Vol.2021-IOT-52 No.33
2021/3/2

SINDANシステムを利用したコンテナプラットフォームにおけるネットワーク環境の計測

石原　知洋[1]　北口　善明[2]　阿部　博[3]

概要：今日，Docker などのコンテナ技術を用いたポータブルなマイクロサービスによるシステム構成は広く行われている．現時点では，それらのシステム構成はクラウドサービス基盤上に構築することが多いが，今後 IoT およびエッジコンピューティングが普及するに従い，マイクロサービスが動作する環境は多種多様なものとなることが予想される．その際に，それらの多種多様な環境において，システム全体，および個々のマイクロサービスがネットワークに対して要求する機能・品質が正しく提供できているかを確認することは不可欠である．本研究では，そのような多種多様なマイクロサービス基盤において，既存のネットワーク品質測定システムである SINDAN システムを利用し，コンテナ技術を用いてポータブルにネットワークの品質を測定する仕組みについて提案をおこなう．

Network Diagnosis on Container Environment using SINDAN monitoring agent

Abstract: Recently, portable microservices based on container technologies such as Docker are widely used for system configuration. However, as IoT and edge computing become more widespread, the environments in which microservices operate are expected to become more diverse. In such diverse environments, it is essential to confirm whether the functions and quality required by the entire system and individual microservices can be provided correctly to the network. In this research, we propose a portable network quality measurement system using container technology, based on the SINDAN system, which is an existing network quality measurement system for such diverse microservice infrastructures.

1. はじめに

近年，ネットワーク上のシステムについて Docker などのコンテナ技術を用いたマイクロサービス基盤上に構築される事例が増加している．コンテナ技術を用いたマイクロサービスアーキテクチャは，ハイパーバイザ型の仮想マシンを利用したシステム構成に比べてオーバーヘッドが低く，また個々のコンテナイメージの再利用が容易であるという理由が，サービス基盤として採用される理由となっている．ハイパーバイザによる仮想化技術においては，ホストのオペレーティングシステム (OS) とは別にそのホスト上で動作する仮想マシン (VM) が個別に完全な OS を持つことで各 VM が独立して動作するが，コンテナ技術はホスト OS のカーネルが提供する機能は共有して利用しつつ，ホスト OS が持つ Linux Network Namespaces の機能によりファイルシステムやネットワークの空間を分離している．

コンテナ技術の利用領域としては，当初はクラウド技術と同じくデータセンターなどの環境が主なものであったが，昨今ではコンテナの再利用性及び可搬性に着目して，移動体や IoT 環境での利用が考えられつつある．クラウド環境の普及により，さまざまなサービスがクラウド基盤上に展開することになったが，システムを構成するコンポーネントが物理的・ネットワーク的に離れた場所に存在することで，システム上で必要な通信がその区間のネットワークを通過することとなる．そのため，システムのコンポーネント同士が近接している場合に比べて，通信帯域を消費するネットワーク区間が増加し，また遅延の増加や通信障害の確率が増加する．そこで，通信使用量および遅延の削減の

[1] 東京大学 総合文化研究科
Graduate School of Arts and Sciences, The University of Tokyo
[2] 東京工業大学 学術国際情報センター
Global Scientific Information and Computing Center , Tokyo Institute of Technology
[3] トヨタ自動車株式会社
TOYOTA MOTOR CORPORATION

ため，移動体や IoT 環境ではエンド端末にネットワーク的に近い機器や，エンド端末自体でタスクを実行するエッジコンピューティングが近年提唱されている．

エッジコンピューティングによるタスクの分散はさまざまな実装方法が提案されており，例えば Denso 社はメジャーなコンテナプラットフォームである Kubernetes を自動車の上に構築する Misaki[1] を提唱している．Misaki は使用するシステム・アプリケーションについてクラウド上と自動車上のそれぞれのコンテナホストの間で自由にタスクを移動し，アプリケーションが求める様々な通信への要求に対応することを目的としている．

一方で，コンテナ技術を用いたシステムを安定させて運用させるためには，システムを構成する個々のコンテナが正常に動作しているか監視することが不可欠となる．従来のようなデータセンター内・データセンター間を想定したコンテナプラットフォームにおいては，ネットワークの機能や状態などは集中的に管理・監視されているため，その管理・監視機能に基づいたシステムの運用が可能であった．しかしながら，エッジコンピューティングなどの，コンテナホストが広域に分散する環境においては，ネットワークの性質や機能，また品質などはかなりのばらつきが発生することとなる．例えば，自動車などの移動体を例に取った場合，そのコンテナホストの接続性は当然ながら無線で提供されることとなる．さらに，自動車の接続性は単一の 4G/5G などの携帯網によるものだけでなく，例えば自宅・ガソリンスタンドなどで Wi-Fi に接続したり，携帯網への接続も冗長性のため複数のキャリアにまたがった接続性を持つことが考えられる．現在の多くコンテナプラットフォームで使用するネットワークは，分散したコンテナホストでコンテナ自身から見た画一的な接続性を VXLAN などのトンネル技術を用いて提供している．これらのオーバーレイのネットワークを提供するための足回りのネットワーク接続性についてはコンテナ内部からは隠ぺいされており，詳しい状態を取得することが困難となる．

そこで本研究では，多種多様なネットワーク接続性を持つコンテナ基盤において，既存のネットワーク品質測定システムである SINDAN システムを利用し，コンテナ技術を用いたポータブルなネットワークの品質計測についての提案をおこなう．本提案の実証モデルとして，現在広く使われているコンテナ実装である Docker を用いた計測用のコンテナについて論じる．

2. コンテナプラットフォームにおけるネットワーク環境

1 章で述べたように，コンテナ環境下において，コンテナに提供されるネットワークは多種多様なものとなる．ここではそれらの多様なネットワークについて，要素ごとに整理する．

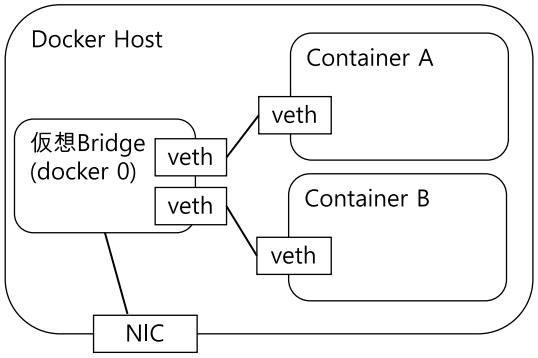

図 1 一般的な Docker Network の構成
Fig. 1 A typical Docker Network configuration.

2.1 コンテナネットワークモデル

コンテナ環境ではコンテナから外部ネットワークへの通信，およびコンテナ間の通信を媒介するために，ホスト OS のネットワークから分離されたネットワークをコンテナに対して提供する．このネットワークの分離は Linux Network Namespaces という機能を用いて実現されており，コンテナ間の通信を制御し，また独立したルーティングテーブルを持つことでそれぞれ独自にネットワークフローを制御することができる．

Docker Network Docker Network はコンテナ実装である Docker で提供されるネットワークモデルである．図 1 に一般的な Docker Network の構成を示す．Docker Network では，コンテナごとに異なるネームスペースを割り当て，各コンテナのネットワークインターフェースをそのネームスペース内に配置する．ネームスペース間は分離されているため，仮想ブリッジを用いてホストネットワークを含む各ネームスペース間の通信を媒介している．通常では各ネームスペースは外部のネットワークとはブリッジングせず，レイヤ 3 的に外部と分離されている．そのため，多くの場合では，ホスト OS 自体が NAT ルータとして動作し，コンテナからの通信をフォワーディングおよびアドレス変換することで外部への通信を実現している．

Container Network Interface(CNI) Docker Network ではホスト内に複数のネームスペースを作成し，ホストのネットワークからの分離をおこない，コンテナ間の通信の媒介は仮想ブリッジにより実現している．しかしながら，大規模環境で複数の物理マシンにわたってコンテナを展開する場合，異なる物理マシンに展開したコンテナ同士はそのままでは相互に通信をおこなうことができない．そのような通信をおこなう場合には，始点から終点までのポートフォワーディングを設定するか，別途トンネルなどを設定する必要がある．これらの自動的な構成を実現するために，いくつかネットワークへの付加機能が開発された．これら

のネットワークへの付加機能について相互運用性を高めるために制定された規格が Container Network Interface(CNI) である．

CNI は物理サーバ間の通信を媒介するために必要最小限の機能を要求しており，CNI に対応しているネットワーク付加機能の実装は複数存在する．

Docker コンテナでは，デフォルトで次の 3 種類の Docker Network が提供される．

none none ネットワークは，loopback インタフェースを提供する．したがって，none ネットワークを利用するコンテナは外部ネットワークとの疎通性を持たない．

host host ネットワークは，Docker ホストの物理インタフェースを共有するモードで，Linux ホスト上でのみ動作する．仮想インタフェースを介さない通信が可能となるため通信性能を最適化することができるが，Docker ホストと密結合するため Docker ホストで利用しているポート番号をコンテナで利用することはできない．

bridge bridge ネットワークは，Docker コンテナ作成時に，ネットワークオプションとしてデフォルトで選択されるモードで，Docker 内に構築された仮想ブリッジネットワークを介して Docker ホストおよび接続されたコンテナ間との通信を実現する．Docker ホストの Docker0 インタフェースを経由して，外部ネットワークとの通信も可能となる．

以上のデフォルトで提供されるもの以外に，ユーザ定義により次の Docker Network を構築できる．

bridge ユーザ定義の bridge ネットワークは，独立したネームスペースでの利用を可能とする．独立したネームスペースを利用することで，デフォルトの仮想ブリッジネットワークにおける制限（コンテナ動作中のアタッチ/デタッチができない，設定変更時に Docker デーモンの再起動が必要など）から開放される．

overlay overlay ネットワークは，Docker クラスタ環境を構築する Docker Swarm にて利用され，複数の Docker ホスト間における分散ネットワークを VXLAN 機能を用いて実現する．

macvlan macvlan ネットワークは，host ネットワークと同様に Docker ホストの物理インタフェースを利用するモードとなるが，独立した MAC アドレスをすることでインタフェースの分離を可能とする．802.1Q によるタグ VLAN インタフェースを利用することで，Docker ホスト側のインタフェースを分離でき，柔軟なトラフィック制御もおこなえる．なお，利用するには Docker ホストが Linux で，プロミスキャスモードをサポートしているネットワークインタフェースが必要となる．

2.2 Docker Network による IPv4/IPv6 ネットワーク構築

2.1 節で整理した Docker Network を利用して IPv4 および IPv6 ネットワークを構築する際に，Docker コンテナにて利用可能なネットワーク構成がどのようになるか，本節にて整理する．

2.2.1 IPv4/IPv6 with host/macvlan

host ネットワークおよび macvlan ネットワークを利用する場合，Docker コンテナは Docker ホストを同じネットワークに直接接続された構成となり，Docker ホストのネットワーク環境を透過的に扱うことができる．特に macvlan の場合には Docker ホストと独立した MAC アドレスをコンテナ側で利用可能となるため，IP アドレスの自動設定機構（IPv4 における DHCP や IPv6 における SLAAC など）の利用が可能となる．

2.2.2 IPv4 with bridge

IPv4 環境を bridge ネットワークにて構築する場合，仮想ブリッジネットワークでは IPv4 プライベートアドレスが利用され，Docker ホストが NAT ルータとして動作することで外部インターネットとの接続性を提供する．したがって，外部ネットワークから Docker コンテナへの接続性を提供する場合には，Docker ホストにおけるポートフォワーディング機能を利用する必要がある．

2.2.3 IPv6 with bridge

IPv6 では IPv4 と比較して潤沢なグローバルアドレスを利用することができるため，bridge ネットワークによる IPv6 環境構築手法において次の 3 つの手法が存在する．

L3 Routing bridge ネットワークでグローバルユニキャストアドレスを利用し，Docker ホストにてルーティングする形態．もっとも制限の少ない IPv6 ネットワーク環境を構築できるが，Docker の bridge ネットワークで利用する IPv6 グローバルユニキャストアドレスを別途用意する必要がある．また，上流のルータにおいて bridge ネットワークへの経路設定も別途必要となる．

NDP Proxy Docker ホストが接続するネットワーク環境の一部を切り出して bridge ネットワークで利用する形態．Docker ホストが接続する外部ネットワークが 2001:db8:1:1::/64 の場合，/80 に分割したネットワーク（この場合 2001:db8:1:1:a::/80 など）を bridge ネットワークに設定する．この時，外部ネットワークと bridge ネットワーク間の通信を実現するために，Docker ホスト上では NDP プロキシにより NDP パケットを中継する必要がある．なお，bridge ネットワークで利用される IPAM ドライバでは，外部ネットワークが SLAAC を利用した動的アドレス割り当ての場合においても，重複アドレス検索が実施されない点で注意が必要となる．

ULA+NAT IPv4 のプライベートアドレスに相当する Unique Local Address (ULA) を利用し，Docker ホストが NAT ルータとして動作する形態．IPv6 の Network Address Port Translation（NAPT: 所謂広義の NAT）は標準化されておらず非推奨ではあるが，Linux の実装として存在しているため IPv4 と同様に利用することができる．本形態は，IPv4 における bridge ネットワーク利用形態と同様の仕組みとなるため，IPv4 による Docker 環境の利用者にとっては受け入れ易いものと考えられる．ただし，IPv6 で ULA を利用しているため一般的なデュアルスタック環境と異なり，IPv4 プライベートアドレスが仕様上優先利用される点に注意が必要である [2].

3. 関連研究

コンテナ，およびクラウド環境の監視については，過去にもいくつかの提案やツールの開発がおこなわれている．cAdvisor(Container Advisor)[*1]は google によって開発された Docker コンテナの監視システムであり，コンテナ環境に計測用のコンテナを作成し，CPU 使用率やネットワークの利用量などの計測をおこなう．同様に，コンテナ環境において自動的に計測用のコンテナをデプロイし，パッシブモニタによってネットワーク環境を計測する仕組みとして ConMon が提案されている [3]．ConMon は各コンテナホストにおいてコンテナのデプロイ状況の監視をおこなうマネージャを配置し，ホスト上に新しいコンテナがデプロイされるたびに計測用コンテナをデプロイし，ホスト内部のコンテナ間の通信をパッシブに監視することでネットワークの状況を測定する．Park らの論文 [4] では，CNI の様々な実装について，その性能測定の方法と測定結果について述べている．

これらの監視については，従来のデータセンターなど，ネットワーク環境が画一的であり，かつ全てのコンテナホストが一元管理・監視されている環境を想定しており，本研究がターゲットとしているようなネットワーク接続性が多様になる環境については想定をしていない．

また，本研究で想定している IPv6 の接続環境については，北口らの研究 [5] において言及されているクライアント接続環境を想定した．

4. SINDAN クライアントを利用した計測コンテナの実装

本研究が提案するコンテナを利用したネットワーク計測の実証のため，Docker コンテナによるネットワーク計測機構を開発した．本計測システムは，既存のネットワーク環境計測システムである SINDAN エージェント [6] を利用し，SINDAN エージェントをコンテナ化することで可搬性のある計測システムを検証した．

4.1 SINDAN エージェント

SINDAN エージェントはインターネット接続環境におけるエンドホストの通信障害を発見したり，障害発生時の問題切り分けのため，定期的にマルチレイヤに渡る計測をおこない，各計測レイヤごとの計測結果を蓄積してネットワーク管理者に提供する．

SINDAN エージェントの計測はハードウェア層，データリンク層，インターフェース層，ローカルネットワーク層，グローバルネットワーク層，名前解決層，アプリケーション層の計測レイヤごとに実施され，下位計測レイヤから順に設定内容の確認，インターフェースの情報，アクティブ計測によるネットワーク状態の測定をおこなう．

4.2 コンテナ環境におけるネットワーク計測

コンテナを利用してネットワーク環境を計測する場合，コンテナの内部のネットワークは，通常ホストのネットワークからネームスペース的に分離されている．そのため，ホスト上で計測可能な項目について，すべてが同様に計測可能とは限らない．また，コンテナにおけるネームスペースの分離はコンテナ起動時の設定によっても制御可能であり，例えば Docker であれば，コンテナ起動時に Docker network を host に指定することでコンテナのネットワークをホストから分離せずに，コンテナからホスト上のネットワークインターフェースおよび経路表について直接アクセスすることが可能となる．

また，サービスを含めたネットワークの健全性をチェックする場合，一部の計測項目についてホスト上の設定ファイルを確認することが必要となる．例えば UNIX 系 OS におけるホストの名前解決についての設定は /etc/resolv.conf に記述されており，ファイルシステム的に分離されているコンテナ内部からは確認することができない．こちらも同様にホストの /etc ディレクトリなどネットワークの設定が記述されている部分について，コンテナ内のディレクトリに読み込み専用でマウントすることで確認することが可能となる．

4.3 コンテナ環境での計測項目の変化

SINDAN エージェントにおける主な計測項目について，コンテナ環境下で計測内容に変化が生じるもの，また Docker network を host として起動した場合と，ホスト OS の/etc ディレクトリをマウントすることで計測可能なものについて記述したものが表 1 である．計測レイヤごとの主な計測内容は下記の通りである．

Docker network として通常のホスト OS と分離される bridge を利用する場合には，ほとんどの項目についてホ

[*1] cAdvisor: https://github.com/google/cadvisor

表 1 SINDAN 計測項目のコンテナ環境における差異と対応可能性
Table 1 Differences in SINDAN measurement items at container environments

計測レイヤ	計測項目	ホスト OS との相違	–net =host	/etc マウント
ハードウェア層	利用している OS 情報	あり	×	✓
	CPU 情報（周波数・温度・電圧）	あり	×	×
	時刻情報（時刻・同期情報）	なし	-	-
	時刻源（NTP サーバアドレス）	あり	×	✓
	MAC アドレス	あり	✓	×
データリンク層	インターフェースの up/down	あり	✓	×
	インターフェースのタイプ（Ethernet, Wi-Fi, 3G, LTE, など）	あり	✓	×
	インターフェースの MTU	あり	✓	×
	インターフェースの media 値（100BaseTX, 1000Base-T など）	あり	✓	×
	無線 LAN の接続 SSID	あり	✓	×
	無線 LAN の接続先 BSSID	あり	✓	×
	無線 LAN のチャネル・RSSI・ノイズ・通信レート	あり	✓	×
インターフェース層	対象インターフェースの IPv4 における設定情報	あり	×	✓
	対象インタフェースの IPv6 における設定情報	あり	×	✓
	IPv4 自動アドレス設定の確認	あり	✓	×
	IPv4 アドレス，ネットマスク	あり	✓	×
	IPv4 デフォルトルータ	推測可能	✓	×
	IPv4 ネームサーバ	あり	×	✓
	IPv6 リンクローカルアドレス	あり	✓	×
	IPv6 RA フラグによる自動アドレス設定の確認	あり	✓	×
	IPv6 RA のプレフィックス情報	あり	✓	×
	IPv6 RA のフラグ情報	あり	✓	×
	IPv6 アドレス・プレフィックス長	あり	✓	×
	IPv6 デフォルトルータ	推測可能	✓	×
	IPv6 ネームサーバ	あり	×	✓
ローカルネットワーク層	IPv4/IPv6 デフォルトルータへの到達性	推測可能	✓	×
	IPv4/IPv6 デフォルトルータへの遅延（最大・最小・平均・偏差）	推測可能	✓	×
	IPv4/IPv6 デフォルトルータへのパケットロス	推測可能	✓	×
	IPv4/IPv6 ネームサーバへの到達性 (ICMP)	推測可能	×	✓
	IPv4/IPv6 ネームサーバへの遅延（ICMP, 最大・最小・平均・偏差）	推測可能	×	✓
	IPv4/IPv6 ネームサーバへのパケットロス (ICMP)	推測可能	×	✓
グローバルネットワーク層	外部サーバへの IPv4/IPv6 到達性	なし	-	-
	外部サーバへの IPv4/IPv6 通信の遅延（ICMP, 最大・最小・平均・偏差）	なし	-	-
	外部サーバへの IPv4/IPv6 パケットロス (ICMP)	なし	-	-
	外部サーバへの IPv4/IPv6 通信経路	推測可能	-	-
	外部サーバへの IPv4/IPv6 Path MTU	あり	✓	×
名前解決層	ローカルネームサーバを用いた IPv4/IPv6 での A/AAAA レコードの応答情報	あり	×	✓
	ローカルネームサーバを用いた IPv4/IPv6 での A/AAAA レコードの応答遅延	あり	×	✓
	ローカルネームサーバを用いた IPv4/IPv6 での A/AAAA レコードの TTL	あり	×	✓
	パブリック DNS を用いた IPv4/IPv6 での A/AAAA レコードの応答情報	なし	-	-
	パブリック DNS を用いた IPv4/IPv6 での A/AAAA レコードの応答遅延	なし	-	-
	パブリック DNS を用いた IPv4/IPv6 での A/AAAA レコードの TTL	なし	-	-
アプリケーション層	外部 ssh サーバに対する IPv4/IPv6 アクセスでの ssh 通信	なし	-	-
	外部ターゲットサーバに対する IPv4/IPv6 でのポートスキャン	なし	-	-
	外部 Web サーバに対する IPv4/IPv6 アクセスでのステータスコード	なし	-	-
	外部 Web サーバに対する IPv4/IPv6 アクセスでの通信スループット	なし	-	-
	外部 Web サーバに対するページ表示の speed index	なし	-	-
	Web による通信速度測定サイトによるスループット	なし	-	-

スト OS 上で測定した場合と異なる情報が取得される．SINDAN エージェントでは，計測ホストの外部リンクの測定としてデフォルトルータに対して ICMP での死活監視・遅延・ジッタ・パケットロスなどの測定をおこなっているが，ホスト OS のデフォルトルータについては，traceroute をおこなった結果を確認することで，2 ホップ目のルータがホスト OS のデフォルトルータとして推測できるため，ホスト OS で測定した場合とほぼ同様の結果を取得することができる．コンテナが使用している Docker network は，bridge モードか，host ないし macvlan モードかの判別についてはコンテナ内部でネットワークのデバイスドライバを確認することで判別することができるので，コンテナ内部でホスト OS のデフォルトルータの推定と，デフォルトルータに対する測定は可能である．

コンテナで測定した場合とホスト OS 上で測定した場合で計測結果に差異がある項目については，Docker Network を host にするか，もしくはホスト OS の /etc ディレクトリをマウントすることで，ハードウェア層の一部の項目を除きコンテナの内部から同様に取得可能になる．しかしながら，それぞれの設定を有効にしてコンテナを作成した場合には，コンテナの本来の利点であるネットワークおよびファイルシステムの分離が不十分なものとなってしまう．

計測結果は計測レイヤが下位のもの，具体的にはインターフェース層・データリンク層・ハードウェア層のほとんどの項目がホスト OS で計測した場合と異なる結果となる．ローカルネットワーク層より上位層の計測についてはほとんどの計測項目が変化がないか，推測可能となる．そのため，コンテナ上で計測をした場合には，通信に問題があった際に通信の品質そのものは計測可能であるが，その品質が十分でなかった場合に何が原因であるかを特定することは困難である．

5. おわりに

本研究では，エッジコンピューティングや移動体を含むポータブルおよび多種多様なマイクロサービス環境において，既存のネットワーク品質測定システムである SINDAN システムを利用し，コンテナ技術を用いてポータブルにネットワークの品質を測定する仕組みについて検討をおこなった．コンテナプラットフォームにおけるネットワーク環境について調査をおこない，多様なネットワーク環境およびそれぞれにおける IPv4/IPv6 ネットワークの提供形態について検証をおこなった．また，コンテナを利用した計測手法について，メジャーなコンテナプラットフォームである Docker を利用し，本提案を実証するシステム開発し，それぞれの計測項目について検証をおこなった．その結果，通信品質自体の測定は可能であるが，その通信品質がどのような要素から決定しているか判断することは難しいということが確認できた．

今後はこれらの検討を踏まえ，コンテナ上でホスト OS 上と同様の結果が得られる限られた計測項目から，下位計測レイヤの状態を推定する手法などについて研究を進める予定である．

謝辞 本研究はトヨタ自動車株式会社との共同研究により実施されたものである．

参考文献

[1] Denso. Kubernetes based connected vehicle platform. In *KubeFest Tokyo 2020*. Denso, June 2020.
[2] D. Thaler, R. Draves, A. Matsumoto, and T. Chown. Default address selection for internet protocol version 6 (ipv6). RFC 6724 (Proposed Standard), September 2012.
[3] F. Moradi, C. Flinta, A. Johnsson, and C. Meirosu. Conmon: An automated container based network performance monitoring system. In *2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, pp. 54–62. IEEE, May 2017.
[4] Y. Park, H. Yang, and Y. Kim. Performance analysisof cni (container networking interface) based container network. In *2018 International Conference on Informationand Communication Technology Convergence (ICTC)*, pp. 248–250. IEEE, October 2018.
[5] 北口善明, 近堂徹, 鈴田伊知郎, 小林貴之, 前野譲二. クライアント OS の IPv6 実装検証から見たネットワーク運用における課題の考察. 情報処理学会デジタルプラクティス, Vol. 9, No. 4, pp. 902–922, October 2018.
[6] 北口善明, 石原知洋, 高嶋健人. センサデバイスを利用したネットワーク状態計測手法の評価. マルチメディア・分散・協調とモバイル (DICOMO) シンポジウム論文集, 第 2017 巻, pp. 1348–1353. 情報処理学会, June 2017.