

susanow 計画:  
環境に対して自動最適化する  
高性能通信基盤の開発

Hiroki SHIROKURA @slankdev HoseUniv/IJii

powered by IPA-MITOU-program

# Self-introduction

城倉 弘樹 (SHIROKURA Hiroki) aka slankdev

- ▶ 法政大学 理工学部 B4
- ▶ 高性能通信, パケット解析, セキュリティ

未踏事業

- ▶ 環境に対して自動で最適化する高性能通信基盤
- ▶ プロジェクト詳細: <http://draft.susanow.dpdn.ninja>
- ▶ ルータや Firewall を環境に合わせて自動で最適化する基盤, ルータそのもの, 管理するシステム全体
- ▶ 高性能で動的な NFV の実現

# NFV #とは...

- ▶ Network Functions Virtualization (NW 機能の仮想化)
- ▶ ルータなど Network Functions(NF) をソフトウェアで実現 Virtual NF(VNF)
- ▶ コストダウン: CAPEX(設備費)/OPEX(運用費) 低下
- ▶ 迅速なサービス展開が可能
  - ▶ サービス機能の迅速な変形
  - ▶ サービス容量の迅速な拡大/縮小

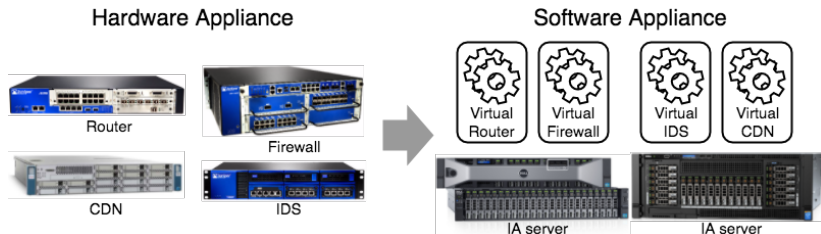


Figure 1: NFV Approach

## NFV: 迅速な変形, 拡大/縮小の例

- ▶ ルーティングと FW を提供するネットワーク
- ▶ DoS 対策を行っていないネットワークだと DoS 攻撃によるサービス停止の危険
- ▶ DoS を検知したタイミングで新たに VNF をデプロイ
- ▶ 変形: 必要に応じてその場その場で NF をつなぎ合わせる
- ▶ 拡大: 最低限のリソースで最大限のパフォーマンス
  - ▶ FW 等はルールによっても必要な計算資源の量が違う。

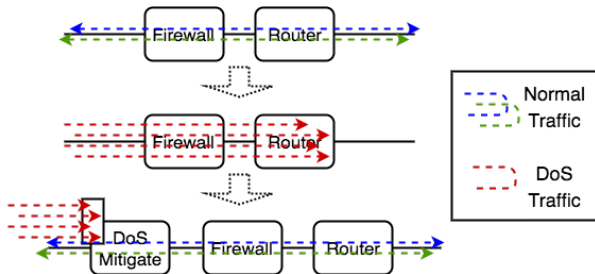


Figure 2: 状況に応じてネットワークを変形

# NFV のアーキテクチャ

## NFV の抽象アーキテクチャ

- VNFs: Router, FW, DPI, etc...
- NFVi: 各種リソースを管理する基盤
- MANO: 全体を管理/連携

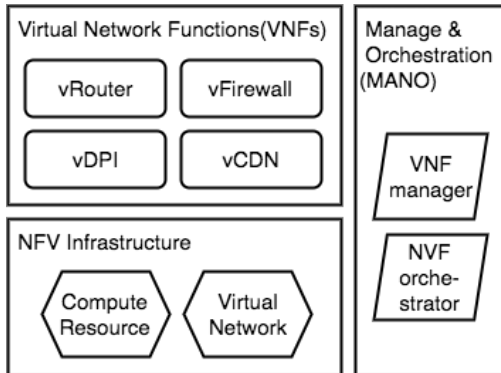


Figure 3: NFV Architecture

## NFV: 実装について

- ▶ VNF は DPDK を用いて開発することで高性能に実現可能
- ▶ DPDK: Data Plane Development Kit
  - ▶ IA サーバ上で高性能通信をするためのフレームワーク
  - ▶ 100G クラスのトラフィックもパケットフォワード可能
  - ▶ 4 つの特徴により実現

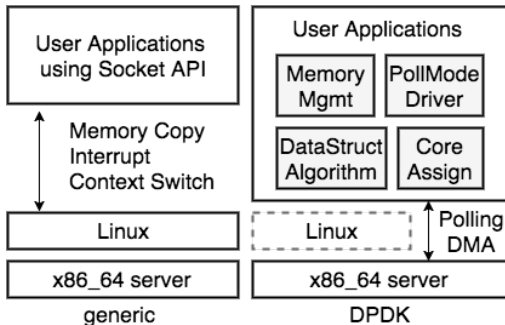


Figure 4: DPDK architecture

# NFV::DPDK: 高い開発コストと局所性, VM オーバヘッド

- ▶ コンピュータ理論に対する精通
- ▶ 特定の状況に合わせて最適化された特殊 VNF が多い
- ▶ VM 環境でのオーバヘッド: 仮装 NIC のメモリコピー

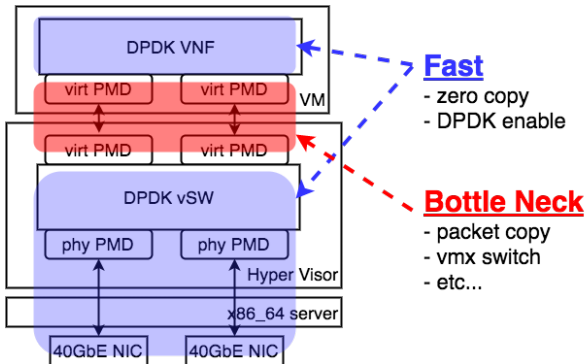


Figure 5: VM Overhead

# susanow 計画

## 主要技術

- ▶ D2: 動的スレッド最適化技術, MANO <- 後述
- ▶ SSN-NFVi: novm-NVFi <- 後述

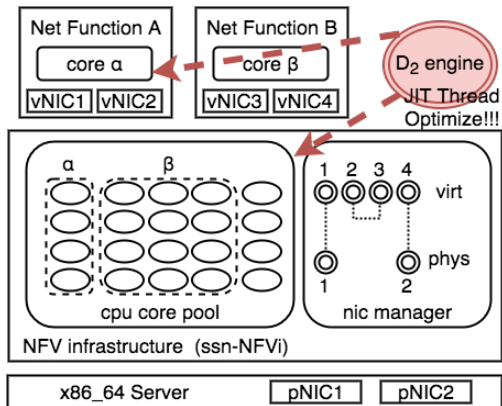


Figure 6: susanow nfvi



## D2: Dynamic Thread Optimization

- ▶ パケット処理ロジックの多重化を動的に行う技術
- ▶ D2-API を用いて, パケット処理のロジックを VNF 開発者が記述
- ▶ スレッドの多重化 (no-lock/no-block) の作業を自動で可能
- ▶ VNF のパケット処理ロジックとマルチスレッドの多重化を分離
- ▶ 適用範囲
  - ▶ vSwitch やルータなどの L2/L3-NF からアプリケーションデータを扱う DPI まで幅広く対応可能

```
void thread() {  
    while (true) {  
        npkts = rx_burst(0, mbufs, 32);  
        for (i=0; i<npkts; i++) {  
            process_packet(mbufs[i]);  
            tx_burst(1, &mbufs[i], 1);  
        }  
    }  
}
```

(n): thread, [n]: port (n:id)

開発者の書いた  
ロジックからVNFを生成

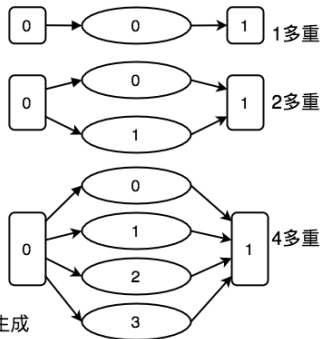


Figure 7:

## D2: Flow of Optimizing

### 1. 発火フェーズ

- ▶ VNF を追加したり減らしたりするタイミング
- ▶ トラフィックが増えたり, 減ったりするタイミング
- ▶ タイマーで一定期間ごとに性能チェック.

### 2. 発見フェーズ (環境情報より発見)

- ▶ NIC のスループット
- ▶ パケット格納用の Queue の統計情報

### 3. 修正フェーズ

- ▶ スレッドの多重度 (基本的にはこれ)
- ▶ NIC の HW 設定をチューニング

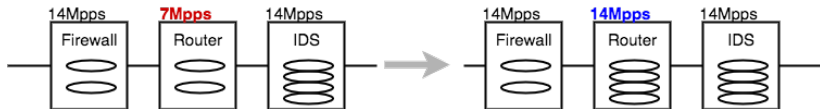


Figure 8:

## D2: Flow of Optimizing (修正フェーズ)

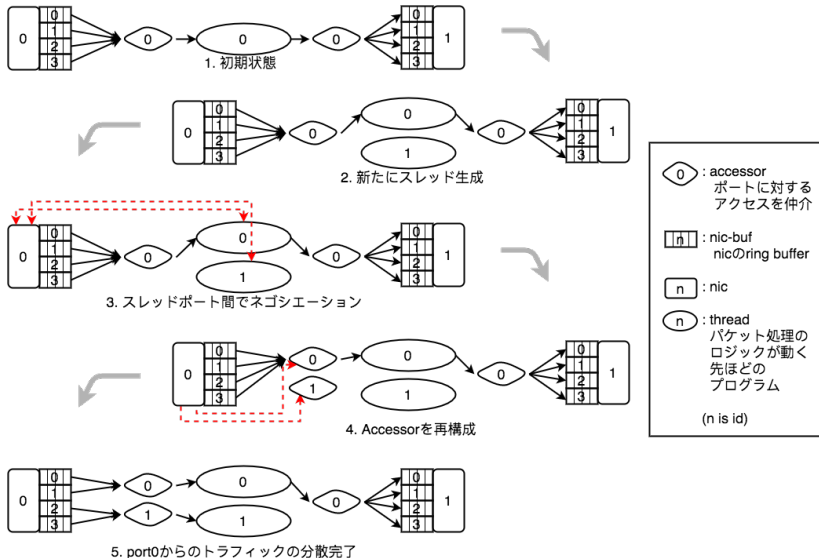


Figure 9:

## D2: まとめ

- ▶ VNF のパケット処理ロジックとその多重化を分離
- ▶ 多重に並列化することにより VNF 性能を向上
- ▶ 任意の並列多重数に最適化可能

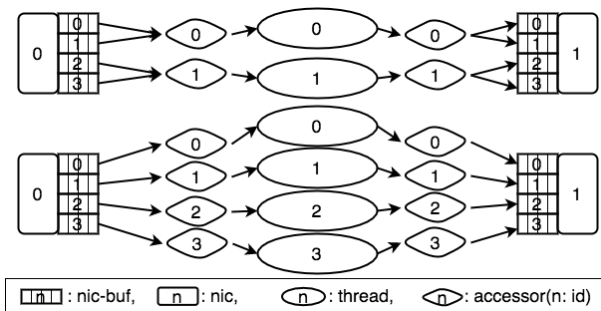


Figure 10: 並列数=2,4 に最適化した例

## SSN-NFVi

- ▶ D2 を用いた VNF をデプロイ可能な NFVi
- ▶ VNF のカプセルを VM では行わず, NFVi と同じプロセスで動かす
- ▶ 対応機能
  - ▶ CPU コアの管理
  - ▶ NIC(virt/phys) 管理, vNIC-patch-panel

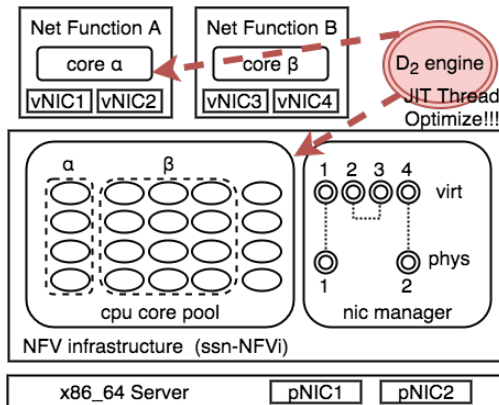


Figure 11: susanow nfvi

# Performance Evaluation

これから調べる予定. 8 回目合宿までに!!

- ▶ D2 オーバヘッド: 何 ns の処理オーバーヘッドか?
- ▶ D2 最適化中のトラフィックはどれだけとまるか

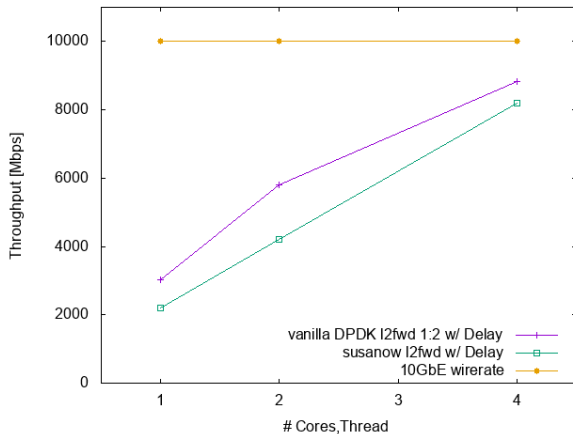


Figure 12: VanillaDPDK と Susanow の比較

# 全体のまとめ

- ▶ D2: 動的スレッド最適化技術の開発
- ▶ ssN-NFVi: nonVM な NFV 基盤の開発
- ▶ [WIP] D2 の最適化処理を制御するエージェント (gRPC で開発中)
- ▶ [WIP] ssN-NFVi 上で動作する VNF 複数種類 (VNF リポジトリ)
  - ▶ DPI, Router, FW, etc..
- ▶ より動的で高性能な NFV の実現

## 以降補足スライド

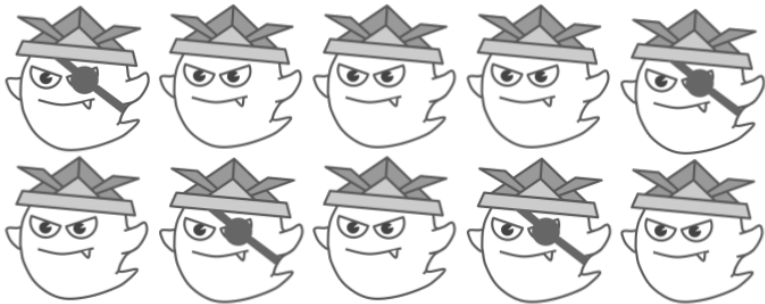


Figure 13: 以降は補足スライド



# 今後やる+プラスアルファ内容

- ▶ インターフェースの改善:
  - ▶ VNF のデプロイインターフェースの改善
- ▶ 性能向上: オーバヘッド部分を改善
- ▶ 複数 NFVi クラスタ対応: VNF のマイグレーション
- ▶ 互換性向上
  - ▶ VM を用いた VNF のデプロイの対応 (すぐできる)
  - ▶ 物理ネットワークアプライアンスの対応 (すぐできる)
- ▶ VNF の実装
  - ▶ 現在開発中: L2fwd, L3fwd, 5tupleACL, DPI
  - ▶ 他にも VNF 案があれば御意見ください