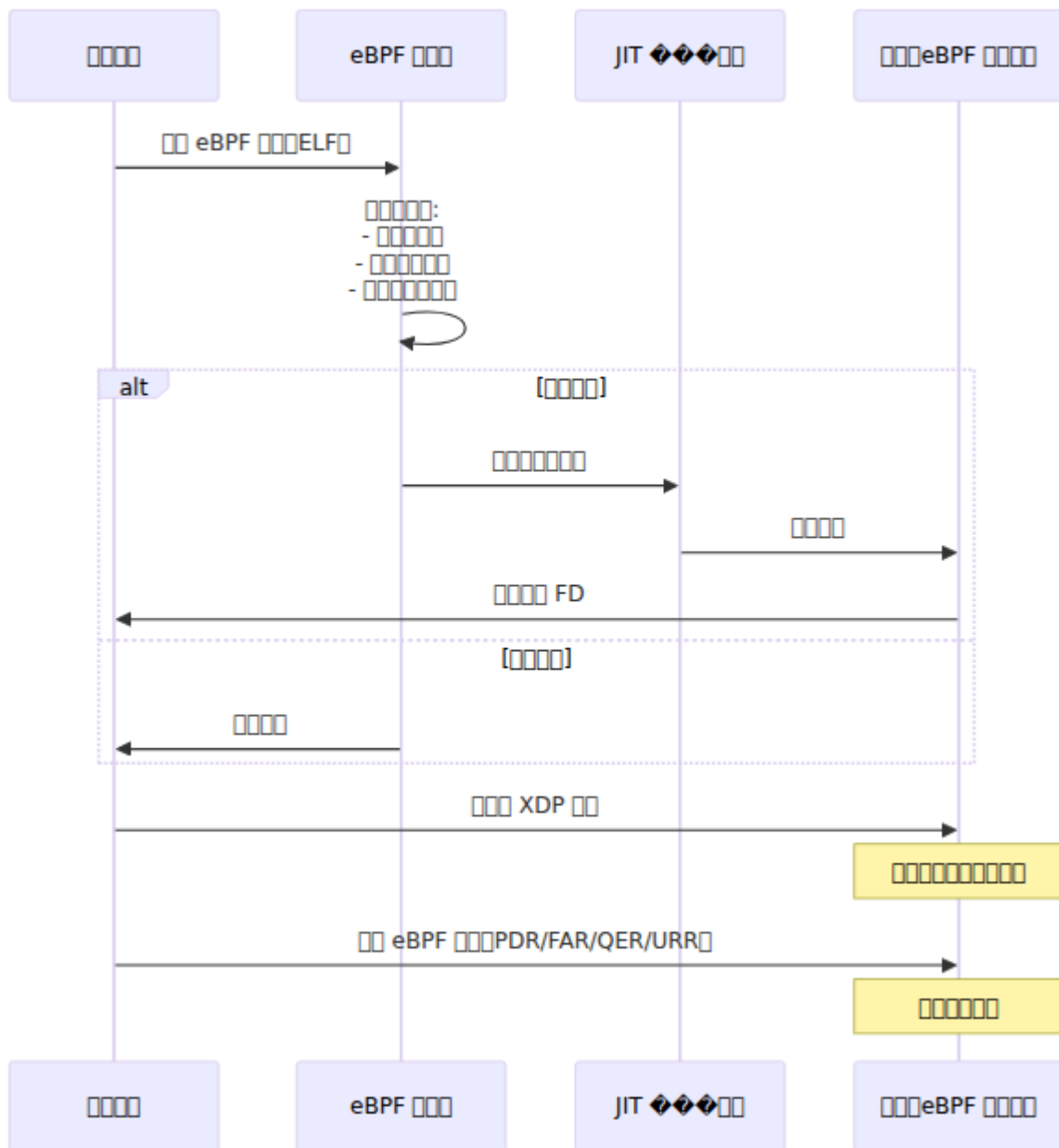
XDP

XDP

XDPはLinux eBPF

XDP

OmniUPF XDP



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.exs □□□

```
xdp_attach_mode = "native"
```

□□□

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

□□□□□□

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

3. XDP 简介

网络数据包处理

- eBPF 处理 SKB 数据
- XDP 处理数据
- 网络数据包处理
- 网络数据包处理

通过 `runtime.exe` 运行

```
xdp_attach_mode = "generic"
```

网络

- 网络
- 网络 SR-IOV 网络
- 网络
- 网络

网络 1-5 Gbps 网络/网络

XDP 简介

eBPF 通过 XDP 处理网络数据包

動作	処理	OmniUPF 動作
XDP_PASS	パケットをそのまま送る	パケットをそのまま ICMP 送る
XDP_DROP	パケットを落とす	パケットをそのまま落とす
XDP_TX	パケットをそのまま送信する	送信
XDP_REDIRECT	パケットをリダイレクトする	パケットを N3 ↔ N6
XDP_ABORTED	パケットを処理しきれずに落とす	eBPF 動作

パケットの処理

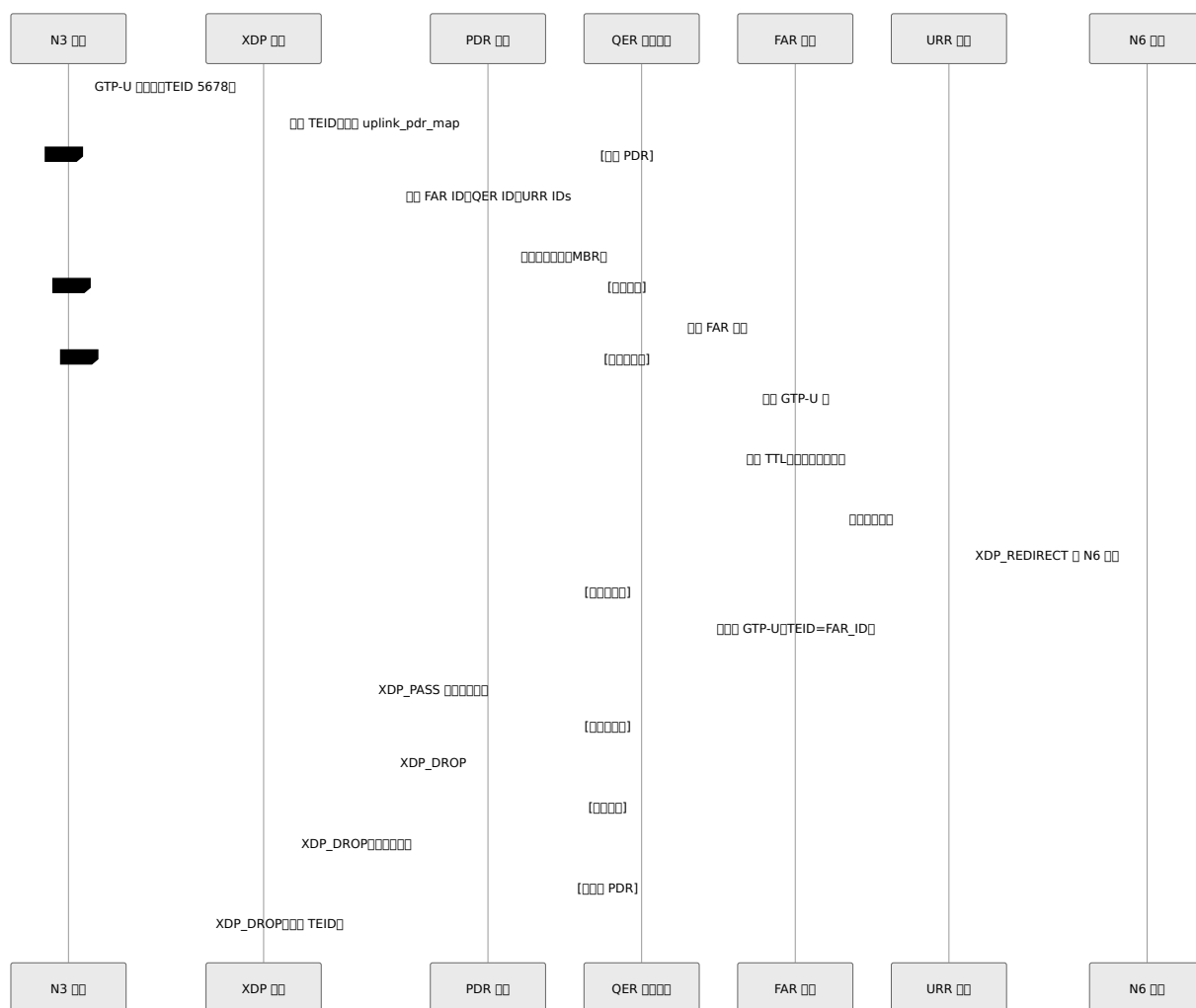
動作

OmniUPF は eBPF を利用してパケットを処理する



動作

- eBPF を利用して eBPF を
- パケットを処理する
- パケットを処理する
- 33 パケット



eBPF



OmniUPF `max_sessions`

PDR `100` = $2 \times \text{max_sessions}$ (`100` + `100`)
FAR `100` = $2 \times \text{max_sessions}$ (`100` + `100`)
QER `100` = $1 \times \text{max_sessions}$ (`100,000,000`)
URR `100` = $3 \times \text{max_sessions}$ (`100,000,000` URR)

`max_sessions` = 65,535

- PDR `100` 131,070
- FAR `100` 131,070
- QER `100` 65,535
- URR `100` 131,070

PDR `100` $3 \times 131,070 \times 212 \text{ B} = \sim 83 \text{ MB}$
FAR `100` $131,070 \times 20 \text{ B} = \sim 2.6 \text{ MB}$
QER `100` $65,535 \times 36 \text{ B} = \sim 2.3 \text{ MB}$
URR `100` $131,070 \times 20 \text{ B} = \sim 2.6 \text{ MB}$
`100` $\sim 91 \text{ MB}$

□□□□

□□□□

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

4. 填充 UDP 头

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

5. 填充 GTP-U 头

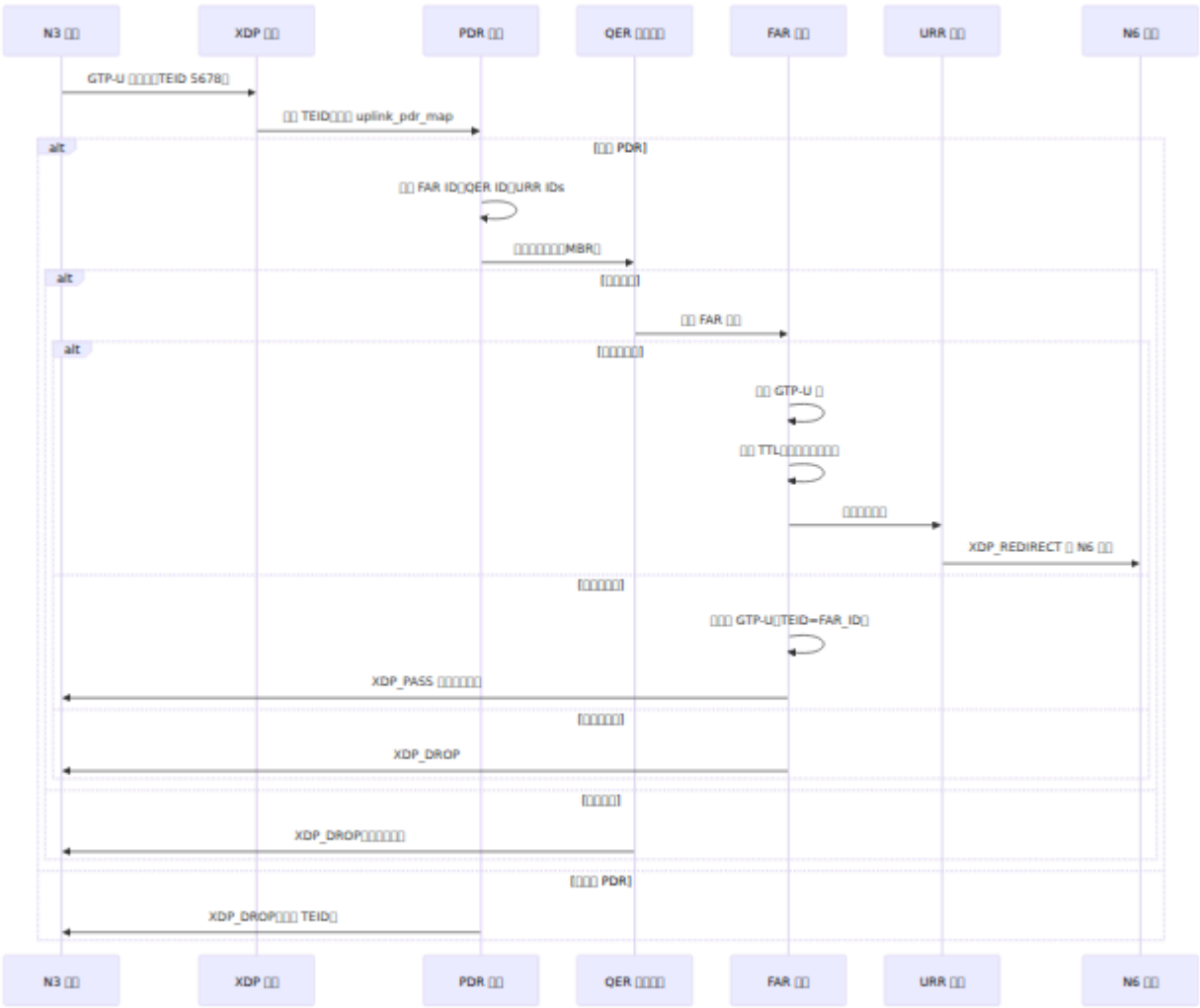
```
gtp->version = 1;
gtp->message_type = GTPU_G_PDU;
gtp->teid = htonl(far_id | (direction << 24)); // 填充 FAR ID 和方向
gtp->message_length = htons(orig_packet_len);
```

6. 填充 XDP_PASS

- 填充 UDP 源端口 22152
- 填充 UDP 目的端口

填充完成

SMF 向 FAR 发送 BUFFER 数据



Parameters

Field	Value	Description
FAR	10,000	FAR ID
QER	100,000	QER ID
TTL	30	TTL
UDP	22152	UDP port
TEID	60	TEID

QoS 問題

問題

OmniUPF 問題 問題 QoS 問題

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'

問題

問題

問題 qer.h 問題

```

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;

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

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

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

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

```

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

QoS 計算

100 MBR = 100 Mbps / 1500 = 1500

1. 計算

```
tx_time = 1500 * 8 * (1,000,000,000 ns/sec ÷ 100,000,000 bps)
tx_time = 1500 * 8 * 10 = 120,000 ns = 120 μs
```

2. 計算

- t=0 から t=120μs まで
- t=100μs まで
- t=150μs まで

3. 計算

```
Max PPS = (100 Mbps ÷ 8) ÷ 1500 = 8,333 PPS/μs
遅延 = 120 μs
```

計算

計算

項目	単位	単位/秒	単位
XDP SmartNIC	100 Gbps	148 Mpps	< 1 μs
XDP 10G NIC	10 Gbps	8 Mpps	2-5 μs
XDP 10G NIC 4	40 Gbps	32 Mpps	2-5 μs
XDP	1-5 Gbps	0.8-4 Mpps	50-100 μs

遅延

ネットワーク遅延XDP遅延

項目	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 sXDP遅延10G NIC

CPU遅延

遅延

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

遅延 CPU遅延

$$\text{CPU \%} \approx (\text{パケット数} / 10,000,000) \times 100\% \text{ パケット}$$

パケット2 Mpps パケット ~20% パケット

パケット

eBPF パケット

- パケット~100 nsパケット
- パケット~300 nsパケット
- パケット~50 nsパケット

パケット

$$\text{パケット} = \text{パケット} \times (\text{パケット} + \text{パケット} \times 64 \text{ パケット})$$

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

パケット

パケット

パケット **UPF** パケット

Setting SMF as parent of SMF would create a cycle

パケット

パケット

- SMF パケット UPF パケット
- パケット UPF パケット UE パケット
- パケット UPF パケット

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

# IRQ 前提条件 XDP 前提条件
systemctl stop irqbalance

# CPU 前提条件
cpupower frequency-set -g performance

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

前提条件

前提条件

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

前提条件100 前提条件20 Gbps前提条件

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

--	--	--	--

- [illegible]

OmniUPF

概要

- 概要
- インストール
- XDP
- 設定
- 起動
- 確認
- NIC
- ログ
- 再起動

インストール

OmniUPF は、4G (EPC) と 5G のネットワークを
Elixir/OTP で構築された Elixir アプリケーション (runtime.exs) として

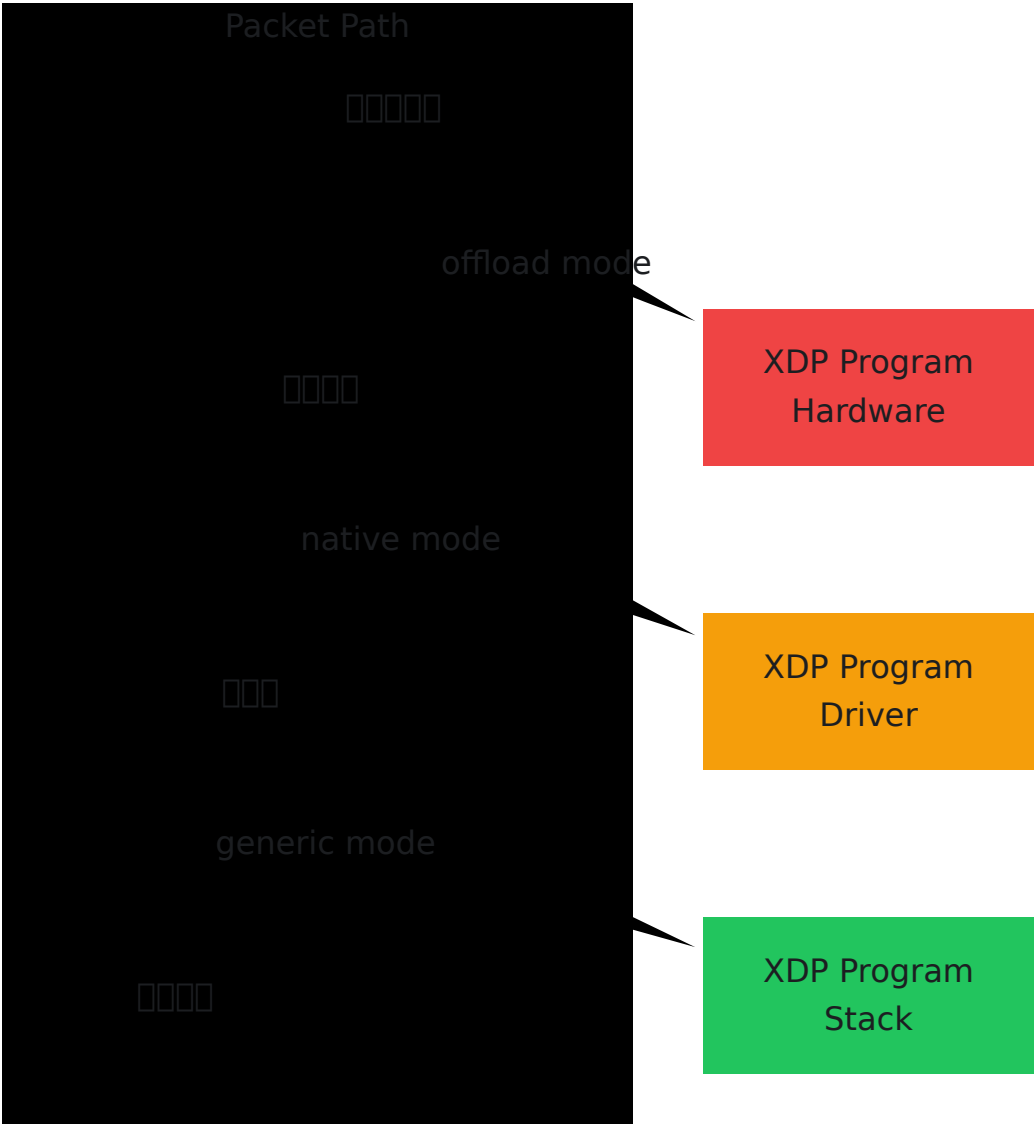
提供されています。 /etc/omniupf/runtime.exs を参照してください。 UPF

起動

OmniUPF を起動するには、

网络性能

网络接口	网络接口	性能	网络接口	NIC 性能
网络接口	网络接口	~1-2 Mpps	网络接口	网络 NIC
网络接口	网络接口	~5-10 Mpps	网络接口 SR-IOV 网络接口	网络 XDP 网络
网络接口	NIC 网络 SmartNIC	~10-40 Mpps	网络接口	网络 XDP 网络 SmartNIC



概要

XDP 概要

概要

- XDP の種類
- XDP の仕組み
- XDP の実装
- XDP の利用方法

概要

- XDP の性能 ~1-2 Mpps
- XDP の XDP 実装

概要 runtime.exs 概要

```
xdp_attach_mode = "generic"
```

概要

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

概要

XDP 概要

概要

- XDP の性能 ~5-10 Mpps
- XDP の仕組み
- XDP の実装

- 直接 SR-IOV 方式

特徴

- 直接 XDP 方式
- 直接 NIC/パケット XDP

実行 runtime.exs 実行

```
xdp_attach_mode = "native"
```

特徴

- 直接パケット
- 直接 SR-IOV 方式
- 直接 XDP 方式 NIC Intel Mellanox 方式

特徴

- 直接 XDP 方式 NIC 方式
- Linux 5.15+ 直接 XDP 方式

直接パケット方式

直接 XDP 方式 SmartNIC 方式

特徴

- 直接 ~10-40 Mpps
- 直接 CPU 方式
- 直接
- 直接 CPU 方式

特徴

- 直接 SmartNIC 方式
- SmartNIC 方式

- 配置

配置 `runtime.exe` 配置

```
xdp_attach_mode = "offload"
```

配置

- 配置
- 配置
- CPU 配置

配置

- 配置 XDP 配置 SmartNIC  Netronome Agilio CX  Mellanox BlueField
 - 配置
-

配置

配置

属性	配置	类型	默认值
<code>xdp_interfaces</code>	N3/N6/N9 网络接口名称 "auto" 表示自动	字符串	"auto"
<code>gtpu_interface</code>	指定 GTP-U 网络接口名称 <code>sgi_interface</code> 指定 GTP-U 网络接口名称 XDP 网络接口名称	字符串	nil
<code>sgi_interface</code>	指定 IP SGi/N6 网络接口名称 <code>gtpu_interface</code> 指定 GTP-U 网络接口名称 XDP 网络接口名称	字符串	nil
<code>n3_address</code>	N3 网络 IPv4 地址 RAN 网络 GTP-U	IP	"127.0.0.1"
<code>n9_address</code>	N9 网络 IPv4 地址 UPF 网络 UPF 网络 ULCL	IP	<code>n3_address</code> 网络

配置 runtime.exs 文件

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

配置 NIC 网络接口名称 GTP-U 网络 SGi 网络接口名称 NIC 网络接口名称 NIC 网络接口名称
XDP 网络接口名称 `gtpu_interface` 指定 GTP-U 网络接口名称 `sgi_interface` 指定 GTP-U 网络接口名称“on-a-
stick”网络接口名称 `xdp_interfaces` 指定 XDP 网络接口名称

```
xdp_interfaces = "eth1,eth2"
gtpu_interface = "eth1"    #  GTP-U N3/N9/S5-S8
sgi_interface  = "eth2"    #  IP SGi/N6
```

PFCP

pfcip_address	PFCP N4/Sxb/Sxc	IP	"127.0.0.1"
pfcip_port	PFCP		8805
node_id	PFCP ID	IP	"127.0.0.1"
heartbeat_interval_ms	PFCP		5000
heartbeat_timeout_ms	PFCP		5000
heartbeat_retries			3
association_setup_timeout	UPF PFCP		5000
pfcip_remote_nodes	UPF IP UPF		[]

runtime.exe


```
pfcip_address = "10.100.50.241"
pfcip_port = 8805
node_id = "10.100.50.241"
heartbeat_interval_ms = 10_000
heartbeat_retries = 5
association_setup_timeout = 5_000
pfcip_remote_nodes = [] # 节点 ["10.100.50.20",
"10.100.50.22"]
```

API 配置

属性	描述	类型	默认值
api_port	REST API 接口Phoenix接口	整数	8080
log_level	日志级别: debug :info :warning :error	字符串	:info

配置 runtime.exs 文件

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

GTP 配置

属性	描述	类型	默认值
gtpu_port	GTP-U 端口	整数	2152
gtp_echo_interval	回声间隔	整数	10
gtp_echo_retries	回声重试次数	整数	3
gtp_peers	GTP 对等体	列表	[]

```
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 ☐☐☐☐

項目	説明	単位	値
max_sessions	セッション数	個	1_000_000
far_map_size	FAR eBPF マップサイズ	バイト	131_070
qer_map_size	QER eBPF マップサイズ	バイト	65_535
urr_map_size	URR eBPF マップサイズ	バイト	65_535

```
runtime.exs
```

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

--	--	--	--	--

[illegible]

변수명	설명	타입	값
buffer_port	UDP eBPF 필터 수신 포트	int	22152
buffer_max_total	전체 FAR 버퍼 크기 (바이트)	int	10_000
buffer_max_per_far	단일 FAR 항목당 최대 버퍼 크기 (바이트)	int	100
buffer_ttl_ms	버퍼 항목의 유효 기간 (밀리초)	int	30_000
buffer_cleanup_interval_ms	버퍼 정리 간격 (밀리초)	int	5_000

이 값을 runtime.exs에 설정

```
buffer_port = 22152
buffer_max_total = 10_000
buffer_max_per_far = 100
buffer_ttl_ms = 30_000
buffer_cleanup_interval_ms = 5_000
```

이러한 설정은 upf_buffer_packets_dropped_total, global_limit, far_limit, expired와 같은 다른 변수들과 함께 사용됩니다.

設定項目

項目名	説明	単位	初期値
feature_ueip	OmniUPF の UE IP 機能の有効化	ブール値	false
ueip_pool	UE IP 割り当てに使用する IP 範囲 (feature_ueip が有効の場合)	CIDR 形式	"10.60.0.0/24"
feature_ftup	OmniUPF の F-TEID 機能の有効化	ブール値	true
teid_pool_start	TEID 割り当ての開始値	整数	1
teid_pool_end	TEID 割り当ての終了値	整数	1_000_000

UE IP 機能の有効化 (runtime.exs) の例

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

F-TEID 機能の有効化 (runtime.exs) の例

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

その他の機能

FRR (Free Range Routing) のインストールと UE の接続テスト (確認)

項目	説明	単位	初期値
route_manager_enabled	UE 管理機能の有効化	bool	true
route_manager_type	管理機能の種類 ("frr" または "static")	string	"frr"
route_manager_nexthop	UE のネクストホップ = 空	string	""
route_manager_vtysh_path	FRR vtysh のパス (route_manager_type が "frr" の場合)	string	"/usr/bin/vtysh"
route_manager_sync_interval_ms	同期間隔 (ミリ秒)	int	600_000
route_batch_interval_ms	バッチ間隔 (ミリ秒)	int	1_000

runtime.exs を編集

```
route_manager_enabled = true
route_manager_type = "frr"
route_manager_nexthop = ""
route_manager_vtysh_path = "/usr/bin/vtysh"
route_manager_sync_interval_ms = 600_000
route_batch_interval_ms = 1_000
```

確認

- UPF 設定

- 可 OSPF 可 BGP 可配置
- 可配置 FRRouting 可

eBPF / XDP 可

可	可	可	可
xdp_interfaces	可 XDP 可配置 "auto" 可配置	可 可 可	"auto"
xdp_attach_mode	XDP 可 可 "generic" 可 "native" 可 "offload"	可 可 可	"generic"
ebpf_pin_path	BPF 可配置	可 可 可	"/sys/fs/bpf/upf_pipeli
xdp_obj_path	可 eBPF 可配置	可 可 可	"/etc/omniupf/ipentrypc
ebpf_backend	可配置 :nif 可 NIF 可配置 :port 可配置 可 :mock 可配置 eBPF 可配置	可 可	:nif

可 runtime.exe 可

```
xdp_interfaces = "auto" # [] "eth0,eth1" [] [] [] []
xdp_attach_mode = "native"
ebpf_pin_path = "/sys/fs/bpf/upf_pipeline"
xdp_obj_path = "/etc/omniupf/ipentrypoint_bpf.o"
ebpf_backend = :nif
```

--	--	--	--

Elixir 0000 (runtime.exs)

```
❏❏: /etc/omniupf/runtime.exs
```

```

Elixir 0.15.4
config :upf_ex, ...

```

```

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
gtpu_port = 2152                     # GTP-U 2152
buffer_port = 22152                  #

#
=====
# eBPF / XDP
#
=====
xdp_interfaces = "auto"              # "auto"
xdp_attach_mode = "native"           # XDP "generic" "native" "c
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

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

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

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

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

#
=====

```

```
# HTTP API 配置
```

```
#
```

```
=====
```

```
api_port = 8080
```

```
log_level = :info
```

```
#
```

```
=====
```

```
# 配置项
```

```
#
```

```
=====
```

```
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

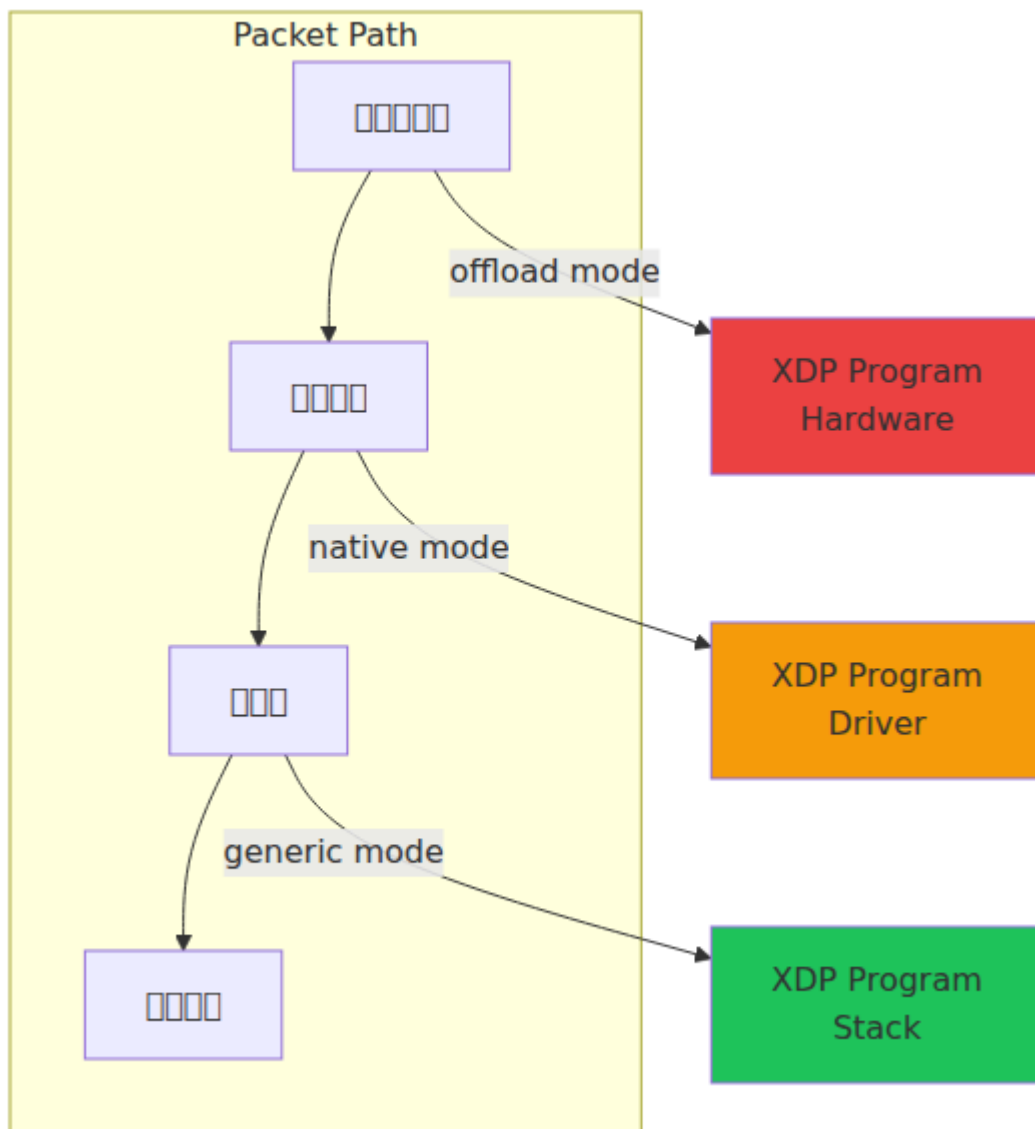
# 手动启动/停止/守护进程
sudo /opt/omniupf/bin/upf_ex start
sudo /opt/omniupf/bin/upf_ex stop
sudo /opt/omniupf/bin/upf_ex daemon # 守护进程
```

安装与配置

安装

OmniUPF 是一个基于 Linux 的 XDP 防火墙，用于保护网络基础设施。

在 **Proxmox** 虚拟机中安装 **XDP** 防火墙，需要配置 **XDP** 规则。



Proxmox VE

网卡

1. 安装 XDP

安装依赖

配置

- 网卡 VirtIO 或 E1000
- XDP 模式 generic
- 速率 ~1-2 Mpps

Proxmox 環境構築

```
iface net0
    VirtIO
vmbr0
```

OmniUPF 環境構築 runtime.exs 設定

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

2. SR-IOV 環境構築 XDP

環境構築

環境

- SR-IOV 環境構築
- XDP 環境構築 native
- 環境構築 ~5-10 Mpps

環境

- SR-IOV 環境構築 NIC 環境構築 Intel X710 Mellanox ConnectX-5
- BIOS 環境構築 SR-IOV
- IOMMU 環境構築 GRUB 環境構築 intel_iommu=on 環境構築 amd_iommu=on

Proxmox 環境構築 SR-IOV

```
# 编辑 GRUB 配置
nano /etc/default/grub

# 添加 GRUB_CMDLINE_LINUX_DEFAULT
intel_iommu=on iommu=pt

# 更新 GRUB 配置
update-grub
reboot

# 为 NIC 添加 VFs 并设置 eth0 为 4 个 VFs
echo 4 > /sys/class/net/eth0/device/sriov_numvfs

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

Proxmox 配置

```
网卡 -> 桥 -> PCI 桥
启用 SR-IOV 支持
配置 VFs
禁用 GPU 直通
PCI-Express 直通
```

OmniUPF 配置 `runtime.exs`

```
xdp_interfaces = "ens1f0"      # SR-IOV VF 接口
xdp_attach_mode = "native"
```

3. PCI 直通 XDP

配置 NIC 直通

配置

- 配置 NIC 直通

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

Proxmox

```

-> -> PCI
NIC 0000:01:00.0
GPU
PCI-Express

```

OmniUPF `runtime.exs`

```

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

```

KVM/QEMU

```

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

```

SR-IOV

```

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

```

VMware ESXi

vSwitch XDP

- VMXNET3
- XDP generic

SR-IOV XDP

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

Microsoft Hyper-V

XDP

-
- XDP generic

SR-IOV XDP

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

VirtualBox

NAT/XDP

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

NIC ☐☐☐

□□ **Mpps** □□□□

Mpps - **Gbps**

VoIP

□ □ □ □ □ □ □ □ □ □ □ □

□□□□□□UPF □ N3 □□□□□ GTP-U □□□□□□□□ N6 □□□□□□ IP □□□□

GTP-U **N3**

- IPv4 20
- UDP 8
- GTP-U 8
- GTP-U 36

□□ **GTP-U** □□□□**N3**□□

- **IP** 20 個 IPv4
- **UDP** 8 個
- 1 個
- 29 個
- **GTP-U** 36 個
- 65 個

1 Mpps GTP-U

$$65 \text{ } \mu\text{s} \times 1,000,000 \text{ pps} \times 8 \text{ b}/\mu\text{s} = 520 \text{ Mbps}$$

□□ GTP-U □□□□N3□1500 MTU□□

- **IP MTU** 1500 bytes IP packet
- **GTP-U** 36 bytes
- 1536 bytes

1 Mpps の GTP-U

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

IP の N6

N6 の IP の GTP-U

N6

- IP 20 字节
- UDP 8 字节
- 1 字节
- 29 字节

1 Mpps の N6

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

N6 の 1500 MTU

- IP MTU 1500 字节
- 1500 字节

1 Mpps の N6

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

Intel X710 NIC の 10 Mpps

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

値 N6 標準値 IP値 GTP-U値

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

値 10 Mpps 標準値標準値1200 標準値標準値 N3 値 N6 標準値標準値 ~10 Gbps**

標準値

値 1200 標準値標準値標準値標準値標準値標準値

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 accelerate packet processing 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	模式	性能
Mellanox ConnectX-5	mlx5	是	内核	~12 Mpps
Mellanox ConnectX-6	mlx5	是	内核	~20 Mpps
Mellanox BlueField	mlx5	是	内核 + 用户态	~40 Mpps
Mellanox ConnectX-4	mlx4	否	内核	~2 Mpps

Broadcom NICs

网卡	驱动	XDP 支持	模式	性能
Broadcom BCM57xxx	bnxt_en	是	内核	~8 Mpps
Broadcom NetXtreme II	bnx2x	否	用户态	~1 Mpps

其他网卡

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

如何 NIC XDP 如何

如何如何如何 XDP 如何

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

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

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

如何 XDP 如何如何

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

# 如何如何XDP 如何如何
# 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如何如何

OS	NIC	OS Kernel	Mpps	Network N3	Notes
Linux	NIC VirtIO E1000	Linux	1-2 Mpps	1-2 Gbps	Default configuration
Linux	Intel X710 Mellanox CX-5	Linux	5-10 Mpps	5-10 Gbps	Requires SR-IOV
Linux	Intel E810 Mellanox CX-6	Linux	10-20 Mpps	10-20 Gbps	Requires SR-IOV
Linux	Mellanox CX-6 Intel E810	Linux	20-40 Mpps	20-40 Gbps	Requires SR-IOV
Linux	Mellanox BlueField Netronome Agilio	Linux	40+ Mpps	40+ Gbps	Requires SR-IOV
Proxmox VM	VirtIO	Proxmox	1-2 Mpps	1-2 Gbps	Default configuration
Proxmox VM	Intel X710/E810 VF Mellanox CX-5 VF	Proxmox	5-10 Mpps	5-10 Gbps	Requires SR-IOV

Network

OS XDP

- XDP
- OS XDP

NIC

- Cilium XDP
- IO Visor XDP

環境構築

ステップ 1 ネットワーク環境の構築

ネットワーク環境を構築し、OmniUPFをSR-IOV

```
# 設定ファイル (/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
```

ステップ 2 仮想ネットワークの構築

仮想ネットワークを構築し、Intel X710 NIC を利用して UPF


```
# 配置文件 (/etc/omniupf/runtime.exs)
xdp_interfaces = "ens1f0,ens1f1"      # N3 的 ens1f0 和 N6 的 ens1f1
xdp_attach_mode = "native"
api_port = 8080
pfcp_address = "10.100.50.241"
pfcp_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"
```

3 Proxmox VM 的 SR-IOV 配置

在 Proxmox VM 中配置 SR-IOV 和 UPF

```
# Proxmox SR-IOV 配置 (/etc/omniupf/runtime.exs)
xdp_interfaces = "ens1f0"          # SR-IOV VF
xdp_attach_mode = "native"
api_port = 8080
pfcp_address = "192.168.100.10"
pfcp_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},
]
```

配置 4G PGW-U 和 4G EPC

配置 OmniUPF 的 4G EPC 和 PGW-U

```
# PGW-U 配置 (/etc/omniupf/runtime.exs)
xdp_interfaces = "eth0"
xdp_attach_mode = "native"
api_port = 8080
pfcp_address = "10.200.1.10"
pfcp_port = 8805
node_id = "10.200.1.10"
n3_address = "10.200.1.10"          # S5/S8 接口 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
```

5 OmniUPF + PGW-U

OmniUPF 5G 4G

```
# (/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"
```

6 SmartNIC

Netronome Agilio CX SmartNIC

```
# SmartNIC 設定 (/etc/omniupf/runtime.exs)
xdp_interfaces = "enpls0np0"          # SmartNIC 設定
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
```

設定ファイル

設定ファイル

設定 max_sessions 1000000

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

設定ファイル~91 MB 100K

□□□□

□□□□□□□□

```
□□□□□ = 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 □□□□

□□□□□□□□□□□□□□ 2 □□□□□□□□

```
# 設定ファイル
ulimit -l

# eBPF 用設定
ulimit -l unlimited
```

機能

- **機能** - eBPF/XDP によるパケット処理
- **機能** - PDR、FAR、QER、URR の管理
- **機能** - トラフィックの計測
- **機能** - Prometheus による監視
- **Web UI** - 管理画面
- **機能** - UPF によるパケット処理
- **機能** - トラフィックの整形

IPv6 /

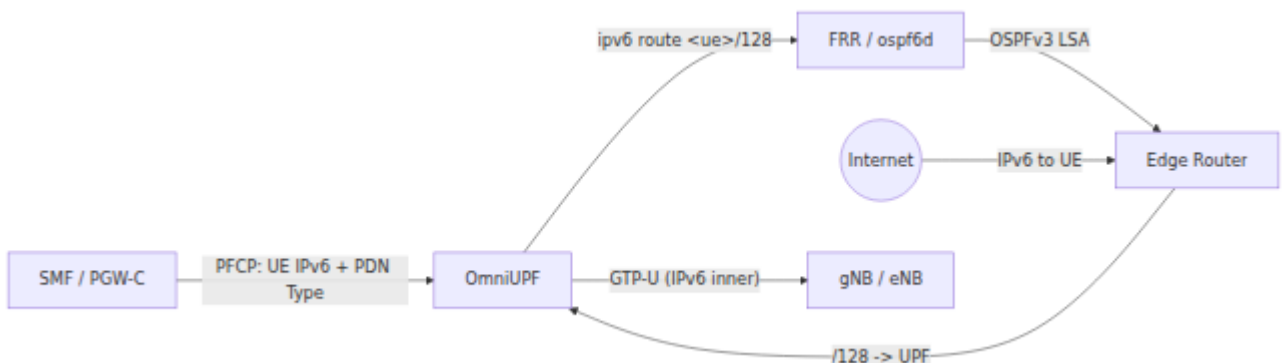
OmniUPF IPv6 PDN UE PCP IPv6 OSPFv3 UE

PDN PAA Gx PCP UPF UE IPv6 SMF/PGW-C

-
-
- UE IPv6
- OSPFv3
-
-

IPv6 IPv4v6 PDN SMF/PGW-C PCP UE IPv6 PDN UPF

- IPv6 UE
- UE /128 OSPFv3 UPF
- GTP-U IPv6



IPv6

IPv6 sysctl drop-in /etc/sysctl.d/

sysctl	Value	Description
net.ipv6.conf.all.disable_ipv6	0	Disable IPv6 on all interfaces
net.ipv6.conf.default.disable_ipv6	0	Disable IPv6 on new interfaces
net.ipv6.conf.all.forwarding	1	Enable IPv6 forwarding on all interfaces

IPv6 OSPFv3

UE IPv6

UPF UE SMF runtime.exs IPv6 SMF/PGW-C PFCP UE

```
# /etc/omniupf/runtime.exs
config :upf_ex,
  ueip_pool: "2001:db8:6f85:70::/64"
```

Parameter	Value	Unit	Default	Description
ueip_pool	IPv6 CIDR	1	-	UPF UE IP IPv4 IPv6 CIDR

OSPFv3 配置

配置 UE IPv6 地址 /128 配置 FRR OSPFv3 配置 IPv4 /32 配置 FRR OSPFv3

```
# /etc/omniupf/runtime.exs
config :upf_ex,
  route_manager_enabled: true,
  route_manager_type: "frr"
```

配置项	配置值	配置说明	配置项
route_manager_enabled	false	配置 UE 是否启用 OSPFv3 true 启用 UE 配置	
route_manager_type	-	配置 FRR/vtysh 配置 ip/ipv6 配置 "frr" 配置 OSPFv3 配置	

FRR 配置 OSPFv3 (ospf6d) 配置 UE /128 配置 IPv6 配置 ipv6 route <addr>/128 配置 - IPv4 ip route 配置 IPv6 配置

-
-
-

-
-
-

##

□□□□

- □□□□ IPv6□□□□□□
- □□□□□□□□ IPv6 □□

□□□□

1. □□ □□□□□□ disable_ipv6=0□ forwarding=1□□
2. □□□□□□□□□□□□□□ IPv6 □□

API

OmniUPF 的 `/api/metrics` 接口与 Prometheus 的 HTTP API 类似
`api_port` 默认为 `8080`，接口为 `/api` - 该 API 用于暴露指标

OmniUPF 的接口

接口

- PFCP** 接口 - 用于暴露 PFCP 相关指标
- 接口 - 用于暴露 / 接口 / 接口
- 接口 - 用于暴露 TEID 接口
- 接口 (**DLDR**) 接口 - PFCP 接口 FAR 接口
- GTP-U** 接口 - 用于暴露 GTP-U 接口
- URR** 接口 - 用于暴露 URR 接口
- eBPF** 接口 - 用于暴露 eBPF 接口
- 接口 - 接口 (N3/N6) `_total` 接口 `expose_xdp_metrics` 接口 (#57) 接口
接口 `/api/v1/xdp_stats` 接口
- 接口 / 接口 - TEID/FAR-ID 接口 UE 接口
- 接口 - 接口 DNS 接口

PFCP 接口

UPF 的 PFCP 接口 (SMF / PGW-C / SGW-C) 接口

名称	类型	单位	描述
<code>upf_pfcg_associations</code>	Gauge	none	统计 PFCP 会话数
<code>upf_pfcg_sessions</code>	Gauge	none	统计 PFCP 会话数
<code>upf_pfcg_peer_restarts_total</code>	Counter	peer	统计 PFCP 对等体重启次数
<code>upf_pfcg_duplicate_requests_total</code>	Counter	none	统计 PFCP 重复请求次数
<code>upf_orphaned_sessions_deleted_total</code>	Counter	none	统计删除的孤儿会话数
<code>upf_pfcg_messages_rx_total</code>	Counter	type, peer	统计接收到的 PFCP 消息数
<code>upf_pfcg_messages_tx_total</code>	Counter	type, peer	统计发送的 PFCP 消息数
<code>upf_pfcg_errors_total</code>	Counter	type	统计 PFCP 错误数 session_content mandatory_ie

Table 1

Counter	Unit	Reset
upf_sessions_created_total	Counter	none
upf_sessions_deleted_total	Counter	none
upf_sessions_modified_total	Counter	none
upf_session_establishment_duration_microseconds	Histogram	none

Table 2

eBPF program that filters UDP traffic on port 22152 and logs the source IP address of the UE. The program is loaded on the UE and filters traffic on the G-PDU and GTP-U ports. The program is loaded on the UE and filters traffic on the G-PDU and GTP-U ports.

buffered/flushed/current 00000000000000000000 upf_walled_garden_* 0000
00 TEID 0000 upf_gtpu_error_indications_sent_total0

00000	00	00	00
upf_userspace_punt_received_total	Counter	none	0000000000 0000000000 0 + 0000 + 00 TEID0
upf_buffer_packets_buffered_total	Counter	none	0000000000 0000
upf_buffer_packets_flushed_total	Counter	none	0000000000 00
upf_buffer_packets_dropped_total	Counter	reason	00000000/00 00000000
upf_buffer_packets_current	Gauge	none	0000000000 0000
upf_buffer_flush_operations_total	Counter	none	0❖❖000000 0000000 FAR 0000
upf_buffer_flush_packets_sent_total	Counter	none	0000000000 0000
upf_buffer_flush_errors_total	Counter	reason	0000000000 0

upf_buffer_packets_dropped_total reason 00000000000000000000000000000000

- global_limit - 0000000000 (buffer_max_total)
- far_limit - 0000 FAR 00000000 (buffer_max_per_far)
- expired - 000000 TTL (buffer_ttl_ms) 00

- `read_error` - 读取失败
- `too_small` - 接收到的 GTP 包太小
- `invalid_gtp_type` - 无效的 G-PDU GTP 类型
- `not_buffering_far` - 远端地址不在缓冲池中
- `error_indication_failed` - 错误指示 TEID 无效的 GTP-U 包

该 TEID 的 GTP-U 包错误指示总数
`upf_gtpu_error_indications_sent_total`

`upf_buffer_flush_errors_total` `reason`

- `far_lookup_failed` - 远端地址查找失败
- `no_forw_action` - 远端地址没有转发动作
- `connection_failed` - 连接失败

丢包率 (DLDR) 统计

该统计项用于统计 PFCP 会话的丢包率 (DLDR) 和 SMF/PGW-C 的 UE 数

统计项	类型	单位	描述
<code>upf_dldr_sent_total</code>	Counter	peer	发送给 SMF/PGW-C 的包数
<code>upf_dldr_errors_total</code>	Counter	none	DLDR 错误总数
<code>upf_dldr_skipped_total</code>	Counter	none	被 FAR 跳过的 DLDR 包数
<code>upf_far_index_size</code>	Gauge	none	DLDR 索引大小 (FAR 索引)

GTP-U 统计

GTP-U 包错误指示总数 TEID 无效的 GTP-U 包

URR ()

upf_urr_reports_sent_total	Counter	trigger	URR
upf_urr_report_errors_total	Counter	none	URR
upf_urr_active_timers	Gauge	none	URR

trigger URR - periodic (PERIO), volume_threshold (VOLTH), time_threshold (TIMTH), quota_validity (QUVTI)

URR Prometheus REST API /api/v1/urr_map URR

eBPF

upf_ebpf_far_entries	Gauge	none	eBPF FAR
upf_ebpf_pdr_uplink_entries	Gauge	none	eBPF PDR TEID PDR
upf_ebpf_pdr_downlink_entries	Gauge	none	eBPF PDR UE IP PDR

upf_ebpf_far_entries FAR-id (upf_far_ids_allocated)far_map FAR id PDR O()

性能指標 (N3/N6)

性能指標 (N3 = GTP-U トラフィック gNB/eNB N6 = S-GW トラフィック) を eBPF upf_ext_stat を CPU 上で

項目	単位	タイプ	説明
upf_n3_rx_packets_total / upf_n3_tx_packets_total	Counter	none	N3 (GTP-U) パケット XDP 処理
upf_n6_rx_packets_total / upf_n6_tx_packets_total	Counter	none	N6 (SGi) パケット XDP 処理
upf_n3_rx_bytes_total / upf_n3_tx_bytes_total	Counter	none	N3 (GTP-U) バイト
upf_n6_rx_bytes_total / upf_n6_tx_bytes_total	Counter	none	N6 (SGi) バイト

expose_xdp_metrics を #57 を upf_ext_stat Prometheus virtio_net XDP TX-ring wedge Sampler config :upf_ex, expose_xdp_metrics: true

_total upf_ext_stat omniupf eBPF .o rate() rate()

JSON GET /api/v1/xdp_stats eBPF rx_n3 / tx_n3 / rx_n6 / tx_n6 rx_n3_bytes / tx_n3_bytes / rx_n6_bytes / tx_n6_bytes

00/0000

0000	00	00	00
upf_routes_total	Gauge	none	0000000000 UE IP 00
upf_teid_allocated	Gauge	none	00000000 TEID
upf_far_ids_allocated	Gauge	none	00000000 FAR ID

000000

000000/00000000000000000000 000000

名称	タイプ	タグ	単位
upf_walled_garden_active_redirects	Gauge	none	個 秒 個 秒
upf_walled_garden_packets_intercepted_total	Counter	none	個 個 個 個
upf_walled_garden_packets_dropped_total	Counter	none	個 個 個 個
upf_walled_garden_packets_forwarded_total	Counter	dst_ip	個 IP 個 個 個 個
upf_walled_garden_bytes_forwarded_total	Counter	dst_ip	個 IP 個 個 個 個
upf_walled_garden_dns_spoofed_total	Counter	domain	個 個 個 DNS 個
upf_walled_garden_dns_forwarded_total	Counter	domain	個 個

Grafana 面板

面板名称

- 面板名称 (upf_pfcps_sessions, upf_pfcps_associations)
- 面板名称 N3/N6 面板名称 /api/v1/xdp_stats 面板名称 #57
- 面板名称 (upf_buffer_packets_current, 面板名称 reason 面板名称)
- DLDR / 面板名称 (rate(upf_dldr_sent_total[5m]))
- 面板名称

面板名称

- 面板名称 - 面板名称
- 面板名称 - 面板名称 api_port 面板名称 UPF 面板名称
- **Web UI** 面板名称 - 面板名称
- 面板名称 - eBPF 面板名称
- 面板名称 - 面板名称 PDR/FAR/QER/URR 面板名称
- 面板名称 - 面板名称
- 面板名称 - 面板名称

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

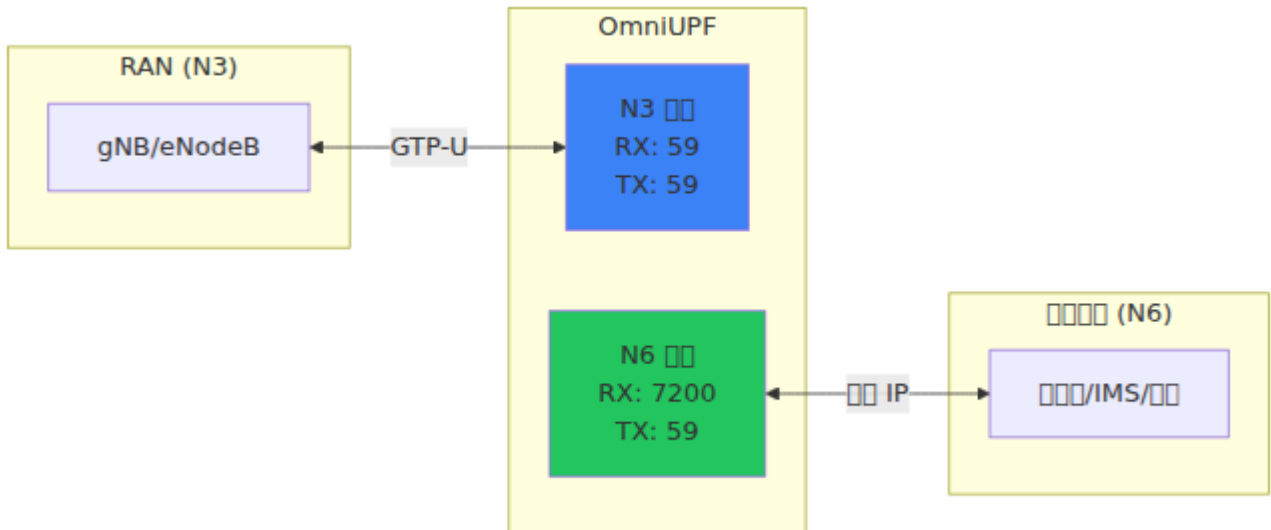
OmniUPF
 OmniUPF
 Web UI
 REST API

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

□□□□

N3/N6 □□□□

N3/N6 □□□□ RAN (N3) □□□□ (N6) □□□□□□□□



□□□

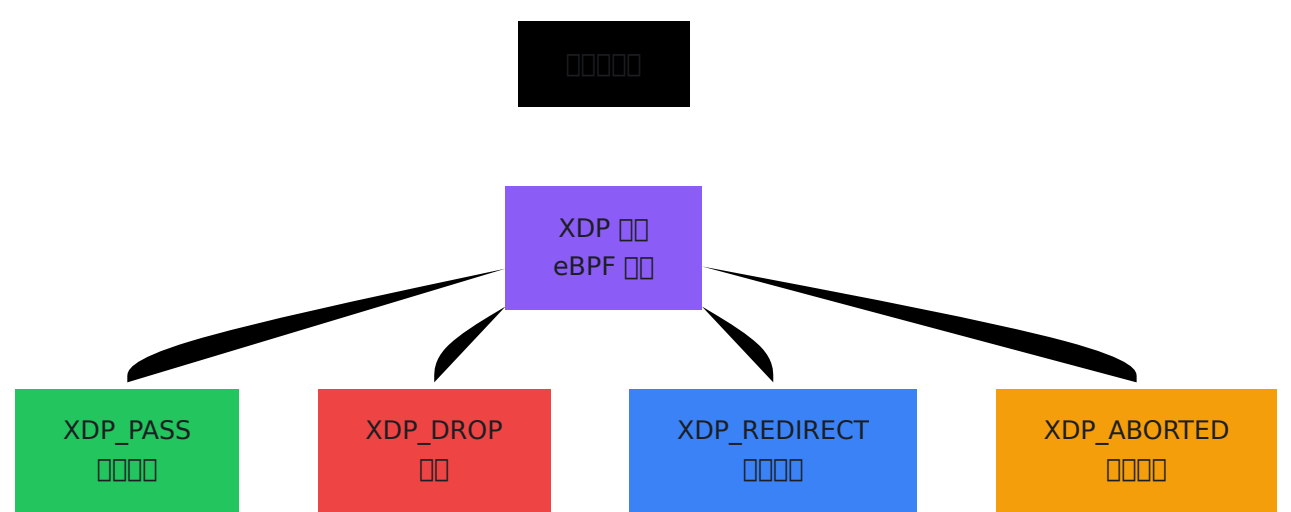
- **RX N3**□□□ RAN □□□□□□□□ GTP-U □□□
- **TX N3**□□□□ RAN □□□□□□□□ GTP-U □□□
- **RX N6**□□□□□□□□□□□□□□□□□□ IP□
- **TX N6**□□□□□□□□□□□□□□□□□□ IP□
- □□□□□□□□□□□□□□□□□□

□□□□□

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

XDP 简介

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



简介

- XDP 是一种高性能的数据包处理技术

- **TX** XDP 000000000000
- 000000000000000000000000
- 0000000000000000000000
- **TX** XDP 00000000

000

- 00 > 00 eBPF 0000000000000000
- 00 > 00 0000000000000000
- 000000000000000000000000
- 0000000000000000000000

000000

00000000000000000000

000000

- **RX ARP** 000000000000
- **RX GTP ECHO** GTP-U 0000/0000000000
- **RX GTP OTHER** GTP 0000
- **RX GTP PDU** GTP-U 0000000000000000
- **RX GTP UNEXP** GTP 000000
- **RX ICMP** 000000000000 ping 0000
- **RX ICMP6** ICMPv6 000
- **RX IP4** IPv4 000
- **RX IP6** IPv6 000
- **RX OTHER** 000000
- **RX TCP** 000000000000
- **RX UDP** 000000000000

000

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

- 透過 TCP 與 UDP 封包傳遞數據
- 透過 ICMP 封包傳遞控制消息

路由表

FIB (Forwarding Information Base) 路由信息庫

IPv4 FIB 路由

- 透過 ARP 表查詢 MAC 地址
- 透過 ND 表查詢 IPv6 地址

IPv6 FIB 路由

- 透過 ND 表查詢 IPv6 地址
- 透過 ND 表查詢 IPv6 地址

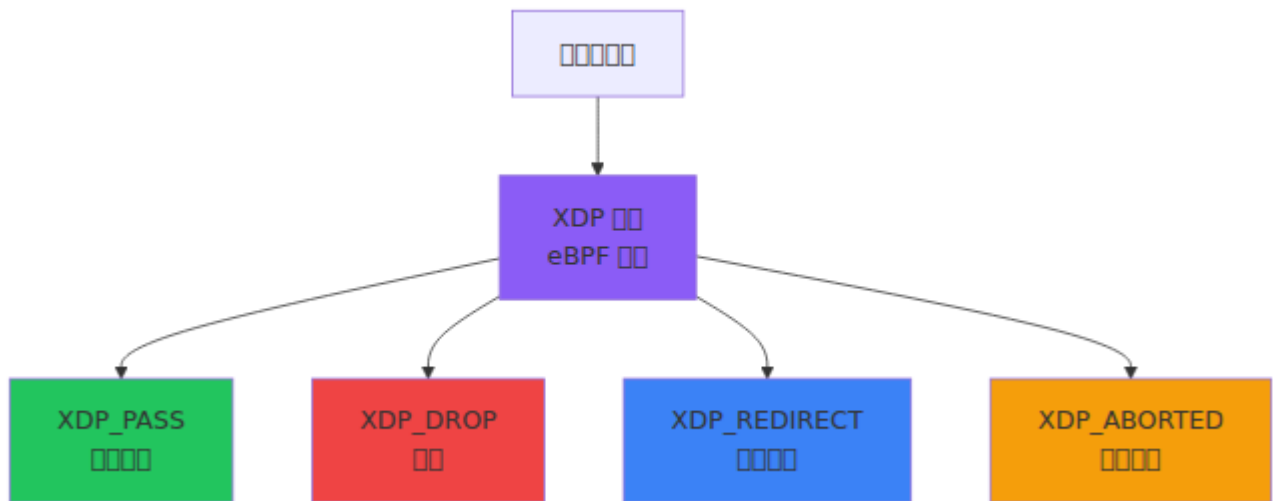
防火牆

- 透過 iptables 配置防火牆規則
- 透過 iptables 配置防火牆規則
- 透過 iptables 配置防火牆規則

網絡安全

eBPF 應用

eBPF 應用於網絡安全、性能監控、應用開發等場景



eBPF

far_map (FAR ID)

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

pdr_map_downlin (PDRs - IPv4)

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

pdr_map_downlin_ip6 (PDRs - IPv6)

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

pdr_map_teid_ip (PDRs)

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

qer_map (QoS)

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

urr_map (URRs)

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

0000

00	00000000
0-50% (00)	0000 - 000000
50-70% (00)	00 - 00000000000000
70-90% (000)	00 - 1 00000000
90-100% (00)	00 - 0000000000000000

0000000

00000000

- 00000000
- 0000000

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 UPF 設定

設定

OmniUPF Prometheus 設定

設定

設定

設定 (pps) = (設定) / (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 計測
- CPU 計測
- 遅延計測
- 帯域

遅延

遅延

$$\text{遅延 (Mbps)} = (\text{遅延時間} \times 8) / (\text{パケットサイズ} \times 1,000,000)$$

遅延

- 100 RX 遅延 500 MB
- 60 遅延 800 MB
- 遅延 = $(300 \text{ MB} \times 8) / (60 \times 1,000,000) = 40 \text{ Mbps}$

遅延計測には `expose_xdp_metrics` 遅延 `_total` 遅延計測
`upf_n3_rx_packets_total` `upf_n3_rx_bytes_total` ... N6 遅延計測 N3 の GTP-U の
`gNB` N6 の SGI 遅延計測 eBPF `upf_ext_stat` 遅延計測 `Sampler` 遅延計測
遅延計測 `virtio_net` TX-ring-wedge 遅延計測 #57 遅延計測
JSON の `/api/v1/xdp_stats` 遅延計測 遅延計測

遅延

- 遅延計測遅延計測遅延計測
- 遅延計測 N3/N6 遅延計測
- 遅延 2 遅延計測

遅延

遅延

$$\text{遅延 (\%)} = (\text{遅延時間} / \text{RX 遅延}) \times 100$$

Network

- **< 0.1%** packet loss
- **0.1% - 1%** packet loss
- **1% - 5%** packet loss QoS
- **> 5%** packet loss

Network

- QER MBR
- eBPF
- TEID UE IP
-

Network

Network

Network

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

Network1

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

Network

- eBPF > 50%

- 网络策略
- 网络 PFCP 策略/规则
- URR 策略

网络策略

网络策略实现

1. **Prometheus** 网络策略指标收集与存储
2. 网络策略 OmniUPF 网络策略
3. **REST API** 网络策略 `/map_info` `/packet_stats` 接口
4. **Web UI** 网络策略展示与配置

网络策略

网络策略

网络策略

```

网络策略 = min(
    PDR 网络策略 / 2, # 网络策略 + 网络策略 PDR 网络策略
    FAR 网络策略 / 2, # 网络策略 + 网络策略 FAR 网络策略
    QER 网络策略      # 网络策略 QER
)
  
```

网络策略

- PDR 网络策略131,070
- FAR 网络策略131,070
- QER 网络策略65,535

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

概要

eBPF を利用する

$$C = \sum (PDR \times (FAR + QER))$$

計算

- 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 $131,070 \times 20 \text{ B} = 2.6 \text{ MB}$
- 合計 ~91 MB 必要

環境構築

- 環境構築 `ulimit -l`
- 2 GB 以上
- 環境構築

実装

環境構築

1. 環境構築

- 5 Mbps
- 1 Mbps
- VoIP ~0.1 Mbps

2. 環境構築

$$C = PDR \times (FAR + QER)$$

3. 環境構築

$$\text{増倍率} = \text{増分} \times 2 \quad \# \quad 100\% \text{ 増}$$

増分

- 10,000 増分
- 増分 2 Mbps
- 増分 20 Gbps
- 増分 40 Gbps (N3 + N6 増分)

増分

増分

1. 増分
2. 増分
3. 増分

増分

$$\text{増分率} = (\text{増分} - \text{増分}) / (\text{増分})$$

増分

- 増分 30,000
- 増分 65,535 増分
- 増分 2,000 増分
- 増分率 (65,535 - 30,000) / 2,000 = **17.8** 増分

増分 12 増分 5 増分

네트워크

네트워크

네트워크 > 1% 네트워크

네트워크

1. 네트워크 → 네트워크
2. 네트워크 네트워크
3. 네트워크 XDP 네트워크 XDP 네트워크

네트워크

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

네트워크

- 네트워크 네트워크 QER MBR
- 네트워크 SMF 네트워크 PDR
- 네트워크

XDP 네트워크

네트워크 XDP 네트워크 > 0

네트워크

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

네트워크

- eBPF 開箱即用
- 開箱即用
- eBPF 開箱即用
- 開箱即用

開箱即用

- 開箱即用 OmniUPF 開箱
- 開箱即用開箱開箱開箱Linux 5.4+開箱
- 開箱 eBPF 開箱
- 開箱開箱開箱開箱

開箱即用

開箱開箱開箱開箱開箱 100%

開箱

1. 開箱開箱
2. 開箱開箱 100%
3. 開箱開箱開箱開箱

開箱開箱

1. 開箱開箱開箱開箱❓❓❓
2. 開箱 SMF 開箱
3. 開箱開箱開箱 FAR 開箱

開箱開箱

1. 開箱 eBPF 開箱
2. 開箱 UPF 開箱開箱開箱開箱
3. 開箱開箱

📌📌📌📌

📌📌📌📌📌📌📌📌CPU 📌📌

📌📌📌

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

📌📌📌📌📌

- 📌📌📌 UPF 📌📌
- 📌📌📌📌 CPU 📌📌📌
- 📌📌📌📌
- eBPF 📌📌📌📌

📌📌📌📌📌

- 📌📌📌 UPF📌📌📌📌📌📌
- 📌📌 CPU 📌📌 RSS📌📌📌📌📌
- 📌📌📌📌📌📌📌
- 📌📌 eBPF 📌📌📌📌

📌📌📌📌

- 📌📌📌 - 📌📌 Prometheus 📌📌📌
- **UPF** 📌📌📌 - 📌📌 UPF 📌📌📌📌
- 📌📌📌📌 - PDR📌FAR📌QER📌URR 📌📌
- **Web UI** 📌📌📌 - 📌📌📌📌📌📌
- 📌📌📌📌 - 📌📌📌📌📌
- 📌📌📌 - eBPF 📌📌📌📌📌📌

PFCP 配置指南

简介

PFCP是3GPP定义的用于控制用户面功能（UPF）的协议。OmniUPF 支持 PFCP 配置。

本文档基于 **3GPP TS 129.244** 规范，描述了 PFCP 的配置。

配置项

OmniUPF 通过 Prometheus 暴露 PFCP 配置项。PFCP 配置项如下：

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

配置项

- 配置项名称
- 配置项类型
- 配置项描述

本文档 **详细** 描述了 PFCP 配置。

配置示例

配置项

序号	配置项	配置值
1	RequestAccepted	配置项名称/配置项类型/配置项描述

00000000

00	00	0000
64	RequestRejected	00000000000000000000000000000000
65	SessionContextNotFound	00000000 SEID 000000000000 UPF 00000
66	MandatoryIEMissing	000000000000000000000000 NodeID 00000000 F-SEID 0000 RecoveryTimeStamp 0
67	ConditionalIEMissing	00000000000000000000000000000000 000000000000
69	MandatoryIEIncorrect	00000000000000000000000000000000 NodeID 000000 RecoveryTimeStamp 00000000 F-SEID 0
72	NoEstablishedPFCPAssociation	00000000000000000000000000000000 00000 PFCP 0000
73	RuleCreationModificationFailure	0 PDR 0 FAR 0 QER 0 URR 000000 eBPF 0000000000000000 eBPF 0000 000000000000000000000000

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

NodeID

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

NodeID FQDN IPv4/IPv6

SMF → UPF: RecoveryTimeStamp
UPF → SMF: MandatoryIEMissing

RecoveryTimeStamp

SMF → UPF:
UPF → SMF: NoEstablishedPFCPPAssociation

PFCP

SMF → UPF:
UPF FARQERURR
UPF PDReBPF
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 □□□□□□□□□□□□

🔍 故障調査と監視

📊 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 📄 📄 📄 📄 📄 📄 📄 📄

📄 📄 📄 📄 📄 📄

📄 MandatoryMissing 📄 📄

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.

NoEstablishedPCFPAssociation

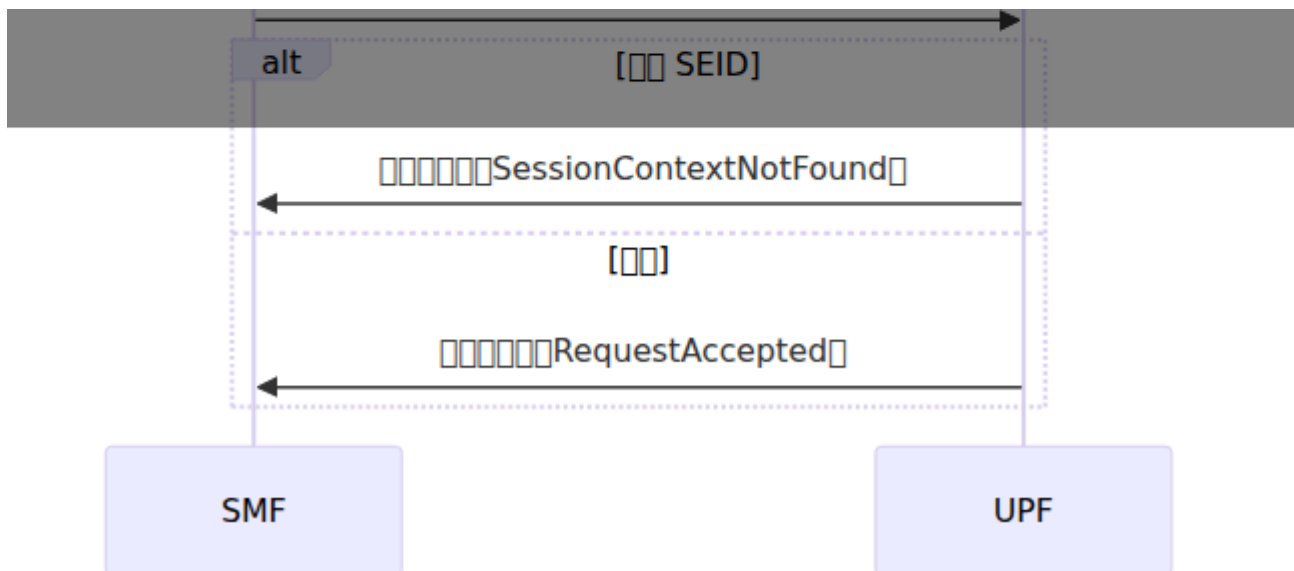
1. GET /api/v1/pfcp_associations
- 2.
- 3.
4. SMF UPF

SessionContextNotFound

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

□□□□□□□□□□

□□□□□□



□□□□□

□□□□□

```

# □□□□□□□
- 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: PfcphoAssociation
  expr: |

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

```

3GPP

OmniUPF

- **3GPP TS 129.244 v16.4.0** - PCF
- **8.2.1** -
- **8.19** -

- **PCF** - PCF
- - upf_pfcpx_errors
- -
- - PCF
- **Web UI** -

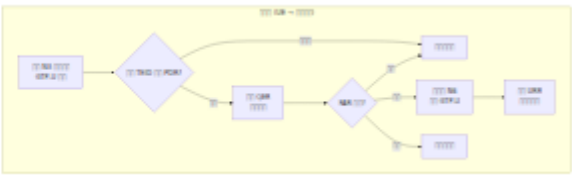
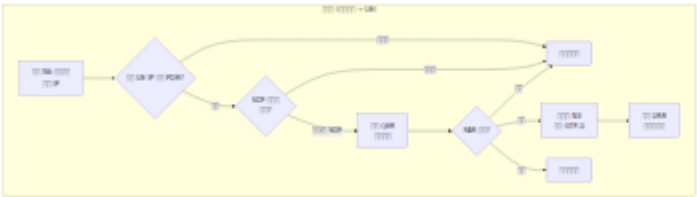
11

- 11

□ □ □ □

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

□□□□□□

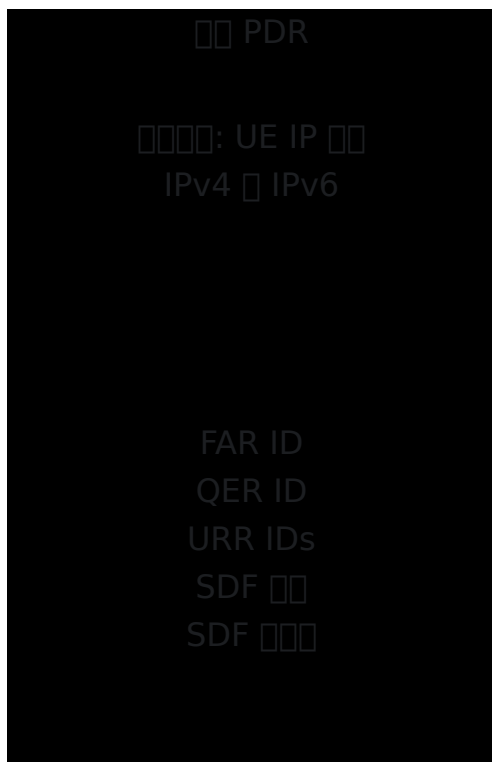


□□□□□□□ (PDR)

□□

PDR □□□□□□□□□□□□□□ UPF □□□□□□□□□□□□□□

PDR ☐ ☐



□□ PDR

□□ PDR □□□□ RAN □ N3 □□□□□□□□□□

TEID (TEID)

- 32
- SMF gNB
- UE

:

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

:

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

:

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

📄 PDR

📄 PDR 📄📄📄📄📄 N6 📄📄📄📄📄📄

📄📄📄: UE IP 📄

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

📄📄📄:

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

- **SDF + Default**: SDF ☐ FAR ☐

- **SDF** ☐: ☐ IP ☐

☐:

1. ☐ IP
2. ☐ **downlink_pdr_map** (IPv4) ☐ **downlink_pdr_map_ip6** (IPv6) ☐
3. ☐ SDF ☐
4. ☐ FAR ID ☐ QER ID ☐ URR IDs
5. ☐

☐:

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

SDF 规则 (SDF Rules)

SDF 规则用于指定流量过滤规则

规则:

- 允许 YouTube 流量
- 为 VoIP 流量提供高 QoS
- 阻止其他流量

规则:

- 协议: TCP, UDP, ICMP
- 端口: 80, 443, SIP 5060
- IP 地址: 10.0.0.0/24
- 规则: 3GPP 规则

规则 SDF 规则:

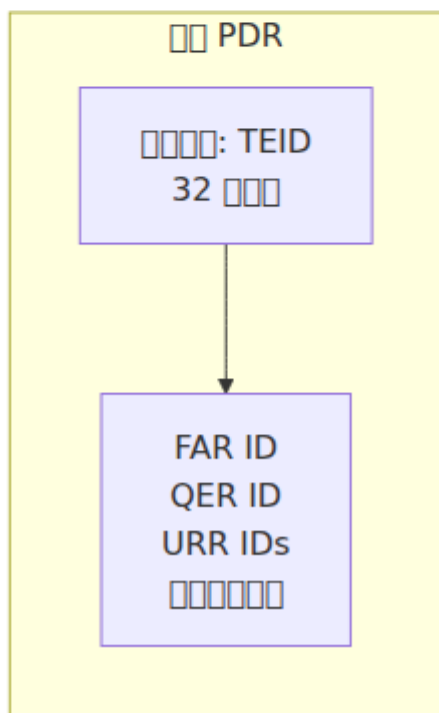
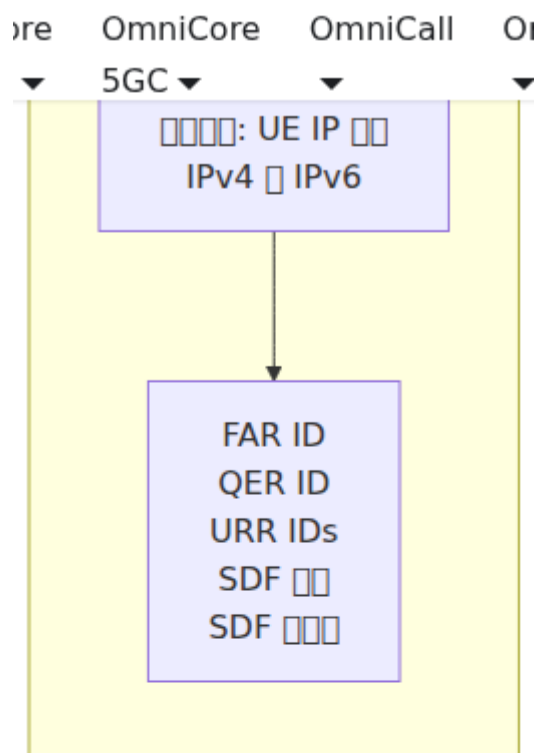
```
PDR ID: 10
UE IP: 10.45.0.1
SDF 规则: SDF Only
SDF 规则:
- 协议: UDP, 端口: 5060-5061 → FAR ID 5 (VoIP FAR)
- 协议: TCP, 端口: 443 → FAR ID 1 (Web FAR)
```

流量过滤规则 (FAR)

规则

FAR 规则用于指定流量过滤规则 GTP-U 流量

FAR



UE IP

FAR

動作	1	2	動作
FORWARD	1	2	パケット転送
BUFFER	2	4	パケットバッファリング
DROP	0	1	パケット丢弃
NOTIFY	3	8	パケット通知
DUPLICATE	4	16	パケット複製

動作:

- Action: 2 (FORWARD) - パケット転送
- Action: 6 (FORWARD + BUFFER) - パケットバッファリング
- Action: 4 (BUFFER) - パケット複製
- Action: 1 (DROP) - パケット丢弃

動作

BUFFER 動作 2 (FORWARD + BUFFER) は UPF がパケットをバッファリングし、UE にパケットを送信する

動作

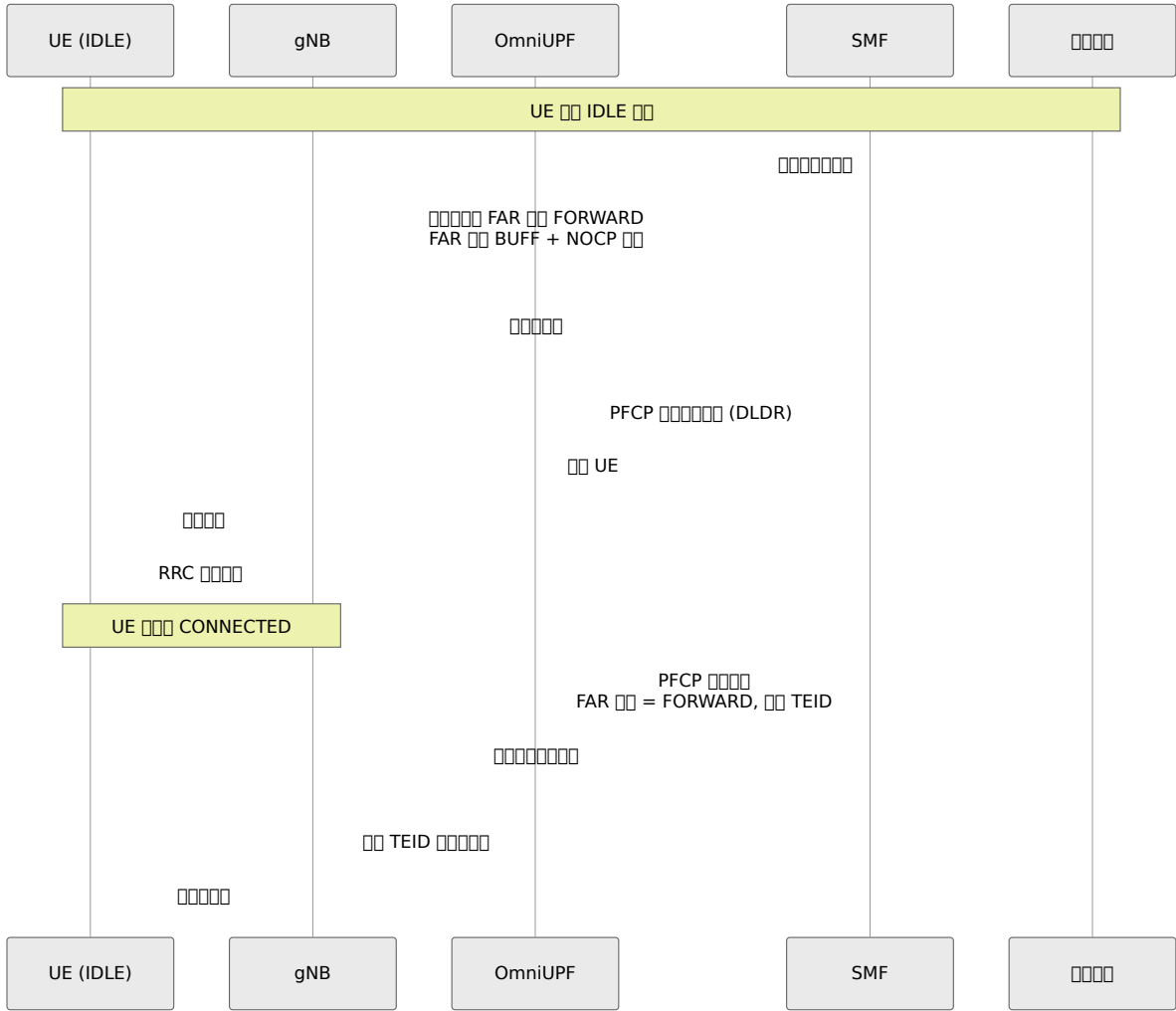
動作: 動作 1 (DROP) は gNB が UE にパケットを送信する前にパケットを丢弃する

1. 動作
2. SMF が DLDR (DL Data Rate) を設定する
3. SMF が UE にパケットを送信する
4. SMF が FAR (Forward Action Rate) を設定する
5. UPF が UE にパケットを送信する

動作: gNB が gNB がパケットを送信する前にパケットを丢弃する

1. gNB が動作
2. SMF が FAR (Forward Action Rate) を設定する

3. 5G Core Network Setup
4. UE connects to gNB
5. SMF sends FAR to UPF with TEID and FORWARD flag
6. UPF sends PFCP to gNB



Core Network Setup

Core Network Setup:

- Core Network Setup: 100,000 packets
- Core Network Setup: 100,000 packets
- **TTL (Packets):** 60 packets
- **TTL (Packets):** 60 packets

Core Network Setup:

- **Core Network Setup:** 10,000 packets

- **原因:** 网络拥塞 FAR 限制

网络拥塞:

- 网络拥塞时 FAR 限制
- 网络拥塞时 `reason="global_limit"` 或 `reason="far_limit"`
- 网络拥塞时网络拥塞 TTL 限制

网络拥塞 (**DLDR**)

UPF 网络拥塞 IDLE UE 网络拥塞 SMF 网络拥塞 PCFP 网络拥塞

DLDR 原因:

- 网络拥塞: 网络拥塞 (DLDR)
- **FAR ID:** 网络拥塞 FAR
- 网络拥塞: 网络拥塞 QFI 网络拥塞

SMF 网络拥塞 **DLDR** 原因:

1. 网络拥塞 AMF → gNB 网络拥塞 UE
2. 网络拥塞 UE 网络拥塞 RRC 网络拥塞
3. 网络拥塞 PCFP 网络拥塞 FAR
4. FAR 网络拥塞 `BUFF+NOCP` 网络拥塞 `FORW`
5. UPF 网络拥塞

DLDR 原因:

- `upf_dldr_sent_total`: 网络拥塞 DLDR
- `upf_dldr_send_errors`: 网络拥塞 DLDR
- `upf_buffer_notify_to_flush_duration_seconds`: 网络拥塞 DLDR 网络拥塞

网络拥塞 网络拥塞 网络拥塞

网络拥塞

网络拥塞 BUFF 网络拥塞:

- FAR 网络拥塞 `|= 0x04` 网络拥塞 2

- **□□**: Action: 2 (FORW) → Action: 6 (FORW+BUFF)
- **□□□□□□□□**

□□□□□□BUFF □ FORW□:

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

□□□□□□ BUFF □□□:

- FAR [RAX] &= ~0x04 [RAX] 2
- [RAX]: Action: 6 (FORW+BUFF) → Action: 2 (FORW)
- [RAX]

□□□□:

- 00 00 FAR 000000000000
- 0000000000 TEID/0000000
- 0000000000
- FAR 0000 FORW 00

□□□□:

- `XXXXXXXXXXXXXXXXXXXX`
- `XXXXXXXXXXXXXXXXXXXX`
- `XXXX reason="cleared"`

□ □ □ □ □ □ □ □

□□□□Web UI□: □□□ □□ □□□

- 00000000
- 000000
- 0000000000 FAR 00
- 00 FAR 00000000
- 000000000000
- 00/00000 FAR 000

- 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

0000000

00 1: IDLE UE 0000

0000:

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

0000:

1. DN 00000000
2. UPF 00 PDR000 FAR
3. FAR 00 BUFF 00 → 0000000
4. UPF 0 SMF 00 DLDR
5. SMF 00 UE
6. UE 000 gNB
7. SMF 00 FAR: Action = 0x02 (FORW)
8. UPF 000 TEID 00000000

00 2: 0000

UE:

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

SMF:

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

图 3: 注册

UE:

- UE 注册

SMF:

1. SMF 向 FAR: Action = 0x04 (向 BUFF)
2. 向 gNB-1 注册
3. 向 gNB-2 注册
4. SMF 向 FAR: Action = 0x02 (FORW) 注册
5. UPF 向 gNB-2 注册

□□□□□□

□□□□□□ GTP-U □□□

□□ **FAR** (N3 → N6):

- □□□□□□: False
- □□: □□ GTP-U□□□□□ IP □□□

□□ **FAR** (N6 → N3):

- □□□□□□: True
- □□ IP: gNB IP □□□□□□198.51.100.10□
- TEID: UE □□□□□ ID
- □□: □□ GTP-U □□□□□□ gNB

Web UI □□ **FAR** □□

□□□□□□□□ ID □□ FAR □□□□

|| :

1. 000000 → FARs 000
2. 000000000 FAR ID
3. 00 "00" 00 FAR 0000

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

QoS □□□□ (QER)

11

OFR

QER □□

QER 00

QFI
QoS 0000

00000
00/00

00000
00/00

QER ID
00000

00 MBR
00? ? ? 00

00 MBR
00000

00 GBR
00000

00 GBR
00000

QoS 設定

QFI (QoS 識別子):

- 5G QoS には 6 種類
- 1-9 の範囲で QFI 9 = 優先度
- 5GC によって

例:

- (0): 優先度
- (1): 優先度

最大ビットレート (MBR):

- 最大ビットレート
- kbps の単位
- **MBR = 0**: 無制限
- MBR の単位

最大保証ビットレート (GBR):

- 保証ビットレート
- kbps の単位
- **GBR = 0**: 無制限
- **GBR > 0**: 保証

QoS 設定

設定 (GBR = 0):

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

□□□ (GBR > 0):

QER ID: 2

QFI: 1

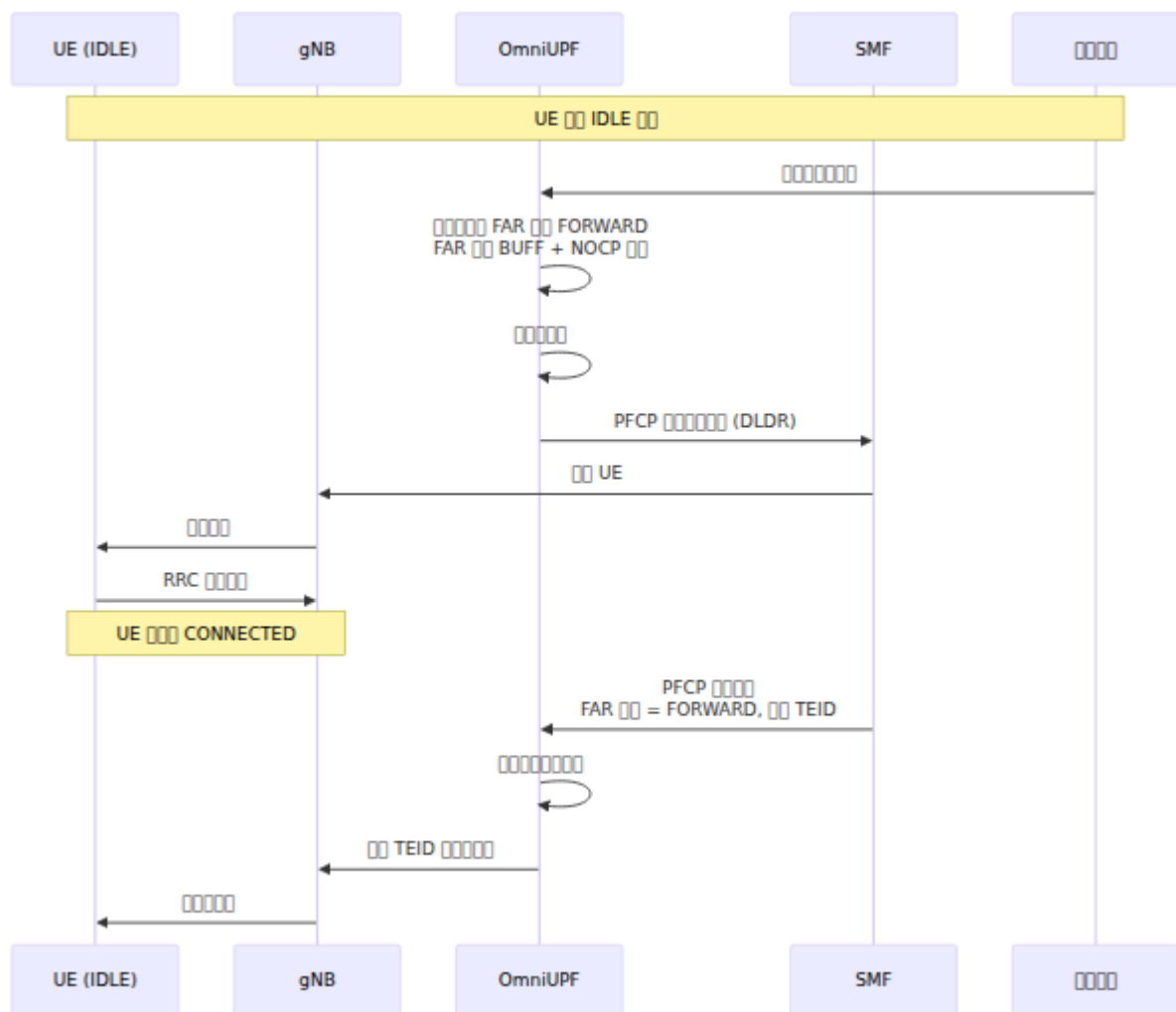
□□ MBR: 10000 kbps (10 Mbps)

□□ MBR: 10000 kbps (10 Mbps)

□□ GBR: 5000 kbps (5 Mbps)

□□ GBR: 5000 kbps (5 Mbps)

QoS



MBR

OmniUPF 는 MBR (Maximum Bit Rate) 을 eBPF (extended Berkeley Packet Filter) 또는 XDP (eXpress Data Path) 를 사용하여 설정합니다.

예시:

예시: MBR 설정

OmniUPF 는 다음과 같이

1. MBR: MBR (Maximum Bit Rate) 설정
2. **MBR** : MBR = 0 (Maximum Bit Rate)
3. MBR:


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

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

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

`tx_time`:

`tx_time`:

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

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

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

`tx_time` 2: `tx_time` `tx_time`

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

`tx_time` 3: `tx_time` `tx_time`

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

`tx_time`

5ms `tx_time`:

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

`tx_time`:

- 5ms
- MBR
-

:

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

(N3 → N6):

1. N3 gNB
2. TEID PDR
3. QER ID QER
4. `ul_gate_status` →
5. `limit_rate_sliding_window()` `ul_maximum_bitrate`
6. N6 URR

(N6 → N3):

1. N6
2. UE IP PDR
3. QER ID QER
4. `dl_gate_status` →
5. `limit_rate_sliding_window()` `dl_maximum_bitrate`
6. GTP-U N3

N9 (SGWU ↔ PGWU):

- N9 QER
- QER SGWU PGWU

MBR

MBR:

- 遅延: GTP-U/UDP/IP 遅延時間約 ~50-60 ms
- 遅延時間: 遅延時間 = 遅延時間
- 遅延時間: 遅延時間時間時間時間時間時間
- 遅延時間: 5ms 遅延時間時間時間時間 MBR

例:

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

遅延時間:

1. 遅延 URR 遅延時間時間時間: `upf_urr_*_volume_bytes`
2. 遅延時間: `(volume_delta_bytes × 8) / time_delta_seconds / 1000 = kbps`
3. 遅延 QER 遅延 MBR 遅延

GBR (遅延時間)

例: OmniUPF 遅延時間 GBR 遅延時間GBR 遅延 QER 遅延時間時間時間時間時間時間時間

GBR 遅延:

- GBR 遅延 PFCP 遅延 SMF 遅延
- GBR 遅延 QER 遅延時間時間 API 遅延
- 遅延時間時間時間 GBR 遅延時間
- GBR 遅延時間時間時間時間時間時間時間

遅延時間:

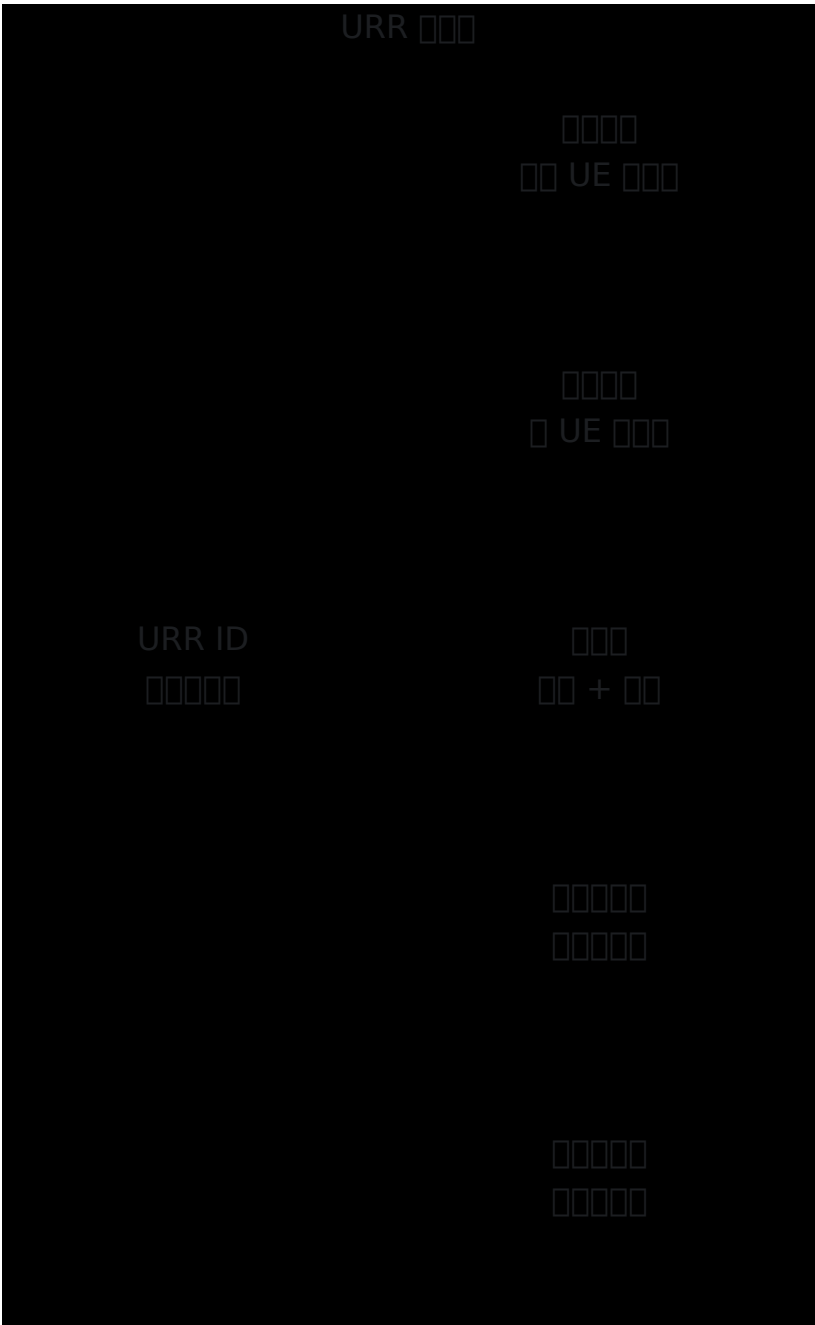
- GBR 遅延時間時間時間時間時間
- 遅延時間時間時間 eBPF QoS 遅延

URR (Uplink Resource Request)

UE

UE sends URR to SMF to request uplink resources for data transmission.

URR Details



Network

Network:

- UE IP address
- GTP-U tunnel
- IP address

Network:

- UE IP address
- GTP-U tunnel
- IP address

Network:

- IP address
- IP address

Network

URR IP address

Network:

- IP address
- IP: 1 GB IP address

Network:

- IP address
- IP: 5 IP address

Network:

- IP address
- QoS IP address
- IP address

000000

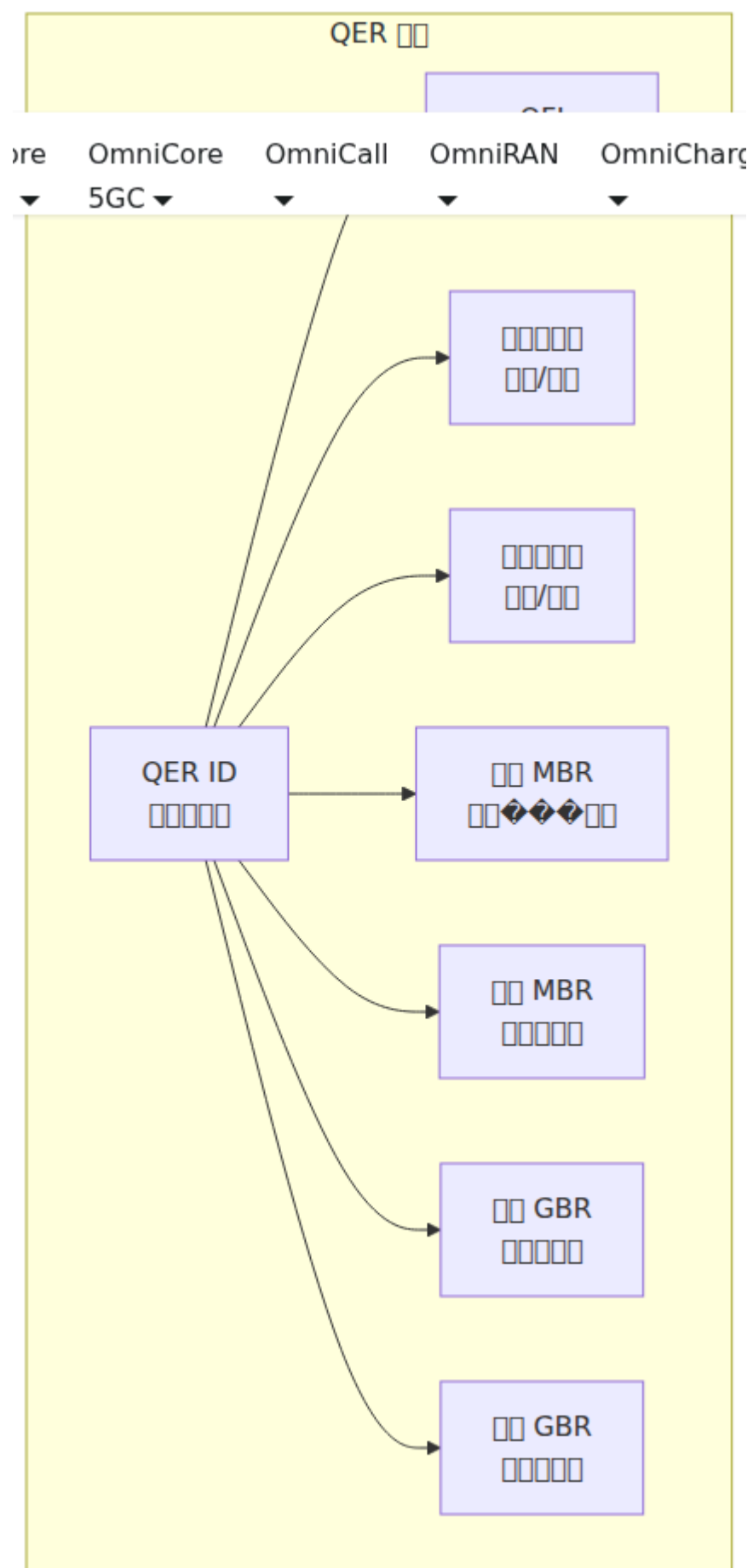
Web UI 0000000000000000

00	00
0 - 1023	B (00)
1024 - 1048575	KB (000)
1048576 - 1073741823	MB (000)
1073741824 - 1099511627775	GB (0000)
1099511627776+	TB (000)

00:

URR ID: 0
0000: 12.3 KB
0000: 9.0 KB
000: 21.3 KB

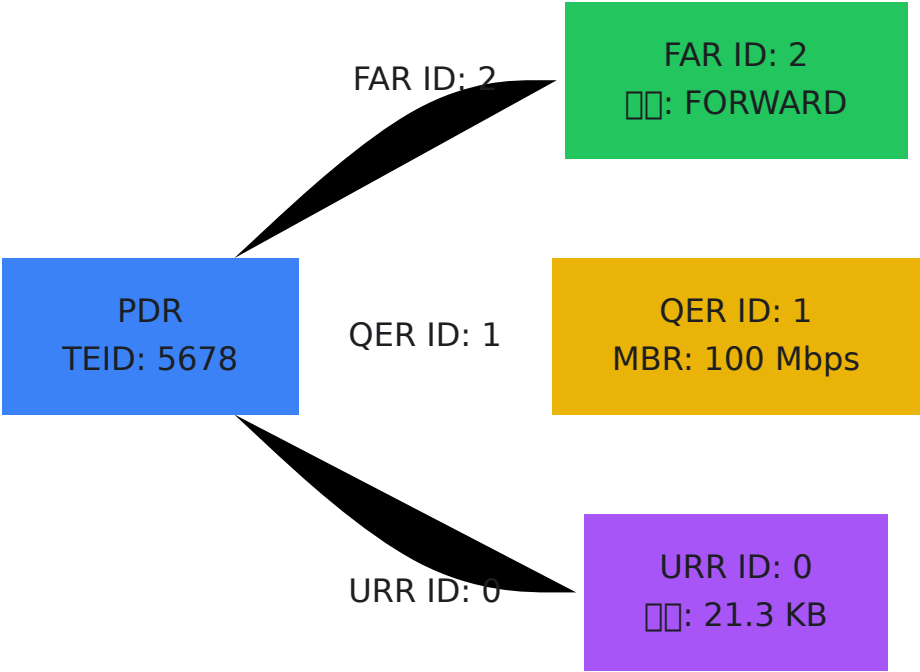
URR □□□□



□□□□

PDR → FAR → QER → URR □

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



□□□□□□

□□ **PDR:**

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

□□ **PDR:**

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

FAR ID 1 (00):

00: 2 (FORWARD)
000000: True
00 IP: 198.51.100.10
TEID: 5678

FAR ID 2 (00):

00: 2 (FORWARD)
000000: False

QER ID 1:

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

URR ID 0:

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

□□□□

□□□□□□□

□□□□□:

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

□□□□□:

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

□□/□□□□

□□: □□□□□□□□□□□□□□□□

□□:

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

□□□□□□□□□□:

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

QoS

QoS:

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

QoS:

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

QoS MBR

QoS

QoS URR:

1. QoS → URRs
2. QoS
3. QoS
4. QoS

QoS:

- QoS
- QoS
- QoS

QoS

QoS

QoS PDR:

5. 1 FAR 10000

10000:

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

URR 1000

1000000:

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

1000000 **SMF**:

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

100000

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

OmniUPF 部署架构图

部署

1. 网络
2. 服务器
3. 数据库
4. 负载均衡
5. PFCP 网关
6. 鉴权服务器
7. XDP 与 eBPF 部署
8. 配置管理
9. 日志收集与处理
10. NIC 部署与配置
11. 网络策略
12. 安全策略

部署

部署架构图展示了 OmniUPF 的部署架构，包括网络、服务器、数据库、负载均衡、PFCP 网关、鉴权服务器、XDP 与 eBPF 部署、配置管理、日志收集与处理、NIC 部署与配置、网络策略、安全策略等组件。

部署架构图

部署架构图展示了 OmniUPF 的部署架构，包括网络、服务器、数据库、负载均衡、PFCP 网关、鉴权服务器、XDP 与 eBPF 部署、配置管理、日志收集与处理、NIC 部署与配置、网络策略、安全策略等组件。


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

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

# 3. 检查 eBPF 是否安装
ls /sys/fs/bpf/

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

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

检查

OmniUPF REST API

检查 **UPF** 是否安装

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

检查 **PFCP** 是否安装

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

检查

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

检查 **eBPF** 是否安装

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

□□□□□□□□□□

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

□□ **XDP** □□□□□

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

eBPF □□□□

□□□□ **eBPF** □□□

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

□□□□□□□□□□

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

□□□□□□□□□□

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

XDP □□□□

□□ **XDP** □□□□□□□□

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

看看 **XDP** 看看

```
bpftool net list
```

看看 **XDP** 看看看看

```
bpftool prog show
```

看看 **XDP** 看看

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

看看

看看 **N4** 看看 **PFCP** 看看??

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

看看 **N3** 看看 **GTP-U** 看看看看看看

```
# 在UPF 侧使用 tcpdump 捕获 XDP 数据包
# XDP 数据包通过 GTP-U

# 配置步骤
# 1. gNB 与 UPF 建立 TAP
# 2. 配置SPAN 到 N3 接口
# 3. 配置 VM

# 在UPF 侧使用 tcpdump 捕获 XDP 数据包
# tcpdump -i <mirror_interface> -n udp port 2152 -w /tmp/n3_capture.pcap

# 使用 API 获取数据
curl http://localhost:8080/api/v1/packet_stats
curl http://localhost:8080/api/v1/n3n6_stats
```

配置完成

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

配置完成

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

查看 ARP 表

```
ip neigh show
```

配置完成

配置“eBPF 数据包捕获”

配置

```
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

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

# 安装
mount | grep bpf
```

安装内核头文件

安装

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

安装Linux 内核头文件

安装

```
# 確認
uname -r

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

# 確認
uname -r # 5.15.0 >= 5.15.0
```

libbpf 確認

確認

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

libbpf 確認

確認

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

# 確認
ldconfig -p | grep libbpf
```

確認

確認

確認

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

编译Elixir 编译选项

编译选项

```
# 编译选项
cd /opt/omniupf && bin/upf_ex eval "IO.puts(:ok)"

# 编译选项
# - 编译选项
# - Elixir 编译选项 : 编译选项
# - 编译选项

# 编译选项 /etc/omniupf/runtime.exs
# 编译选项
# xdp_interfaces = "eth0"
# xdp_attach_mode = "generic"
# api_port = 8080
# pfc_address = "10.0.0.1"
```

编译选项

编译

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

编译选项

编译选项

```
# 確認
ip link show

# 確認
ip addr show eth0

# /etc/omniupf/runtime.exs
# xdp_interfaces = "ens1f0" # 確認

# 確認
ls /sys/class/net/
```

確認

確認

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

確認 8080 8805 9090 確認

確認

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

# 確認
sudo kill <PID>

# /etc/omniupf/runtime.exs 確認 OmniUPF 確認
# api_port = 8081
# pfcport = 8806
```

確認 **PFCP** ID

確認

ERR0[0000] invalid pfc_p_node_id: must be valid IPv4 address

PFCP ID IPv4

```
# /etc/omniupf/runtime.exs
# IP
node_id = "10.100.50.241"

#
# node_id = "localhost"
# node_id = "upf.example.com"
```

PFCP

PFCP

- Web UI “”
- SMF “PFCP ”

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

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

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

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

检查配置

检查 **PFCP**

检查

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

PFCP 检查 ID 配置

检查

```
# 在 /etc/omniupf/runtime.exs 中配置 ID 和 N4 的 IP
node_id = "10.100.50.241"    # 设置 N4 的 IP 地址
```

检查 **SMF**

检查

```
# ping SMF
ping <SMF_IP>

# show SMF route
ip route get <SMF_IP>

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

SMF connects UPF IP

Steps

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

PFPCP connects

Logs

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

Logs

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

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

PFPCP connects

PFPCP connects

000000

```
# 000 SMF 000000000000
ping -c 100 <SMF_IP> | grep loss
```

```
# 00000000000000000000
# - 00000000
# - 000000/0000000000
# - 00000000
```

0000000000

000000

```
# 0 /etc/omniupf/runtime.exs 0
heartbeat_interval_ms = 30_000      # 0 5000 000 30000 00
heartbeat_retries = 5                # 00000000
heartbeat_timeout_ms = 10_000        # 00000000
```

0000000000

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

000

- 0000000 0 RX/TX 000
- UE 0000000000

000

```
# 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"

#
```

```
# /etc/omniupf/runtime.exs
xdp_interfaces = "ens1f0" # 'ip link show' 確認
```

確認

確認

- RX 確認 TX 確認
- 確認 > 1%

確認

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

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

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

確認

確認 **PDR** 確認 **TEID** 確認 **UE IP**

確認

```
# 確認セッション
curl http://localhost:8080/api/v1/sessions

# 確認セッション詳細
# - PFCP 確認
# - SMF 確認
# - 確認

# PDR 確認
bpftool map dump name pdr_map_teid_ip | grep -c key
bpftool map dump name pdr_map_downlin | grep -c key
```

確認

確認

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

# UE IP 確認
ip route get 10.45.0.100

# 確認
sudo ip route add 10.45.0.0/16 dev eth1 # UE 確認 N6
```

QER 確認

確認

- 確認
- 確認
- URR 確認
- 確認 XDP 確認

確認

1. 確認 **MBR** 確認

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

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

2. 取得 QER 詳細

```
# 取得 QER 詳細 (QER ID = 1)
curl http://localhost:8080/api/v1/qer_map | jq '.data[] |
{qer_id, ul_gate: .ul_gate_status, dl_gate: .dl_gate_status}'
```

3. 取得 URR 詳細

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

# 計算吞吐量 (Throughput)
# throughput_kbps = (volume_delta_bytes × 8) /
time_delta_seconds / 1000
```

4. 取得 MBR 詳細

- 吞吐量 $\approx$ MBR の 95-98%
- 取得した MBR 値を確認
- 取得した MBR 値と比較

まとめ

- 取得した MBR 値と SMF から PFCP 経由で取得した QER 値と MBR
- 取得した MBR 値と SMF から取得した MBR 値を比較
- 取得した MBR 値と SMF から取得した QoS 値を比較

取得 MBR 値

OmniUPF 需要の MBR 必要 - MBR

-
- MBR
-
- 5ms

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

- RX N3 TX N3
- RX N6 TX N6

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

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

RX N3 TX N6

FAR N6

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

```
# [] N6 []
ip route get 8.8.8.8 # []
```

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

[]RX N6[] TX N3[]

[] PDR [] GTP []

[]

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

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

```
# [] gNB []
ping <GNB_N3_IP>
```

XDP [] eBPF []

[] XDP [] XDP []

[]XDP []

[]

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

❏❏❏

```
# ❏❏❏❏ XDP ❏❏  
grep XDP /boot/config-$(uname -r)  
  
# ❏❏❏❏  
# CONFIG_XDP_SOCKETS=y  
# CONFIG_BPF=y  
# CONFIG_BPF_SYSCALL=y  
  
# ❏❏ dmesg ❏❏❏❏❏❏❏  
dmesg | grep -i bpf
```

❏❏❏❏❏❏❏❏

❏❏❏❏ **XDP** ❏❏

❏❏❏❏❏

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

XDP ❏❏❏❏❏❏

❏❏❏❏❏

```
# ❏❏ OmniUPF ❏❏❏❏❏❏❏❏❏❏  
journalctl -u omniupf | grep verifier  
  
# ❏❏❏❏❏  
# - eBPF ❏❏❏❏❏❏❏❏❏❏❏❏❏❏❏  
# - ❏❏❏❏❏❏❏❏eBPF ❏❏❏❏❏❏❏  
  
# ❏❏ eBPF ❏❏❏❏❏❏❏❏❏❏❏❏❏  
sudo sysctl kernel.bpf_stats_enabled=1
```

🔍 XDP 🔍

🔍

- XDP 🔍 aborted > 0
- 🔍

🔍

```
# 🔍 XDP 🔍  
curl http://localhost:8080/api/v1/xdp_stats | jq '.aborted'  
  
# 🔍 XDP 🔍  
watch -n 1 'curl -s http://localhost:8080/api/v1/xdp_stats'
```

🔍 eBPF 🔍

🔍

```
# 🔍 eBPF 🔍  
dmesg | grep -i bpf  
  
# 🔍 OmniUPF 🔍 eBPF 🔍  
sudo systemctl restart omniupf  
  
# 🔍 eBPF 🔍  
# 🔍 BPF_ENABLE_LOG=1 🔍 OmniUPF
```

🔍 eBPF 🔍

🔍

- 🔍
- 🔍 100%

🔍

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

# 取得高使用率データ
curl http://localhost:8080/api/v1/map_info | jq '.[[]] | select(.usage_percent > 90)'
```

確認

```
# 1. 取得全セッション
curl http://localhost:8080/api/v1/sessions | jq '.[[]] | {seid, uplink_teid, created_at}'

# 2. 取得 SMF 取得全 API
curl http://localhost:8080/api/v1/sessions | jq '.[[]] | {seid, uplink_teid, created_at}'

# 3. 取得高使用率セッション
watch -n 5 'curl -s http://localhost:8080/api/v1/map_info | jq ".[[]] | select(.map_name==\"pdr_map_downlin\") | .usage_percent\"'
```

確認

```
# /etc/omniupf/runtime.exs
max_sessions = 200_000 # 100_000

# 取得全セッション
far_map_size = 400_000
qer_map_size = 200_000
```

OmniUPF

前提条件

ネットワーク環境

環境

- 速度 < 1 Gbps の NIC を使用する
- CPU を使用する

コマンド

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

# NIC の統計を確認
ethtool -S eth0 | grep -i drop

# XDP の統計を確認
ip link show eth0 | grep xdp
```

確認

ネットワーク **XDP** の設定

確認

```
# /etc/omniupf/runtime.exs に
xdp_attach_mode = "native" # XDP を NIC/ネットワーク
```

確認

確認

```
# 1 NIC 1 RSS111111111111
ethtool -L eth0 combined 4 # 1 4 1 RX/TX 11

# 1 RSS 11111
ethtool -l eth0

# 1111111111 CPU
# 1111 /proc/interrupts 1111 irqbalance 11111111
```

11111

111111

```
1111111 Elixir 1111111111
11 /etc/omniupf/runtime.exs 1111111111
```

11111111

111

- Ping 11 > 50ms
- 11111111

111

```
# 111 UE 111
ping -c 100 <UE_IP> | grep avg

# 1111111111
curl http://localhost:8080/api/v1/upf_buffer_info | jq
'.total_packets_buffered'

# 1111111111
curl http://localhost:8080/api/v1/packet_stats | jq '.route_stats'
```

111111


```
# 检查 NIC 状态
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

网络性能优化包括 **XDP** 等技术

Proxmox: VM 网络 XDP 配置

配置

- 配置 XDP 程序
- 配置网络

VM 网络 SR-IOV

配置

1. 配置网络

```
# /etc/omniupf/runtime.exs
xdp_attach_mode = "generic"
```

2 SR-IOV

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

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

# 3. Proxmox UI VF VM
# → PCI → VF
```

`/etc/omniupf/runtime.exs`

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

VMware:

- OmniUPF

vSwitch MAC

```
# 1. vSwitch 连接到 vSphere 网络
# 2. 1. vSwitch → 网卡
# 3. 网卡 → MAC 地址
# 4. MAC 地址 → 虚拟机
```

VirtualBox: 网络

网络

- 网络 < 100 Mbps

VirtualBox 支持 SR-IOV 支持 XDP

网络

```
# 1. /etc/omniupf/runtime.exs 1
xdp_attach_mode = "generic" # VirtualBox 网络
```

VirtualBox 网络

- VirtIO-Net 网络
- “网络”网络
- VM 网络 CPU 网络
- 网络 NAT
- KVM/Proxmox 网络

NIC 网络

NIC 网络 XDP

网络

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

❌❌❌

```
# 查看 NIC 信息
ethtool -i eth0 | grep driver

# 查看驱动支持的 XDP
modinfo <driver_name> | grep -i xdp

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

❌❌❌❌

❌❌ **1**❌❌❌❌❌❌

```
# 在 /etc/omniupf/runtime.exs 中
xdp_attach_mode = "generic"
```

❌❌ **2**❌❌❌ **NIC** ❌❌❌

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

# 查看驱动支持
# Intel 驱动
# 在 https://downloadcenter.intel.com/ 中
```

❌❌ **3**❌❌❌ **NIC**

```
# 查看 XDP 支持的 NIC
# - Intel X710, E810
# - Mellanox ConnectX-5, ConnectX-6
# - Broadcom BCM57xxx (bnxt_en 驱动)
```

問題発生時の調査手順

確認

- XDP モジュール
- NIC モジュール

確認

```
# 確認
dmesg | tail -100

# 確認
journalctl -k | grep -E "BUG:|panic:"
```

確認

```
# 1. パッケージ更新
sudo apt update
sudo apt upgrade
sudo reboot

# 2. /etc/omniupf/runtime.exs の XDP モジュール
# xdp_attach_mode = "native"

# 3. パッケージ更新
# xdp_attach_mode = "generic"

# 4. NIC の Linux モジュール
```

IPv6 SLAAC / UDP モジュール-XDP + virtio

確認

```
# IPv6 UE 50% SLAAC
# RA  UPF  RAN
#  eNB/gNB  UDP
#  RA
nstat -az | grep UdpInCsumErrors      #  SLAAC
```

- **IPv6 SLAAC** / —
- SGW-U PGW-U SGW-U+PGW-U

TX `CHECKSUM_PARTIAL` `skb` `tx-checksum-ip-generic: on` **XDP** `skb` SGW-U RA IP + TEID `udp->check` `skb` `csum_start/csum_offset` `virtio` NIC UDP `CHECKSUM_NONE` eBPF NIC — SLAAC XDP `skb` **virtio** NIC

```
#  NIC  TX
#  UDP  XDP
#
ethtool -K <n3_iface> tx off
ethtool -K <n6_iface> tx off

# "off"
ethtool -k <n3_iface> | awk '/^tx-checksumming/{print}'

# SLAAC  UdpInCsumErrors
```

NIC XDP `xdp_attach_mode: native`

環境構築

OS 環境構築

OS

- SMF 環境構築
- UE 環境 PDU 環境

OS **PF** 環境構築 環境構築

OS

```
# OS OmniUPF 環境構築
journalctl -u omniupf | grep -i "session establishment"

# OS PF 環境構築
curl http://localhost:8080/api/v1/sessions | jq 'length'

# 環境構築 PF 環境構築
tcpdump -i any -n udp port 8805 -w /tmp/pfcp_session.pcap
```

環境構築

環境構築

環境構築

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

# 環境構築 eBPF 環境構築
```

OS **PDR/FAR** 環境構築

環境構築

```
# 检查 OmniUPF 日志
journalctl -u omniupf | grep -E "invalid|error" | tail -20

# 配置
# - 设置 UE IP 地址 10.0.0.0 网段
# - 设置 TEID 范围
# - PDR 和 FAR
# - 设置 FAR 规则

# 检查 SMF 配置
```

配置 UEIP/FTUP

配置

```
# 在 /etc/omniupf/runtime.exs 中
feature_ueip = true
ueip_pool = "10.60.0.0/16"

feature_ftup = true
teid_pool_start = 1
teid_pool_end = 100_000
```

配置

配置

配置

- 配置
- 配置

配置


```
# 確認
curl http://localhost:8080/api/v1/upf_buffer_info

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

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

確認

FAR 確認 FORWARD

確認 SMF 確認 PCFP 確認 FAR

確認

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

# 確認 BUFF = 1
# 確認 FORW = 2

# 確認 BUFF 確認 SMF
# - 確認 PCFP
# - 確認 FAR 確認 FORW
```

確認 TTL

確認 FAR 確認

確認 TTL Elixir `/etc/omniupf/runtime.exs` 確認

確認

確認 FAR 確認

インストール済 Elixir をインストールして `/etc/omniupf/runtime.exs` をインストール

インストール

インストール

```
# /etc/omniupf/runtime.exs
log_level = :debug    # :debug | :info | :warning | :error
```

```
# OmniUPF を再起動
sudo systemctl restart omniupf

# ログを確認
journalctl -u omniupf -f --output cat
```

eBPF インストール

```
# eBPF をインストール
sudo bpftool install

# eBPF を確認
sudo bpftool dump
```

UDP XDP インストール

UDP XDP インストール

UDP をインストールして `tcpdump` をインストールして XDP をインストールして N3 の GTP-U をインストールして UDP をインストールして XDP をインストールして UPF をインストールして `tcpdump` をインストール

インストール

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

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

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

VMware, KVM

```
# UPF VM
# Linux VM tcpdump
# UPF N3

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

環境構築

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

仮想 UPF 環境構築

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

確認

環境構築確認

1. 環境構築

```
# 確認
uname -a
cat /etc/os-release

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

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

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

2. 网络功能

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

网络

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

OmniUPF 安裝 / 設定說明書

目錄

1. 簡介
2. 需求
3. PFCP 設定
4. 網路設定
5. 防火牆
6. 路由設定
7. 設定 DNS URL
8. API
9. Prometheus 設定
10. 測試

安裝

本說明書適用於在 Linux 系統上安裝 OmniUPF。UPF 安裝需要 Mikrotik 的 mangle 規則和 DNAT 設定。

安裝時，SMF 會透過 `redirect_information` 與 PFCP 進行通訊。OmniUPF 會透過 PFCP 與 SMF 進行通訊。

- **DNS** 設定：設定 DNS 伺服器 IP，例如 Apple、Android、Windows 等。
- **IP** 設定：設定 IP 地址和子網掩碼。
- **防火牆**：設定防火牆規則。

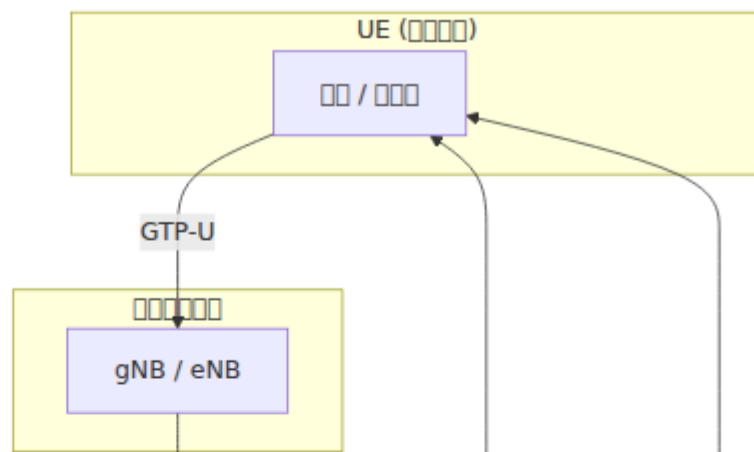
安裝完後，需要設定 SMF 的 FAR 和 FORW 規則。

測試

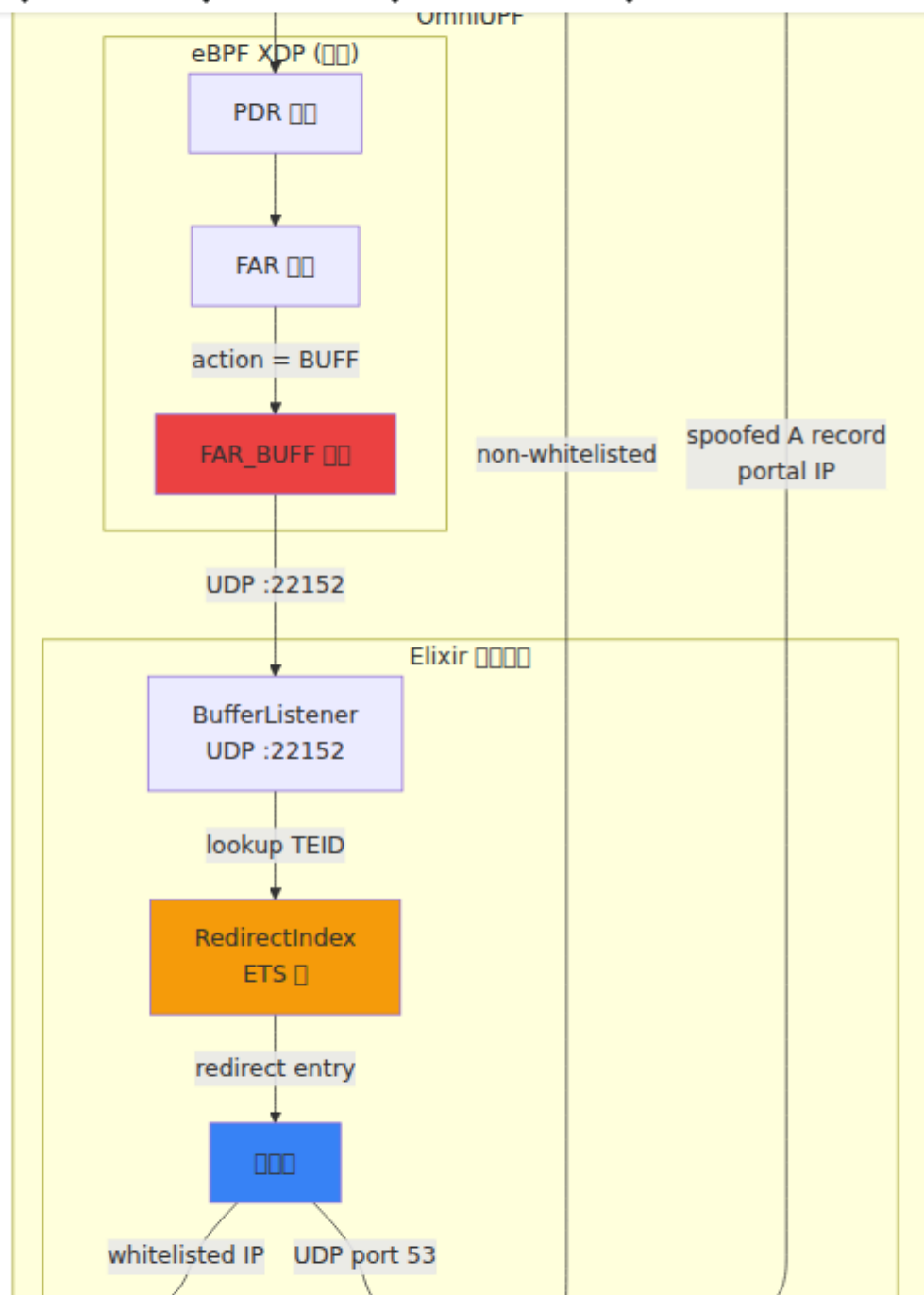
- 使用 **eBPF** 工具 -- 設定 `FAR_BUFF` 測試。

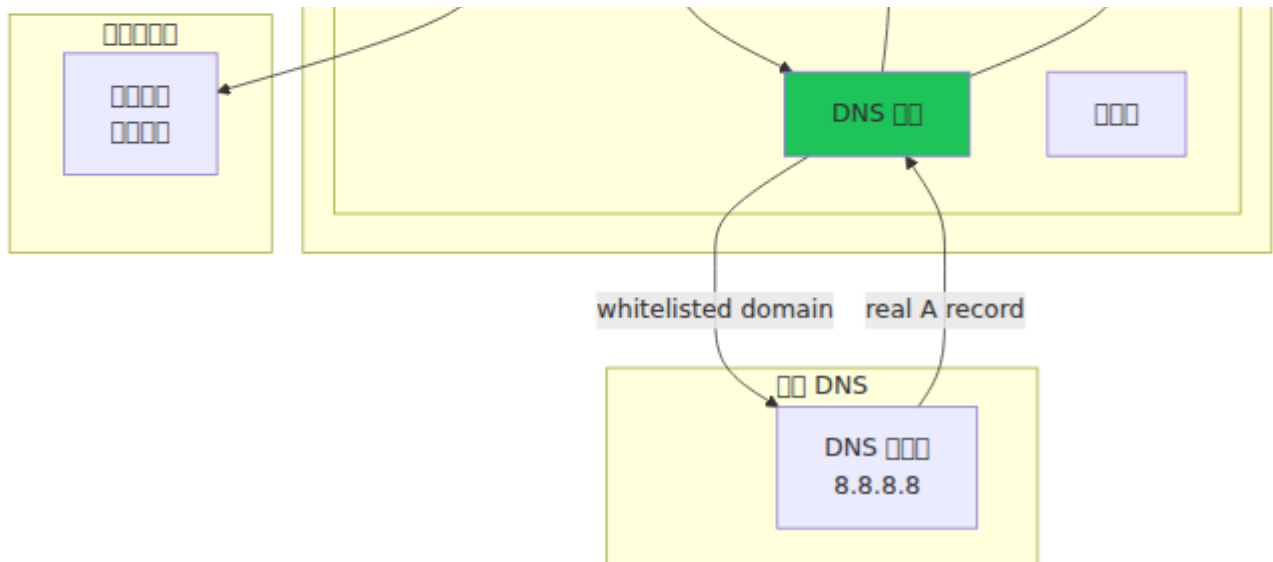
- 3GPP -- 3GPP TS 23.002 IP address URL SMF `redirect_information` IE
 - 3GPP **UPF** -- SMF "3GPP" UPF
 - 3GPP -- 3GPP FAR 3GPP FORW 3GPP GTP 3GPP UE
-





ore OmniCore OmniCall OmniRAN OmniCharge Platform 5GC





1. **UE** 向 **GTP-U** 发送DNS 请求HTTP 请求 通过 UPF
2. **eBPF PDR** 向 网络 PDR 发送 FAR
3. **FAR** 向 **BUFF** 发送FORW 请求-- eBPF 接收UDP 包 22152 通过 BufferListener
4. **BufferListener** 向 TEID 发送 RedirectIndex ETS 包
5. 向 TEID 发送 RedirectIndex 包**Dispatcher** 接收 IP 包
6. 接收
 - 接收DNS 请求 A 请求 IP
 - 接收DNS 请求 接收 IP
 - 接收 IP 请求 IP 请求
 - 接收
7. **DNS/GTP-U** 向 网络 GTP-U 发送 DL TEID 请求 gNB 接收 UE

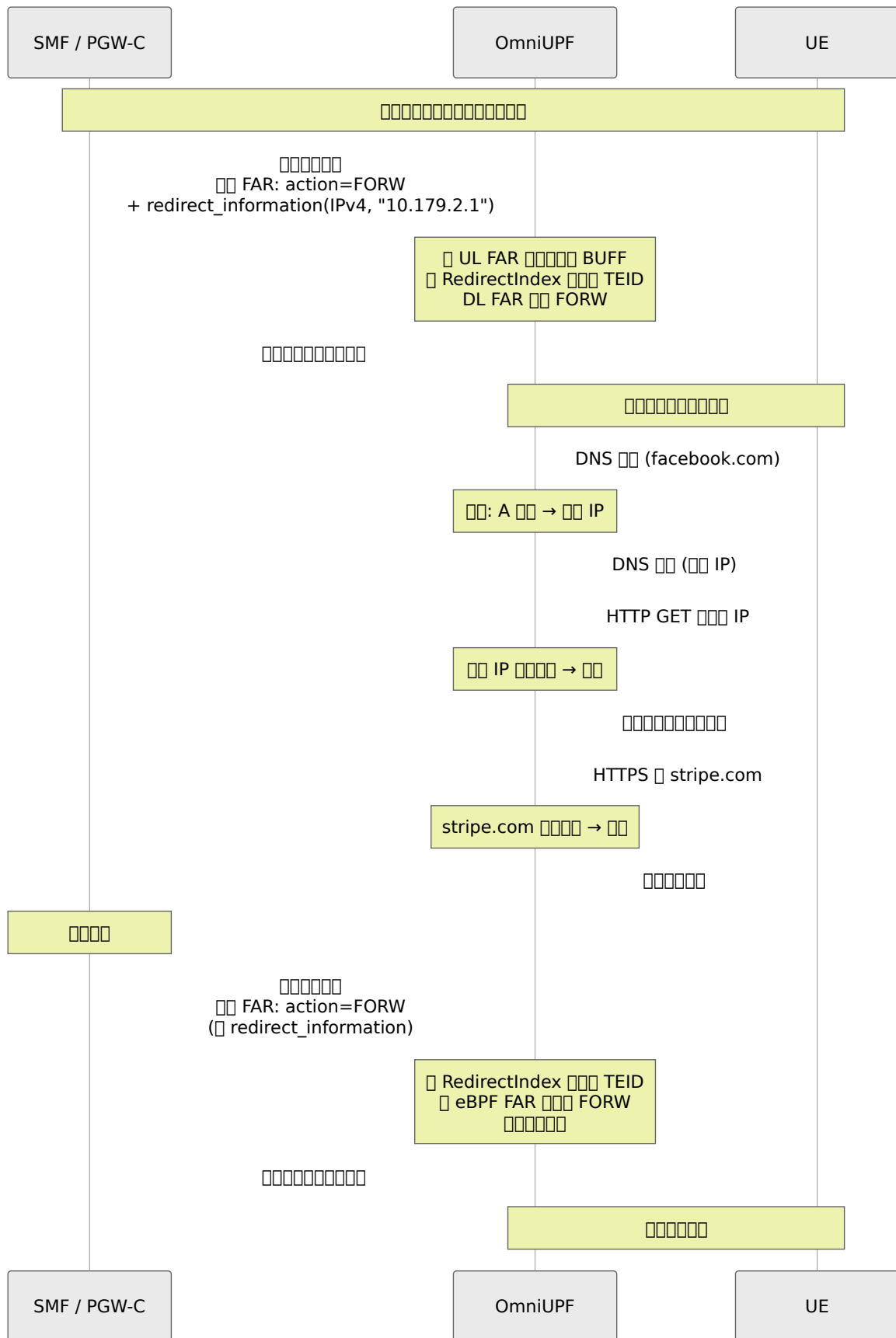
4G EPC 網元OmniUPF 網元 **SGW-U** 網元 **PGW-U**SGW-C 網元 PGW-C 網元
 PCRF 網元SGW 網元 N9 接口 網元 PGW 網元 TEID 網元 N3 接口PGW
 網元

1. PGW TEID SEID PDR

2. 000000 PFCP 000000000000SGW 000000000000 FAR00 teid 0 PGW UL PDR
TEID 000000000000 outer_header_creation00 FAR 000 PGW 0 N9 00 FAR0
3. 00 PGW UL FAR 0 SGW 00 gNB 0 UL FAR 0 eBPF 00000000 BUFF0000000000
eBPF 000000000000 eNB 000000 SGW 0 TEID0000 PGW 0 TEID0
4. 00 **SGW** 00 **gNB** 0 **UL PDR** 0000 RedirectIndex0000 PGW UL PDR00SGW UL
PDR TEID 0 eBPF 000 eNB 0000000000000000
5. 00 **DL** 000000000000 UE00000000 **SGW** 000 **DL FAR**000 eNB 0 FAR0 remoteip
!= n3_address00DL TEID 000000000000000000000000000000000000 - 00000000000000000000
0000 eNB TEID 0000

000 **UPF** 000000 SGW0SMF 000 UPF 000000000000 UL FAR 0 TEID 00 - SGW 0000
000000000000 UL PDR 000000

PFCP ☐ ☐ ☐



?? PFCP IEs

SMF 向 FAR 发 forwarding_parameters 和 redirect_information 消息

IE	描述
apply_action	向 SMF 发送 FORW 和 UPF 的 BUFF
redirect_information	重定向信息
forwarding_parameters	和 redirect_information 一起 outer_header_creation

3GPP TS 29.244 的 8.2.20-1

名称	值	UPF 描述
IPv4	0	IPv4 地址 IP
IPv6	1	IPv6 地址 IP
URL	2	URL 地址 IP 地址 IP 地址 UPF 地址 URL 地址 -- 地址 API 地址 Web 地址
SIP URI	3	地址

地址

地址 DNS 地址 地址 IP 地址

--	--	--	--	--

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

IP

- 跨平台iOS/Android
- 支持Windows

□□□□ IP □□□□□□□□□□ URL □□□□□□□□□□□□□□/□□□□

□□□□□□□□ **IP** / □□□□□□□□

IP 192.168.1.100 HTTP Host 192.168.1.100

□ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □

1. 检查 DNS 是否 IP -- 检查连接
(connectivitycheck.gstatic.com) captive.apple.com
www.msftconnecttest.com)
2. UE 检查 HTTP 是否 IP 是否 Host 是否 Host:
connectivitycheck.gstatic.com 检查连接

3. UPF 部署環境構築完了確認 Host 確認

確認 IP 確認環境構築完了確認 -- 確認環境構築完了確認 Host: 確認環境構築完了確認 IP 確認環境構築完了確認 -- 確認環境構築完了確認
確認 Host 確認環境構築完了確認確認環境構築完了確認確認環境構築完了確認 IP 確認環境構築完了確認CRM確認
確認環境構築完了確認

確認環境構築完了確認

- 確認環境構築完了確認 IP 確認 IP 確認環境構築完了確認 Host確認
- 確認環境構築完了確認確認環境構築完了確認確認環境構築完了確認 Host: 確認環境構築完了確認

確認 確認 walled_garden_portal_ip 確認 URL 確認 redirect_information 確認確認
確認 確認 HTTP 確認 // 確認 IPUPF 確認環境構築完了確認 Host 確認環境構築完了確認 Host 確認

確認

確認環境構築完了確認確認環境構築完了確認 .deb 確認環境構築完了確認 /etc/omniupf/runtime.exs 確認確認
確認 (rel/env.sh) 確認環境構築完了確認確認 RELEASE_CONFIG_DIR=/etc/omniupf 確認
Erlang 確認環境構築完了確認 config/runtime.exs 確認環境構築完了確認 UPF

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

# 
walled_garden_enabled = true

#  IP  / 
walled_garden_portal_ip = "10.179.2.1"

#  IPv6  IPv6 
walled_garden_portal_ip6 = "fd00:179:2::1"

#  DNS 
walled_garden_dns_resolver = "8.8.8.8"

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


11

項目	型別	デフォルト値	説明
walled_garden_enabled	bool	false	Walled Garden 機能を有効にするかどうか。true なら有効、false なら無効。FAR 経由で redirect_info を取得する。
walled_garden_portal_ip	string (IPv4)	10.179.2.1	Walled Garden のポータル IP アドレス。redirect_info の URL のスキーマは SIP URI であり、IPv4/IPv6 のいずれかである。URL の形式は [scheme]://[ip]:[port]
walled_garden_portal_ip6	string (IPv6)	nil	Walled Garden のポータル IPv6 アドレス。Walled Garden 経由で IPv4 の URL を取得する。
walled_garden_dns_resolver	string (IPv4)	8.8.8.8	Walled Garden 経由で DNS 解決を行う際の DNS サーバの IP アドレス。
walled_garden_whitelist	list	[]	Walled Garden 経由で DNS 解決を行う際の DNS サーバの IP アドレスのリスト。黄色の背景色。

--	--	--	--	--

IP UPF SMF -- SMF

ドメイン

ドメイン	サブドメイン	ターゲット
stripe.com	stripe.com api.stripe.com js.stripe.com	evilstripe.com notstripe.com
*.stripe.com	api.stripe.com js.stripe.com dashboard.stripe.com	stripe.com evilstripe.com
hcaptcha.com	hcaptcha.com newassets.hcaptcha.com	evihcaptcha.com

ドメイン `stripe.com` のサブドメイン `foo.stripe.com` をターゲットとして
`evilstripe.com` をドメインとして

IP

DNS を変更して IP を 203.0.113.7 に変更する

1. `api.stripe.com`
2. DNS を変更して `203.0.113.7`
3. `203.0.113.7` の IP を変更
4. HTTP/HTTPS を `203.0.113.7` に変更

IP を変更する

ドメイン

ドメイン `Stripe` の `hCaptcha`

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

Whitelisted URLs

PFCEP 消息中 消息头部分 SMF 的 FAR 的 `redirect_information` IE 消息头

- **IPv4** 消息头 IP 地址部分 IPv4 地址
- **IPv6** 消息头 IPv6 地址部分 IPv6 地址
- **URL** 消息头 FAR 消息头部分 DNS 消息头 IP 地址 IP 地址 URL 地址 -- 地址
消息 API 消息

消息头部分 MVNO 消息头部分

Item	SMF redirect_information	Source IP	Destination IP
A	IPv4: 10.179.2.1	10.179.2.1	10.179.2.1
B	IPv6: 2001:db8::1	2001:db8::1	2001:db8::1
C	URL: https://topup.mvno.com/recharge	topup.mvno.com IP	https://topup

UPF 配置 IP 和 URL API

API

配置 /v1/walled_garden 和 /api 的 api_port 为 8080 的 Phoenix HTTP

GET /v1/walled_garden

返回 IP 和 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": "203.0.113.7", "type": "resolved"},
    {"ip": "203.0.113.6", "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 리다이렉트 PFCP 세션 SEID / API 리다이렉트 - 리다이렉트 PFCP 세션 SEID FAR ID
리다이렉트 redirect_information API 리다이렉트 리다이렉트 SMF 리다이렉트

리다이렉트

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

項目	項目	項目	項目
seid	項目	項目	項目項目 PFCP 項目項目 SEID
url	項目 項目	項目	項目項目 - 項目 IPv4/IPv6 項目 URL項目項目 URL項目項目項目項目項目 IP 項目項目 IP項目項目 URL 項目項目項目 API 項目項目

項目項目**200 OK**項目

```
{
  "status": "redirect activated",
  "info": {
    "seid": 1,
    "sgw_seid": 7,
    "portal_ip": "10.179.2.1",
    "ue_ip": "10.60.0.1",
    "gnb_ip": "10.179.1.21",
    "ul_teids": ["0x4000", "0x4006"]
  }
}
```

sgw_seid 項目項目 N9 項目項目SGW+PGW 項目項目項目項目項目 TEID 項目 BUFF 項目項目 ul_teids
項目項目項目項目項目項目項目項目項目 TEID項目

項目項目

項目	項目
404	項目項目項目SEID 項目項目項目 PFCP 項目項目
400	項目項目項目

DELETE /v1/walled_garden/:seid

刪除指定 SEID 的遠端策略。FAR 類型為 N9，SGW 類型為 FAR，eBPF 類型為遠端策略。action=FORWARD，TEID 為遠端策略 ID。

請求參數 :seid - 遠端策略 SEID

響應 200 OK

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

響應

狀態	響應
404	未找到

GET /v1/walled_garden/whitelist

獲取遠端策略 IP 列表。IP 列表包含 DNS 類型 IP 列表和 API 類型 CIDR 列表。

響應

```
{
  "ips": [
    {
      "ip": "10.179.2.1",
      "type": "portal"
    },
    {
      "ip": "203.0.113.7",
      "type": "resolved"
    }
  ],
  "cidrs": ["192.168.100.0/24"]
}
```

POST /v1/walled_garden/whitelist

添加遠端策略 IP 列表。CIDR 列表包含遠端策略 UPF 類型 IP 列表 - 遠端策略 runtime.exs 列表。walled_garden_whitelist

IP

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

CIDR

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

ip cidr CIDR 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

upf_walled_garden_active_redirects

```
# upf_walled_garden_active_redirects
```

111

```
00: upf_walled_garden_packets_intercepted_total 00: 000 00: 0000000000000000
0000000000000000
```

```

00: upf_walled_garden_packets_dropped_total 00: 000 00: 0000000000000000
0000 DNS 000

```

```
00: upf_walled_garden_packets_forwarded_total 00: 000 00:
```

- `dst_ip` - 目标 IP 地址: 目标 IP 地址

```
00: upf_walled_garden_bytes_forwarded_total 00: 000 00:
```

- `dst_ip` - 目标 IP 地址: 目标 IP 地址 (32 位无符号整数)

```
00: upf_walled_garden_dns_spoofed_total 00: 000 00:
```

- **domain** - 指定するドメイン: 指定された DNS 名前空間

```
00: upf_walled_garden_dns_forwarded_total 00: 000 00:
```

- domain** - 指定されたドメイン: 指定された DNS 名前空間

Queries

```
# Count of active redirects
upf_walled_garden_active_redirects

# Rate of intercepted packets per second
rate(upf_walled_garden_packets_intercepted_total[5m])

# Total dropped packets
rate(upf_walled_garden_packets_dropped_total[5m])

# Rate of forwarded bytes per second, summed by destination IP
sum by (dst_ip)
(rate(upf_walled_garden_bytes_forwarded_total[5m]))

# Top 5 destination IPs by forwarded bytes rate
topk(5, sum by (dst_ip)
(rate(upf_walled_garden_bytes_forwarded_total[5m])))

# Top 10 domains by spoofed DNS rate
topk(10, sum by (domain)
(rate(upf_walled_garden_dns_spoofed_total[5m])))

# Top domains by forwarded DNS rate
sum by (domain) (rate(upf_walled_garden_dns_forwarded_total[5m]))

# Ratio of dropped to intercepted packets
sum(rate(upf_walled_garden_packets_dropped_total[5m]))
/ sum(rate(upf_walled_garden_packets_intercepted_total[5m]))
```

Queries

Queries

1: Count of active API endpoints

2:

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

配置:

1. 配置 Portal IP 地址 `curl http://<portal_ip>/`
2. 配置 Portal URL 为 Apple 的 `GET /hotspot-detect.html`
3. 配置 `GET /v1/walled_garden` 接口
4. 配置 Prometheus 的 `upf_walled_garden_dns_spoofed_total` 接口
5. 配置 GTP-U 地址为 DL FAR 的 FORW

配置:

配置: 配置 Portal 地址为 Stripe 的

配置:

- 配置 Portal 地址
- 配置 Portal 的 CDN IP 地址
- 配置 Portal 的 URL

配置:

1. 配置 `GET /v1/walled_garden` - 配置 `whitelisted_ips` 接口 IP
2. 配置 Prometheus: `upf_walled_garden_dns_forwarded_total{domain="stripe.com"}` 接口
3. 配置 Portal 地址为 `js.stripe.com` 和 `m.stripe.network`
4. 配置 `upf_walled_garden_bytes_forwarded_total` 和 `dst_ip` 接口

配置:

配置: SMF 配置 `redirect_information` 接口

配置:

- `walled_garden_enabled` 为 `false`

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:

304: HTTP 304 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

304: GET /v1/walled_garden CRM curl Host:

```
curl -s -H 'Host: portal.example'
http://<portal_ip>/ # ->
curl -s -H 'Host: connectivitycheck.gstatic.com'
http://<portal_ip>/ # ->
curl -s -H 'Host: <portal_ip>'
http://<portal_ip>/ # ->
```

例: 当 IP 为 10.10.10.10 时, 通过 IP 地址 10.10.10.10 UE 向 HTTP 服务器 10.10.10.10 发起 Host 请求, Host: connectivitycheck.gstatic.com 通过 Web 服务器 10.10.10.10 Host 请求, 通过 -- 请求 UPF 请求 Host 请求, 通过 10.10.10.10 请求。

说明:

1. 通过 IP 地址 IP 请求 Host: 请求。
2. 通过 IP 地址 IP 请求 Host: 请求。
3. 通过 curl 请求 IP 请求。

配置说明

通过 IP 地址 IP 请求 IPProto_RAW 请求 255 请求 Erlang 请求 :socket 请求 IP 请求 eBPF 请求 sendto 请求。

说明: 当 GTP-U 请求 IP 请求 IP 请求 UE IP 请求 IP 请求 IPProto_RAW 请求 IP 请求 UPF 请求 N6 请求/请求。

说明

項目	設定	説明
権限	<code>EPERM</code> - 権限不足 root <code>CAP_NET_RAW</code>	OmniUPF が root 権限で <code>CAP_NET_RAW</code> 権限
ネットワークインターフェース UE	N6 インターフェース	UPF インターフェース
ログ <code>Walled garden: raw socket open failed</code>		<code>systemd</code> で <code>AmbientCapabilities=CAP_NET_RAW</code>
ネットワーク UE 側 IP 設定	N6 側 NAT 設定	ネットワーク UE IP 設定 UPF 側

UPF 側で `Walled garden forward failed` や `raw socket open failed` のような
ログ `upf_walled_garden_packets_forwarded_total` や
`upf_walled_garden_bytes_forwarded_total` Prometheus で監視する

Prometheus 監視

例: Prometheus の `dst_ip` や `domain` のようなラベルを Prometheus に登録する

```
# Prometheus /24 ネットワーク 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. 專案架構
3. 環境建置
4. 資料庫設計
5. 後端 API 開發
6. 前端 UI 開發
7. 測試
8. 部署
9. 維護
10. XDP 開發
11. 總結

簡介

OmniUPF Web UI 是一個基於 Phoenix LiveView 開發的 Web 介面，用於管理 5G Core Network 的 UPF 設備。

- 支援 PFCP 協議 PDU 操作
- 支援 PDR、FAR、QER 及 URR 的設定
- 提供設備狀態監控功能
- 支援 eBPF 規則管理
- 提供系統配置管理

安裝

透過 REST API 安裝 OmniUPF 系統。

- 支持 PFCP 功能
- 支持网络切片
- 支持网络切片管理
- 支持网络切片
- 支持 eBPF 功能

网络切片

网络切片

支持 HTTPS 的 OmniUPF 部署

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

支持 443 端口的 HTTPS

部署

部署文件 `config/config.exs` 支持 OmniUPF 部署

部署文件 `UPF` 部署

`upf_hosts` 支持 UI 部署 OmniUPF 部署

部署

部署文件

- `/sessions` - PFCP 功能
- `/rules` - PDR、FAR、QER、URR 功能
- `/buffers` - 支持网络切片
- `/statistics` - 支持 XDP 功能
- `/capacity` - eBPF 功能
- `/upf_config` - UPF 部署

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

URL`/sessions`

OmniUPF `PFCP`

PFCP

`PFCP` `SMF/PGW-C`

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

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

`PFCP` `UE PDU`

項目	説明
UE SEID	UPF ユニーク識別子
SM SEID	SMF ユニーク識別子
UE IP	IPv4 または IPv6 アドレス
TEID	GTP-U トンネル ID
PDRs	パケット検出ルール
FARs	フィルタリングルール
QERs	QoS パラメータ
URRs	ユーティリティルール
その他	その他のパケット検出ルール

パケット

- UE IP アドレスを UE IP とする
- TEID トンネル ID とする
- PDR/FAR/QER/URR JSON を取得
- 10 秒間隔

パケット検出

パケット検出“パケット”パケット検出

- パケット検出 (PDRs) トンネル ID、UE IP、FAR ID、QER ID、SDF パケット検出 JSON
 - PDR ID トンネル ID - トンネル ID トンネル ID トンネル ID トンネル ID トンネル ID
 - PDR トンネル ID $\neq$ 0 トンネル ID トンネル ID
 - PDR トンネル ID トンネル ID トンネル ID
 - PDR トンネル ID トンネル ID トンネル ID
- パケット検出 (FARs) トンネル ID トンネル ID

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

Network Elements *PDRs* *FARs* *QERs*

UE

UE **UE** **UE**

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

Network Elements

- **UE**
- **UPF** **UE**
- **UE**

UE Information

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

UPF

UPF 10 QoS rules

- UPF QoS rules
- UPF QoS rules
- QoS rules

UPF

URL /rules

UPF QoS rules

PDR rules - QoS rules

UPF PDR rules

PDRs (N3 → N6):

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

PDRs (N6 → N3):

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

IPv6 00 PDRs0

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

FAR 000 - 0000000

0000 FAR 000000000000

000

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

0	00
FAR ID	000000000000
00	00000000FORWARD0DROP0BUFFER0DUPLICATE0NOTIFY0
00	000000000000/000
00	0000000000TEID0IP 000

FAR 00000

- **FORWARD (1)**000000000000

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

1.

- “**QoS**” “**QoS**”
-
- eBPF FAR

QER - QoS

QoS

1.

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

QER ID	QoS PDR
MBR	kbps
MBR	kbps
GBR	kbps
GBR	kbps
QFI	QoS 5G

QoS

- **MBR = 0** 
- **GBR = 0** 
- **GBR > 0** 

URR -

□ □ □ □ □ □ □ □ □ □ □ □ □ □

111

- 同様の URR ID を持つ同様の URR
- 同様の PDR と同様の URR ID を持つ同様の URR
- 同様の PDR と同様の URR を持つ同様の
- 同様の URR と 100 と 1000

項目	説明
URR ID	URR ID (PDR 参照番号)
UE	UE (UE 参照番号)
UE	UE (UE 参照番号)
UE	UE (UE 参照番号)
UE	UE (UE 参照番号)
UE	UE (UE 参照番号)

□□□□□

- BKBMBGBTB
-
-

111

- 00000000 URR
- 00000000 0 0000 URR000000

○○○○○○○○○○○○○○○○○○

- ○○○○○○ FAR ○○○○○○○○
- ○○○○○○○○○○○
- ○ **FARs**○○○○○○○○ FAR ○○
- ○○ **FAR** ○○○○○○○ FAR ○○○○○○○○○
- ○○○○○○○○○○○○○
- ○○○ **TTL**○○○○○○○○○○○○

○ **FAR** ○○○○

○○○○○○○○○ FAR ○○○

○	○○
FAR ID	○○○○○○○○○
○○○○○	○ 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

概要

バージョン 5 からサポートされています。

インストール

URL `/statistics`

特徴

OmniUPF は Prometheus をサポートしています。 👉

特徴

特徴

- 柔軟なポリシー
- 柔軟なポリシー
- 柔軟なポリシー
- **GTP-U** をサポート GTP-U をサポート

OmniUPF は UPF をサポートしています。

特徴

特徴

- 柔軟なポリシー
- 柔軟なポリシー
- 柔軟なポリシー

特徴

XDP

eXpress Data Path をサポート

- **XDP** をサポート XDP をサポート

- **XDP** 可程式化網路封包處理
- **XDP** 可程式化 XDP 封包處理
- **XDP** 可程式化 XDP 封包處理

可程式化 XDP 封包處理

XDP 可程式化

- 可程式化網路封包處理
- eBPF 可程式化
- 可程式化網路封包處理
- 可程式化

N3/N6 可程式化

可程式化網路封包處理

N3 可程式化 RAN 可程式化

- 可程式化 **N3** gNB/eNodeB 可程式化
- 可程式化 **N3** gNB/eNodeB 可程式化

N6 可程式化網路封包處理

- 可程式化 **N6** 可程式化網路封包處理/IMS 可程式化
- 可程式化 **N6** 可程式化網路封包處理

可程式化網路封包處理

可程式化網路封包處理

可程式化

可程式化

1. 可程式化網路封包處理/可程式化
2. 可程式化網路封包處理
3. 可程式化 **N3** 可程式化 **N6** 可程式化

問題集

1. 問題集
2. 問題 XDP
3. 問題

問題

1. 問題 XDP
2. 問題 XDP
3. 問題 N3/N6

問題

1. 問題
2. 問題 UPF
3. 問題

問題

問題 5

問題

URL/capacity

問題

問題 UPF 問題 eBPF

eBPF 問題

問題 eBPF

項目	説明
概要	eBPF を利用して uplink_pdr_map と far_map
目的	パケットの転送先を決定する
対象	パケットの転送先を決定する
前提条件	パケットの転送先を決定する
手順	パケットの転送先を決定する
結果	パケットの転送先を決定する

概要

パケットの転送先を決定する

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

パケットの転送先を決定する

uplink_pdr_map

- パケット TEID をパケット PDR
- パケット転送率を決定する
- パケット転送率を決定する

downlink_pdr_map / downlink_pdr_map_ip6

- パケット UE IP をパケット PDR
- パケット UE IPv4/IPv6 をパケット PDR
- パケット転送率を決定する

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. ــ
3. ــ

ــــــــــــــــ

eBPF ــــــــــــــــ UPF ــــــــــــــــ UPF ــ

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

ــــــــــــــــ

ــــــــــــــــ = (ــــــــــــــــ + ــــــــــــــــ) × ــــــــ

ــــــــــــــــــــــــ 100 ــــــــــــــــ 64 ــــــــــــــــ PDR ــــــــــــــــــــــــ 64 MB ــــــــــــــــــــــــ

ــــــــــــــــ

ــــــــــــــــ 10 ــــــــــــــــــــــــ

ــــــــــــــــ

URLــــــــ /upf_config

ــــــــ

ــــــــــــــــ UPF ــ

UPF ــــــــ

ــــــــــــــــ UPF ــــــــ

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

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

通過**eBPF**通過

通過通過通過通過通過

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

通過通過通過 eBPF 通過通過通過通過通過通過通過通過

通過

通過 **UPF** 通過

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

通過通過通過

1. 通過通過通過通過通過通過
2. 通過通過通過
3. 通過通過

通過通過

1. 通過 UPF 通過通過
2. 通過通過通過通過
3. 通過

通過通過

URL `/routes`

概要

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

前提条件

本ドキュメントは、以下の環境で動作します。

- OSPFとBGPがインストールされている。
- UEのIPアドレスが設定されている。
- OSPFとBGPの構成が正しい。
- ネットワークの接続が正常である。

UE IP の設定

UEのIPアドレスを設定するための手順は以下の通りです。

項目	設定値
IP アドレス	192.168.1.1
UE IP の設定	OSPF と BGP の両方で IPv4 と IPv6 を設定する

手順

- OSPF と BGP の両方で UE の IP アドレスを設定する。
- OSPF と BGP の両方で IP アドレスの範囲を設定する。
- OSPF と BGP の両方で IP アドレスのマスクを設定する。

OSPF の設定

OSPFの設定は、以下の手順で行います。

項目	説明
Router ID	OSPF の識別子
Interface	OSPF を有効にする IP アドレス
Area	OSPF のエリア
Cost	OSPF のコスト
Priority	OSPF の優先度
Authentication	OSPF の認証
Timers	OSPF のタイマー

OSPF の設定

- Router ID の設定
- Interface の設定

BGP の設定

BGP の設定

項目	説明
IP	BGP 宛先 IP
ASN	自治システム番号
経路	BGP 経路
入/出	入/出トラフィック
接続数	接続数
接続タイプ	接続タイプ BGP
接続タイプ	接続タイプ BGP

BGP 経路

- 経路の BGP 経路
- 経路の BGP 経路

経路の BGP 経路 ID は ASN の BGP 経路

OSPF 経路

経路の UE の OSPF の LSA の経路

項目	説明
ルータ ID	LSA の識別子
タイプ	LSA の種類
リンク ID	リンクの識別子
リンクタイプ	OSPF のリンクタイプ (E1, E2)
コスト	OSPF のコスト
フラグ	LSA のフラグ
メトリック	LSA のメトリック

機能

- ルータ UE の OSPF
- リンクタイプ
- LSA のフラグ

設定

設定

- FRR の設定
- UE の設定
- リンクタイプ

設定

- OSPF の設定
- OSPF と BGP の設定

11

--	--	--	--	--	--	--	--	--

1. 配置路由
2. 配置 OSPF 邻居关系
3. 配置 BGP 邻居关系
4. 配置路由策略/路由映射

UE

1. 四台 UE IP 地址分别为 UE
2. 四台 OSPF 网络地址
3. 四台 UE 网络地址分别为 LSA 四
4. 四台网络地址分别为 UPF 四

□ □ □ □ □ □ □ □ □ □

1. 配置 OSPF 路由
2. 配置 BGP 路由
3. 配置静态路由
4. 配置 OSPF/BGP 路由

UPF

1. 配置 OSPF 进程
2. 配置接口 IP 地址
3. 配置 OSPF 接口
4. 配置 BGP 邻居

□ □ □ □ □ □ □

1. UE 注册失败
2. 鉴权失败
3. UE OSPF LSA 异常
4. UE BGP 异常

環境

環境 10 環境環境環境環境環境環境 UE 環境

環境

環境環境 UPF 環境 FRR環境環境環境環境

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

XDP 環境

URL環境/xdp_capabilities

環境

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

環境

環境環境環境環境環境

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

XDP 概要

概要として XDP を使ったネットワーク処理

XDP_DRV 概要

- 100~5-10 Mpps 対応
- ネットワークドライバ XDP を利用
- ネットワーク XDP 対応 NIC 対応 i40e ixgbe mlx5 など
- ネットワークドライバ XDP を利用
- ネットワークドライバ XDP を利用 XDP XDP XDP

XDP_SKB 概要

- 100~1-2 Mpps
- ネットワークドライバ XDP を利用
- ネットワークドライバ XDP を利用
- ネットワークドライバ XDP を利用
- ネットワークドライバ XDP を利用

概要として XDP を使ったネットワーク処理

- ネットワーク XDP を利用
- ネットワークドライバ XDP を利用

概要として XDP を使ったネットワーク処理

- ネットワークドライバ XDP を利用
- ネットワークドライバ XDP を利用

XDP *XXXXXXXXXXXXXXXXXXXX Mpps XXXXXXX*

XX

XXXXXXXXXXXXXXXXXXXX

XXXXXXXXXX

- "✓ XXXXX XDP_DRV XXXXXXXXXXXXXXX"
- XXXXXXXXXXXXXXX

XXXXXXXXXX

- "△ XXXXX XDP_DRV XXXXXXXXXXXXXXX"
- XXXXXXXXXXXXXXXXXXXXXXX
- "△ XXXXXXXXXXXXXXX XDP_DRV"
- XXXXXXXXXXXXXXX

XXXXXXXXXX

- XX XDP XXXXXXX

Mpps 計算式

計算式は Mpps と Gbps の関係

計算式

計算式 Mpps

- 0.1 - 100 Mpps
- XDP Mpps
-

計算式

- 64 - 9000
- 1200 GTP
- GTP

計算式

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

計算式

計算式 Gbps

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

計算式 Gbps

-

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

1000000

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

100000

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

100 Mpps

1000000000000000

1000 Mpps

- 100000000
- 1000000000000000
- 1000000000

100000000

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

GTP 100000

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

□□

□□ **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. □□□□□□□□□□□□□□□□

配置

配置 XDP_DRV

- 配置 XDP 驱动 NIC Intel i40e/ixgbe Mellanox mlx5
- 配置 NIC 驱动程序
- 配置 RSS 驱动程序
- 配置 NIC 驱动程序

配置 XDP_SKB

- 配置
- 配置
- 配置

配置

- 配置 CPU 驱动程序
- 配置
- 配置 RSS 驱动程序

配置

XDP 配置 30 配置

配置

URL </logs>

配置

配置 OmniUPF 配置

配置

- 配置 Phoenix LiveView 配置
- 配置

- 設定ファイル
- 設定ファイルのバックアップ

設定

OmniUPF の設定 Elixir Logger の設定

- **DEBUG** デバッグログ
- **INFO** インフォメーションログ
- **WARNING** 警告ログ
- **ERROR** エラーログ

ログ

ログの出力先

1. 標準出力
2. SMF ログ
3. PCF PFCP ログ

PCF PFCP ログ

1. PCF PFCP ログ
2. ログレベル/フォーマット
3. ログ出力先

ログの出力先

1. ログレベル/フォーマット
2. eBPF ログ
3. FAR/PDR ログ

网络层

网络层

网络层

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

网络层

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

网络层

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

网络层

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

网络

- 网络层的主要功能 5-10 网络层
- 网络层的主要设备

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

5G Core Network

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

OmniUPF 与 XDP 部署指南

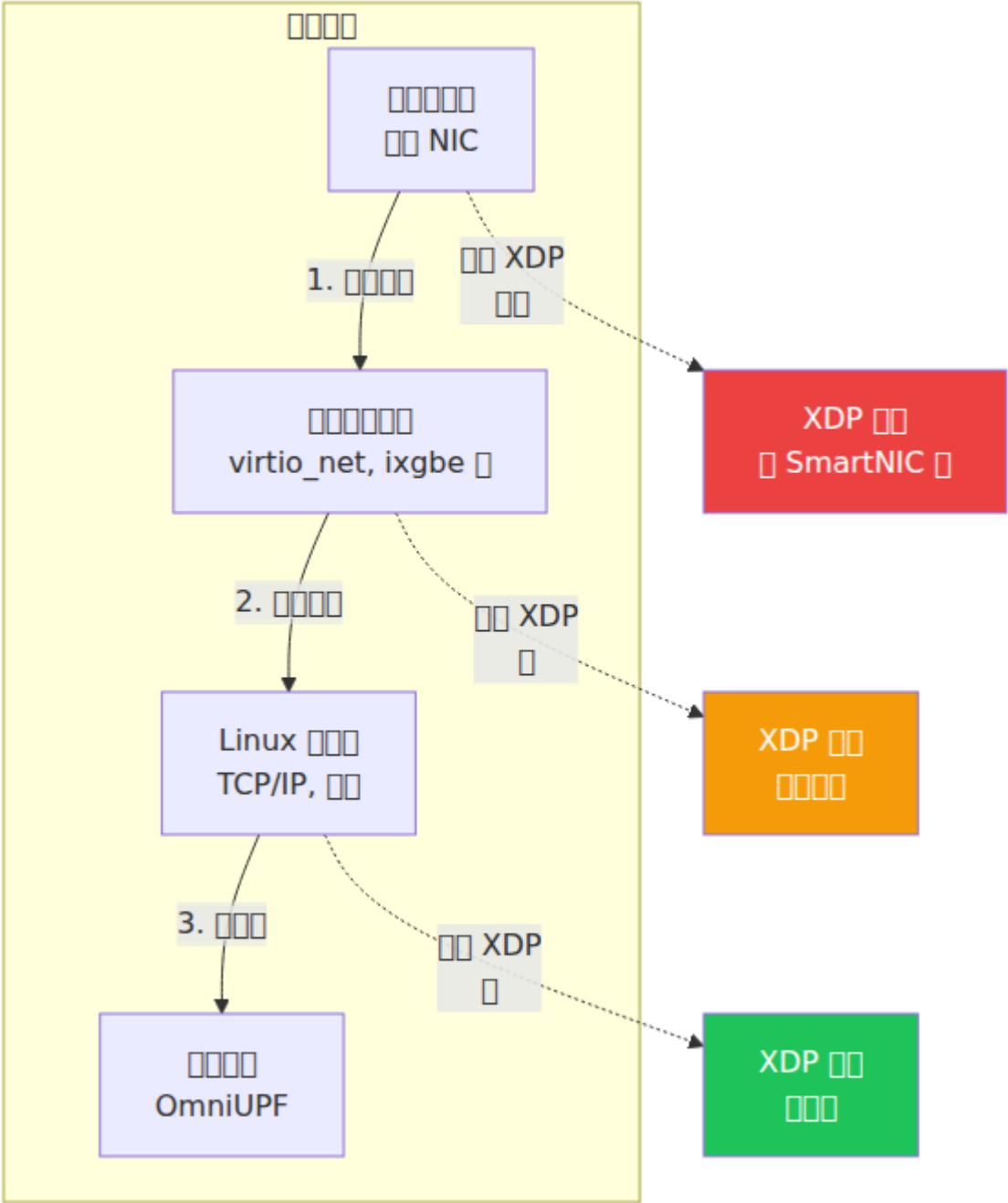
目录

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

简介

OmniUPF 与 **XDP (eXpress Data Path)** 是 Linux 内核中用于高性能网络处理的技术。XDP 允许在用户空间或内核空间直接处理网络数据包，绕过传统的网络栈，从而显著降低延迟并提高吞吐量。

XDP 与 **eBPF** 结合使用，可以实现灵活的网络策略和流量控制。



XDP 在 OmniUPF 之前处理数据包

XDP 对比

对比项	Linux 对比	对比项	对比项
对比项	Linux 对比	对比项	NIC 对比
对比项	~1-2 Mpps	~5-10 Mpps	~10-40 Mpps
对比项	~100 μs	~10 μs	~1 μs
CPU 对比	对比	对比	对比
NIC 对比	对比 NIC	对比 XDP 对比	对比 XDP 对比 SmartNIC
对比项 对比项 对比项	对比项	对比项	对比PCI 对比
对比项	对比项	对比项	对比???
对比项	xdp_attach_mode: generic	xdp_attach_mode: native	xdp_attach_mode: offload

对比项 对比项

对比项

对比

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

□ □ □ □

概要

- 処理速度 ~5-10 万pps (Mpps)
- 遅延 ~10 μs
- **CPU** 依存
- 高負荷 CPU 高 NIC 依存

特徴

- 柔軟性
- 拡張性
- 高スループット
- 低遅延

NIC 種類

ソフトウェア XDP 高速 XDP 専用 NIC 高速 XDP

NIC 種類

- Intel `ixgbe` 10G `i40e` 40G `ice` 100G
- Broadcom `bnxt_en`
- Mellanox `mlx4_en` `mlx5_core`
- Netronome `nfp`
- Marvell `mvneta` `mvpp2`

NIC 種類

- VirtIO `virtio_net` KVM Proxmox OpenStack ✓
- VMware `vmxnet3` ✓
- `hv_netvsc` Hyper-V ✓
- `ena` AWS ✓
- SR-IOV `ixgbevf` `i40evf` PCI 高速 ✓

VirtualBox は XDP 対応しない

11

```
# /etc/omniupf/runtime.exs
xdp_interfaces = "eth0"
xdp_attach_mode = "native"
```

NIC Proxmox

SmartNIC

11

00 XDP 0000 NIC 0000 SmartNIC0000 eBPF 00000000 CPU 000000000000000000000000
 000000

□ □ □ □

- 1000~10-40 Mbps (Mbps)
- 1000000 ~1 M
- **CPU** 10000000 NIC 10000

--	--	--	--

- 網路規格 網路規格 UPF 至少 10G 網路
- 網路設備網路設備
- 網路設備 網路設備 CPU 網路

□ □ □ □

Netronome Agilio SmartNIC XDP

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

XXXXXXXXXX PCI - XXXXXXXXXXXXXXXX

11

```
# /etc/omniupf/runtime.exs
xdp_interfaces = "eth0"
xdp_attach_mode = "offload"
```

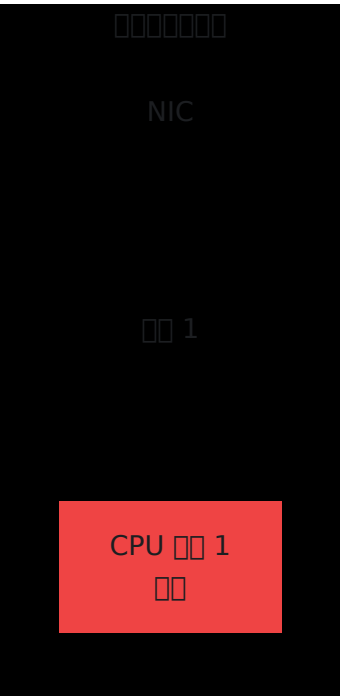
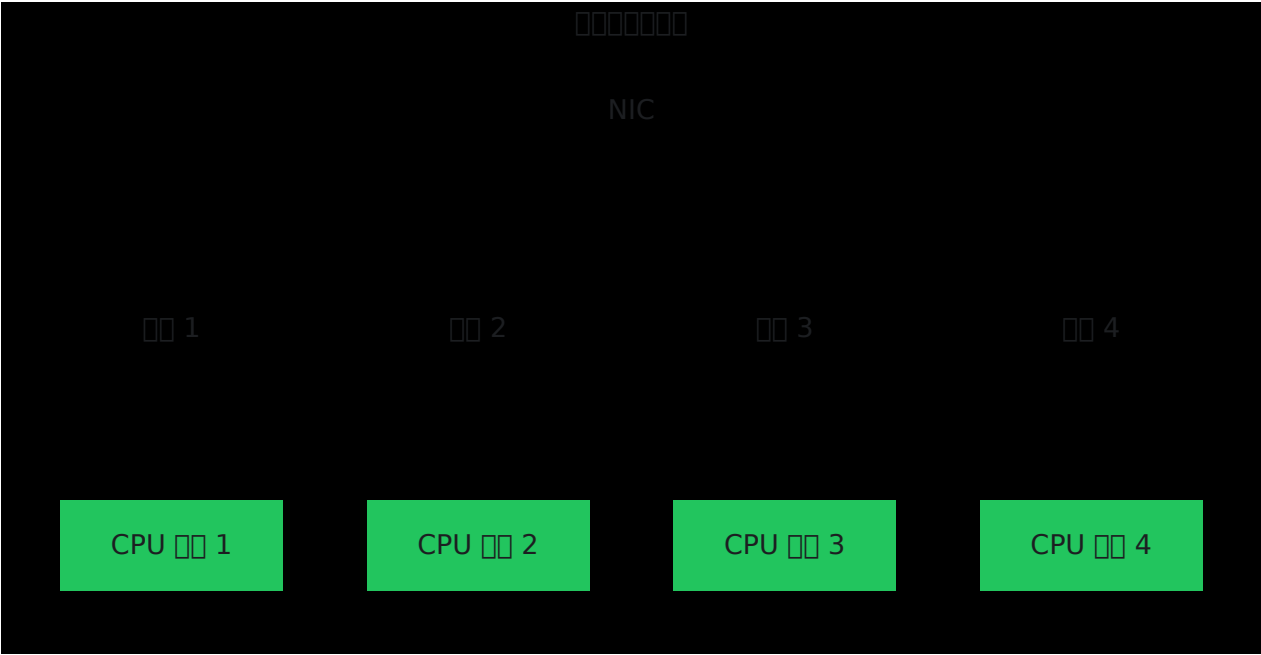
Proxmox VE XDP

Proxmox VE 上の **VirtIO** ネットワークカード virtio_net の XDP 実装

1

□□□□□□□□□□

- 16개의 프로세서가 CPU를 공유 → 병목
- 16개의 프로세서가 CPU를 공유 → 병목



第 2 節 Proxmox 安裝

安裝 AArch Proxmox Web UI

1. 安裝 Proxmox Web UI
 - 在 Proxmox Web 安裝 Proxmox Web UI
 - 安裝 Proxmox

2. 設定

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

3. 設定

- 名前空間名を設定
- 名前空間に **8** vCPU を割り当てる
- 名前空間を設定

4. 設定

- 名前空間を設定

Proxmox での操作

```
# SSH 接続 Proxmox へ

# 名前空間 ID
qm list

# 名前空間 XXX を作成
qm set XXX -net0 virtio=XX:XX:XX:XX:XX:XX,bridge=vmbr0,queues=8

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

# 名前空間をシャットダウン
qm shutdown XXX

# 名前空間を再起動
qm start XXX
```

注意事項

- **4** vCPU を割り当てる
- **8** vCPU を割り当てる
- **16** vCPU を割り当てる

3 設定ネットワーク

ネットワークSSH 接続可能にする

```
# ネットワーク
ethtool -l eth0

# 確認
# eth0 確認
# Combined:      8          <-- 確認
# ネットワーク
ls -ld /sys/class/net/eth0/queues/rx-* | wc -l
ls -ld /sys/class/net/eth0/queues/tx-* | wc -l

# ネットワーク 8確認
```

4 設定 OmniUPF ネットワーク XDP

OmniUPF 設定

```
# ネットワーク
sudo nano /etc/omniupf/runtime.exs
```

ネットワーク XDP 設定

```
# ネットワーク
xdp_attach_mode = "generic"

# ネットワーク
xdp_attach_mode = "native"
```

OmniUPF 再起動

```
sudo systemctl restart omniupf
```

5. 检查 XDP 是否安装

检查

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

# 检查
# xdp_attach_mode:native
# XDPAttachMode:native
# 检查 XDP 是否安装 "eth0" 是否 2
```

API 检查

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

# 检查
# "xdp_attach_mode": "native",
```

Proxmox 检查

检查“是否 XDP 安装”

检查

- 检查是否安装 `ethtool -l eth0`
- 检查是否 `uname -r` 是否 ≥ 5.15
- 检查 VirtIO 是否安装 `lsmod | grep virtio_net`

检查是否 1 检查

检查

- 检查是否 检查是否安装
- 检查 `qm shutdown XXX && sleep 5 && qm start XXX`
- 检查 Proxmox 是否安装 `grep net0 /etc/pve/qemu-server/XXX.conf`

VMware ESXi / vSphere

VMware

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

VMware ESXi / vSphere XDP

VMware ESXi / vSphere

VMware `vmxnet3` ネットワークアダプタ XDP

要件

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

手順

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

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

4. ネットワークアダプタを確認

```
ethtool -l ens192 # 查看网卡
```

安装 **OmniUPF** `runtime.exs` 文件

```
xdp_interfaces = "ens192" # VMware 网卡 ens192
xdp_attach_mode = "native"
```

KVM / libvirt 安装

安装 **virsh** 工具

```
# 查看虚拟机
virsh edit your-vm-name
```

配置网络

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

配置网络

```
ethtool -l eth0
```

安装 Hyper-V

Hyper-V 安装 `hv_netvsc` 驱动 XDP

安装

- Windows Server 2016 安装
- 安装 Linux 内核 4.3 版本

- 2 VM

Network

Hyper-V PowerShell

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

OmniUPF runtime.exs

```
xdp_interfaces = "eth0"
xdp_attach_mode = "native"
```

VirtualBox

VirtualBox XDP

VirtualBox e1000 virtio-net XDP

```
xdp_attach_mode = "generic" # VirtualBox
```

XDP

XDP

1. 检查 OmniUPF 日志

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

# 输出
# ✓ "xdp_attach_mode:native"
# ✓ "XDP 模式"
# ✗ "错误" 或 "警告"
```

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 次
  "xdp_drop": 1234,     # 丢弃
  "xdp_pass": 5678,     # 通过
  "xdp_redirect": 9012, # 重定向
  "xdp_tx": 3456        # 发送
}
```

4. 確認

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

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

5. 確認

確認

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

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

確認 XDP

“確認 XDP”

確認

```
確認 XDP 確認 eth0
```

確認

1. 確認

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

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

2. 確認

```
uname -r
```

```
# 5.15 以上 XDP
```

3. XDP 確認

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

```
# XDP 確認
```

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

確認

- 5.15+
- virtio_net 確認 `modprobe virtio_net`
- XDP 確認

確認

確認

```
確認 XDP 確認
```

確認

```
確認 dmesg 確認
```

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

準備

1. 仮想マシン **XDP**

- VirtualBox 仮想マシンで XDP
- 仮想 NIC 設定

2. 確認

- `ethtool -l eth0`
- 受信 > 1 メガバイト

3. 仮想 **XDP** 環境

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

# 確認
# CONFIG_XDP_SOCKETS=y
# CONFIG_BPF=y
```

環境

- 仮想マシン Proxmox 上で
- 仮想マシンで
- 仮想マシンで XDP

インストール

インストール

確認

1. インストール

```
# 查看网卡队列
ethtool -S eth0 | grep rx_queue

# 查看网卡队列
```

2. 查看 CPU 使用情况

```
# 查看系统 CPU 使用率
mpstat -P ALL 1

# 查看系统 CPU 使用率
```

3. 查看 XDP 配置和统计

```
# 查看 bpftool 配置
sudo bpftool net list

# 查看 XDP 配置
```

配置项

- 配置项 8-16 配置
- CPU 使用率  配置
- 配置项 CPU 使用率

查看 XDP 配置项 `xdp_aborted > 0`

配置

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

配置

XDP 和 eBPF 的入门教程

1. 安装 eBPF 依赖

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

2. 验证安装

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

# 测试 100% 成功
```

总结

- 了解 eBPF 的基本概念
- 了解 eBPF 的用途
- 了解 Linux 的 eBPF 支持

Proxmox 入门教程

使用 `ethtool -l eth0` 查看 1 个网卡

安装

1. 安装 Proxmox 虚拟机

```
# 在 Proxmox 中
grep net0 /etc/pve/qemu-server/YOUR_VM_ID.conf

# 设置 queues=8
```

2. 验证安装

```
# 在 Proxmox 中
qm status YOUR_VM_ID

# 等待 "status: stopped" 出现
```

等待

```
# 在 Proxmox 中
# 等待 1000ms
qm shutdown YOUR_VM_ID
sleep 10
qm start YOUR_VM_ID

# 检查网络接口
ethtool -l eth0
```

等待网络接口 eth0 出现

等待 XDP 出现

等待

等待 XDP 出现

等待

XDP 需要 `CAP_NET_ADMIN` 和 `CAP_SYS_ADMIN` 权限

等待

1. 以 **root** 身份运行 **OmniUPF**

```
sudo systemctl restart omniupf
```

2. 等待 **systemd** 启动

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

3. `docker` `--privileged`

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

Table

OmniUPF

	1.5 Mpps	8.2 Mpps	5.5
	95 μ s	12 μ s	8
CPU 1 Gbps	85% 1	15%	5
	~1.2 Gbps	~10 Gbps	8

XDP

Omnitouch 100%

XDP NIC

NIC OmniUPF XDP

Intel NICs (Intel)

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

Mellanox/NVIDIA NICs (InfiniBand)

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

Broadcom NICs

Model	Speed	Driver	XDP Support	Notes
BCM57xxx Series	10/25/50GbE	bnxt_en	Yes ✓	Used in Dell/HP servers

Other NICs (Various)

OS	NIC Type	Driver	XDP Support	Notes	OS
Proxmox/KVM	VirtIO	virtio_net	Yes ✓	Kernel bypass	Linux
VMware ESXi	vmxnet3	vmxnet3	Yes ✓	Kernel bypass	Linux ESXi 6.7 and later
Hyper-V	VMNIC	hv_netvsc	Yes ✓	Kernel bypass	Windows Server 2016 and later
AWS	ENA	ena	Yes ✓	Kernel bypass	EC2 Linux instances
VirtualBox	VMNIC	VMNIC	Yes ✓	Kernel bypass	Linux

Network Interface Cards (NICs)

Supports XDP and eBPF NICs

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

Supports NICs that support XDP

Supports

Supports OmniUPF

服务器 (1-10 Gbps)

- **NIC:** Intel X520 (10GbE 双口)
- 网络: 支持 XDP
- 网络: 支持 UPF 网络 8-10 Gbps
- 成本: 约 \$100-200/节点/节点

服务器 (10-50 Gbps)

- **NIC:** Intel X710 (40GbE) 或 Mellanox ConnectX-4 (25GbE)
- 网络: 支持 XDP
- 网络: 支持 UPF 网络 25-40 Gbps
- 成本: 约 \$300-800

服务器 (50-100+ Gbps)

- **NIC:** Mellanox ConnectX-5/6 (100GbE)
- 网络: 支持 XDP
- 网络: 支持 UPF 网络 80-100 Gbps
- 成本: 约 \$1,000-2,500

虚拟机 (Proxmox/KVM)

- **NIC:** VirtIO 网络 8-16 网络
- 网络: 支持 XDP
- 网络: 支持 UPF 网络 5-10 Gbps
- 成本: 相对较低

网络架构

网络架构 OmniUPF 网络

NIC/网卡	网卡	网卡
Realtek NICs	不支持 XDP 的 Linux 网卡	Intel i350 网卡
VirtualBox	不支持 XDP 的	不支持 Proxmox/KVM
网卡 NICs	不支持 XDP 的	不支持 Intel/Mellanox
网卡 NICs (<2014)	不支持 XDP 的	Intel X520 网卡

网卡

网卡

1. 网卡: 网卡 Linux 网卡 XDP

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

2. 网卡: 网卡 ≥ 5.15 网卡 XDP

```
uname -r
```

3. 网卡: 网卡 NIC 网卡 (RSS/VMDq)

4. 网卡 PCI 网卡: 网卡 PCIe 网卡

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

5. 网卡:

- 网卡: 网卡 NIC
- 网卡: 网卡 VirtIO 网卡 SR-IOV 网卡
- 网卡: 网卡 NIC 网卡

目次

- 概要: [CONFIGURATION.md](#) - 設定
 - トラブルシューティング: [TROUBLESHOOTING.md](#) - 問題解決
 - 概要: [ARCHITECTURE.md](#) - eBPF と XDP の仕組み
 - 概要: [MONITORING.md](#) - 監視とログ
-

インストール

Proxmox 上で XDP を実行 (TL;DR)

```
# Proxmox 上で実行
qm set <VM_ID> -net0 virtio=<MAC>,bridge=vmbr0,queues=8
qm shutdown <VM_ID> && sleep 10 && qm start <VM_ID>

# 設定
ethtool -l eth0 # 8 個のキュー
sudo nano /etc/omniupf/runtime.exs # xdp_attach_mode = "native"
sudo systemctl restart omniupf
journalctl -u omniupf --since "1 min ago" | grep xdp # ログを確認
```

❏ **XDP** ❏❏❏❏❏

```
# ❏❏❏❏  
curl -s http://localhost:8080/api/v1/config | grep xdp_attach_mode  
  
# ❏❏❏❏❏❏  
curl -s http://localhost:8080/api/v1/xdp_stats | jq  
  
# ❏❏❏❏  
ethtool -l eth0
```

OmniUPF API

概要

OmniUPF API は RESTful 形式で eBPF 形式の API を UPF 向けに提供します。

API

基本

- **PFCP** 形式で UE IP と TEID を
- **PFCP** 形式で

ルール

- ルール (PDR) (IPv4/IPv6)
- ルール (FAR) (IPv4/IPv6)
- QoS ルール (QER) (QoS)
- ルール (URR) (IPv4/IPv6)

操作

- ルール FAR (GET /buffer, GET /buffer/:far_id)
- ルール (POST /buffer/:far_id/flush, DELETE /buffer/:far_id, DELETE /buffer)
- ルール (POST /buffer/:far_id/notify)
- ルール DLDR (GET /buffer/notifications)

その他

- ルール (GTP, IP, TCP, UDP, ICMP, ARP)
- **XDP** ルール (IPv4/IPv6)
- **N3/N6** ルール (RAN)
- ルール FIB ルール (IPv4/IPv6)

API

- **UE** API UE IP gNB API (GET /routes)
- **FRR** 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
```

API ☐ ☐

11

OmniUPF API PDR URR

□□□□□□□□

1. 페이징 파라미터

- `page` 파라미터 1 부터
- `page_size` 파라미터는 100부터 1000

2. 오프셋 파라미터

- `offset` 파라미터
- `limit` 파라미터는 1000

예제

```
# 페이징 파라미터 50 사용
GET /api/v1/pfcp_sessions?page=2&page_size=50

# 오프셋 파라미터 100, limit 파라미터 50
GET /api/v1/pfcp_sessions?offset=100&limit=50

# 페이징 파라미터 100 사용
GET /api/v1/pfcp_sessions
```

결과

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

참고

- `/api/v1/pfcp_sessions` - PFCP 파라미터

- `/api/v1/pfcp_associations` - PCFPCP 关联
- `/api/v1/routes` - UE IP 路由
- `/api/v1/uplink_pdr_map` - 上行 PDR 映射
- `/api/v1/uplink_pdr_map/full` - 上行 PDR 完整 SDF 映射
- `/api/v1/downlink_pdr_map` - 下行 PDR IPv4 映射
- `/api/v1/downlink_pdr_map/full` - 下行 PDR IPv4 完整 SDF 映射
- `/api/v1/downlink_pdr_map_ip6` - 下行 PDR IPv6 映射
- `/api/v1/downlink_pdr_map_ip6/full` - 下行 PDR IPv6 完整 SDF 映射
- `/api/v1/far_map` - 远端策略规则映射
- `/api/v1/qer_map` - QoS 策略规则映射
- `/api/v1/urr_map` - 用户报告规则映射

缓冲管理

- `GET /api/v1/buffer` - 获取 FAR 缓冲配置
- `GET /api/v1/buffer/:far_id` - 获取 FAR 缓冲配置
- `GET /api/v1/buffer/notifications` - 获取 DLDR 通知
- `DELETE /api/v1/buffer` - 删除缓冲配置
- `DELETE /api/v1/buffer/:far_id` - 删除 FAR 缓冲
- `POST /api/v1/buffer/:far_id/flush` - 刷新缓冲配置
- `POST /api/v1/buffer/:far_id/notify` - 通知 DLDR 缓冲

配置

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

路由管理

- `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

其他

- Web UI 中 `page_size=100`
- 中 `page_size=1000` 中
- `pagination.total_pages` 中
- `page_size` 中 API 中

CORS 中

中 (CORS) 中 API 中 Web UI 中 API

Prometheus 中

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

中

- 中 PFCP 中
- 中
- XDP 中
- 中
- eBPF 中
- URR 中

中 中 中

中

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

UE 控制平面

控制平面

- API 平面 - 控制平面与 API 平面
- 数据平面 - Web UI 控制平面

控制平面

UPF 控制平面与 FRR 控制平面 控制平面与 UE IP 控制平面 UE 控制平面

控制平面 FRR

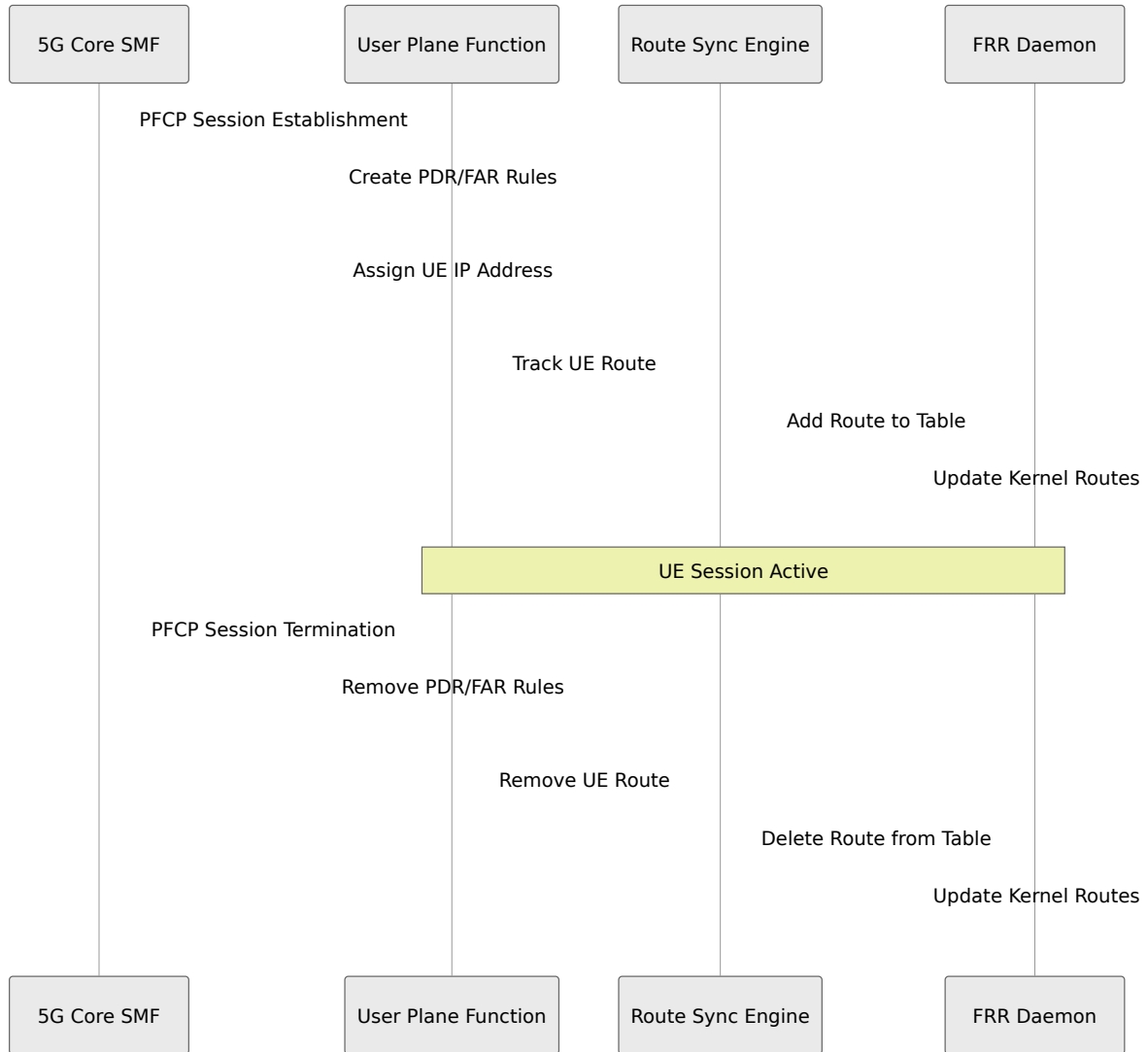
FRR 控制平面 控制平面与 Linux 控制平面 Unix 控制平面 BGP 控制平面 OSPF/RIP 控制平面 FRR 控制平面

控制平面



UE Session Establishment

Sequence Diagram



Flowchart

UPF 接收 UE IP 并生成 PDR/FAR 规则

1. 接收 **UE** 的 IP 地址并生成 PFCP 消息中的 UE IP
2. 生成 PDR/FAR 规则并发送给 RSE
3. RSE 通过 **FRR** 的 API 将路由信息发送给 FRR 进程
4. FRR 进程更新内核路由表/转发信息库

FRR

Ansible

FRR 使用 **Ansible** 进行配置 Ansible 使用 FRR 使用 **Jinja2** 模板
Ansible 使用 UPF 模板

Ansible 使用 FRR Jinja2 模板

```
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
```

Ansible

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

4. 使用Ansible 配置FRR的IP 地址 ID 配置 UPF 和 FRR 的

Jinja2 模板配置

- **OSPF** 配置 ID 和邻居关系
- **BGP** 配置ASN和邻居关系
- 配置静态路由 `redistribute static` 和 UE 路由
- 配置路由策略  
- 配置OSPF/BGP 路由

UPF 配置 FRR 的 UPF 路由和 PFCP 配置 FRR vtysh 配置 UE IP 地址 IPv4 和 /32 IPv6 和 /128 地址

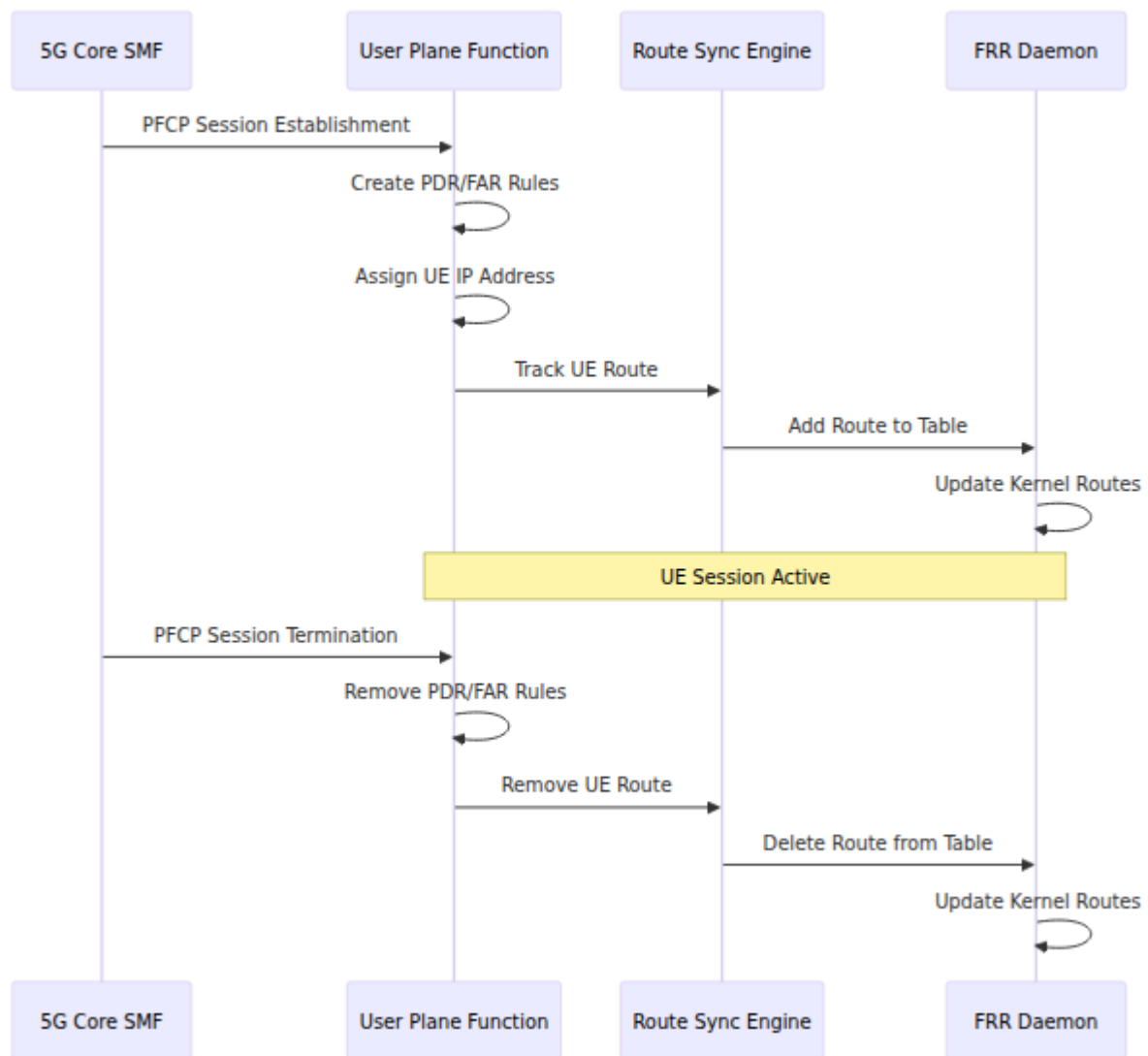
1. 配置 **FRR** 的 UPF 路由 vtysh
2. 配置 `redistribute static` 和 FRR 路由
3. 配置 **FRR** 的 OSPF/BGP 路由
4. 配置 UE 路由和 UPF 路由

配置 UPF 和 FRR 的 vtysh 配置 FRR 的 OSPF/BGP 路由
配置 `redistribute static` 和 `redistribute kernel`

配置

- 使用 Ansible 配置 FRR Jinja2 模板配置 UPF 路由
- **Ansible** 配置 Jinja2 模板配置 OSPF 和 BGP 路由
- **UPF** 配置 UPF 路由和 PFCP 配置 UE IP /32 路由
- 配置 UPF 和 FRR 路由和 UE 路由
- 配置 Ansible 配置    配置 UPF 路由

□□□□



□□□□□

Web UI □□

UPF □□□□□□□□ □□ □□□□□□

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

- **BGP** 透過 BGP 實現跨網域的路由
- **OSPF** 透過 OSPF 實現 UE 與網元之間 LSA 的傳遞

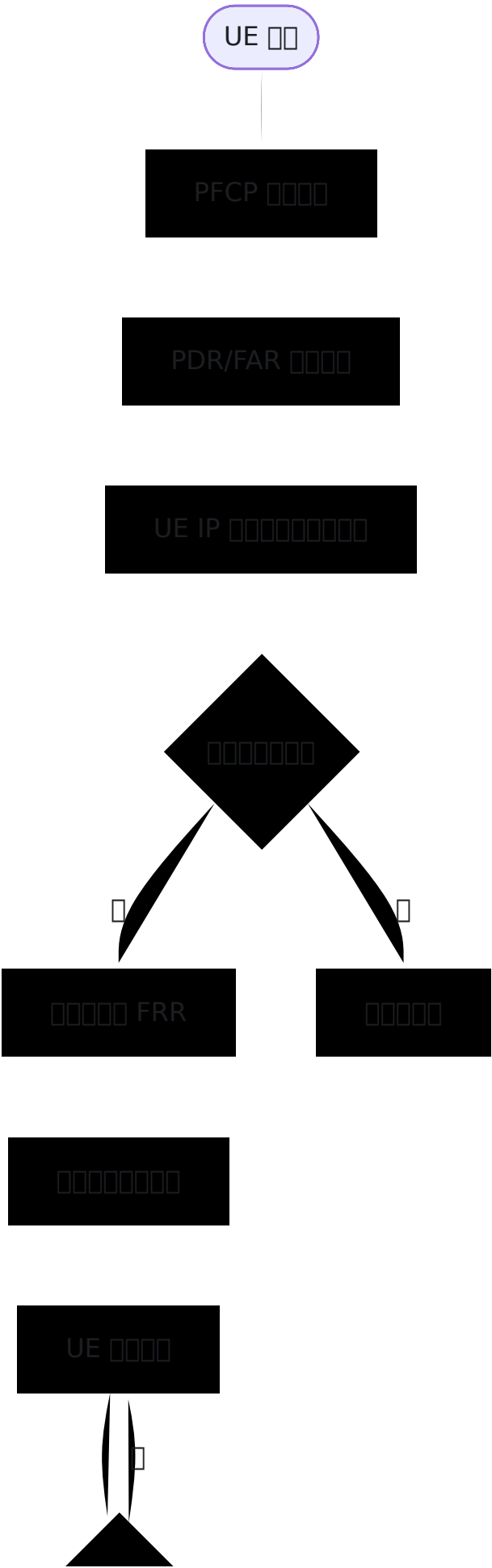
透過 OSPF 實現 UE 與網元之間 LSA 的傳遞

實現

透過 Web UI 實現 網元 與 網元 之間的配置

1. 透過 UPF 實現 UE 與網元
2. 透過 FRR 實現網元
3. 實現網元
4. 實現網元
5. 實現網元







□



□□

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

□□□□

API □□

UPF □□□□□□□□□□

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

□□□ API □□ - □□□□ □□□□□□□□□□□□□□□□

配置

IPv4 地址 100.64.18.5 配置

- 接口配置
- 路由配置
- 静态路由
- 动态路由

IPv6 配置

IPv4 与 IPv6 配置

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

IPv6 配置 FRR 配置 OSPFv3 与 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 LSA 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

- UPF UE IP
- FRR
- FRR OSPF
- OSPF