ネットワーク環境におけるマルチメディア並行計算プラットフォームの実現

前川 博俊, 斉藤 隆之, 千葉 哲央, 岸 達男, 小早川 雄一
(株)ディジタル・ビジョン・ラボラトリーズ

我々は, 来たるべき成熟したネットワーク環境におけるマルチメディア情報サービスの実現を目的として, そのシステム構築基盤としての並行計算プラットフォームを開発している. そこでは, アプリケーション空間における相互参照の管理による高速なメッセージ通信や, ネットワークワイドな連続メディア処理の機能を実現する. ネットワークの多様な形態や構成にも対応し, 柔軟でオープンなネットワーク計算の基盤を提供する.

1. はじめに

ネットワーク技術や蓄積技術の発達とともに, 素材や表現方法の異なる情報をネットワークワイドな多様な局面で柔軟に利用できるようなシステムの実現が求められ, そのための基盤技術が整いつつある. 我々は, そのようなシステムのためのネットワーク計算環境の実現を目的として, マルチメディア処理機能を備え柔軟なネットワーク空間の運用が可能な並行計算プラットフォームを開発している. [1-3]

分散処理やネットワーク計算に関する従来の計算システムでは, マルチメディア処理で特徴的な連続メディアデータの分散処理機能やフレキシブルでオープンなネットワーク機能への対応が十分ではない. クライアント・サーバ方式[4, 5]は, 動的に機能や接続形態を変更したり複数のモジュールが相互対話的に動作する用途よりも, 機能や仕様が確定した計算に向いている. リソース記述子の指定に基づいてネットワーク上のデータやプログラムをダウンロードし計算を進める方式[6, 7]は, 実現される対話性やシステムの持つ機能の発展性に限界がある. CORBA[8]のような均一空間上でオブジェクトの所在を特定するパラダイムを持ったプラットフォームは, 広域のネットワーク計算には不向きであり, 分散処理の特質を隠蔽していることによる計算効率の低下を伴う. PVM[9], MPI[10]といった分散システムの環境やツールは, より密に結合されたシステム用にデザインされたものである. [11]では並列モデルの体系化によって並列性の向上を試みているが, これは高性能計算を目的としている. 分散オブジェクト指向言語や分散実行環境がかねてより開発されているが[12-16], ネットワーク計算の種々の機構はさらにその上に構築する必要がある. 一方ネットワーク構成の観点からは, ネットワークの物理的構成からネットワーク上での計算の概念形成に至る広い範囲で, その形態と運用に柔軟性を持たせる試みがなされている. [17, 18]

我々が想定する高度なネットワーク環境上でのマルチメディア情報サービスを実現するには, このような機能やシステムの特長の発展とともに, アプリケーションとネットワークの構成の柔軟性やネットワークワイドな連続メディア処理機能を備えたシステムの出現が求められる.

我々がプラットフォームを開発する目的は, マ

Development of A Multimedia Concurrent Computation Platform for Network Environment, Hirotoshi Maegawa, Takayuki Saito, Tetsuhiro Chiba, Tatsuo Kishi, and Yuichi Kobayakawa, Digital Vision Laboratories Corporation
{hmaegawa, tsaito, chiba, kishi, kobayakawa}@dvl.co.jp

ルチメディア処理機能を備えネットワーク空間上でのシステム構成の柔軟性に対処した並行計算系を実現し，その機能の有効性と発展性を検証することである．また一方での開発の動機は，来たるべき高度なネットワーク社会におけるマルチメディア情報サービスシステムの実現を検証するためのプラットフォームを提供することにある．我々は，映像や音声を含むマルチメディア情報を商品としてオープンなネットワーク上で自由に取り引きするための機構の実現を目的として，システムミドルウェアの開発を行なっている．[2, 3] 本研究はその一環である．

本稿では，我々が開発しているプラットフォームの並行計算とマルチメディア処理のパラダイム，メッセージパッシングを基本とした分散処理とネットワークに対応した連続メディア処理の方式，および，ネットワーク空間におけるオブジェクト参照の管理の方式について述べる．

2. マルチメディア並行計算のパラダイム

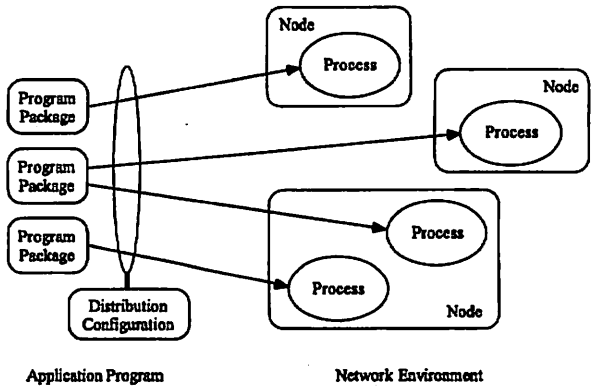
2.1. 並行計算の基本パラダイム

ネットワーク環境上に，並行オブジェクトからなる分散アプリケーション空間を構成し，その空間上で次のパラダイムを持った並行計算を実現する：

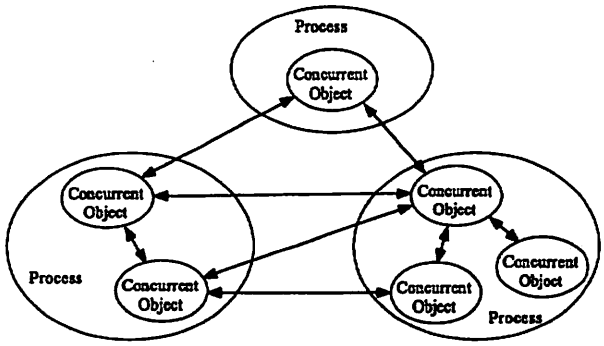
- 分散手続き(関数)呼出
- 分散オブジェクト指向計算
- ネットワーク大域共有変数

並行オブジェクトは，アプリケーション空間上の計算の実体であり，リモート呼出可能な分散関数，分散オブジェクト，あるいは，大域共有変数である．並行オブジェクトは，相互共有の記憶を持たない．分散オブジェクトはその内部に，ローカルオブジェクトを所有できる．ローカルオブジェクトは，分散オブジェクト内で共有記憶を持つ．分散関数は，実装上もオブジェクトであるが，ローカル記憶を持たない．[1] 分散オブジェクトの生成は，そのための分散関数による．

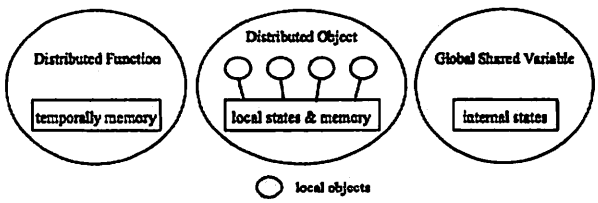
アプリケーション空間に配置する並行オブ



(a) Program-Packages and Processes



(b) Concurrent-Objects and Processes



(c) Concurrent Objects

Fig. 1. Concurrent Objects and Application Space

[1] 分散関数オブジェクトは内部状態として，リモートコーラなどの制御情報を持つ．

ジェクトは，ネットワークノードにプロセスを生成して管理する．プロセスは，オブジェクト配置とプロセス間通信の機能を持つ．並行オブジェクトと，プロセス，アプリケーションプログラムの関係を，Fig.1 に示す．

上記の計算を行なうための並行オブジェクト間の通信（関数やオブジェクトメソッドの呼出，共有変数へのアクセス）は，次の方法により行なう．

- 非同期メッセージ送信（返値を持たない一方向の通信）
- 完全同期呼出（返値が来るまで呼出側がブロックする）
- 遅延評価型同期呼出（返値にアクセスしたときブロックする）

これらの通信を総称してメッセージと呼ぶ．メッセージ上では並行オブジェクト間で，後述のマルチメディアデータを含む任意のデータ構造体を送ることができる．

メッセージによって起動される計算は，関数やメソッドの存在するプロセスにおいて実行する．実際に稼働させるプログラミング言語によって，メソッドがオブジェクトに属するのであれば，その実行はそのオブジェクトで行なう．メソッドが凡関数から派生するものであれば，実行はそのメソッドの所在で行なう．

計算の演算対象のスコープは，並行オブジェクトの内部である．リモートオブジェクトのオペランドを直接見ることはできない．

2.2. メッセージ評価のスケジュール

メッセージの受信による並行オブジェクト上の計算の起動とその実行は，プロセス内で管理する．プロセスは，メッセージに付帯した時間属性と優先度に基づいて，評価すべきメッセージを選択して実行状態とする長期スケジューリングと，さらに実行状態にある複数のオブジェクトを並行して処理する短期スケジューリングによって，メッセージ評価の実行を制御する．並行オブジェクトのうちの分散オブジェクト上での計算を要するメッセージの評価は，そのオブジェクトにおいて排他的かつアトミックに実行する．

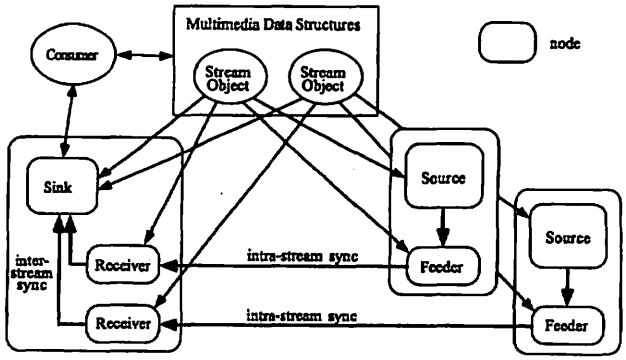


Fig. 2. Continuous-media Processing and Multimedia Data Structures

2.3. マルチメディア処理のパラダイム

連続メディアストリームは，ストリームのソースとシンクの記述子と，その指定に基づいてネットワークにわたるリアルタイムストリーム配送機構を生成し制御する機能とを備えたストリームオブジェクトによって表現し処理する．このオブジェクトは，プログラミング上アトミックに扱うことが可能で，また，ネットワーク上を移動させ機能させることができる．

さらに，複数のストリームオブジェクトと他のオブジェクトを組み合わせたマルチメディアストラクチャを扱うことができる．このストラクチャは同様にメッセージ上での送信が可能であり，また，ストリーム間同期などの機能を付帯機能として備える（Fig.2）．

2.4. 通信メディアの多重化と非均質性

メッセージ送信の際，使用するネットワークの種類を指定することが可能であり，通信するデータのメディアの種類と制御の内容に応じて異なる通信方法が選択できる．また，メッセージ通信とストリーム配送機構を中継ノードを用いて多段的に扱うことにより，非均質なネットワーク環境にも対応できる．

2.5. アプリケーション空間とスケーラビリティ

マルチメディア並行計算を実現するアプリケーションプログラムは，プロセスに相当する単位でプログラムパッケージとして分割し，ネットワーク上に分散配置する．[2] 分散配置は，アプリケーションプログラム上のコンフィグレーション設定によりシステムが行なう．同一のプログラムパッケージから複数のプロセスを生成してもよい．

アプリケーションプログラムは，プログラムパッケージのネットワークノードへの配置替え，あるいは，パッケージ内の並行オブジェクトのクラスタ構成の変更により，アプリケーション構成のトポロジーを保存したままで，ネットワーク上へのスケーラブルな再配置が可能である．並行オブジェクトへのリモート参照は，ネットワークノードとプロセスの特定によって管理する．従って，リモート参照の集合によって構成されるアプリケーション空間は，物理的なネットワーク構成に依存しない柔軟性を持った空間設定が可能である．

3. 分散処理の方式

3.1. メッセージパッシングアーキテクチャ

並行オブジェクト間の通信は，メッセージ送信を基本として実現する(Fig.3, 4)．

メッセージ送信では，引数の構造体を認識して送信可能な論理形態に変換し，送信先のオブジェクトに対するリモート参照から送信先のノードとプロセス，オブジェクトを特定し，送信の処理を行なう．送信先がドメインやオブジェクトクラスタになっていれば，マルチキャストを行なう．参照子がプロキシオブジェクトを指していれば，プロキシを中継してメッセージ送信を行なう．また送信先の所在によっ

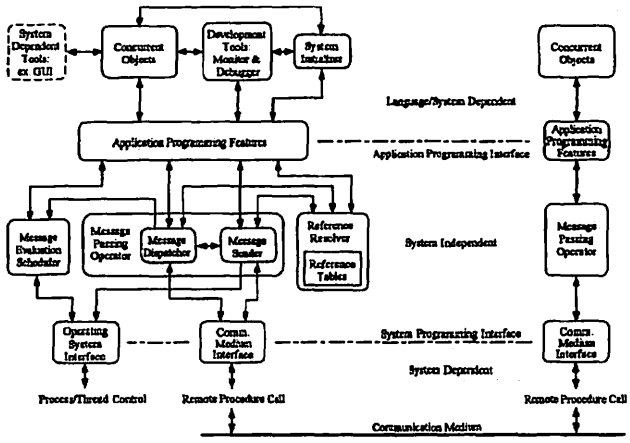
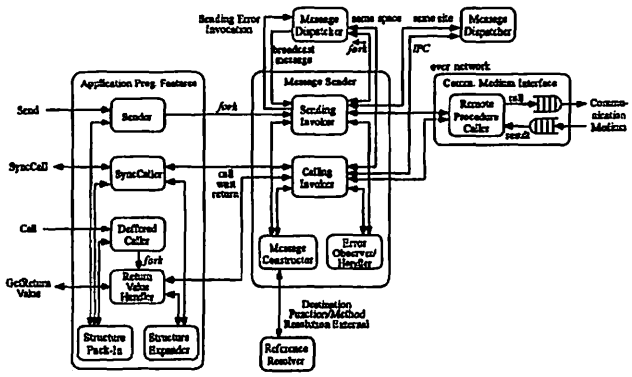
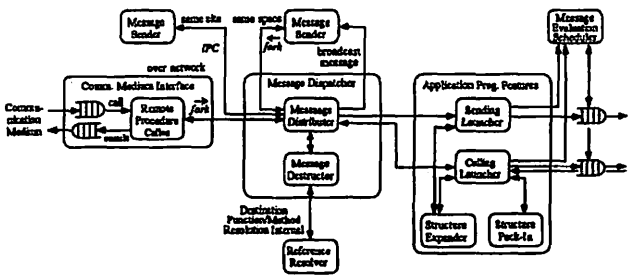


Fig. 3. Message-Passing Architecture



(a) Message Sender



(b) Message Dispatcher

Fig. 4. Message-Passing Operator

[2] Common Lisp のパッケージとは異なる．

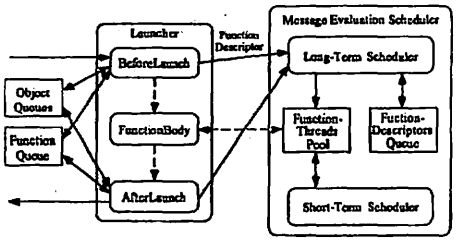


Fig. 5. Message Evaluation Scheduler

て，ネットワーク上のRPC，ノード内IPC，あるいは，プロセス内通信を選択して送信する．

並行オブジェクトは，アプリケーション空間において相互のリモート参照を管理しており，それによって高速メッセージ送信を実現する．メッセージの送信先の参照の管理とメッセージ送信時の参照解決の手法については，4.1.で述べる．

メッセージ受信では，受信オブジェクトを特定し，その処理を行なうための関数記述子を生成し，スケジューラに通知して実際の処理を行なう．関数記述子は，分散関数とオブジェクトメソッドの呼出あるいは共有変数のアクセスを起動するための情報と，メッセージの時間属性と優先度を保持したものである．

スケジューラは，メッセージ受信部から関数記述子を受け取り，長期および短期のスケジューリングを行なう(Fig.5)．分散オブジェクトでのメッセージ評価の排他性の処理は，スケジューラ前段の関数起動部で行なっている．メッセージの時間属性としては，送信タイムや，評価のデッドライン計算の非精密性の許容度，メッセージの同期性と周期性といったものがあるが，メッセージ評価の起動のための長期スケジュールは，現時点では送信タイムと優先度に基づいて行なっている．そこで可動状態となったメッセージ評価は，その関数記述子から実行スレッドを生成し，それらの実際の駆動である短期スケジュールは，OS のプロセスとスレッドの制御によって行なっている．

3.2. プログラミングインタフェースと多言語対応

並行オブジェクトとメッセージ通信のためのプログラミングインタフェースをメッセージ通信機構上に用意することによって，アプリケーション言語として複数のプログラミング言語に対応できる．プログラミング上に提供するのは，並行オブジェクトの基本定義，メッセージ送信関数(Send, SyncCall, Call)，プログラム配置定義，参照の輸出と輸入の宣言の機能である．

現在我々はアプリケーション言語として，C++[19]と Common Lisp[20]をサポートしている．[3] C++と Common Lisp のいずれも計算は，並行オブジェクト間は並列に，並行オブジェクト内では逐次的に行なう．

Common Lisp オブジェクト(CLOS オブジェクト)を複数引数に持つメソッドが呼ばれたとき，前述のように計算スコープは局所的であり，直接アクセスできる分散オブジェクトは高々ひとつであり，残りはリモート参照子である．

3.3. データ構造体の処理

並行オブジェクトは共有記憶を持たないので，メッセージ上でデータ構造体を扱うには，データ構造を論理形式で処理しなければならない．データ構造をトレースする機能と，トレースされた構造体要素を論理形式に変換する機能，論理形式から元のデータと同じ構造体を生成する機能とから成る．

これらの機能は Common Lisp では，データの動的型チェックが可能でありプラットフォームが提供する．C++ではトランスレータによって，トレース機能とデータ生成機能を予めソースコード上に実現する．但し現時点では，トレースとデータ生成のための補助機能をユーザ定義により提供する．

3.4. 連続メディアの処理

ストリームオブジェクトは，Fig.6 に示したような

[3] さらに Java への対応を検討している．

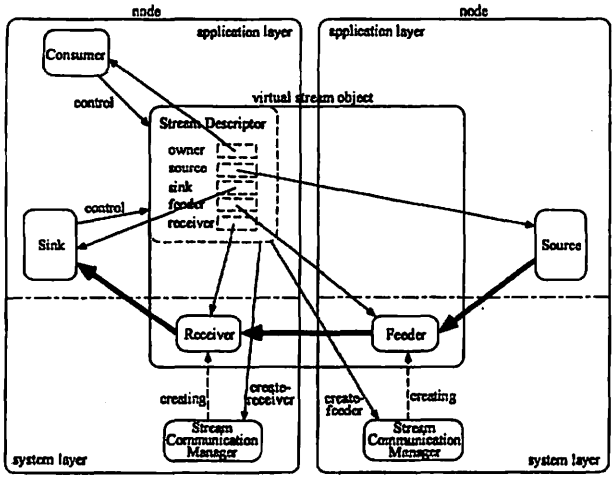


Fig. 6. Continuous-Media Stream Object

構造を持つ．ストリームオブジェクトに対してストリームシンクが割り当てられたとき，オブジェクトは自動的にストリーム配送機構を生成し，データ転送可能な状態とする．停止や早送りといったストリームの処理は，オブジェクトに対して行なえば，オブジェクト内で自動的に配送機構の各モジュールを適切に制御する．ストリームオブジェクトは，ネットワーク上を転送可能であり，ストリーム配送機構は必要に応じて生成する．このストリーム配送機構は，我々が開発しているネットワークスケーラブルなデータ配送機能を用いて構成する．[21, 22]

ストリームオブジェクトは，データ構造体の中で他のストリームオブジェクトと組み合わせて使用されたとき，オブジェクトの相互機能としてメディア間同期の機能を提供する．[4]

4. 空間管理と参照解決

並行オブジェクト間のメッセージは，オブジェクト相互間のリモート参照をアプリケーション空間内で管理することにより，高速の通信を実現する．

4.1. 参照・名前空間とその管理

並行オブジェクトを特定するリモート参照は，プロセスと並行オブジェクトの識別子の組で表現する．プロセスは，ネットワークノードとノード内の識別子で表現する．これらの識別子は，論理名をつけて扱うことも可能である．但し，名前を扱うことによる応分のオーバヘッドを伴う．

リモート参照の識別子は，参照先のオブジェクトを保持しているプロセス，あるいは，プロセスを保持しているプラットフォームで一意に特定したものを輸入して得る（次項参照）．ノードは，ネットワークシステムの物理的識別子あるいは名前で表現する．

Fig.7 に，リモート参照の管理と参照解決の論理機構を示す．論理名によるリモート参照も同じ機構上で扱うが，構成が輻輳するため，識別子による管理と解決のみ示してある．

メッセージ送信のパスは，通信メディアの指定によって，送信時に決定する．ノード間でのメッセージの通信は，多様な通信メディアの種類に対応することができる．[5]

メッセージ受信側では，送信側から送られたリモート識別子からそのプロセスのローカル識別子に変換し，プロセス内の関数やオブジェクトメソッドを特定する．メッセージ送信に関する参照表のうち，リモート参照表とローカル参照表をそれぞれ，Fig.8 に示す．

本システムの空間管理では，任意の並行オブジェクトやプロセスをまとめてドメイン空間を構成することができる．ドメインはまた，別のドメインを含んでもよい．ドメインはアプリケーション空間を跨っていてもよく，個々のアプリケーション空間を超えた計算空間を提供する．

---

[4] これは現在アプリケーション制御での同期によって行なう．上述のデータ転送機構においてフレーム間同期などの機能を組込機能として実現する予定である．

[5] 通信メディアは現時点では（狭義の）インタネットにのみ対応し，ONC (Open Network Computing) RPC を使用している．

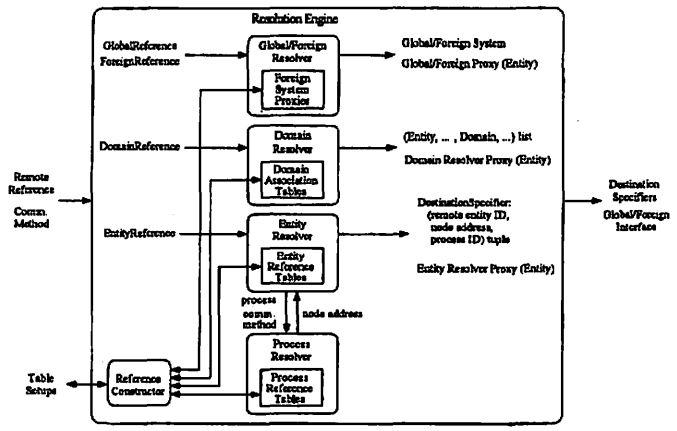
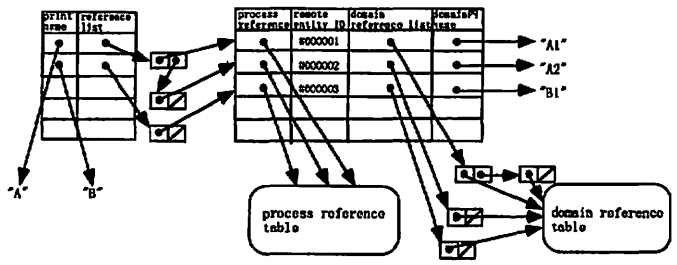
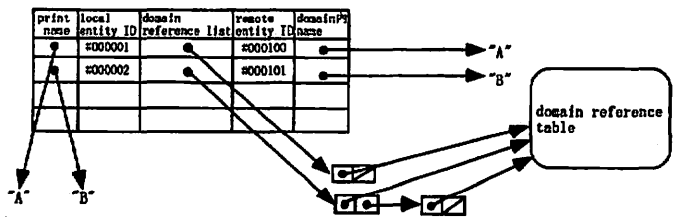


Fig. 7. Reference Resolution and Resolution Engine for Message Sending



(a) Remote Entity Reference Table



(b) Local Entity Reference Table

Fig. 8. Entity Reference Tables and ID Transformation

4.2. アプリケーションプログラムの配置とダウンロード

アプリケーションプログラムはプログラムパッケージを単位として，コンフィグレーション指定によって分散配置する．プラットフォームは，パッケージの配置指定に従ってプロセスを生成する．プロセスは，並行オブジェクトとして定義されたモジュールにローカル識別子と外部から参照するためのリモート識別子を割り当てる．パッケージが外部の並行オブジェクトを参照している場合は，コンフィグレーションで指定された対応するプロセスに問い合わせてそのオブジェクトのリモート識別子を取得し，リモート参照を生成する．この手続きによって，相互参照に関する管理表を作成する．

この相互参照の生成は動的に行なうことが可能であり，アプリケーションの動的でインクリメンタルなダウンロードを可能とする．

5. まとめ

本論文では，我々が開発しているマルチメディア並行計算プラットフォームについて述べた．このプラットフォームは，ネットワーク環境上で，高速メッセージ通信を用いた並行計算系を実現し，そこで稼働するアプリケーションには柔軟な空間構成とネットワークワイドな連続メディア処理の機能を提供する．ネットワーク環境は異なる種類のものが混在していたり経路が多重化していてもよく，多様な環境上でのマルチメディアアプリケーションの構築を促進できる．ネットワーク空間では個々のアプリケーション空間の他に特定のドメイン空間の構築が可能であり，柔軟でオープンなシステムの構築をサポートする．

このシステムは現在，SparcStation20/Solaris2.4 と 10base-T Ethernet の環境上で，C++とCommon Lispをアプリケーションプログラミング言語とするプロトタイプを構築中である．

今後機能拡張を進めるとともに，プログラミングにおける多言語対応，リアルタイム処理，マルチメディア処理サポートの機能を充実させ，所期の目的にそった検証を行なう予定である．

謝辞

本研究の遂行にあたって多大なる助言を頂いた五十嵐達治部長を始め当研究所諸氏に，感謝いたします．

参考文献

[1] 前川，"HDマルチメディア情報サービスプラットフォーム—非同期ネットワーク計算と連続メディア処理—，" 情処第 52 回全国大会講演論文集, vol. 3, pp. 225-226, Mar. 1996.

[2] 萩原 他，"HDマルチメディア情報サービスプラットフォーム—システムコンセプト—，" 情処第 52 回全国大会講演論文集, vol. 3, pp. 223-224, Mar. 1996.

[3] 萩原 他，"HDマルチメディア情報サービスプラットフォーム，" 画像電子学会 第6回メディア統合技術研究会 MT-6-S1-2,1996.

[4] R. M. Adler, "Distributed Coordination Models for Client/Server Computing," IEEE Computer, vol. 28, no. 4, pp. 14-22, Apr. 1995.

[5] T. G. Lewis, "Where Is Client/Server Software Headed?" IEEE Computer, vol. 28, no. 4, pp. 49-55, Apr. 1995.

[6] E. Yourdon, "Java, the Web, and Software Development," IEEE Computer, vol. 29, no. 8, pp. 25-30, Aug. 1996.

[7] D. Kramer, The Java Platform: A White Paper, Mountain View, USA: Sun Microsystems, 1995.

[8] The Common Object Request Broker: Architecture and Specification, New York, USA: Wiley, 1991.

[9] A. Beguelin, et. al., "A Users' Guide to PVM (Parallel Virtual Machine)," ORNL/TM-11826, Oak Ridge National Laboratory, Oak Ridge, USA, Jul. 1991.

[10] M. Snir, et. al., MPI: The Complete Reference, Cambridge, USA: MIT Press, 1996.

[11] A. S. Grimshaw, "Easy-to-Use Object-Oriented Parallel Processing with Mentat," IEEE Computer, vol. 26, no. 5, pp. 39-51, May 1993.

[12] 小中 他，"超並列オブジェクトベース言語 Ocore の並列計算機上での実装，" 情処論文誌, vol. 36, no. 7, pp. 1520-1528, Jul. 1995.

[13] 塚本 他，"クラスの共有と配送に基づくオブジェクト指向分散システムの設計と実現，" 情処論文誌, vol. 37, no. 5, pp. 853-864, May 1996.

[14] B. Achauer, "The DOWL Distributed Object-Oriented Language," Comm. ACM, vol. 36, no. 9, pp. 48-55, Sep. 1993.

[15] R. Chandra, et. al., "COOL: An Object-Based Language for Parallel Programming," IEEE Computer, vol. 27, no. 8, pp. 13-26, Aug. 1994.

[16] A. Yonezawa, et. al., "Implementing Concurrent Object-Oriented Languages on Multicomputers," IEEE Parallel and Distributed Technology, vol. 1, no. 2, pp. 49-61, May 1993.

[17] 白鳥 他，"やわらかいネットワークの開発に向けて—知識型設計方法論—，" 情処研究会報告 DPS62-11, pp. 79-86, Sep. 1993.

[18] 富永，"フレキシブルネットワーク—総論—，" 電子情報通信学会誌, vol. 77, no. 4, pp. 350-354, Apr. 1994.

[19] B. Stroustrup, "An Overview of C++," SIGPLAN Notices, vol. 21, no. 10, pp. 7-18, Oct. 1986.

[20] G. L. Steel, Jr., Common Lisp: The Language, Bedford, USA: Digital, 1990.

[21] 谷，"HDマルチメディア情報サービスプラットフォーム—スケーラブルなマルチメディアデータ配送—，" 情処第 52 回全国大会講演論文集, vol. 3, pp. 227-228, Mar. 1996.

[22] 谷 他，"HDマルチメディア配送システムの開発—リンク接続による End-to-End Qos 管理機構—，" 情処本ワークショップ論文集, Oct. 1996.