Ticket-based Access Control を適用した IoT データ流通方式におけるノード性能とセキュリティ評価

吉井 優輝† 坂野 遼平‡ 水野 修†‡

†工学院大学大学院 〒163-8677 東京都新宿区西新宿 1-24-2
‡工学院大学 〒192-0015 東京都八王子市中野町 2665-1
E-mail: †y.masaki@ieee.org, ‡{banno, omizuno}@cc.kogakuin.ac.jp

あらまし 我々は，Fog Computing を用いた IoT データ流通による新たなサービスモデルを提案している．ユーザやサービスプロバイダが所持し配置するフォグノードを介して IoT データを流通させることで，新たなサービスの創出やサービスの品質向上が期待できる．本研究を実現するためには，予期しないプライバシデータの流通がないこと，IoT データ流通方式におけるスケーラビリティの確保および想定されるセキュリティリスクを保証できることが重要となる．本報告では，チケット認証・認可基盤をユーザフォグノードおよびサービスフォグノードの中間に配置し，チケットと呼ばれるトークンを用いて IoT データのアクセス制御を行う Ticket-based Access Control を提案する．シミュレーションによりノード間の処理遅延時間を評価し，ノード内のチャネル数を増加させることで処理遅延時間の短縮につながることを示した．また，想定されるセキュリティリスクについて考察し，そのリスクを保証するための手法について述べる．

キーワード Fog Computing，アクセス制御，チケット，認証・認可

1. はじめに

IoT デバイスとクラウドコンピューティングの中間で IoT データの機械学習やデータクレンジングなどの事前処理を行う Fog Computing [1] が注目されている．図 1 に Fog Computing の概要を示す．Fog Computing は IoT デバイスから生成される IoT データがクラウドサーバに集中することで引き起こされるネットワークの輻輳を軽減するために提案されている．Fog Computing を構成するフォグノードはクラウドサーバ上のアプリケーションを自身に展開する．そうすることでクラウドに IoT データを送信せず，フォグノードで IoT データを解析し得た結果をサービス利用者に返信することで伝送遅延の軽減を可能とする．

我々は，Fog Computing を用いた IoT データ流通による新たなサービスモデルを提案している．Fog Computing を介してサービスモデルを構築する各プレーヤ間で IoT データを流通させることで，新たな IoT サービスの創出および IoT サービスの品質向上が期待できる．

提案するサービスモデルを実現するための要件として以下の 4 つを挙げる．

[要件 1] IoT データ流通による新たなサービスモデルが必要であること．

[要件 2] IoT データ流通時におけるプライバシに関わる IoT データのアクセス制御があること．

[要件 3] IoT データ流通方式におけるスケーラビリティが担保されていること．

[要件 4] サービスモデル上のセキュリティリスクに対する対処がなされていること．

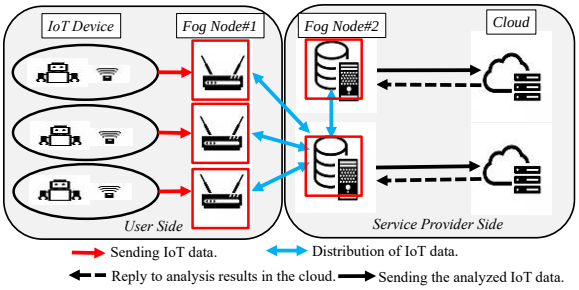


図 1 Fog Computing

本報告では，IoT データ流通によるサービスモデル実現のための方式の 1 つである Table-based Access Control [2, 3] の課題として以下の 2 点の解決を図るため Ticket-based Access Control (以下，Ticket-AC) を提案する．

[課題 1] ユーザはサービスプロバイダの数だけアクセス制御のパラメータを設定しなければならない．

[課題 2] サービスプロバイダは柔軟なアクセス制御を細かい要件に応じて実施したい．

新たなプレーヤであるチケット認証・認可基盤運用者が必要であるため新たにサービスモデルを構築し，シミュレーションにより評価を行い，Ticket-AC で重要なチケット認証・認可基盤とサービスフォグノード，チケット認証・認可基盤とユーザフォグノード間の処

理遅延時間を評価した．

この報告の構成は以下のとおりである．2 章では，既存研究である Fog Computing を用いた IoT データ流通によるサービスモデル，サービスモデル上で動作する Tag ID-based Publish/Subscribe モデル (以下, Tag ID-based Pub/Sub モデル)，Table-AC について述べる．3 章では，Table-based AC の課題および方針について述べる．4 章では，Ticket-based AC の動作について述べる．5 章ではシミュレーションによる結果と考察について述べ，セキュリティリスクの洗い出しおよび対処法について述べる．6 章でまとめと今後の展望についてのべる．

2. 既存研究

2.1 IoT データ流通に基づくサービスモデル

従来提案していたサービスモデルの概要を図 2 に示す．サービスモデルを構成するプレーヤはユーザとサービスプロバイダの 2 者である．SP (Service Provider) は複数のアプリケーションを運用している．また自身が所有する SFN (Service Fog Node: サービスフォグノード) を配置し利用する．ユーザはサービスプロバイダと契約し，IoT サービスを享受する．ユーザが所持する dev (Device: IoT デバイス) を配置している UFN (User Fog Node: ユーザフォグノード) に接続し IoT データを収集している．UFN は契約関係のある SFN に IoT データを送信する．

ユーザと契約関係がない SP が，ユーザと契約している他 SP と契約関係がある場合，ユーザを指し示す ID と IoT データ取得のための ID を SP 間で共有することで ID を利用可能となる．よって ID 解決のためのメカニズムを提供するプレーヤを配置する必要がない．よって SP とユーザの 2 者でこのサービスモデルは完結する．

上記サービスモデルによって，本研究における要件 1 を満たしている．

具体的な IoT データ流通方式は 2.2 節で述べる．

2.2 Tag ID-based Publish/Subscribe モデル

IoT データ流通には，Publish/Subscribe モデル (以下，Pub/Sub モデル) [4] を用いることを想定する．Pub/Sub モデルを採用したプロトコルおよびシステムは MQTT (MQ Telemetry Transport) [5, 6] や Apache Kafka [8], AMQP (Advanced Message Queuing Protocol) [9] などが挙げられる．IoT データ取得には Topic と呼ばれるユーザが任意に定めた ID が用いられる．Topic は IoT デバイスの数だけ存在する．よって，図 2 のモデルではサービスプロバイダから IoT データの要求をユーザフォグノードに行う際，IoT デバイスの数だけ Topic を

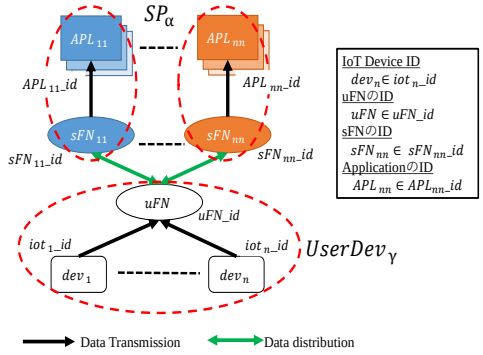


図 2 IoT データ流通に基づくサービスモデル

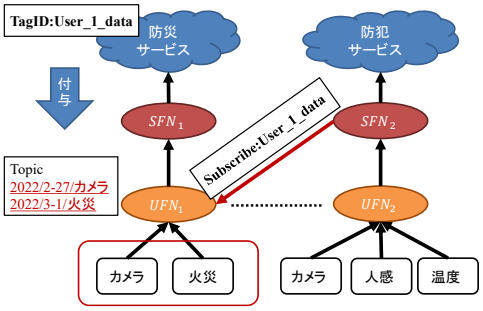


図 3 Tag ID-based Pub/Sub モデル

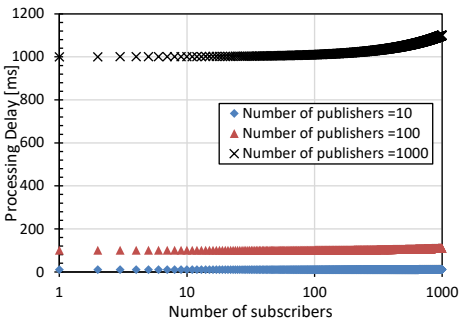


図 4 Topic-based Pub/Sub による処理遅延時間のプロット

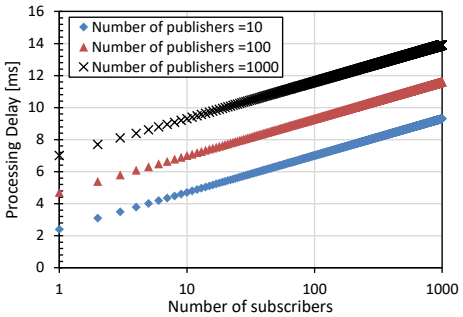


図 5 Tag ID-based Pub/Sub による処理遅延時間のプロット

購読 (以下，Subscribe) する必要がある．

SFN が 1 台の UFN に対して Subscribe を行い UFN が Subscribe を受信するまでの処理遅延時間を考える．Topic 数をN_{tp}とした場合，SFN は最大N_{tp}回の Subscribe を行う必要がある．SFN の総数をS_{all}とする．SFN と

UFN 間の通信遅延をαとし，1 台の SFN 内に発生する処理時間をβとしたとき，処理遅延時間T_uは (1) 式と見積もることができる[1].

$$T_u = \alpha \times N_{tp} + N_{tp} \times (\beta \times S_{all}) \quad (1)$$

$S_{all} = 1000$とし，$N_{tp} = 10, 100, 1000$と変動させる．$\alpha = 1$ [ms] とし，$\beta = 0.0001$ [ms] とする．上記の式をプロットしたものを図 4 に示す．処理遅延時間は，IoT デバイス数が増加することで，垂直に大幅に増加することが分かる．通信回数が増加することが原因である．

また，Topic はユーザが任意に定めることが可能なため Topic がユーザによって変更されサービスプロバイダに通知がない場合，Topic による IoT データ取得が困難となる．そのため，新たな命名規則が必要となるそこで我々は Tag ID-based Pub/Sub モデル [2] を提案している．図 3 に動作を示す．ユーザ#1 は監視カメラと火災報知器 (カメラ，火災と表記) を設置し自身の UFN_1 に接続している．監視カメラと火災報知器にはそれぞれ Topic として “2022/2-27/カメラ” と “2021/3-1/火災” が設定されている．次に UFN_1 は防災サービスを運営するサービスプロバイダと契約関係があるため Tag ID “User_1_data” が付与される．UFN_1 は Topic を管理しており，Topic と Tag ID の対応関係を保持する．防犯サービスを運用するサービスプロバイダは UFN_1 を配置しているユーザと直接的な契約関係は結んでいないが，防災サービスを運営しているサービスプロバイダと契約関係があるとすると防災サービスで付与した Tag ID を共有でき，UFN_1 に対して Subscribe が可能となる．よって Subscribe のための ID の命名規則を契約関係が存在する範囲内で解決し，サービスプロバイダからデータ要求数を削減することが可能である．

SFN が UFN に対して Tag ID を用いて Subscribe し，UFN が Subscribe を受信し，Topic に変換するまでの処理遅延時間T'_uを考える．Tag ID 数をN_{Tg}としたとき，Tag ID を Topic に変換するテーブルの行数N_Tは (2) 式となる．

$$N_T = N_{Tg} \times N_{tp} \quad (2)$$

二分探索法を用いて探索する場合，探索時間は$O(\log(N_T))$と表せる．各パラメータを (1) 式を構築した際と同様のものを用いた場合，T'_uは (3) 式となる．

$$T'_u = \alpha \times \log(N_T) + \beta \times S_{all} \quad (3)$$

上記の式は，サービスプロバイダがユーザに対して Tag ID を一つ割り当てている場合を想定したものである．(3) 式をプロットしたものが図 5 となる．変数値は，図 4 のときと同様である．図 5 より線形的に処理遅延時間が増加する点からテーブルの探索時間が支配的であることが分かる．Tag ID を用いることにより通信回数削減が可能であることから，Topic を用いる場合よりもスケーラビリティ的に優位であると言える．そのため，要件 3 を満たす．

[1] 詳細なシミュレーションを行っているわけではないためあくまでも参考数値である．

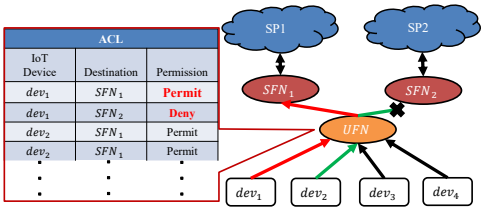


図 6 Table-based Access Control

表 1 シミュレーションパラメータ

Parameter	***Value***
Simulator	NS3 (ns-3.29) [9]
Model	M/M/1
Average arrival time (IoT Data)	1[s] to 60 [s]
Average processing time (SFN)	1, 2, …,10 [ms]
Average processing time (UFN)	1 [ms]
Average waiting time (SFN and UFN)	Exponential Distribution
Service discipline	FCFS
Number of publisher (IoT devices)	96
Number of broker (User fog node)	1
Number of subscriber (Service fog nodes)	2
Number of Topics	96
Number of Tag IDs	8
Simulation Topology	Star topology
Simulation time	3600 [s]
Number of trials	30

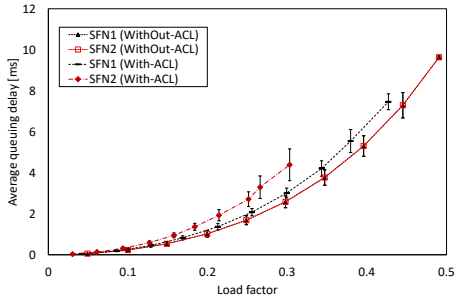


図 7 サービスフォグノードのキューイング遅延時間

2.3 Table-based Access Control

Tag ID-based Pub/Sub モデルは契約関係の範囲内で ID の命名規則を解決し，かつ Subscribe 数を削減可能

である．しかし，プライバシに関するデータを Publish する IoT デバイスの Topic に対して Tag ID が付与された場合，Tag ID による Subscribe を行うとプライバシに関するデータがユーザの意図しない形で流通する可能性がある．そこで Tag ID-based Pub/Sub モデル上で動作する Table-AC [2, 3] を提案している．

Table-AC の動作について述べる．図 6 に Table-AC の動作概要を示す. Table-AC を構成する要素は，IoT デバイス，データの送信先及びデータ送信の許可/不許可である．dev のデータ送信がSFN_2に対して不許可であった場合，データは送信されない．dev_1のデータ送信がSFN_2に対して許可された場合，データは送信される．また，SFN (Subscribe) 数をN_{sub}, Publisher (IoT デバイス) 数をN_{pub}とし，テーブルの行数をN_{table}としたとき以下の関係式となる．

$$N_{table} = N_{sub} \times N_{pub} \quad (4)$$

またテーブル探索で二分探索法を用いれば探索時間は$O(\log(N_{table}))$ となる

Table-AC を Tag ID-based Pub/Sub モデルに適用した場合と適用しない場合でシミュレーションを実施した．シミュレーションパラメータを表 1 に記載する．紙幅の都合上，シミュレーションシナリオは割愛するが文献 [2, 3] と同様のシナリオを用いている．図 7 にサービスフォグノードのキューイング遅延を示している．縦軸は平均キューイング遅延であり，横軸はサー日 s フォグノードの利用率である．SFN2 に着目し，Table-AC 適用前適用後と比較すると，キューイング遅延は最大約 25%削減できている．また，利用率も約 60%削減で来ていることが分かる．ユーザフォグノード内の Table-AC により SFN に送付するプライバシに関わる IoT データのフィルタリングが行われているため, SFN が受信する IoT データ数が減り，かつ待ち行列が短くなったためである．

上記の結果から分かる通り，Table-AC もスケーラビリティ的に優位かつプライバシに関わる IoT データの流通防止が見込める方式である．よって，本研究における要件 2 と 3 を満たす．

3. 研究課題と方針

Table-AC を IoT データ流通方式に適用することにより，プライバシに関わる IoT データ流通防止が可能である．また文献 [2, 3] および 2.3 節で述べたシミュレーション結果により，サービスフォグノードのトラヒック削減およびキューイング遅延の削減が可能であることを示した．しかし，Table-AC はサービスプロバイダの数に応じて多くのパラメータをユーザが設定する必要がある．また，テーブルの形式が固定されているため，“IoT データを 1 時間だけ使用する” や “災害のときのみ使用する” など細かい要件に応じてアクセス制御を設定することは容易ではない．統一的なアクセス制御がユーザおよびサービスプロバイダに必要である．

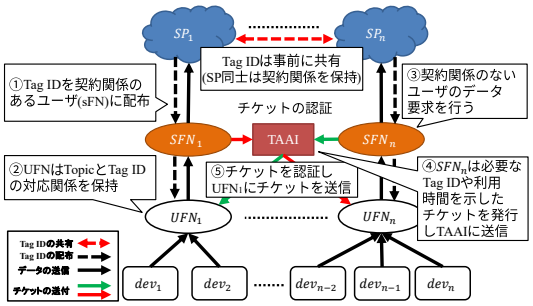


図 8 Ticket-based Access Control

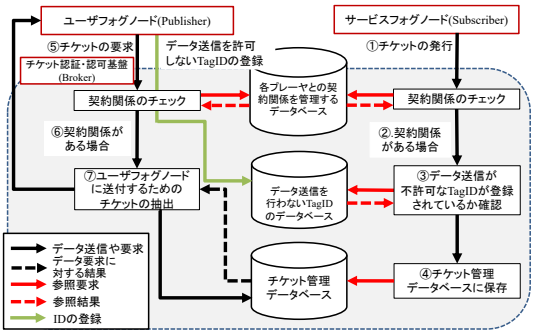


図 9 TAAI 内の動作

本課題に対して，アクセス制御のためのトークンとしてチケットを用いる Ticket-based Access Control [9] を提案する．具体的な提案手法は 4 章で述べる．

4. Ticket-based Access Control

4.1 Ticket-based Access Control の動作

図 8 に Ticket-AC の動作について示す．まず，従来のサービスモデルに新たなプレーヤであるチケット認証・認可基盤運用者を追加する．このプレーヤはチケット認証・認可基盤 (Ticket Authentication and Authorization Infrastructure: TAAI) を設置し運用している. ユーザとサービスプロバイダは，チケット認証・認可基盤運用者と契約関係を保持する. チケット認証・認可基盤は契約関係のあるユーザおよびサービスプロバイダのフォグノードと接続される．まず，Tag ID と Topic の対応関係は 2.2 節で述べたとおりである．次に，SP_1 と SP_n は契約関係があるため，Tag ID を事前に共有する．SP_n は契約関係のないユーザが所持する UFN_1 を介して IoT データを取得する場合，SFN_n を用いてチケットを生成する．チケットの構成要素は要求する IoT データの Tag ID や IoT データの利用時間，イベント情報などが考えられる. SFN_n は生成したチケットを，TAAI に Publish する．TAAI はチケットを認証

した後，自身が保持するチケット管理データベースにチケットを保存する．UFN_1 よりチケットの Subscribe があった場合，チケットを UFN_1 に Publish し，UFN_1 はチケットの内容に基づき，SFN_n に直接 IoT データを Publish する．

図 9 に TAAI 内の動作を示す．TAAI はユーザやサービスプロバイダとの契約関係を管理するデータベース(以下，クライアント管理データベース)，IoT データ送信を行わない Tag ID のデータベース (以下，Tag ID 管理データベース)，チケットを保存するデータベース(以下，チケット管理データベース) を保持する．

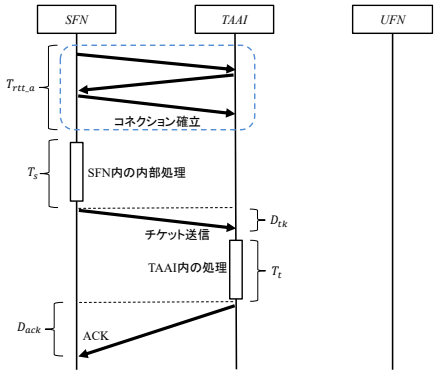
サービスフォグノードからチケットが送付された際，契約関係のチェックのためサービスプロバイダを指し示す ID とクライアント管理データベースが照合される．契約関係がある場合，IoT データ送信が不許可である Tag ID が登録されているかについて Tag ID 管理データベースを参照する．登録されている場合，チケットはチケット管理データベースに保存される．ユーザフォグノードよりチケットの要求があった場合，ユーザを指し示す ID とクライアント管理データベースが照合される．契約関係がある場合，ユーザフォグノード宛てのチケットがチケット管理データベースより抽出され，ユーザフォグノードに送付される．

4.2 Ticket-based Access Control のモデル化

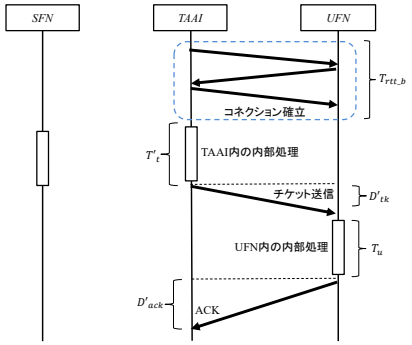
Ticket-AC の処理遅延時間は各ノードの処理遅延時間およびチケットの伝送遅延，ノード間の往復遅延をもとに算出する [10]．図 10 に TAAI のシーケンス図を示す．図 10 (a) に示す各パラメータについて述べる．T_{rtt_a}は SFN と TAAI 間の往復遅延である．TCP の 3-way handshaking によるコネクション確立までの時間にあたる．T_sは SFN 内の内部処理遅延であり，D_{tk}はチケット送信遅延である．チケット伝送遅延はチケットのデータサイズと帯域によって変化する．T_tは TAAI の内部処理遅延であり，D_{ack}は TAAI から SFN に送信する ACK の送信遅延である．ACK パケットのサイズと帯域によって変化する．次に，図 10 (b) に示す各パラメータについて述べる．T_{rtt_b}は TAAI と UFN 間の通信遅延である．T'_tは TAAI から UFN にチケットを送信する場合の TAAI 内の内部処理遅延である．D'_{tk}は TAAI から UFN にチケットを送信するときのチケット送信遅延である．T_uは UFN 内の内部処理遅延であり，D'_{ack}は UFN から TAAI に送信する ACK の送信遅延である．

各パラメータを用いた簡易的な処理遅延時間モデルが以下式 (5), (6)となる．PD_aが SFN と TAAI 間の処理遅延時間，PD_bが TAAI と UFN 間の処理遅延時間である．

$$PD_a = T_{rtt\_a} + T_s + D_{tk} + T_t + D_{ack} \quad (5)$$



(a) SFN から TAAI 間のシーケンス



(b) TAAI から SFN 間シーケンス

図 10 TAAI のシーケンス図

$$PD_b = T_{rtt_b} + T'_t + D'_{tk} + T_u + D'_{ack} \quad (6)$$

また，T_tおよびT'_tは各データベースの処理遅延時間の総和T_D，T'_Dと TAAI の平均処理率$1/\mu_t$を用いて以下式 (7) および (8) となる．

$$T_t = T_D + \mu_t \quad (7)$$

$$T'_t = T'_D + \mu_t \quad (8)$$

5. 評価

5.1 TAAI 内データベースの遅延時間測定

本提案方式は，シミュレーションで評価する．TAAI 内で発生する処理遅延時間はデータベースの大きさに依存することが想定される．そこで，実機に RDBS および KVS をインストールし，ダミーデータを登録する．そしてベンチマークにより，処理遅延時間を測定する．各データベースの処理遅延時間の総和をT_DおよびT'_Dとする．

クライアント管理データベースおよび Tag ID 管理データベースのベンチマークに MySQL version 14.14 Distribution 5.7.35 [11] を用いる．チケット管理データベースのベンチマークに Redis 6.2.5 [12] を用いる．

クライアント管理データベースではテストデータを 1 万行ランダムに登録する．同時接続数 (スレッド数) を 1, 2, 4, ..., 128 と変化させ，60 秒間データベースサーバに負荷をかける．I/O 性能やデータ探索に掛かった総合時間を処理遅延時間とする．

Tag ID 管理データベースではテストデータを 100 万

行ランダムに登録する．データベースサーバに負荷をかける方法はクライアント管理データベースのときと同様である．

チケット管理データベースではデータサイズは 1000, 2000, 4000, ..., 32000 [byte] と変化させる．チケットのリクエスト数（チケット登録/要求）は 20 万件で固定し，クライアント数は 1, 2, 4, ..., 128 と変化させる．

ベンチマークで用いた機器とソフトウェアを表 2 に示す．各ベンチマークはデータベースサーバと同一の機器で実行している．

クライアント管理データベースの処理遅延時間の最小値は 50 [msec] から 142 [msec] であった．最大値は 183 [msec] から 951 [msec] であった．Tag ID 管理データベースの処理遅延時間の最小値は 56 [msec] から 325 [msec] であった．最大値は 333 [msec] から 3881 [msec] であった．チケット管理データベースのリクエスト処理数はチケットサイズ 4000 [byte] まで最大で約 120000 [Requests/sec] であった．この逆数 0.0083 [msec] をシミュレーションで用いる．

上記のベンチマーク結果は，用いるソフトウェアや機器によって大きく変化する．あくまでも参考数値であり，数値の上昇傾向がシミュレーションでは重要となる．

各ベンチマーク結果をシミュレーションでは指数分布の範囲で変化させる．

5.2 シミュレーション評価

本研究における要件 3 の評価のため，Ticket-AC 適用による各ノード間の影響を確認する必要があるためシミュレーションにより評価を行う．シミュレーションパラメータを表 3 に示す．シミュレーションは，待ち行列理論を用いる．シミュレーションモデルは M/M/c/K とする．各ノードのキュー長は 1000 とする．キュー長を十分大きくし，チケットの破棄を考慮しない．TAAI は 1 台とし SFN と UFN のノード数は 1, 2, 4, ..., 128 と変化させる．チケットの伝送遅延は約 20.0 [msec] とする．TAAI の平均処理率は 1.0 [1/msec] とし，SFN と UFN の平均処理率は 0.5 [1/msec] とする．伝送遅延時間や平均処理率は，指数分布の範囲で変化する．伝送遅延や平均処理率の逆数である平均処理時間，データベースの処理遅延時間をパラメータとし (5) 式および (6) 式を用いてノード間のトータル処理遅延時間を算出する．各ノードのチャネル数はパケットを同時に処理できる数を示し，本シミュレーションでは，チケットを同時に処理できる UFN と TAAI 内の窓口数を示す．SFN はチケットを Publish するのみの動作となるので，チャネル数は 1 と固定する．UFN と TAAI のチャネル数を 1, 5, 10 と変化させる．シミュレーションプログラムは C++で実装する．

表 2 使用機器とソフトウェア

CPU	Intel Core i7-6700 (Core: 4, Thread: 8)
RAM	7.7 GiB
OS	Ubuntu 18.04.6 LTS [13]
Storage	HDD: 500 GB
RDB	MySQL version 14.14 Distribution 5.7.35
KVS	Redis 6.2.5
Benchmark	Sysbench version 1.0.20 [14] redis-benchmark [15]

表 3 シミュレーションパラメータ

Parameter	Value
Simulation model	*M/M/c/K*
Queue length (K)	1000
Number of TAAI	1
Number of SFN	2, 4, ..., 128
Number of UFN	2, 4, ..., 128
Number of channels (c)	1, 5, 10
Average interval for sending a ticket	20.0 [msec]
Average processing rate $(1/\mu_t)$	1.0 [1/msec]
Average processing rate $(1/T_s$ *and* $1/T_u)$	0.5 [1/msec]
Simulation time	3600.0 [s]
Simulation trails	30

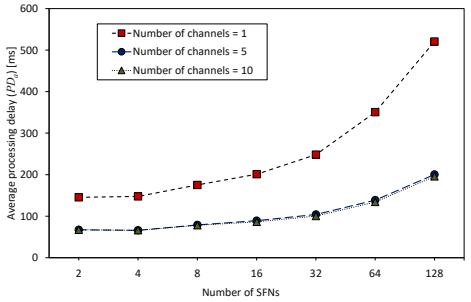


図 11 SFN から TAAI 間の処理遅延時間

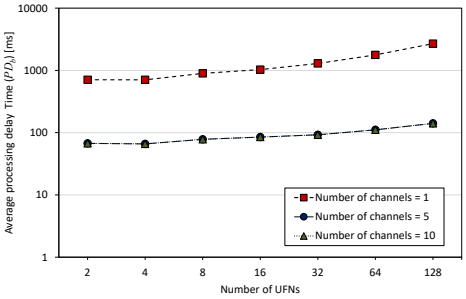


図 12 TAAI から UFN 間の処理遅延時間

図 11 に SFN から TAAI 間の処理遅延時間のシミュレーション結果を示す．図 11 より TAAI のチャネル数が 1 のとき，平均処理遅延時間は最大約 520 [msec] であった．次にチャネル数が 5 のとき平均処理遅延時間

は最大約 195 [msec] であった．平均処理遅延時間はチャネル数が 1 のときと比べ，最大約 62%削減された．これは TAAI 内で同時に処理できるチャネルが多くなったため，チケットの待ちが削減されたためである．次にチャネル数を 10 にしたとき，チャネル数 5 のときとほぼ変わらない結果となった．これはチャネル数 5 で十分にチケットを処理できることを示している．

図 12 に TAAI から UFN 間の平均処理遅延時間のシミュレーション結果を示す．またこのシミュレーションでは，複数存在する UFN のうち 1 台 (ノード ID = 1) の UFN に着目して平均処理遅延時間を算出している．また縦軸は対数軸に設定する．

図 12 より TAAI のチャネル数が 1 かつ UFN のチャネル数が 1 のとき，平均処理遅延時間最大約 2678 [msec] であった. UFN のチャネル数が 1 であるためチケットの待ちが発生し，UFN 内で処理遅延が発生したためである．また，UFN の利用率も非常に大きくなっていることが要因として考えられる．次に TAAI と UFN のチャネル数が 5 のとき平均処理遅延時間は最大約 140 [msec] であった．チャネル数が増えたことにより，チケットの待ちが削減されたためである. TAAI と UFN のチャネル数が 10 のとき，チャネル数 5 のときとほぼ結果は変わらなかった．チャネル数が 5 で十分な性能を発揮していることを示す．

チャネル数は実機のプロセッサ性能に依存する．そのため，チャネル数を増やすことはプロセッサ性能を向上させることに相当するため，コストの面でトレードオフとなる．本シミュレーションでは，設定したパラメータにおいて，ノードのチャネル数の上限値を定めることができたことが大きな貢献点と言える．また本研究における要件 2 であるスケーラビリティの担保の観点から，チャネル数を上昇させることで，処理遅延時間の短縮が見込めることから，要件 2 を満たしていると言える．

5.3 セキュリティリスクの洗い出しと対処法

Ticket-AC を適用した IoT データ流通によるセキュリティリスクの分析を行う必要がある．サービスプロバイダからユーザ，ユーザからサービスプロバイダの視点から分析する．

サービスプロバイダ視点では，クラウドサーバから SFN，SFN から TAAI，TAAI から UFN の順番で行う．

ユーザ視点では，UFN から TAAI，UFN から SFN，SFN からクラウドの順にセキュリティリスクを分析する．

以下に，セキュリティリスクの分析を示す．

セキュリティリスクの分析

(i) サービスプロバイダからユーザ

(1) クラウドサーバから SFN: クラウドサーバは SFN に対して IoT データを要求するコマンドを送信する. クラウドサーバから SFN の通信回線が盗聴された場合, SFN の不正利用により, IoT サービスを正常に運用できなくなる可能性がある．
損なわれるもの: [機密性] [可用性]

(2) SFN から TAAI: SFN はチケットを生成し，TAAI に Publish する．チケットの内容を読み取れてしまう場合, Tag ID を読み取られてしまい，悪意のある第三者によってチケットが生成されてしまう可能性がある．また，チケットの中身を別の内容に書き換えられてしまえば，ユーザに対する信用性を失う可能性がある．チケットの数が多い場合, TAAI 付近の帯域ひっ迫およびキューイング遅延増加に繋がる．
損なわれるもの: [機密性] [完全性] [可用性]

(3) TAAI から UFN: TAAI より UFN に対して，チケットを Publish する．チケットの内容を読み取れてしまう場合，悪意のある第三者によってチケットが生成されてしまう可能性がある．また，チケットの中身を別の内容に書き換えられてしまえば，ユーザに対する信用性を失う可能性がある．チケットの数が多い場合，ユーザフォグノード付近の帯域ひっ迫およびキューイング遅延増加に繋がる．
損なわれるもの: [機密性] [完全性] [可用性]

(ii) ユーザからサービスプロバイダ

(1) UFN から TAAI: UFN から TAAI に対してチケットの Subscribe を行う. UFN および TAAI 間に悪意のある第三者の盗聴の可能性がある．Subscribe の内容の書き換えを行い, Subscribe に対する結果を書き換えられてしまう可能性がある．
損なわれるもの: [完全性]

(2) UFN から SFN: UFN は，SFN に IoT データを Publish する．IoT データが通信経路内で，改ざんされた場合，ユーザが利用する IoT サービスの品質が低下する可能性がある．
損なわれるもの: [機密性] [可用性] [完全性]

また上記のセキュリティリスクに対する対処法を以下に示す．

セキュリティに対する対処法

(i) サービスプロバイダからユーザ

(1) クラウドサーバから SFN: 通信経路における盗聴の脅威に対して，IPsec VPN などの既存技

術 [16] で対応可能である.

(2) SFN から TAAI: 既存の暗号化方式適用によってチケットの内容の暗号化を行う. またチャレンジレスポンス認証と併用することによって対処可能である. また, TAAI のプロセッサ性能をスケールすることにより性能低下を防ぐ.

(3) TAAI から UFN: 既存の暗号化方式適用によってチケットの内容の暗号化を行う. またチャレンジレスポンス認証 [17] と併用することによって対処可能である. また, ユーザフォグノードのプロセッサ性能をスケールすることにより性能低下を防ぐ.

(ii) ユーザからサービスプロバイダ

(1) ユーザフォグノードから TAAI: UFN と TAAI の通信路に IPsec VPN を適用することにより, 盗聴や Subscribe 改ざんの脅威に対処可能である [16].

(2) UFN から SFN: SFN と UFN 通信経路に IPsec VPN を適用することで, UFN と SFN 間の盗聴やデータ改ざんの脅威に対処可能である.

上記の点から, 本研究における要件 4 を満たすための見通しを得られた.

6. まとめ

本報告では, IoT データ流通方式に適用した Ticket-AC の性能をシミュレーションにより評価した. TAAI 内のチャネル数を増やすことにより, SFN から TAAI 間の処理遅延時間は最大約 62%, TAAI から UFN 間の処理遅延時間は約 95%削減されることを示した.

また, セキュリティリスクに対する対処法にかんしても既存の技術で対処可能である見通しを立てた.

上記の点から以下を満たした, もしくは見通しを立てた.

[要件 1] IoT データ流通による新たなサービスモデルが必要であること.

[要件 2] IoT データ流通時におけるプライバシに関わる IoT データのアクセス制御があること.

[要件 3] IoT データ流通方式におけるスケーラビリティが担保されていること.

[要件 4] サービスモデル上のセキュリティリスクに対する対処がなされていること.

今後は実機実装を行い, 想定環境に基づくセキュリティ評価を行う必要がある.

謝辞

本研究は科研費 (18K11276) の助成を受けたものです.

参 考 文 献

[1] M Yannuzzi, R. Irons-Mclean, F. van Lingen, S. Raghav, A. Som-araju, C. Byers, T. Zhang, A. Jain, J. Curado, D. Carrera, O. Trullols and S. Alonso, "*Toward a converged OpenFog and ETSI MANO architecture*," 2017 IEEE Fog World Congress (FWC), Nov. 2017.

[2] M. Yoshii, R. Banno, O. Mizuno, "*Evaluation of table-based access control in IoT data distribution method using fog computing*", IEICE Communications Express 10(10) 822-827.
DOI: 10.1587/comex.2021xbl0134.

[3] M. Yoshii, R. Banno, O. Mizuno, "*Performance Evaluation of Table-based Access Control List Applied to IoT Data Distribution Method using Fog Computing*," Proceedings of the IEICE International Conference on Emerging Technologies for Communications (ICETC), E1-1, Dec. 2, 2020.
DOI: 10.34385/proc.63.E1-1.

[4] P. T. Eugster, P. A. Felber, R. Guerraoui, and A.-M. Kermarrec, "*The Many Faces of Publish/Subscribe*", ACM Computing Surveys, 2003.

[5] MQTT, https://mqtt.org/ (accessed Jan. 6, 2022)

[6] R. Banno, J. Sun, S. Takeuchi, K. Shudo, "Interworking Layer of Distributed MQTT Brokers", IEICE Transactions on Information and Systems, Vol. E102-D, No. 12, pp. 2281-2294 (Dec. 2019)

[7] J. Kreps, N. Narkhede, J. Rao, "*Kafka: a Distributed Messaging System for Log Processing*", http://notes.stephenholiday.com/Kafka.pdf, 2011.

[8] AMQP, https://www.amqp.org/ (accessed Jan. 10th, 2022)

[9] M. Yoshii, R. Banno, O. Mizuno, "*Evaluation of Ticket-based Access Control Method Applied to IoT Data Distribution*", IEICE Communications Express (accepted),
https://doi.org/10.1587/comex.2021XBL0214

[10] 関根 響, 金井 謙治, 甲藤 二郎, "IoT ネットワーク環境下における IoT 向け通信プロトコルの遅延特性評価", 信学技報, NS2018-213 (Mar .2019)

[11] MySQL, https://www.mysql.com/jp/.
(accessed Jan. 6th, 2022)

[12] Redis, https://redis.io/. (accessed Jan. 6th, 2022)

[13] Ubuntu, https://releases.ubuntu.com/18.04/.
(accessed Jan. 6th, 2022)

[14] Sysbench, https://github.com/akopytov/sysben-ch/releases/tag/1.0.20. (accessed Jan. 6th, 2022)

[15] Redis-benchmark, https://redis.io/topics/bench-marks. (accessed Jan. 6th, 2022)

[16] 水野修, "ビルシステムのサイバーセキュリティ対策", 電気設備学会誌, Vol. 39, No. 6, pp. 314-317 (Jun. 2019)
https://doi.org/10.14936/ieiej.39.314

[17] rfc1994, https://datatracker.ietf.org/doc/html/rfc1994 (Jan. 10th, 2022)