P4 を用いた MQTT-SN アクセラレーションの実装

Implementation of MQTT-SN Acceleration using P4

大澤 昂生 †　日比 智也 ‡　坂野 遼平 †
Koki Osawa　Tomoya Hibi　Ryohei Banno

† 工学院大学 情報学部 情報通信工学科
Kogakuin University, Tokyo, Japan

‡ NTT 未来ねっと研究所
NTT Network Innovation Laboratories, Kanagawa, Japan

1．はじめに

IoT (Internet of Things) デバイスが世界的に急速に普及している．総務省の情報通信統計では 2018 年に全世界で 307.1 億台の IoT デバイスが稼働していると統計づけられ，2021 年には 400 億台を超えると見込まれている[1]．IoT デバイスの通信プロトコルに MQTT (MQ Telemetry Transport) [2]が存在する．MQTT は Publish/Subscribe 型の通信プロトコルである．この方式ではセンサデータ等のデータを提供するクライアント (Publisher) が Broker と呼ばれるサーバにデータを出版 (Publish) し，Broker は対象データを購読 (Subscribe) しているクライアント (Subscriber) にデータの転送を行う．よって，IoT システムの規模が大きくなればなるほど Broker への負荷集中が懸念される．

本研究では，MQTT の派生プロトコルである MQTT-SN を対象とし，Broker の一部機能を Broker 直前のネットワーク内にて行うことで，Broker の使用リソースの削減と周辺ネットワークでのトラフィックの削減を行うことを目的とする．

2．提案手法

本研究では，ネットワーク内に P4 言語 [3] 対応のスイッチを置き，そのスイッチに Broker の一部機能を実装する．この Broker の一部動作を実装した P4 スイッチをアクセラレータと呼称する．

提案手法の概要を図 1 に示す．アクセラレータでは，3 つの動作パターンを想定している．最初に，Subscribe メッセージを受け取った場合である．この場合，アクセラレータは P4 コントローラと連携して IP アドレスや Topic 名などの情報の記録・保持を行う．次に指定条件に合致する Publish メッセージを受け取った場合である．この指定条件とは，QoS レベルが 0 であり Retain フラグの立っていない Publish メッセージである．この場合，Subscribe メッセージ受信時に記録したデータベースを基に Broker を介さずに Subscriber に対し Publish メッセージの配信を行う．最後にこれらの条件以外のものに関してである．この場合は，アクセラレータにて制御を加えてしまうと Broker の動作に悪影響を与える場合もあると考えられる為，従来通りルーティングを行う．

本研究では，MQTT-SN [4]を対象として実装し評価を行った．MQTT-SN は，TCP 上で動作する通常の MQTT とは異なりトランスポート層に UDP を用いる．P4 を用いる方式の有効性評価を行うため，トランスポート層の処理がよりシンプルである MQTT-SN を実装評価にて用いた．アクセラレータの実装に関しては，Subscribe 部はスタティックな設定を用いて，それ以外の部分は P4 にて実装した．

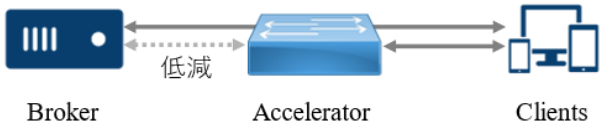


図 1　提案手法概要

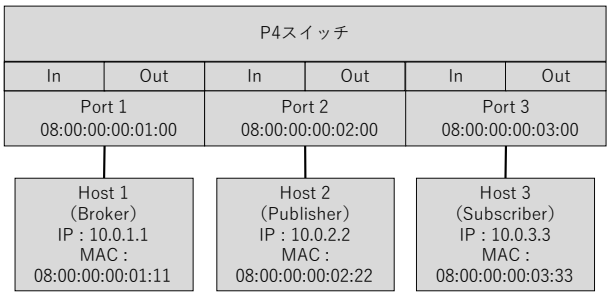


図 2　実験トポロジ

3．評価

3-1. 評価方法

評価はアクセラレータ機能を使う場合，アクセラレータ機能を使わない場合での各ホストの送受信パケット数を測定し行った．また，この評価には図 2 に示すトポロジを使用した．具体的な評価方法として，MQTT-SN にて Publisher から Broker に対し Publish メッセージを 100 回送信するときの各ホストの送受信パケットを WireShark にてキャプチャした．本評価での Publish メッセージはアクセラレータの条件に当てはまる，QoS レベルが 0 かつ Retain フラグの立っていないものを送信した．なお，Publisher が Publish メッセージを送信し始める前に Subscriber は Subscribe メッセージを Broker に宛て送信しており，Subscribe の動作が完全に終了した後に Publisher は Publish メッセージを送信している．

評価環境は Windows の PC 上にインストールされた VirtualBox にて，VM として動作させている Ubuntu16.04.7 にて行った．VM は P4 言語公式 GitHub [5]にて提供されているものを利用した．この VM では BMv2 という P4 ソフトウェアスイッチが使用可能なため，この VM 上にて提案手法で述べた P4 プログラムを実装した．同 VM 上の Mininet にて図 2 のトポロジを設定し評価を行った．Broker は RSMB (Really Small Message Broker)[6]という MQTT-SN に対応した Broker を利用し，クライアントは GitHub にて公開されている mqtt-sn-tools[7]を利用した．使用した環境について，表 1 に VM を動かしている Windows PC の環境を，表 2 に VM 周りの環境設定および使用ソフトウェアを，それぞれ示す．

表 1　Windows PC 環境

項目	内容
CPU	Intel Core i5-10400 6core/12thread @2.90GHz
RAM	32GB
OS	Windows 10 Pro Education 64bit
VM Software	Oracle VirtualBox Version 6.1.12 r139181 (Qt5.6.2)

表 2　VM 環境

項目	内容
割り当てCPU	2 Core
割り当てRAM	2GB
OS	Ubuntu 16.04.7 LTS
Network Analyzer	Wireshark 2.6.10
Broker Software	Mosquitto RSMB (Really Small Message Broker) Version 1.3.0.2
Client Software	mqtt-sn-tools Vresion 0.0.6
P4 Software Switch	BMv2 Version 1.13.0

3-2. 評価結果

評価結果を表 3 に示す．表 3 の従来手法のデータをグラフ化したものを図 3 (a)に，提案手法のデータをグラフ化したものを図 3 (b)に示す．表 3 から，Broker に流れる パケット数を見ると提案手法が既存手法に比べ 100 パケット分少ないことが分かる．この 100 パケットというのは Publish メッセージのパケット数と等しい．提案手法，および既存手法の Broker に流れるパケット数には MQTT-SN のクライアント接続にかかわるパケットも含まれている．MQTT-SN のプロトコル仕様より，本実験シナリオにおける接続にかかわるパケット数は 6 パケット分である．評価結果より，提案手法の Broker が受信したパケットおよび送信したパケットは 6 パケットずつである．ここから，提案手法はクライアントとの接続のための通信以外していないということが分かる．つまり，提案手法にて Broker は Publish パケットを一度も受信していなく，受信していないがために Publish にかかわる処理や送信も行っていないということが分かる．さらに，Publisher 及び Subscriber の送受信パケットは提案手法及び既存手法両方とも同じパケット数を送受信している．ここから，Broker を介さずとも MQTT-SN の動作が適切に行われていることが考察できる．

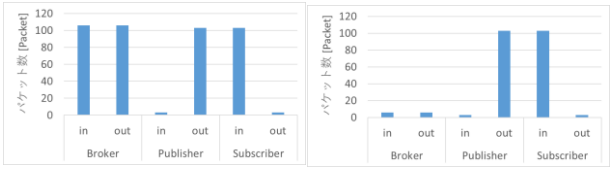
今回の評価では 100 個のアクセラレータ対象 Publish メッセージを送信しているため，提案手法と既存手法の差が受信側，送信側両方とも 100 パケットとなっている．アクセラレータに十分な性能があると仮定すると 1 つの Publish メッセージを送るだけであったらこの差は 1 になり，逆に膨大な数の Publish メッセージを送信するとこの差は膨大なものとなることが考察できる．このことから，Publish メッセージの数に比例しメッセージ数の削減が期待できる．

4.　おわりに

本研究では，IoT システムに多く使われる MQTT という通信プロトコルにおいて Broker と呼ばれるサーバの機器リソースの削減，および周辺ネットワークでのトラフィック削減を目的とした．この目的に対し本研究では P4 を用いて Broker の処理の一部をネットワーク内で行う手法を提案した．本研究では MQTT-SN を対象とし，実際に P4 言語を用いて P4 スイッチに Broker の一部処理を行わせるよう実装および，評価を行った．

表 3　通過パケット数測定結果

	P4スイッチ					
	Broker		Publisher		Subscriber	
	in	out	in	out	in	out
従来手法	106	106	3	103	103	3
提案手法	6	6	3	103	103	3



(a)従来手法　　(b)提案手法

図 3　評価結果グラフ

評価結果から，提案手法では本来 Broker で処理されるパケットがアクセラレータにて処理されていることを確認した．さらに，Publisher および Subscriber も従来方式と同じ挙動をしていることも確認した．このことから，本研究成果であるアクセラレータは適切な動作をしていることが確認できた．最後に，研究目的である Broker の使用リソースの削減，周辺ネットワークにおけるトラフィックの削減に関しては，Broker の処理をアクセラレータにて適切に処理できているということから，本提案方式は有効な手段であることを確認した．

謝辞　本研究の一部は，JSPS 科研費 19K20253 の支援を受けて行われたものである．

参考文献

[1] 総務省, “令和元年版情報通信白書”, https://www.soumu.go.jp/johotsusintokei/whitepaper/ja/r01/html/nd112120.html, 閲覧日 2020/05/25

[2] OASIS, “MQTT Version 3.1.1”, http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.html, 閲覧日 2020/12/23

[3] Pat Bosshart, Dan Daly, Glen Gibb, Martin Izzard, Nick McKeown, Jennifer Rexford, Cole Schlesinger, Dan Talayco, Amin Vahdat, George Varghes, David Walker, “P4: programming protocol-independent packet processors”, ACM SIGCOMM Computer Communication Review, July 2014

[4] mqtt.org, “MQTT-SNv1.2”, https://www.oasis-open.org/committees/download.php/66091/MQTT-SN_spec_v1.2.pdf, 閲覧日 2020/12/23

[5] github.com/p4lang, “P4 Tutrial”, https://github.com/p4lang/tutorials 閲覧日 2020/12/23

[6] github.com/eclipse, “mosquito rsmb” https://github.com/eclipse/mosquitto.rsmb 閲覧日 2020/12/23

[7] github.com/njh, “mqtt-sn-tools” https://github.com/njh/mqtt-sn-tools 閲覧日 2020/12/23