異言語プラットフォーム間で動作する P2P 型複製オブジェクト環境の構築

市川泰宏[†1]　山本佑樹[†1]　高田秀志[†2]

近年，リアルタイムなコミュニケーションを通して行う協調作業がさまざまな場面で取り入れられてきた．また，人々が使用する端末として，さまざまな種類の端末が広く普及してきた．それにより，今後さまざまな端末を用いて協調作業を行うことができる環境が求められると考えられる．我々は，複製計算に基づくリアルタイムな協調作業支援システムのためのフレームワークを開発している．本稿では，使用する端末によって開発環境や実行環境が異なる点に着目し，このフレームワークを異言語プラットフォーム上でも動作可能にするためのソフトウェア構成について述べる．また，提案したソフトウェア構成を用いて，処理の遅延時間をやフレームワークの開発コストの削減率を測定し有用性を示す．

A P2P-based Replicated Object Environment Running on Heterogeneous Language Platforms

Yasuhiro Ichikawa,[†1] Yuki Yamamoto [†1]
and Hideyuki Takada[†2]

Recently, collaborative works through realtime communication are being adopted in various environments. Also, mobile phone installed with Android OS, iPhone, and iPad have also appeared. Thus, there is a high demand on environment for users to perform collaborative works using their terminals. We are developing a framework for collaborative works based on the replicated computing. In this paper, we focus on the difference of development and runtime environment depending on terminals, and provide software architecture which makes the framework possible to run on heterogeneous language platforms. We have also examined the amount of reduction for framework development costs to evaluate the usefulness of the system.

1. はじめに

近年，電子会議システムや分散エディタといったコンピュータを用いた協調作業支援システムが企業や教育現場で取り入れられてきた．協調作業支援システムを用いて行う協調作業の中でも，リアルタイムな協調作業は作業の効率化や新しいアイデアの創出などに対して有用である．リアルタイムな協調作業を支援するシステムにおいて，端末間で通信をするための通信方式には，一般的に P2P 方式とクライアントサーバ方式がある．リアルタイムな協調作業支援システムを構築する場合，これらの通信方式のうち，P2P 方式には，クライアントサーバ方式のようにサーバを必要としないため，サーバの稼働状況にアプリケーションの稼働率が影響されないといったメリットがある．また，無線 LAN の環境がなくても，端末同士を Bluetooth を用いて P2P 接続することで協調作業を手軽に行うことができるというメリットもある．近年，高性能な携帯端末が一般に広く普及してきた．これにより，今まで PC を用いて行われていた協調作業が，さまざまな端末を用いて行われるようになると考えられる．また，携帯端末上で協調作業が行われるようになると，自身の端末を持ち寄るだけで場所や時間に捕われず，手軽に協調作業を行うことができる環境が求められると考えられる．このような，さまざまな種類の端末を持ち寄るだけで手軽に協調作業が行えるような環境を実現するには，クライアントサーバ方式に比べて手軽に協調作業を行うことができる P2P 方式を用いた協業作業支援システムが有効であると言える．

協調作業支援システムでは，端末間でアプリケーションのビューやアプリケーションに対する操作を同期する必要がある．ビューや操作の同期を実現する環境の一つに，複製計算モデル[1] に基づく“P2P 型複製オブジェクト環境”がある．P2P 型複製オブジェクト環境では，P2P ネットワーク上に存在するすべての端末にオブジェクトの複製が配置され，そのオブジェクトのふるまいが同期される．我々は，P2P 型複製オブジェクト環境を提供するリアルタイムな協調作業支援システムのためのフレームワーク“CUBE”[2] を構築している．CUBE では，P2P 型複製オブジェクト環境を Java を用いて実現し，ある端末で発生したメソッド呼び出しやオブジェクトの生成が他の端末に伝播される“オブジェクトミラーリング”と呼ばれる機能を提供している．本論文では，先に述べたような要求に基づき，人々が自身の持つさまざまな種類の端末を持ち寄るだけで協調作業を行うことができる環境を実

†1 立命館大学大学院
†2 立命館大学

現するため，プラットフォームの異なる端末でオブジェクトミラーリングを実現する手法を提案する．

プラットフォームが異なることで生じる問題の一つに，プラットフォームによって使用できる開発言語が異なるという点がある．異言語プラットフォーム間で動作するリアルタイムな協調作業支援システムを構築するには，各言語で CUBE フレームワークを開発する必要がある．しかし，各言語で CUBE で提供している機能をすべて実現する場合，CUBE の機能や対応言語の増加に伴いフレームワークの開発コストが増加してしまう．そこで，CUBE フレームワークのコアとなる機能のみを各言語で開発し，残りの機能を擬似的に提供することで，フレームワークの開発コストを削減するソフトウェア構成について検討する．

本研究では，Java で開発された PC 用アプリケーションと Objective-C で開発された iPodTouch 用アプリケーションでリアルタイムな協調作業を行うためのフレームワークを構築した．また，構築したフレームワークを利用し，PC と iPodTouch で動作するアプリケーションを開発した．このアプリケーションを用いて協調作業を行ったとき，リアルタイムな協調作業を実現できているかの評価を行うために，異種端末間でのメッセージ送受信の遅延時間を計測した．さらに，コアとなる機能のみを実現することで，フレームワークの開発コストがどの程度削減されたかをソースコード量を指標に調査した．

以下，2 章では CUBE ついて説明し，異種端末間での P2P 型複製オブジェクト環境を構築するときの問題点を述べる．また，3 章では，異種端末間で P2P 型複製オブジェクト環境を構築するときのソフトウェア構成，および，異種端末間でのメソッド呼び出しの伝播方法について述べる．4 章では，構築したシステムの評価と考察を述べ，5 章では，結論と今後の課題について述べる．

2. 異種端末混在環境における P2P 型複製オブジェクト環境

人々が所有する情報端末が多様化してきたことにより，コンピュータを用いた協調作業にもさまざまな種類の端末が使用されるようになってきた．異種端末間で動作する協調作業支援システムを開発するとき，端末によってアプリケーションの実行環境が異なるため，使用できる開発言語が異なる場合があるという問題点がある．本章では，P2P 型複製オブジェクト環境を提供する CUBE フレームワークについて述べた後，異種端末間で P2P 型複製オブジェクト環境を実現するときの問題点について述べる．

2.1 リアルタイムな協調作業支援システムのためのフレームワーク CUBE

リアルタイムな協調作業支援システムを開発するためには，各端末でアプリケーションのビューやアプリケーションに対する操作を同期させる必要がある．このような機能を実現する環境の一つに，先に挙げた複製計算モデルに基づく P2P 型複製オブジェクト環境が存在する．CUBE は，リアルタイムな協調作業支援システムを構築するためのフレームワークであり，そのようなシステムを開発するときに必要となる機能を提供している．提供している機能の 1 つにオブジェクトミラーリングがある．オブジェクトミラーリングは，ある端末で発生したメソッド呼び出しを他の端末に伝播することで，端末間の複製オブジェクトのふるまいを同期する機能である．各端末にはオブジェクトを一意に識別するオブジェクト ID が割り振られた複製オブジェクトの他に，他端末にメソッド呼び出しを伝播する機能をもつ代理オブジェクトが配置される．各端末に配置された複製オブジェクトと代理オブジェクトは，それぞれ 1 対 1 で対応している．代理オブジェクトと複製オブジェクトは同一のインタフェースを保持しているため，アプリケーションの開発者は複製オブジェクトのメソッドを呼び出すのと同様に，代理オブジェクトのメソッドを呼び出すことができる．代理オブジェクトで呼び出されたメソッドの情報を格納したメッセージは，ネットワークを通して他の端末に伝播される．このとき伝播されるメッセージには，メソッドのシグネチャ，メソッドの引数の値，オブジェクト ID，および，端末を識別するためのホスト ID が格納されている．図 1 にオブジェクトミラーリングにおけるメソッド呼び出しの伝播の仕組みを示す．各端末には，メソッド呼び出しの実行とメッセージの送信を行うためのミラーリングスレッドと，メソッド情報の受信を行うためのメッセージ受信スレッドがある．また，図中の実線矢印は端末 A で発生したメソッド呼び出しが端末 B に伝播されるまでの流れを表す．(図中の矢印に示した数字は以下の数字と一致する)

(1) 端末 A の代理オブジェクトに対してメソッド呼び出しが発生する．
(2) メソッド呼び出しを受け取った代理オブジェクトは，対応する複製オブジェクトのメソッドを呼び出す．
(3) 代理オブジェクトはネットワークを通じて端末 B にメソッド呼び出しの情報を格納したメッセージを伝播する．
(4) メッセージを受信したメッセージレシーバはそのメッセージをオブジェクトマネージャに受け渡す．
(5) オブジェクトマネージャはメッセージに格納されたオブジェクト ID から対応する複製オブジェクトを選択し，メッセージに格納されたメソッドのシグネチャと引数の値からリフレクションを用いて，対応する複製オブジェクトのメソッドを呼び出す．

これにより，端末 A に配置された複製オブジェクトのふるまいが，端末 B に伝播され，端

末 A と端末 B で複製オブジェクトのふるまいが同期される．

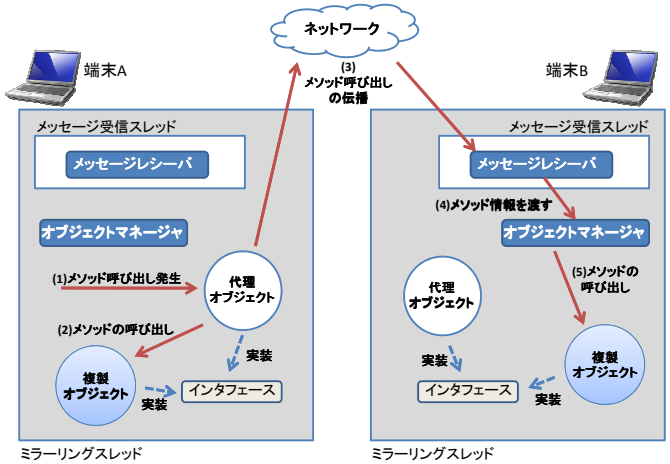

図 1　オブジェクトミラーリングの仕組み

また，CUBE は，オブジェクトミラーリング以外にも協調作業に必要なさまざまな機能を提供している．CUBE が提供しているすべての機能は，機能の役割によってコア機能とサブ機能に分けられる．コア機能には，オブジェクトミラーリングの機能や協調作業時に端末の途中参加を可能にする中途参加機能[3]，端末間でアプリケーションの整合性を保つ一貫性保証機能[4] がある．また，サブ機能には，協調作業を行うときに端末間でグループの形成を可能にするグループ管理機能[5] や，処理能力の低い端末を利用するときにも高レスポンス性を維持するために処理能力が低い端末で処理するメソッド呼び出しの数を削減するメソッド呼び出しの省略機能[6] がある．これらの機能の内，コア機能はリアルタイムな協調作業支援システムを実現するために必要な機能であり，サブ機能はアプリケーションの性質によって必要になる機能である．

2.2　異種端末間で動作する P2P 型複製オブジェクト環境の構築と問題点

異種端末間で P2P 型複製オブジェクト環境を実現するためのソフトウェア構成は 2 通り考えられる．一つは異なる実行環境をもつ端末が P2P ネットワーク上に配置され，すべての端末がお互いにメッセージのやり取りをする構成である．この構成を実現するためには，CUBE で提供しているコア機能とサブ機能の両方をすべての言語で開発する必要がある．そのため，すべての実行環境において CUBE で提供しているすべての機能を利用することができるが，フレームワークの開発コストが増加してしまう．

もう 1 つは，異なる言語で実装されたアプリケーションの動作する端末間でメッセージのやり取りをするときに，仲介を果たす主言語を決め，その主言語を実行環境にもつ端末のみが P2P ネットワーク上に配置される構成である．この構成では，主言語以外を実行環境にもつ端末は，主言語を実行環境にもつ端末を介することで，一部の機能は制限されてしまうが，CUBE のサブ機能を擬似的に実現することができるようにする．そのため，前者の構成とは異なり各言語で開発しなければならない機能は CUBE で提供しているコア機能のみである．その結果，システムの開発コストを削減することができる．

また，異種端末間で P2P 型複製オブジェクト環境を実現するときの問題点として，端末によって使用できる開発言語が異なるため，同じ機能を持つメソッドでも，メソッド名や引数の順番が異なるという点がある．そのため，メソッド名や引数の対応がとれず，伝播したメソッド呼び出しの情報から同様の機能を持つメソッドを，リフレクションを用いて呼び出すことができない．このような問題点を考慮した上で，次章では，異種端末混在環境で動作する P2P 型複製オブジェクト環境を構築するための，特定の言語を主言語としたソフトウェア構成と，メソッド対応表を用いた異言語間でのメソッドの対応付け方法について述べる．

3.　親子関係を用いた異種端末混在環境におけるオブジェクトミラーリング

本章では，異種端末混在環境で動作する P2P 型複製オブジェクト環境を構築するために，端末間で親子関係をとるソフトウェア構成と，メソッド対応表を用いた異言語間でのメソッドの対応付け方法について述べる．

3.1　ソフトウェア構成

協調作業を行う端末間で親子関係をとるソフトウェア構成で P2P 型複製オブジェクト環境を実現する一例として，Java アプリケーションと Objective-C アプリケーションを用いて，オブジェクトミラーリングを行うことを考える．図 2 にそのソフトウェア構成を示す．図に示す通り，P2P ネットワーク上に Java アプリケーションが動作している PC を配置し，主言語を Java とする．Objective-C アプリケーションが動作している iPodTouch は，P2P ネットワーク上の端末 1 台とメッセージの送受信をする．これらの端末の内，iPodTouch とメッセージの送受信を行う PC を親端末，iPodTouch を子端末と呼ぶ．PC と iPodTouch で送受信されるメッセージに格納されている情報は，PC 同士でメッセージの伝播をすると

きと同様の情報である．親端末と子端末では，この情報をもとにメソッドを呼び出し，オブジェクトのふるまいを同期する．

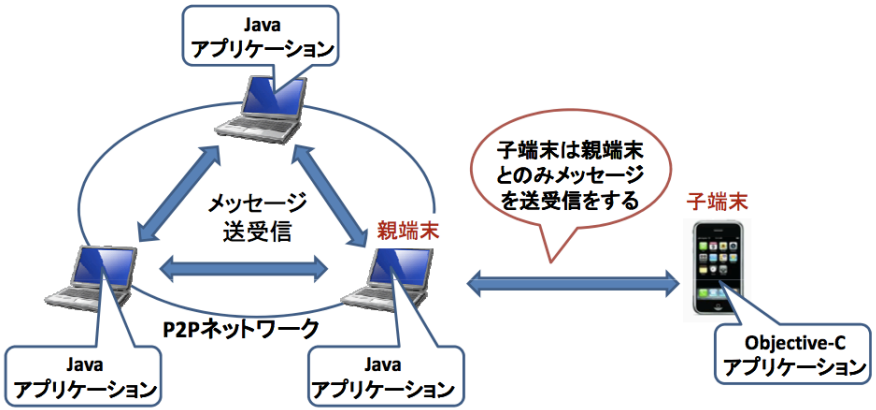

図 2　親子関係を用いたオブジェクトミラーリング

3.2　端末内のスレッド構成とメッセージの流れ

図 3 に異種端末間でオブジェクトミラーリングを行うときのスレッド構成とメッセージの流れの全体像を示す．他端末へメッセージの伝播をするとき，P2P ネットワーク上の端末同士はメッセージをマルチキャストし，親端末と子端末同士はメッセージをユニキャストする．

また，図に示すように，開発言語の異なるアプリケーションでオブジェクトミラーリングを行うために，親端末ではミラーリングスレッドやメッセージ受信スレッドの他に，異種端末通信用スレッドを立てる．異種端末通信用スレッドは，親端末である PC と子端末である iPodTouch の間でメッセージを送受信するためのスレッドである．親端末に対してメッセージが送信されると，ミラーリングスレッドではメッセージ受信スレッド上に配置されたメッセージレシーバが，異種通信用スレッドでは異種通信ブリッジがそれぞれメッセージを受信する．その後，異種通信ブリッジはそのメッセージを子端末である iPodTouch に伝播する．子端末は，そのメッセージをメッセージ受信スレッド上のメッセージレシーバで受信する．つまり，親端末上の異種通信用スレッドが P2P ネットワーク上で子端末の代わりをすることで，子端末上のオブジェクトのふるまいを同期する．

次に，親端末で発生したメソッド呼び出しが子端末に伝播されるときの具体的なメッセージの流れを図 4 に示す．(以下の数字は図中の数字と対応する)

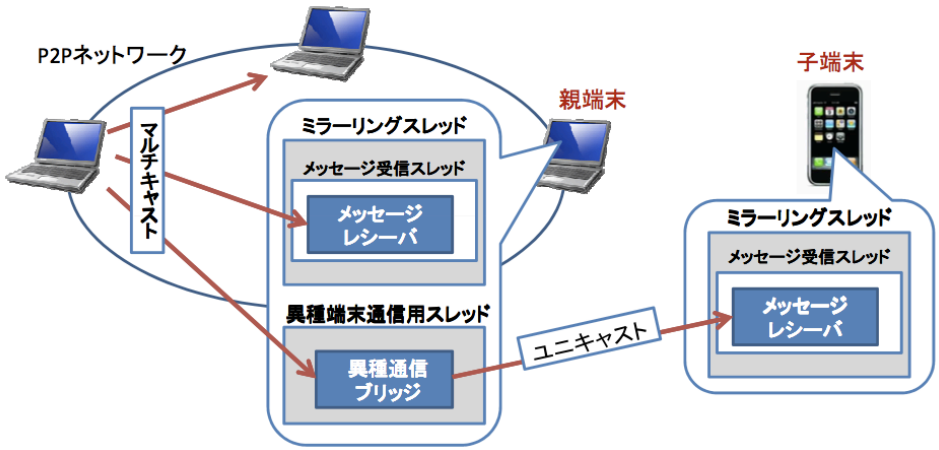

図 3　異種端末間でのオブジェクトミラーリングの全体像

(1)　親端末で，ミラーリングスレッド上の代理オブジェクトに対してメソッド呼び出しが発生する．

(2)　メソッド呼び出しを受け取った代理オブジェクトは，対応する複製オブジェクトのメソッドを呼び出す．

(3)　代理オブジェクトは発生したメソッド呼び出しの情報を格納したメッセージを，ネットワークを通して他端末にマルチキャストする．送信されたメッセージは他端末に送られるとともに，親端末自身に返ってくる．親端末は，そのメッセージをミラーリングスレッド上のメッセージ受信スレッド内にあるメッセージレシーバと，異種通信用スレッド上の異種通信ブリッジ内のマルチキャストメッセージレシーバで受信する．このとき，メッセージレシーバは，ミラーリングスレッドの複製オブジェクトに対して，メソッド呼び出しが二重に発生しないように，受信したメッセージに付与されたホスト ID を確認し，自端末から送信されたメッセージを破棄する．

(4)　メッセージを受信した異種通信ブリッジは，子端末に対してそのメッセージをユニキャストする．

(5)　子端末では，メッセージレシーバがそのメッセージを受信し，オブジェクトマネー

ジャにメッセージを受け渡す．

(6) オブジェクトマネージャはそのメッセージをもとに，自身の複製オブジェクトのメソッドを呼び出す．

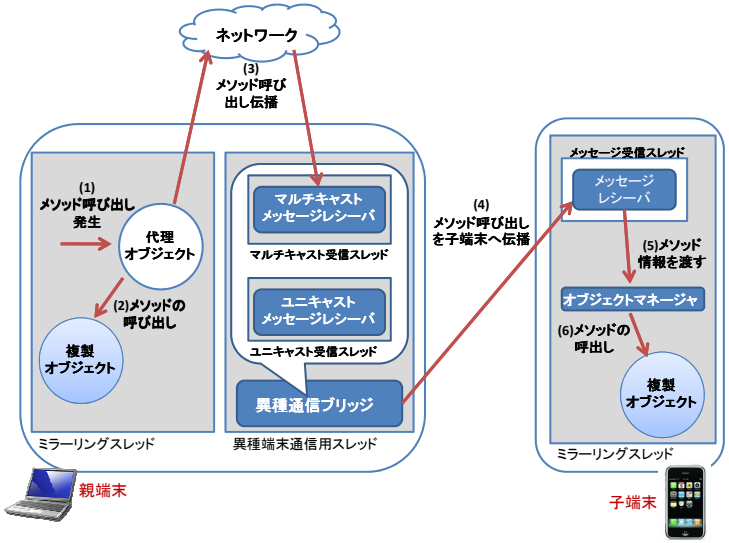

図 4 親端末から子端末へのメソッド呼び出しの伝播

次に，図 5 に子端末で発生したメソッド呼び出しが親端末に伝播されるときのメッセージの流れを示す．(以下の数字は図中の数字と対応する)

(1) 子端末で代理オブジェクトに対してメソッド呼び出しが発生する．

(2) メソッド呼び出しを受け取った代理オブジェクトは，対応する複製オブジェクトのメソッドを呼び出す．

(3) 代理オブジェクトは発生したメソッド呼び出しの情報を格納したメッセージを，ネットワークを通して親端末にユニキャストする．

(4) 親端末では，異種端末通信用スレッド上の異種通信ブリッジ内のユニキャストメッセージレシーバが子端末からのメッセージを受け取り，異種端末通信用スレッド上の代理オブジェクトのメソッドを呼び出す．

(5) メソッドが呼び出された代理オブジェクトは，ネットワークを通して他端末にメッセージをマルチキャストする．送信されたメッセージは他端末に送られるとともに親端末自身に返ってくる．親端末では，そのメッセージをミラーリングスレッド上のメッセージ受信スレッド内のメッセージレシーバと異種端末通信用スレッド上の異種通信ブリッジ内のマルチキャストメッセージレシーバで受信する．このとき，異種通信ブリッジでは，子端末から送られたてきたメッセージを子端末に再度送り返すことを防ぐために，受信したメッセージに付与されたホスト ID を確認し，子端末から送信されたメッセージを破棄する．

(6) メッセージを受信したメッセージレシーバは，オブジェクトマネージャにメッセージを受け渡す．

(7) オブジェクトマネージャはそのメッセージをもとに，複製オブジェクトのメソッドを呼び出す．

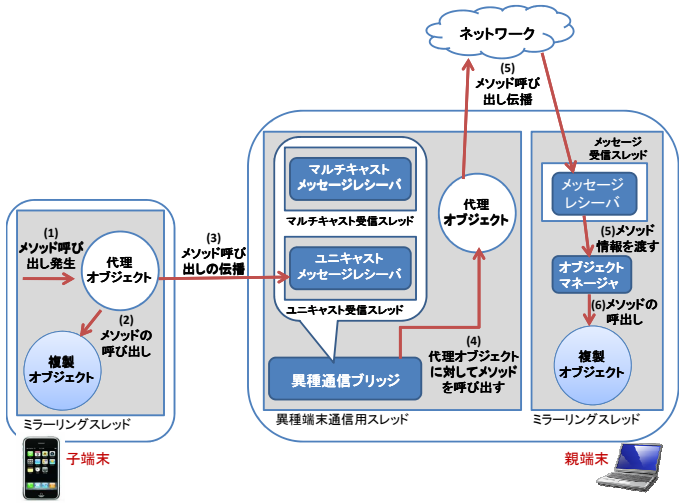

図 5 子端末から親端末へのメソッド呼び出しの伝播

3.3 グループ管理機能の擬似的な実現

本節では，端末間で親子関係をとるソフトウェア構成を用いて異種端末間で協調作業を行うときに，子端末で CUBE のサブ機能を擬似的に実現する方法について述べる．サブ機能の一つであるグループ管理機能は，グループに対して，それぞれマルチキャストアドレスが設定され，代理オブジェクトが特定のマルチキャストアドレスに対してメッセージを送信することで，端末間でグループの形成を可能とする機能である．この機能は，グループごとにマルチキャストアドレスが設定されることで実現できる．そこで，親端末の異種端末通信スレッドをグループごとに立て，スレッド上の異種通信ブリッジ内にあるソケットに，それぞれグループに対応したマルチキャストアドレスを設定することで，子端末で擬似的にグループ管理を実現する．このように，子端末側で必要な処理を親端末が代理することで機能を擬似的に実現する．

3.4 異種端末間におけるメソッド呼び出しの伝播

先に述べたように，開発言語が異なることで対応の不一致が生じる点として開発言語によるメソッド名の違いがある．例えば，int 型の引数を二つ持つ setSize というメソッドを用いる場合，Java ではメソッド名を setSize と表すのに対して，Objective-C では setSize::あるいは第 2 引数の意味を表すための任意のラベルを用いて setSize:Width:と表す．また，メソッド名が同じでも，メソッド定義によっては引数の順番が一致しない可能性もある．この問題を XML を用いてメソッド情報の対応付けをすることで解決する．ソースコード 1 にメソッド対応表の記述例を示す．ソースコードに示すように，sameMethod 要素内に端末間で対応する 1 組のメソッドを記述する．sameMethod 要素内に記述されたメソッド情報は，メソッドごとに method 要素内に記述される．また，method 要素内では，methodName 要素にメソッド名を記述し，argOrder 要素に引数の名前を記述する．methodName 要素の属性で，そのメソッドの開発言語を指定し，argOrder 要素内の属性では引数の順番を指定する．このようにして，端末同士のメソッドの対応を取る．メソッド情報の対応付けは，すべて親端末内の異種端末通信用スレッド上にある異種通信ブリッジで行われる．

ソースコード 1　methodTable

```
<?xml version="1.0" encoding="UTF-8"?>
<methodNameList>
    <sameMethod>
        <method>
                <methodName lang="Java">setSize</methodName>
                <argOrder order="2">width</argOrder>
                <argOrder order="1">height</argOrder>
        </method>
        <method>
                <methodName lang="Objective-C">setSize:Width:</methodName>
                <argOrder order="1">height</argOrder>
                <argOrder order="2">width</argOrder>
        </method>
    </sameMethod>
    <sameMethod>
        <method>
                <methodName lang="Java">changeColor</methodName>
        </method>
        <method>
                <methodName lang="Objective-C">changeColor:</methodName>
        </method>
    </sameMethod>
</methodNameList>
```

4. 実験・評価

本手法に基づいて構築したシステムが実用的に動作するかどうかを検証するために，異なるプラットフォームを持つ端末間でメソッド呼び出しを伝播したときに，端末間でのメソッド呼び出しの伝播による遅延時間を測定し，その評価と考察を行った．また，端末間で親子関係をとるソフトウェア構成をとることで，フレームワークの開発コストをどの程度削減できたかの検証も行った．

4.1 メソッド呼び出しの伝播による遅延時間の測定

本手法に基づいて構築したシステムが実用的に動作するかを検証するために，本手法を適用した 2 種類の GUI アプリケーションを PC，iPodTouch の両方で開発し，そのアプリケーションを用いて実験を行った．一般的な GUI アプリケーションで行われる操作には，ボタンを押すというような 1 回のメソッド呼び出しで完結する操作と，オブジェクトを移動するというような同じメソッドを連続して何度も呼び出す操作があると考えられる．これらの操作の結果起こるメソッド呼び出しを，それぞれ非連続的なメソッド呼び出し，連続的なメソッド呼び出しと呼ぶ．本実験では，PC 側で発生した，非連続的なメソッド呼び出しと連続的なメソッド呼び出しが iPodTouch に到達し，実行された後，iPodTouch から送信されたメソッド呼び出しの完了通知を PC が受信するまでの時間をそれぞれ測定した．また，同様に iPodTouch 側で発生したメソッド呼び出しが PC で実行され，iPodTouch がその完了通知を受け取るまでの時間も測定した，実験は，無線 LAN 環境 (54mbps) で行った．また．使用端末として，ノート PC[2.4GHz Intel Core 2 Duo]，第 4 世代の iPodTouch[Apple A4 プロセッサ 1GHz]，および，第 2 世代の iPodTouch[ARM11 プロセッサ 533MHz] を用いた．性能差のある 2 種類の携帯端末を用いることで，携帯端末の性能差による遅延時間に違

いについても検証した．

4.1.1 実験 1：非連続的なメソッド呼び出しの遅延時間測定

本実験では，非連続的なメソッド呼び出しの伝播による遅延時間を測定する．ある端末から別の端末に対してメソッド呼び出しを伝播したときの遅延時間を測定するためには，端末間で正確に時計を一致させなければならない．しかし，異なる端末同士の時間を正確に一致させることは不可能である．そこで，ある端末で発生したメソッド呼び出しが別の端末に到達し，実行された後，その端末から送信されたメソッド呼び出しの完了通知を元の端末が受信するまでの時間を測定し，評価した．今回は，PC で発生したメソッド呼び出しがオブジェクトミラーリングによって iPodTouch に伝播され，iPodTouch 内でそのメソッドが実行された後，iPodTouch によって送信されたメソッド呼び出しの完了通知を PC が受け取るまでの時間を 100 回測定した．また，同様に，iPodTouch で発生したメソッド呼び出しが PC で実行された後，PC から送信されたメソッド呼び出しの完了通知を iPodTouch が受信するまでの時間も 100 回測定し，それぞれ平均と分散を算出した．このときの実験結果を表 1 と表 2 に示す．表 1 は第 4 世代の iPodTouch を用いたとき，表 2 は第 2 世代の iPodTouch を用いたときの実験結果である．

表 1 実験結果 1(第 4 世代)

	平均 (秒)	分散	最大値 (秒)	最小値 (秒)
PC → iPodTouch → PC	0.288	0.00259	0.412	0.206
iPodTouch → PC → iPodTouch	0.129	0.0276	0.239	0.068

表 2 実験結果 1(第 2 世代)

	平均 (秒)	分散	最大値 (秒)	最小値 (秒)
PC → iPodTouch → PC	0.471	0.00453	0.756	0.350
iPodTouch → PC → iPodTouch	0.257	0.00360	0.368	0.145

4.1.2 実験 2：連続的なメソッド呼び出しの遅延時間測定

本実験では，連続的なメソッド呼び出しの伝播による遅延時間を測定するために，実験 1 と同様にメソッド呼び出しの端末間での往復時間を測定した．連続的なメソッド呼び出しは，非連続的なメソッド呼び出しとは違い，一度に複数回のメソッドが連続して呼び出される．そのため，メソッド呼び出しの伝播先の端末でメソッド呼び出しの実行が滞り，メソッド呼び出しの伝播に遅延が生じる．そこで，連続したメソッド呼び出しの最終的な遅延時間を測定するために，連続した呼び出されたメソッドの内，最後のメソッド呼び出しの遅延時間を測定した．今回は，PC で 100 回メソッドを呼び出し，100 回目のメソッド呼び出しがオブジェクトミラーリングによって iPodTouch に伝播され，iPodTouch 内でそのメソッドが実行された後，iPodTouch によって送信されたメソッド呼び出しの完了通知を PC が受け取るまでの時間を 10 回測定した．また，iPodTouch で 100 回メソッドを呼び出し，100 階目のメソッド呼び出しが PC に到達し，実行された後，iPodTouch がメソッド呼び出しの完了通知を受け取るまでの時間も 10 回測定し，それぞれ平均と分散を測定した．このときの実験結果を表 3 と表 4 に示す．表 3 は第 4 世代の iPodTouch を用いたとき，表 4 は第 2 世代の iPodTouch を用いたときの実験結果である．

表 3 実験結果 2(第 4 世代)

	平均 (秒)	分散	最大値 (秒)	最小値 (秒)
PC → iPodTouch → PC	11.516	0.0393	11.790	11.114
iPodTouch → PC → iPodTouch	2.891	0.139	3.483	2.358

表 4 実験結果 2(第 2 世代)

	平均 (秒)	分散	最大値 (秒)	最小値 (秒)
PC → iPodTouch → PC	36.344	0.0854	36.816	35.742
iPodTouch → PC → iPodTouch	9.767	0.566	11.376	8.557

4.1.3 考　　察

表 1 と表 2 に示した結果より，非連続的なメソッド呼び出しの伝播において，本手法に基づいて構築したシステムは一定のレスポンス性を確保することができたと考えられる．さらに，表 3 と表 4 より，連続的なメソッド呼び出しにおいて，PC で発生した連続的なメソッド呼び出しが iPodTouch で実行された後，PC が実行完了通知を受け取るまでの時間は，第 4 世代の iPodTouch では平均 11.516 秒，第 2 世代の iPodTouch では平均 36.334 秒となった．また，PC が iPodTouch に対して 1 回目のメソッド情報を送信してから 100 回目のメソッド情報を送信し終えるまでの時間と，iPodTouch が PC から 1 回目のメソッド情報を受信してから 100 回目のメソッド情報を受信し終えるまでの時間を 10 回ずつ測定した．その結果，第 4 世代の iPodTouch を用いた場合，それぞれ平均 1.921 秒と平均 8.978 秒となり，第 2 世代の iPodTouch を用いた場合，ぞれぞれ平均 2.065 秒と平均 26.646 秒となった．以上の結果より，iPodTouch の処理性能が原因でメソッド情報の受信処理に遅延

が生じているのではないかと考えられる．さらに，表 3，表 4 より，iPodTouch で発生したメソッド呼び出しが PC で実行された後，iPodTouch が実行完了通知を受け取るまでの時間の平均は，第 4 世代，第 2 世代でそれぞれ 2.891 秒，9.767 秒であった．このような遅延時間を確認できたが，実際のアプリケーションの動きを目視で確認すると，実験結果ほどの遅延時間は感じられなかった．このことから，PC から返ってくるメッセージを iPodTouch で受信するときに，メッセージが滞り，遅延の原因になっていると考えられる．これらの結果より，本手法を適用したアプリケーションを用いて，異種端末間で協調作業をするときのメソッド呼び出しの遅延は，携帯端末の性能に大きく左右されていることが分かった．このような，端末の性能による連続したメソッド呼び出しの遅延には，2.2 節で述べた，CUBE で提供しているメソッド呼び出しの省略機能が有効に働くと考えられる．

4.2 ソースコード量の比較

本手法を用いることで，フレームワークの開発コストがどの程度削減されたかの評価をソースコードの行数を比較して行う．Java で開発された CUBE の全機能の行数とコア機能の行数から，CUBE で提供されているすべての機能を Objective-C で開発したときの行数を算出する．算出に用いる計算式を式 1 に示す．

$$\text{Objective-C CUBE の全機能} = \frac{\text{Java CUBE の全機能*Objective-C CUBE のコア機能}}{\text{Java CUBE のコア機能}} \tag{1}$$

実際のソースコードの行数は Java CUBE の全機能が 7205 行，Java CUBE のコア機能が 2300 行，Objecitive-C のコア機能が 966 行であった．したがって，式 1 より，Objectice-C で CUBE のすべての機能を開発したときのコード量は 3 千行程度と推測できる．つまり，2 千行程度のコード量を削減できたことになる．

5. おわりに

本論文では，異種端末間で動作する P2P 型複製オブジェクト環境のソフトウェア構成として，端末間で親子関係をとり，親端末上に子端末に対応するスレッドを立てることで子端末を管理する構成を提案した．また，異種端末混在環境で P2P 型複製オブジェクト環境を実現するときの問題点として，端末によってアプリケーションのプラットフォームが異なることにより，端末間でメソッドの対応がとれなくなる点を挙げ，その解決策として，XML を用いたメソッド対応表によるメソッドの対応付け方法を提案した．さらに，異種端末間でメソッド呼び出しを伝播したときの遅延時間について評価を行い，本手法の有用性を確認した．

現状の実装では，P2P ネットワーク上で新しく複製オブジェクトが生成されたときに，そのオブジェクト生成を子端末に伝播することができていない．また，子端末が協調作業に参加するときの親端末の選択方法や，親端末に障害が発生したときの対処についても考慮されていない．今後は，オブジェクト生成の伝播や親端末選択の選任アルゴリズム，親端末に障害が発生したときの子端末の受け渡し方法について検討していく．

参 考 文 献

1) David P. Reed,“Designing Croquet’s TeaTime - A Real-time, Temporal Environment for Active Object Cooperation”, Object-Oriented Programming, Systems, Languages & Applications 2005, 2005.
2) Shogo Noguchi, Hideyuki Takada,“CUBE: A Synchronous Collaborative Applications Platform Based on Replicated Computation”, CollabTech2009, pp.19-24, 2009.
3) 鈴本悟, 櫻内彬夫, 高田秀志,“異種端末が混在した P2P 型複製オブジェクト管理手法とその評価”, 情報処理学会 第 72 回全国大会, 2ZC-6, 2010.
4) 植田亘, 高田秀志,“複製計算モデルに基づくオブジェクト共有手法における複製オブジェクトの楽観的な一貫性管理手法の提案”, 第 9 回情報科学技術フォーラム, M-012, 2010.
5) 野口尚吾, 高田秀志,“分散環境における局所的なオブジェクト複製を可能とする階層的グループ管理手法 ”, 第 8 回情報科学技術フォーラム, M-053, 2009.
6) 山本佑樹, 植田亘, 高田秀志,“異種端末混在環境における端末の性能を考慮したオブジェクト複製手法”, マルチメディア, 分散, 協調とモバイル (DICOMO2010) シンポジウム, pp. 1459 - 1466, 2010.