組み込み機器用リモートオペレーションミドルウェア

Remote Operation Middleware for Embedded Equipment

山本　周二[*1]　和田　春美[*1]　桑原　博文[*1]
YAMAMOTO Shuji　WADA Harumi　KUWAHARA Hirofumi

REO (REmote Operation middleware)は，リモートアプリケーション構築のためのミドルウェアの一種である。このミドルウェアは組み込み機器用のライブラリ群として提供される。たとえば，複数のクライアントからのリクエスト処理，アカウント管理，排他制御，組み込み機器上のアプリケーションと通信プロトコルの分離などのライブラリがある。これらREOの機能を利用することで，アプリケーション開発者は汎用のツールあるいは汎用のミドルウェアを利用するよりも容易にリモートアプリケーションを構築することができる。REOは特に組み込み機器に搭載することを前提とし，プログラムサイズ及びリソースをできるだけ抑えた設計を行った。またタスクやスレッドの実行方法などのREOのOS依存機能及び通信機能をコンポーネント化し，組み込み機器に搭載される多種のOS及び通信プロトコルに対して容易な対応を可能とした。

We developed REO as a middleware to develop remote application system. REO (REmote Operation middleware) consists of some libraries mostly for embedded equipment. REO works with standard common tools and middlewares such as java, Web, etc., and it provides useful functions such as multi-client processing, user authentication, exclusive control and decoupling applications on embedded equipment from communication protocols. Therefore, using this REO, application designers can save design cost and time for construction of remote application systems and that is easier than with standard/common tools. Also, because REO is designed for mostly embedded equipment, REO's resource requirements are suppressed as small as possible for installing into the embedded equipment. Moreover, providing REO's communication function and OS-based function such as task and thread executions, designers can have much flexibility for multiple OSs or platforms in embedded equipment and easily can emigrate applications for a system requirement.

1. はじめに

近年，ネットワークのインフラが整備されるにつれほとんどの機器に汎用の通信インタフェース機能が搭載されている。これに伴い，リモートメンテナンス/リモート監視などネットワークを利用したアプリケーションへの期待/要求が高まってきた。組み込み機器においてもEthernet等のインタフェースを備えているものが多くなり，ネットワークを利用したデータ授受や機器の管理などへの適用が注目されている。このようなアプリケーションはその用途毎に専用にアプリケーションを開発しているのが現状である。そのアプリケーションは複雑であるが多くの場合共通な基本機能の組み合わせで実装することができる。共通となる基本機能をツール(WebBrowser, pop client, ftpなど)とシステムを構築するためのミドルウェアに分けると，REOはネットワークを利用したリモートアプリケーション構築のための「ミドルウェア」と位置付けられる。リモートアプリケーション開発の効率向上とより高度なリモートアプリケーション開発の簡易化を目的としてREOを開発した。

2. REOの特長

REOはアプリケーション機能をコンポーネントとして実装し，コンポーネントでシステムを構築するという分散オブジェクトシステムの考え方をベースに設計している。コンポーネントには，通信のコンポーネント，オブジェクト管理のコンポーネント，アプリケーションのコンポーネントなどがあり，リモートアプリケーション開発者は機能をアプリケーションのコンポーネントとして開発する。更に，REOはコンポーネントの開発環境，システム動作のモニタなど利用環境も含んだミドルウェアである。

*1 R&Dセンター ITプロジェクトセンター

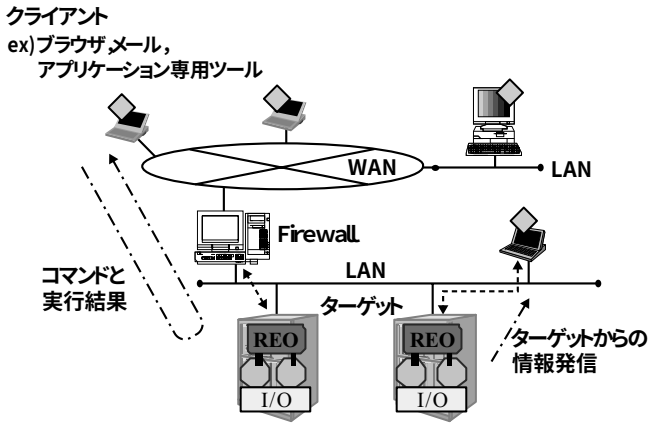

図１　システム構成例

2.1 システム構成例

図１は２つのターゲットをLAN上およびWAN上のクライアントからリモートでアクセスする構成例である。インターネット上のリモートクライアントからのアクセスはそのシステムの運用方法によりFirewall経由などの適した方法が採られる。このようなシステム構築を目的としたアプリケーション開発のためのミドルウェアがREOである。

2.2 REOの位置付け

REOは**図２**のようにJava[*2]，CORBA[*3]などの汎用機能或いはミドルウェア上に構築している。REOは主に機器側のアプリケーション開発のための諸機能を提供し，開発者はREO機能とミドルウェア機能を用いて機器上で動作するアプリケーションを構築する。

3. REOの諸機能

アプリケーション開発者は，リモートから使われているかどうかを意識することなく，あたかもローカルアプリケーションのようにREOを利用したターゲット上のアプリケーション(以下EO=Execution Objectと呼ぶ)を設計することが出来る。

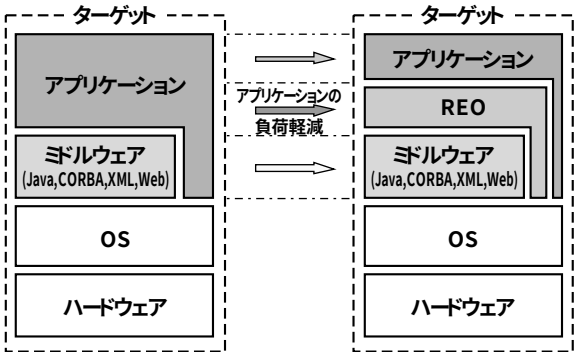

図２　アプリケーションのREO化

3.1 サーバとしての機能

・マルチクライアント処理
複数クライアントからの１つのEOに対する要求を，あたかも１つのクライアントからの要求であるかのように変換する機能。EO開発者は複数クライアントからの要求処理をEOに実装する必要がない。

・マルチプロトコル処理
それぞれのクライアントが利用しているプロトコルをREO内の共通プロトコルに変換する機能と，返信時にREO内のプロトコルをそれぞれのクライアント用のプロトコルに変換する機能。EO開発者はクライアントとのプロトコルが何であるかに関係なくEOを設計できる。更に，EOを修正することなく新たなプロトコルを追加できる。

・マルチフェイス
クライアントに送り出すデータをクライアントで必要とする或いは適したフォーマットに変換する機能。そのデータの受け側であるクライアントはREOにフォーマットを指示するだけでよい。また，データを送り出す側であるEOがデータのフォーマットをクライアント別に指定することもできる。

・データキャッシュ
ターゲットがI/Oから収集するデータをREO内にキャッシュする機能。複数のクライアントが同じデータを要求する時などには，REO内にそのデータをキャッシュすることでクライアントへのレスポンスを高め，I/Oの通信路の負荷低減を行うことができる。

・REO動作監視
ターゲット上で動作しているREOへの通信路での障害発生，或いはREOが停止したことをクライアントが検知できる機能。検知後に管理者への通知，データのバックアップなどのクライアント特有の処理を加えることができる。

・ユーザ認証，アカウント管理
クライアントを識別し，クライアントが利用できるターゲットの機能を制約するためにクライアントを認証する機能。どの機能を利用できるかは**図１**のようなシステム構築者がターゲッ上にアカウント情報を設定する。

・排他制御
クライアントが，あるターゲットの機能を利用している時に，他のクライアントがそのターゲットで利用できる機能に制限を加える機能。１つのクライアントが

テストモードで操作をしている時に，他のクライアントが実動作モードに変更することを禁止するなど，複数のクライアントからのターゲット操作の競合回避を目的とする。

・動的モジュールローディング
複数のターゲットが新しいEOを必要とする時，システム動作時にそのEOをクライアントからターゲットに送り込みターゲット上で実行させる機能。システムを止めること無くターゲットを操作する機能の追加，変更が行える。

3.2 システム構築のための機能

・プラグアンドプレイ
REOを搭載している機器のシステムへの接続をクライアントが検知できる機能。検知後に各クライアントはクライアント特有の方法でその機器を利用することができる。

・レトロフィット
ターゲット上の既存アプリケーションをクライアントがEOとしてアクセスするために，ターゲット上でインタフェースを変換する機能。ターゲット上にREOを搭載できない場合には，REOを搭載した他のハードウェアをアダプタとしてターゲットに接続し，そのハードを通して利用する方法もある。

3.3 REO運用のための特長

・マルチプラットフォーム
Java，ITRONなどの各プラットフォーム上でそれぞれのREOが動作する。

・運用および開発のための環境
EO開発のためのライブラリと，EOおよびREOの動作モニタが提供される。

4. REOの基本構成

REOの基本構成を**図３**に示す。REOはターゲット上で動作する。

・リモート通信部
各種プロトコルによる通信機能を持ち，REOとクライアント間でのデータ送受信を行う。WebBrowserとのCGIによる通信，E-mailによる通信などを通信方式毎のプロトコル処理を行う。アプリケーション独自の通信方式をここに加えることができる。

・AP(Application)プロトコル処理部
各通信方式毎にアプリケーションに特化した通信データの解析を行う。例えば，ソケット通信の場合は，パケットサイズ/パケットの解析，E-mailではメール内容の解釈など。

・リクエスト管理部

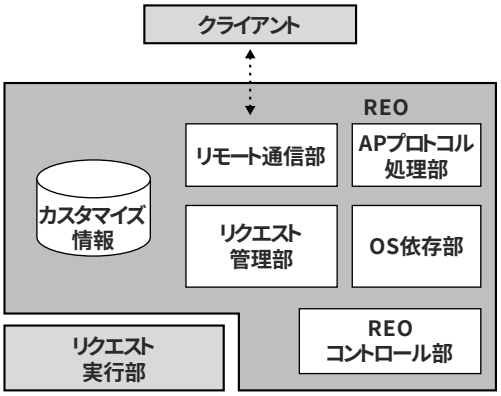

図３　REOの構成

クライアントとEOをつなぐproxyとgatewayの役割を持つ。「3.REOの諸機能」に挙げたほとんどの機能をここが持つ。

・OS依存部
スレッド/プロセスの起動方法，排他制御の方式などOSに依存する機能をここで隠蔽する。

・REOコントロール部
REOの動作をコントロールする。EOの強制終了，カスタマイズ情報の変更など。

・リクエスト実行部
クライアントからのリクエストを処理するEOの集合である。この機能はすべてアプリケーション開発者がREOとのインタフェースを基に設計し，REOに設定する。

・カスタマイズ情報
(ユーザ名とパスワードの)アカウント情報，通信プロトコルのパケットサイズ，リクエストコマンドとそれを処理するためのEOの対応表などアプリケーション個別の情報を持つ。この情報の設定は機器管理者が行う。

5. 基 本 動 作

図４にREOの基本動作を示す。EOの動作には２種類の動作形態がある。１つは，クライアントからターゲットに対してリクエストを発行し，そのリクエストをEOが処理し，その結果をリクエストの発行元に返信する形態である。この動作形態は，リクエストの処理が終わったEOは次のリクエストを受け付けるまで何の動作も行わない。もう１つの動作形態は，ターゲット上で常時動作しているEOの処理結果をEOが発信する形態である。例えば，ターゲットが収集しているデータをクライアントがモニタするとかターゲットの異常監視を行う場合である。EOがどちらの形態をとるかをEOの設計時に決める。

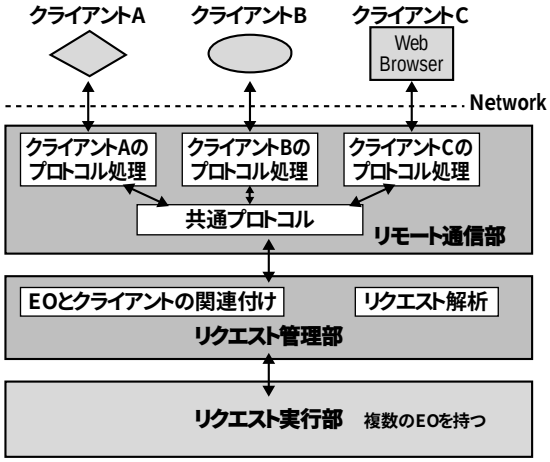

図４　REOの基本動作

5.1 リクエスト受信，応答返信動作

リモート通信部はクライアントとの通信手段別に受け口を持ち，そこでデータの受信と受信データのフォーマット解析を行った後，リクエスト管理部に解析結果を渡す。リクエスト管理部では受け取ったデータ，つまりクライアントからのリクエストを解析しリクエストを処理すべきEOを判断し，3.1で示したアカウントチェック，排他制御などのREOの機能を行った後，リクエスト実行部に処理を行うEOと解析データを渡す。リクエスト実行部はプラットフォーム特有の方法でEOの起動/実行を行う。

複数クライアントからの同一EOのデータをモニタするリクエストでは，EOからのデータの発信をREOが受けると，そのEOをその時点でモニタしている複数のクライアントにデータをマルチキャストする。EOのデータを要求しているクライアントが無くなった時にはREOがそのことをそのEOに知らせる。データキャッシュがある場合にはEOにコマンドを送らず，データキャッシュのデータをクライアントに返す。

これらの処理において，EOはクライアントとの関係をまったく知る必要がない。例えば，EOはどのクライアントからどのような通信方式でデータを受信したか，返信時もどのクライアントへはどのような通信方式でデータを送れば良いか，EOが送り出すデータをいくつのクライアントがリクエストしたのか，を判断する必要が無い。EOは単にREOにデータをメソッドコールで送り出すだけでよい。必要な処理はすべてREOが行っている。また，データの送り出し時に通信路に障害がありデータを送り出せなかった時にはその障害がEOに伝えられるため，EOは管理者へのデータ発信などしかるべき処理を行うことができる。

6. EOの実装方法

アプリケーション開発者の主な作業はEOを実装することとカスタマイズ情報を設定することである。ここではリクエスト実行部でのEOの実装について示す。

6.1 EOの実装

アプリケーション構築者はREO仕様で定めるインタフェースに従ってEOを実装する。このインタフェースは極めて少なく，以下に示す３つのメソッドをEOに実装することがほとんどの作業である。

起動時：EO. exec（CanonicalData cd）
呼出時：EO. call（CanonicalData cd）
返信時：EO. send（CanonicalData cd）

ここでCanonicalDataとはREOが処理するデータの共通表現である。CanonicalDataにはクライアントから送られた全情報が含まれ，付加情報としてクライアントのIDなどクライアントに関する諸情報を含む。通常処理を行う上ではEOが付加情報を扱うことはない。

7. おわりに

REOはリモートシステムの構築に必要な機能を備えた主に組み込み機器のネットワーク対応アプリケーションを対象としたミドルウェアである。アプリケーション開発者は，REOの機能と特長を有効に利用することで高度なリモートアプリケーションを短期間で開発することができる。

本特集の他テーマである「オンライン厚さ計WEBFREX[*4]保守・運転監視ツールWsquare[*4]」のミドルウェアとしてREOが活用されている。ここでのプログラムサイズはJava版で約110 kB，ワークエリア約500 kBと必要リソースの小型化と実行時のオーバーヘッドを抑えた設計となっている。これは既存装置をシステムを変更することなくリモート処理機能をシステムに追加した例であり，REOがレトロフィットに適用できる実例でもある。

さらに，REOは組み込みの機器への適用だけではなく，広く分散アプリケーション構築のミドルウェアとして利用できる機能と特長をもっている。今後はREOをベースにより多くのアプリケーションへの展開を図っていく予定である。

*2 JavaはSun Microsystems Inc.の商標です。
*3 CORBAはObject Management Group Inc.の登録商標です。
*4 WEBFREX，Wsquareは横河電機の登録商標です。