oneM2M 規格のオープンソースソフトウェアと自作ソフトウェアの比較

大西亮吉[1] 笹原将章[†1] 佐藤一馬[1]

概要：車両に搭載された多様なセンサのセンシング情報がセンタサーバに集められ，故障診断や仕様改善，経路案内等に役立てられている．情報収集のための通信プロトコルとして oneM2M 規格に着目し，Release 1 の直後から自作してきたソフトウェア MDDC と，後発のオープンソースソフトウェア OM2M の機能面・性能面の比較を行った．機能面では，OM2M の方がより新しい oneM2M 規格を網羅的に実装していたが，車載機には必須となるモビリティを管理するような機能は確認できなかった．性能面では，MDDC の方が OM2M よりも接続時間が短く，伝送時間は長い様子が確認できた．更にそれぞれのベースとなる通信プロトコルとして HTTP と CoAP の比較も行い，CoAP はメッセージサイズが 1kB を超えると，伝送時間が長くなる様子が確認できた．これらの結果の考察を行い，今後の開発の方向性について述べる．

Comparison between Open Source Software and Lab Prototype on oneM2M Standards

RYOKICHI ONISHI[1] MASAAKI SASAHARA[†1] KAZUMA SATO[1]

1. 背景

車両には多様なセンサが搭載され，部品（タイヤの空気圧やエンジン，電池，電源など）や運行（測位，進行方向など）に関する情報を生成する．情報はセルラー等により固定網側のセンタサーバに集められ, 故障診断や仕様改善, 経路案内等に役立てられてきた．更には，高精度な地図の構築や車両以外の情報，例えば搭乗者の健康状態や道路インフラの劣化具合，農作物の生育状況，大気汚染の程度の把握なども期待される．これらの情報を収集する方策について, (i) 出版購読モデルによる通信量の削減やサーバ負荷の軽減, (ii) oneM2M 規格準拠による相互接続性の確保とモバイルネットワークの機能利用について，筆者らは研究してきた[1].

1.1 実現に向けた課題

研究を始めた 2015 年初頭は，oneM2M 規格の初版（Release 1）が発行された時期であり，広く一般に利用可能なソフトウェアは存在しなかった．そのため，規格を参照して必要な部分だけソフトウェアを実装し，研究に利用してきた．それから 2 年以上が経過し，様々なオープンソースソフトウェアが登場してきた．自作ソフトウェアの使用を継続するべきか，オープンソースソフトウェアに切り替えるべきかが課題となる．

1 （株）トヨタ IT 開発センター
TOYOTA InfoTechnology Center, Co., Ltd.
†1 現在，（株）デンソー
DENSO Corporation

1.2 課題解決の方策

一般にオープンソースソフトウェアの開発は，多くの技術者の貢献によって進められており，自作ソフトウェアよりも継続的な発展が見込まれ，結果として機能の充実や動作の安定が期待できる．一方で，自作ソフトウェアにも固有の利点があるものと考えられ，オープンソースソフトウェアへ提案することも考えられる．本論文では，oneM2M 規格に関する自作ソフトウェアとオープンソースソフトウェアの機能面・性能面の比較を行い，今後の開発の方向性について考察する．

2. 自作ソフトウェア MDDC

筆者らは，自作したソフトウェアを MDDC (Mobile Device Data Collection)と呼ぶ．モバイルデバイスの情報を収集するシステムという意味を持つ．センタサーバを IN (Infrastructure Node)，車載機を MN (Middle Node)，車載センサを ASN (Application Service Node)として oneM2M 規格を参照して実装した. IN, MN, ASN は同一の MDDC を備え，アプリケーションへのインターフェースとなる AE (Application Entity) と共通サービスを提供する CSE (Common Service Entity)を含む．ノード内で AE は CSE へ接続し, ノード間は CSE 同士で接続される. これは oneM2M の一般的な機能構成であり，図 1 のようになる．[1]では MN 間接続による車車間通信の評価を行ったが，MN を固定網側のエッジサーバ，ASN を車載機とすることで，近年 ETSI 等で協議されているようなエッジコンピューティン

グ MEC(Mobile/Multi-access Edge Computing)の評価を行うことも可能と考えられる．

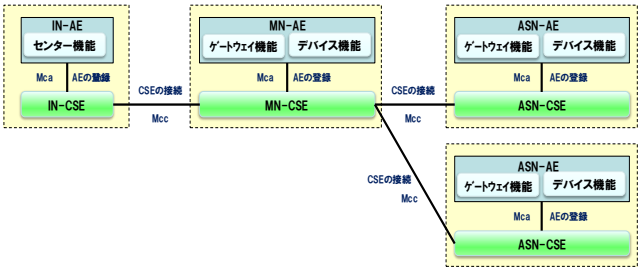

図 1　oneM2M の機能構成

MDDC において，AE は Python 3.4.0，CSE は Ruby 2.2.0 のスクリプト言語で実装され，それぞれデータベース（PostgreSQL 9.3.9）内の専用に割り当てられた領域を利用する．oneM2M は通信プロトコルの規格であり実装言語を規定しないため，あえて異なる言語で実装してみた．ウェブサーバ（Nginx 1.4.6）は AE や CSE のデータベースにアクセスして，ブラウザからの状態確認や設定変更を可能とする．AE-CSE 間，CSE-CSE 間の下層の通信プロトコルは HTTP，または CoAP から選択できる．CoAP の Ruby 実装は[2]，Python 実装は[3]を利用した．CSE はサーバ機能を備え，AE のもつクライアント機能と接続する．CSE はクライアント機能も保有し，他の CSE と接続する場合に使用する．接続はクライアントからサーバへの要求によって開始することができる．そして，サーバからの応答を保留する long-polling によって，疑似的にサーバプッシュを実現する．MDDC のソフトウェアの構成を図 2 に示す．

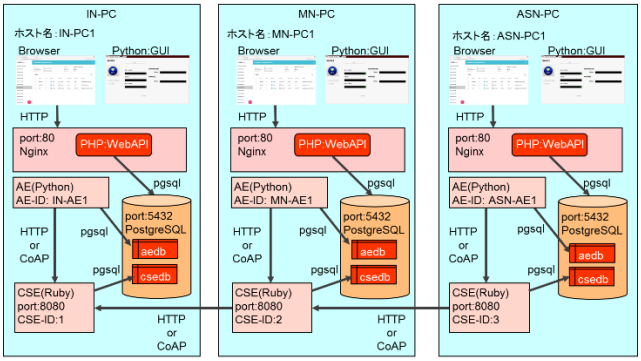

図 2　MDDC のソフトウェア構成

3. オープンソースソフトウェア OM2M

oneM2M 規格を実装したオープンソースソフトウェアは，[4]によれば Eclipse OM2M, OCEAN Mobius, IoTDM, OASIS SI とされる．本論文では 2016 年 7 月にリリースされた OM2M-1.0.0 を取り上げて調査を行った．OM2M は Eclipse Foundation のプロジェクトで，ライセンスは Eclipse Public License 1.0 で定義される．このライセンスは，ライセンスと著作権の表示，ソースコードの開示が必須事項であり，商用利用，配布，修正，個人利用，サブライセンス・特許の利用許可が認められる．作者に責任を求めることは禁止事項とされる．

OM2M では，CSE，AE 共にサーバ機能とクライアント機能を併せ持ち，双方向の通信が可能になっている．各機能は plugin というモジュール形式で構成されており，様々な module を組み合わせて機能を実現できる．下層の通信プロトコルは HTTP と CoAP を選択することができる．実装言語は Java であり，サーバは Servlet によって実現される．oneM2M のリソースはデータベース（OODB）で管理される．OM2M のソフトウェア構成を図 3 に示す．

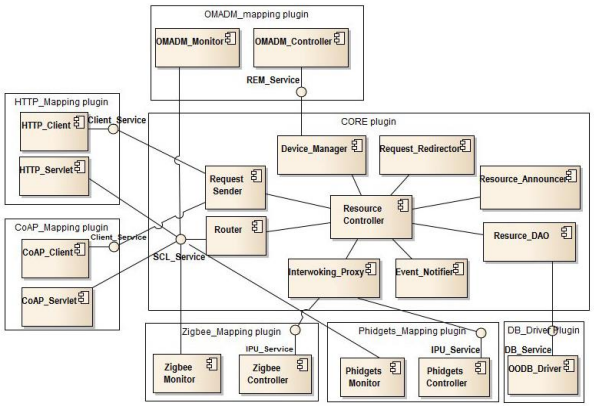

図 3　OM2M のソフトウェア構成

4. MDDC と OM2M の機能比較

MDDC と OM2M における oneM2M 規格の実装バージョンは，表 1 の通りである．OM2M の方がより新しい規格を参照していることが分かり，MDDC のように単独で開発するよりもペースが早いことが分かる．

表 1　OM2M と MDDC の実装バージョンの比較

	MDDC	OM2M
oneM2M Release	0.9 1.0.1(一部)	2
Functional Architecture	1.6.1 (2015/01)	2.10.0 (2016/08)
HTTP Protocol Binding	1.0.1 (2015/01)	2.6.1 (2016/08)
CoAP Protocol Bindign	1.0.1 (2015/01)	1.3.2 (2016/03)

oneM2M の共通サービスには，アプリケーション管理，セキュリティ，セッション管理，グループ管理など 12 項目の機能が存在する．各機能の実装状況は表 2 の通りであった．このうち OM2M では 12 項目すべてにおいて，おおむね実装されているのに対し，MDDC は 5 項目に留まる．4 項目は同等の機能を独自に実装した．

メッセージフォーマットのうち，リクエストメッセージについて調べた．実装状況は表 3 の通りである．21 項目のパラメータがあるが，OM2M はすべて実装されているのに対し，MDDC は 12 項目に留まる．

これは MDDC が研究対象となるユースケースに必要な

範囲のみ実装しているためであり，広く一般的な利用に供するには機能が不足していることが分かる．

表 2　共通サービスの実装状況

共通サービス	MDDC	OM2M
アプリケーション管理	×	○
セキュリティ	○	○
サービスのセッション管理	※	○
グループ管理	×	○
データ管理	○	○
通信管理･配布機能など	※	○
デバイス管理	○	○
デバイス発見	※	○
位置情報	※	○
デバイス登録	○	○
通知機能	○	○
ネットワークサービス連携	×	○

○:実装済
×:未実装
※:独自実装

表 3　リクエストメッセージの実装範囲

Primitive Parameter	短縮名	MDDC	OM2M
requestPrimitive	rqp	○	○
Operation	op	○	○
To	to	○	○
From	fr	○	○
Request Identifier	rqi	○	○
Resource Type	ty	○	○
Name	nm	○	○
Content	pc	○	○
Original Timestamp	ot	×	○
Request Expiration Timestamp	rqet	○	○
Result Expiration Timestamp	rset	×	○
Operation Execution Time	oet	×	○
Response Type	rt	×	○
Result Persistence	rp	×	○
Result Content	rcn	○	○
Event Category	ec	×	○
Delivery Aggregation	da	×	○
Group Resuest Identifier	gid	×	○
Filter Criteria	fc	×	○
Filter Usage	fu	○	○
Descovery Result Type	drt	○	○

4.1　デバイスのモビリティ管理機能

oneM2M 規格[5]の Annex B.6.1.1 ASN/MN-CSE Initiated Connectivity Establishment Procedure によれば，デバイス同士が接続した場合，下位 CSE (ASN/MN-CSE）が上位 CSE (IN-CSE)の<CSEBase>の直下に<remoteCSE>リソースを作成し，PoA 属性（Point of Access）に ID と IP アドレスを紐づけて登録することが規定されている．これにより，互いのアプリケーションを発見して通信することができる．端末の移動に伴い，IP アドレスが変わった場合は更新される．

MDDC はデバイスのモビリティを前提としているため，この機能は実装されているが，OM2M では確認できなかった．接続先のデバイスの IP アドレスは設定ファイル（config.ini）に以下のように予め記述されており，登録や更新を行う手段は確認できなかった．

org.eclipse.om2m.remoteCseAddress=10.9.5.17

Annex に記載された仕様であるため，実装が後回しになっている可能性がある．

5.　MDDC と OM2M の性能比較

続いて，MDDC と OM2M の通信性能を比較するために，接続時間と伝送時間をそれぞれ評価した．プロトコルは HTTP と CoAP の比較を行う．なお，圧縮や暗号化は行わない．評価機材の構成は，インフラ側ノード（IN）の PC と車両側ノード（MN）の PC をスイッチングハブで接続するシンプルなものである．ノードの移動は行わない．PC は NEC VersaPro VD-D を使用した．CPU は Intel Core i5-2520M@2.50GHz，メモリは 8GB，NIC は Intel 82579LM でギガビットイーサネット対応，OS は Ubuntu 14.04.3，IP アドレスは固定とする．スイッチングハブは，Corega CG-SW08GTX2W でギガビットイーサネット対応である．

IN と MN 間のネットワーク転送速度は，iperf で測定した結果，平均で 935Mbps であった．時刻同期は ntpupdate により実施した．Offset の値から平均値±3σ を計算すると，0.1±0.641[msec]であり，1 ミリ秒以下の精度である．

5.1　接続シーケンスの評価

5.1.1 MDDC の接続シーケンス

MDDC の接続は図 4 のように行われる．CSE と AE の接続では，CSE がサーバ，AE がクライアントとなる．CSE 間接続では，IN-CSE をサーバ，MN-CSE をクライアントとした．サーバはクライアントからの要求に対する応答を保留する long-polling によってメッセージの送信を行う．IN における AE と CSE の接続は予め終えているものとし，IN と MN の CSE 間接続（①）と，MN における AE と CSE の接続（②③④⑤）を測定した．

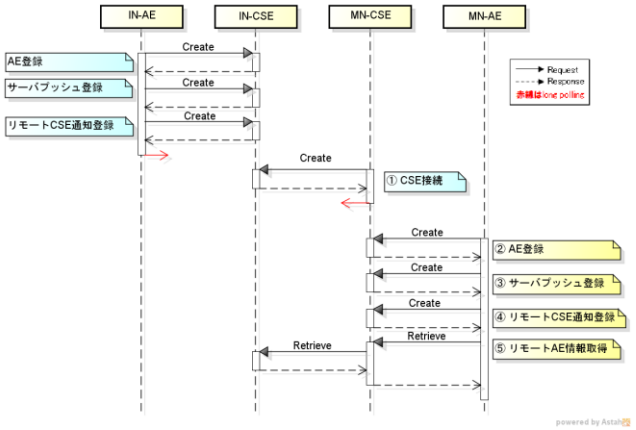

図 4　MDDC の接続シーケンス

5.1.2 OM2M の接続シーケンス

OM2M の接続は図 5 のように行われる．AE と CSE の双

方にサーバが立ち上がっており，直接送受信を行うことができる. IN における AE と CSE の接続は予め終えているものとし，IN と MN の CSE 間接続（①）と，MN における AE と CSE の接続（②⑤）を測定した.

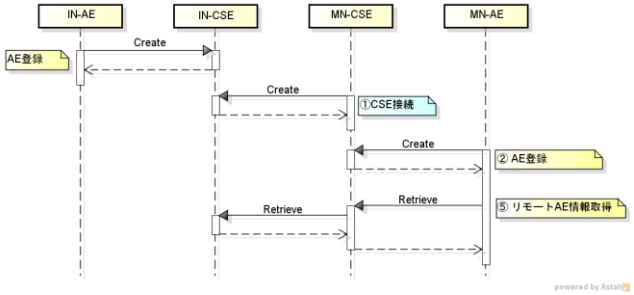

図 5　OM2M の接続シーケンス

5.1.3 接続シーケンスの評価結果

MDDC と OM2M の接続シーケンスの評価結果は，表 4 のようにまとめられる．10 回の試行を行い，各ステップの開始時刻から次のステップの開始時刻までの所要時間の平均値をミリ秒単位で示す．MDDC と OM2M の比較では，MDDC の方が多くのシーケンスを要するにも関わらず，OM2M の半分以下の時間で接続が完了した. MDDC はスクリプト言語で実装されておりプログラム起動の負荷が低い一方，MDDC は Java で実装されており，VM の起動や起動後に初めてのオブジェクトを利用する場合などのロード処理に時間がかかっていると思われる．HTTP と CoAP の比較では，CoAP の方が速いことが分かる.

表 4　接続シーケンスの所要時間

	MDDC		OM2M	
	HTTP	CoAP	HTTP	CoAP
①CSE登録	89	65	437	85
②AE登録	58	39	489	516
③サーバプッシュ登録	36	19	---	---
④リモートCSE通知登録	86	59	---	---
⑤リモートAE情報取得	153	95	241	20
合計時間	422	301	1167	620

単位：ミリ秒

5.2 伝送シーケンスの評価

5.2.1 伝送シーケンス

MDDC と OM2M の伝送シーケンスはそれぞれ図 6, 図 7 のように行われる．伝送シーケンスでは，MN から IN に対して通信するラウンドトリップタイムを測定した．MDDC の long-polling を利用した通信が確認できる.

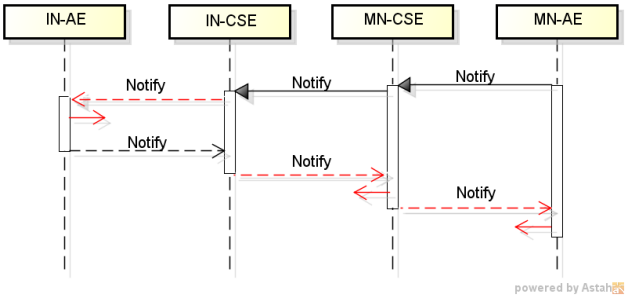

図 6　MDDC の伝送シーケンス

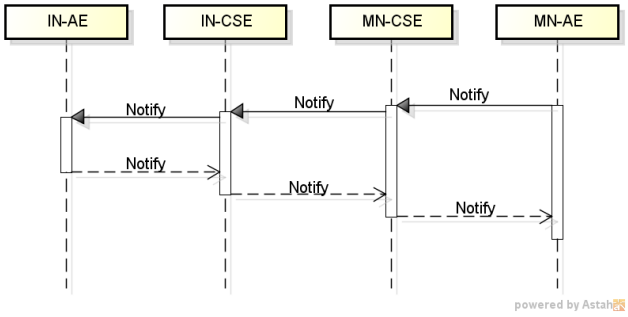

図 7　OM2M の伝送シーケンス

5.2.2 伝送シーケンスの評価結果

伝送シーケンスでは 4 種類（10B, 1kB, 100kB, 10MB）のコンテンツサイズについて調べた．10 回の試行を行い，平均値を求める．MDDC と OM2M の評価結果は，表 5 のようにまとめられる．グラフ化したものを図 8 に示す.

10Byte の小さなメッセージのみ, MDDC が高速であったが，その他は OM2M の方が高速であった．メッセージサイズが大きいほど顕著になり，HTTP の 10MB では OM2M が約 6 倍高速となる．OM2M がサーバ方式で直接送信が可能であるのに対して，MDDC が long-polling 方式でサーバプッシュのためのリクエストが不利になっているものと考えられる.

HTTP と CoAP の比較では 10Byte, 1kB では CoAP が高速，100kB, 10MB では HTTP が桁違いに高速であった．CoAP のメッセージの上限サイズは 1024Byte であり，パケット分割処理が不利になっているものと考えられる.

表 5　伝送シーケンスの所要時間

単位：ミリ秒

メッセージサイズ	MDDC		OM2M	
	HTTP	CoAP	HTTP	CoAP
10Byte	79	40	217	197
1kB	89	51	31	57
100kB	135	9283	28	1130
10MB	2,085	-----	345	25614

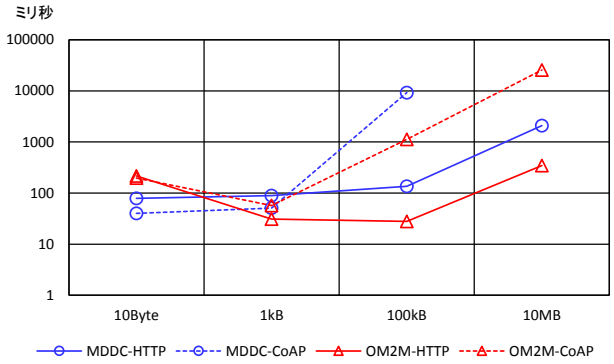

図 8　伝送シーケンスの所要時間のグラフ

6. まとめ

車両情報を収集するためのプロトコルとして，oneM2M 規格に着目し，Release 1 の直後から自作したソフトウェア MDDC と，後発のオープンソースソフトウェア OM2M の機能面・性能面の比較を行った．

機能面では，OM2M では，oneM2M で規格化された最新の機能が網羅的に実装されていた．一方で，モバイルノードからインフラノードに対する登録・変更・削除機能は oneM2M 規格では Annex とされており，OM2M での実装は確認できなかった．この機能は車両のようなモバイル端末には必須であるため，MDDC では実装されている．

性能面では，MDDC が OM2M よりも接続時間は短かったが，逆に伝送時間は長くなった．OM2M は Java 言語で実装されており，初期ロードやプロセスの起動処理で不利と考えられる．一方，MDDC はサーバプッシュに long-polling を採用しているため，OM2M のサーバ・クライアントを併設する方法に比べて不利であると考えられる．

HTTP と CoAP の比較も行った．CoAP はパケットのメッセージサイズの上限が 1024B であり，メッセージサイズがこれよりも小さい場合は伝送時間が短く，大きくなるにつれて伝送時間が長くなり，パケット分割処理の影響が大きい様子が確認できた．

結論として，オープンソースソフトウェアである OM2M の機能面・性能面の充実が確認できた．今後の開発の方向性として，MDDC の開発はここで終了し，得られた知見をもってオープンソースソフトウェアの開発に合流したい．

2017 年 3 月に oneM2M のショーケース第二弾が東京で行われ，oneM2M 規格をベースにした複数の展示が公開された．ショーケースに必要な分だけソフトウェアを自作した展示も，説明からいくつか確認できた．本研究の知見の共有により，M2M の標準規格やオープンソースソフトウェアに関する議論が深化することを期待したい．

参考文献

[1]　笹原ほか: oneM2M 規格に基づいた車両情報収集サービスの試作（デモセッション），DICOMO 2016，情報処理学会 (2016).
[2]　Pure Ruby implementation of RFC 7252 (Constrained Application Protocol (CoAP)) Version 0.1.1（オンライン），入手先 <https://github.com/nning/coap> 参照 (2017-05-07).
[3]　The Python CoAP library Version 0.1（オンライン），入手先 <https://github.com/chrysn/aiocoap> 参照 (2017-05-07).
[4]　oneM2M open source implementations, oneM2M partners（オンライン），入手先 <http://wiki.onem2m.org/index.php?title=Open_Source> 参照 (2017-05-07).
[5]　TS-0001 Functional Architecture Version 2.12.2, oneM2M partners (2017).