CONFERENCE DIGEST

ロボティクス・メカトロニクス講演会2009

2009 JSME Conference on Robotics and Mechatronics

ROBOMEC 2009 in FUKUOKA

豊かな暮らしを創生するロボティクス・メカトロニクス

Robotics and Mechatronics for Creating an Affluent Society

太宰府天満宮本殿

Sun. 24th ~ Tue. 26th May, 2009

Fukuoka International Congress Center
Fukuoka, Japan

主催　社団法人 日本機械学会 ロボティクス・メカトロニクス部門

The Japan Society of Mechanical Engineers, Robotics and Mechatronics Division

軽量 CORBA RtORB による OpenRTM の実装と評価

An Implementation of OpenRTM by using lightweight CORBA called RtORB

○　原　功（産総研）　　正　安藤　慶昭（産総研）
正　神徳　徹雄（産総研）　　末廣　尚士（産総研）

Isao　HARA, AIST, _isao-hara@aist.go.jp
Noriaki ANDO, AIST
Tetsuo KOTOKU, AIST
Takashi SUEHIRO, AIST

In this paper, we describe an implementation of OpenRTM-aist with a lightweight CORBA which named RtORB(Robot Technology Object Request Broker). RtORB has implemented in C-language and is required almost half the computational resource of omniORB4 which is most popular CORBA implementation. Then it provides the almost the same performance as omniORB4. RtORB could enable to work RT-Component on various computational platforms.

Key Words: OpenRT Platform,RT-Middleware, CORBA

1. はじめに

我々は、既存のさまざまなロボット関連の要素技術の再利用性を高め、知能化技術および次世代ロボットの研究開発の効率化するために、RT ミドルウェアに基づくソフトウェアプラットフォームの研究開発を進めている。RT ミドルウェアは、様々な要素技術を RT コンポーネントと呼ぶソフトウェアモジュールとして作成し、これらのモジュールを通信ネットワークを介して組み合わせることで、様々なネットワークロボットシステムの構築を可能にするミドルウェアである。

現在、我々が実装している RT ミドルウェア（OpenRTM-aist-0.4.2）は、Linux や Windows の上で動作し、C++, Python, Java などの開発言語をサポートし、マルチ OS、マルチ言語対応を実現している。

しかしながら、現在の実装ではコンポーネント間のデータ通信レイアに、分散オブジェクト技術の一つである CORBA (Common Object Request Broker Architecture)を用いているために、そのパフォーマンスや移植性に制限があった。そこで我々は、RT コンポーネント間通信で必要な機能のみを実装した軽量化 CORBA を実装し、RT ミドルウェアのデータ通信レイアに適用した。

本稿では、この軽量化 CORBA である RtORB （Robot Technology Object Request Broker）の実装と従来の RT ミドルウェアで利用してきた CORBA 実装である omniORB4 との比較について述べる。

2. OpenRTM-aist におけるコンポーネント間通信機能

RT ミドルウェアの実装である OpenRTM-aist-0.4.2 におけるソフトウェアアーキテクチャを図 1 に、RT コンポーネント構造を図 2 に示す。

図 2 に示したように RT コンポーネントは、Data Port, Service Port の 2 種類の通信ポートを持ち、ネットワーク上に分散配置された他の RT コンポーネントとの周期的、非周期的な通信を提供している。現在の OpenRTM-aist では、RT コンポーネント間の通信レイアとして omniORB4 を用いていることでこの機能を提供している。omniORB4 は、OMG(Object Management Group)によって規定された CORBA2.6 に準拠した分散オブジェクトミドルウェアの実装である。CORBA2.6 では、分散オブジェクトを実現するために様々な機能が規定されている。omniORB4 では、その多くの機能が実装されており、分散オブジェクトの開発時に複雑なコード記述の必要性と多くの計算資源を必要としている。

一方、OpenRTM-aist で重要な通信機能は、多様な OS で動作し、様々な言語により実装された RT コンポーネント間の頑健なデータ通信を実現することである。また、現有資産と相互運用性を考慮すると、OpenRTM-aist の通信レイアに必要な機能は、CORBA2.6 以降の仕様で作成された他の CORBA 実装との透過性および標準的な基本ライブラリのみを利用した実装である。

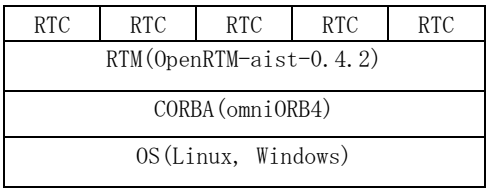

Fig. 1 Architecture of OpenRTM-aist-0.4.2

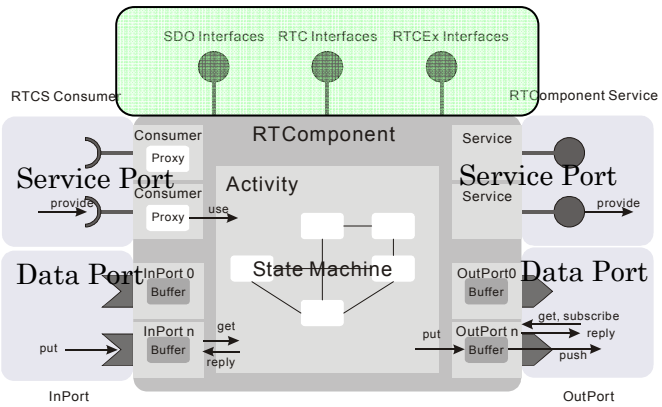

Fig. 2 Overview of RT-Component

3. 軽量 CORBA　RtORB

前節で述べたように、OpenRTM-aist の機能を提供するための通信ミドルウェアとして、最新の COBRA の仕様である CORBA3.1 を基にし、RT コンポーネント間通信を実現する軽量 CORBA の実装を行った。また、実装言語としては、組み込み用の CPU 等に対する移植性および OpenRTM-aist の正規の実装系

である C++との親和性を考慮し、通信コアの部分を POSIX 準拠の C 言語で実装し、C++によるラッピング機能の実装を行った。

3.1 通信コア部の実装

一般に、CORBA によるオブジェクト間通信には、GIOP(General Inter-ORB Protocol)と、その TCP/IP の実装である IIOP(Internet Inter-ORB Protocol)が利用されている。そのため、RtORB の通信コア部では、TCP Socket と select システムコールを利用し、最低限のスレッドのみを利用する実装を行った。実装には、POSIX 準拠の C 言語を用い、COBRA3.1 の仕様に規定されている CDR (Common Data Representation) による marshaling/demarshaling、IOR(Interoperable Object Reference)の実装を行った。さらに、様々なデータを容易に変換するための CORBA_any 型の実装も行った。この通信コア機能を提供する Shared ライブラリは、Linux ディストリビューション Ubuntu－8.10、gcc-4.3.2 を利用した場合およそ 176k バイトであった。

3.2 C++への対応

上述のように、RtORB では、コアとなる通信部分を、C 言語で記述した。一方、OpenRTM-aist は、C++による実装が行われており、データ通信に関わる部分は CORBA C++言語マッピングを前提に実装されている。CORBA では、C 言語と C++言語の言語マッピングは個別に定義されており、そのままでの利用は容易ではない。そのため、各データ構造の互換性確保のための実装が必要になる。この大部分は、Struct（構造体）型と Sequence（配列）型の言語マッピングによる差異の吸収と CORBA_any 型によるデータ型の変換機能を提供する部分である。実装した Shared ライブラリは、前述の通信コア部と同じ条件で約 147k バイト程度であった。

3.4 IDL コンパイラ

RtORB では、動的なインターフェースの生成機能を有しておらず、プログラムの構築には IDL コンパイラを必要となる。IDL コンパイラは、IDL で記載された分散オブジェクトのインターフェース仕様を実行可能なプログラムへ変換をおこなうコードジェネレータである。既存の CORBA の実装系の中で、C 言語によりマッピングを有し、自由に利用可能な実装系として Orbit2 が知られている。Orbit2 は、GNOME プロジェクトで開発され、IDL の構文解析のための基本ライブラリも整備されている。そこで、我々は RtORB の IDL コンパイラとして、Orbit2 で利用されている実装を流用した。また、C++への対応に関しては、この IDL コンパイラによって生成された Stub, Skelton, Skelton-impl のそれぞれのファイル中に対応するコードを自動生成させることで、C 言語でも C++言語でも同様な利用をできるようにした。

3.5 他の CORBA 実装との比較

実装した RtORB と最も一般的な実装である omniORB4 との機能比較を表 1 に示す。比較項目は、CORBA 製品の比較サイト (http://www.puder.org/corba/matrix/) を参考にした。これからもわかるように、RtORB は、既存の他の CORBA オブジェクトとは GIOP でのみ通信を行うことが可能になっており、必要最小限の実装になっている。さらに RtORB では、通信部分のデータ形式変換部分も容易に交換可能な実装を行っているために、ICE や SOAP といった異なるプロトコルを持った分散オブジェクトミドルウェアとの相互利用への拡張も容易に実現可能である。

3.5 RtORB の利用ライセンスおよび公開について

RtORB は、実行系のすべてのコードを完全にスクラッチから書き起こしたものであり、他のライブラリ等のライセンスに影響を受けない。また RtORB の開発は、ロボット知能ソフトウェアプラットフォームの１つとして開発を進めているため、利用ライセンスとしてオープンソースソフトウェアライセンスの１つである EPL v1.0(Eclipse Public License Version 1.0)に準拠している。これにより利用者は、有償、無償にかかわらず再配布、改変可能にしている。

また、RtORB の利用、ダウンロード、サポートに関する情報は、ロボット知能ソフトウェアプラットフォームのオフィシャルサイト[1]から得ることができる。

Table 1 CORBA Product Matrix between RtORB and omniORB

	RtORB	omniORB4
Open Source	○	○
Language Mappings	C, C++	C++,Python
GIOP/IIOP	1.2	1.2
BOA	○	○
POA	○	○
DII (Dynamic Invocation Interface)	×	○
DSI (Dynamic Skelton Interface)	×	○
IFR (Interface Repository)	×	○
VTS (Value Type Semantics)	×	○
Min (Minimum CORBA)	△	○
AMI (Asynchronous Messaging Interface)	×	△
EVENT (Event Service)	×	○
NAME(Naming Service)	○	○
NOTF (Notification Service)	×	○

4.性能評価

RtROB の動作性能を評価するため、OpenRTM-aist において、6 次元の double 型（TimedDoubleSeq 型）のデータを送受信するコンポーネントを作成し、データ送受信の時間計測を行った。テストは、Ubuntu-8.10 をインストールした Core2Duo T9500-4GB メモリの計算機上で、各コンポーネントおよび RT システムエディタを動作させて行った。

表 2 に、omniORB 版と RtORB 版の RT コンポーネントが消費しているメモリ量を示す。この情報は、/proc/[プロセス番号]/status で得られる値であり、表中に示された各項目は、

VmPeak: プロセスがある時点で使っていた最大メモリ
VmLock:スワップアウトされているメモリ
VmHWM:ある時点で使っていた最大物理メモリ
VmData:動的仮想メモリの領域サイズ
VmStk:スタックサイズ
VmExe:テキスト領域のサイズ
VmLib:ロードされたライブラリのサイズ
VmPTE:ページテーブルのサイズ

を表している。この表から、RtORB は omniORB4 の半分程度のメモリしか消費していないことがわかる。RtORB では、メモリ使用量などの計算資源を節約するために、通信時に使用するバッファのサイズを固定し、すべてのリクエストで共有す

る方式をとっている。このような実装によって、様々な利用形態での計算資源の上限を設定できるようにしている。これは、システムが完成した後は、コンポーネント間の通信量は大きく変動することは少ない場合が多いため、通信バッファをユーザにより固定する機能の追加を実現するためである。

表 3 に、性能評価用の RT コンポーネント間のデータ送信を 30000 回行ったときの実行時間を示す。データ送信の実行時間は、まず、各データパケットに送信時刻のデータを入力し、受信側でそのパケット内の時刻と受信時刻の差として算出した。表 3 中の「RtORB-omni」という表記は、送信側を RtORB 版の RT コンポーネント、受信側を omniORB 版の RT コンポーネントとした場合を表している。また、各コンポーネントの実行コンテキストの実行周期を、1,000,000Hz とし性能評価テストを試みた。

この表の平均的な実行時間から、RtORB は、omniORB と同等以上のデータ通信機能を実現していることがいえる。ただし、最大時間は RtORB-RtORB の場合には、他の実装の倍近くの時間を要している。この原因として、RtORB では、マルチスレッドによる処理は行っていないために、同時に発生したリクエストに対し順次処理による影響であると考えられる。

Table 2 Required computer resource of Simple RT-Component

	RtORB版	omniORB4版
VmPeak	24120 kB	52024 kB
VmLock	0 kB	0 kB
VmHWM	3784 kB	6360 kB
VmData	17776 kB	41328 kB
VmStk	84 kB	84 kB
VmExe	112 kB	92 kB
VmLib	3840 kB	9844 kB
VmPTE	24 kB	36 kB

Table 3 Result of benchmarks

	平均(usec)	最大(usec)	最小(usec)
RtORB-RtORB	111	19819	43
RtORB-omni	97	11016	35
omni-RtORB	928	11929	34
omni-omni	459	11526	22

5. おわりに

本稿では、ロボットソフトウェア開発プラットフォームである OpenRTM-aist の軽量化を行うために、基本機能 CORBA のみを実装した RtORB の概要について述べ、omniORB4 との性能を比較に関して述べた。RtORB は、OMG で規定されている最新仕様である CORBA3.1 に準拠して実装されているが、OpenRTM-aist の動作に必要な最低限の通信機能を提供している。また、基本機能となる通信コア部はすべて C 言語により記述したため、他の CORBA 実装よりも必要計算資源を抑えることが可能になり、通信機能の面においても omniORB4 とほぼ同等の性能を実現することができた。今後は、CORBA 標準のプロトコルのみならず、共有メモリやマルチキャスト通信への対応も検討をすすめ、より広範囲のロボット知能化技術の RT コンポーネント化に貢献していく予定である。

文　献

[1] OpenRT Platform Official Site: http://www.openrtp.jp/

[2] OpenRTM OfficialSite: http://www.openrtm.org

[3] 原功　他,“ロボット知能ソフトウェアプラットフォーム”, 日本ロボット学会学術講演会, vol.64-626 , pp.3854-3861, 1998.