# 分散協調型計算ミドルウェアと マルチメディア共有操作環境の開発

Development of Distributed Collaborative Computation Middleware and Multimedia Collaborative Manipulation Environment

斉藤隆之 [1], 安斎利洋 [2], 中山真樹 [3], 芦野俊宏 [4], 中村理恵子 [5], 岩元隆幸 [6], 古川令子 [7], 柴田司都 [8]
Takayuki SAITO, Toshihiro ANZAI, Maki NAKAYAMA, Toshihiro ASHINO, Rieko NAKAMURA, Takayuki IWAMOTO, Reiko FURUKAWA, Shidu SHIBATA

1) (株)情報数理研究所 (〒171-0014 東京都豊島区池袋 2-43-1 池袋青柳ビル tsaito@imslab.co.jp)
2) 〒179-0084 東京都練馬区氷川台 2-7-27-201 unxi@ja2.so-net.ne.jp
3) 〒146-0092 大田区下丸子 4-5-14-203 PEC03611@nifty.ne.jp
4) (株)情報数理研究所 (〒171-0014 東京都豊島区池袋 2-43-1 池袋青柳ビル ashino@imslab.co.jp)
5) 〒206-0012 多摩市貝取 1-34-1-708 rieko@tt.rim.or.jp
6) (株)情報数理研究所 (〒171-0014 東京都豊島区池袋 2-43-1 池袋青柳ビル iwamoto@imslab.co.jp)
7) (株)情報数理研究所 (〒171-0014 東京都豊島区池袋 2-43-1 池袋青柳ビル rtanaka@imslab.co.jp)
8) (株)情報数理研究所 (〒171-0014 東京都豊島区池袋 2-43-1 池袋青柳ビル mitani@imslab.co.jp)

**ABSTRACT.** ANCL, Ad-hoc Network Computing Library, is the middleware designed to be an infrastructure for distributed collaborative computation. ANCL is mainly intended to be used with mobile PC and wireless network and it enables to establish ad-hoc computer cluster dynamically and enables to share information between PC's by messaging and shared object. On the top of ANCL, we developed graphic centric collaboration environment Interwall. Interwall gives shared canvas for PC's on ad-hoc cluster and rich drawing facilities to interact for members. In this paper, system design of ANCL and Interwall is described. Further, new programming paradigm to develop distributed collaborative systems which introduced by ANCL is presented.

## １．背景

IEEE802.11 や Bluetooth に代表される無線ネットワーク技術の普及に伴い、各個人が携帯するコンピュータ同士が近距離で連携しあって機能する分散システムが実現可能となってきた。そのアプリケーションとしては、携帯型コンピュータ間での情報の共有と同期をベースとしたプレゼンテーション、ブレインストーミング、ファイルの共同編集などが考えられる。これらのアプリケーションが、任意の場所で任意のコンピュータを持ち寄るだけで利用可能になれば、それは企業、学校、コミュニティにおける情報共有やコラボレーション支援に対する有効なソリューションになると考えられる。

このような分散システムの特徴はアドホックな情報共有である。アドホックな情報共有の第一の条件は、それぞれの計算機が情報を共有する任意の計算機と自由に通信することによって、通信のハブまたは情報共有のサーバとして特化したシステムの存在を排除することである。第二は、動的な構成員の参加と離脱を可能とすることであり、これによって生じる共有情報の欠損、矛盾の可能性などに対処することを可能とするシステム構成である。

システムの構成はあらかじめ決定されておらず、任意の場所で任意のコンピュータが出会い接続することによりシステムが構成される。かつ、接続は１対１ではなく、同時に多数のコンピュータが連携できる必要がある。そして、コンピュータはインクリメンタルにシステムに参加でき、またシステムから離脱できなくてはならない。システムからのコンピュータの離脱は、ソフトウェア的あるいはハードウェア的障害やネットワークからの切断によっても起こりうる。特に、無線ネットワーク環境においては、通信環境の悪化や通信圏外への移動などの原因により、ネットワークの切断が容易に起こる。したがって、いずれかのコンピュータの安定した存続を仮定することはできない。

従来の分散システムの構成法としては、単純なクライアントサーバーモデルに基づくものと、サービスを提供するオブジェクトをネットワーク上に分散配置し、クライアントからのサービス要求を適切なオブジェクトに中継する ORB(Object Request Broker)あるいは RMI(Remote Method Invocation)がある。これら従来の構成法では、サーバーあるいはオブジェクトの配置と運用があらかじめ適切に行われている必要があり、また、サーバーやオブジェクトがクライアントとの通信中に消失する状況に対して脆弱であるため、アドホックネットワークにおける情報共有には、従来のものに代わる新しい分散システムの構成法が求められる。

## ２．目的

我々は、このアドホック性を特徴とする分散システムを分散協調型計算システムと呼び、アドホックネットワークにおけるこの分散協調型計算システムの構成法を開発し、これに基づくソフトウェア実装として、分散協調型計算ミドルウェア ANCL(Ad hoc Network Computing Library)を開発した。さらに、このミドルウェアの有効性を検証するためのアプリケーションとして、グラフィカルなコラボレーション環境を提供するシステム Interwall を開発した。Interwall は、アドホックに接続したコンピュータ間に共有キャンバスを設営し、描画、画像、文字を織り交ぜた多彩な表現が可能なコラボレーション環境を提供する。

本稿では、ANCL の設計指針と実現した機能の概要について述べるとともに、これら機能をどのように活用して Interwall を開発したかについて述べる。また、Interwall の応用事例についても紹介する。

## ３．分散協調型計算ミドルウェア

### （1） 要件、設計、実装方式

ANCL の設計を進めるにあたり設定した要件について述べる。前節において述べたアドホックな情報共有実現に必要な条件は、システム設計の単位として以下のようにまとめることができる

- 任意個のノードがひとつのシステムを構成できること
- サーバーや外部のサービスに依存しないこと
- 新しいノードが随時にシステムに参加できること
- いずれのノードも随時にシステムから離脱できること

ここで、ノードとは、ANCL の上に構築されたアプリケーションプログラムの実行主体(プロセス等)を意味し、ハードウェアとしてのコンピュータとは区別して用いている。ひとつのコンピュータの中に複数個のノードが存在してもよい。

さらに、実用性や利便性を考慮してつぎの要件を加えて設定した。

- さまざまなプラットフォームで機能するか移植が容易であること
- アプリケーションプログラムのインストールと起動が容易であること

サーバーや外部のサービスに依存しないためには、システムが必要とするすべての機能をノードに内在させる必要がある。そして、それら機能をノード群に遍在させることによって、任意のノードが随時にシステムから離脱可能であることを実現するアプローチをとった。また、個々のノードが機能的に均質になるように設計することで、機能を遍在させている。ハードウェアや OS の非均質性はあるが、機能的には全てのノードは均質である。

すなわち、ANCL では、機能的に均質なノードが相互に接続しあうことにより分散システムのすべてが構成される。そこには、サーバープロセスやデーモンプロセスと一般に呼ばれるような他のノードによって代替することの不可能な機能を持ったノードは存在しない。

ANCL は、分散システムの構成要素(ノード)の均質性によってアドホック性と変動性に対応する。均質性がもたらす単純さは、システムのインストールや設定を簡略化する。

マルチプラットフォーム性を実現するために、開発および実行環境として Java を選択した。Java1.1 で開発したが、Java2 環境下でも使用できる。後述する Interwall は Java2 で開発した。ネットワーク環境としては TCP/IP を前提として開発した。IEEE802.11 準拠の無線 LAN のアドホックモードで利用することを重要な環境要件としたが、IEEE802.3 あるいはイーサネットでの利用も可能である。また、他のネットワーク環境への移植も可能である。

ANCL は以下の機能から構成した。(図 1)

- クラスター形成機能：ノードが自律的に集合体を形成する機能
- メッセージ通信機能(メッセージチャネル)： ノード間のメッセージ通信の機能
- オブジェクト共有機能(Object Sharing Space: OSS): データのノード間共有の機能
- アプリケーション起動機能(Application Manager): アプリケーションのインストールと起動を容易にする機能
- ノード間同期制御機能(Lock Manager)
- モニターユーティリティ: アプリケーション開発支援のためのモニタープログラム群

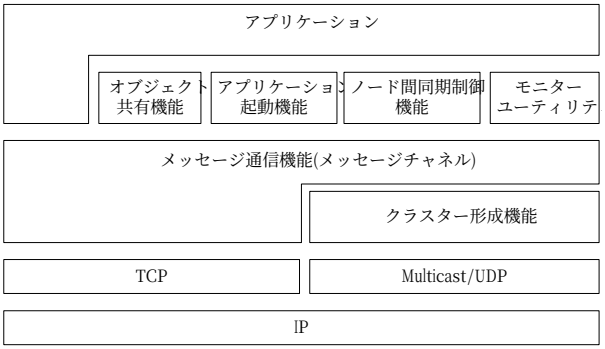


**図１** ANCL の機能構成

以下の節では、これらの機能の概略を述べる。

### （2） クラスター形成機能

ANCL では、ひとつの分散システムを構成するノードの集合体をクラスターと呼ぶ。ノードが稼動を始めたとき、自分が属すべきクラスターについての事前の設定はなされていない。ANCL は、ノードがお互いを自律的に

発見してクラスターを認識し、かつノードの参加と離脱を補足する機能－クラスター形成機能をもつ。

クラスター形成機能は、ノード情報と呼ばれるノードの属性情報を納めたパケットを IP マルチキャストを使用して定期的に反復して送信する。ノード情報は、IP アドレスと TCP の接続要求待ちポート番号など通信に関する情報と、ユーザー名やクラスター名などの各種識別名から構成される。

自ノードのノード情報を送信するのと並行して、他ノードからのノード情報を受信し、自ノードと同一のクラスター名をもつノード情報をテーブルに登録する。テーブルに登録されたノード群がクラスターとして認識される。このテーブルのレコード(ノード情報)には、寿命が設定されており、一定時間同じノード情報が受信されない状態が続くと削除される。このタイムアウト機構によって、障害やネットワークからの切断によるノードの離脱を検出する。

クラスター形成機能は、ノードの参加と離脱を検出すると、これをイベントとしてイベントリスナーに通知する。イベントリスナーは、ANCL 内部のオブジェクトのほかアプリケーションのオブジェクトであってもよく、API を通じて自由に設定できる。イベントリスナーは、このイベントを通じて最新のクラスターの状態を取得できる。

（3） メッセージ通信機能

ノードの集合としてのクラスターが認識できると、それらノード間でメッセージを配送するためのネットワークを形成する。このネットワークは、ノード間を TCP 接続でつないだもので、これをメッセージチャネルと呼ぶ。メッセージチャネルのトポロジーとしては、全結合や木構造があり、特にスケーラビリティの面からはスパンニングツリーが適している。ノードの識別子のリストから求めたトポロジーに基づいて、各ノードは接続すべき相手ノードを決定し、接続を行う。ノードの参加あるいは離脱が発生した直後では、情報の遅延によってノード間でクラスターの認識に不整合が生じることがあるが、時間の経過とともにこの不整合は解消する。

２本以上の接続をもつノードは、メッセージの中継ノードとして機能し、クラスター内の任意のノード間でのメッセージ通信が可能である。メッセージ配送には、単一ノードを宛先とするユニキャスト型と、全ノードに同報されるマルチキャスト型がある。メッセージには、イベントとしての種類をあらわす識別番号が付与されており、受信されたメッセージは、この識別番号と対応付けられたノード内のオブジェクト(イベントリスナー)にイベントとして伝えられる。この識別番号とオブジェクトとの対応付けは、API を通して自由に設定できる。また、識別番号は、個々のメッセージ毎に送信時に API で設定する。

メッセージチャネルは、クラスターに属すノードだけで構成された閉じたネットワークであり、そのメッセージ通信機能は、ノード間で情報の共有と状態の同期を行うための基盤である。また、メッセージ受信をイベントとして伝える機能は、メッセージ駆動型(イベント駆動型)のプログラミングパラダイムを与える。

（4） オブジェクト共有機能

メッセージ通信が、ノード間での情報の共有と状態の同期の実現基盤であるが、これに加えて ANCL は、ノード間でのデータの共有機構(Object Sharing Space. 以下 OSS)を提供する。OSS は、Java オブジェクトをノード間で共有することができる仮想的なオブジェクトプールであり、オブジェクトの登録、取得、削除、置換の操作がある。オブジェクトの登録の際には、名前やクラス名などの属性情報(ラベル)を付与し、このラベルをキーとして取得、削除、置換の各操作を行う。

共有対象であるオブジェクトの参照情報をノード間で交換するプロトコル、交換した参照をもとにオブジェクトを取得するプロトコル、オブジェクトの削除と置換のプロトコルは、すべてメッセージチャネルのメッセージ通信で実現した。オブジェクトの参照情報の交換プロトコルは、各ノードが保持しているオブジェクトの参照表をマルチキャストメッセージで同報することによって実現した。また、取得のプロトコルは、ユニキャストメッセージで実現した。削除と置換のプロトコルについては、操作対象オブジェクトの参照をマルチキャストメッセージで全ノードに送信することによって実現した。

登録、削除、置換の操作によりオブジェクトプールに変化があると、イベント通知が行われる。アプリケーションは、このイベントをもとに新しく登録されたか、あるいは置換されたオブジェクトを取得したり、削除されたオブジェクトを特定することができ、これに基づいてノードの内部状態を遷移させることができる。

オブジェクトの取得がノード間で行われると、そのオブジェクトの複製が取得した側のノードに生成したことになる。OSS は、この複製オブジェクトも新たな取得操作の対象となるように、参照情報の交換を行う。取得をトリガーとして複製オブジェクトがノード間に伝播する。つまり、取得頻度の高いオブジェクトほど、複製オブジェクトがより多くのノードに存在することになり、ノードの消失に対する耐障害性が得られる。

OSS では、登録され OSS の管理下に置かれたオブジェクトのメソッドをローカルあるいはリモートから呼び出したり、その内部状態を変更することは禁止した。オブジェクトはノードに局在化した存在であるため、そのメソッド呼び出しを許すことは、ノードの消失に対する脆弱性の原因となるからである。OSS に登録されたオブジェクトの内部状態を変更するには、新しいオブジェクトによる置換操作によって行う。

（５） アプリケーション起動機能

分散協調型計算システムを利用するときに、はじめに問題となるのは、システムを構成する各ノードがお互いに整合性の取れたバージョンのプログラムで機能していなければならないことである。従来の分散システムでは、サービス提供者や運用管理者の操作によるか、もしくはユーザー間の調停とユーザー個々のインストール作業によって、プログラムバージョンの整合性を確保することが行われてきた。これに対して、我々が目指す分散協調型計算システムはコンピュータが集合しただけで即座に使えることが望まれるため、ユーザー間の調停作業やプログラムのインストール作業を可能なかぎり自動化することが要求される。

ANCL では、Java のインクリメンタルなクラスローディングの仕組みと OSS を活用して、ノード間でアプリケーションのクラスを共有し、必要に応じてこれを取得して起動する機能－アプリケーション起動機能を実現することによってこの問題を解決した。ノードは、アプリケーション起動機能を ANCL ミドルウェアに対するアプリケーションと位置付けて起動する。アプリケーション起動機能は、GUI を備えており、ローカルもしくは OSS に登録されている JAR ファイルを選択、取得し起動することができる。ひとつのノードが、ローカルに保持している JAR ファイルからアプリケーションをローディングして起動するとき、この JAR ファイルを OSS に登録する。他ノードでは、OSS に登録された JAR ファイルをアプリケーション起動機能の GUI を通じて選択、取得し、この JAR ファイルからアプリケーションを起動する。

このアプリケーション起動の方式によって、最低ひとつのノードがアプリケーションプログラムを持っていれば、他のノードはあらかじめそのプログラムを持っていなくても、そのアプリケーションプログラムによる分散協調型計算システムに参加することができる。

（６） ノード間同期制御機能

メッセージチャネルと OSS が提供しているのは、ノード間での並行かつ非同期な計算方式である。これに加えて ANCL では、その適用領域を広げるためにノード間での同期機構を開発し、資源競合を伴うアプリケーションに対して解決手段を提供することにした。

この同期機構(LockManager)は、モニターオブジェクトを保持するリーダーノードをノード間で協調して選出する。全ノードは、このリーダーノードに存在するモニターオブジェクトとメッセージで通信することにより同期処理を実現する。リーダーノードあるいはモニターオブジェクトの所在は、アプリケーションからは隠蔽される。そして、モニターオブジェクトは、その複製がノード間で作られており、リーダーノードの交代に対応することができる。

（７） アプリケーション開発ユーティリティ

ANCL は、アプリケーション開発を支援する目的で、クラスター、メッセージチャネル、OSS の状態をモニターする以下のユーティリティを提供する。

- クラスターに属すノードのリストを表示する機能
- メッセージチャネルのグラフ構造を表示する機能
- 受信したメッセージのログを表示する機能
- OSS に登録されているオブジェクトの属性情報を表示する機能

これらの機能は、ANCL の各機能からのイベントに基づいてリアルタイムに表示を更新する。これらの機能は、単体で使用できるほか、ANCL の API クラスとともにクラスライブラリとして提供されているので、アプリケーションに組み込んで使用することもできる。

（８） 他のミドルウェアとの比較

従来、分散システム構築のミドルウェアとしては、CORBA 準拠 ORB[4]，Jini[5]，Java RMI に代表される分散オブジェクト指向のシステムと、IBM MQSeries，Java Message Queue[6]に代表されるメッセージシステムがある。以下、これら従来のミドルウェアを、我々が目的とする分散協調型計算システムに適用するときの問題点について述べ、ANCL の有効性をまとめる。

ORB に基づくシステムでは、アプリケーションシステムを構成する機能(サービス)のそれぞれがオブジェクトとして実装され、所定のノードに配置される。ORB は、オブジェクトの所在を管理し、クライアントとオブジェクトの間の通信を仲介する。ORB によって、クライアントがオブジェクトの所在を知る必要はなく、また、オブジェクトの配置変更と、耐障害性や負荷分散のためのオブジェクトの多重化も可能である。しかし、ORB そのものとオブジェクトの整合性のとれた配置と運用が前提となっているため、企業基幹系のような管理運用体制を整備できる基盤型システムの構築には適しているが、システム全体を動的にアドホックに形成することには適していない。

Jini やそのベースとなっている Java RMI においても、ミドルウェアがクライアントと分散オブジェクトとの通信を仲介する。Jini においては、分散オブジェクト(サービス)の所在の登録を受け付けるとともに検索要求に応答するディレクトリサーバーを動的に設営できるため、CORBA に比べてシステム形成の柔軟性に富み、家電ネットワークなど小規模で構成変更の頻度が高いコンスーマサイドのシステムを構築するのに適している。しかし、クライアントがサーバーである分散オブジェクトに要求を送信して応答を受け取るというパラダイムであるため、分散オブジェクトが存在するノードが障害や離脱によって消失する状況への対応は脆弱となりがちである。この問題は CORBA においても同様である。

一方、メッセージシステムでは、メッセージ配送を集中管理するいくつかのメッセージ配送サーバー(メッセージルーター)が相互接続することによりバックボーンを構成し、末端のノードがこれらメッセージルーターに接続することによってノード間のメッセージ配送が実現される。したがって、メッセージシステムでは、あらかじめメッセージルーターを配置してメッセージ配送のバックボーンを構築し運用しておく必要がある。このことが、CORBA と同様に、その適用範囲を企業基幹系のような基盤型のシステムに限定している。

ANCL は、我々が目的とするアドホック性と変動性を有した分散協調型計算システムを実現するために、従来の分散システムミドルウェアの課題を解決すべく開発された。今後、さまざまなアプリケーションへの適用が期待できる。

## ４．グラフィカルコラボレーション環境 Interwall

### （１） Interwall の着想

ANCL をミドルウェアとする最初のアプリケーションである Interwall は、The Wall(1998)[1]を前身とし、概念や実装の一部を引き継いでいる。The Wall は、「特別に企画されたアートセッションを支援する」という目的のもとに､安斎､中村によって発案されたソフトウェアで、劇場やスタジアムのような世界に唯一無二の場に制作者と鑑賞者が集まってくるというメタファーをもっている。したがって The Wall の設計思想は、クライアントサーバー構造と密に結びついている。

The Wall の応用事例は、限られたアーティストのためのセッションにという当初の目的に収まらず、設計者の意図を超えて教育現場などにも拡大していった。そのため、前準備を必要とするサーバーの設置などが障壁になってきた。また、学習者の自主性によってアドホックに開始されるセッションでは、特異な中心をもつ構造が矛盾を引き起こすことが予想された。

そこで、教育や企業活動を想定した The Wall の発展形が模索され、そこに開発中の ANCL が結びついて非サーバー型の Intewall が発想された。以下、The Wall の概説、Interwall への移行とその問題点、ANCL による解決について順に述べる。

### （２） The Wall の概要

The Wall は、Wall とよばれる円筒形の仮想壁面をインターネット上に構築し、複数のアーティストによる協調ペインティングを可能にするシステムである。円筒の側面には、Bark という不定形の描画オブジェクトが配置される。Bark はαチャンネルをもった半透明合成の可能なビットマップで、下層の Bark 群にα合成される。個々の Bark は、それぞれ参加メンバーの一人をオーナーとする。その結果、Wall という公的な場に対して個人がメッセージを投げかけるという対話的なコラボレーションが成り立つ。

Wall 上には、無数の Bark が円筒の各所で層状に堆積していき、その履歴を含めた全体をセッションと呼ぶ。セッションは年輪を重ねるように成長し、最新の壁面は Bark が樹皮のように重なり合って見える。(図 2) The Wall のクライアントは、Wall 上に Bark を重ねる描画編集機能と、Bark をはがして履歴をみる探索機能を備える。

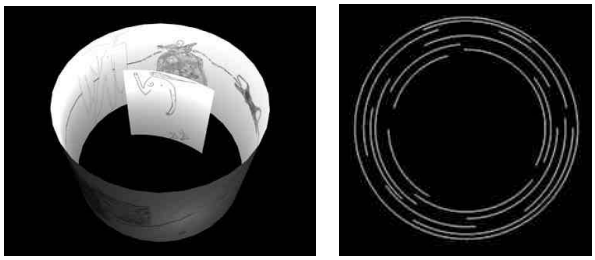

図２ wall の概念図

The Wall システムにおける Bark の描画機能を特徴づけるのは、インターロックメカニズムである。Bark の描画中は､この排他機構によって Bark リージョンに他の Bark リージョンが幾何学的に重なって予約されることがない。このため、Bark のオーナーは、安心して占有領域への描画に専念できる。描画が終了し Bark を Wall 上にコミットすることによって、この占有は解除され、そのリージョンは解放されて他の参加者がこのリージョンと重なった Bark を設定することが可能となる。

このように The Wall の参加者が描画中に自己の領域を占有することを可能とする Bark の概念は The Wall よりさらに先行する Moppet(1997)などの経験から導入されたものである。[2][3] Moppet では排他制御は存在せず、全てのユーザは同時にキャンバスのどこにでも筆を入れることができた。しかしその自由さの反面、個々のユーザは「もっと落ち着いて絵を描きたい」というフラストレーションをもった。その反省から作業プロセスを個人単位に分節化したのが Bark という単位であり､それを保証するのがインターロックである。共有ペイントシステムや共有ホワイトボードシステム等の類似アプリケーションとの決定的な違いがここにある。

### （３） クライアントサーバーモデルの利点

円筒の壁に参加者が集まってくる、という The Wall のメタファーは、前述のようにクライアントサーバーモデルと親和性が高い。The Wall の処理プロセスの要点をまとめると、以下のようになる。

<u>サーバ側</u>

クライアントのログイン・ログアウトの受付
クライアントから要求されたインターロックの排他制御
クライアントから描画済み Bark を受付
描画済み Bark から新しい壁面イメージを合成してク

ライアントへ送信

クライアント側

サーバーへログイン・ログアウトの要求
サーバーへインターロックの要求と、結果の受け取り
インターロック領域内への描画
描画済みの Bark をサーバーへ送信
サーバーから新しい壁面イメージの受信

これらの諸機能をさらにおおまかに分類すると、サーバー＝壁が担う基本的な役割は次の２点に集約されることがわかる。

a) 同期機能

サーバーはクライアントのロック要求に対して優先判定を行い、排他制御をおこなう。

b) データベース機能

サーバーは情報の持続性を保証する。参加者がゼロになっても、サーバーが走りつづけている限りセッションは継続する。

### （４） クライアントサーバーモデルの限界

以上のように、The Wall はサーバーと密接に設計されているが、同時にクライアントサーバーモデルが一般的に抱えるデメリットを継承することになる。それは次の３点に集約される。

a) サーバーへのロジック集中による脆弱性

すべてのロジックの進行をサーバーが一手におこなうため、サーバーが停止するとあらゆるユーザの行動が不可能になる。またサーバー上の情報の紛失は、全体の共有資産の紛失にもつながりかねない。

b) 情報分配のオーバーヘッド

システムのスループットが、サーバーのボトルネックに左右される。特にビットマップイメージを送受信する設計では、クライアントの待ち時間ストレスが大きい。

c) 運用管理の負荷

運用にあたって必ずサーバーを維持する必要があり、管理者なくしてはアプリケーションの利用さえままならない。

こういったクライアントサーバーモデルのもつ重い足かせを外し、機動性とフレキシビリティを得るため、先に述べたように非サーバー型で、管理の手間がかからず、「いつでもどこでも」共同作業の可能な The Wall である Interwall が考えられた。ここで問題になるのは、クライアントサーバーモデルの利点として先に述べた同期・排他制御とデータの持続性を、非サーバー型においていかに実現するかである。

### （５） ANCL による解決

The Wall のもつ特徴を損なわず、しかもサーバーのもつ特性をサーバーレスで実現するために、前章で述べた ANCL のアーキテクチャに基づいて The Wall が全面的に再設計され、Interwall の開発が開始された。Interwall と ANCL は並行して開発が進められた。このため、Interwall に仮想的なサーバー機能を提供することなどを中心に、ANCL の開発に数多くの要求がフィードバックされ、ANCL の設計と実装した機能の検証が行われた。

以下に、ANCL によってサーバー依存の機能がどのように代替されたかを列挙する。

a) ノード間同期制御機能によるインターロック制御

ANCL のノード間同期制御機能(Lock Manager)を利用することにより、中心的なサーバーが存在しない状況下においても、ノード間の排他制御をおこなうことができる。そのためサーバーが存在しない環境でもインターロック機構が実現した。(図 3)

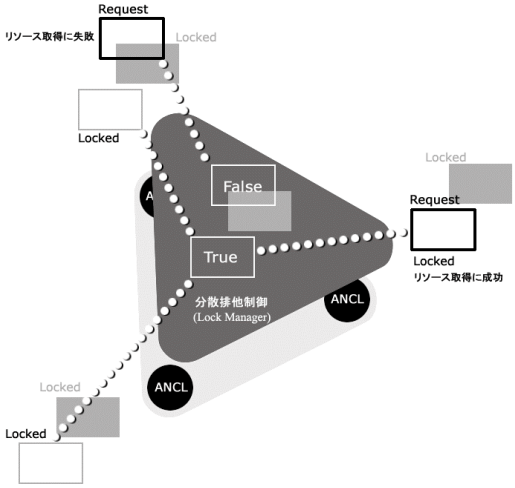


図３ LockManager によるインターロック制御

b) OSS によるアプリケーションデータの分散共有

Wall や個々の Bark などを個別に ANCL の OSS に登録するだけで、それは ANCL を通じてオブジェクトとして取り出すことができる。これを利用し、Interwall では各ノードがそれぞれ、オブジェクトとして取り出した Wall や Bark から自ら壁面イメージを合成する。もしそこにさらに新しい Bark が追加されれば、その Bark オブジェクトのみを OSS から取り出し、やはり各ノードが再合成する。(図 4) したがって、Wall の更新のために、全面イメージを再分配する必要がなくなり、通信時の障害への耐性が向上し、再分配のオーバーヘッドが殆どなくなるとともに、ユーザにとっての待ち時間が劇的に減少することとなった。

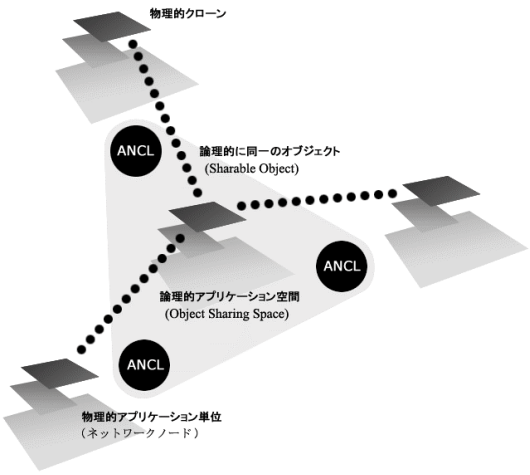


図４ OSS によるアプリケーションデータの分散共有

c） メッセージチャネルによるリアルタイムイベントの共有

ANCL のメッセージチャネルにより、揮発性の情報でもオブジェクトを直接やりとりすることができる。これによって、プロトコルベースのクライアントサーバーモデルでは実現が難しかった描画プロセスのリアルタイム共有も可能になった。(図 5)

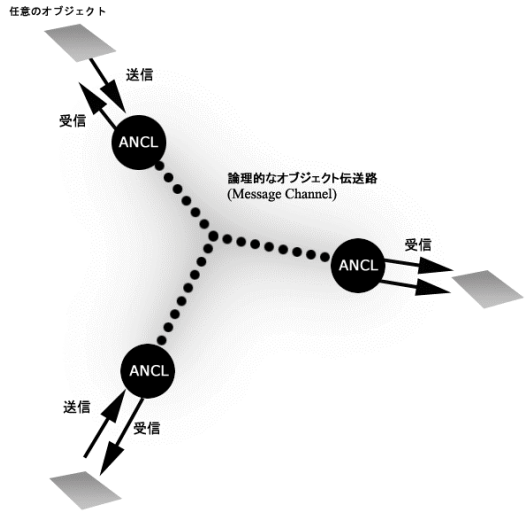


図５ メッセージチャネルによるリアルタイムイベントの共有

（６） Interwall の実行プロセス

実際の Interwall の動作のプロセスを以下に紹介する。

- 無線 LAN（もちろん有線 LAN でもよい）を装備したパソコン上で ANCL を起動する。
- TCP/IP リーチャブルなパソコン同士は自動的にクラスターを形成して相互認識する。
- ANCL から Interwall を起動する。起動後の Interwall においては、相互認識しているノード間におけるオブジェクトやメッセージの共有が ANCL によって自動的におこなわれる。
- 任意のノードにおいて Wall を生成する。すると他のノードでも Wall が現れる
- 任意のノードから Wall 上の任意領域を Bark としてインターロックする。その領域上にはペイントツールによる描画が可能になる。
- その Bark の領域は他のノードからも認識され、その領域上に対しては排他制御が働き、新しい Bark のインターロックができない。
- 描画が終わった Bark は、占有を解いて Bark を OSS に登録する。
- その新規 Bark だけが新しいオブジェクトとして追加され、それを取得した各ノードが Wall を再合成して、結果的に同じ Wall の状態に更新される。
- 解放された領域は、また新しいインターロックを受け付けることができる。領域が重ならない限りは、同時にいくつでもインターロック領域は Wall 上に存在できる。
- メニューから"リアルタイムスイッチ"を入れると、一筆ごとに描画イベントがメッセージとして共有され、ある Bark 上の描画プロセスが、全ノード上でリアルタイムに共有される。

以上のように、Interwall では、サーバーという中央集約的な存在に依存せずに、複数のノードがコラボレーションの空間の状態を協調して更新してゆくことが、ANCL をベースに実現されている。なお、ここに紹介した概念や手法は、Interwall という特定のアプリケーションに限らず、さまざまなアプリケーションに有効であると考えられる。

（７） Interwall の応用事例

Interwall の応用事例として、神奈川県横浜市立大口台小学校で実施された屋外での体験学習を紹介する。2000 年 10 月、横浜市金沢区野島公園において「野島クエスト」と名付けられた授業が行われた。高学年の生徒 138 名は 24 の班に分かれ、各班にモバイル PC 一式が配布された。Interwall のアドホック性を活かすため、これらモバイル PC を歩行しながら操作できる装備と環境を提供した。その装備は以下のようなものである。(図 7)

- ペン入力が可能なノートパソコン
- 無線 LAN
- 補助バッテリー
- 画板型フレーム
- 背負い式モバイルバッグ
- デジタルカメラ


図７ Interwall セッションのための装備

各班は既定のテーマに沿って、野島公園内のスナップショットをデジタルカメラに収めた。無線 LAN の到達圏内にあるノードは、Interwall の共有空間に画像やコメントをレイアウトし、体験を共有することができた。(図 8)

図８ 野島クエスト

## ５．まとめ

アドホックネットワーク上での分散システムを構成するための分散協調型計算ミドルウェア ANCL を開発した。ANCL によって、機能的に均質なノードが自律的に接続しあい、ノード間でメッセージ通信、データ共有、同期制御を行う基盤環境が提供できる。ANCL が提供する機能を使用することにより、サーバーのような中央集約的な存在を排除したノード同士の協調的な情報共有と同期をベースとするアプリケーションを構築することができる。

この ANCL のアプリケーションとして、グラフィカルなコラボレーションシステム Interwall を開発し、ANCL の有効性を検証した。Interwall は、任意のノードがアドホックに協調して円筒形の仮想壁面を形成し共有する。この壁面上で描画オブジェクトを対話的に堆積させることができるグラフィカルなコラボレーション環境を提供する。

Interwall のコラボレーションシステムとしての有効性を検証するために、教育現場での応用を行った。教育現場という多様な面での柔軟性が要求される環境において Interwall は、従来のクライアントサーバー方式のシステムにはない自由度を実現した。Interwall は、新しいタイプのグラフィカルコラボレーションシステムとして、その発展と応用が期待できる。

ANCL は、Interwall の開発においてその目標を達成したが、現状の ANCL がアドホックネットワークにおける分散協調型計算システムが抱えるすべての問題に対処できているわけではない。たとえば、クラスターの分離と結合に際して、共有情報の整合と統合の方法が未解決の問題であり、今後の研究課題である。

今後、さまざまなタイプのアプリケーションに対して ANCL の適用を試みていくことによって、そのプログラミングパラダイムと機能をより精緻なものにするとともに実用化と普及を目指す。

## ６．謝辞

Interwall の教育現場への適用に際しては、現場経験の豊富な中川一史氏（金沢大学）、佐藤幸江教諭（横浜市立大口台小学校）、河崎睦教諭（綾瀬市立綾西小学校）に少なからぬ助言と助力を頂戴した。また、横浜市立大口台小学校生徒諸君の豊かな遊び心は、システムの開発と応用における問題抽出の手助けになったばかりでなく、われわれの創意にも力を添えてくれた。鎌田恭彦（映像作家）、遊佐辰也（写真家）の両氏には芸術性の高い記録映像を残していただいた。感謝の意を表したい。

## ７．参加企業及び機関

株式会社情報数理研究所

## ８．参考文献

[1] 安斎利洋, 中村理恵子: 連画コラボレーションを支援するパノラマ空間ペイントシステム ― The Wall, 情処研報 2000-IM-36-9/2000-EIP-7-9, 情報処理学会, 2000.

[2] 木原民雄, 安斎利洋, 中村理恵子, 太田博満: プラネタリウムに描画する多人数インタラクティブ全天周映像システム, 情処研報 1999-DPS-91-5, 情報処理学会, 1999.

[3] 木原民雄, 安斎利洋, 森脇裕之, 寺中勝美: 子供連画のための Moppet ペイントシステム, マルチメディア通信と分散処理ワークショップ, 1996.

[4] Object Management Group(OMG): The Common Object Request Broker: Architecture and Specification, Revision 2.3, June 1999.

[5] Sun Microsystems: Jini Architecture Specification Version 1.1 October 2000.

[6] Sun Microsystems: Java Message Service Version 1.0.2 November 9, 1999.