BBSakura Networks

自作k6 Extension利用したNFVへの負荷計測手法の紹介

BBSakura Networks はやさかたける

JANOG 53

自己紹介

- はやさかたける(早坂彪流)
  - インターネットではtakemio,takehayaとか
- 京都市在住, 宮城県出身,社会人3年目
- 初めてのJANOG現地参加🐣
- 普段はモバイルコアの開発やってる
- 趣味: 最近自宅インフラ入門した
- 悩み: お引越し先探し
- 所感: JANOG 47（福岡）でもLTをしました。
  その時は学生で若者支援で行く予定だったんですが
  コロナでリモートだったので今日はめっちゃエモいです😂

BBSakuraNetworks

- BBIXとさくらインターネットの合弁会社
- 2019年8月1日設立
- ネットワークサービスのソフトウェア開発。親会社を通じてソフトウェアを世に出している
- すべての「モノ」がつながる社会の実現に向け、プラットフォーム(OCX)
  提供を通じてネットワークサービスのクラウド化を進めてます

開発や運用をしているサービス

- **さくらのセキュア<u>モバイル</u>コネクト（フルMVNO)**
- さくらのショートメッセージサービス
- BBIXお客様向けポータル
- OCX (Open Connectivity eXchange)

モバイルの開発で得た知見の話をします！！！

背景と問題

弊社が運用するモバイルコアネットワークというのはNF(V)の集合体です。
色々な接続点を通じてNF同士を繋げ、そのNFの中を呼が通り、協調動作することで初めて携帯電話等(UE)は外部ネットワークに接続できます。

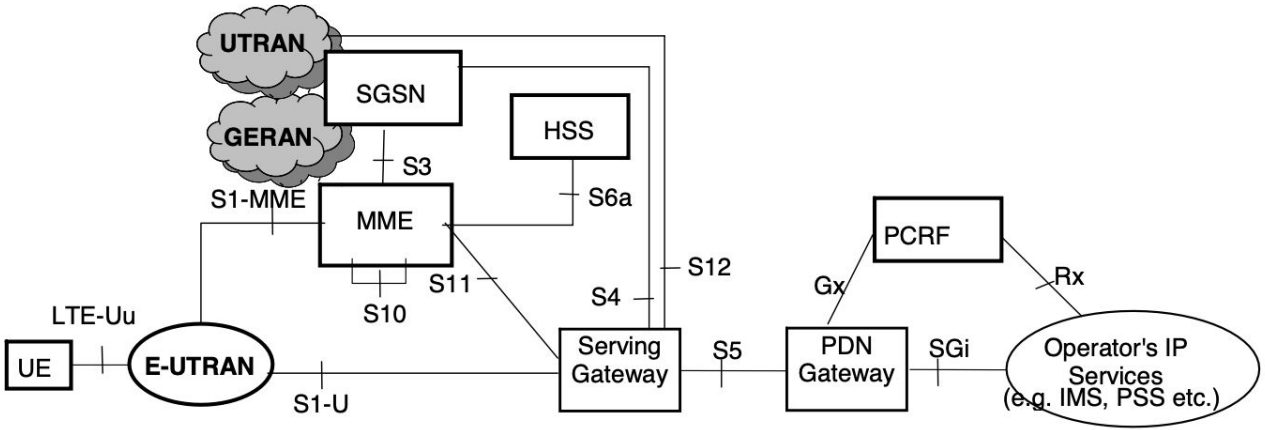


Figure 4.2.1-1: Non-roaming architecture for 3GPP accesses

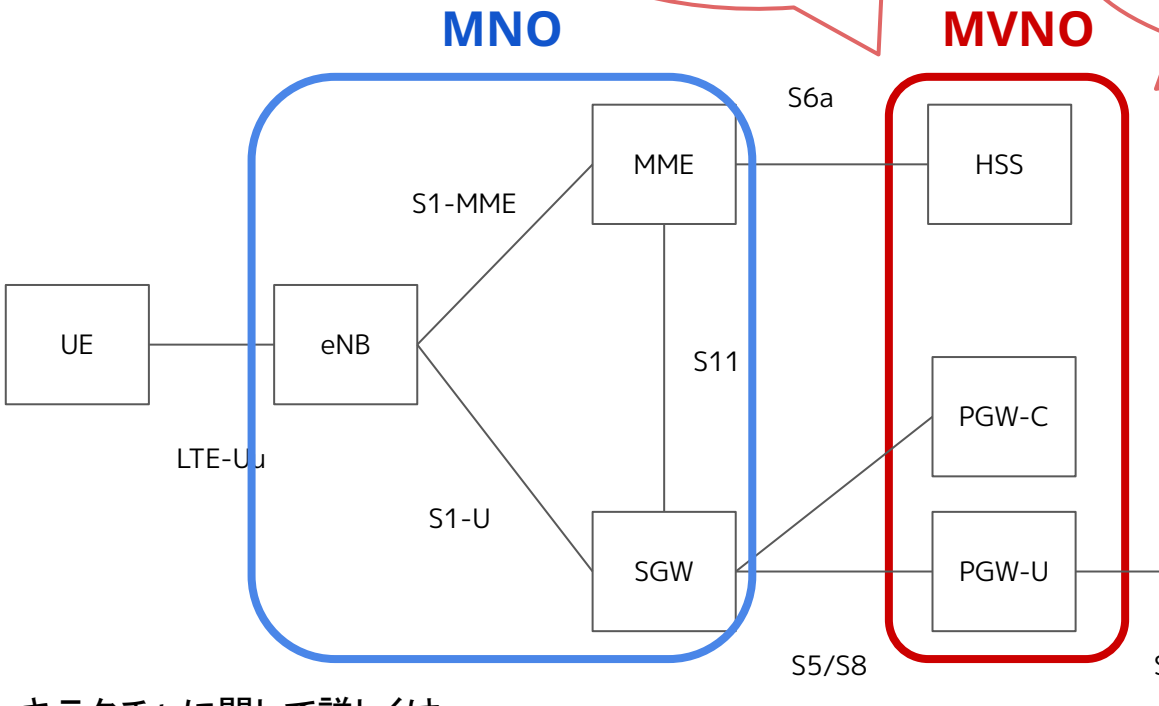
3GPP 23.401 4.2 Architecture reference model

背景と問題

BBSakura Networks

弊社の取り扱う NFたち （全部自作）

MNOの持ってるNFと （実質）直接繋がる...!!! **ちゃんとしないと**やべぇ😇



UE: User Equipment
eNB: eNodeB, evolved Node B
MME: Mobility Management Entity
HSS: Home Subscriber Server
SGW: Serving-Gateway
PGW-C: Packet Data Network-Gateway Controlplane
PGW-U: Packet Data Network-Gateway Userplane

アーキテクチャに関して詳しくは
3GPP TS23.401とGSMA IR88を参照

詰まるところの問題を整理すると...?

MVNOはSIMを登録・認証認可や外部接続性を司るサービスを持っており、クラウド上にNFVとして展開し、それをMNOが持つNFと接続している。

様々なステークホルダーがいる中でウッカリ障害を起こすとかなり困る😇

（総務省報告とかも....）ちゃんとしないといけない...😱

Q. じゃあそもそも「ちゃんとする」というのは？

A. 「システムの健全性」を高く保ち運用すること

システムの健全性を事前に測るには？

cf. Why Your Organization Should Perform Load Testing

> 負荷テストは、システムへの実際のトラフィックをシミュレートしてそのパフォーマンスを評価し、潜在的な問題を発生前に検出して防止するのに役立つ

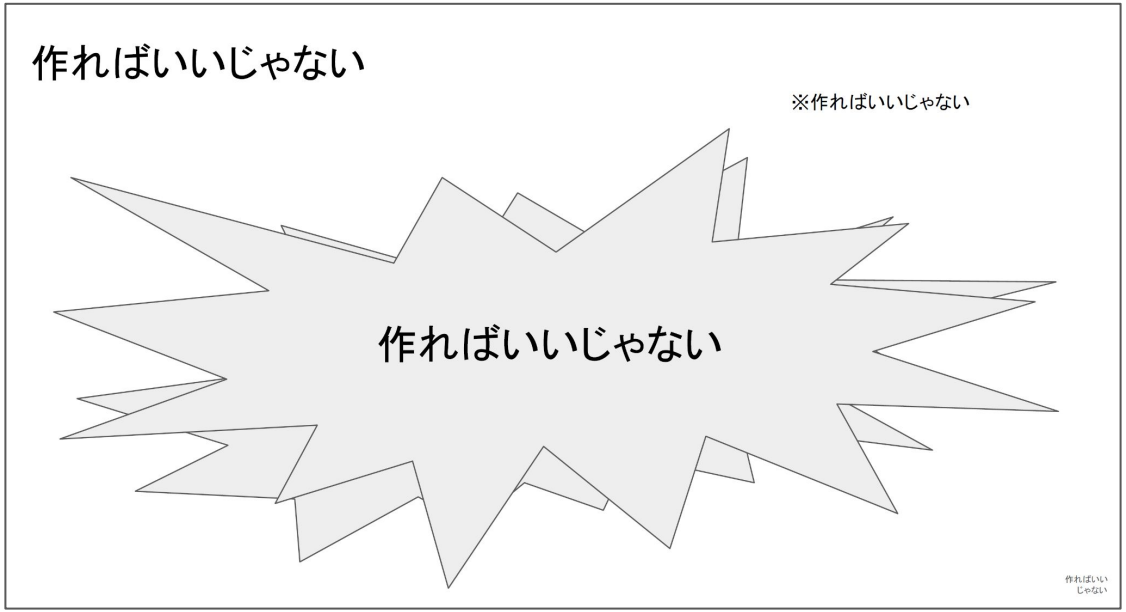
=>（一般論として）**パフォーマンス計測が有効である**

が、OSSにそんなものはなく、、、
商用のプログラマブルで尚且つパフォーマンス計測のツールは高級...😇💸💸💸
特にCplaneは対応するべきプロトコルのスタックもそれなりに複雑😇😭

では僕らの取るべき次の手は、一体なにをするべきなのか...?

自作したらええやん

- パフォーマンステスター、なければ作ればいいじゃない
  - 完全に声に出したい日本語です



野良OSS活動のすすめ by Yoshiyuki Kurauchi | Open Mobile Network Infra Meetup #3

U-Planeに関してはこちらを参照

- PyCon APAC 2023でTRexと言うDPDKで出来たOSSをしばいて
  PGW-Uに対して実施した話を紹介してます。興味があれば是非どうぞ:)

自作パケット処理系の性能測定と可視化&改善のPDCAを回して最強のパケット処理系の作り方を学ぼう

解決策

- k6: Grafana Labsが作ってるGo製の
  パフォーマンス測定ツール
  - JSでテストシナリオを記述することができる
    - cf. https://github.com/dop251/goja
  - Goでプラグインを書いてJSでテストシナリオが書ける
    つまりGoの既存資産が使える！！！！！
  - k8sに乗せてスケールさせることもできる
  - メトリックも色々なダッシュボードがあり便利
- ✨自作k6プラグイン✨: xk6-gtp
  - gtpv2のechoリクエストのexampleを右に示します
  - Githubで絶賛公開中！実験プロジェクトです！
    Starよろしくお願いします！！！
  - https://github.com/bbsakura/xk6-gtp

```
import { check } from 'k6';
import exec from 'k6/execution';

import gtpv2 from 'k6/x/gtpv2';

let client;

export default function (){
    if (client == null) {
        client = new gtpv2.K6GTPv2Client();
        client.connect({
            saddr: `127.0.0.${exec.vu.idInTest}:2124`,
            daddr: "127.0.0.1:2123",
            count: 0,
            IFTypeName: "IFTypeS5S8PGWGTPC"
        });
    }
    const res =
client.checkSendEchoRequestWithReturnResponse(
"127.0.0.1:2123")
    check (res, {
        'success': (res) => true === res,
    });
}
```

使い方: CSR (Create Session Request) の例

BBSakura Networks

- SGWからPGW-Cに対してCSRを投げている例
  - CSR (Create Session Request): モバイル端末が外部接続するためのセッションを生成するリクエスト
- 生成済みのインスタンスに含まれるメンバ関数に対して宛先とパラメーターを渡すだけで動く
- 正しくResponseを受け取ればcheckのカウントが上がるので、それの回数を見ることでパフォーマンスのメトリックとすることが可能

```
import { check } from 'k6';
import exec from 'k6/execution';
import gtpv2 from 'k6/x/gtpv2';
let client;
export default function (){
    // 略
    const csr_res =
client.checkSendCreateSessionRequestS5S8(
        "127.0.0.1:2123",
        {
            imsi: "123451234567891",
            msisdn: "123451234567891",
            mei: "123451234567891",
            mcc: "123",
            mnc: "123",
            tac: 1,
            rat: "EUTRAN",
            apn: "apn",
            eci: 1,
            epsbearerid: 1,
            uplane_ie: {
                teid: 1,
            },
            ambrul: 100000000,
            ambrdl: 100000000,
        }
    )
    check (csr_res, {
        'csr is success': (res) => true === res,
    });
}
```

実装の所感

- Goを書いてJavascriptで動かすというのは不思議な感覚だった
- ドキュメントが足りなすぎる😇ので結局 k6本体のコードを読むことに...
- 実際にGTPパケットを投げる実装は go-gtp に丸投げしたのでそのハンドラをちまちまと書いてる感じになった。~~あまりパケットの構造を書いてる感じがなくて楽しいかと言うと諸説ある~~
- go-gtpのラッパーなので、テストがE2Eぽい感じになっていく
- 一方ダッシュボード付きでお手軽に自分の投げたい負荷生成機を作れるのはすごい。すごすぎる。

実装でハマったこと:2度Listenしてしまう問題

BBSakura Networks

- go-gtpの仕様上インスタンスを作るときにport listenを行うのが普通で、k6上でナイーブな実装をすると2度Listenしてしまい、ポートがすでにBindされてるのでエラーを吐く😇
- k6のライフサイクルは3つのステップに分かれてて、事前処理, 負荷走行, 事後処理の順番で動作し、これらは予約語に基づいた関数にMappingされている。
- k6の実装はJSのコードをGojaRuntimeに3度読ませていて、そのままコードを解釈させている。
  この時接続のConnをGlobalスコープで初期化してしまうと複数初期化してしまう問題がある😇
  - この辺はドキュメント化されてる話ではなかったのでコードを読むことになったハマりポイントだった....😭
- 最終的にはnullチェックと負荷走行の関数で初期化するワークアラウンドで解決 💡

```
let client = new gtpv2.K6GTPv2Client();
client.connect({ /*略*/});

export function setup()
{ // setup code }

export default function (data)
 { // VU code }

export function teardown(data)
{ // teardown code }
```

実装のコツ: Send and Recv の仕組みをうまく作る

BBSakura Networks

- Reqを投げたらRspが返ってくると言うのがHTTPだと普通ですよね。
  これはモバイルの通信も同じです。残念ながらHTTPでは無いので、
  我々もそこの仕組みもちゃんと作ってあげる必要があります😇
- 具体的にはパケットを受け取るビジーループを行い、パケットを受け取った際にエラーコードを確認するハンドラを追加する必要がありました
  - 受け取りたいパケットのハンドラは多岐に渡るのでGoのジェネリクスを利用したら便利でした

今回自作した
xk6-gtpのやる
範囲は赤枠のここ
CSRとMBRが取れ
ていることが検証
には必要でした

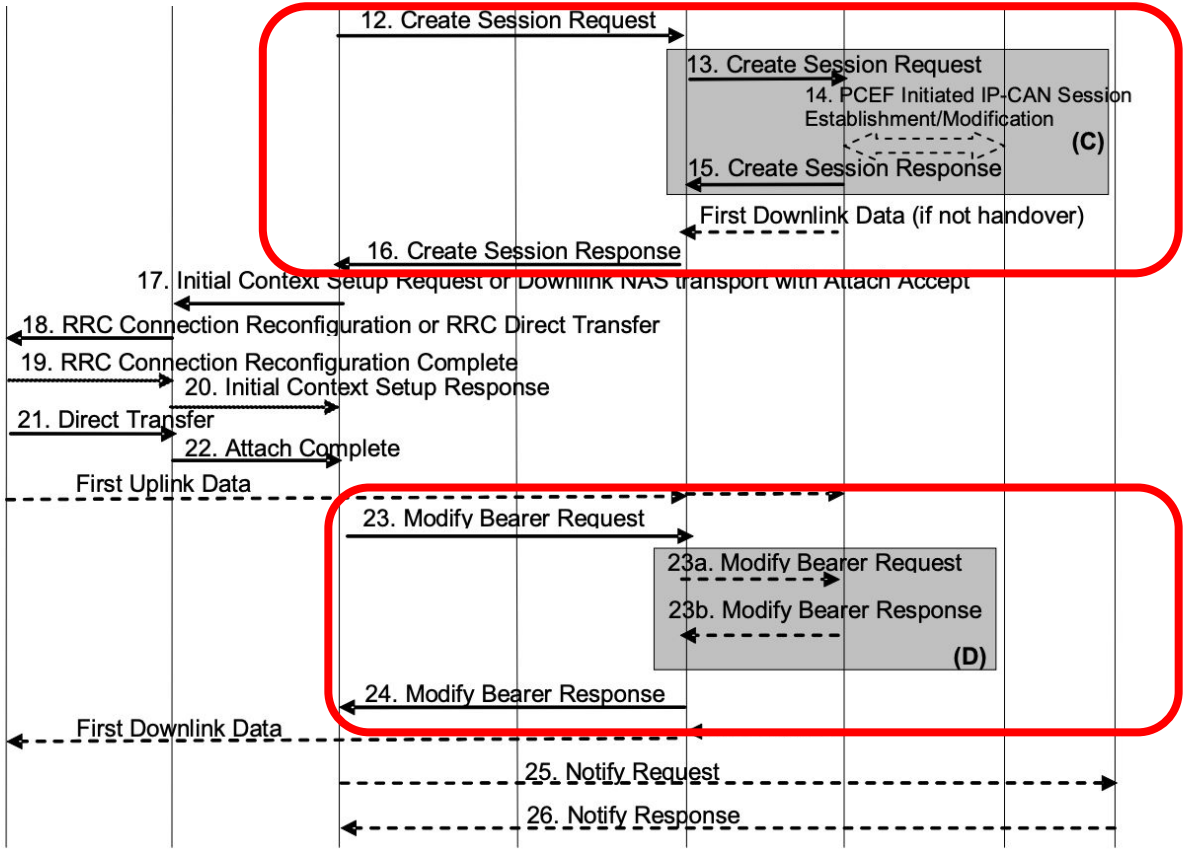


Figure 5.3.2.1-1: Attach procedure

終わりに

BBSakura Networks

- このプラグインは同僚と一緒に作ってますのでまずは感謝を。

- 今後の課題
  - とりあえず動くところまでは行けましたが、振り返ると実質SDKの設計みたいなものだったので、
    パラメーターが多いものをどのように抽象化するか・より高速化するにはどうしたらいいかの
    検討や作り直しの余地がありそうだなとの感想を抱いています。より使いやすくしたいです。
- まとめ
  - NFVのパフォーマンステストにはk6と呼ばれるパフォーマンステストツール
    を拡張してみると色々便利に使えることがわかった。
  - ステートを持つようなk6プラグインを作る場合はパケットを投げたら受け取って判断して取り回すなどの
    工夫が必要であることもわかった。
  - k6を使った楽しいNFV開発運用ライフを送ろう！
- 自作k6プラグイン xk6-gtp もよろしくね。GTPv2のみ対応だよ。
  今回説明しなかったところとか詳しく知りたい人は是非コード見てみてね！
  自分のPGW-Cをムキムキにしたい人はどうぞ！！

Enabling a Connected Future.

BBSakuraNetworks