free5GCの構造と特徴

2021-4-13

hirono

自己紹介

広納幸治 / HIRONO Koji

APRESIA Systems株式会社

- 2019年11月から5GのコアやRANのソフトウェア開発に関わる
  - 無線の知識0からのスタート

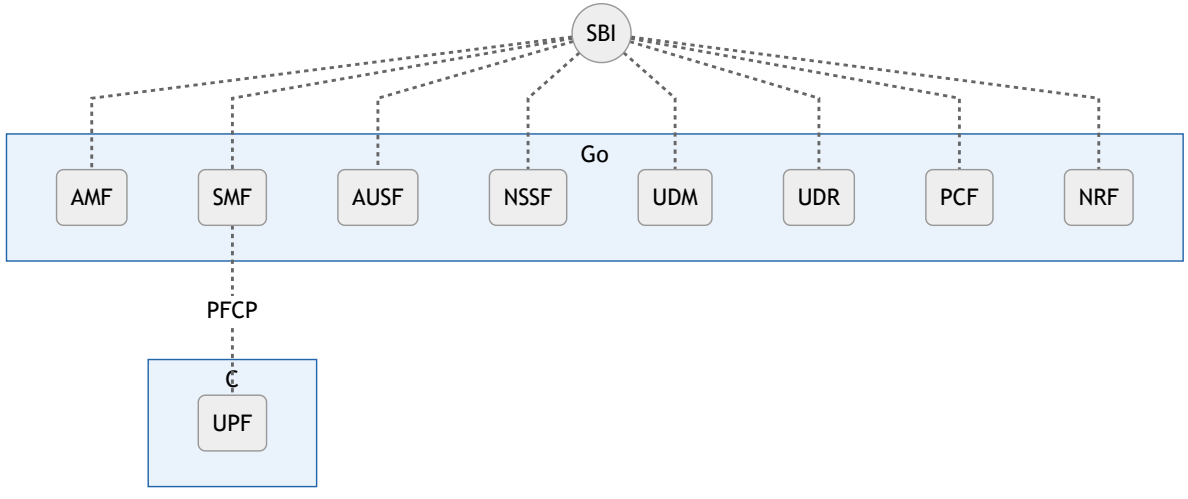
スタイル

- ソフトウェアで解決する
- 世の中にないものは自分で作る
- あらゆる策を考えた上で正面突破

Agenda

1. UPFを除くNFは全部Webサーバーアプリ
2. Goの特徴
3. データベースはMongoDB
4. 設定はYAML
5. UEの登録は専用のWeb UI
6. ディレクトリ構成は独自方式
7. AMFの構造
8. SMFの構造
9. UPFの構造
10. gtp5gのパケットの流れ(Uplink)
11. gtp5gのパケットの流れ(Downlink from DN)
12. gtp5gのパケットの流れ(Downlink from UPF)
13. 商用利用の障壁だった過去
14. まとめ

1. UPFを除くNFは全部Webサーバーアプリ



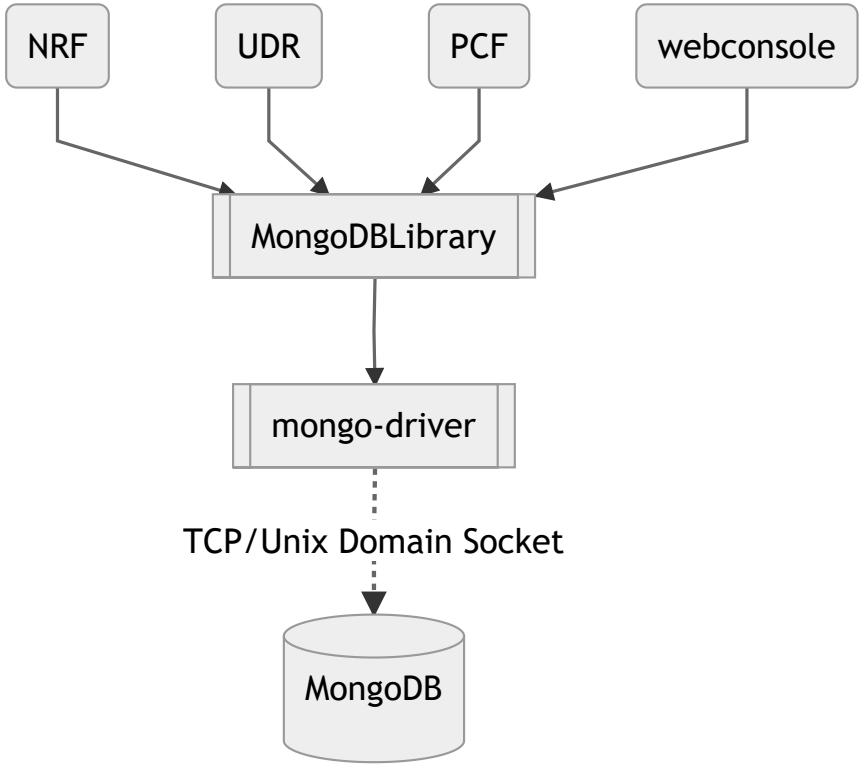
- UPF以外のNF
  - Goで実装
  - 無線の知識不要
  - Webサーバーアプリの知識があれば作れる
- UPF
  - Cで実装

2. Goの特徴

- 高いスケーラビリティ
  - goroutineによるM:Nスレッドモデル
- 高い生産性
  - 標準パッケージが充実
  - 数行でHTTPサーバーが動く

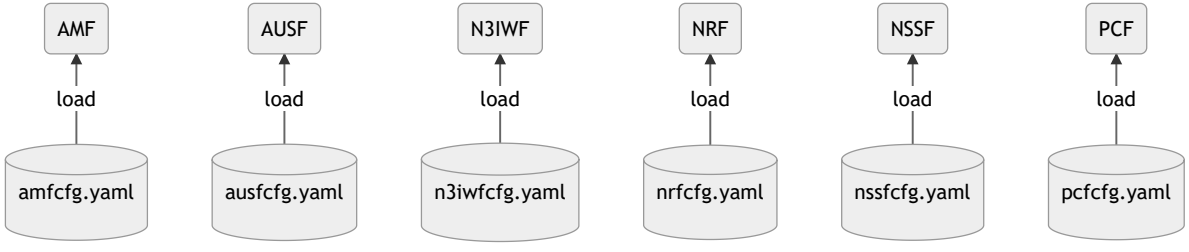
```go
package main

import (
    "net/http"
)

func handler(w http.ResponseWriter, r *http.Request) {
    w.WriteHeader(http.StatusOK)
}

func main() {
    http.HandleFunc("/", handler)
    http.ListenAndServe(":8080", nil)
}
```

3. データベースはMongoDB



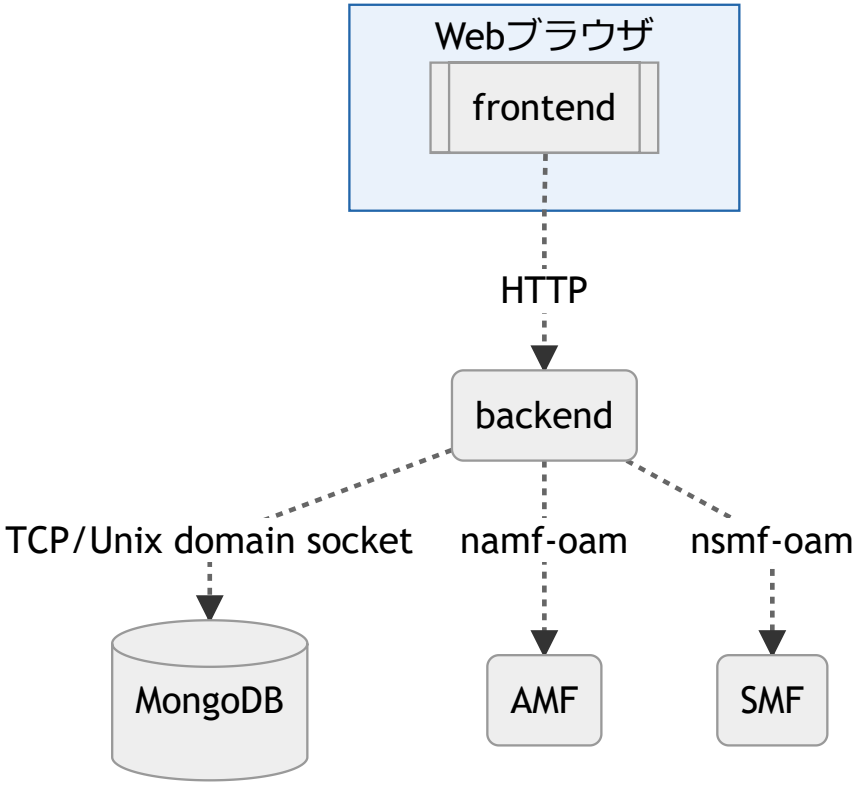
- MongoDBの特徴
  - スキーマの事前定義不要
  - 事前の初期化は不要
- MongoDBに接続するNF
  1. NRF
  2. UDR
  3. PCF
  4. webconsole

4. 設定はYAML



- NF毎に設定ファイルを保持
- 設定する内容が重複する部分もあるがNFを別ホストで分離することを考えると妥当
- 起動した後に変更は不可

5. UEの登録は専用のWeb UI



- frontend
  - Reactで実装
- backend
  - Goで実装
  - MongoDBに直接接続
  - UEの状態取得
    - namf-oam
    - nsmf-oam

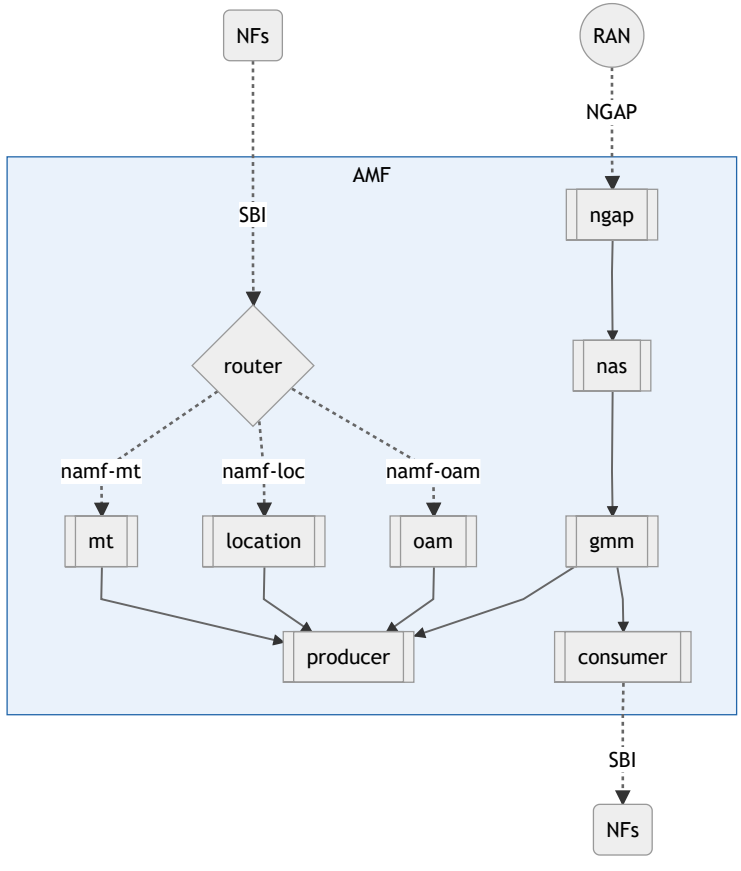
6. ディレクトリ構成は独自方式

Linuxの標準的な構成(FHS準拠)ではない

- Linuxの標準的な構成(FHS準拠)
  - 実行ファイルは/usr/bin配下
  - 設定ファイルは/usr/etc配下
- free5GCの構成
  - "free5gc"という名前のルートノード必須
  - ソースコードのあらゆる箇所で埋め込まれている

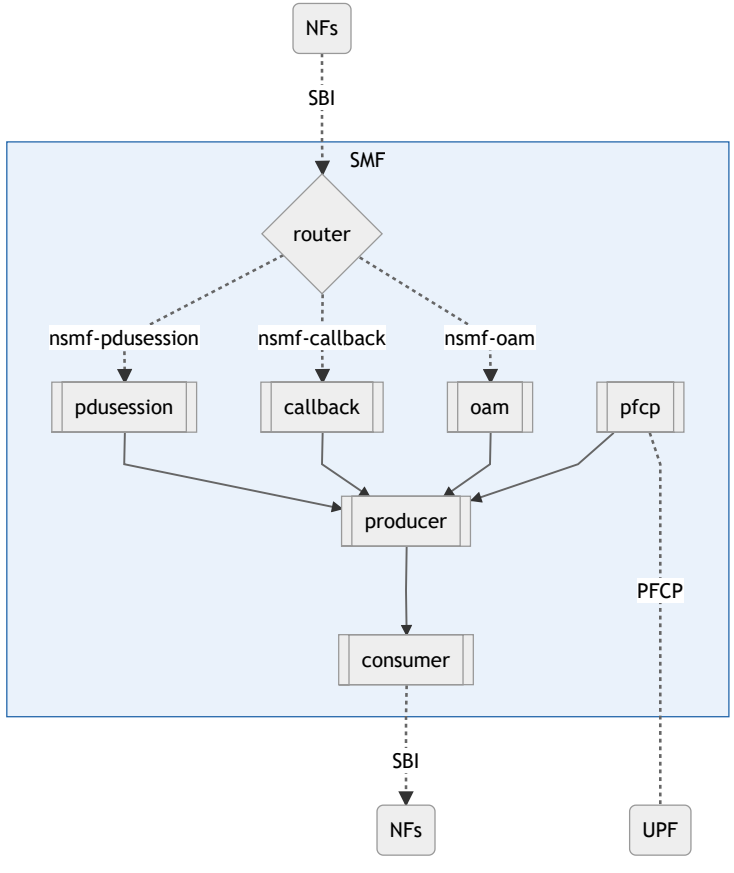
```
$ grep -r Free5gcPath .
./amf/service/init.go:          DefaultAmfConfigPath := path_util.Free5gcPath("free5gc/config/amfcfg
./amf/util/path.go:     AmfLogPath           = path_util.Free5gcPath("free5gc/amfsslkey.log")
./amf/util/path.go:     AmfPemPath           = path_util.Free5gcPath("free5gc/support/TLS/amf.pem")
./amf/util/path.go:     AmfKeyPath           = path_util.Free5gcPath("free5gc/support/TLS/amf.key")
./amf/util/path.go:     DefaultAmfConfigPath = path_util.Free5gcPath("free5gc/config/amfcfg.yaml")
./amf/util/path_debug.go:       AmfLogPath           = path_util.Free5gcPath("free5gc/amfsslkey.log"
./amf/util/path_debug.go:       AmfPemPath           = path_util.Free5gcPath("free5gc/support/TLS/_c
./amf/util/path_debug.go:       AmfKeyPath           = path_util.Free5gcPath("free5gc/support/TLS/_c
./amf/util/path_debug.go:       DefaultAmfConfigPath = path_util.Free5gcPath("free5gc/config/amfcfg.

...(snip)...
```

7. AMFの構造



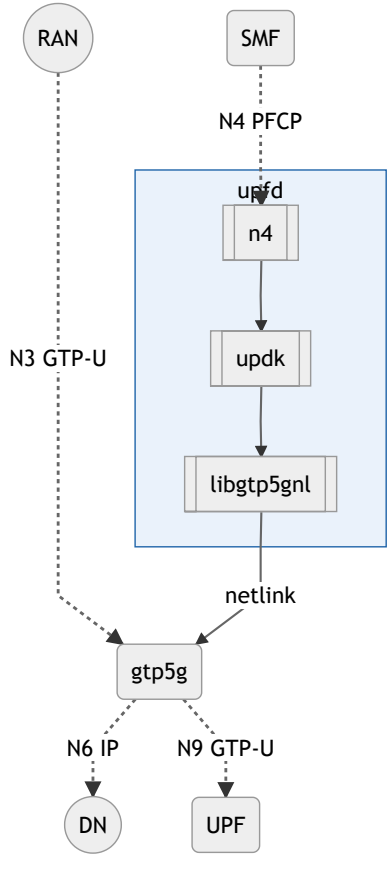
- router
  - サービス毎に振り分け
- producer
  - 内部データを保持
  - 外部からのリクエストに対して最終的に処理
- consumer
  - 外部へのリクエストを処理
- gmm
  - 5GMMを実現
  - 状態遷移マシン

8. SMFの構造



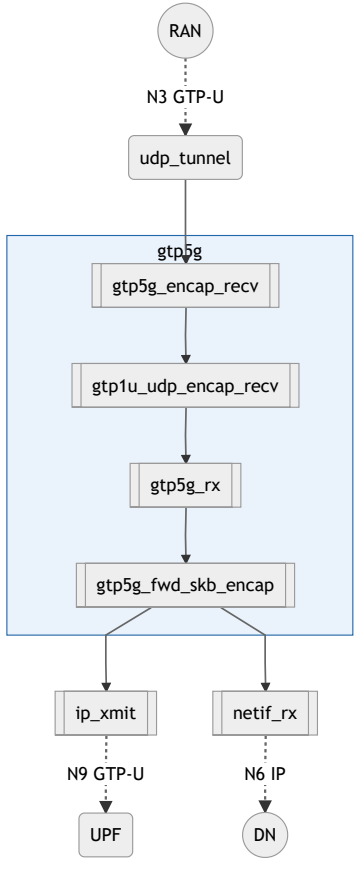
- router
  - サービス毎に振り分け
- producer
  - 内部データを保持
  - 外部からのリクエストに対して最終的に処理
- consumer
  - 外部へのリクエストを処理
- pfcp
  - UPFとPFCPをやりとり

9. UPFの構造



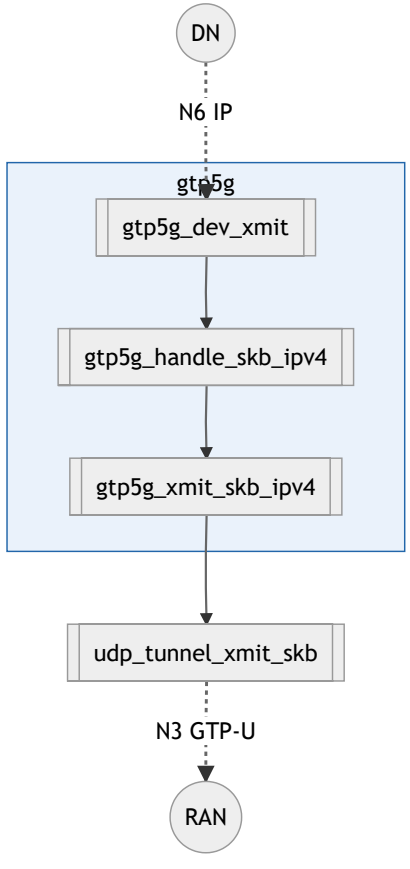
- upfd
  - n4
    - SMFとPFCPをやりとり
  - updk
    - PFCPを内部データに変換
  - libgtp5gnl
    - netlinkを経由してgtp5gにPFCPを設定
    - libgtpnlからfork
- gtp5g
  - カーネルモジュール
  - U-Planeのパケットを転送
  - stage2の途中から登場

10. gtp5gのパケットの流れ(Uplink)



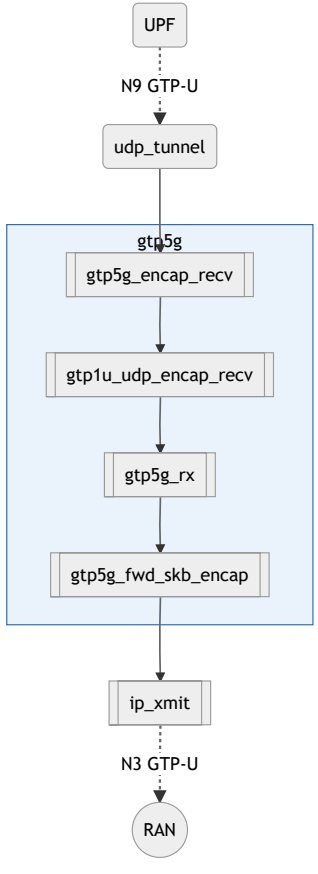
- gtp5g_encap_recv
  - udp_tunnelからコールバック
- gtp1u_udp_encap_recv
  - 受信パケットのチェック
  - PDRの検索
- gtp5g_rx
  - FARに従って処理
- gtp5g_fwd_skb_encap
  1. DN
     - GTPデカプセル
     - netif_rxで転送
  2. 後段UPF
     - ヘッダー書き換え
     - ip_xmitで転送

11. gtp5gのパケットの流れ(Downlink from DN)



- gtp5g_dev_xmit
  - エントリポイント
- gtp5g_handle_skb_ipv4
  - PDRの検索
  - FARに従って処理
  - pktinfoの組み立て
- gtp5g_xmit_skb_ipv4
  - pktinfoに従って転送
  - udp_tunnel_xmit_skbでGTPカプセル

12. gtp5gのパケットの流れ(Downlink from UPF)



Uplinkと同じ関数を通る

- gtp5g_encap_recv
  - udp_tunnelからコールバック
- gtp1u_udp_encap_recv
  - 受信パケットのチェック
  - PDRの検索
- gtp5g_rx
  - FARに従って処理
- gtp5g_fwd_skb_encap
  - ヘッダー書き換え
  - ip_xmitで転送

13. 商用利用の障壁だった過去

	過去	現在
ソースコード	lib部分はバイナリ提供のみ • 逆アセンブルで解析は可能 • 修正は不可能	全て公開

ライセンス	1. アカデミックライセンス
	  • 商用利用不可
	2. 商用ライセンス
	  • 別途問い合わせ必要
	  • 詳細不明
		Apache 2.0
		• 商用利用しやすい

14. まとめ

NFの構造と特徴

- UPFを除くNFはWebサーバーアプリ
  - 無線の知識不要
  - Goによる高い生産性とスケーラビリティ
- UPF
  - 専用のカーネルモジュールgtp5gを用意

巨人の肩の上にのる矮人

- 完成度100%とは言えないがfree5GCをベースとして拡張していけそう