情報指向ネットワークにおけるプログラマビリティ実現のための一実装

中村　真也[†]　阿多　信吾[†]　岡　育生[†]

† 大阪市立大学大学院工学研究科　〒 558–8585 大阪市住吉区杉本 3–3–138
E-mail: †nakamura@c.info.eng.osaka-cu.ac.jp, ††{ata,oka}@info.eng.osaka-cu.ac.jp

あらまし　IoT (Internet of Things) 技術は近年, 急速に発展し, 注目されている. IoT で主に用いられ るウェアラブル端末やヘルスケア機器は小型・軽量化が求められ, 低コストである必要があるために端末の処理能力が乏しくなる. その一方で, M2M (Machine-to-Machine) 通信といった端末間での連携が求められつつ, デバイスの制御を必要とされるため, ネットワークと連携した効率的な処理と通信が望まれる. このような背景から, ネットワークと協調してデータの取得と処理, その結果をもとに相互に連携する仕組みが求められる. 情報指向ネットワーク (ICN, Information Centric Network) は,「コンテンツ名」をもとに経路制御する点で, この仕組みと親和性が高くネットワーク内処理と機器の情報取得・制御が連携させプログラマビリティを持つネットワークアーキテクチャが実現できると期待されている. しかし, 現在情報指向ネットワークにおいて, このような処理を実装しているアーキテクチャは考えられていない. 本研究では情報指向ネットワークの一実装である NDN (Named Data Networking) の転送プログラムである NFD (NDN Forwarding Daemon) を用いて, IoT 端末で行う処理の一部をネットワーク層の通信機器に行うように拡張し, 処理ごとに仮想化された複数の NFD を連携させることで, 拡張性と柔軟性の高い IoT 向け情報指向ネットワークアーキテクチャを提案する.

キーワード　情報指向ネットワーク, IoT, ネットワークプログラマビリティ, 仮想化

An Implementation for Realization of Programmability with Information-Centric Network

Masaya NAKAMURA[†], Shingo ATA[†], and Ikuo OKA[†]

† Graduate School of Engineering, Osaka City University
3–3–138 Sugimoto, Osaka, 558–8585 Japan
E-mail: †nakamura@c.info.eng.osaka-cu.ac.jp, ††{ata,oka}@info.eng.osaka-cu.ac.jp

Abstract　Recently IoT (Internet of Things) technologies are evolving rapidly and are occupying the interest of Network fields. IoT devices have low performance so that it is mainly used as wearable or health care devices that is required to be light weight, small-sized and low cost. On the other hand, it is required to communicate interactively like M2M (Machine-to-Machine) and to process as UI (User-Interface). From this kind of circumstance, IoT devices would do communicate with network and process interactively with the result from the network. ICN (Information-Centric Network) architecture is suited for ICN architecuture with programmability that combines network processing and IoT devices-task in terms of controling packets with Content Name. There is no architecture that realize this processing. With NFD (NDN Forwarding Daemon) of NDN (Named Data Networking) project, this paper suggests the extended flexible ICN architecture for IoT that lets device in network layer processes IoT devices-task and utilize virtualized multi-NFD.

Key words　Information-Centric Network, IoT, Network-Programmability, virtualization

1. はじめに

近年インターネットは大きく進化し, 機器の遠隔操作, デジタル情報の流通, および人を中心としたコミュニケーションツールとしての役割だけでなく, 社会に存在するあらゆる「モノ」がネットワークを通じて接続し, それらが相互連携することによって, 社会システムの自動化等を実現する, 統合的な情報インフラとして変わりつつある. 特に IoT (Internet of Things) あるいは M2M (Machine to Machine) と呼ばれる技術の台頭により, 工場, 農業, 医療, スマートシティなど, 今後さまざまな分野にお

いて革新的なサービスの創出が大きく期待されている.IoT では, 従来とは桁違いのデバイス（端末）がネットワークに接続され, それらが相互に通信を行うことが想定されており, 民間の予想では 2020 年までにその数が 500 億から 1 兆に到達するとされている（例えば [1]）.

IoT が主となるネットワークでは, これまでの人のコミュニケーションを中心としたネットワークとは本質的に異なる. 例えば, 多くのノードがセンサやタグなど超小型かつ単純な機器で構成されているため, 個別ノードについては計算資源の観点からは極めて制約が大きい. したがって, 通信状況を勘案したインテリジェントな処理を各ノードに担わせることは容易ではなく, 輻輳制御等をエンドノードで行わせるかどうかも含めて検討する余地がある. また, センサノードは単純機能を長期間にわたり提供し続ける必要があり, その性質上電池駆動による維持が主となっていることから, ノードの消費電力については可能な限り小さくするべきであり, ノードの処理および通信において省電力化が求められるなどの制約も有する.

特に IoT において機器同士が連携をして動作をする M2M では, 従来のセンサネットワークのようにセンサ情報の送信ノードとセンサ情報の収集ノード（シンク）に役割が二分されており, 基本的に単方向の情報流通を主として考えていたセンサネットワークとは本質的に異なる. すなわち, 機器同士の相互連携等, 双方向の通信についても一層考慮する必要がある. また, 機器連携において多様な情報にもとづいたより柔軟な制御を実現するためには, 情報の収集, 加工, 集計等を行うだけでなく条件分岐や繰り返し制御などが具備されていることが望ましい. 加えて容易なカスタマイズを実現するためには, プログラム可能な M2M 通信による機器連携が実現されること, すなわち M2M 通信におけるプログラマビリティの確保が期待される.

しかしながら,IoT の M2M 通信および機器連携においてプログラマビリティを提供することは容易ではない. その理由として, 前述の通りセンサデバイスあるいはアクチュエータなどのエンドノード自体の計算能力は非常に小さく, さらに省電力も求められることからエンドノードにおいてプログラミング言語を備えたプログラマビリティを有することは現実的ではない点があげられる. またセンサノードの多数は汎用 CPU を搭載しておらず, そもそもノード処理自体がプログラマブルではないなどの制約もある. 一方, センサネットワークのようにシンクにおいて情報を収集, 分析し, その結果をもとにシンクからアクチュエータデバイスの制御を行う手法も考えられるが, 一切の情報収集と制御がシンクノードに集中されることとなり, エンドノード同士の直接的かつ自由な連携を行うことには適していない, シンクノードが単一障害点となるなどの制約がある. このような問題を解決するため, 本稿では機器連携のプログラマビリティをネットワーク上で分散的に提供可能な手法について, これを情報指向ネットワークにより実現することを目標とする.

情報指向ネットワークとは, 近年ポストインターネットアーキテクチャの一つとして注目されているネットワーク技術である. ここでは, 従来の IP アドレスに変わり, コンテンツ名自体をアドレスとして経路制御をおこなうことで, 現在のインターネットにおける主たる情報取得である「何（what）」を主体としたコミュニケーションを実現する. 情報指向ネットワークでは, 情報（コンテンツ）主体の経路制御を行うことで, 従来のインターネットにおいて存在する「情報主体と情報を保有する機器の強い結合」を緩和させ, 情報指向ネットワークが標準機能として有するコンテンツキャッシュにより情報をネットワーク内に自由に配置させることが可能になるほか, 情報指向ネットワークにおける「アドレス」に相当する「(可読性を有する）名前」にさまざまな意味を持たせることで, 単純な静的コンテンツの指定だけでなく, 指定されたパラメータによって動的にコンテンツを変化させるなど, 柔軟な情報取得が実現できる. 具体的にはこれまでにも例えば動画取得の場合において, フレームレートや解像度, エンコード方法などを名前に指定してコンテンツ要求を行うアプリケーションや,VoIP における SIP シグナリングを名前の要求手順にマッピングする方法などが考案されている [2].

この性質を応用すれば,M2M における機器連携について, 機器の制御を名前に組み込んだコンテンツ名を要求することで, 機器制御と情報取得を統合的に実行可能な通信が容易に実現できる. 特にエンドノードは単なる名前の指定のみで情報取得に加えて機器制御が実現できるため, 計算資源に限りがある組み込みデバイスにおいては, 名前指定の柔軟性が直接的に制御の柔軟性として機能できる. 以上のように, 情報指向ネットワークの特徴は IoT の課題を解決するのに非常に適合性が高いといえる.

しかしながら, 情報指向ネットワークを用いてより高度な M2M の機器連携を実現させるためには, 前述の通りネットワーク上でプログラマビリティを提供し, いわゆる中継ノードにおいてカスタマイズ可能な処理（ネットワーク内制御：in-network processing）を提供する必要がある. 情報指向ネットワークにおいてもこれまでの研究で画像のトランスコーディングなどのネットワーク内制御を実現する手法などについては検討されてきている [3] が, 情報指向ネットワーク自体にプログラマビリティを提供し, 高いカスタマイズ性を実現するためのアーキテクチャについてはまだ検討がなされていない. そこで本稿では, 情報指向ネットワークにプログラマビリティを提供する一手段として, コンテナ仮想化を用いて名前空間ごとに異なる転送処理を実装した仮想中継ノードを連結させ多様なネットワーク内制御を提供し, これを組み合わせることによってプログラマビリティを実現するネットワークアーキテクチャを提案する.

2. 関 連 研 究

ICN の概念を IoT に導入するアイディア自体は新しいものではなく、すでに多くの研究者により検討がなされているところである。例えば [4] では、NDN テストベッド上で IoT を実現するためのケーススタディについて示されている。また、[5] では ICN の利用シナリオについて整理されて示されているが、ここでも IoT は有力なユースケースとしてあげられている。その他の IoT のユースケースの例としてホームネットおよび HEMS (Home Energy Management System) への導入 [6] [7]、

表 1　Forwarding モジュール

Forwarding Pipelines	Incoming Interest Pipeline
	Outgoing Interest Pipeline
	Incoming Data Pipeline
	Outgoing Data Pipeline
Forwarding Strategy	Triggers
	Actions

UCLA の建物管理への導入 [8]、スマートグリッドへの応用 [9] などが検討されている。また、ICN 上で IoT を実現する場合の検討課題として、トラヒック制御 [10]、アドレッシング、ルーティング [11]、セキュリティ、資源発見 [12] などが挙げられている。[13] では、Publish/Subscribe 機構を用いた IoT の実現手法について検討されている。[14] ではワイヤレスセンサネットワークに NDN アーキテクチャを適用した場合のトラヒック集約手法について検討されている。IoT トラヒックの ICN コンテンツキャッシュの適用については、[15] [16] で検討されている。しかしながらこれらの関連研究では、そのほとんどがセンサ情報の取得の範疇に留まっており、機器制御を含めた柔軟な連携機構の実現までに至っていない。

3. 提 案 手 法

ICN は機器制御と情報取得の統合的な処理の実現が容易に可能なアーキテクチャとして注目されている．ICN が経路制御に用いるコンテンツ名を制御名として活用し, ネットワークに端末の処理を連結することで端末の情報取得と制御を組み合わせた柔軟な処理を実現する．本論文では, 情報指向ネットワークの統一的なアーキテクチャがないため, NDN (Named-Data-Networking) プロジェクトの NFD の (NDN Forwarding Daemon) を用いて実装する [17]．NFD を実行するためには C++, Boost C++ library, ndn-cxx が実行できる環境である必要がある．

NFD には Interest と Data の受信時に処理内容を記述できる転送戦略 (Forwarding Strategy) が実装されている．NFD は C++ で記述された ICN 転送デーモンであり, 6 つのモジュールで構成されている．そのうちの 1 つである Forwarding モジュールが基本的なパケットの制御を行っている．　表 1 は NFD の Forwarding モジュールの構成を示している．図 1 は NFD の各 Forwarding モジュールの関数の呼び出し図を示している．図 2 は ICN のデータ構造を示している．

- Interest

データ要求パケット, コンテンツ名を持つ.

- Data

Interest と同じコンテンツ名の Data がペアになっており, これを保持しているノードが PIT を参照し送信する.

- FIB (Forwarding Information Base)

Interest を転送する際にどの Face から来たかを示す．同コンテンツ名の Data を返信する際に参照する.

- CS (Content Store)

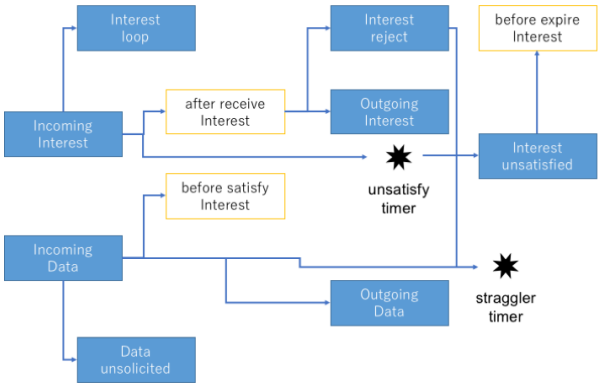

図 1　Forwarding モジュール

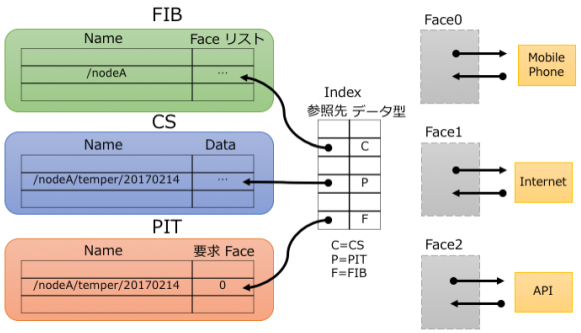

図 2　ICN のデータ構造

需要の高い Data を CS にキャッシュして, Interest で要求されれば送信する

- PIT (Pending Interest Table)

各ノードがもつ Interest の経路情報.

- Face

ICN でのインターフェース

Forwarding Pipelines

- Incoming Interest Pipeline

疎通確認用パケットであるループバックパケットであるかを確認し, PIT への記録, パケットの　Nonce (パケットループを検知するための乱数) を参照しループしていないか確認, CS・FIB を探索などどを行う．CS に該当すれば Outgoing Data Pipeline を呼び出す. なければストラテジを呼び出す.

- Outgoing Interest Pipeline

ループバックパケットでなければ Interest 送信関数 (send-Interest) を呼び出す.

- Incoming Data Pipeline

ループバックパケットか確認して, PIT を探索する．該当しなければ Data Unsolicited を呼び出し破棄する．該当すれば, CS に追加し全ての PIT Entry に対し PIT を削除し Outgoing Data Pipeline を呼び出す.

- Outgoing Data Pipeline

ループバックパケットか確認し, Data を PIT を元に送信する.

Forwarding Strategy

ストラテジクラスを継承し, 以下のメソッドをオーバーライドすることで本実装を実現する.

- Triggers

After Receive Interest:

Incoming Interest Pipeline での処理が終わった後に呼び出される.Interest 受信時のストラテジ層の振る舞いを定義できる.

Before Satisfy Interest:

Incoming Data Pipeline から呼び出される.Data 受信時のストラテジ層の振る舞いを定義できる.

Before Expire Interest:

PIT エントリが失効する際に呼び出される.

- Actions

Send Interest:Interest を送信する.

Reject Pending Interest:Interest を棄却する.

ストラテジディレクトリ (NFD/daemon/fw/) にストラテジクラスを継承したプログラムファイルを配置し NFD を再コンパイルすることで定義したストラテジによる処理を実行することが可能になる. [18]

IoT 端末の多種多様な処理要求に応えるためには各々の端末の要求に対し処理を定義する必要がある. NFD では定義された名前空間ごとに, 受信したコンテンツ名に対し各々の処理をストラテジにより定義することが可能である. しかし, 単独の NFD に対し, 新しい名前空間と対応する処理を追加するためには, 一度 NFD を停止させ, 新たに追加した処理に対するストラテジのプログラムファイルを配置し再度コンパイルを行った後に NFD を再起動させる必要がある. その間, 他の Interest が到着しても処理要求に応じることなく棄却されるため, 通信に影響を与えるおそれがある. この問題を解決するため, 名前空間ごとに独立した NFD を実行しそれらを接続させることによりある名前空間に対する処理の導入および変更に対し, 他の NFD が影響されない構成を考える. これらを単独の物理ノード上で実行させるためコンテナ仮想化による NFD の多重化を行う.

コンテナ仮想化には Docker を用いる. NFD を実行可能なマスタイメージを作成し, マスタイメージからコンテナを起動する. 名前空間によって異なる処理を提供するため, 名前空間に応じてコンパイルされた NFD をマスタイメージに取り込む. このとき, 各コンテナはマスタイメージからファイルシステムの差分管理を行うオーバーレイファイルシステム機能により NFD ごとにマスタイメージが複製されることはない. Docker では複数のコンテナ同士を接続している仮想ネットワークを自由に編成することが可能で, さらに各ノードごとに 1 つのコンテナのみをホストネットワークモードで動作させることで他ノードおよびノード内の他のコンテナとの接続を実現する. このコンテナをハブコンテナと呼ぶ. また, 名前空間ごとに定義された処理を行うコンテナを処理コンテナと呼ぶ. ハブコンテナはインターフェースを 2 つ持ち, ノードが接続しているネットワーク (IoT ネットワーク) とコンテナのみで構成しているネットワーク (コンテナネットワーク) に接続する. ハブコンテナは受信したパケットのコンテンツ名の先頭プレフィクスを元

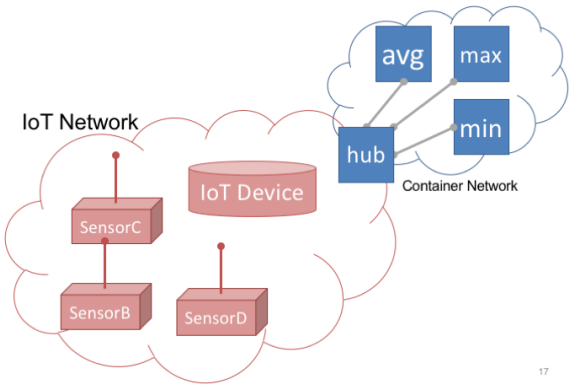

図 3　提案アーキテクチャ

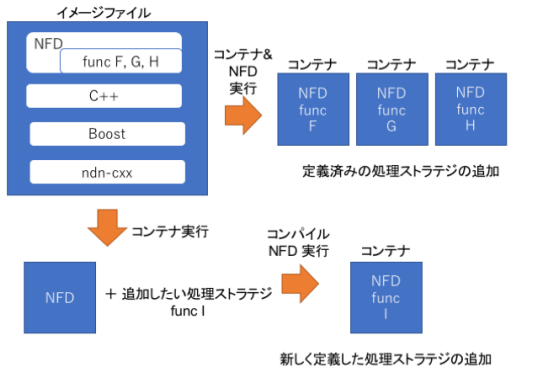

図 4　処理コンテナの追加方法

にコンテナへ転送する. 例えば,

Interest:/func-F/var1/var2/var3

を受信すると, 処理 F を実行できる処理コンテナ F に転送する. また, 先頭プレフィクス以外の要素はプログラム言語における関数の引数を意味しており, 処理コンテナはこれらの引数に応じた処理を実効する [19].

提案アーキテクチャの処理コンテナの追加例を図 4 に示す. コンテナは全て同一の NFD を実行できるイメージから起動させる.

（1） ハブコンテナをホストネットワークモードで動作させ, IoT ネットワークに接続する.

（2） ハブコンテナの NFD を IoT ネットワーク上の端末 A と接続する.

`(nfdc register ndn:`(注1) `udp://nodeA)`(注2)

（3） 処理 F を行う処理コンテナ F を動作させ, NFD をハブコンテナの NFD と接続しコンテナネットワークを構築する.

`(nfdc register ndn:  udp://hub-container)`

（4） 処理コンテナ F が受信した Interest に対し処理を実行するようにストラテジを適用する.

(注1)：ルート名前空間

(注2)：コンテンツ名 ndn: を nodeA に UDP プロトコルで転送できるように接続

(nfdc set-strategy ndn: ndn:/func-F-strategy)(注3)

（5） ハブコンテナが, 処理 F を記述した Interest を受信した際, 処理コンテナ F に転送するように設定する.

(register ndn:/func-F udp://container-F)

（6） 次に新しく処理 G を定義する. 処理コンテナ G を動作させる. しかし, 元のイメージに処理 G は定義されていないので新たにストラテジディレクトリにファイルを追加する.

（7） コンテナ上でコンパイルを行い, NFD を実行しハブコンテナと接続する.

(nfdc register ndn: udp://hub-container)

具体的な例として, ある部屋にエアコン X と温度センサ 3 台 (B, C, D) がネットワークを形成している場合を考える. エアコン X はセンサ 3 台の温度の最大値が閾値を超えれば動作するサービスを行いたい. エアコン X は十分な処理能力を持っておらず実現できないので, 同一ネットワークにある処理端末 A に

Interest : /Function-F/SensorB/SensorC/SensorD

を送信する. ここで, 同一ネットワークにある処理を担う端末 A が Interest を受信する. A には提案しているコンテナ仮想化された NFD が存在している. このコンテナがハブとしてコンテナネットワークに転送し, 処理 F を担うコンテナ F が Interest を "/" で分割した処理プレフィックスより後ろの Interest をハブコンテナを通じて IoT ネットワークから取得する. 図 3 は提案アーキテクチャのネットワーク構成 を示している. 受信した

Data : /SensorB, /SensorC, /SensorD

を元に定義された処理を実行し, 要求端末 X　に転送する. 具体的な例として, 部屋にエアコン X と温度センサ 3 台 (B, C, D) がネットワークを形成している. エアコン X はセンサ 3 台の温度の最大値が閾値を超えれば動作するサービスを行 いたい. エアコン X は十分な処理能力を持っておらず実現できないので, 同一ネットワークにある処理端末 A に

Interest : /max/SensorB/SensorC/SensorD

を送信する. 端末 A はセンサの温度を集め, 処理を行い 3 台 のセンサの最大値を算出しエアコンにデータを転送する. エアコンはネットワークに温度情報の最大値を要求するコンテンツ名を送信するだけで行いたい処理を果たすことができた. 季節が変わり温度の最小値を求める場合, コンテナに最小値を求める処理を記述し新しくコンテナを稼働させることで実現できる.

提案アーキテクチャによって, ネットワークにプログラマビリティを有したアーキテクチャを実現する. プログラミング言語は様々な関数を定義し, 関数呼び出しを繰り返すことで個々の関数は単純な振る舞いしかしないにも関わらず, 統合的には多様で柔軟な振る舞いを記述することを実現できる. この性質をプログラマビリティと呼び, 提案アーキテクチャでは処理コンテナ 1 つがプログラミング言語の関数のような振る舞いをする. プログラム言語の多くがオブジェクト指向であるように, プログラマビリティとオブジェクト指向と親和性が非常に高い. IP アドレスで構成されるネットワークの通信形態は通信相手を指定したホスト指向性である. ホスト指向性の IP アーキテクチャにプログラマビリティを持たせるために, 関数や引数と言った仕組みを実現することは難しい. しかし, ICN の持つ情報指向性はオブジェクト指向性と非常に似ており, プログラマビリティの実現に最適である. オブジェクト指向とは, 相互にオブジェクトが連携し多様で複雑な処理を実現することに優れている. ICN のコンテンツ名に処理を記述し, ノードが処理を実行することを関数と見立てると擬似的にオブジェクト指向を達成する. よって, 擬似的にオブジェクト指向なプログラマビリティを有するネットワークを実現する.

(注3)：受信したコンテンツ名 ndn: のパケットに対しストラテジ F を呼び出す

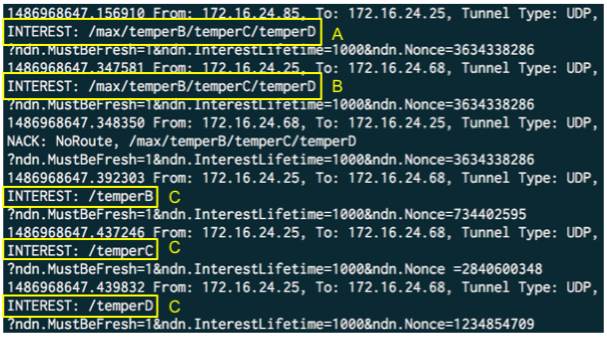

図 5　処理コンテナのログ 1

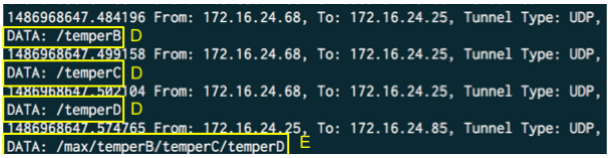

図 6　処理コンテナのログ 2

4.　評　　価

図 5,6 は処理コンテナにネットワーク内のセンサの温度情報の最大値を求める処理を行っている. 設計通りの動作をしていることを示す. (A) 端末 X が 命令を記述している Interest を受信する
(B) 隣接ノードに Interest を転送
(C) 処理実行に必要な情報を集めるために Interest を転送
(D) 要求したデータを取得
(E) 処理を実行しデータを生成, ノードへ返信

次に, NFD の仮想化によるメモリのオーバーヘッドを計測する. ハブコンテナと処理コンテナを 1 3 台起動し, vmstat を用いてメモリ占有量を測定する. コンテナ 1 台あたりにおよそ 10 20 MB のメモリが占有されていることがわかった.

5.　まとめと今後の課題

本稿では ICN を用いて通信制御とデバイス制御を連携しネッ

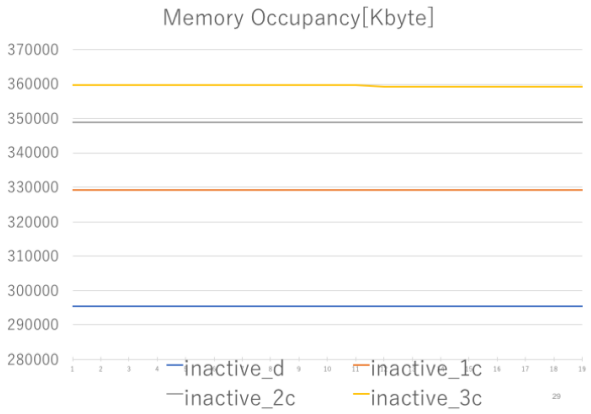

図 7　メモリ占有量

トワークにプログラマビリティを持たせることが実証できた. 処理を担う端末に集中させることで個々の端末に要求される処理能力を下げ, また, コンテナ仮想化を行うことで様々な要求に対しサービスを止めることなく対応できるアーキテクチャを実現した.

コンテナが 1 つの関数オブジェクトのように引数から, コンテナの振る舞いを決定し, ノードから温度の最大値 (/max) , 最小値 (/min) , 平均値 (/avg) を求めることができる. しかし, 1 つのコンテナで処理が完結しておりプログラマビリティ性が低い. 今後の課題として, より汎用的なネーミングスキーマを検討する.

/function-F/meta-info/var1/var2/var3

処理プレフィクスの直後にメタ情報を付与し, コンテナの振る舞いにさらに自由度を与えることができる. メタ情報として, 1 つの処理内容のプレフィックスでの範囲を示すようにし, 処理の最小単位を明示的にすることで, ハブコンテナのストラテジに受信した Interest のプレフィックスの右側からデータを生成するように定義すると,

（1） /func-F/2/func-G/2/var1/var2/func-H/2/var3/var4
（2） /func-F/2/func-G/2/var1/var2/Data-H
（3） /func-F/2/Data-G/Data-H

処理 G と処理 H により得られた値を引数として, 処理 F を行う. プログラミング言語の関数形式で記述すると以下の関数を意味している.

func-F(func-G(var1, var2), func-H(var3, var4))

これによりコンテナの処理によって得られた結果 (Data) を引数として記述することも可能になる. 今後はこれらの実装および検証を行っていく予定である.

6. 謝　　辞

本研究の一部は, 独立行政法人情報通信研究機構 (NICT) 委託研究「欧州との連携による情報指向ネットワーキングに関する実証的研究開発」の成果である.

文　　献

[1] Cisco, “Visual Networking Index Forecast and Methodology, 2015âĂŞ2020”, (2016).
[2] B. Ahlgren, et. al., “A Survey of Information-Centric Networking,” IEEE Communications Magazine, vol. 50, no. 7, pp. 26-36, July 2012, DOI: 10.1109/MCOM.2012.6231276.
[3] V. Jacobson, et. al., “Networking named content,” in Proceedings of the ACM Conference on Emerging Networking Experiments and Technologies (CoNEXT ’09), pp. 1âĂŞ12, Rome, Italy, December 2009.
[4] E. Baccelli, C. Mehlis, O. Hahm, T. Schmidt, M. Wahlisch, “Information Centric Networking in the IoT: Experiments with NDN in the Wild,” Proc. ACM ICN 2014, pp. 77-86, September 2014.
[5] K. Pentikousis (Ed), et. al., “Information-Centric Networking: Baseline Scenarios,” RFC 7476, March 2015
[6] J. Zhang, Q. Li, E. Schooler, “iHEMS: An information-centric approach to secure home energy management,” Proc. IEEE SmartGridComm 2012, pp. 217-222, Nov. 2012, DOI: 10.1109/SmartGridComm.2012.6485986
[7] R. Ravindran, T. Biswas, X. Zhang, A. Chakraborti, G. Wang, “Information Centric Networking based Homenet,” Proc. IFIP/IEEE ManFI 2013, pp. 1102-1108, May 2013
[8] W. Shang, et. al., “Securing Building Management Systems using Named Data Networking,” IEEE Network, vol. 28, no. 3, pp. 50-56, May/June 2014, DOI: 10.1109/MNET.2014.6843232
[9] K. Katsaros, et al., “Information-Centric Networking for Machine-to-Machine Data Delivery: A Case Study inSmart Grid Applications,” IEEE Network, vol. 28, no. 3, pp. 58-64, May/June 2014.
[10] J. Francois, T. Cholez, T. Engel, “CCN Traffic Optimization for IoT,” Proc. International Conference on the Network of the Future (NOF 2013), October 2013, DOI: 10.1109/NOF.2013.6724509
[11] Y. Zhang, D. Raychadhuri, R. Ravindran, G. Wang, “ICN based Architecture for IoT,” Internet draft, draft-zhang-iot-icn-architecture, work in progress, June 2014.
[12] M. Amadeo, et. al., “Information-Centric Networking for the Internet of Things: Challenges and Opportunities”, IEEE Network Magazine, Aug 2015 of Reliable Intelligent Environments, vol. 1, no. 1, pp. 47âĂŞ58, 2015.
[13] G. Polyzos1, N. Fotiou, “Building a reliable Internet of Things using Information-Centric Networking,” Journal
[14] Y. Abidy, B. Saadallahy, A. Lahmadi, O. Festor, “Named Data Aggregation in Wireless Sensor Networks,”
[15] M. Hail, M. Amadeo, A. Molinaro, S. Fischer, “Caching in Named Data Networking for the wireless Internet of Things,” Proc. Internationl Conference on Recent Advances in Internet of Things (RIoT 2015), April 2015.
[16] S. Vural, P. Navaratnam, N. Wang, C. Wang, L. Dong, R. Tafazolli, “In-network caching of Internet-of-Things data,” Proc. IEEE ICC 2014, June 2014. DOI: 10.1109/ICC.2014.6883811
[17] Zhang, Lixia, et al. "Named data networking (ndn) project." RelatÃşrio TÃĉcnico NDN-0001, Xerox Palo Alto Research Center-PARC (2010).
[18] Afanasyev, Alexander, et al. "NFD developer’s guide." Technical Report NDN-0021, NDN (2014).
[19] Fink, John. "Docker: a software as a service, operating system-level virtualization framework." Code4Lib Journal 25 (2014).