SDN における自己追従型細粒度ネットワーク管理運用アーキテクチャ

柴田 幸太郎† 中山 裕貴‡
林 經正‡ 阿多 信吾†

† 大阪市立大学 大学院工学研究科 ‡ 株式会社ボスコ・テクノロジーズ

SDN を用いた管理運用技術

- SDN (Software Defined Networking) の台頭
  1. SDN 技術を用いたネットワーク管理運用に関する研究が活発化
  2. ネットワーク構成だけでなく仮想ネットワーク機能も動的に構成することが可能になった
  3. 多様なアプリケーションに対応可能なトラヒック制御の実現が期待されている
- SDN を用いたアプリケーショントラヒック制御
  1. トラヒックパターンは生成するアプリケーションに依存する傾向が強い
  2. アプリケーショントラヒックを適切な粒度のセッションフローに分類して、必要なネットワーク機能も提供

     例）「接続始めのセッションだけ速く、残りは遅く」
     「認証セッションだけ速く、データ転送は遅く」
     「正常トラヒックは速く、異常トラヒックは遅く」

制御ポリシの変動

1. アプリケーションやトラヒックパターンは時間経過に伴い大きく変化
2. 制御ポリシ自体も時間遷移と共に変化
3. トラヒック制御のフレームワークだけでは不十分
4. トラヒックだけでなくポリシ設定も動的に制御・適応可能なフレームワークが必要

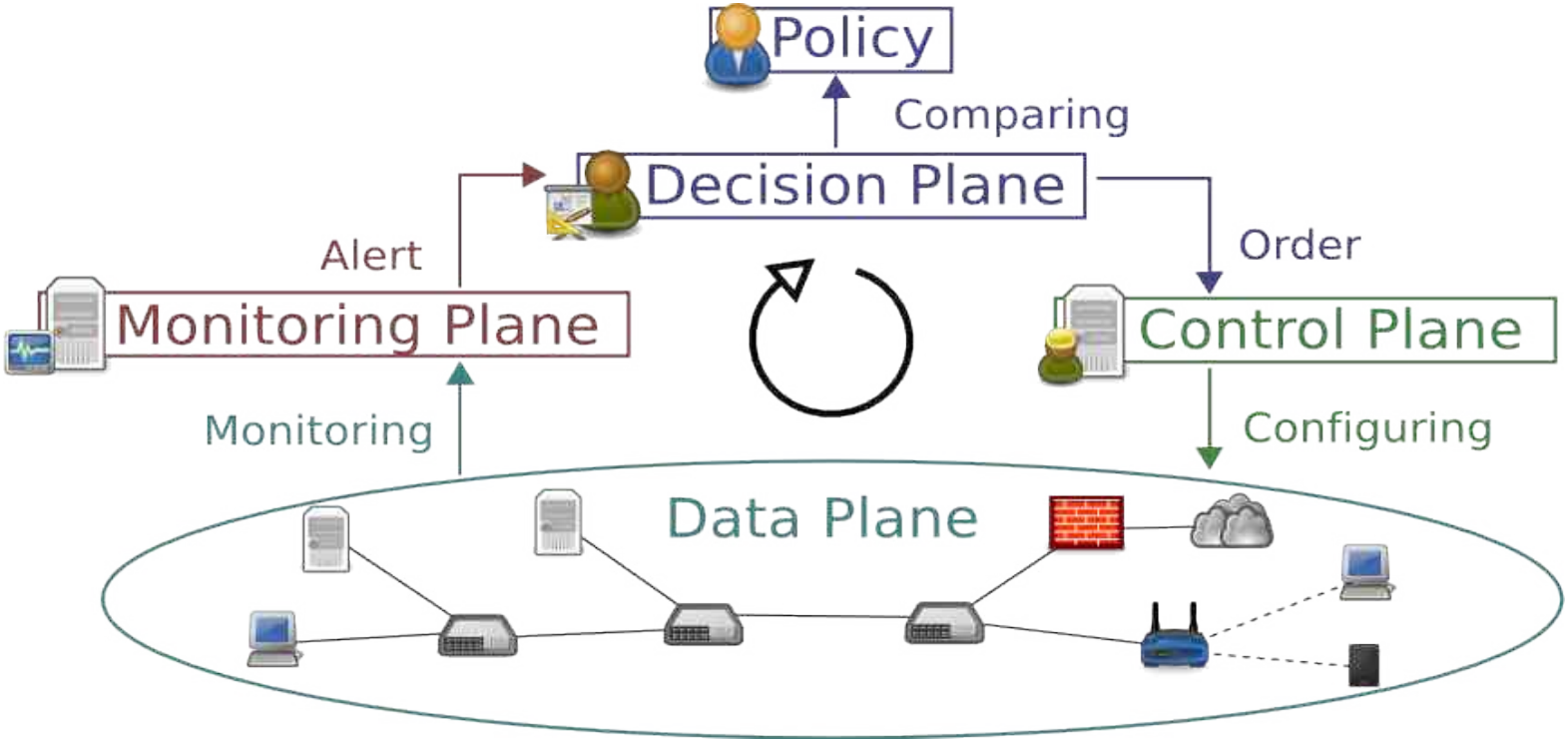
以上から本研究では、

ネットワークトラヒックだけではなくネットワーク機能などの資源を統合的に運用する適用性の高い追従型ポリシ制御フレームワーク

を提案する。

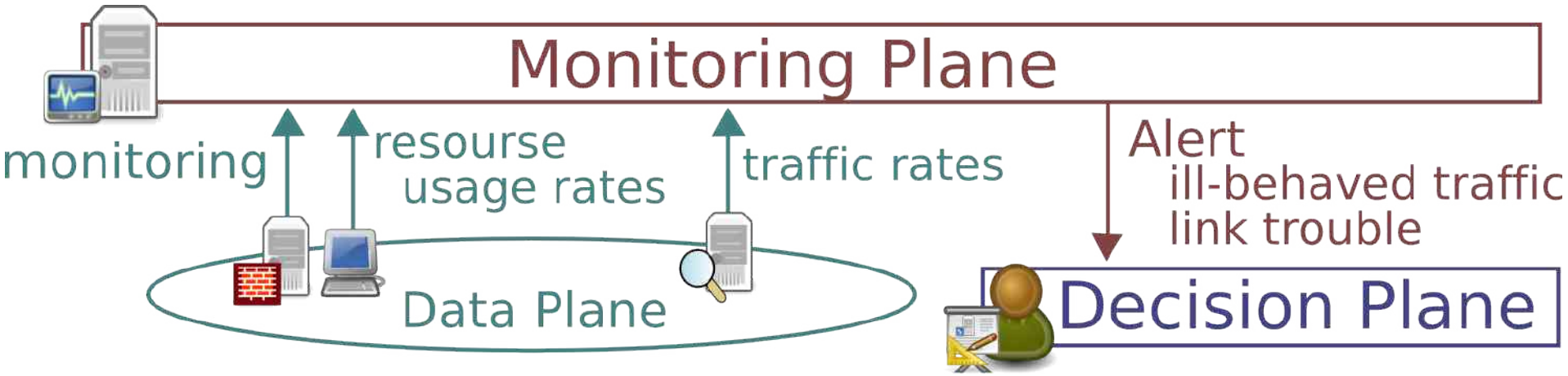
提案アーキテクチャ概要

- データプレーン（データプレーン）
  - 管理対象のネットワークは SDN スイッチで構築
  - 計算機リソースもネットワーク内に存在
- データプレーンを管理するプレーン
  - 監視プレーン、決定プレーン、制御プレーンの3つから構成



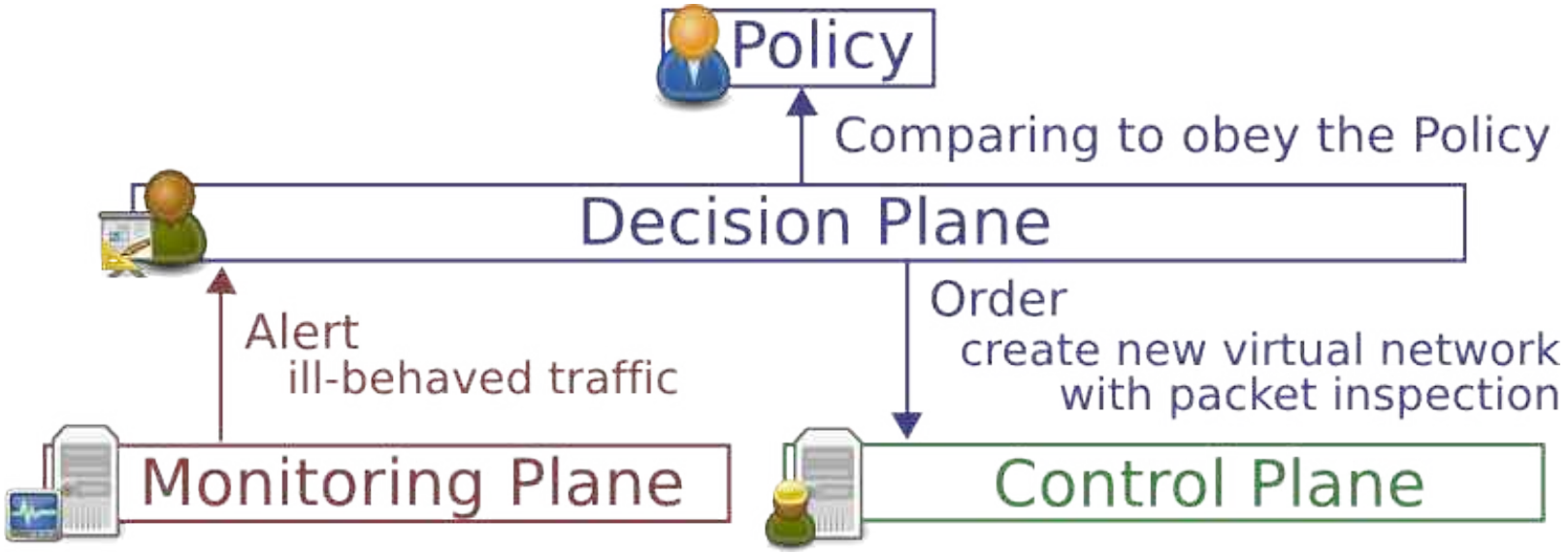
監視プレーン

- 端末、リソースプール、および SDN スイッチの利用状況等の情報を収集
  - 収集したネットワーク情報を他のプレーンへ提供する
- 決定プレーンへアラートを通知
  - 決定プレーンの動作の引き金となる
- 監視プレーンが独立することで、決定プレーンで物理資源を仮想的に管理することが可能に
  - 異なる環境へのポリシの置き換えが容易になる



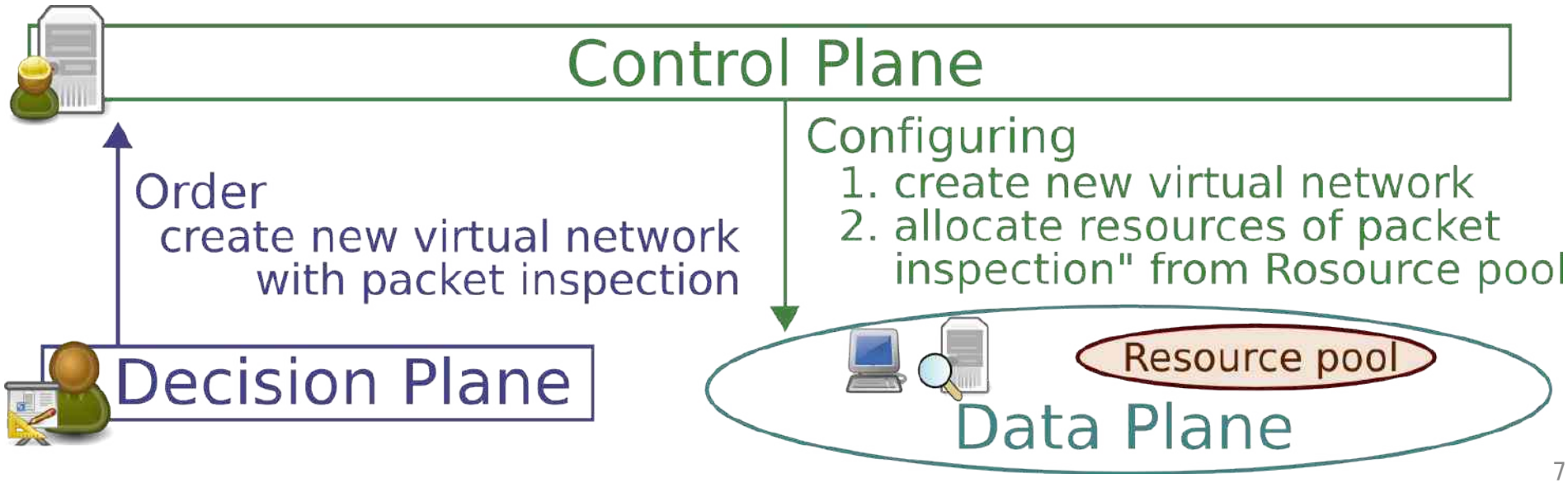
決定プレーン

- 問題の最適な再設定の構成を準備
  - ポリシを適応しつつも制限された物理資源下で、目的の機能を最大限活用する解決策
- 再設定の構成を制御プレーンへ送付
  - 仮想ネットワークの生成・削除、トポロジ構成の変更
  - 仮想リンクの帯域割当の変更
  - 仮想ネットワーク機能の提供



制御プレーン

- デバイスコントローラへ実制御のためのメッセージを送信
  - 決定プレーンから送られた設定更新計画に従う
  - SDN スイッチ、計算機リソースプール
- 新しいアラートの設定・更新
  - 次の制御ループで利用するため



PDCA サイクル

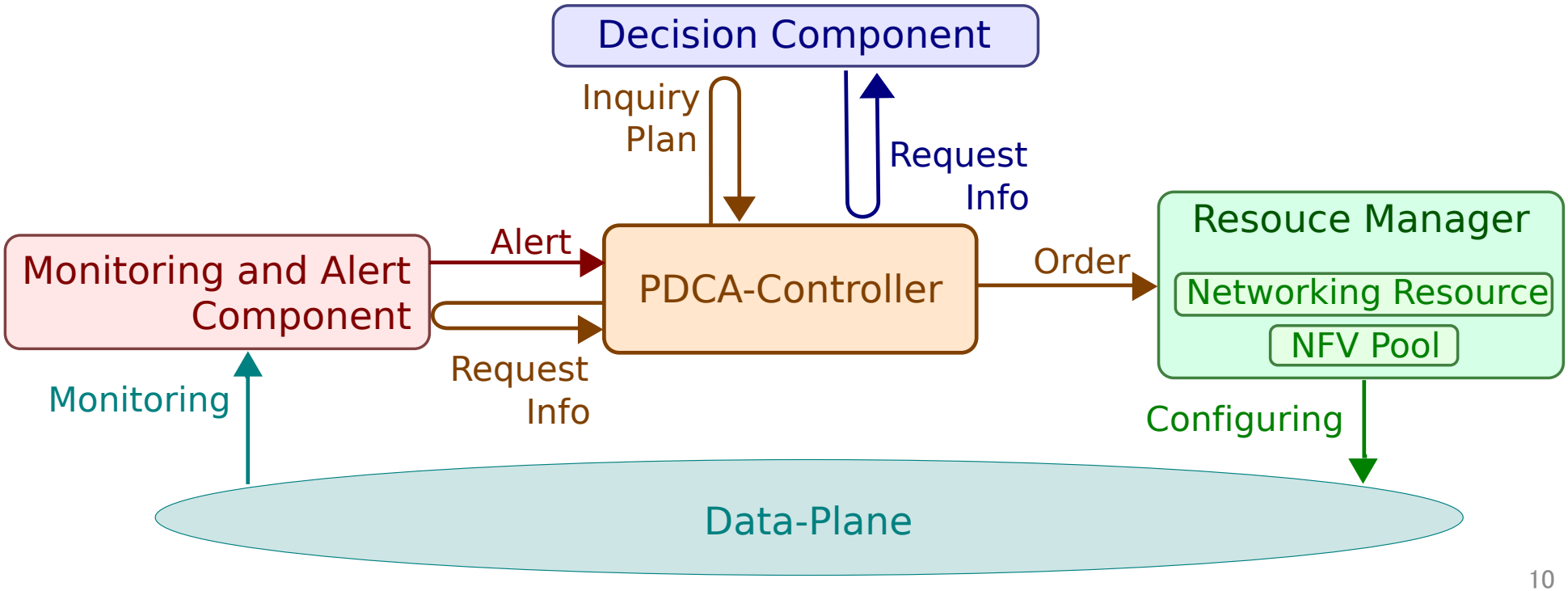
- フィードバックを含む段階的な更新を行うアジャイルな制御手法
  - 時々刻々と変化するトラヒックに対して持続的に適応可能なポリシフレームワークを構築するために利用
- Plan -> Do -> Check -> Act の4フェーズから構成
  - 問題決定 （Plan フェーズ）
    - 監視プラーンからのアラートを解析
    - 問題解決のためのガイドラインを作成
  - 問題設定および制御 （Do フェーズ）
    - ガイドラインに従い実際に設定更新
  - 設定の検証とフィードバック （Check フェーズ）
    - 設定更新により問題が解決されたかの確認
    - 次のサイクルのためのアラートの設定
  - アラート （Act フェーズ）
    - ポリシを維持するためのアラートの有効化
    - 変化した環境に対応するための新しいポリシの更新

ポリシスライス

- PDCA サイクルが独立して動作可能なポリシの最小構成
  - ポリシはポリシスライスの集合として表現
  - ポリシをフレームワーク内において容易かつ段階的に実装するため
  - 制御対象となるトラヒックに対し、検出から対応、検証までの責任を持つ
- グローバル PDCA サイクルを導入
  - 複数のポリシスライスの競合を避けるため
  - ポリシスライスの再構成を行う

構成コンポーネント

- PDCA サイクルを実現するためのシステムコンポーネント
- 4つのコンポーネントから構成



PDCA コントローラ

- グローバル PDCA サイクルとポリシスライスの PDCA サイクルを駆動するためのコンポーネント
- PDCA の各フェーズに対応する4つのステート
  - Plan ステート
    - アラートを元に問題の解決策を準備
    - アラートのフィルタリングや集約化
  - Do ステート
    - 解決策に従い、他コンポーネントへ制御要求を送信
  - Check ステート
    - リソース状態のレポートから問題解決が行われたかを確認
  - Act ステート
    - 次の PDCA サイクルに必要なアラートを設定
    - グローバル PDCA サイクルによるポリシスライスの再構成
    - 管理者による新しいポリシの導入

計測および通知コンポーネント

- 全ての管理リソースの監視を集中制御
  - デバイスの監視
    - エージェントを用いた内部状態の取得
    - 疎通確認やポートスキャンによる外部からの情報取得
  - アラートの生成
  - 他コンポーネントへ収集した監視情報を提供

決定コンポーネント

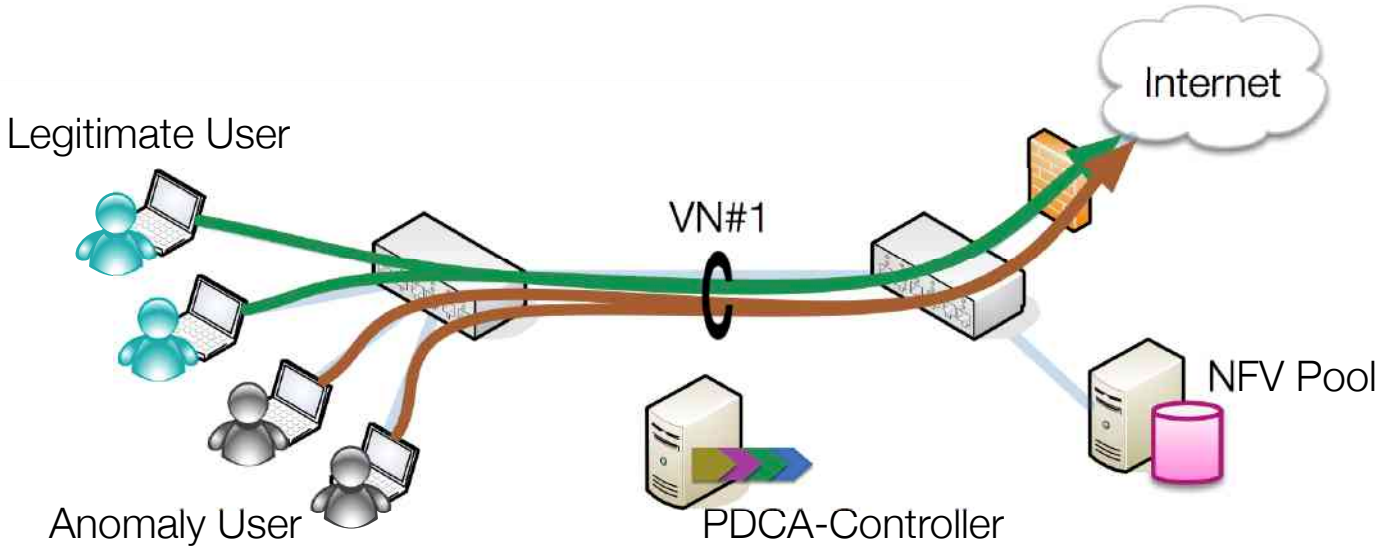
- アラートに関連付けられた問題を解決する設定の更新計画を提供
  - アラートメッセージ
    - アラートの詳細情報や識別子、深刻度などを含む
  - 問題解決に必要な追加情報を PDCA コントローラへ要求
    - ネットワークの利用状況
    - 計算資源の利用状況

リソースマネージャ

- 全てのネットワークあるいはトラヒック制御のための計算資源を管理
  - 動的な仮想ネットワークの生成・削除のためのネットワーク資源
  - 動的な機能割当のための計算資源
- 一時的な資源の割当
  - テスト環境として利用する使い捨てのネットワーク
  - 高負荷なネットワーク機能を必要な時にだけ提供

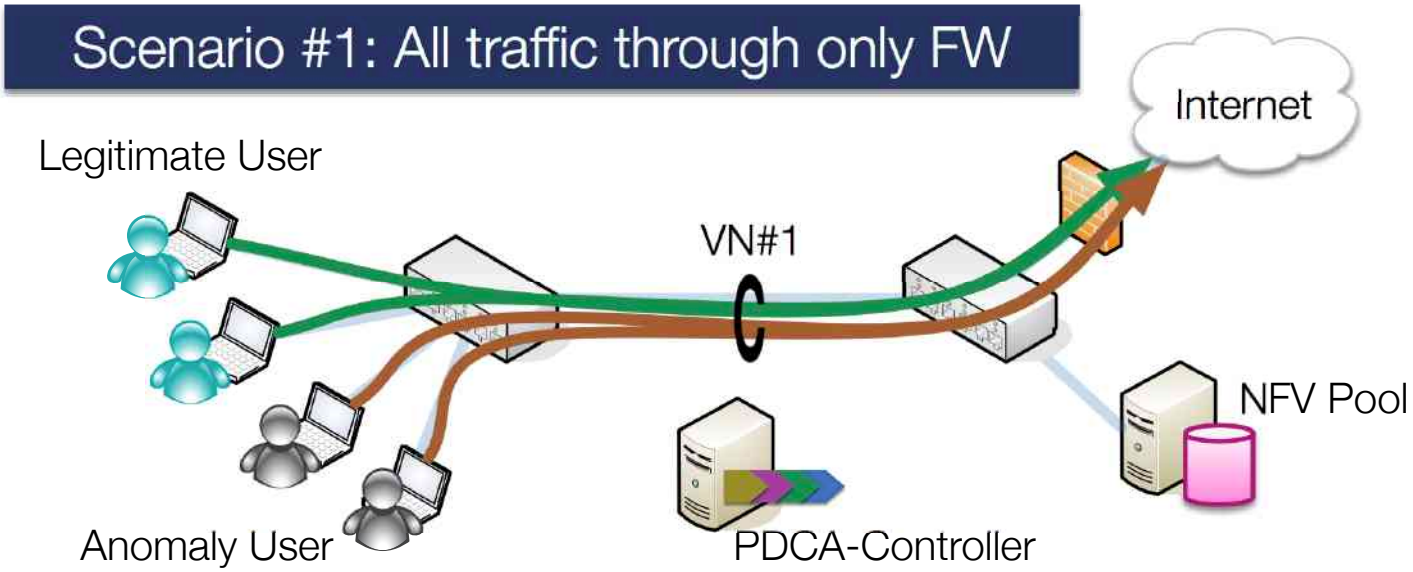
実験概要

- 目的
  - 提案したアーキテクチャがどのように協調動作するかを確認
- 環境
  - OpenFlow スイッチ、 NFV プール、PDCA サイクルのコンポーネントが存在
  - OpenFlow スイッチには合計 20 のユーザが接続
  - ネットワークはファイアウオールを介してインターネットに接続



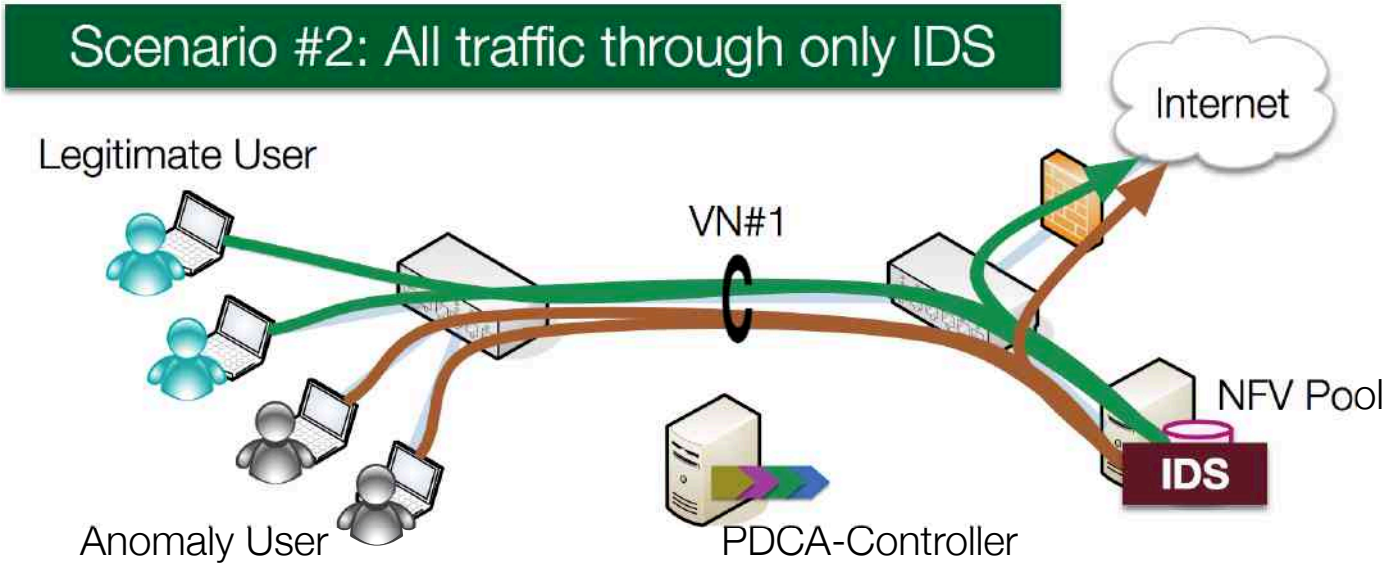
実験シナリオ #1

- 実験開始時 (0 秒)
- 全てのトラヒックはファイアウオールを経由
- 正常コンテンツと異常コンテンツをアップロードするノードが混在（それぞれユーザの半数）
  - アップロードが終わり次第、直ちに再アップロードが行われる



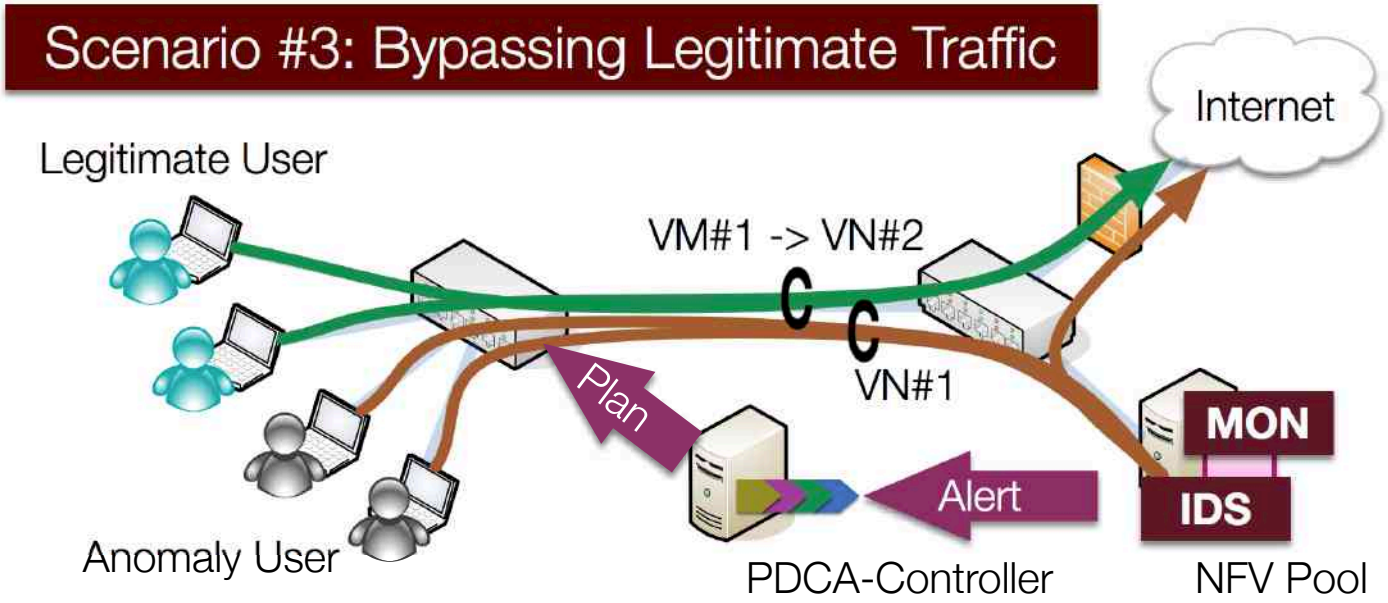
実験シナリオ #2

- シナリオ #1 が始まって *15* 秒後
- 管理者が異常トラヒックの存在に気づく
  - 全トラヒックが IDS を経由するように
    ポリシの設定を更新

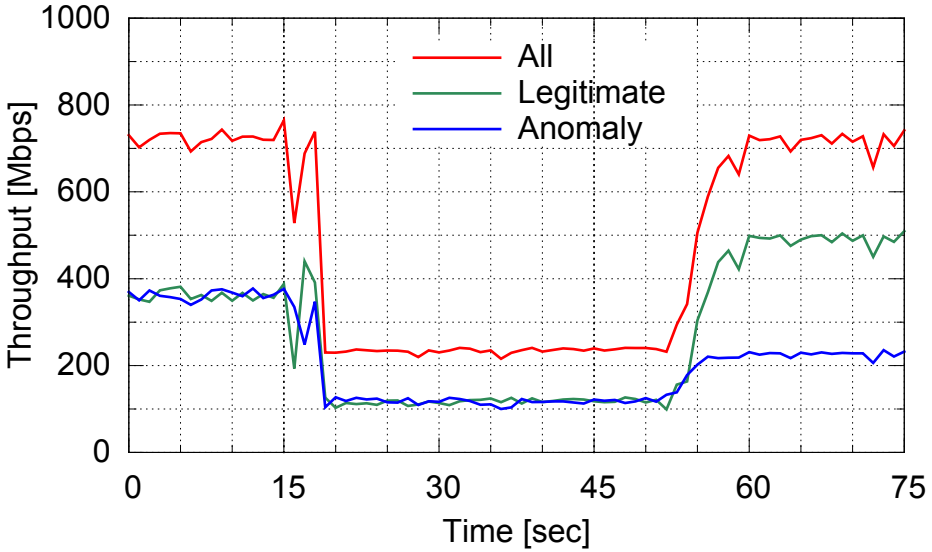


実験シナリオ #3

- シナリオ #2 が始まって *30* 秒後
- IDS を経由することでスループットが低下
  - ユーザからの苦情が発生
- セキュリティを確保しつつ、スループットを向上させたい
  - 異常トラヒックだけ IDS を経由させるようにするようにポリシの設定を更新



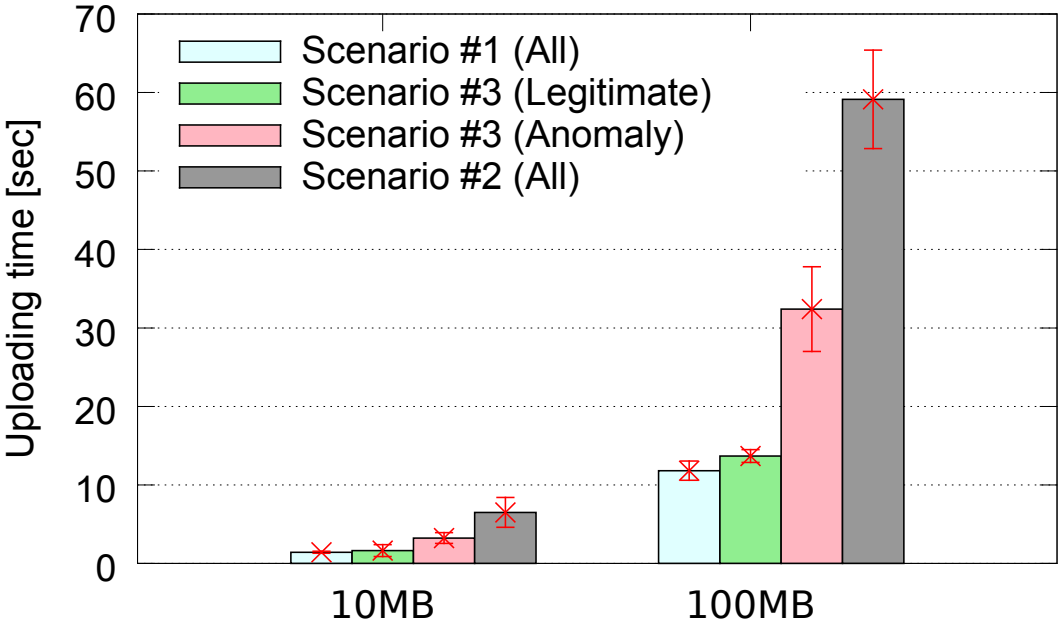
実験結果: トラヒックの変動



- 時間について
  - シナリオ #1 (*0-15* 秒)
  - シナリオ #2 (*15-45* 秒)
  - シナリオ #3 (*45-75* 秒)

- シナリオ #3 で約15秒に渡りトラヒックが徐々に増加した理由
  - 新しいポリシの適応は新しく生成されたフローにのみ行われるから
  - ネットワークの切替に掛かるオーバーヘッドは 平均 163 ミリ秒
- シナリオ #3 で異常トラヒックのスループットが約 250 Mbps に改善した理由
  - 正常トラヒックの負荷がなくなり IDS の計算資源を全て異常トラヒックに割り当てることができたから

実験結果: アップロード時間の比較



- アップロードしたファイルサイズ
  - 10 MB
  - 100 MB

- シナリオ #3 で正常コンテンツだけでなく異常コンテンツのアップロード速度も向上した理由
  - 正常トラヒックの分離により IDS への処理負荷が軽減されたため

まとめ

- 本稿のまとめ
  - SDN でネットワークと仮想的なネットワーク機能のリソースを、共に運用・管理するための自己最適化手法を提案した。
- 今後の取り組み
  - より多くのユースケースについて提案フレームワークを適用し、より複雑なポリシについての実験を行う。
  - スケーラビリティ評価のために、大規模なネットワーク環境で実験を行う。

ありがとうございました