実CNFにおける メッセージバッファ構成の縮小

～キャッシュ効率化に向けた基礎設計～

○山田 歩人† 川島 龍太†
中山 裕貴†† 林 經正†† 松尾 啓志†

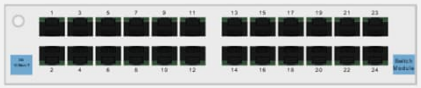
† 名古屋工業大学大学院
†† 株式会社ボスコ・テクノロジーズ

大規模ネットワークの理想形

専用ハードウェア 

Programmable Network Function

✓ 柔軟なD-Plane処理の実装

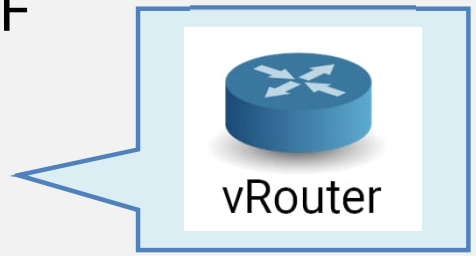


White Box Switch

Softwarized NF

✓ 自由度の高い開発
✓ AI/ML技術との親和性

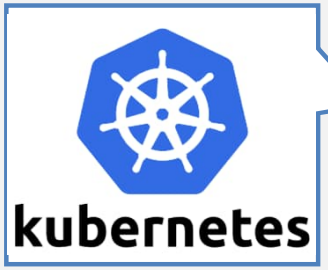
汎用サーバ

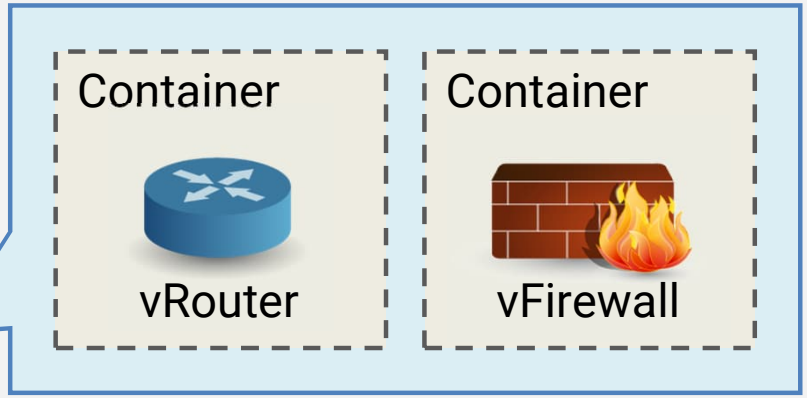


CNF: Cloud-native NF

☑ 多様な要件への対応

✓ 迅速なデプロイ
✓ クラウドのエコシステム



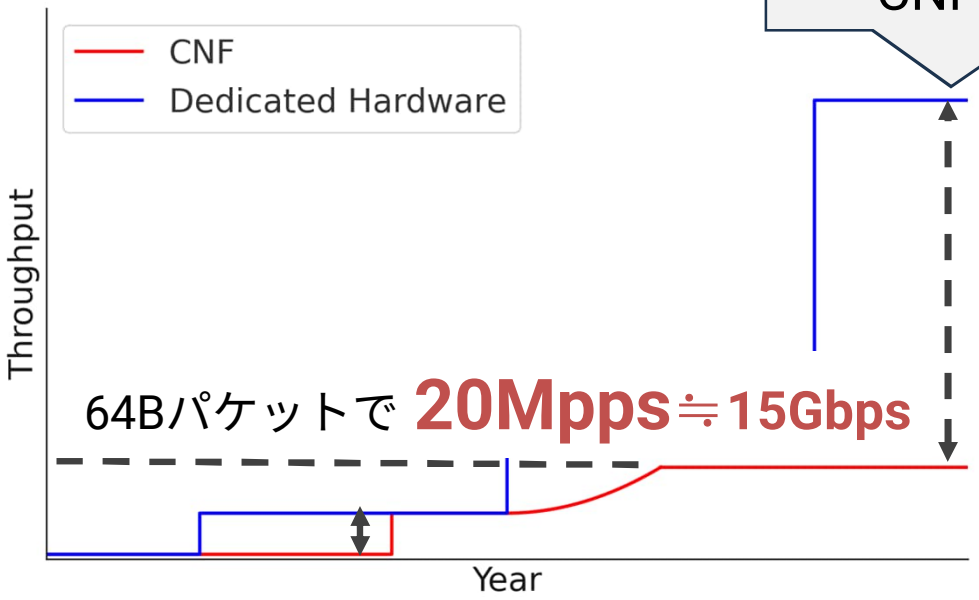


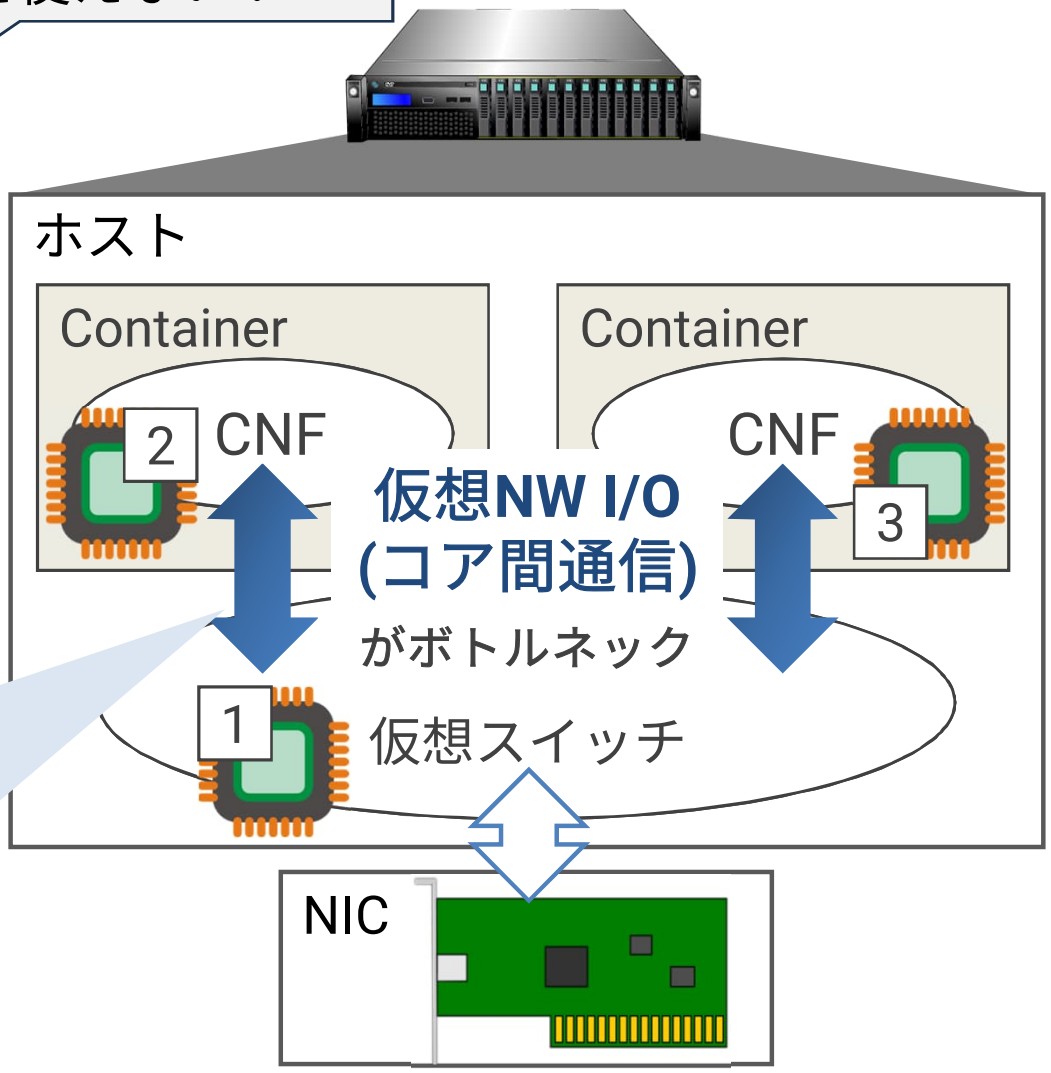
CNFが主流でないのは何故か？

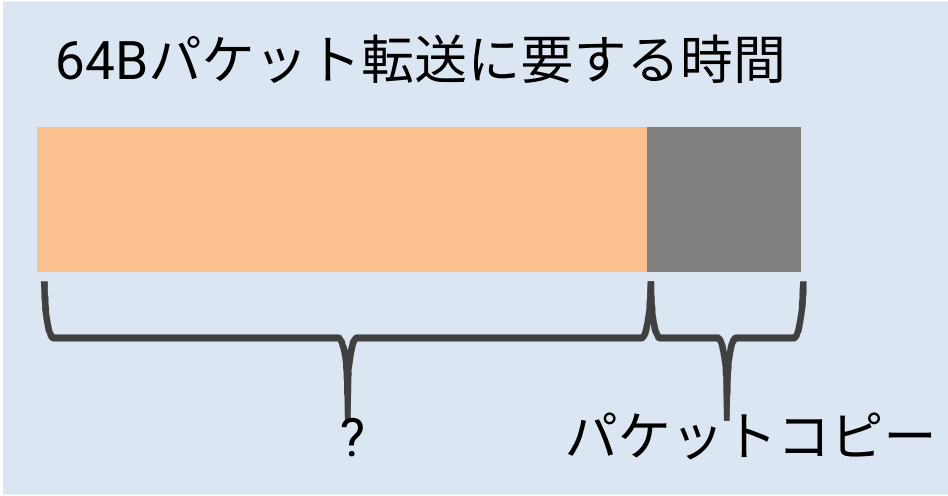
CNFの現状

性能が低い

この差を埋めないと
CNFを使えない！



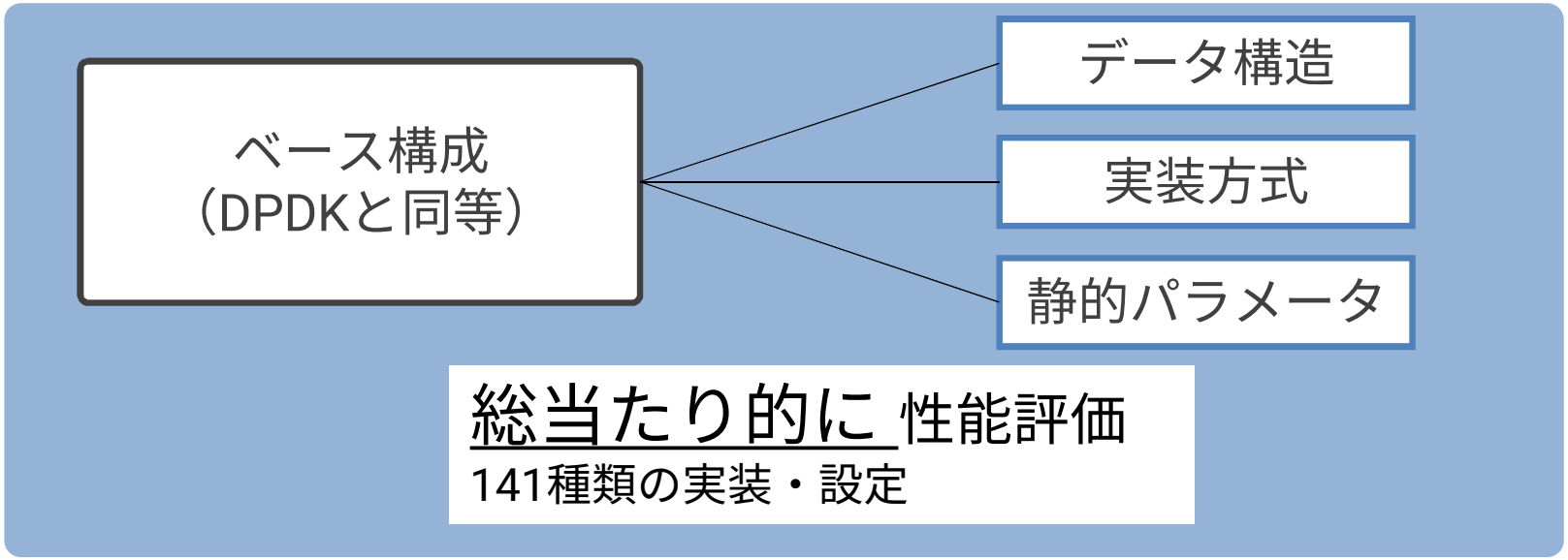




CNFにおける「真の」ボトルネックは何か？

先行研究†による分析

CPUキャッシュの利用効率向上が必要？



得られた知見

- スループットは**100Mpps**を超えうる
- **キャッシュ無効化**が真の性能低下要因
- 無効化抑制のカギは**メッセージバッファ**

メタ情報領域の排除

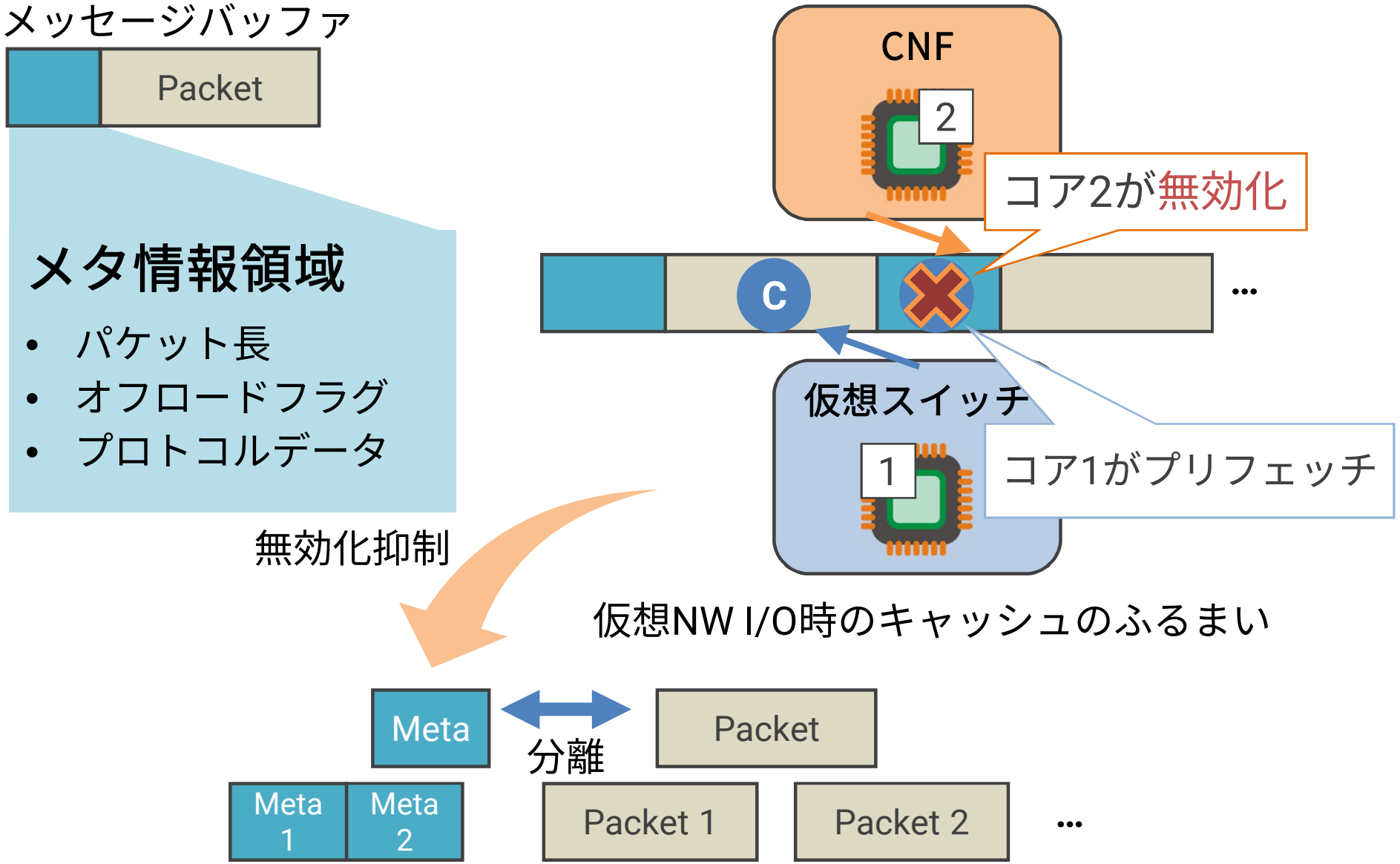
×機能性

バッファ構造の精査が不十分

さらなる変更とその評価が必要

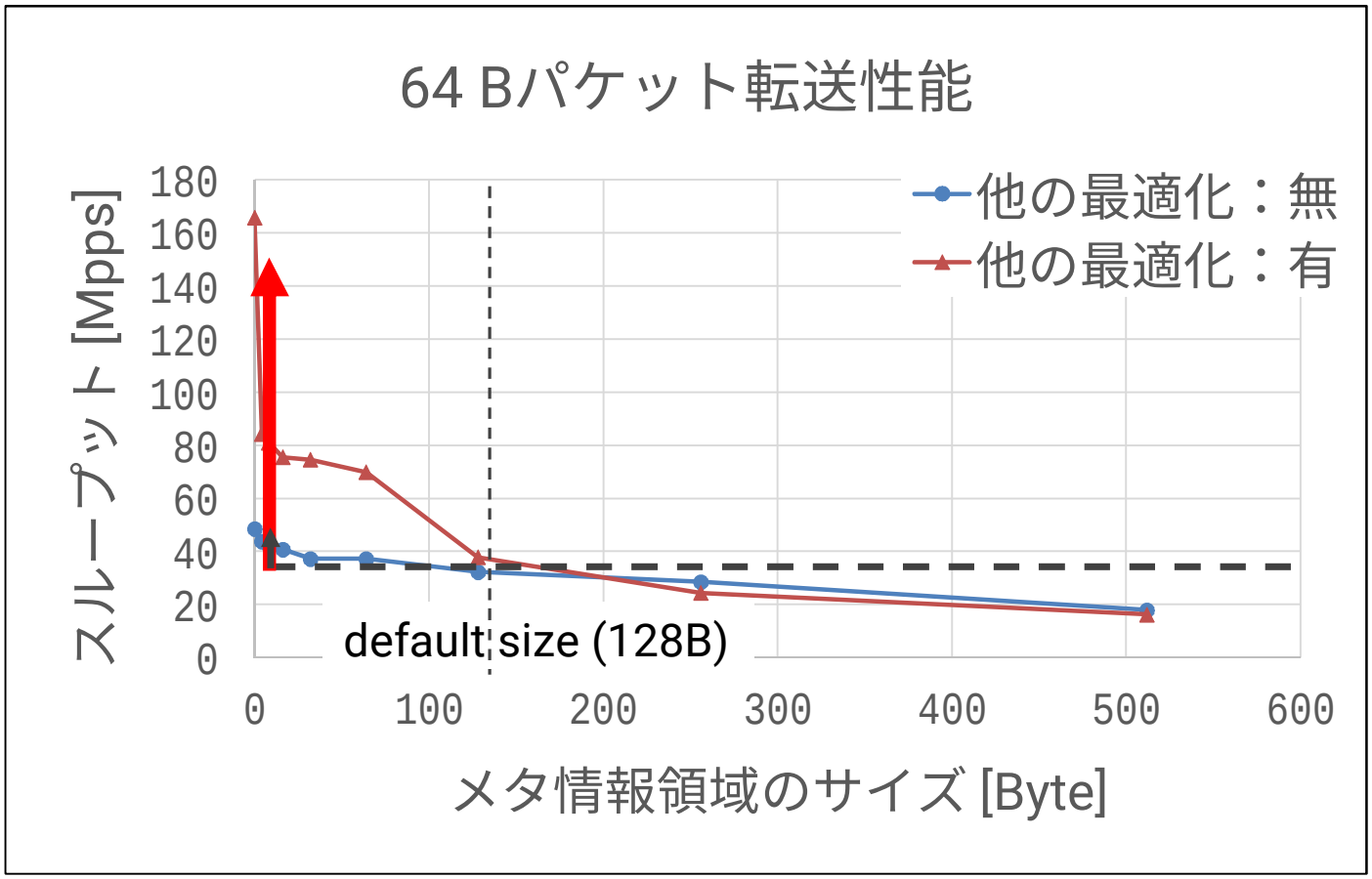
† D. Takeya, et al., “Understanding Roadblocks in Virtual Network I/O: A Comprehensive Analysis of CPU Cache Usage”, IEEE NetSoft ’23

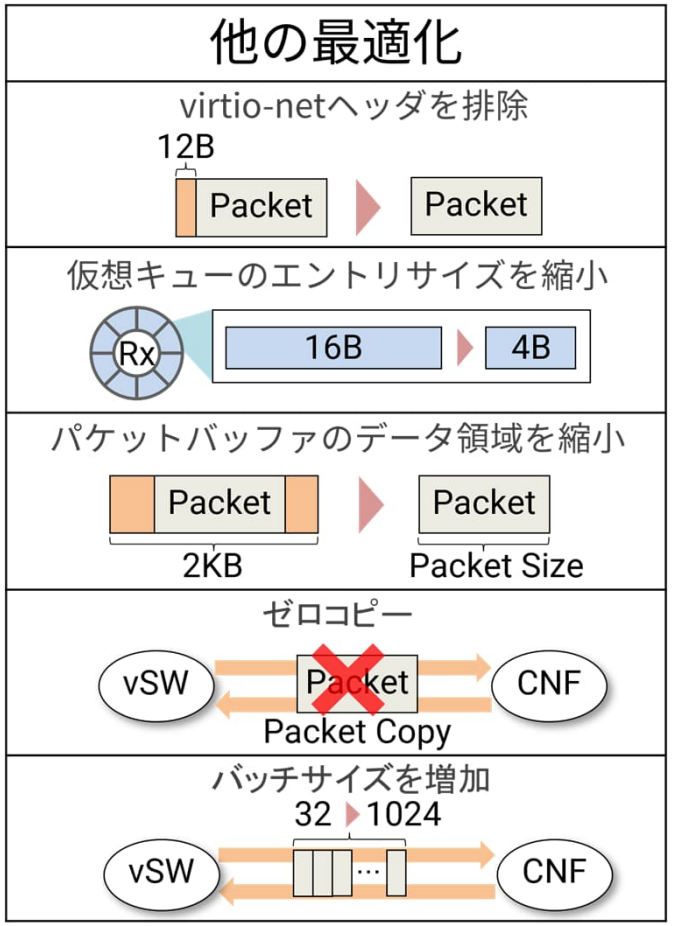
メッセージバッファ再考



メタ情報領域の配置・集約の方法 を検討しなければならない

メタ情報領域のサイズ縮小





再考の効果を得るためには，メタ情報領域の縮小が必要

現実的な状況下でどの程度まで縮小できるのか？

実際のNFが使用するメタ情報

cf. DPDKのメタ情報領域 (128 B)

単純

L2/L3スイッチ

6B (パケット長，ポート番号)

- ACL
- Security Gateway

6B (パケット長，ポート番号)

ロードバランサ

6B

DPI

14B = 6B + 8B (タイムスタンプ)

複雑

その他

- オフロードフラグ
- プロトコルデータ
- マルチセグメント用データ
- フロー分類用データ

89.1–95.3%サイズ削減可能

本発表の位置づけ

研究全体の目的

実用性※を担保しつつCNFで先行研究相当の性能を出す

※ 実用性：既存フレームワーク上で動作する「任意の」NFに適応できる

メッセージバッファ再考の流れ

サイズ

- 実用性に影響
- 必要条件

本発表

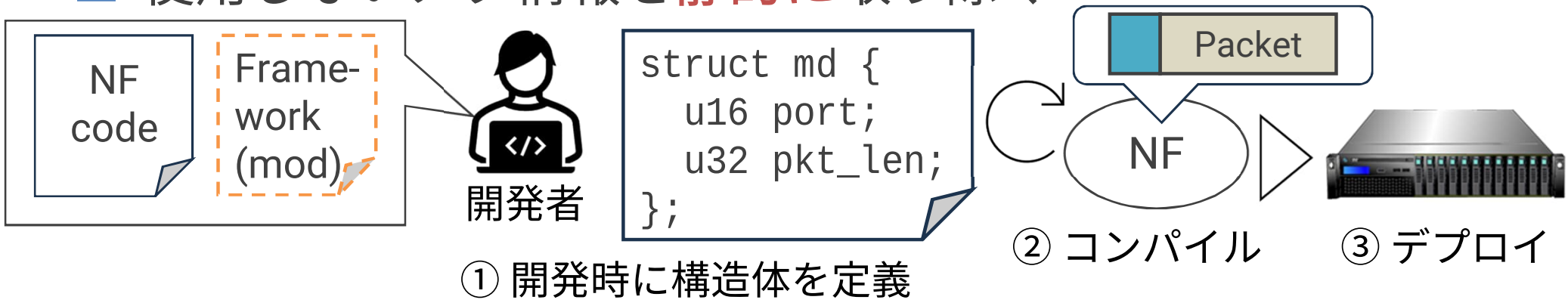
配置・集約

サイズと合わせて
性能を左右

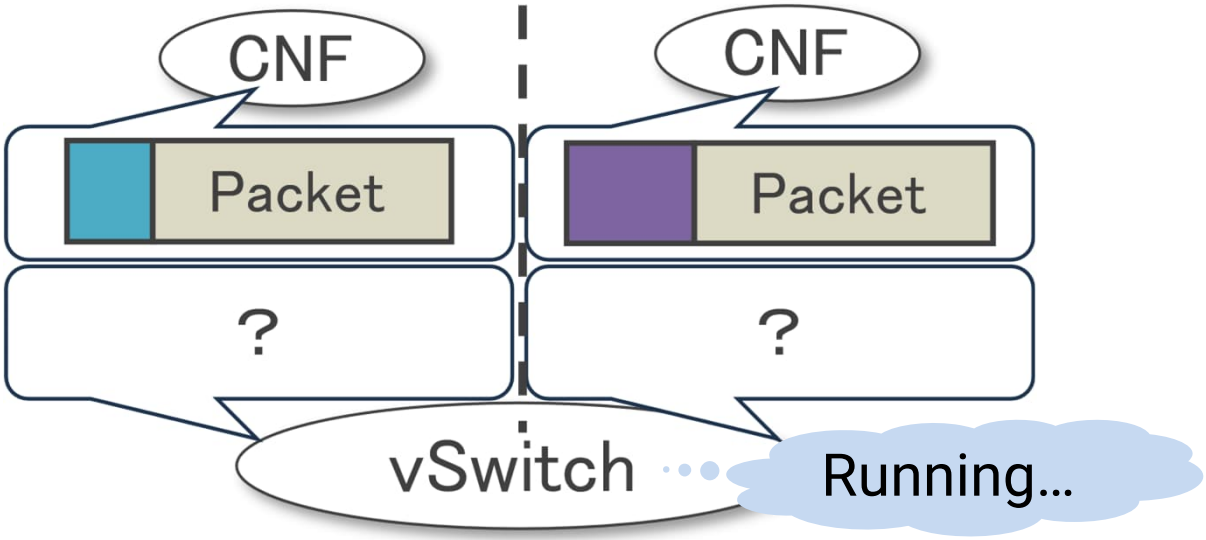
次のテーマ

関連研究

- PacketMill†
  - 使用しないメタ情報を**静的に**取り除く



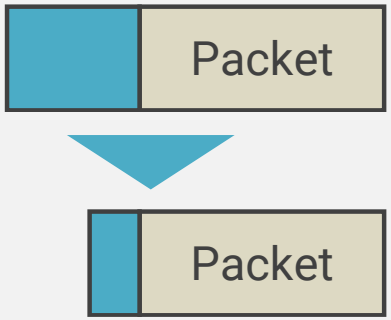
マルチテナント環境で利用できない



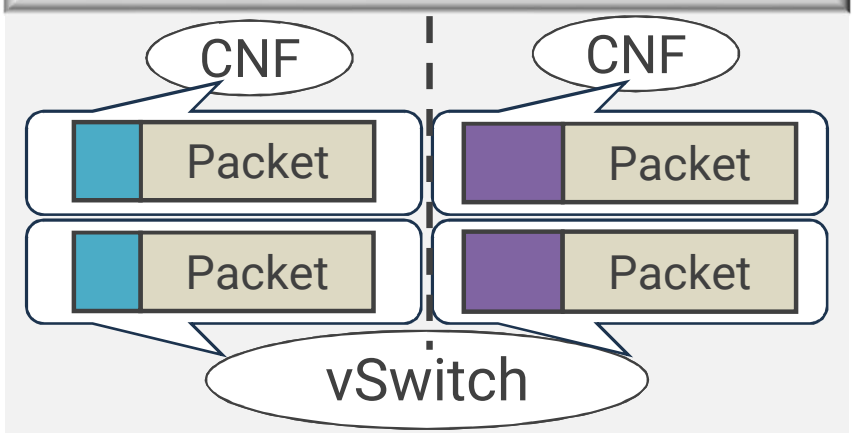
† A. Farshin, et al., "PacketMill: toward per-Core 100-Gbps networking", ASPLOS '21

要件定義 (機能要件)

1. メタ情報領域の縮小



2. 仮想・マルチテナント環境



3. 対象とするNF・仮想スイッチ

限定しない

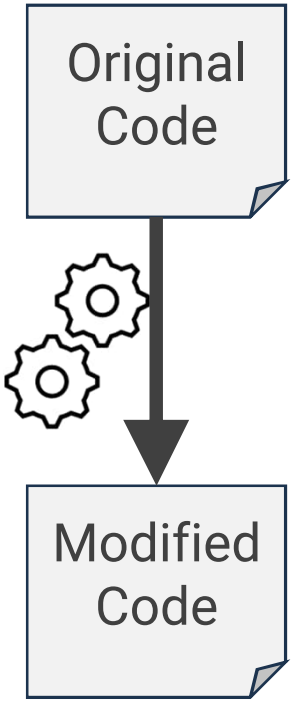
4. 対象のパケット処理フレームワーク

限定しない

要件定義 (非機能要件)

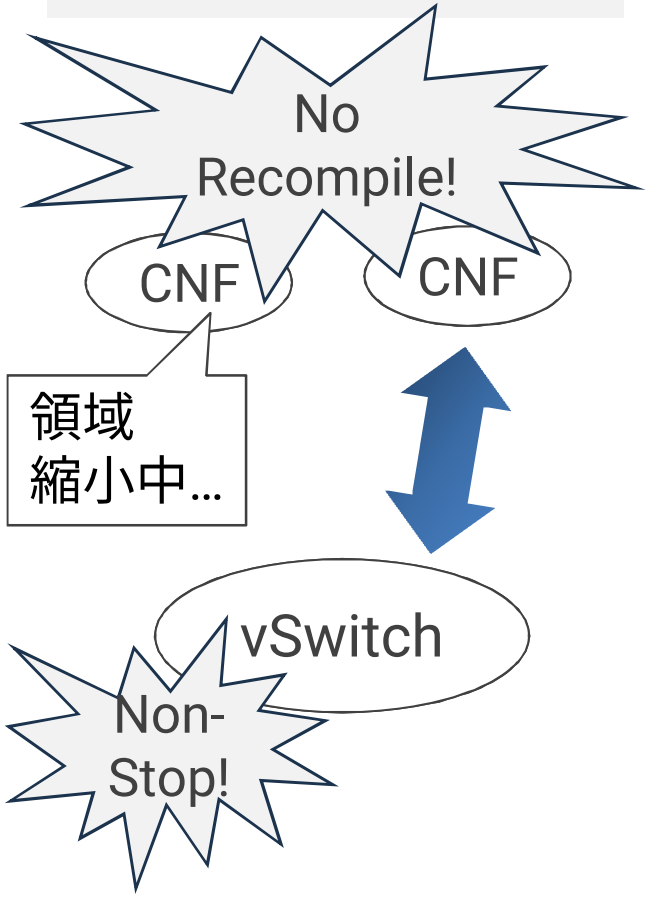
5. NF開発

既存のコードの変更は
機械的に変更可能な
レベルに抑える



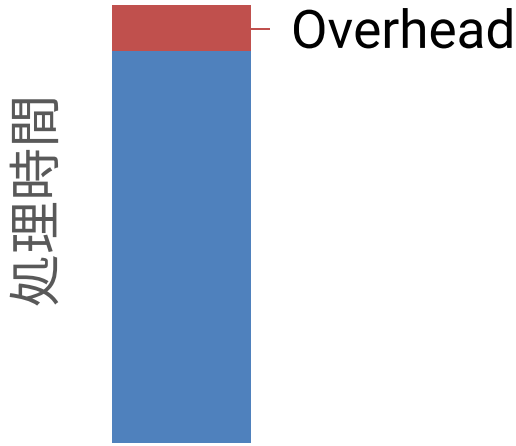
6. 運用

NFの再コンパイルと
仮想スイッチの再起動
を必要としない



7. 性能

提案手法の導入による
オーバヘッドは
無視できる



提案手法の戦略

宣言制に基づくメタ情報領域の構築

1.縮小, 3.NF/vSW, 4.FWK

仮想スイッチによるメタ情報領域の掌握

2.仮想, 6.運用

メタ情報へのアクセス処理のプラグイン化

1.縮小, 3.NF/vSW, 4.FWK, 5.開発, 6.運用, 7.性能

宣言制に基づくメタ情報領域の構築

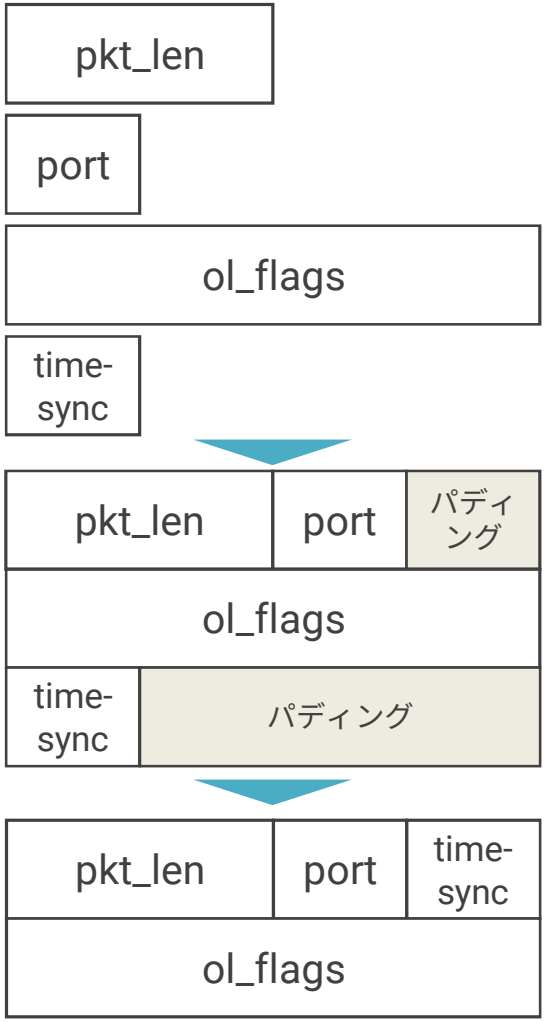
NFが「実際に使用する」メタ情報を事前に宣言

- パケット処理フレームワークおよびバージョン
- メタ情報の識別子

```
{
  “framework”: {
    “name”: “DPDK”,
    “version”: “22.11”
  },
  “metadata”: [
    “port”,
    “pkt_len”,
    “timesync”,
    “ol_flags”
  ]
}
```

メタ情報領域構築アルゴリズム

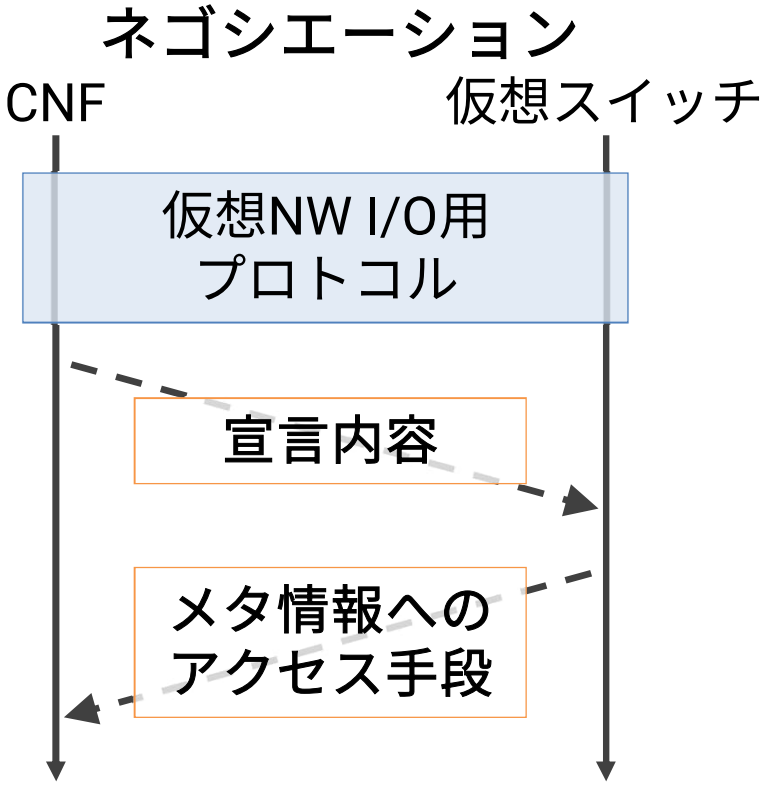
- Structuring
- Padding
- Reordering (optional)

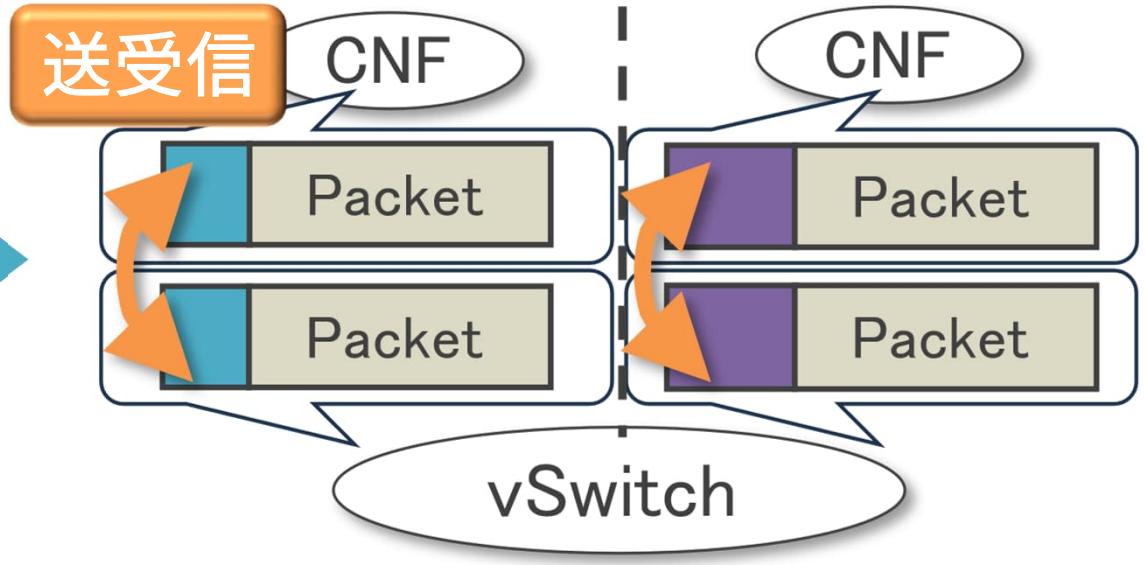


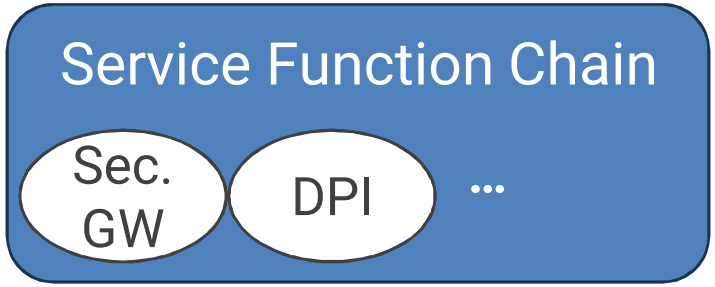
CNFに応じて最適なメタ情報領域を自動構築

仮想スイッチによるメタ情報領域の掌握

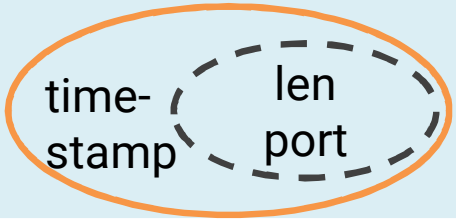
仮想・マルチテナント環境をサポートするため，
仮想スイッチが各CNFのメタ情報を管理する







メタ情報の「和集合」で全CNFに対応



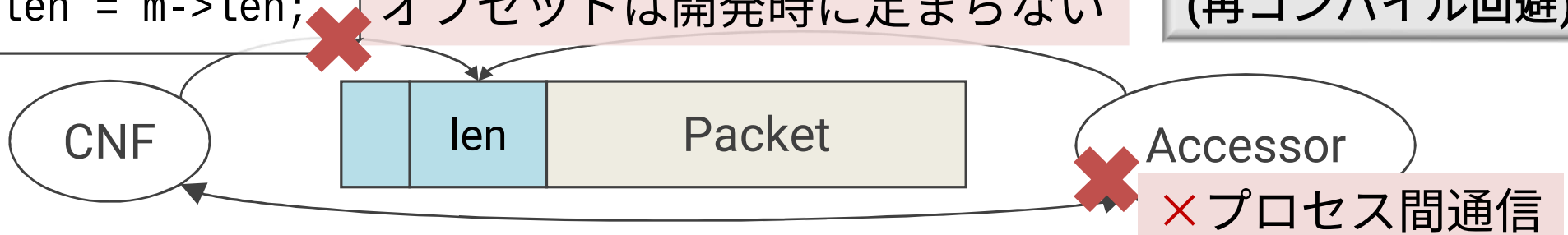
メタ情報へのアクセス処理のプラグイン化

NFはメタ情報へのアクセス手段を新たに要する

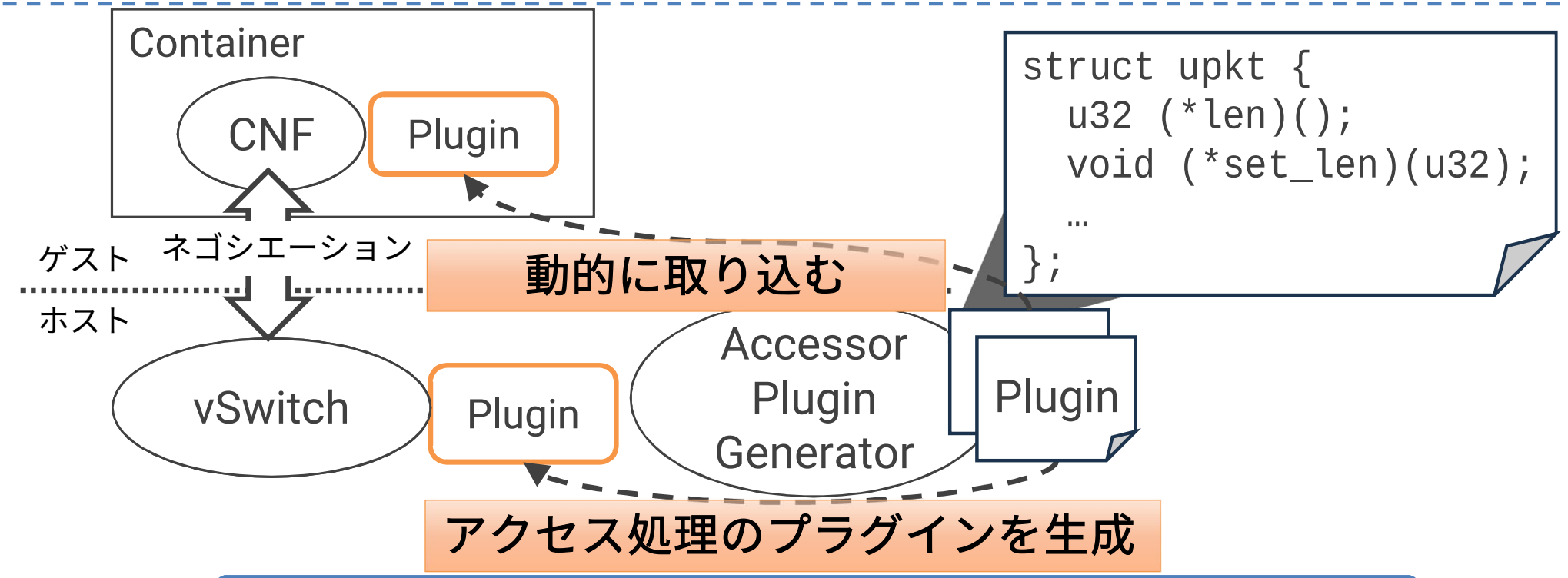
要件6：運用
(再コンパイル回避)

```
u32 len = m->len;
```

オフセットは開発時に定まらない

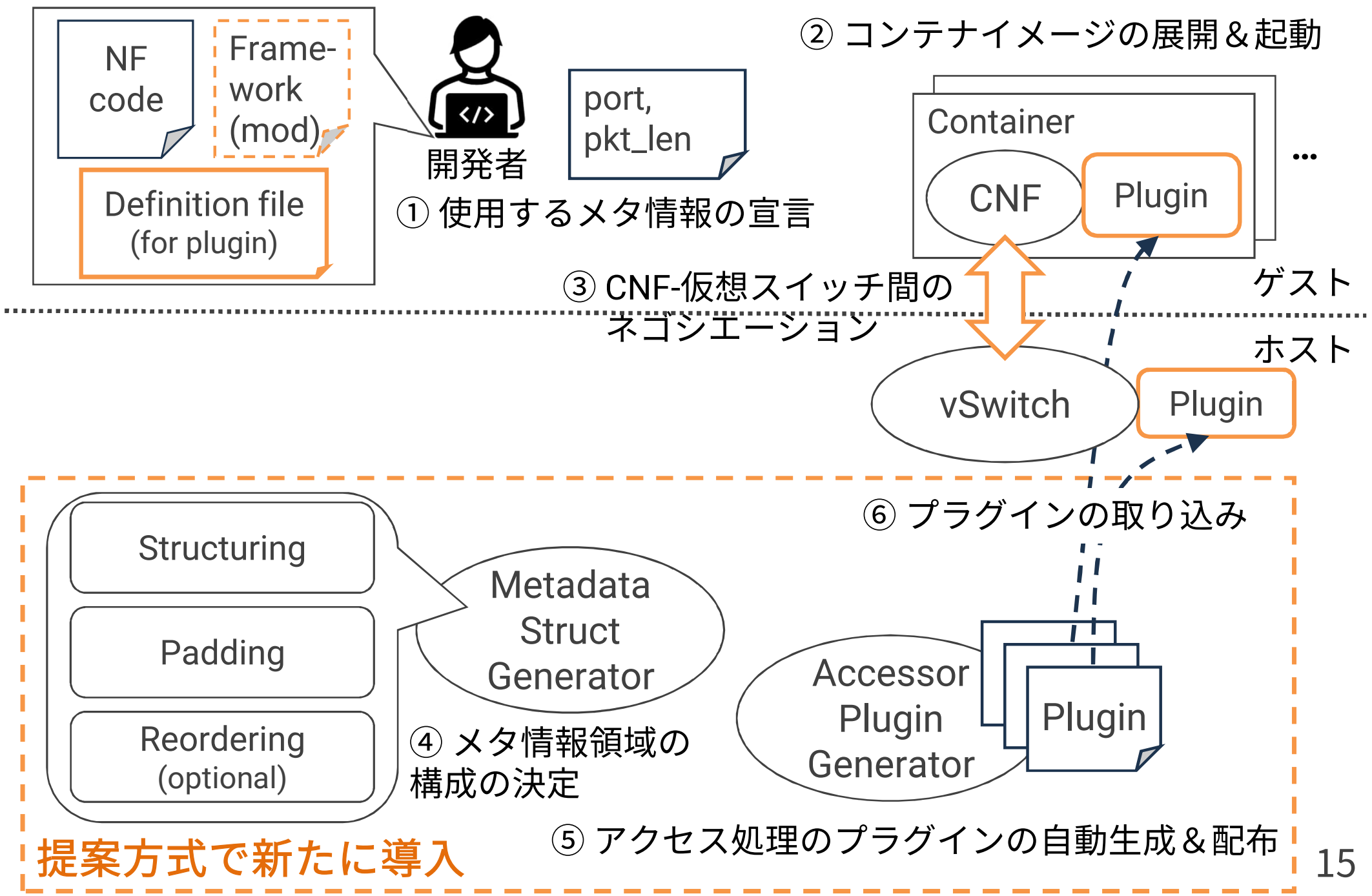


×プロセス間通信

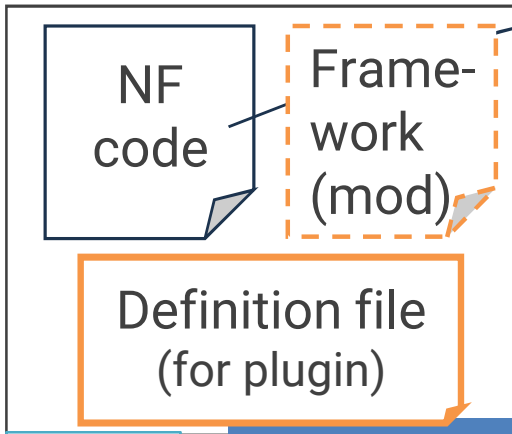


再コンパイルせずにアクセス手段を提供するため，
アクセス処理をインターフェースと実装に分離する

システムの全体像


NF code
Frame-work (mod)
Definition file (for plugin)
開発者
port, pkt_len
① 使用するメタ情報の宣言
② コンテナイメージの展開＆起動
Container
CNF
Plugin
...
③ CNF-仮想スイッチ間のネゴシエーション
ゲスト
ホスト
vSwitch
Plugin
⑥ プラグインの取り込み
Structuring
Padding
Reordering (optional)
Metadata Struct Generator
④ メタ情報領域の構成の決定
Accessor Plugin Generator
Plugin
⑤ アクセス処理のプラグインの自動生成＆配布
提案方式で新たに導入

プラグインの利用方法と実装



開発者

関数

アクセス処理の内部実装

```
void
set_len(struct upkt *up, uint32_t len)
{
  up->md->len = len;
}
```

オフセット

```
inline void
set_len(struct upkt *up, uint32_t len)
{
  uint8_t *buf = (uint8_t *)up->buf;
  *((uint32_t *)&buf[table[LEN_IDX]]
    = len;
}
```

バリエーションがある

利用方法

```
struct upkt up;
init_plugin(&up);
```

初期化

```
// Rx
recv(&up);

// Main processing
uint32_t len = up.len();
```

アクセス処理

```
up.set_len(&up, len);
…
// Tx
send(&up);

exit_plugin();
```

終了

機械的に改変可能

考察

- 性能オーバヘッドは無視できる程度か？

アクセス方法	サイクル数 [cycles / proc]
従来	1.08
関数	4.02
オフセット	1.08

評価環境

CPU	Intel Core i9 11900K @ 3.50GHz
OS	Linux Kernel 5.14.0-284.11.1

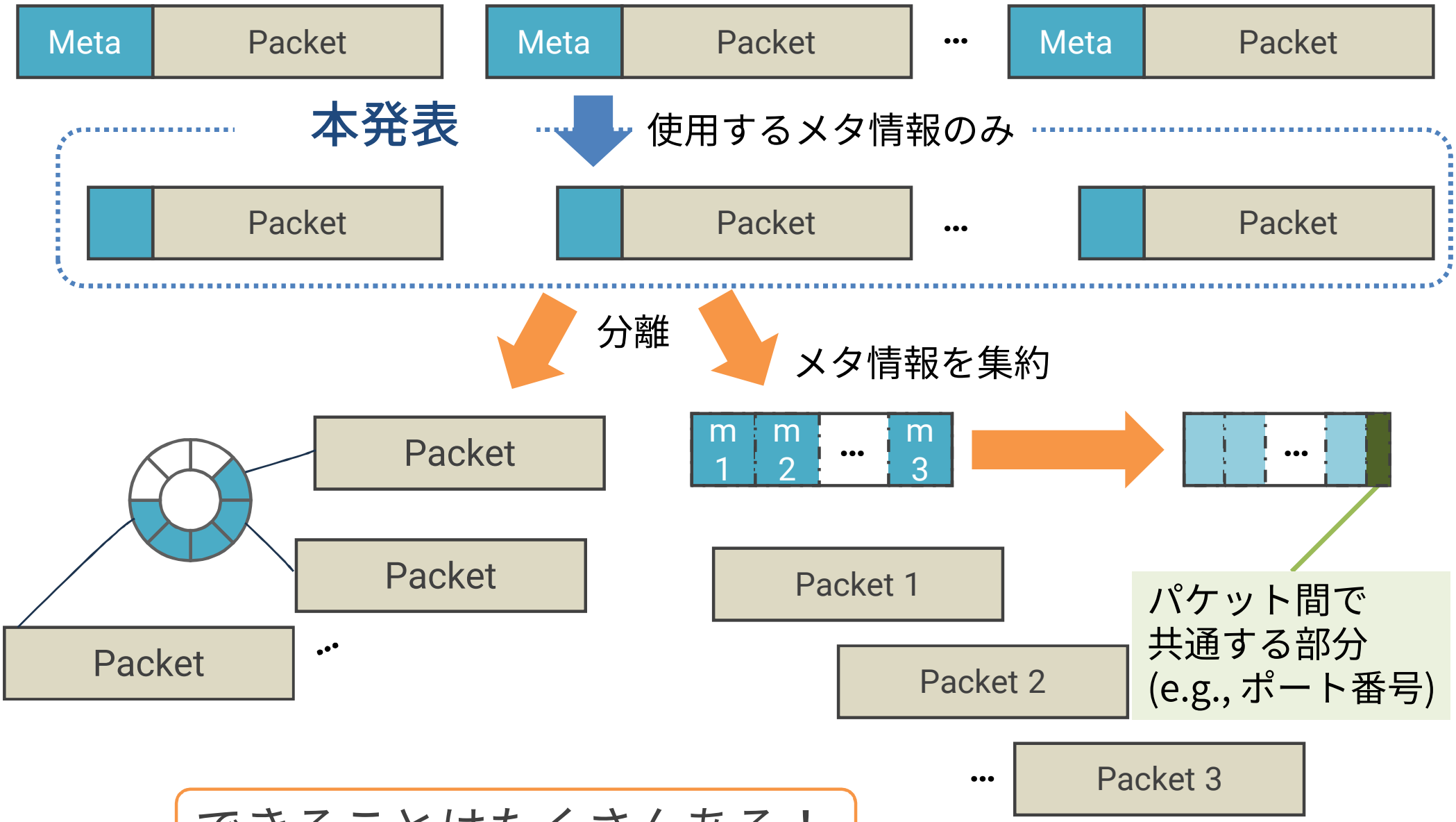
性能劣化の懸念はない

- 実用的か？

	提案手法	既存手法(PacketMill)
メタ情報領域の縮小	☑	☑
仮想・マルチテナント環境	☑	-

提案手法は妥当

今後の取組：メタ情報領域の配置・集約



できることはたくさんある！

まとめと今後の取組

まとめ

- CNFの性能劣化問題を解決するには，
  メッセージバッファの枠組みを根本から見直す必要がある
  - 1. メッセージバッファにおけるメタ情報領域の縮小
  - 2. メタ情報領域の配置やメタ情報の集約方法の検討
- 「1. メタ情報領域の縮小」を解決した
  - 仮想・マルチテナント環境に適合できる縮小方式を提案
  - 性能向上の必要条件を満足

今後の取組

- メッセージバッファの配置やメタ情報の集約方法の検討
  - メッセージバッファについて根本から再考し，
    サイズ削減と合わせて先行研究相当の性能を狙う