様　式　Ｃ－１９、Ｆ－１９－１、Ｚ－１９　（共通）

科学研究費助成事業　　研究成果報告書

令和　６ 年　 ６ 月 １９ 日現在

機関番号：３４４１２
研究種目：基盤研究(C)（一般）
研究期間：2020～2023
課題番号：２０Ｋ１１８０２

研究課題名（和文）分散型SDNコントローラの統一的制御を可能にするデータベース定義型制御機構の構築

研究課題名（英文）Construction of Database-Defined Management Architecture That Enables Unified Control of Distributed SDN Controllers

研究代表者
佐藤　寧洋（Sato, Yasuhiro）

大阪電気通信大学・情報通信工学部・准教授

研究者番号：８０５７１５５４

交付決定額（研究期間全体）：（直接経費）　　2,600,000円

研究成果の概要（和文）：SDN におけるネットワーク制御情報をデータベースで一元管理することによってコントローラ間の同期問題を解決するとともに、ネットワーク設定状態の可視化や、異なるコントローラが混在した環境であっても同一の制御機構で制御できることを示した。本研究で取り組む予定であったデータベースシステムの構造設計、自律的に動作する機能コンポーネントおよび変換ドライバの開発についてそれぞれ目標通り実現できた。自作エンジンの応用例の一つとして、ユーザの通信挙動に基づく信頼度を指標とした経路制御エンジンを検討し、実運用に向けた具体的な機能コンポーネントについて実装した。

研究成果の学術的意義や社会的意義
年々ネットワークの大規模化が進み、ネットワーク制御も複雑化してきており、SDN などの技術が積極的に利用されている。本研究で実現したデータベース型制御機構は、SDN 制御における分散型コントローラに対する柔軟性や統合的な制御を促進する技術であるといえる。多種多様な SDN コントローラに対して、実装や機器の差異を吸収することができる本提案手法は、運用面の管理・設備コストを低減するだけでなく、ユーザからの細粒度な要求に対して柔軟に制御しうる制御機構を提供しており、その点でも社会的意義は大きいといえる。

研究成果の概要（英文）：We demonstrate that the proposed architecture can achieve unified control even in network environments in which different SDN controllers are mixed by solving the synchronization problem among distributed controllers using a generic database system. The goals of this research, designing the database system structure, constructing the designed databases, developing autonomous operating functional components such as path calculation and resource allocation engines, and developing conversion drivers that convert the intermediate codes into appropriate configurations suitable for each controller system, were achieved as planned.

研究分野：情報ネットワーク

キーワード：SDN　ネットワーク制御　データベースシステム

１．研究開始当初の背景

Internet of Things（IoT）の普及により、インターネット上で扱う情報量や制御の粒度が大きく変化しつつある。現在の IP ネットワークでは、管理者が手動で設計・制御を行っているため、多様化するユーザの要求に応えることが困難なため、柔軟なネットワーク制御が可能である Software-Defined Network （SDN）技術が注目されている。SDN では、データプレーンにあるスイッチを集中管理しているが、ネットワークの大規模化によるスケーラビリティの低下や単一障害点によるネットワーク全体の停止が問題となるため、図 1 に示す分散型コントローラ機構が採用されている。分散型コントローラ機構では、管理ドメイン上に複数のコントローラを配置し、各コントローラが管理するスイッチを分散させている。各スイッチはコントローラから一括制御されており、ネットワークごとのポリシーやネットワーク資源情報にもとづいて経路情報などが更新されている。大規模ネットワークではスイッチに到着するパケット数が増加するため、コントローラへの問い合わせが膨大となり、結果としてスループットが低下してしまう。そのため、図に示すとおり複数のコントローラを配置し、管理するスイッチを分散させることで、コントローラへの負荷を低減させている。しかし、その一方 でネットワーク全体の状態や経路情報などを不整合なく管理するためには、複数の分散コントロー ラ間での情報同期が必要となる。既存研究では、コントローラ間のメッセージのやりとりによって 情報同期を実現しているが、同期する情報の量や頻度と同期にかかるコストはトレードオフの関係 であることからそもそも調整が困難であり、コントローラの実装が複雑化することから、SDN コ ントローラの機能として実装されることが望ましいとはいえない。

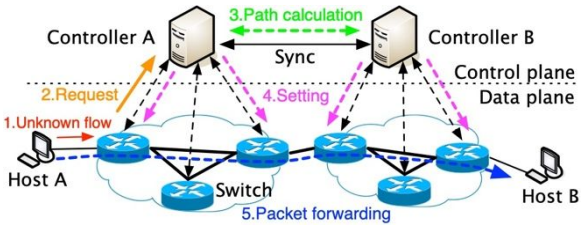

図 1 分散型コントローラによるネットワーク制御

さらに、Network Function Virtualization（NFV）による SDN コントローラの仮想化についても近年検討されており、SDN コントローラ機能を仮想マシン上やコンテナ上にインスタンスとして複数配置し、アプリケーションからの要求に応じて物理ネットワークを制御することが提案されている。データセンタなど高度に仮想化された環境下において、物理ネットワークにおける負荷状況に応じてコントローラ数を増やしたり、仮想ネットワークを新たに作成したりすることで、必要なリソースを必要なだけ割り当て、より効率的で信頼性の高い制御を実現できうることが示されつつある。そのような状況化においても、コントローラ同士の情報共有やネットワーク状態の把握は不可欠であることから、分散型コントローラと同様の問題が発生する。

２．研究の目的

分散型コントローラの統一的制御を実現するデータベース定義型 SDN 制御機構の構築が目的である。ネットワーク制御情報をデータベースにおいて一元管理・定義し、ネットワーク全体に係わる制御情報の一貫性保持などをデータベースシステムが有する機能で提供することによって、分散型コントローラ機構における情報同期問題を解決することを目指す。

図 2 に本研究の概要を示す。既存のコントローラで実行している経路計算やリソース管理などの処理をコントローラから抽象化し、その抽象化した機能群を自律的に動作可能な機能コンポーネント（**エンジン**）として設計・実装する。データベースシステムで管理する情報については、さまざまな SDN コントローラの設定ファイルの基礎となるような汎用的な情報で構成し、各コントローラに応じた**変換ドライバ**を提供することで、異種コントローラが混在しているネットワーク環境であっても、統一的な管理が実現できることを実験環境において検証する。また、本提案手法にもとづいて、各コントローラを仮想化することで、コントローラ自身の負荷分散や応答性能の向上が見込めることを示す。

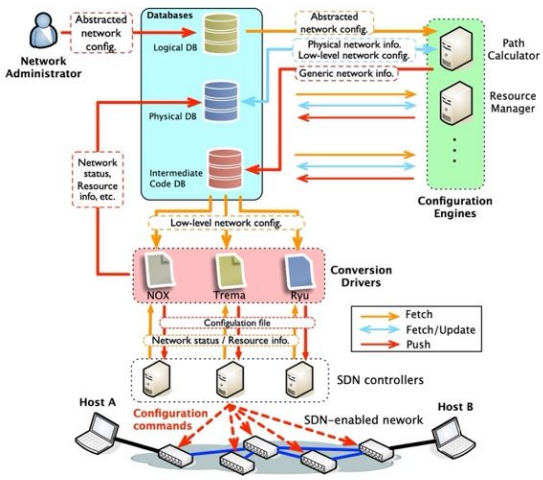

図 2　データベース定義型 SDN 制御機構の概要

３．研究の方法

物理ネットワークの結線構造やリソースの状況、経路情報、管理者が指定するポリシーなどの設定情報などの全ての制御情報をデータベースにおいて定義し、データベース上の情報更新に

よってネットワーク制御を実現する。また、データベース上の設定情報を処理し、経路やリソース計算をするための機能コンポーネントを定義・実装し、各機能コンポーネントを自律的に稼働させる。さらに、機能コンポーネントによって生成された制御情報をもとにさまざまなコントローラに対応した設定情報へと変換する変換ドライバを用意し、異種コントローラが複数混在するネットワーク環境であっても、データベースシステムを基盤とする統一的なネットワーク制御を実現する。上記のことを実現するために本研究では以下の 3 点に取り組んだ。

(1) データベースシステムの構造設計および制御情報の抽象化

ネットワーク制御情報の種類別に、Logical DB、Physical DB、Intermediate Code DB の三つのデータベースを構築する。Logical DB には管理者が物理ネットワーク上に構成する論理的なネットワークの設定情報や各種機能・処理を実装した機能コンポーネント名を格納する。Physical DB では物理ネットワークに存在するノードやインターフェースの情報、リソースの利用状況に関する物理的な接続情報を管理することとし、初期情報として事前にデータベースに登録する。Intermediate Code DB は機能コンポーネント群によって生成された中間コードを管理するためのデータベースである。中間コードとは、特定の SDN コントローラに依存しない汎用性の高い情報とし、本研究では OpenFlow プロトコルを想定した、特定のパケットに対するアクション、VLAN タグ ID などの転送に必要な識別子などで構成する。物理ネットワークからの資源情報について更新があった場合は、該当する機能コンポーネントに対してプッシュ型の通知を行い、即応的なネットワーク制御ができるようにする。

(2) 自律的に動作する機能コンポーネントおよび変換ドライバの開発

機能コンポーネントは、Logical DB と Physical DB に格納されている設定・制御情報をもとに経路などの計算処理を実行し、その結果として中間コードを生成、Intermediate Code DB に格納する。機能コンポーネントは、これまでコントローラ内部において従属的に動作していた処理を抽象化し、自律動作可能なコンポーネントして実装する。各コンポーネントは、原則として一つの機能だけを有することとし、本研究ではダイクストラ法による最短経路を計算するコンポーネントをまず開発する。データベースから経路計算を実施するネットワークの情報を SQL クエリやデータベースプロシージャによって取得し、ダイクストラ法による計算を実行、得られた経路情報から転送リンクごとに割り当てた VLAN ID などを中間コードとして出力させる。加えて、その他のルーティングプロトコルやリソース情報にもとづく帯域制御なども機能コンポーネントとして実装する。変換ドライバは、中間コードをもとにさまざまな OpenFlow コントローラの設定ファイルを生成する。各コントローラに対応した設定ファイルのテンプレートを設計し、そのテンプレートをもとに、パケットに対するアクションや出力先、具体的なインターフェース番号などを中間コードから取得することで、設定ファイルを構成する。REST API に対応した Floodlight や Ryu コントローラでは、変換ドライバが直接 API を利用し、設定を反映させる機能を実装する。

(3) 実環境における動作検証と性能測定

上記の(1)、(2)で設計・開発したデータベースシステムおよび機能コンポーネントを小規模ネットワーク上で動作させ、SDN コントローラおよびスイッチが問題なく稼働することを確認する。さらに、他のコントローラシステムとの性能比較のため、実環境において未知のフローに対するフローエントリの生成時間やスループットなどを測定する。基本的な動作検証の後は、スイッチ数を増加させたり、異種コントローラを導入したりすることで、拡張性に関する評価を実施していくとともに、各コンポーネントを NFV による仮想化インスタンスとして動作させることで、データセンタなど大規模クラウドネットワーク環境を想定した動作検証を実施する。

４．研究成果

(1) データベースシステムの構造設計および制御情報の抽象化

データベースシステムには代表的な関係データベースシステムである、MySQL を採用し、各 DB に必要なテーブルを定義した。Physical DB では、物理ネットワークの接続状態や SDN スイッチの情報を管理することが主となるため、図 3 に示すテーブル構成とした。Interface_Traffic テーブルには、端末の物理的な情報である MAC アドレスや出力ポート番号を格納している。Node テーブルには各端末のホスト名や端末が接続されているスイッチの Datapath ID（スイッチの識別子）を格納している。dpid テーブルには、ネットワーク内にあるスイッチに関する情報を管理しており、スイッチを特定するための Datapath ID などを格納するように構築した。管理者などからのネットワーク設定情報を格納するための Logical DB のテーブル構造を図 4 に示す。管理者からのネットワーク設定にあたっては、Physical DB の情報をもとにネットワーク接続状態を視覚的に表示し、設定しやすくするための Web インターフェースを開発した（図 5）。Web 上で設定された情報を Logical DB に格納す

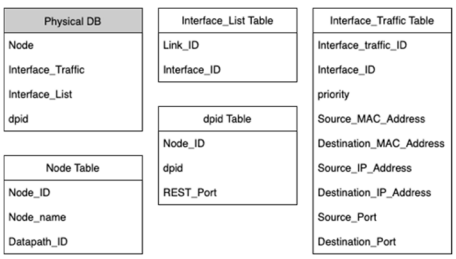

図 3　Physical DB のテーブル構造

ることで、将来的には設定の一部を認証されたユーザに対して公開するなどの機能拡張が考えられる。Physical DB と Logical DB の内容をもとに、必要な経路計算を行い、中間コードを作成するのが次節で説明するエンジンである。中間コードを格納する Intermediate Code DB は OpenFlow で扱うことができる制御情報に準拠し、レイヤ 2 からレイヤ 4 までのヘッダ情報を格納できるように構築した（図 6）。

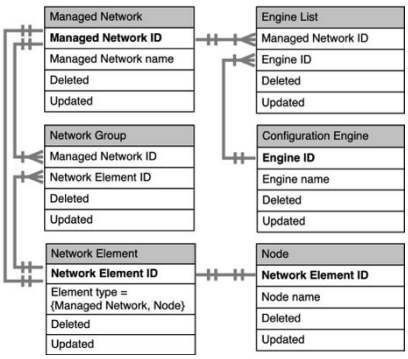

図 4　Logical DB のテーブル構造

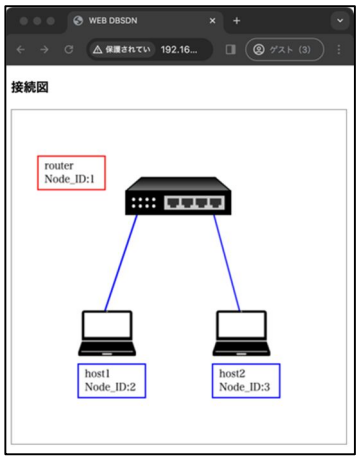

図 5　設定用 Web インターフェース

それぞれのデータベースは MySQL 上に作成し、必要な情報を取得できるようにした。ネットワークの規模が大きくなり、より高速なデータベースアクセスが必要となった場合は、分散データベースや他のデータベースシステムへの移行が考えられるが、制御情報をコントローラから分離しているため、コントローラの複雑性を高めることなく、データベース機能の拡張によって必要な性能を確保できると考えられ、本研究の利点の一つであるといえる。

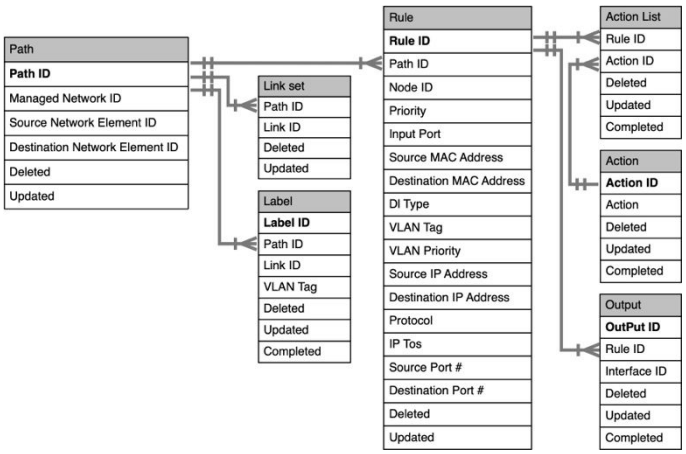

図 6　Intermediate Code DB のテーブル構造

(2) 自律的に動作する機能コンポーネントおよび変換ドライバの開発

本研究における機能コンポーネントを（設定）エンジンと定義し、設定された端末間の経路情報作成などを担当する。経路情報については、まず 2 端末間の疎通性を確保することを目標に、Physical DB と Logical DB から接続されている端末の IP アドレスや MAC アドレス、接続されている SDN スイッチ（コントローラが設定を反映させる必要のあるスイッチ）を取得し、具体的に設定に必要な制御情報を生成し、Intermediate Code DB に格納するエンジンを Python プログラムとして実装した。図 7 に実装したエンジンの実行中の様子を示す。設定に必要な情報を JSON 形式で出力し、Intermediate Code DB へ格納している。また、通信内容をスイッチから定期的に取得し、各ユーザの通信挙動から独自の指標として「信頼度」を定義・計算し、その値に基づいて通信の帯域割り当てを制御する制御エンジンについて検討・実装した。この制御エンジンについては実環境での検証は未完であるため、帯域制御に対応した変換ドライバの実装を進めている。

```
~~~ Connected to database! ~~~
----------------------
Check flow table     :1
Initialize flow table:2
Add flow entry       :3
Enter number     →3
----------------------
Start broadcasting:1
Remove Action      :2
Add flow entry     :3
Enter number     →3
----------------------
Enter the name of the switch to configure:router
Node_ID: 1
Node_name: router
Datapath_ID: 1
Port: 8080
IP_Address: 192.168.45.2
----------------------
Generated URL
http://192.168.45.2:8080/stats/flowentry/add
----------------------
Enter the host name to add the flow entry:host1
Node_ID: 2
Node_name: host1
Datapath_ID: 1
Priority: 10
Source_MAC_Address: 84:47:09:11:64:92
Source_Port: 5
----------------------
JSON output
{"dpid": 1,"priority": 10,"match":{"eth_dst":"84:47:09:11
:64:92"},"actions":[{"type":"OUTPUT","port":5}]}
----------------------
<Response [200]>
----------------------
END
```

図 7　エンジン稼働中の様子

変換ドライバは設定方法が異なるコントローラごとに実装する必要がある。中間コードをスイッチの設定方法に適した形式に変換し、同じ設定情報から異なるコントローラへ同じように設定できるようにした。本研究では Ryu コントローラにおける REST API の構成法について調査し、Intermediate Code DB の内容から適切な設定ができるような URL を構成した。同様に、異なるコントローラ実装である、OpenDaylight、Floodlight、trema についても調査し、各コントローラに適した変換ドライバを実装し、データベースを中心とした基本的な設定が行えることを明らかにした。

(3) 実環境における動作検証と性能測定

上記（1）、（2）の検証用ネットワークとして、図 8 に示す実験環境を構築した。アライドテレシス製 CentreCOM AT-X230-10GT を OpenFlow スイッチとして用い、クライアント端末を想定

した汎用 PC をデータプレーンに接続した。また、コントロールプレーンとして、管理者用端末、OpenFlow コントローラをそれぞれ用意し、本課題におけるデータベースサーバとエンジン、変換ドライバは仮想化サーバ上に構築した。これらの端末には汎用 Linux サーバ OS を利用した。さらに、異なるコントローラによる統一的制御を検証するために、同じネットワーク構成で、コントローラを OpenDaylight、Floodlight とし、OpenFlow スイッチを増設した環境を構築した。異なるスイッチに接続されているクライアント端末間の接続設定について、異なるコントローラであっても、本手法のエンジンおよび変換ドライバによる設定反映によって、統一的な制御が行えることを明らかにした。コントローラおよび本手法の性能評価については、具体的に結果として示すところまで進めることができておらず、現在引き続き評価実験を進めている。

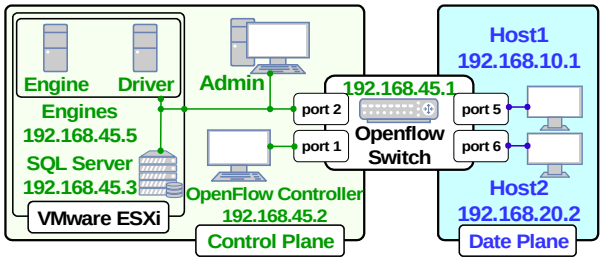

図 8　構築した実験環境

５．主な発表論文等

〔雑誌論文〕　計0件

〔学会発表〕　計10件（うち招待講演　0件／うち国際学会　1件）

１．発表者名 岩本廉磨、佐藤寧洋
２．発表標題 データベース定義型SDNにおけるRyuコントローラを使用したネットワーク制御機能の実装
３．学会等名 FIT2023
４．発表年 2023年

１．発表者名 山本裕也、佐藤寧洋
２．発表標題 ユーザの通信挙動に基づく信頼度計算とその利用法
３．学会等名 2023年電子情報通信学会ソサイエティ大会
４．発表年 2023年

１．発表者名 Renma Iwamoto、Yasuhiro Sato
２．発表標題 Design of the Configuration Engines with Ryu REST API in Database-oriented SDN Architecture
３．学会等名 12th International Conference on Networks, Communication and Computing（国際学会）
４．発表年 2023年

１．発表者名 岩本廉磨、佐藤寧洋
２．発表標題 データベース定義型SDNにおけるWEBアプリケーションを使用した接続図表示機能の実装
３．学会等名 2024年電子情報通信学会総合大会
４．発表年 2024年

１．発表者名
山本裕也、佐藤寧洋

２．発表標題
フロー情報に基づく端末信頼度の計算とその検証

３．学会等名
2024年電子情報通信学会総合大会

４．発表年
2024年

１．発表者名
坂下峻昭、佐藤寧洋

２．発表標題
OpenFlowを利用したクライアントネットワークの切替手法

３．学会等名
2022年電子情報通信学会ソサイエティ大会

４．発表年
2022年

１．発表者名
坂下峻昭、佐藤寧洋

２．発表標題
VLAN を利用したクライアントネットワークの切替手法

３．学会等名
2021年電子情報通信学会ソサイエティ大会

４．発表年
2021年

１．発表者名
田中智也、佐藤寧洋

２．発表標題
データベース定義型 SDN におけるGUIベース制御機能の実装

３．学会等名
電子情報通信学会ソサイエティ大会

４．発表年
2021年

１．発表者名 田中智也、佐藤寧洋
２．発表標題 データベース定義型SDNにおける視覚的なネットワーク接続制御方法
３．学会等名 電子情報通信学会総合大会
４．発表年 2022年

１．発表者名 田中智也、佐藤寧洋
２．発表標題 データベース定義型SDN制御におけるネットワーク情報の可視化機能の実装
３．学会等名 電子情報通信学会総合大会
４．発表年 2021年

〔図書〕　計0件

〔産業財産権〕

〔その他〕

―

６．研究組織

	氏名 （ローマ字氏名） （研究者番号）	所属研究機関・部局・職 （機関番号）	備考

７．科研費を使用して開催した国際研究集会

〔国際研究集会〕　計0件

８．本研究に関連して実施した国際共同研究の実施状況

共同研究相手国	相手方研究機関