L-007

libcurl を用いた IoT 向けデータ送信 S/W の省メモリ化
Memory-saving methods for IoT device data transmission S/W using libcurl

折本 拓真† 鷲尾 元太郎† 田村孝之†
Takuma Orimoto Gentaro Washio Takayuki Tamura

1. はじめに

近年、工場や家庭、マンションなどに存在する様々な機器に通信機能を持たせ、クラウドに接続し遠隔監視・遠隔制御の機能追加や機器稼働ログ分析による機能改良などの利便性を向上させる IoT システムが求められている[1]。我々は、ゲートウェイ(GW)上で宅内やビル内の機器データを収集し、クラウドに送信する機器データ送信 S/W(以下、送信 S/W とする)を開発した。

送信 S/W の実装環境の候補を図 1 に示す。各環境は実装メモリが 100MB 級通信モジュールであり、Linux を搭載している。

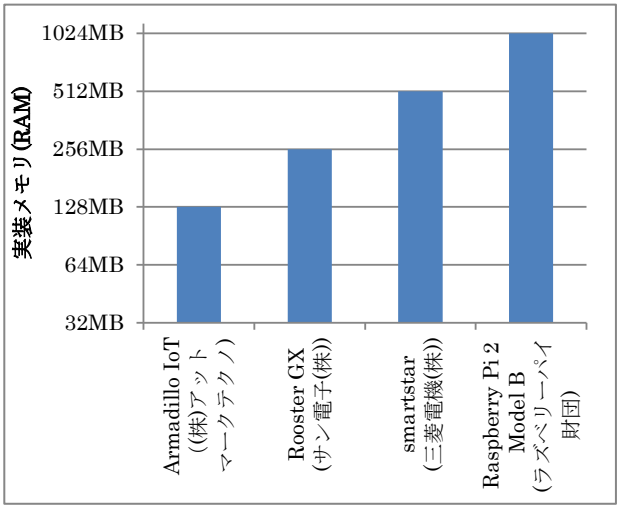

図 1 送信 S/W の実装環境候補

送信 S/W は、開発効率向上のため Linux で標準的な HTTP/HTTPS 通信ライブラリの libcurl、XML ライブラリの Libxml2 を使用した。libcurl の簡易インタフェースを用いたプロトタイプ版では、機器データサイズに比例してメモリを消費するため、大量の機器データを送信出来ないという制約がある。今後、機器稼働ログの分析手法高度化に伴い、詳細なログを長時間保存することの重要性が増し、一度に送信する機器データサイズが増大することが想定される。

本稿では、対応策としてストリーム型送信方式を導入することを検討し、更なる省メモリ化に向けて不要機能削除や軽量な SSL/TLS ライブラリによる送信 S/W のコード領域削減可能量を調査した。

2. 送信 S/W の概要

図 2 に送信 S/W の構成図を示す。送信 S/W は機器データをクラウドにあるデータ蓄積アプリに送信するインタフェースをデータ収集アプリに提供する。データ収集アプリは、機器データを収集し送信 S/W に受け渡すアプリである。

データ収集アプリは、初めに一括して機器データをメモリ上に読み込み、送信 S/W を呼ぶ。送信 S/W は、データ収集アプリから受け取った機器データを Libxml2 に渡し、規定の XML データに変換する。次に、libcurl、OpenSSL を用いてクラウドに機器データを送信する。

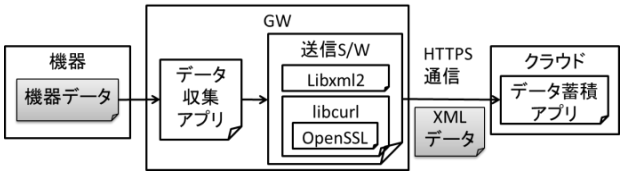

図 2 送信 S/W を用いた機器データ収集システムの構成図

3. 送信 S/W のデータ領域削減

3.1 課題

機器データ送信時における GW のメモリの内訳を示す模式図を図 3 に示す。データ収集アプリが送信対象の機器データをメモリ上に保持した状態で、送信 S/W が XML データを保持する。そのため、確保するメモリの最大値は機器データ、XML データの各サイズの合計となる。ログデータなど数 KB の少量の機器データを送信する場合、数百 MB 級の通信モジュールでは問題無く送信可能である。しかし、今後センシング間隔の短縮や機器数の増加により、数十 MB の大量の機器データを一括で送信するケースがあり、その場合機器の空きメモリが無くなり機器データの送信が出来ない。

図 3 機器データ送信時の GW のメモリの内訳

3.2 ストリーム型送信方式による対策

大量の機器データを送信する場合、プロトタイプ版ではデータ収集アプリが機器データを上限サイズ以下に分割、送信 S/W が複数の XML データを送信、クラウド側で結合処理が必要である。そこで、単一の XML データを少量ずつ送信するストリーム型送信方式を導入した。図 4 にプロトタイプ版とストリーム型送信方式の比較図を示す。プロトタイプ版では、1 回の送信に 1 つの XML データを送信していたが、ストリーム型送信方式では、クラウド側で 1 つの XML データになるように XML データの断片データを送信する。具体的には、libcurl の

† 三菱電機株式会社, Mitsubishi Electric Corporation

CURLOPT_READFUNCTION を用いたコールバック関数内で機器データサイズを指定し、データ収集アプリから機器データを受け取り、XML データへ変換する。その後、libcurl へ XML データを受け渡し libcurl により送信する。コールバック関数の呼び出し、送信の処理を機器データの末端まで少量ずつ繰り返す。送信 S/W が機器データサイズを指定することにより、送信 S/W で保持するデータ量は一定になり、大容量の機器データの送信を可能とした。

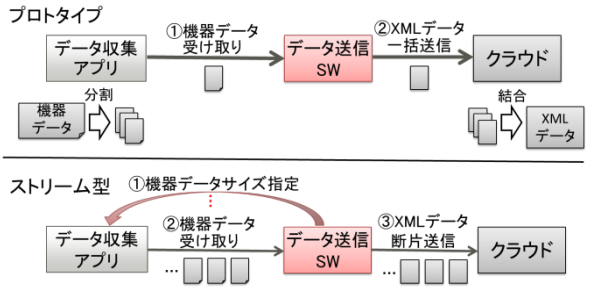

図 4　送信方式の比較

3.3 評価

評価用の GW は Atmark-techno 社製の Armadillo-IoT[2]を使用した。Armadillo-IoT はクロック周波数 400MHz、RAM 128MB を搭載している。

送信する機器データサイズ別に機器の空きメモリ量を測定した結果を図 5 に示す。プロトタイプでは、空きメモリ不足により 16MB 以上の機器データの送信が不可であったのに対し、提案方式では 16MB 以上の機器データの送信が可能となり、空きメモリ量はほぼ一定であった。以上より、ストリーム型送信方式を用いることで、送信する機器データサイズに関わらず送信可能となることが確認できた。

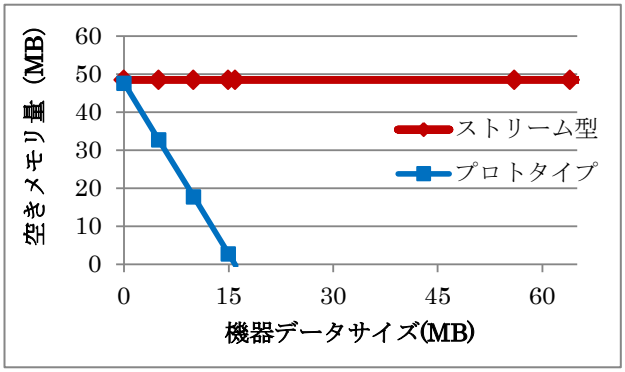

図 5　機器データサイズ別機器の空きメモリ量

4. 送信 S/W のコード領域削減

送信 S/W のデータ領域削減について 3 章で検討したため、本章では更なる省メモリ化に向けて送信 S/W のコード領域削減を主とする。そのため、以下の手順で送信 S/W のコード領域削減検討を行った。

方針(1) プログラムの動作に不要なライブラリがあれば、除去するようにビルドオプションを変更する。
方針(2) 送信 S/W のライブラリのコード領域を調べ、大きい順に以下を行う。
方針(3) プログラムの動作に必要なライブラリであれば、より小さな実装に置き換える。

初めに、GNU TLS や SSH などの不要な通信プロトコルを除去するように libcurl のビルドオプションを変更した。これにより、libgnutls.so や libssh2.so、libsals2.so への依存性を取り除き、送信 S/W のコード領域を約 2.9MB 削減できた。次に、コード領域の一番大きい OpenSSL をより軽量なライブラリに置き換えることを考え、OpenSSL と互換性があり軽量である wolfSSL を導入した。その結果、送信 S/W のコード領域を約 1.5MB 削減できることを確認した。

プロトタイプの送信 S/W を①、libcurl のビルドオプションを変更したものを②、②に加え wolfSSL を導入したものを③として、コード領域を比較した図を図 6 に示す。ビルドオプションの変更、wolfSSL を導入することにより送信 S/W を約 4.4MB 削減できることが分かった。

また、送信 S/W のコード領域において割合の多い libc を組み込み系の uClibc に置き換えることで、送信 S/W のコード領域を約 2.9MB まで削減可能の見込みである。

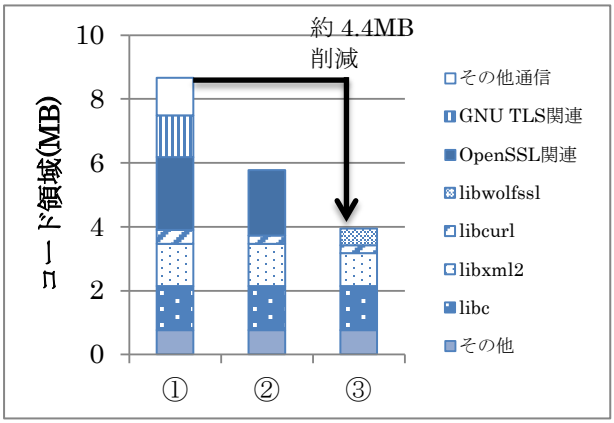

図 6　コード領域比較

5. おわりに

本稿では、Linux OS で標準的な libcurl を使用した送信 S/W において、大量の機器データ送信に対応するためデータ領域削減方法の検討と更なる省メモリ化に向けてコード領域削減可能量を調査した。その結果、前者ではストリーム型送信方式を導入することで解決することを示し、後者では不要機能の削除、軽量なライブラリに置き換えることで、約 2.9MB まで削減可能であることが分かった。

今後は、送信 S/W の更なる軽量化を進めるとともに、Linux 環境以外の機器に対応した実装方式を検討していく。

参考文献

[1] 伊藤正裕，原田雅史，田村孝之，河村美嗣，児玉拓，“三菱電機スマート制御クラウドサービス“DIAPLANET””，三菱電機技報，89，No. 8，430～433(2015)
[2] IoT デバイスプラットフォーム Armadillo-IoT(オンライン)，http://armadillo.atmark-techno.com/armadillo-iot(参照 2016-06-20)