SRv6 網の高度な運用を目指した、eBPF/XDP プログラムの自作と効率化

2024年1月19日

NTT コミュニケーションズ株式会社
田島裕也

イントロ

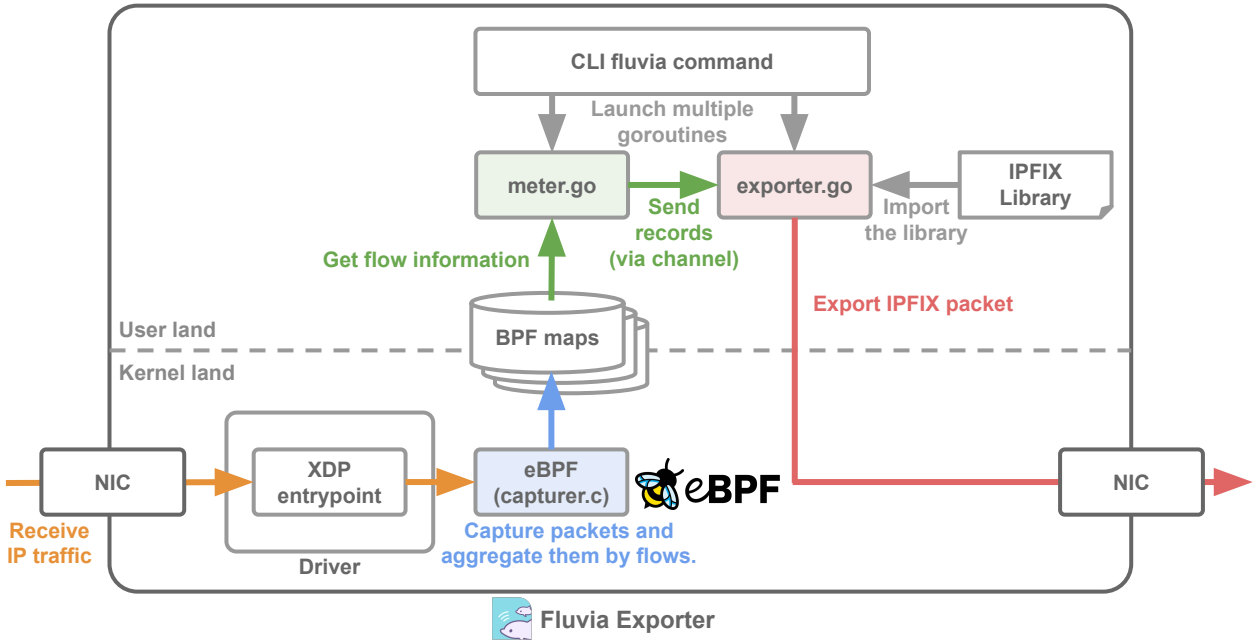
- **ネットワークのオペレーションには可視化が不可欠**
  - **我々の例:SRv6 を用いた社内網を運用中**
    - 将来の商用サービス化を目指し、高度な Traffic Engineering (TE) や障害対策を実装
    - 新規サービス実現のため、満たしたい要件が存在
      - 個別の SLA 達成を目指し、フロー毎の遅延や帯域、経路を TE のメトリクスとして活用

- **SRv6 ヘッダ内の情報や hop-by-hop の遅延など、取得したいが実装のない項目が存在**
  - 以下を取得するプログラム **(IPFIX exporter)** を eBPF/XDP で実装＆社内網に導入
    1. SRv6 ヘッダの各フィールド (SRv6 フローの識別情報)
    2. IPv6 hop-by-hop option ヘッダに付与されたタイムスタンプ

  → **サービスへの影響を抑えるため、転送遅延が少なくなる形に改良！**

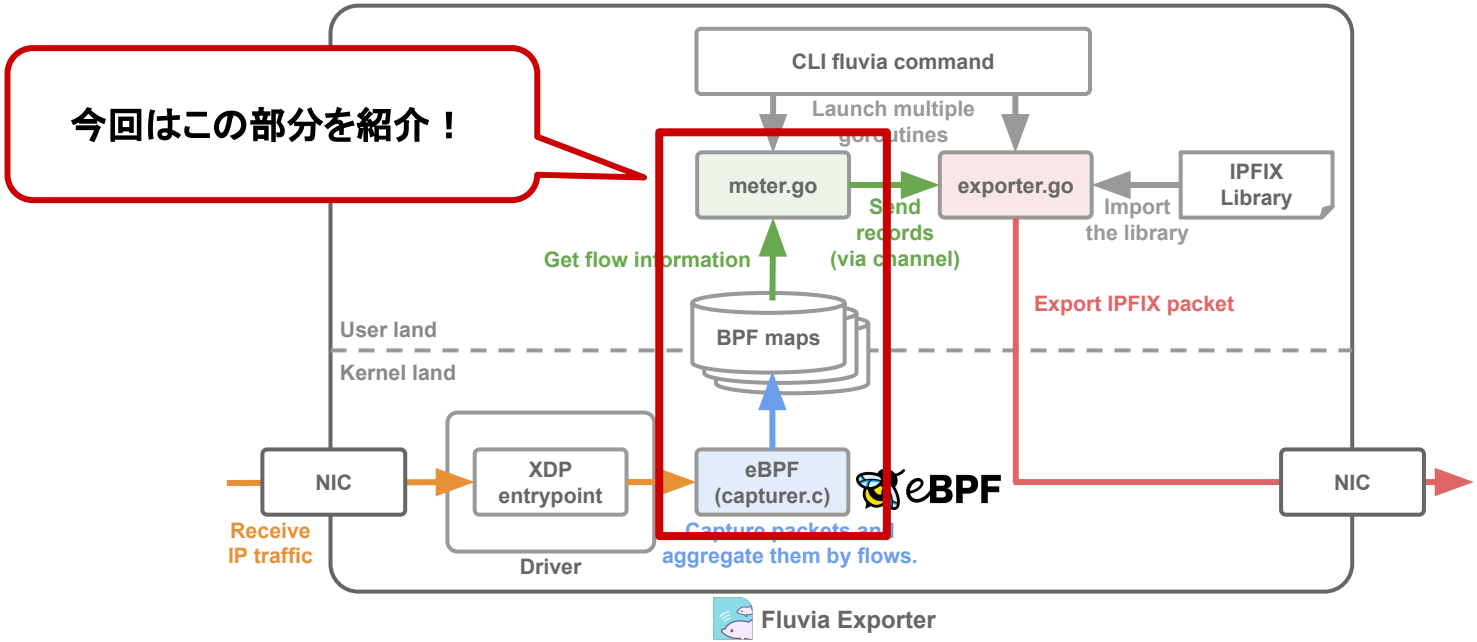
（参考）IPFIX exporterのアーキテクチャ

- 3つのコンポーネント（capturer, meter, exporter）
  - パケットの取得、解析、IPFIX の export を実施
- eBPF/XDP ベースの IPFIX exporter
  - eBPF は capturer（パケットの取得）を担当。meter（パケットの解析）へ eBPF map を介して情報伝達



（参考）IPFIX exporterのアーキテクチャ

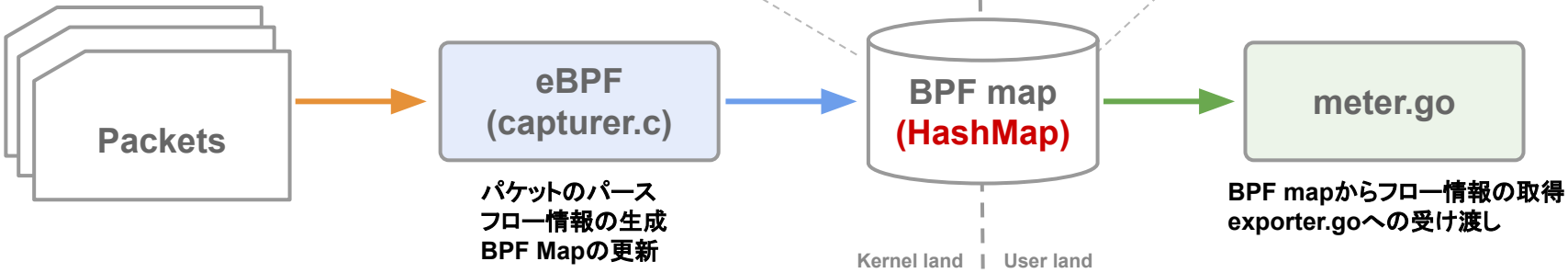
- 3つのコンポーネント（capturer, meter, exporter）
  - パケットの取得、解析、IPFIX の export を実施
- eBPF/XDP ベースの IPFIX exporter
  - eBPF は capturer（パケットの取得）を担当。meter（パケットの解析）へ eBPF map を介して情報伝達



改良前の実装（愚直な実装）

- SRv6 ヘッダの情報をパケットから抽出し、それをBPF Map に保存
- Map には BPF_MAP_TYPE_LRU_HASH を使用
  - Key には IP アドレスや SRv6 のアトリビュートを使用し、Value はカウンタとして使用

Src IP	Dst IP	…	Count
2001:db8::1	2001:db8::2	…	1
2001:db8::3	2001:db8::4	…	3



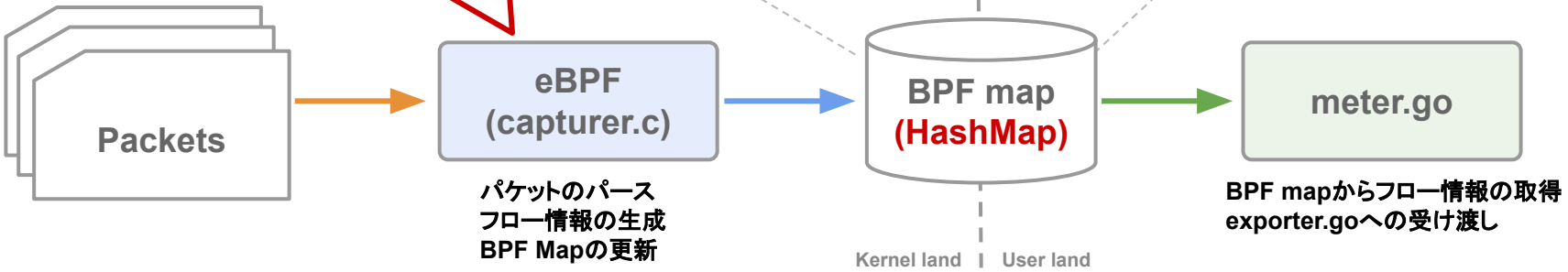
改良前の実装（愚直な実装）

- SRv6 ヘッダの情報をパケットから抽出し、それをBPF Map に保存
- Map には BPF_MAP_TYPE_LRU_HASH を使用
  - Key には IP アドレスや SRv6 のアトリビュートを使用し、Value はカウンタとして使用

本実装ではeBPFプログラムの処理が完了した後に、パケットがカーネルに引き渡され、転送が再開される

= eBPFの処理が転送をブロックする

Src IP	Dst IP	…	Count
2001:db8::1	2001:db8::2	…	1
2001:db8::3	2001:db8::4	…	3

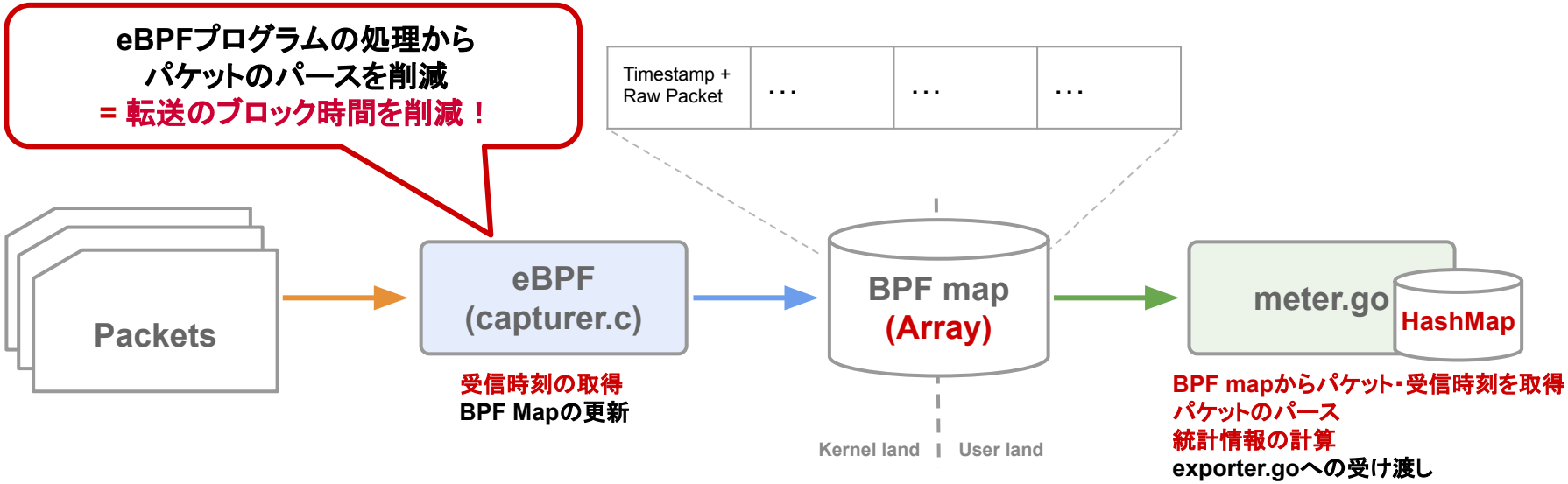


パケットのパース
フロー情報の生成
BPF Mapの更新

BPF mapからフロー情報の取得
exporter.goへの受け渡し

改良後の実装

- **効率的な転送を目指し、eBPFでの処理をユーザランドへと移行**
- BPF_MAP_TYPE_PERF_EVENT_ARRAY を使用
  - HashMap ではなく、Array（配列）
  - 本実装では、パケットを受け取った時刻と生のパケットをユーザランドに送る



パイプラインの比較（改良による効果）

観点	改良前の実装	改良後の実装
パケット転送への影響	△ XDP の処理の中で Map へのエントリの保存もしくは更新をパケットをカーネルに流す前に行っている	○ XDP は Array へパケットとタイムスタンプを流すだけで、パケットのパースや統計情報の計算はユーザランドで実行可能
パースのしやすさ	△ eBPFの制限によってパース処理を記述するのが難しい	○ eBPFの制限がないため比較的容易
統計情報の蓄積	× 固定長の BPF map では限界がある	○ 可変長の Mapで保存できる - Golang の HashMap

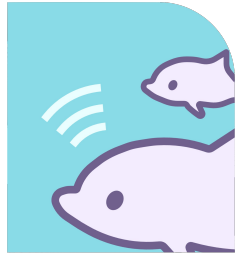
→ 我々が目指すサービス要件を満たしつつ、転送効率への影響を最小化

まとめ

- **BPF map 設計の改良により、転送効率を犠牲にしないフロー情報の取得を実現**
  - カーネルでの処理をユーザランドに移行することで、転送遅延を最小化

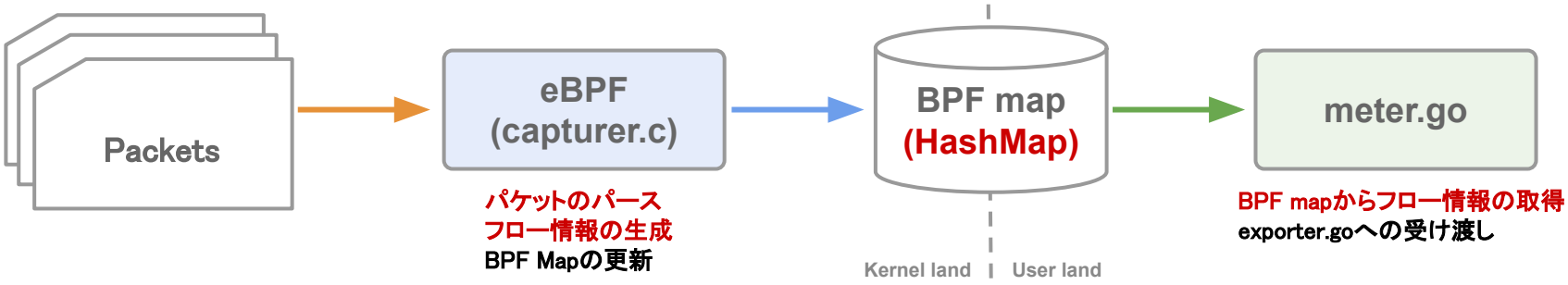
- **今回取得に成功した情報を基に、将来的なSRv6 網の可視化や自動化、高度なTE が可能に**
  - eBPF/XDP により、アイデアの高速な実現に成功
    - **ただし、実用化のためには高速な転送が不可欠 → 今回の工夫により対応！**
  - 今後も更に高度なネットワークサービスを目指し、SRv6網を運用していきます！

- **OSS として公開中！ Contribution もお待ちしています**
  - **Fluvia Exporter: https://github.com/nttcom/fluvia**

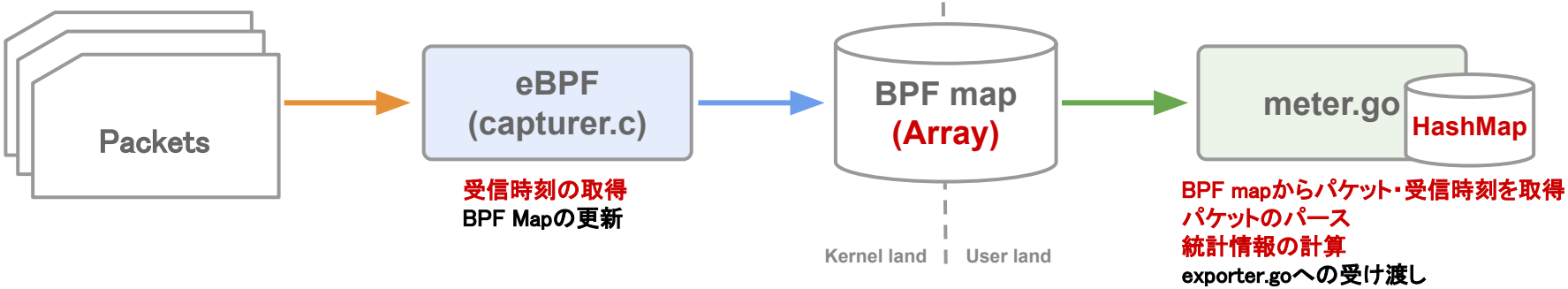


（参考）パイプラインの比較

改良前



改良後



（参考）パイプラインの比較