Vol.2017-IOT-38 No.8
2017/6/24

広域ネットワーク情報共有基盤KANVASにおけるBGP情報収集・蓄積機構

島松健太[†2,a)]　近藤賢郎[†2,b)]　金子晋丈[,c)]　寺岡文男[,d)]

概要：KANVAS は当研究室で研究・開発を進めているネットワークの統合的な管理や情報共有を行うためのフレームワークであり，ネットワーク情報の収集の統一化と共に広域ネットワーク管理や障害検知に利用することを目指している．KANVAS はコレクタ，ストレージサーバ，アクセスサーバの 3 つから構成されており，それぞれがネットワーク情報の収集・蓄積・提供を担当するモジュールである．本稿は KANVAS における BGP の経路情報の収集・格納の機構を提案するものであり，BGP 関してこれらのモジュールの設計・実装を行う．また，単に収集された BGP 経路情報を蓄積するだけではなく，グラフデータベースにおける検索を効率良く行うために，BGP 経路情報の構造化やプレフィックス最長一致のためのキャッシュ構造を提案する．KANVAS は起動時に BGP ルータから収集したフルルート 62 万経路を格納する必要があり，これには約 2 時間半かかるが，その後の経路の更新にかかる時間は十分に小さく実用的である．

Shimamatsu Kenta[†2,a)]　Kondo Takao[†2,b)]　Kakenko Kunitake[,c)]　Teraoka Fumio[,d)]

1. 背景

インターネットは別々の経路制御ポリシーを持ったネットワークの集合，すなわち個別の組織が管理・運用している独立したネットワーク (AS) のネットワークであり，中央制御の仕組みを持たない．これはインターネットの堅牢性や公平性において，重要な性質であり，インターネットの発展に大きく貢献した．

それぞれのネットワークの内部構造や経路制御ポリシーが全く異なるにも関わらず，インターネット全体としての経路制御を行なえるのは，BGP と呼ばれる経路情報を交換するための単一標準のプロトコルが存在しているためである．BGP によって経路情報の交換とネットワーク間の大雑把な経路制御を行えば，内部の細かい経路制御に関しては個別のネットワークが独自の経路制御ポリシーに従って行うため，原理的には，すべての IP アドレスに対してパケットを送信することが可能となる．

しかし，インターネットの巨大化やトラフィックの増加に伴い，各々のネットワークもまた巨大化・複雑化しており，管理・運用が難しくなっている．同時に，ネットワークは常に外部の攻撃や機器の故障，設定ミスなどによる通信障害の危険に晒されており，有事の際には，可能な限り早く異常検知及び原因特定を行い，問題を解決することが求められる．このような背景から，多くのネットワーク管理システムやそのためのプロトコルが生み出されてきた．

近年では，SDN (Software Defined Network) による柔軟な広域ネットワークの制御やプレフィックスハイジャックなどのローカルネットワークからは観測できない攻撃への対処などの観点からネットワーク間での情報共有の必要性が高まりつつある．しかしながら，ネットワーク内部の情報はセキュリティの観点などから一般的に公開されておらず，また，別々のネットワーク管理システムの間で情報を共有する仕組みも確立されていないため，ネットワーク間での連携が難しい状況にある．

これらの背景を踏まえ，当研究室では広域ネットワーク情報共有基盤 KANVAS (Knowledge base system in wide Area Networks with Versatility,Availability and Scalability)[1][2][3] の研究・開発を行っている．本稿では KANVAS における BGP 経路情報の収集・蓄積に関するものであり，

†1 現在，慶應義塾大学理工学部
Presently with Faculty of Science and Technology, Keio University
†2 現在，慶應義塾大学大学院理工学研究科
Presently with Graduate School of Science and Technology, Keio University
a) penta@inl.ics.keio.ac.jp
b) latte@inl.ics.keio.ac.jp
c) kaneko@ics.keio.ac.jp
d) tera@ics.keio.ac.jp

経路情報の収集を行うコレクタ，収集された情報を蓄積するストレージサーバ，収集された情報に効率よく利用するためのアクセスサーバの３つの設計・実装を行う．また，収集された BGP 経路情報を単に蓄積するだけではなく，効率の良い検索を行うための経路情報の構造化やアプリケーションから IP アドレスによるクエリを受けた際に必要となるプレフィックスの最長マッチのためのキャッシュ構造を提案する．

2. 動機

KANVAS は従来のネットワーク管理システム (NMS) が個別に行っていたネットワーク情報の収集・蓄積を一括して行い，KANVAS を導入したネットワーク同士で情報を共有することにより，NMS を含めたアプリケーションに対してより広域で網羅的なネットワーク情報を提供する．以下では，KANVAS を提案するに至った動機について述べる．

2.1 収集・蓄積の統一

NMS は，その目的に応じて必要なネットワーク情報を収集して，わかりやすい形でユーザに提供するものであり，サーバの死活監視やネットワークの障害検知などの目的で利用される．収集すべきネットワーク情報は NMS 毎に異なり，それぞれの NMS は独自の方法でネットワーク情報を収集し，互換性のないフォーマットでローカルストレージへ保存する．これらの情報はその MNS 内のみでの利用を想定しており，別のアプリケーションで再利用したり，情報を互いに交換したりすることは考慮されていない．KANVAS では従来 NMS が個別に行っていたネットワーク情報の収集を肩代わりし，収集されたネットワーク情報を RDF を用いて明文化された形式で保存する．KANVAS を利用するアプリケーションは REST API や RDF のクエリ言語を使って蓄積された情報を利用したり，逆に計算によって得られた情報を格納することにより，他のアプリケーションとの連携が可能になる．また，NMS ごとに重複したネットワーク情報を収集する必要がなくなり，実装上・実用上の無駄を削減することも可能である．一般的に NMS は目的にかかわらず，何かしらのネットワーク情報のモニタリングを行うが，KANVAS ではこのような常時最新の情報を取得し続ける必要のあるアプリケーションのために Publisher・Subscriber の機能を提供する．

2.2 障害検知と原因推測

KANVAS では単にネットワークの情報を収集するだけではなく，それらの情報からネットワークの障害を検知し，その原因を推測する研究を行っている．障害検知に関しては，Zabbix[4] や Nagios[5] のような特定の問題を検知する NMS はすでに存在するが，原因推測に関しては難しく，あまりよく実装されていないのが現状である．理由として，障害原因を推測するためにはあらゆるネットワーク情報を収集し，様々な角度からそれらの情報を吟味しなくてはならないが，通常の NMS において，十分な量の情報を収集するのは無駄が多く，また推測自体が多くの推論規則を必要とするために一般的に困難である．多くの場合はターゲットとなる問題を限定し，その問題の発生を検知するのに必要最小限の情報のみを収集して，利用するに留まっている．KANVAS では，すべてのネットワーク情報を収集・蓄積・提供すること自体が一つの目的であるため，障害原因の推測に必要な情報を無理なく収集することが可能であり，また，同時に推測によって得られた情報をユーザに提供するだけでなく，他の NMS から利用することにより，より高度に自動化されたネットワーク管理を実現することができると考えられる．

2.3 情報共有と広域ネットワーク制御

AS はその内部情報をほとんど公開することなくインターネットに参加することできるため，AS 間での情報共有は BGP によるものを除けば，殆ど行われていないのが実情である．従来，それぞれのネットワークは送られてきたパケットを次の AS 中継するまでの間のみ責任を持てばよく，すなわち自身の AS 内でのみ障害が起きていないことを保証すれば十分であった．しかし，現代で必要不可欠な社会基盤となったインターネットにおいては，単なる疎通性の確保以上のことが求められるようになっている．具体例としては，動画，音楽，画像などのマルチメディアコンテンツが増加により，トラフィックが増大し，状況に応じて適切に経路を選択することで負荷分散を行う必要が出てきたことである．また，別の例として，交通機関や金融機関などの会社で利用されるミッションクリティカルな通信に対しては帯域・遅延・スループットについての信頼性と耐障害性を保証することが必要とされることが挙げられる．これらの要求は単一の AS 内に限れば可能であるが，複数の AS を中継する場合においては，他の AS がブラックボックス化されており内部状態が一切わからない以上，不可能である．

KANVAS では，それぞれの AS 間である程度ネットワーク情報を交換することにより，インターネット全体の構造や状態を把握することが可能となり，より柔軟なネットワーク制御を行うことができるようになる．例えば，別の AS で障害が発生し，自身の AS において特定の IP アドレスに対して通信が行えなくなった際にも，その障害を検知して自身のネットワーク側で，別の経路に切り替えて障害を回避することができる．また，エンド間の通信においては，複数の経路や任意の経路を利用する研究 [6][7] が存在するが，宛先 IP アドレスに対する経路はインターネット側，すなわち，BGP やそれぞれの AS の経路制御ポリシー

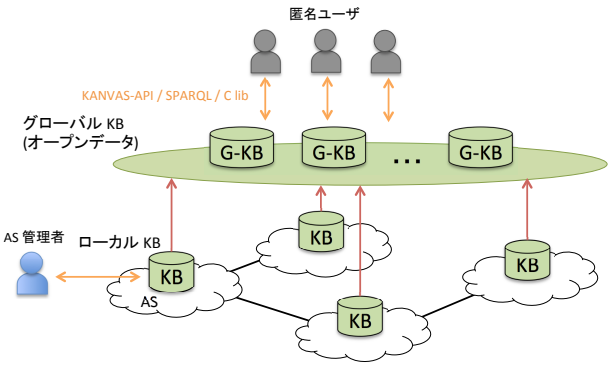

図 1 インターネット規模で見た KANVAS の外観

が決定するため，こういった研究では中継ノード (異なる IP アドレス) を利用することによって擬似的にユーザー側で経路の選択を可能にしている．この時，どこに中継ノードを設けるのが最も適切であるかを知ることが重要であるが，こういった場合にも KANVAS の利用が見込まれる．

2.4 インターネット計測

インターネット計測ではインターネット上の様々な場所に traceroute を発行するノードや BGP コレクタを設置する必要があり，計測の精度はこれらの観測点の分布と数に依存することになる．KANVAS はこのようなインターネット計測における観測点としての役割も果たせるので，より高精度な計測に貢献することが期待できる．

3. KANVAS 概要

KANVAS は自身の AS からネットワーク情報を収集すると同時に，KANVAS を導入した別の組織と情報を共有することで，インターネット規模でのネットワーク知識の活用を目指したフレームワークである．KANVAS を導入するユーザとしては，主にバックボーンネットワークを運用する ISP や学術組織を想定しているが，一般のユーザもオープンデータとして公開されているネットワーク情報にアクセスすることが可能である．

図 1 はインターネット規模で見た KANVAS の外観を示したものである．KANVAS においてネットワーク情報を収取・蓄積して利用可能な状態にする仕組みをナレッジベースシステム (KB) と呼び，同一ポリシで運用されるネットワーク (AS) 毎に設置され，互いにインターネットを介して相互に接続される．

KB は蓄積する情報の性質によって 2 つに分類される．1 つはローカルな KB であり，すべての AS に 1 つづつ配置され，AS 内で収集された情報がすべて保存される KB である．2 つはグローバルな KB であり，ローカルな KB に蓄積された情報のうち，一般公開が可能なデータ (オープンデータ) のみが蓄積される KB である．ローカル KB の情報は管理者のみがアクセス可能であるのに対して，グローバル KB の情報は誰でも自由に利用することができる．AS 内で収集された情報はまず始めにローカル KB に保存され，それらの情報のうち匿名性が高いと判断された部分のみがグローバル KB に送信され，一般公開される．グローバル KB は分散システムであり，要所々々に設置され，互いに情報を共有し同期することで一貫性を保持する．

KANVAS はトポロジやリンク，経路，デバイス，フローなどのあらゆるネットワーク情報を収集するが，これらは主に，経路制御プロトコルやデバイス管理プロトコルから収集される．例えば，経路制御プロトコルの具体例としては，AS 内の経路制御を行う OSPF や AS 間の経路制御を行う BGP などがあり，デバイス管理プロトコルとしては SNMP や NETCONF などがある．

ネットワーク情報の保存形式は KANVAS が提供するネットワークオントロジ Bonsai によって定義される．オントロジとはある領域内の概念の関係性を形式的に表現したものであり，したがって，Bonsai とはネットワークという領域において，概念の関係性を定義したものである．Bonsai の表現にはリソース記述言語 RDF (Resource Description Framework) を用いる．RDF は元々はウェブ上にあるリソースを記述するためのフレームワークであり，リソースの関係性を主語，述語，目的語の組の集合によって記述する．例えば，“ルータはインターフェースを持つ” という表現は，主語の “ルータ” と目的語の “インターフェース” というリソースを “持つ” という述語のよって接続したグラフとして表現することが可能であり，このようにネットワークに関するあらゆる物事の関係性を逐一記述したものが Bonsai である．RDF の構成要素はシンプルであるが，グラフ，リスト，テーブルなどのあらゆるデータ構造を記述することが可能である．Bonsai の具体例として，図 2 に Bonsai の概略を示す．

ユーザが KB に蓄積された情報にアクセスするには KANVAS のライブラリが提供する REST API を利用するか，もしくは RDF ストレージに対して RDF クエリ言語 SPARQL を用いて問い合わせを行うことで可能である．

3.1 KANVAS アーキテクチャ

図 3 に示すように，KANVAS は大きく分けてコレクタ，ストレージサーバ，アクセスサーバの 3 つから構成される．コレクタはデータの収集を行うモジュールであり，基本的に収集対象となるプロトコル毎に別々のコレクタが存在する．例えば，経路情報の収集対象である OSPF や BGP では，それぞれ OSPF 用と BGP 用の別々のコレクタから情報が収集される．ストレージサーバはコレクタによって収集されたネットワーク情報を Bonsai 定義に従いに RDF ストレージに保存するモジュールであり，一般公開可能なものはグローバルなストレージサーバに送信される．アクセスサーバはアプリケーションから API を呼び出した際に

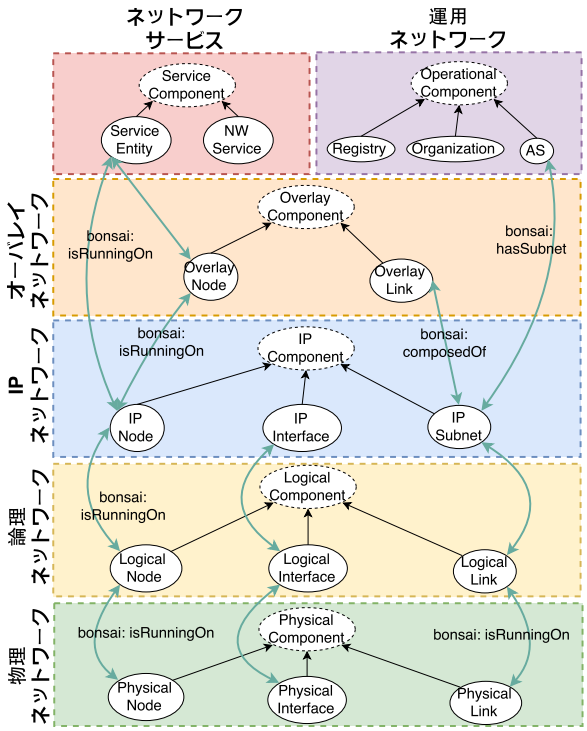

図 2 ネットワークオントロジ Bonsai

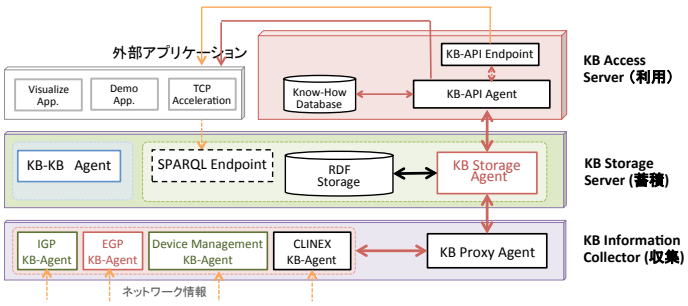

図 3 KANVAS モジュール図

直接 HTTP リクエストが送信されるサーバであり，REST によるクエリをストレージサーバが直接解釈可能な形式に変換する作業，およびそのために必要なキャッシュの生成等を行うサーバである．AS 内の典型的な KB の構成としては，ストレージサーバが 1 つ存在し，複数のコレクタと単一のアクセスサーバがストレージサーバに対して接続する形となる．これらのモジュールは実行時の実態としてはそれぞれ単一のプロセスであり，プロセス間の通信は KANVAS 独自のプロトコルを利用して行う．KANVAS がこのように収集・蓄積・提供でモジュール分割されているのは，主に機能の拡張性とスケーラビリティのためである．

4. 設計

本節では KANAVAS における BGP 情報の収集と蓄積に関する設計を行う．

4.1 構成要素

主な構成要素は以下の 4 つである．

- BGP 経路情報の収集を行う “BGP コレクタ”
- 情報を KANVAS のデータフォーマットに変換して送信する “BGP KB-Agent”
- 情報を受け取って格納する “KB ストレージサーバ”
- ストレージサーバに効率よくアクセスを行うためのキャッシュを保持する “KB アクセスサーバ”

4.2 BGP 情報の収集

BGP ルータから経路情報を収集する際に一般的に用いられる方法はいくつかある．いずれの方法にも共通する点は，BGP コレクタと呼ばれる情報収集専用の BGP ルータを設置して，実稼働する BGP ルータとピアリングすることで経路情報を収集することである．BGP コレクタが収集した情報をどのような形で出力するかによって 3 通りに分類される．

最も一般的な方法は，ソフトウェアルータ等に搭載されている RIB の情報をファイルにダンプする機能を利用する方法である．BGP 等のプロトコルに対応しているソフトウェアルータとして Quagga が挙げられるが，Quagga では BGP の RIB 情報を MRT 形式でファイルに書き出す機能があり，一定時間ごとに経路情報の完全なスナップショット (RIB) をダンプする事ができる．また，より短い時間間隔で差分 (UPDATE) のみを書き出す機能を備えており，ファイルサイズを抑えつつ，更新を出力することが可能である．この方法の利点は手軽であることであるが，欠点としてファイルへの書き出しを一定時間おきにしか行えないのでリアルタイム性に欠けるという点がある．世界各地の BGP ルータから経路情報を収集して，ウェブで一般公開している Route Views?では，この方法を用いている．

別の方法としては，オープンソースのソフトウェアルータを改造する方法があり，1 つ目の方法と比較して，新しい経路情報をすぐに出力することができるという利点があるが，Quagga のようなコード数が多いソフトウェアを書き換えるのは高コストである．

新しい方法としては，BGP Monitoring Protocol (BMP) を利用する方法がある．BMP は BGP の経路情報を TCP で送信するためのプロトコルであり，2016 年 6 月に RFC により標準化された．この方法のメリットは，大容量のローカルストレージを必要とせず，実稼働する BGP ルータからリアルタイムに BGP 情報を収集できることである．商用のルータでも BMP に対応したルータは増えつつあり，今後はこの方法が最も標準的な BGP 情報の収集方法になると考えられる．また，BMP に対応したソフトウェアルータとしては goBGP[8] があり，BGP コレクタとしては最有力候補である．

4.2.1 BGP コレクタ

BGP コレクタは実稼働する BGP ルータから経路情報の受信のみを行う情報収集用の BGP ルータであり，収集対象となる BGP ルータに対して通常の BGP ルータと同様に BGP セッションを確立することで経路情報の取得を行う．BGP コレクタからの情報の出力は前節で紹介した MRT 形式でのファイルダンプを利用する．BGP コレクタは一定時間おきに，その時刻における経路情報の完全なスナップショット (RIB) を RMT フォーマットでファイルに出力し，またそれより短い時間周期で差分 (UPDATE) のみを出力する．RIB と UPDATE のファイル出力の周期は一般的にそれぞれ，30 分，5 分程度であるが，よりリアルタイムなモニタリングを行う必要がある場合であれば UPDATE の出力周期を 1 分にすることもできる．この方法の利点と欠点は既に述べたとおり，手軽な代わりにリアルタイム性に欠けることである．

4.2.2 BGP KB-Agent

BGP KB-Agent は BGP コレクタが一定時間おきに出力する MRT 形式のダンプファイルを読み込み KANVAS 独自のデータフォーマットに変換した後，KB ストレージサーバに送信する．このモジュールは基本的には読み込んだデータを一方的に送信する役割であり，一部制御用のコマンドを除けば，データの受信は行わない．また，BGP コレクタがファイルダンプを生成するタイミングを BGP KB-Agent は把握していないため，最新のファイルを読み込むために一定時間おきにダンプファイルのポーリングを行う．

4.2.3 KB ストレージサーバ

KB ストレージサーバの役割は主に KB-Agent から送られてきたデータの蓄積と KB アクセスサーバから送られてきたクエリへの応答である．KB ストレージサーバは KB-Agent から送られてきたデータを受け取り，ネットワークオントロジ Bonsai の定義に従って情報をローカルストレージへ格納する．この時，ストレージには未加工の BGP 経路情報，AS パスから生成された AS トポロジ，構造化された BGP 経路情報の 3 種類を格納する．BGP 経路情報を構造化するのはアプリケーションからクエリがあった際の検索を高速化するためである．

4.2.4 KB アクセスサーバ

KB アクセスサーバではストレージサーバに格納されたプレフィックスを記録し，アプリケーション側から IP アドレスによる問い合わせがあった際に，プレフィックスの最長一致を行うことで対応する経路情報のプレフィックスを求める．ストレージサーバ側は情報を取得するために厳密に一致するプレフィックスを必要としているが，アプリケーション側はストレージ内にどのようなプレフィックスが格納されているか把握していない．そして，一般的にアプリケーション側は IP アドレスにより経路情報のクエリを行う事が多いと考えられるため，効率よく対応する経路情報を取り出すために，このようなプレフィックスの最長一致を行う仕組みが必要となる．

表 1 BGP 経路情報の具体例

経路	プレフィックス	AS パス
R1	p1	AS1 AS2 AS3 AS4
R2	p1	AS1 AS2 AS4
R3	p2	AS1 AS2 AS3 AS5
R4	p3	AS1 AS6 AS5

4.3 AS トポロジの生成と BGP 経路情報の構造化

図 4 は表 1 に示すような経路情報を構造化したものである．例えば AS トポロジのリンクは AS パスが R1 のように “AS1 AS2 AS3 AS4” であった場合，単純に AS パスを分解することで AS リンク (AS1, AS2), (AS2, AS3), (AS3, AS4) が得られる．R2, R3, R4 についても同様の操作を行うと図の下部に示すような AS トポロジが得られる．上部の AS パス木は観測 AS である 1 を頂点にして AS パスの共通経路を束ねて木構造にしたものである．AS トポロジの AS はその AS が所有するプレフィックスにリンクしており，プレフィックスからは更にそのプリフィックスの経路情報 (エントリ) にリンクしている．また，経路エントリは上部の AS パス及び図中では省略されているが，next_hop, local_pref などの詳細情報とリンクしている．

このような構造を利用することで，特定の条件による検索を高速に行うことができるようになる．例えば図 4 において，AS1 AS2 を経由する経路情報をすべて取得したいとした場合，AS パス木の AS1, AS2 と辿ってその下に続く経路がすべて条件に合致する経路となる．このような検索を行う際に，経路情報が構造化されていなければすべての経路情報を一旦取得し，条件に合致するかどうかを調べなければならず，非常に効率が悪くなる．

4.4 最長プレフィックスマッチのためのキャッシュ構造

KANVAS のような経路情報データベースにおいてプレフィックスの最長マッチは重要な機能の一つである．KANVAS の設計理念によれば永続化が必要な情報は RDF で表現することが望ましいが，それは最長プレフィックスマッチに必要な構造をグラフで表すということであり，不可能ではないものの非効率的である．グラフによる実装にはトライ木を用いたものなどが存在するが，このような参照を使って上下のノードをつなげる方法はメモリを多く消費することに加え，最長マッチを行う際には木の頂点から葉の方に向かって参照をたどる必要あるため，メモリ的にも時間的にも高コストである．そこで，本研究では IPv4 において，RDF ストレージ (グラフ) の代わりにメモリ上で高速かつ省メモリで最長プレフィックスマッチを行うた

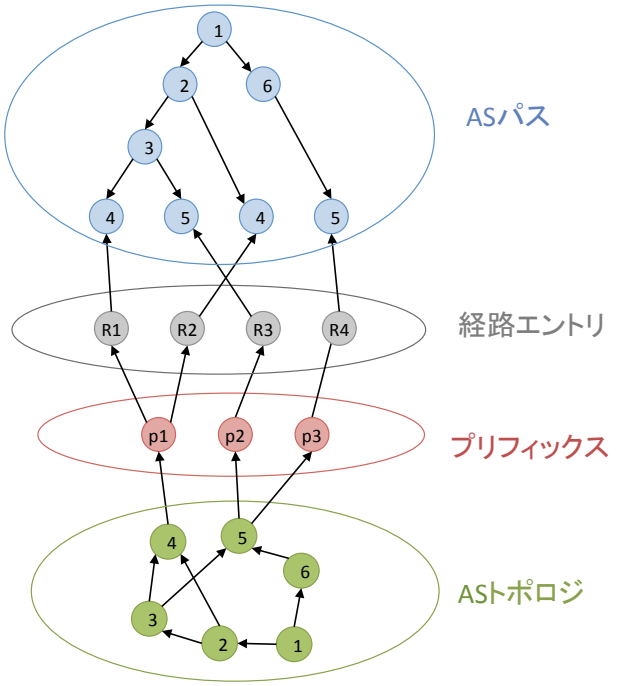

図 4 構造化された BGP 経路情報

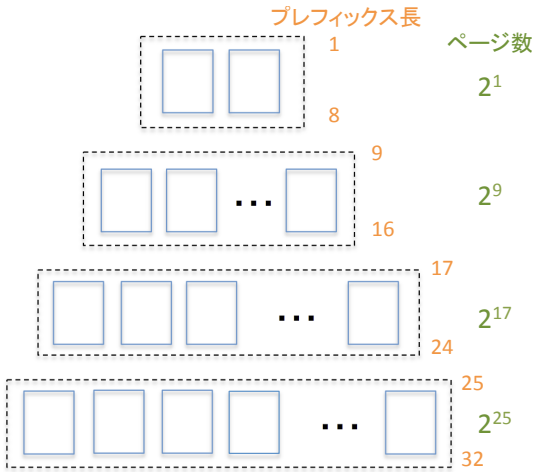

図 5 プレフィックス木 (IPv4)

めのデータ構造を提案する．

論理的な構造は二分木であり，頂点をマスク長が 0 であるプリフィックスとして下に行くほどマスク長が 1 づつ増え，木の枝はそれぞれプレフィックスの 1 ビット分の 0 と 1 に対応する．それぞれのノードは真偽値を持ち，プレフィックスが存在していれば真，存在していなければ偽となる．したがって，この構造は高さが 33 でノードが真偽値を取る完全二分木となる．図 5 はこの構造を高さ 8 毎に 4 つに分割し，さらに図 6 のような高さ 8 の完全二分木の配列として表したものである．この高さ 8 の完全二分木をページと呼ぶことにする．1 ページには 255 個のノードが含まれ，それぞれの値は真偽値 (=1bit) であるから，1 ページは 32byte (256bit) で表すことが可能である．

この構造は以下のような特徴があげられる．

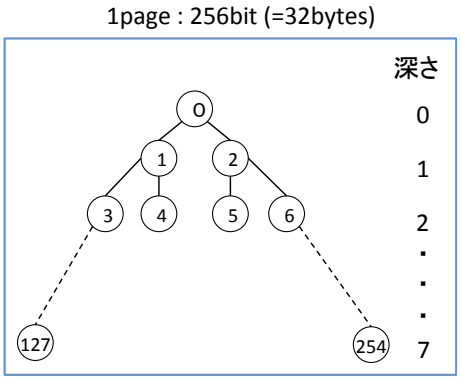

図 6 プレフィックス木のページ

- ページ内のノードの値が全て偽であればページを割り当てる必要がないため，実際にメモリ上に割り当てられるページは少ない．
- 子要素を表現するのに参照 (ポインタ) を使う必要がない．
- あるプレフィックスに対するページの位置とページ内のノードの位置が計算により求まるため，頂点から枝をたどる必要がない．
- 最長プレフィックスマッチは与えられたプレフィックスに対応する位置から上に遡り，最初に見つかったプレフィックスが最長プレフィックスである．
- 現実にアナウンスされるプレフィックスは殆どがマスク長 24 以下であり，そのためプレフィックス木の最下段はほとんど割り当てられることがない．

これらの特徴により，省メモリかつ高速に最長プレフィックスマッチが可能となる．

例えば，ある IP アドレスで最長一致を行い，対応するプレフィックス (マスク長 16) を知りたいとする，この時，IP アドレスから対応する要素の位置（ページ番号，ノード番号）を計算し，その要素から真である要素が見つかるまで親ノードを遡る．この場合，マスク長 25〜32 と 17〜24 のページはそもそも割り当てられていないため，要素の値を確認するまでもなくスキップされ，次の親ページのマスク長 16 で最長一致が完了する．このように割り当てられていないページを飛ばすことで効率よく最長一致が行える．

5. 実装

BGP の経路情報を収集して RDF ストレージに格納するまでの一連のモジュールのプロトタイプを Linux 上で実装した．実装言語は BGP KB-Agent が C++でその他は Java を用いた．BGP KB-Agent における BGP ダンプファイルの読み込みには libbgpdump[9] を使用した．BGP 経路情報と AS トポロジについては RDF フレームワークである RDF4J[10] を利用して格納を行ったが，構造化した BGP 経路情報ついては RDF による表現が煩雑であったことや可視化が難しかったことから，RDF ライブラリの代わ

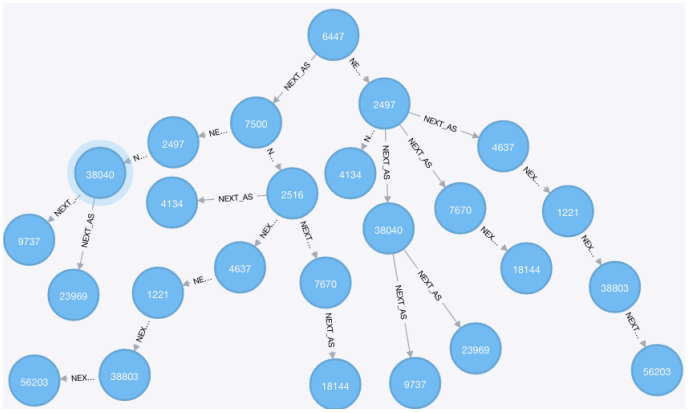

図 7 AS パスの構造化

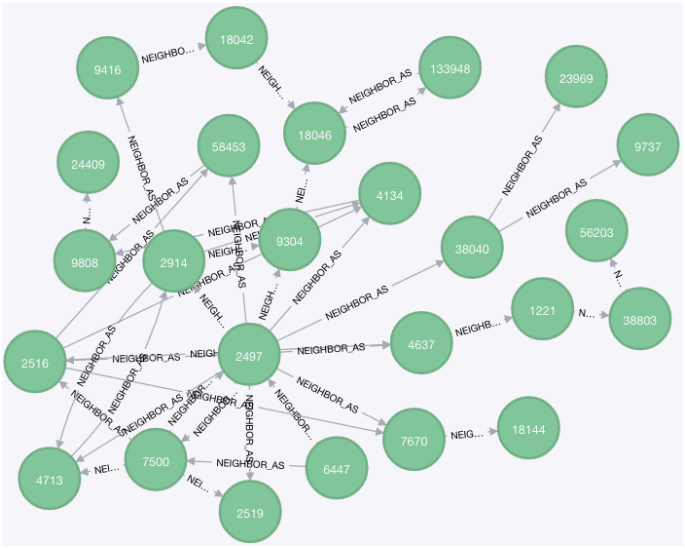

図 8 AS トポロジ

りに通常のグラフライブラリである Neo4j[11] を利用した．BGP KB-Agent はダンプファイルから読み込んだ BGP 経路情報を KB プロトコルの定義に従いシリアライズした後に，TCP でストレージサーバへ送信する．KB プロトコルの定義については省略する．ストレージサーバは BGP 経路情報を受け取ると，その経路情報をそのまま RDF ストレージに格納すると共に，AS パスを分解することにより AS の隣接情報を生成して AS トポロジと構造化された AS パスとして適宜格納する．ストレージへの格納は見た目上は 1 経路毎に行っているが，実装上は 1000 経路毎にディスクへの書き込みを行う．つまり，それぞれの経路の格納の段階では実際にはディスクには書き込まれずにメモリ上に存在しており，1000 経路分格納し終わった時点でコミットして，実際にディスクへと書き込まれる．一般的にグラフライブラリはこのようなコミットロールバックの機能を提供しており，トランザクションの間にエラーが発生してデータの整合性が保てなくなった場合に，書き込みを取り消す事ができるようになっている．また，RDF4J においてディスクへの書き込み準備はオーバーヘッドが非常に大きく，書き込みを頻繁に行うとパフォーマンスが大幅に低下することを確認している．本機構において，1000 経路毎に書き込みを行うのは主に後者の理由である．

5.1 AS パス木と AS トポロジの生成

図 7 と図 8 はそれぞれ生成した AS パス木と AS トポロジの一部を neo4j のウェブインターフェースで表示したものである．AS トポロジは AS の接続関係を表したものであり，隣接する AS は図 8 からも分かるように，NEIGHBOR_AS という述語によって接続される．接続の向きは基本的に片方向であり，両方向の経路が AS パス内にある場合にのみ双方向に接続されている．これは単純に実装の問題である．

図 7 は AS パスを構造化して観測 AS を頂点とする木構造にしたものであり，AS パスは NEXT_AS という述語で連結することで表現される．AS パス内の AS は同じ AS であっても経路が違えば別物として扱われるため，同じ AS 番号が複数存在しうる．

これらの図では AS パスと AS トポロジの接続関係が示されていないが，実際には図 4 に示したように経路エントリとプレフィックスを跨いで相互に接続されている．

5.2 プレフィックス木

各要素 (プレフィックス) が存在するかどうかは boolean で表現できるため，プレフィックス木の 1 ページは 256 ビットの BitSet で表現可能である．プレフィックス木ははじめすべての要素が偽，すなわちプレフィックスが一切存在していない状態から始まる．プレフィックスを新たに登録する際には，プレフィックスのアドレスとマスクから対応するページの位置を割り出し，そのページが存在するかどうかを確認し，もしなければページを割り当てる．その後，アドレスとマスクからそのプレフィックスがページ内の何番目の要素に対応するかを計算し，その要素の値を真とする．ページの参照を記録するデータ構造としてはマスク長 1〜24 までのページに関しては配列を用いているが，最下層の 25〜32 に関しては Map を利用している．これは最下層のページ数が 2^{25} (=33,554,432) と非常に多く，配列を確保するだけでも多くのメモリを消費してしまうにも関わらず，現実問題としてマスク長 25 以上のプレフィックスはほとんど存在せず，無駄が多くなるためである．

プレフィックスの最長マッチを行う際は，与えられた IP アドレス (又はプレフィックス) から対応するページとページ内の要素を計算し，そこから上方向に向かって真の要素が見つかるまで 1 つづつ要素を遡っていく．図 6 からも分かるように，ある要素の親要素はその要素から 1 を引いて 2 で割ることで得られる．ページの移動に関しても，数は異なるが対称な木構造であるという点で同様であり，同じ要領で計算可能である．

表 2 格納時間とストレージ容量

項目	格納時間	所要容量
BGP 経路情報	122 秒	307MB
AS トポロジ	92.5 秒	86.4MB
BGP 経路情報の構造化	141 分 43 秒	519.4MB
プレフィックスキャッシュ	84 ミリ秒	約 4MB

6. 評価

今回実装を行った EGP KB-Agent と KB ストレージサーバの性能評価を行なった．評価に使用したマシンのスペックは CPU が Intex Xeon E3-1271，OS は Arch Linux (kernel 4.8.13)，ストレージには Intel SSD MLC である．評価項目は以下の 4 つである．

- BGP 経路情報を RDF4J で格納するのにかかる時間とストレージ容量
- 生成された AS トポロジを RDF4J で格納するのにかかる時間とストレージ容量
- 構造化された BGP 経路情報を Neorj で格納するのにかかる時間とストレージ容量
- プレフィクス木を生成するのにかかる時間とメモリ容量

表 2 に評価項目 4 つの格納時間とストレージ容量を示す．今回利用した BGP 経路情報は Routeviews プロジェクトが BGP コレクタを設置して収集したものであり，ウェブ上で公開されている．この BGP コレクタは複数の AS の BGP ルータとピアリングすることで経路情報を得ているが，使用したのは AS7500 (WIDE) 経由の IPv4 フルルートのみである．

評価の結果，フルルートを RDF ストレージに格納した際のストレージ容量は経路情報と AS トポロジを合計しておおよそ 400MB 程度であることから，複数の BGP ルータからフルルートの情報を収集して RDF ストレージに保存することも十分に可能であることがわかった．また，フルルートを RDF ストレージに格納するのに要する時間は 200 秒程度であり，これは BGP ルータがフルルートをピアから受け取るのに数十分かかることを考慮すると十分に速いと言える．しかしながら，Neo4j を用いてフルルートの経路情報を構造化して保存しようとした場合，格納に約２時間半かかり，500MB 以上のストレージを要する．プレフィックスキャッシュに関してはプレフィックス木の論理構造を愚直に実装した場合，500MB を要していたが，提案手法では 4MB 程度のメモリに収まった．

7. 結論

本稿では情報共有基盤 KANVAS における BGP の経路情報の収集・格納に関するモジュールの設計・実装を行った．本研究では BGP 情報を単に RDF ストレージへそのまま格納するだけではなく，AS トポロジの生成やクエリを受けた際に必要となる最長プレフィックスのためのキャッシュの生成，そして RDF 以外のグラフライブラリを用いた BGP 経路情報の構造化を行った．

評価の結果，フルルート経路情報を複数の BGP ルータから受け取り，経路情報および生成された AS トポロジを RDF ストレージに格納することは十分に可能であるという結論に至った．しかし，BGP 経路情報を構造化して保存するのには他と比較して多くの時間と容量を要するため，必ずしも現実的ではない．これは，構造化された AS パスが AS トポロジに比べて非常に巨大であることに起因すると考えられる．代替案として，ストレージには元の情報と AS トポロジのみを格納し，構造化した経路情報はキャッシュとしてアクセスサーバに置くという方法が考えられるが，メモリ上でのキャッシュが十分に実用的な大きさの範囲に収まることを検証する必要があるだろう．

今回の実装では BGP コレクタ側でファイルにダンプした経路情報を読み込んでストレージサーバに送信するという方法で実装を行ったが，既に述べたとおりこの方法はダンプを行う時間周期の分だけ更新の遅延が生じる．今後の課題として，よりリアルタイムなモニタリングを可能にするために，BGP ルータと直接コネクションを確立して BMP により経路情報を受け取る方法に切り替える必要があるだろう．

参考文献

[1] 川口慎司：KANVAS : ネットワーク知識の活用に向けたオントロジを利用したオープンな情報共有基盤，慶應義塾大学大学院理工学研究科修士論文 (2016).
[2] Ohshima, R., Kawaguchi, S., Kamatani, O., Akashi, O., Kaneko, K. and Teraoka, F.: Construction of Routing Information Knowledge Base towards Wide Area Network Management, *The 10th International Conference on Future Internet, CFI '15, Seoul, Republic of Korea, June 8-10, 2015*, pp. 76–83 (2015).
[3] 川口慎司：ノード間クロスレイヤ協調機構 CLINEX の改良，慶應義塾大学理工学部情報工学科卒業論文 (2014).
[4] Zabbix: http://www.zabbix.com/.
[5] Nagios: https://www.nagios.org/.
[6] Peter, S., Javed, U., Zhang, Q., Woos, D., Anderson, T. and Krishnamurthy, A.: One Tunnel is (Often) Enough, *Proceedings of the 2014 ACM Conference on SIGCOMM*, SIGCOMM '14, pp. 99–110 (2014).
[7] Godfrey, P. B., Ganichev, I., Shenker, S. and Stoica, I.: Pathlet Routing, *Proceedings of the ACM SIGCOMM 2009 Conference on Data Communication*, SIGCOMM '09, pp. 111–122 (2009).
[8] goBGP: https://osrg.github.io/gobgp/.
[9] bgpdump: .
[10] RDF4J: http://rdf4j.org/.
[11] Noe4J: https://neo4j.com/.