2004－UBI－3　(6)
2004／1／20

Smart Furniture間の柔軟なサービスローミングを実現するミドルウェアの構築

米澤 拓郎[1] 小泉 健吾[1] 守分 滋[2] 永田 智大[2] 徳田 英幸[12]

[1] 慶應義塾大学環境情報学部 [2] 慶應義塾大学大学院 政策・メディア研究科

本稿では，Smart Furniture が作り出す即興的な知的空間，Micro Smart Hot-spot(MSH) を複数協調動作させた Micro Smart Hot-spot Network(MSHNet) を提案する．MSNet はそれぞれの MSH を構成するセンサ，デバイス，アプリケーションを協調させて新しいサービスを実現する．本稿では特にユーザ，機器，アプリケーションの移動を支援し，ユーザの動きに合わせた柔軟なサービスローミングを実現するミドルウェア NICOLA と SaRaRi を設計した．NICOLA は複数の MSH の挙動をユーザが動きによって柔軟に制御できるシステムであり，SaRaRi はアプリケーションに時間連続性のある移動を支援する．評価としてプロトタイプのアプリケーションを作成し，MSNet の有効性を実証した．

Middleware for Flexible Service Roaming between Smart Furnitures

Takuro Yonezawa[1] Kengo Koizumi[1] Shigeru Moriwake[2] Tomohiro Nagata[2]Hideyuki Tokuda [12]

[1] Faculty of Environmental Information,　Keio University
[2]Graduate School of Media and Governance,　Keio University

In this paper, we suggest a "Micro Smart Hot-spot Network(MSHNet)". It consists of many cooperative Micro Smart Hot-spot(MSH)s provided extemporaneously by Smart Furniture. MSHNet can provide new service using sensors, devices and applications in each MSH. This paper especially focuses on the mobility of users, devices and applications, and describes the two middleware NICOLA and SaRaRi designed to realize flexible service roaming involving user's mobility. NICOLA controls many MSH with user's action, and SaRaRi provides mobility with continuity to applications. Lastly, effectiveness of MSHNet is substantiated with a prototype application.

1　はじめに

ユビキタスコンピューティング環境は，多様なセンサやデバイスとそれらのネットワークが埋め込まれ，ユーザに様々なサービスを提供できる．我々はこの環境を実現するために，知的空間 Smart Space[1] を構築した．さらに，建物として構築された静的な知的空間に対し，低コストで即興的に知的空間を作り出すための研究を行っている．この即興的に作り出される知的空間を MSH(Micro Smart Hot-spot）と呼ぶ．MSH は既存の Hot-spot とは異なり，ネットワークの接続性を提供するだけでなく，タッチパネルやスピーカー，センサなどの多様な入出力機能を利用して様々なサービスを提供する．この MSH を実現する手段として，我々は Smart Furniture[2] というデバイスを開発した．

本稿ではこの MSH を複数つなげた MSHNet(Micro Smart Hot-spot Network) を提案する．MSHNet とは，MSH 間のネットワーク通信だけを実現した環境ではない．それぞれの MSH に埋め込まれたセンサやデバイス，それぞれに動くアプリケーションが相互に協調して動作する環境を指す．MSHNet を実現することで，単体の MSH より広範囲を知的空間化できる．本稿では特に複数の MSH 内のユーザや, ユーザが持つ小型機器の移動に対する, アプリケーションコードの移動性を実現することに焦点をあてた．

この MSHNet を実現するため，NICOLA，SaRaRi という２つのミドルウェアを作成し，評価を行った．NICOLA は複数の MSH の挙動を空間の状況に応じて制御する空間プログラミング機構を提供し，SaRaRi は複数の MSH 間において，アプリケーションに時間的連続性を保証した移動支援を提供する．

本稿の構成として，第２節で関連研究をまず述べ，MSHNet について説明する．第3節，第４節では MSHNet を実現するミドルウェア NICOLA と SaRaRi について説明する．第５節では NICOLA と SaRaRi によって構築された応用アプリケーションを用いて MSHNet の評価を行う．そして第６節にまとめとして，結論と今後の展望を述べる．

2　即興的広域ユビキタス空間

この節ではまず既存の知的空間について述べ，次に MSH と，それを実現するデバイス Smart Furniture について述べる．その後に我々の提案する MSHNet について述べ，それを実現するためのミドルウェアについて概説する．

[1] 本研究の Copy and Move モデルに関する研究は文部科学省科学技術振興調整費「人間支援のための分散リアルタイムオペレーティングシステム基盤技術の研究」の支援による．

2.1 関連研究

従来より多くの研究者がユビキタスコンピューティング環境を構築しようと試みてきた．Microsoft の EasyLiving[3]， MIT の Oxygen[4]， CMU の AURA[5] 等のプロジェクトでは，実際にプロトタイプのユビキタスコンピューティング環境を実現した．これらの研究は，いずれも特別な空間を建物内に作り上げたものである．センサやデバイス類がネットワークで相互接続され，それらを扱うミドルウェアが開発されている．

我々も同様に Smart Space と呼ばれる知的空間を構築した．Smart Space は次のような特徴を持つ．
(1) 個々のユーザの認識が可能．
(2) ユーザやデバイスの位置が検知可能．
(3) デバイス類の協調動作が可能．

これらは，VNA[6]，Wapplet[7] 等のミドルウェアによって実現された．VNA は機器の機能を統合し，Wapplet はモバイル機器を他の機器と協調動作させるためのフレームワークを提供する．

2.2 MSH

建物に部屋として知的空間を構築すると，高コストで，移動することが困難であるという欠点があった．MSH の目的とは，移動ができて低コスト知的空間を即興的につくりだすことである．MSH は，Hot-spot の特性である単純なネットワーク到達性に加え，様々なサービスを提供する．例として，MSH に埋め込まれたセンサを利用した状況認識型のアプリケーションが挙げられる．MSH の使用が想定される環境として，プライベートな家庭環境に加え，Office やバス停，イベント会場などの公共空間が挙げられる．

MSH を実現する手段として，我々は Smart Furniture を考案した．Smart Furniture は基本的にタッチパネルと埋め込み型の PC から構成されているが，用途に応じてカメラ，スピーカー，照明等のデバイスや，温度や音声，位置を認識するセンサを取り付けることが可能である．

本稿で使用する Smart Furniture には位置情報センサ [8] がとりつけられており，ユーザの存在の有無だけではなく，その空間でのユーザの細かな位置を捉えることが可能である (図 1)．

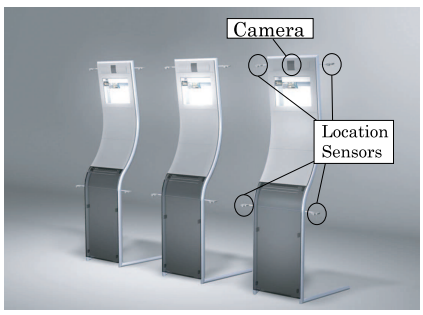


図 1: Smart Furniture の構成

2.3 MSHNet

1 つの MSH ではカバーできる空間の範囲と提供できるサービスに限界がある．MSHNet は MSH 同士に単純なネットワーク通信だけを提供するだけでなく，それぞれのセンサ，デバイス，アプリケーションが相互に協調して動く環境を実現する．この MSHNet を実現することで，サービスを実現するために必要な台数の Smart Furniture を置くだけで広範囲の知的空間を創造できる．

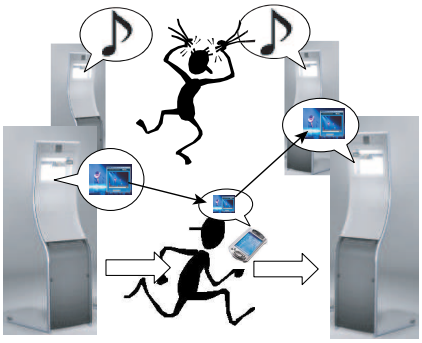


図 2: MSHNet におけるアプリケーション例

MSHNet によって実現されるアプリケーション例を，図 2 に示す．今，位置情報センサを取り付けられた Smart Furniture が複数設置され，MSHNet を構築している．Smart Furniture 同士は Internet に接続され，相互に通信が可能である．手前のユーザは PDA を持ち，MSHNet 内を動きまわる．手元の PDA にはアプリケーションが動作し，ユーザが Smart Furniture に近づいたり離れたりすることでアプリケーションが PDA，Smart Furtiture 間を移動する．ここで MSHNet はユーザの動きに主眼を置いたサービスローミング機構を実現している．一方，奥のユーザは踊っている．MSHNet はユーザの「踊る」という動きを認識し，ユーザの近くの Smart Furniture から音楽を再生する．これは，ユーザの位置を考慮したアプリケーションを実現している．

本稿ではこのように，複数の MSH が協調して，ユーザの動きに合わせたサービスを提供する環境を実現する．

2.4 MSHNet を実現するミドルウェア

本稿で実現する MSHNet 構築に必要な機能を挙げる．

- サービス発見機構

MSHNet は複数の MSH を協調させて動作させる．よって，空間内に存在する MSH，及びそれを構成するデバイスやサービスを発見する機構が必要である．

- ユーザ・機器の位置・動きを認識する機構

ユーザ，機器は MSH 間を移動することが想定される．その位置・動きを捉える機構が必要である．

- MSHNet を制御する機構

ユーザが実現したい要求に応じて MSHNet の動作を定義できる機構が必要である．

- アプリケーションの移動性と時間的連続性を保証する機構

ユーザは MSHNet 内を移動するので，ユーザに連続してサービスを提供するためにはアプリケーションが時間的な連続性を保ったまま複数の MSH 間を移動しなければならない．よってこの機構が必要である．

我々は上にあげた機能を実現する手法として，NICOLA と SaRaRi というミドルウェアを設計・実装した．NICOLA はサービス発見機構，ユーザ・機器の位置・動きを認識する機構，MSHNet を制御する機構の 3 つを実現する．SaRaRi はアプリケーションの移動性と連続性を保証する機構を実現する．この 2 つのシステムについての詳細は次の 3 節，4 節で説明する．

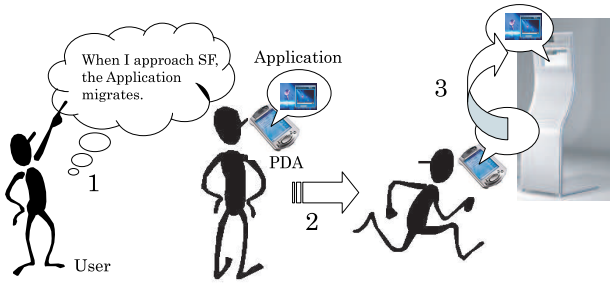
3 NICOLA - Space Programming Framework

この節では MSHNet を実現するためのミドルウェアの一つ，NICOLA について説明する．NICOLA は MSHNet 内で一つのサーバーとして動作することを想定する．

3.1 Space Programming

NICOLA は，室温や人の存在など, ある空間のコンテキスト (状態) における機器の挙動をユーザ自身が柔軟に定義できる環境を構築する．NICOLA では, その定義を文章を記述することによって行う．この文章を Space Program と呼び，Space Program を作成する行為を Space Programming と呼ぶ．

Space Programming の例として，図 3 のようなシナリオを想定する．ユーザは Smart Furniture に近付くと PDA 上のアプリケーションが Smart Furniture に移動する，という定義を行う．そして実際に空間内でユーザが定義した動きを行うと，予めユーザが定義したように機器が動作する．複数の Space Program を組み合わせて，ユーザは空間内に様々なサービスを構築していく．



1. User defines event and action .
2. Event occures.(ex. User approaches to SF.)
3. Action is executed.(ex. Application migrates.)

図 3: Space Programming の例

一般にユーザの要求は空間内のコンテキストの変化に対し生じる．例えば，空間の温度が上昇した時や，自分の位置が変化した時に起こる．この因果関係を記述するために，Space Program は If-then のイベント駆動型の構造を選択した．Space Program は If-部分のイベント部と，それ以後のタスク部に分かれている．ユーザは空間内にあるイベントがおこった時，あるタスクを実行させる，という動作原理を記述する．このイベント駆動型の記述方式はプロダクションシステムで用いられるプロダクションルールの記述方法と同じ性質を持っている．可読性に優れ，ルールの追加削除が容易である．Space Program の例を下に挙げる．

1. If UserA approach Display within 1 m , Display on .
2. If UserA separate SmartFurniture at 2 m , SmartFurniture send Video Phone to PDA .

1 番目のプログラムは UserA がディスプレイの 1 m 以内に近付くとそのディスプレイのスイッチが ON になる，ということを定義している． 2 番目のプログラムは UserA が Smart Furniture から 2 m 離れると，SmartFurniture 上で動いているテレビ電話が PDA に移動する，ということを定義している．次にイベント部分とタスク部分についての詳細を述べる．

- イベント部

イベント部ではユーザが捉えたい空間内の状態を記述する．ユーザが記述できるイベントの例を表 1 に示す．

イベント	move	approach	gesture	···
定義	動く	近づく	指定した手ぶりを行う	···

表 1: イベントの種類

上の例は位置情報センサから得られるコンテキストの変化をイベントとして捉える．

- タスク部

タスク部では，ユーザが指定したイベントが発生した時に，どの機器にどのようなサービスを行わせるかを記述する．提供するサービスはその機器ごとに異なる．下の表に機器とその機器が提供するサービスの例を示す．

機器	Light	Display	Amp	···
タスク	on off	on off	volume up volume down volume mute	···

表 2: 機器とそのサービスの種類

3.2 システム概要

NICOLA システムの構成図を図 4 に示す．NICOLA は以下の 5 つのモジュールからなる．それぞれについて述べていく．

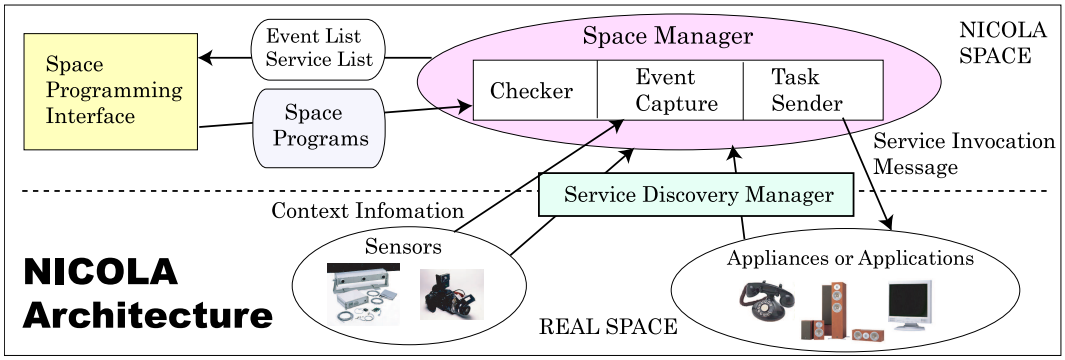


図 4: NICOLA Architecture

- **Space Programming Interface**

Space Programming Interface はユーザに Space Program を容易に記述できるインタフェースを提供する．例として,GUI ベースのインタフェース, 音声入力インタフェース等が挙げられる．また，後述する Service Discovery Manager によって作成された Event List と Service List を参照し，ユーザにデバイス及びサービス情報を提供する．

- **Service Discovery Manager**

Service Discovery Manager は空間内にどのような機器が存在するのか，またその機器上で提供されるサービスはどのようなものかを発見する機構である．この機構で得られた機器・及びサービス情報は Event List, Service List として管理される．

- **Checker**

Checker はユーザの作成した Space Program を分析し，どういうプログラムをユーザが作成したか解析する．また，ユーザが作成したプログラムに誤りがあると，エラーを返す機能を備える．

- **Event Capture**

Event Capture は各センサから送られるコンテキストを解析し，空間内で発生したイベントを認識する．本稿のシステムでは位置情報センサを使用おり，そのセンサ情報の流れを時間軸に沿って分析してユーザの動きを捉る．Event Capture はプロダクションシステムにおける知識ベースに相当する．Event Capture で認識できるイベント数を増やすことで，ユーザの要求に柔軟に応えられる．

- **Task Sender**

Task Sender はサービスを実行するように各機器にメッセージを送る．情報機器上のサービスを起動する手法としては，従来より研究が盛んである．分散オブジェクト技術として代表的なものに CORBA，JavaRMI と EJB，COM+，VNA 等がある．このモジュールではこれらの技術の使用を想定する．

この Checker， Event Capture， Task Sender は NICOLA の最も主要な機能郡 Space Manager を構成している．今回はこの Space Manager と Space Programming Interface の一部を実装した．

3.3 動作概要

NICOLA の動作概要を説明する．NICOLA は以下の順序で動作する．

(1) Space Program の作成

まずユーザが Space Programming Interface を用いて Space Program を生成する．図 5 は今回実装した GUI ベースのインタフェースである．ユーザはユーザ自身の特定の動きをイベントと定義し, それに対する機器のサービスを結びつけ, 視覚的に Space Program を作成する．

図 5: GUI ベースの Space Programming Interface

(2) Space Program の意味解析

ユーザの作成した Space Program は Space Manager の Checker に読み込まれる.Checker はユーザが定義したイベントとタスクを解析し, それぞれの情報を Event Capture と Task Sender に送る．

(3) イベントの認識

Event Capture は Checker から指定されたイベントが発生するのを監視する．各センサから送られるコンテキストを解析し，ユーザが定義したイベントの発生を確認すると，Task Sender に処理を開始するように伝える．

(4) サービスの起動

空間内でイベントが発生すると，処理は Task Sender に移る．Task Sender は Checker から受け取った情報を用い，ユーザが定義した通りに機器上のサービスを起動させる．

4 SaRaRi

本節では，MSHNet におけるアプリケーションの時間的連続性をもったローミングを実現するミドルウェア SaRaRi について説明する．まず SaRaRi の基礎となる Copy and Move モデルについて述べ，次に SaRaRi のシステム概要について解説し，最後にアプリケーションの動作概要について説明する．

4.1 Copy and Move モデル

MSHNet によって実現されるサービスの一つに，ユーザの移動に応じて MSH 間をローミングするアプリケーションが挙げられる．このサービスをローミングするという先行研究として Wapplet[7] が挙げられるが，Wapplet は，一度アプリケーションの実行を中断してローミングを実現する．

しかし，この手法を用いると，アプリケーションを移送している間，ユーザに何もサービスが提供されないサービス断絶時間が生じてしまう (図 6)．音楽プレーヤーや電話などのマルチメディア・アプリケーションのような，時間的連続性が重要な意味を持つデータを扱うアプリケーションにとって，サービス断絶時間は問題となる．例えば，電話での会話が途切たり，音楽が聴けない時間が存在することになる．

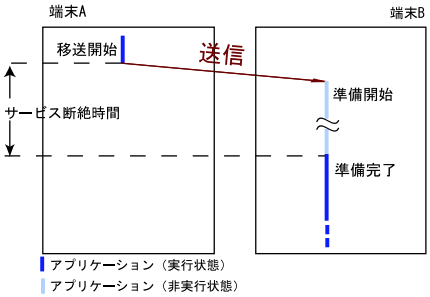


図 6: サービス断絶時間の発生

そこで本システムでは，ユーザが，アプリケーションを移動させる時にサービス断絶時間をなくす Copy and Move モデルを提案する．Copy and Move モデルでは，実行中のアプリケーションを非実行状態で複製し，その複製を他端末へ送信するという手法を用いることによって，アプリケーションの時間的連続性を実現している．しかし，このままでは移送する度に複製が増え続け，端末の計算資源を浪費してしまう．そこで，SaRaRi は複製状態監視機能を備えており，その複製が実行状態へ遷移したことを検知した後にオリジナルを削除する (図 7)．

4.2 システム概要

本項では，SaRaRi の構成と実装について説明する．SaRaRi は，Copy and Move モデルを用いたサービス断絶時間のないアプリケーション移送を提供するミドルウェアである．SaRaRi では，アプリケーションをオブジェクトとして扱う．図 8 に SaRaRi の構成図を示す．構成は**管理部**，**送受信部**，**状態監視部**の三つに分けられる．また，実行中のオブジェクトを保管する ObjectStack も存在する．ここでは順に各部について述べた後，SaRaRi 上で動作するオブジェクトについて解説する．なお，SaRaRi の開発言語は Java を用いた．これは，Java を用いることによって，バイト列化されたオブジェクトのネットワーク間移送が容易に行なえるためである．

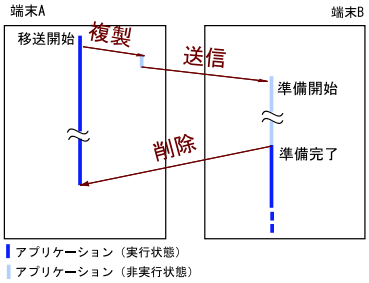


図 7: 断絶時間のないサービスローミングのモデル図

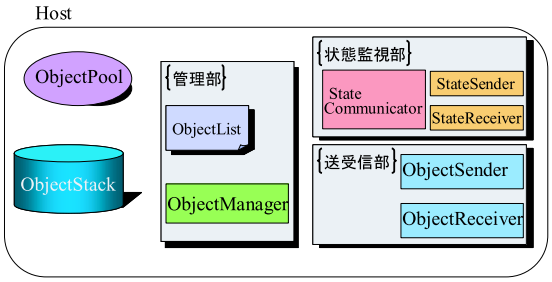


図 8: SaRaRi の構成図

4.2.1 システム構成

ここではシステムを構成する管理部，送受信部，状態監視部について述べる．

(1) 管理部

管理部は移送するオブジェクトを取りまとめる．管理部には次のモジュールが存在する．

- ObjectManager
  管理部の中心的な役割を担う．具体的には，ObjectStack からのオブジェクトの取り出し，オブジェクトの削除，ObjectList の更新，及び送受信部へのオブジェクトの受け渡しを行う．
- ObjectList
  ObjectStack に保管されているオブジェクトの一覧表を管理する．更新は ObjectManager によって行われる．

(2) 送受信部

送受信部はオブジェクトを他端末へ送信，もしくは他端末から送られてきたオブジェクトを受信する．送

受信部には次のモジュールが存在する．

- ObjectSender
  オブジェクトを他端末へ送信する．
- ObjectReceiver
  送信されてきたオブジェクトを受け取る．

(3) 状態監視部

状態監視部は，オブジェクトが実行状態に遷移するまで監視する．状態監視部には次のモジュールが存在する．

- StateSender
  オブジェクトが実行状態に遷移したという通知を送信側端末へ送信する．
- StateReceiver
  オブジェクトが実行状態に遷移したという通知を受信側端末から受け取る．
- StateCommunicator
  そのオブジェクトが移送されてきたものであった場合，オブジェクトが実行状態に遷移したことを ObjectSender を用いて通知する．そのオブジェクトが他端末へ送信したものであった場合は，StateReceiver で他端末からの通知を待つ．

4.2.2 オブジェクト

Object は，SaRaRi によって複製時やネットワーク送信時などに呼び出される共通のインタフェースを実装する必要がある．そこで SaRaRi では Migrator クラスを用意し，各オブジェクトはこのクラスを継承することによってインタフェースを実装する．その Migrator クラスが提供するインタフェースの名称と機能を表 3 に示す．

メソッド	説明
copy()	オブジェクトを複製する
dispatch()	オブジェクトを移送可能な状態にする
arrive()	オブジェクトを実行状態にする
finish()	オブジェクトを終了し，削除する

表 3: Migrator クラスのインタフェース

4.3 動作概要

SaRaRi のオブジェクト移送手順は 4 段階に分かれている．本項では各段階について順に述べる．送信側の手順を図 9 に，受信側の手順を図 10 に示す．

(1) オブジェクトの複製

ObjectList に掲載されたオブジェクトに対して移送要求が発生すると，まず ObjectManager が目的のオブジェクトを ObjectStack から検索して取り出す（図 9 の①）．そして，そのオブジェクトの実行状態を維持したままオブジェクトを複製する (図 9 の②)．

(2) 複製されたオブジェクトの移送

複製されたオブジェクトが，ObjectManager から ObjectSender へ渡される (図 9 の③)．ObjectSender は，オブジェクトをバイト列化し，ネットワークを通

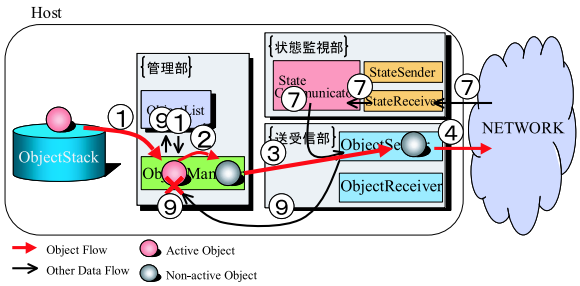


図 9: 送信側端末における移送手順

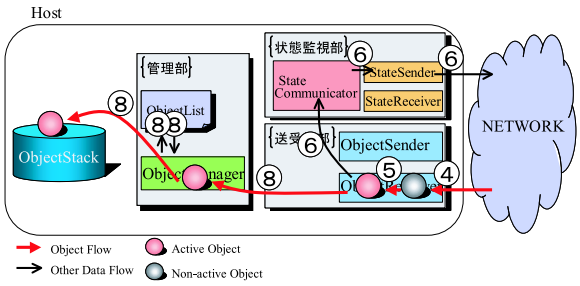


図 10: 受信側端末における移送手順

して受信側端末の ObjectReceiver へ送信して待機状態になる (図 9・図 10 の④)．

(3) オブジェクトの実行状態遷移までの待機

受信側端末の StateCommunicator によって，ObjectReceiver が保持しているオブジェクトは実行状態に遷移するまで監視される (図 10 の⑤)．遷移したことが検知されると，受信側端末では，StateSender によって送信側端末へ ack が返される (図 10 の⑥)．送信側端末では，StateReceiver が ack を受け取り，StateCommunicator を通して ObjectSender へ通知される (図 9 の⑦)．

(4) オブジェクトの格納と削除

受信側端末では，ObjectReceiver がオブジェクトを ObjectManager へ渡し，ObjectManager がリストを更新してオブジェクトを ObjectStack に格納する (図 10 の⑧)．送信側では，ObjectSender から通知が来たことを知らされた ObjectManager が，移送した複製オブジェクトのオリジナルを削除し，ObjectList を更新してオブジェクトの移送が完了する (図 9 の⑨)．

5 アプリケーションの実装と評価

MSHNet を移動するアプリケーションとして，テレビ電話アプリケーション「MovePhone」を作成した．本節では，まず MovePhone の実装について述べる．次に MovePhone を用いて，2.4 で挙げた機能要件が満たされたかどうかを評価する．

5.1 MovePhone の実装

MovePhone は Smart Furniture に接続された USB カメラとマイクから動画と音声をキャプチャして，通信

相手にRTP[11]を用いて送信し，相手からも同様に動画と音声を受け取るアプリケーションである．送信方法にはマルチキャストを用いているので，複数の通信相手に動画と音声を送信することができる．MovePhoneの実装にはJMF(Java Media Framework)[12]を用いた．NICOLAがMovePhoneの移動のタイミングと移動先を指定し，SaRaRiが実際にMovePhoneを移動させる．

5.2 評価

本項では，本稿で実現したMSHNetの性能を評価するため，2.4の機能用件のうち，特にアプリケーションの時間的連続性について評価を行う．MovePhoneが，どれほどのオーバーヘッドでMSH間を移動するのかを示し，そのMSHNetが提供するアプリケーションの移動性について評価する．次にテレビ電話映像のRTPパケットのシーケンス番号の推移を示し，MSHNetが提供するアプリケーションの移送時における時間的連続性について評価する．なお，計測環境としてPCを3台用意し，2台をSmart Furniture用，もう一台をMovePhoneの通信相手用PCとして使用した(表4)．それぞれの端末は100Base-Tで相互に接続されている．

	SF1用PC	SF2用PC	通信相手用PC
PC	ThinkPad X30	ThinkPad X30	ThinkPad X30
CPU	1.2GHz	1.2GHz	1.2GHz
メモリ	760MB	256MB	760MB
OS	Windows XP	Windows XP	Windows XP

表 4: 評価用端末の性能表

5.2.1 アプリケーションの移動性

ここではMovePhoneを用いてアプリケーションの移動性について評価する．まずNICOLAで以下のSpace Programを作成した．

```
If User approach SF2 within 1.5 m , SF1 send MovePhone to SF2 .
```

このSpace Programはもしユーザが Smart Furniture2の1.5m以内に近付いたら，Smart Furniture1からSmart Furniture2に向けてMovePhoneが移動するということを定義している．この”近付く”というイベントをNICOLAが認識してからアプリケーションの移送が完了するまでの各段階における所要時間を図11に示す．

イベントの発生からMovePhoneの移送完了までの合計時間は約4.1秒であった．最も時間を必要としたのはオブジェクトが実行状態に移るまでに要した時間であり，約3.8秒かかった．この時間は，JMFを用いた動画表示の開始までにかかった時間である．また，２番目に必要としたのがオブジェクトを送信する時間であり，約160ミリ秒であった．この時間はJavaを用いたオブジェクトのバイト列化と非バイト列化，およびバイト列の送信にかかった時間である．つまり，この２点はMovePhoneのようなオブジェクトを移送する際に必然的にかかる時間であるため，本稿で構築したMSHNetによるアプリケーション移送のオーバーヘッドはこの２点を除いた時間であると言える．

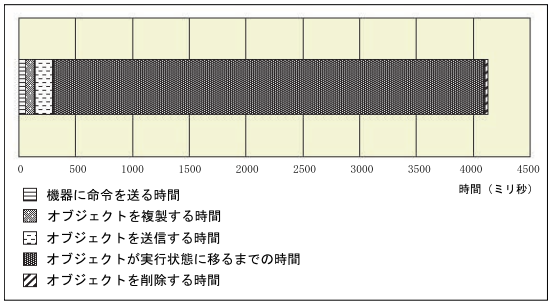


図 11: イベントの発生から移送完了までの時間

そこで，本稿で実現したMSHNetのオーバーヘッドを評価するため，この２点以外の時間について考察する．NICOLAがイベントをSaRaRiに通知し終わるまでの時間は63ミリ秒，オブジェクトの複製を作成するまでの時間は83ミリ秒，オブジェクトを削除するまでにかかった時間は30ミリ秒であった．したがって，本稿で構築したMSHNetのオーバーヘッドは合計176ミリ秒であり，MovePhoneの移送に要した時間全体の約４％であった．

5.2.2 アプリケーションの連続性

前小項での，MovePhoneが送信され実行状態になるまでに要した約3.9秒以上の時間は，従来のサービスローミング手法ではユーザにとってサービス断絶時間となる．そこで本小項ではSaRaRiによってアプリケーションの連続性が実現されたことを示す．MovePhoneはテレビ電話アプリケーションであるため，ユーザにサービスが提供されている時間は，Smart Furniture上に相手の動画が表示され音声も聞くことができ，なおかつ相手にもこちらの動画と音声が届いている状態と定義できる，本評価では受信側の動画に焦点を当て，相手の動画がSmart Funitureの画面上に表示されてから消えるまで，相手から送られてきたRTPのシーケンス番号を計測する．

アプリケーションの送信側Smart Furnitureを端末A，受信側Smart Furnitureを端末Bとする．端末A上でMovePhoneが起動され相手の動画を表示してから端末Bへの移送完了後に削除されるまで，また，端末B上で送られてきたMovePhoneが再び画面に相手の動画を映し出してから実験者によって終了されるまで，RTPパケットのシーケンス番号を約120ミリ秒間隔で計測した．得られた値の中からMovePhoneが端末Aから端末Bへ移送された部分を抜き出し，SaRaRiを用いない場合のグラフを図12に，SaRaRiを用いた場合のグラフを図13に示す．

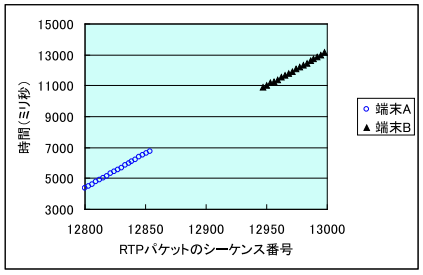


図 12: RTP のシーケンス番号の推移（SaRaRi なし）

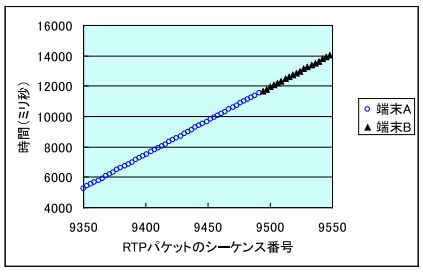


図 13: RTP のシーケンス番号の推移（SaRaRi あり）

図 12 では，MovePhone を端末 A から端末 B へ移送させることによって約 4.1 秒間 RTP パケットが得られていない．このことから約 4.1 秒間，送られてきた RTP パケットが破棄され，通信相手の動画が両端末の画面に表示されなかったことがわかる．これに対し，SaRaRi を用いた図 13 では，端末 A で最後の RTP パケットを計測してから 120 ミリ秒後に端末 B の最初の RTP パケットが計測された．SaRaRi では端末 B が動画表示を開始してから端末 A の動画表示を終了するため，端末 A と端末 B で同じシーケンス番号が計測されるべきだが，前小項でのその所要時間が約 30 ミリ秒であることから，本小項の計測でも 120 ミリ秒以内に終了したと考えられる．したがって，120 ミリ秒以内の重複はあるものの，前項で示した約 3.9 秒のサービス断絶時間が 0 ミリ秒になった言える．

6 まとめ

本稿では，Smart Furniture によって即興的に構築される知的空間 MSH を複数つなげた MSHNet を提案した．MSHNet は，それぞれの MSH に接続されたセンサやデバイス，それぞれに動くアプリケーションを協調動作させる環境を実現し,Smart Furniture 単体によって構築される範囲よりも，より広範囲を知的空間化できる．

MSHNet を構築するに当たり，本稿では特に複数の MSH 内のユーザ，機器の移動に対する，アプリケーションコードの移動性を実現することに焦点を当て，NICOLA と SaRaRi という 2 つのミドルウェアの構築を行った．NICOLA は，複数の MSHNet の挙動を空間の状況に応じて制御する Space Program を実現し，SaRaRi は複数の MSH 間で，サービス断絶時間を伴わない，Copy and Move モデルに基づいたサービスローミングを可能にした．

マルチメディア・アプリケーションのプロトタイプとして実装した MovePhone を用いて，NICOLA と SaRaRi によって実現した MSHNet を，アプリケーションの時間的連続性について評価した．NICOLA が SaRaRi に MovePhone 移送命令を出してから移送が完了するまでの時間のうち，MSHNet によって生じたオーバーヘッドは 176 ミリ秒であり，移送時間全体の 4%であった．さらに残りの 96%に相当する約 3.9 秒は，従来のサービスローミング手法ではサービス断絶時間となるのに対し，MSHNet では 120 ミリ秒以内のサービス重複を伴うだけでサービス断絶時間とならないことを示した．

今後はさらに対象とする領域を増やし，MSHNet のサービス発見機構や，ローミング時のサービス重複の際の一貫性の維持，より多くのサービスを用いてより複雑な MSHNet のサービスを実現する．

参考文献

[1] Okoshi, T., Wakayama, S., Sugita, Y., Iwamoto, T., Nakazawa, J., Nagata, T., Furusaka, D., Iwai, M., Kusumoto, A., Harashima, N., Yura, J., Nishio, N., Tobe, Y., Ikeda, Y., and Tokuda, H.: Smart Space Laboratory Project: Toward the Next Generation Computing Environment. *IEEE Third Workshop on Networked Appliances (IWNA)*2001

[2] Ito, M., Iwaya, A., Saito, M., Nakanishi, K., Matsumiya, K., Nakazawa, J., Nishio, N., Takashio, K. and Tokuda, H.: Smart Furniture: Improvising Ubiquitous Hot-spot Environment*IEEE 3rd International Workshop on Smart Appliances and Wearable Computing*. May. 2003 pp. 248-253

[3] Brumit, B., Meyers, B., Krumn, J., Kern, A. and Shafer, S.: Easyliving:Techonologies for intelligent environments *International Workshop on Cooperative Buildings. 1999*

[4] MIT Project Oxygen. http://oxygen. lcs. mit. edu/

[5] Garlan, D., Siewiorek, D., Smailagic, A. and Steenkiste, P.:Project Aura: Toward Distraction-Free Pervasive Computing. *IEEE Pervasive Computing*. April-June 2002

[6] Nakazawa, J., Tobe, Y. and Tokuda, H.: On dynamic service integration in vna architecture.*IEICE Transactions on Fundamentals of Electronics. Communications and Computer Sciences*. vol. 7. no. E84-A. pp. 1610-1623.2001.

[7] Murase, M., Iwamoto, T., Nagata, T., Nishio, N. and Tokuda, H.: Implementation and evaluation of wapplet framework.*Proceedings of International Workshop on Networked Appliances*. 2002.pp. 275-284

[8] 西田佳司, 西谷哲史, 相澤洋志, 堀俊夫, 溝口博: ポータブルな超音波３次元タグ -簡単なキャリブレーション手法-*The 21st Annual Conference of the Robotics Society of Japan*(2003)

[9] Manuel Roman, Christopher K. Hess. Renato Cerqueira. Anand Ranganathan. Roy H. Campbell. and Klara Nahrstedt.: Gaia: A Middleware Infrastructure to Enable Active Spaces. *In IEEE Pervasive Computing*. pp. 74-83. 2002.

[10] Alan F. Blackwell and Rob Hague. AutoHAN: An Architecturefor Programming the Home. *In IEEE Symposia on Human-Centric Computing Language and Environments*. pp. 150-157. IEEE. 2001.

[11] RFC 1889 - RTP: A Transport Protocol for Real-Time Applications

[12] Sun Microsystems. Java Media Framework http://java. sun. com/products/java-media/jmf/