システムソフトウェアと
オペレーティング・システム 78－9
（１９９８．５．７）

ポゼッションモデルに基づく分散協調支援アプリケーション

望月 祐洋[1]　　徳田 英幸[1,2]

[1]慶應義塾大学大学院 政策・メディア研究科
[2]慶應義塾大学環境情報学部
{moma, hxt}@mkg.sfc.keio.ac.jp

概要

本論文では，分散環境上のデバイスやアプリケーションの構成要素を，***Soul*** と ***Body*** という抽象化に基づいてアプリケーション内部で統一的に扱い，単純操作による組み合わせを可能にする枠組となるポゼッションモデルを提案する．本モデルに基づく分散協調支援アプリケーションは，利用環境や状況の変化に応じた構成や動作の変更を，ユーザの手動設定やシステムの自動設定によって実現する．本モデルは，移動環境や遍在環境を前提とする適応的アプリケーションの構築に適用できる．本論文での応用例として，ビデオ会議アプリケーションと位置情報を利用した映像ソース動的選択システムを採り上げ，その実現方法について報告する．

Distributed Collaborative Applications Based on A Possession Model

Masahiro Mochizuki[1]　Hideyuki Tokuda[1,2]

[1]Graduate School of Media and Governance, Keio University
[2]Faculty of Environmental Information, Keio University
5322 Endo, Fujisawa, Kanagawa 252, Japan
{moma, hxt}@mkg.sfc.keio.ac.jp

Abstract

In this paper, we propose a software model named ***Possession Model***, which is designed to provide applications with a consistent view of application components and distributed devices using the abstraction of ***Bodies*** and ***Souls***. Our model enables users to interact with the application components and distributed devices with a simple operation named ***possession***. Distributed applications based on our model realize cooperative adaptive behaviors. The adaptation action can be invoked manually or automatically by monitoring the changes of working environment and situation under which applications are used. Our model is applicable to the adaptive systems tailored to the mobile and ubiquitous computing environment. We describe the design and implementation of the prototype applications, adaptive video conferencing system and dynamic selection system for multiple movie sources.

1 はじめに

複数ホスト上のアプリケーション間の連携によって一定の機能やサービスをユーザに提供する場合，アプリケーションの構成要素や，それらがアクセスするネットワーク上のデバイスに対する一貫した制御手段および構成法の実現が課題の一つになる．

本論文では，分散環境上のデバイスやアプリケーションの構成要素を，***Soul*** と ***Body*** という抽象化に基づいてアプリケーション内部で統一的に扱い，単純操作による組み合わせを可能にするポゼッションモデルを提案する．本モデルにより，分散アプリケーションの構成や動作を環境や状況の変化に応じてユーザが自由に変更する，あるいはアプリケーションが自動的に変更するシステムを実現できる．現在，移動計算機環境の研究分野を中心に，利用環境や利用状況の変化に応じてオペレーティングシステムやアプリケーションのモジュール構成，動作，機能，サービス内容を動的に適応させるシステムの研究が活発に行われている．

本研究の成果は，このうちアプリケーションレベルの適応に応用可能であるとともに，適応動作の影響範囲を一台の移動計算機上に限定せず，ネットワーク上の複数アプリケーションへの伝播を可能にする．例えば，複数ユーザが，ネットワーク上の複数台の計算機上で共通アプリケーションを動作させて分散協調作業を行う場合，ある計算機上で生じた適応動作の結果を，他の計算機に反映させる局面が想定される．本モデルとその応用システムは，アプリケーションの機能変更や構成変更結果を複数アプリケーションの任意の構成要素に伝達し，かつシステム実行中に伝達先を動的に選択，変更する機構を提供する．次節では関連研究について述べ，3 節でポゼッションモデルについて説明する．4 節でポゼッションモデルに基づく基盤システムプロトタイプの設計と実装について述べ，5 節でその応用例としてビデオ会議アプリケーションおよび位置情報を利用した映像ソース動的選択システムの実現について報告する．6 節でシステム基本性能評価結果について述べ，7 節でまとめと今後の課題を述べる．

表 1: 関連研究の分類

項目	Odyssey (CMU)	Service Proxy (JAIST)	Ad-hoc Mobile Service (UCB)	Multi-layered mobile multimedia architecture (OGI)	Possession Model
適応目的	利用可能な計算機資源の再配分	計算機の移動支援，連続メディアアプリケーションの継続利用	携帯端末の位置依存サービスへの適応	移動環境での作業継続性の保証，マルチメディアアプリケーションの継続利用	ユーザの作業環境や作業状況へのアプリケーションの適応
適応分類	アプリケーションアウェアな適応	アプリケーションアウェアな適応，ユーザアウェアな適応	ユーザアウェアな適応	アプリケーションアウェアな適応，ユーザアウェアな適応	マルチユーザアウェアな適応
適応開始のトリガ	ネットワークバンド幅，遅延，ディスクキャッシュ，CPU，バッテリ，課金コスト等の変化	計算機の構成変更 (PC カードの挿抜，交換)	特定サービス領域からの移動	ネットワークインタフェースの切り替え (有線から無線へ)	デバイスの状態変化，ユーザからの要求．
環境変化の検知機構	Odyssey (カーネル，Warden，Viceroy)	カーネルでイベントを検知し，クライアントのサービスプロキシに通知	サーバビーコニングによるクライアント位置確認 (ドメイン毎の SIP およびクライアント毎の SIC の協調)	pmid (デーモン) によるイベント一元管理，pulse プロセスのハートビート (ICMP echo request/reply を利用)	デバイス ***Body*** と ***Soul*** が必要に応じて協調
適応要素	動画の空間的解像度 (圧縮度の異なる QuickTime ファイルおよびモノクロ)，画像の空間的解像度，処理サーバの切り替え (ローカル／リモート)	連続メディアの空間的解像度および時間的解像度	クライアント GUI の Look & Feel の変更，オブジェクト制御のインタフェース	連続メディアの最高フレームレート，空間的解像度とフレームレートとのトレードオフの調整	GUI の Look & Feel の変更，アプリケーションの構成と機能の変更
応用事例	ビデオサーバ／プレイヤ，Web ブラウザ，音声認識サーバ／クライアント	マルチキャスト対応ビデオビューワ，移動アプリケーション	A/V 機器の制御，サーバ自動設定，位置情報表示，ロケーション追跡，ローカルプリンタアクセス	ストリーミングアプリケーション	ビデオ会議アプリケーション，位置情報による映像ソース動的選択

2 関連研究

今後，分散協調支援アプリケーションの実現のために重要となる適応的システムの関連研究を項目別にまとめた内容を表1に示す．

Odysseyでは，システムとアプリケーションが協調して行う適応のモデルをアプリケーションアウェアな適応と特徴づける．アプリケーションアウェアな適応モデルでは，適応の範囲に完全自由放任(laissez-faire)から完全アプリケーション透過(application-transparent)まで幅を持たせている[6]．我々は，適応モデルに関してユーザからの視点を考慮し，アプリケーションアウェアという分類に加えユーザアウェア，さらにマルチユーザアウェアという分類までモデルを拡張する．ここで，ある適応がユーザアウェアであるとは，システムまたはアプリケーションの適応結果をユーザが可知であることを意味する．これには適応の旨をユーザに明示的に通知する方式や，表に示されるような適応要素の変化によって(結果的に)間接通知する方式が存在する．アプリケーションアウェアな適応であることが，必ずしもユーザアウェアな適応であることとは一致しない．さらに，複数ユーザの利用を前提とするアプリケーションでは，適応結果が複数ユーザに可知であるか否かを考慮せねばならず，この意味でマルチユーザアウェアな適応の概念が重要である．適応の性質がどのように分類されるかは，適応の目的や，応用例となるアプリケーションの性質に大きく依存する．Service Proxy[5]やMulti-layered mobile multimedia architecture[4]は，連続メディアの空間的解像度，時間的解像度の変化がユーザから認知可能である点，ユーザに適応要素の設定を明示的に行わせる点で，一部ユーザアウェアな適応に分類される．一方Ad-hoc Mobile Service[3]は，ユーザが持ち歩く携帯端末の機能を，移動先で提供されるサービスに適応させるシステムを提案しており，ユーザアウェアな適応に分類される．

本研究の適応目的は，ユーザが行う作業へのアプリケーションの適応であり，マルチユーザアウェアな適応に分類できる．デバイスの状態変化やユーザからの指示を適応開始のトリガにし，適応内容としてGUIのLook & Feelや，アプリケーションの構成，機能の変更を行う．この際，環境変化の検知は，デバイス***Body***と***Soul***の協調によって実現する．本論文では，ビデオ会議アプリケーションと位置情報を利用した映像ソース動的選択システムを応用事例として採り上げている．

3 ポゼッションモデル

本節では，ポゼッションモデルの基本概念について，その目的と要件を中心に説明する．さらに，モデルの中心概念となるアプリケーション構成要素の***Body***と***Soul***としての抽象化と，ポゼッションについて順に説明する．

3.1 ポゼッションモデルの基本概念

ポゼッションモデルは，(1)デバイスとアプリケーション構成要素を統一的に扱う枠組，および(2)構成要素を単純操作によって組み合わせ，アプリケーションの構成や機能の変更をするための枠組を提供する．

これらを実現するために，本モデルでは次の3つの要件に基づくシステムの抽象化を行う．

1. 制御構造の記述と制御対象の分離
2. 制御対象の動的選択の実現
3. オブジェクト間の協調関係の設定支援

第1の要件は，同一の制御対象に異なる制御方法を適用する，あるいは異なる制御対象に同一の制御方法を適用することを可能にする．これによって，ユーザアプリケーションの構成要素やさまざまなデバイスを対象に，複数ユーザが記述した多様な制御内容を適用する．

第2の要件は，分散環境上の複数の制御対象を，ネットワーク透過に選択することを可能にする．選択した制御対象を動的に変更したり，一度に複数の対象を同時制御することも可能にする．

第3の要件は，分散環境上の複数オブジェクトが協調してユーザにある機能を提供する場合に，協調関係を組み変えることで全体としての機能を変更することを可能にする．

以上の3要件を満たすために，我々は，アプリケーションを構成するオブジェクトを，**制御を担当する部分*Soul***(魂)と，**制御対象となる部分*Body***(肉体)とに分離する抽象化を行った．そして，図1の概念図に示すように，***Soul***が***Body***に憑依して(possess)一体化することで，完全なオブジェクトとして振る舞うモデルを提案する．

以下の小節で，***Soul***, ***Body***およびポゼッションの詳細について順次説明する．

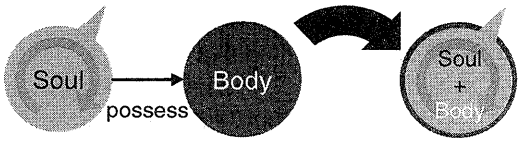

図 1: ポゼッションモデルの概念図

3.2 *Body* の概要

Body は，アプリケーションの構成要素となるオブジェクトのうち，制御が適用される部分を抽象化した実体である．

Body の具体例として，アプリケーションウィンドウ上に描画される文書，静止画像，動画像などのオブジェクトが挙げられる．また，ウィンドウ上の描画オブジェクトのような視覚的実体をともなうものに限らず，ビデオキャプチャデバイス，APM (Advanced Power Management) デバイス，GPS (Global Positioning System) デバイスなど種々の計算機デバイスも ***Body*** として一元的に抽象化している．

Body には，それぞれ一意な名前がつけられており，ユーザはネットワーク透過に識別可能である．

3.3 *Soul* の概要

Soul は，ユーザが ***Body*** に対して通常行う制御を自動化するために，制御手順や制御内容を記述に基づいて，実際に制御を担当する部分である．

Soul の生成とともに実行スレッドが一つ生成され，***Body*** が公開するインタフェースを介し，***Body*** を操作，制御する．

Body の制御を ***Soul*** で行うことによって，一つの ***Soul*** が複数の ***Body*** を制御したり，逆に複数の ***Soul*** が一つの ***Body*** を制御するなど，特定の制御対象に束縛されない制御形態を実現できる．

複数の ***Soul*** が一つの ***Body*** に同時アクセスする場合，***Body*** のメソッド呼び出しは整列化される．

Soul はネットワーク上のホスト間を移動可能であり，***Soul*** の動作内容や処理負担に応じて実行ホストを変更することで，システムの効率的な運用が可能になる．

3.4 ポゼッション

3.4.1 ポゼッションの基本概念

ポゼッションとは，元々魂が肉体にのり移る憑依のアナロジに基づく概念であり，具体的には，***Soul*** が，自身に記述されている制御内容を ***Body*** に反映させるために ***Body*** の制御権を獲得する操作である．***Soul*** の制御対象が特定の ***Body*** に限定されないため，***Soul*** 実行中においてもポゼッションの対象を自由に変更する柔軟性を実現する．

ポゼッションは，***Soul*** の possess メソッド呼び出しで実行される．***Soul*** はポゼッション後，憑依した ***Body*** を，インタフェースに定められた操作を用いて制御する．憑依した ***Body*** を解放するためには，unpossess メソッドを呼び出す．

3.4.2 ポゼッション対象の変更

Soul と ***Body*** への分離の結果，***Soul*** が possess メソッドを呼び出すことで，制御対象となる ***Body*** を適宜変更可能である．

Body と ***Soul*** の生成時には，各ホストに一つ存在するネームサーバに識別子が登録される．***Soul*** が possess メソッドを呼び出す際に，制御対象として選択した ***Body*** の識別子をキーにネームサーバに問い合わせ，***Body*** への参照を取得することによってポゼッションが実行される．

Soul は ***Body*** を制御するために ***Body*** インタフェースを実装しているが，このため ***Soul*** が ***Body*** を制御するだけでなく，***Soul*** が他の ***Soul*** に制御することも可能である．

ポゼッション対象変更の簡単な例として，ホスト A 上の ***Soul*** が，同じくホスト A 上のビデオキャプチャデバイス ***Body*** とビデオビューワ ***Body*** の 2 つをポゼッション対象として，ローカルなビデオビューワ機能を実現する場合を考える．***Soul*** が，ポゼッション対象をホスト B のビデオキャプチャデバイス ***Body*** に変更することで，ホスト A 上のビデオビューワにリモート (ホスト B) の映像を表示できる．あるいは，ポゼッション対象を MPEG ファイルの読み出し ***Body*** に変更することで，ライブ映像から蓄積メディアの映像に表示内容を切り替えられる．

この例からわかるように，ポゼッションという一貫操作によって制御対象を変更し，機能変更を実現できる点に特長がある．

3.4.3 複数対象のポゼッション

Soulは，ポゼッション中の***Body***への参照をハッシュ表に保持しており，同時に複数の***Body***を制御できる．この際，ポゼッション対象の***Body***がローカル，リモートのどちらに存在しようと，ネットワーク透過な制御が実現される．

複数対象のポゼッションの例として，ホストA上の***Soul***が，他の複数ホスト(ホストB, C,..., n)上のビデオキャプチャデバイス***Body***と，これらに対応してホストA上に複数作成されるビデオビューワ***Body***を対象とするポゼッションを行う事例を挙げられる．これによって，多地点に設置されたビデオからの情報を収集し，表示する機能を持つporthole[1]のようなシステムを実現できる．また，ホストA上の複数ビデオビューワ***Body***はそのままで，それらが参照する映像ソースをホストA上の同一ビデオキャプチャデバイス***Body***に変更するだけで，フレームの逐次描画機能を実現できる．

3.4.4 ポゼッション時のアクセス制御

ポゼッションモデルでは，***Soul***と***Body***に設定される**強度(strength)**という属性値に基づくアプリケーションレベルでの単純なアクセス制御方法を提供している．

Soul自身に設定されている強度値が***Body***の強度値を上回れば，ポゼッションを実行する．例えば，あるユーザが自ホストのデバイス***Body***を他のユーザからアクセスされたくないと考えれば，デバイス***Body***の強度を最高値に設定することでリモートの***Soul***からのアクセスを拒否できる．

また，ユーザは***Body***にアクセス制御リストを設定でき，この場合には強度値のチェックに併せて***Soul***の所有者属性(ユーザID)に基づく権限のチェックを行い，アクセスの最終的な可否を決定する．

4 基盤システムプロトタイプの設計および実装

本節では，ポゼッションモデルに基づく分散協調支援アプリケーション実現のための基盤システムプロトタイプであるポゼッションシステムの設計と実装について述べる．

4.1 ポゼッションシステムの設計

図2にポゼッションシステムの構成を示す．ポゼッションシステムは***Field***, ***Medium***, ***Soul***, ***Body***の4つの構成要素から成り立っている．***Field***は，他の3要素を生成するための基本機能を提供する．***Field***はホスト毎に1つ存在し，***Field***間でIPマルチキャストによってホスト情報を交換する．***Medium***は***Field***上に複数作成され，***Body***の名前空間を提供する．表2に，各構成要素の基本APIを示す．

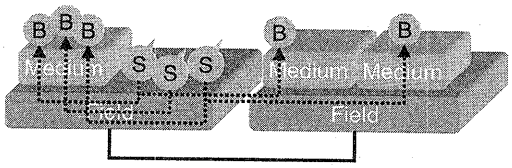

図2: ポゼッションシステムの構成

表2: システム基本API

構成要素	メソッド名	内容
Field	createSoul	***Soul***の生成
	killSoul	***Soul***の削除
	listSoul	***Soul***一覧の表示
	createMedium	***Medium***の生成
	killMedium	***Medium***の消去
	listMedium	***Medium***一覧の表示
Medium	createBody	***Body***の生成
	killBody	***Body***の消去
	listBody	***Body***一覧の表示
Soul	possess	ポゼッション実行
	unpossess	ポゼッション解除
	transfer	所属***Field***の移動
Body	—	—

4.1.1 ネームサービス

ポゼッションシステムは，***Soul***と***Body***の制御にJava Remote Method Invocation (RMI)[7]を利用している．***Soul***と***Body***には生成時に一意な識別子を割り振られ，名前情報はJavaのリモートオブジェクトレジストリである`rmiregistry`で一括管理している．

4.1.2 通常 *Body* とデバイス *Body*

ポゼッションシステムにおけるアプリケーションの構成要素を抽象化した ***Body*** を**通常 *Body*** とすると，アプリケーションからアクセス可能な計算機デバイスを抽象化した ***Body*** を**デバイス *Body*** と呼ぶ．

通常 *Body* が ***Medium*** 毎に作成されるのに対し，**デバイス *Body*** は ***Field*** 毎に作成される．表 3 に**通常 *Body*** としてビデオビューワ用 ***Body*** に特化したインタフェースの抜粋を示す．また，表 4 にデバイス ***Body*** のインタフェースを示す．現状では，各種デバイスに共通の汎用インタフェースのみを提供している．

表 3: **通常 *Body*** のインタフェース (抜粋)

インタフェース	内容
setPosition	***Body*** の座標設定
setScale	***Body*** の拡大／縮小
setFlip	***Body*** の反転表示
setImage	***Body*** の画像設定

表 4: デバイス ***Body*** のインタフェース (抜粋)

インタフェース	内容
device_open	***Body*** のオープン
device_close	***Body*** のクローズ
device_read	***Body*** からのデータ読み出し
device_write	***Body*** へのデータ書き込み

4.2 ポゼッションシステムの実装

本システムの実装は，FreeBSD2.2.5 上の JDK1.1.5 を利用して Java 言語 [2] によって行った．デバイス制御部分は，JNI (Java Native Interface) の枠組に基づき C 言語で実装した．

4.2.1 ポゼッションシェル

本プロトタイプでは，***Body*** と ***Soul*** の生成，消去，possess 操作を制御するためのユーザインタフェース例として，UNIX シェル類似のコマンドラインインタフェースを用意した．表 5 に，コマンド一覧を示す．

コマンドラインで入力する他，一連のコマンドをファイルに記述し，シェルスクリプトとして実行できる．現在，スクリプト制御構造の記述はサポートしていない．

表 5: ポゼッションシェルコマンド一覧 (抜粋)

コマンド	内容
cd objectID	操作対象オブジェクトの変更
ls [-msba]	既存オブジェクトの一覧表示
rm objectID	オブジェクトの削除
mv soulID x y	***Soul*** が憑依中の ***Body*** を座標 (x, y) に移動
mv soulID hostname	***Soul*** を指定ホストに移動
screate	***Soul*** の生成
possess soulID bodyID	***Soul*** の ***Body*** への憑依
start soulID	***Soul*** の動作開始
stop soulID	***Soul*** の動作停止

4.2.2 *Body* の実装例

図 3 に，***Body*** の実装例を示す．APM デバイス ***Body***，SCC (IBM Smart Capture Card II) デバイス ***Body***，GPS (SONY PACY-CNV10) デバイス ***Body***，MPEG1 ソフトウェアデコーダ ***Body***，ビデオビューワ ***Body***，静止画ビューワ ***Body***，テキスト ***Body*** が実装済みである．

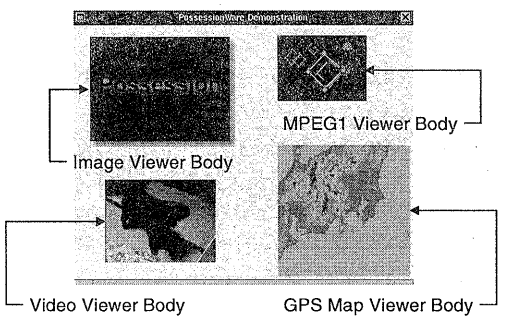

図 3: ***Body*** の実装例 (抜粋)

5 応用例

本節では，ポゼッションモデルの応用例として，ビデオ会議アプリケーションと位置情報を利用した映像ソース動的選択システムを採り上げる．それぞれの応用例が，どのように実現されているのかを ***Soul*** と ***Body*** 構成を図示しつつ説明する．

5.1 ビデオ会議アプリケーション

図4に，ポゼッションモデルに基づくビデオ会議アプリケーションの利用画面を示す．また，図5に，そのときの ***Soul*** と ***Body*** の構成を示す．図5の Bv はビデオビューワ ***Body***，Bt はテキスト ***Body***，D はビデオキャプチャデバイス ***Body*** を意味し，Sv および St は，それぞれ対応する ***Soul*** を意味している．また，図中の矢印は ***Soul*** から ***Body*** に対して向けられており，***Soul*** が ***Body*** を possess している状態を示している．

図 4: ポゼッションモデルに基づくビデオ会議アプリケーションの画面

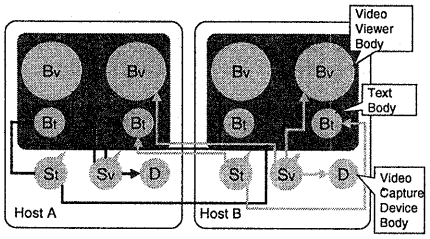

図 5: ビデオ会議アプリケーションにおける ***Soul*** と ***Body*** の構成

アプリケーションの構成変更例として，ホストA上のビデオキャプチャ ***Soul*** をホストBに移動させた場合の ***Soul*** と ***Body*** の構成を図6に示す．これによって移動後の ***Soul*** は，ビデオキャプチャ処理をホストB上で実行する．このとき ***Soul*** は，ホストA上のビデオキャプチャデバイス ***Body*** を possess したままであるため，ホストA上のカメラ映像がキャプチャされる．したがってユーザにとっては，***Soul*** の移動にともなう外見上の変化は生じていない．しかし，実際はホストAとホストB間のデータフローに変化が生じている．

ユーザが映像ソースを変更したければ，ポゼッションシェルからの操作で，***Soul*** のポゼッション対象をホストA上のビデオキャプチャデバイス ***Body*** からホストB上のビデオキャプチャデバイス ***Body*** 他の映像ソースに変更すればよい．

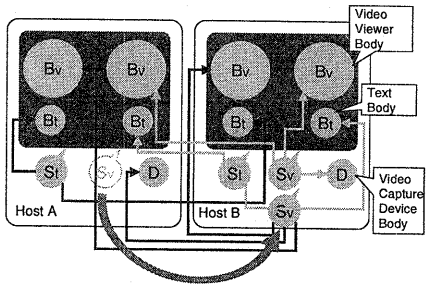

図 6: ビデオ会議アプリケーションにおける ***Soul*** と ***Body*** の構成 (***Soul*** 移動後)

5.2 位置情報を利用した映像ソース動的選択

別の応用例として，GPSデバイス ***Body*** から得られる位置情報に基づき表示映像を動的に選択するシステムについて述べる．図7に ***Soul*** と ***Body*** の構成を示す．モバイルホスト (MH) が，地点Aから地点Bへ移動するのにともない，サーバホスト上の ***Soul*** (S_2) が位置情報の変化を検知してビデオキャプチャ ***Soul*** (S_1) に通知し，映像ソースをビデオキャプチャデバイス ***Body*** から得られるライブ映像からMPEG デコーダ ***Body*** から得られるMPEG1映像に切り替える．

このようなシステムを，無線LAN環境と組み合わせることで，無線が届く範囲で，かつGPSで位置情報を取得できる範囲という制約の元で特定の場所で特定の映像を流す情報サービスに利用できる．

位置情報の取得方法に関しては，計算能力の制約が大きいモバイルコンピュータやPDA上で位置情報を定期取得する方式の問題点について指摘がなされている [3]．

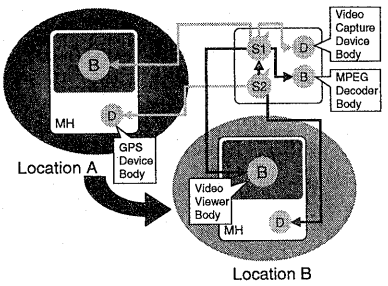

図 7: 位置情報を利用した映像ソース動的選択システムの構成

6 基本性能評価

基本性能測定のために，2台のノートブックコンピュータ (TOSHIBA DynaBook SS-R590) を使用した．それぞれ CPU は Pentium 90MHz で，主記憶 40MB，ネットワークインタフェースとして 3Com EtherLinkIII 3C589C を用いた．

キャプチャしたビデオフレームのピクセルサイズは 160×120，デプスは 16 である．表 6 に，ビデオ会議アプリケーションの利用時に測定したフレーム描画時間を示す．測定には，単独アプリケーションとして実装されたビデオビューワ，および RMI 経由でフレーム描画を行うビデオ会議アプリケーションを用い，各 100 回の計測を行った．RMI の際には，フレームサイズ分の int 型配列を引数として渡した．

まず，ビデオビューワの場合の測定結果では，1フレームの描画に平均 43 ミリ秒を要した．このときのフレームレートは約 23fps である．

次に，ビデオ会議アプリケーションにおける RMI 経由のフレーム描画には平均 80.93 ミリ秒，***Soul*** 移動後のリモートからの描画には平均 91.19 ミリ秒を要し，RMI を利用しない場合に比べ描画時間に約 2 倍の差が生じているという結果が得られた．

表 6: フレーム描画時間

	描画時間 (ms)	フレームレート (fps)
通常ローカル	65.71	15
RMI ローカル	80.93	12
RMI リモート	91.19	10

この他，引数なしの RMI 実行時間を測定した結果は平均 10.29 ミリ秒であった．***Soul*** と ***Body*** 間の通信に RMI を利用した現在の実装では，連続メディアデータの通信や計算機の移動を考慮した通信機構はサポートしておらず，通信性能面に多くの課題が残されている．このため，Java RMI のリモートリファレンス層，トランスポート層にマルチキャストに基づくオブジェクト参照や移動中のコネクション管理を行う適切なメカニズムを組み込む必要がある．

7 まとめと今後の課題

本論文では，分散環境上のアプリケーション構成要素やデバイスを統一的に扱う枠組であるポゼッションモデルを提案した．ポゼッションモデルの適用例として，分散協調支援アプリケーションを採り上げ，ビデオ会議アプリケーション，および位置情報を利用した映像ソース動的選択の例を説明し，それらを実現する ***Soul*** と ***Body*** の構成と，それら協調メカニズムを示した．

今後の課題として，RMI の性能によって制約を受けている通信部分の実行性能の向上について踏み込んだ研究を行う必要がある．また，ポゼッションモデルの適用例として情報家電の制御を採り上げ，その適用可能性について研究を進める予定である．

謝辞

筆者らは，慶應義塾大学 MKng プロジェクトのメンバー，および徳田研究室 KMSF グループのメンバーの協力と助言に深謝します．

参考文献

[1] Dourish, P. and Bly, S.: Portholes: Supporting Awareness in a Distributed Work Group, *Proceedings of ACM Conference on Human Factors in Computing Systems (CHI'92)*, pp. 541–547 (1992).

[2] Gosling, J., Joy, B. and Steele, G.: *The Java Language Specification*, Addison Wesley, Reading, Massachusetts (1996).

[3] Hodes, T. D., Katz, R. H., Servan-Schreiber, E. and Rowe, L.: Composable Ad-hoc Mobile Services for Universal Interaction, *Proceedings of MOBICOM'97*, pp. 1–12 (1997).

[4] Inouye, J., Cen, S., Pu, C. and Walpole, J.: System Support for Mobile Multimedia Applications, *Proceedings of the 7th International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV '97)* (1997).

[5] Nakajima, T. and Hokimoto, A.: Adaptive Continuous Media Applications in Mobile Computing Environment, *IEEE International Conference on Multimedia Computing and Systems (ICMCS'97)* (1997).

[6] Noble, B. D., Satyanarayanan, M., Narayanan, D., Tilton, J. E., Flinn, J. and Walker, K. R.: Agile Application-Aware Adaptation for Mobility, *Proceedings of the 16th ACM Symposium on Operating System Principles* (1997).

[7] Sun Microsystems, Inc.: Remote Method Invocation Specification (1997). http://java.sun.com.