Touch-and-Connect：ユビキタス環境における接続指示フレームワーク

岩崎陽平[†] 河口信夫[†,††] 稲垣康善[†††]

近年，様々な機器が高性能なコンピュータを内蔵し，至るところにコンピュータが存在するユビキタスコンピューティング環境が実現しつつある．しかし，機器間を接続して利用するためには，一般にアドレスや名前の指定などの煩雑な設定が必要となる．本論文では，接続したい両機器のボタンを押すことにより，機器間の接続を直接的に指示できるフレームワーク Touch-and-Connect を提案する．本手法では，状態を表示可能なボタンを用いてユーザの操作を排他制御し，複数の人間が独立に操作を行う状況においても誤接続を防止する．ブロードキャスト通信を用いた本手法のプロトコルは，管理サーバを必要とせず，動的な端末の入退出も考慮しているため，必要に応じて一時的に構築されるアドホックネットワークでも利用可能である．本フレームワークのプロトタイプシステムを実装してその実現可能性を示し，被験者実験によりその使いやすさを評価した．

Touch-and-Connect: A Connection Request Framework for the Ubiquitous Computing Environment

Yohei Iwasaki,[†] Nobuo Kawaguchi[†,††] and Yasuyoshi Inagaki[†††]

The spread of short distance wireless communication technology is making it possible for various information appliances to communicate through a network. However, to connect and to use them, we must generally perform complicated setup operations such as setting addresses or names. In this paper, we propose a user-friendly connection request framework called “Touch-and-Connect”. With this method, we can connect any networked devices only by touching them. This system is designed so that it can work without a server, and even if many individuals operate it independently, no incorrect connections occur. And we have exemplified the usefulness of this framework with subject experiments.

1. はじめに

近年，ハードウェア技術の進歩により，様々な機器が高性能なコンピュータを内蔵して小型化し，至るところにコンピュータが存在するユビキタスコンピューティング環境が実現しつつある．Weiser[1)] は，人間がコンピュータのことを意識しなくても，コンピュータが自律的に我々の生活を支援してくれるという，透明（invisible）なコンピュータシステムのビジョンを描いた．しかし，ユーザの多種多様な意図を，ユーザからのいっさいの入力なしにコンピュータが理解することはいまだ困難である．したがって，コンピュータシステムの複雑さをユーザに意識させることなく，ユーザの意図をシステムに容易に伝えられる手段が望まれる．

一方，無線 LAN や Bluetooth などの短距離無線通信技術の普及により，AV 機器，文房具，家電製品など，今までネットワークに参加することのなかった様々な機器が，ネットワークの一員となりつつある．これらの機器をネットワーク経由で接続し，互いに連携させることができれば，ユーザにとって有用なサービスを実現できる．ネットワーク上の機器間を接続するための技術仕様として，すでに Universal Plug and Play[2)] や，ECHONET[3)] などが提唱されているが，ユーザが望んでいる連携状態をシステム側に伝えるためには，一般に煩雑な設定が必要となる．たとえば，2 つの小型の無線機器間の接続を想定した場合，小さなディスプレイ上で多くの候補の中から接続先機器の名前を選択するというような設定は直観的ではなく，このような煩雑な設定なしに，機器間を容易に接続で

† 名古屋大学大学院情報科学研究科
Graduate School of Information Science, Nagoya University
†† 名古屋大学情報連携基盤センター
Information Technology Center, Nagoya University
††† 愛知県立大学情報科学部
Faculty of Information Science and Technology, Aichi Prefectural University

きる手段が望まれる．

ユーザの行動をセンサなどで認識し，機器間の接続などの設定を自動的に行う研究が行われている．たとえば，Follow-me application[4]では，ユーザの位置をセンサにより取得し，ユーザに最も近いディスプレイが自動的に作業環境として選択される．しかし，多数の機器が存在する環境では自動設定には限界があり，ユーザが本当に望んでいる状態を，システムに容易に伝えられる手法が必要である．Reactive environments[5]では，ユーザが接続したい両機器に取り付けられたボタンを押すことにより，直接的に接続を指示できるが，複数のユーザが独立に操作を行う状況を想定していない．また既存のシステムの多くは，一般にサーバにより集中管理されており，モバイル環境や，必要に応じて一時的に構築されるアドホックネットワークでの利用は困難である．

本論文では，接続したい両機器に直接触ることにより，様々な機器間の接続を容易に指示できるフレームワーク Touch-and-Connect を提案する．本手法では，ユーザの操作間の排他制御を行うロックメカニズムにより，複数の人間が独立に操作を行う状況においても誤接続を防止する．ブロードキャスト通信を用いた本手法のプロトコルは，管理サーバを必要とせず，動的な端末の入退出も考慮しているため，必要に応じて一時的に構築されるアドホックネットワークでも利用可能である．

2. Touch-and-Connect フレームワーク

本章では，機器間の接続指示フレームワーク Touch-and-Connect を提案する．本手法を用いれば，ユーザは接続したい両機器に直接触ることにより，アドレスや名前を指定することなく，様々な機器間の接続を容易に指示できる．機器に直接触る必要があるため，手の届く範囲にある比較的近くの機器への適用を 1 つの利用シーンとして想定する．

本フレームワークでは，Bluetooth や無線 LAN などの短距離無線通信ネットワークを仮定する．赤外線通信と比較すると，短距離無線通信を用いることにより，自由な位置で機器を利用でき，また 3 台以上の機器間の接続，多対多の接続など，より多くの機器との連携を実現できる．また，基地局型の無線ネットワークだけでなく，アドホックネットワークでも利用可能とすることを，本フレームワークの要件とする．ただし，比較的近くの機器間の通信を想定しているため，マルチホップのメッセージは想定せず，全機器間は直接通信可能とする．アドホックネットワークとは，必要に応じて一時的に構築されるネットワークであり，(1) 管理サーバや事前の設定を前提としない，(2) 端末の入退出によりネットワーク構成が動的に変化する，などの特徴がある．アドホックネットワークでも利用可能とすることにより，いつでも，どこでも，提案する接続指示手法を使うことができる．たとえば，外出先でモバイル機器間を接続する場合や，事前に設定がないノート PC 間を接続するような場合にも利用可能となる．

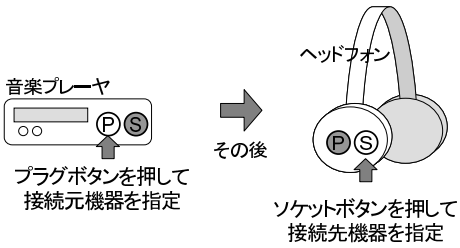


図 1 ボタン 2 つ版インタフェース
Fig. 1 Two-button interface.

2.1 ボタン 2 つ版インタフェースとロックメカニズム

本フレームワークでは，ユーザが接続したい機器を指示するための方法（接続指示インタフェース）を複数用意している．ボタン 2 つ版インタフェースは，この中で最もシンプルで標準的なインタフェースであるため，以後，これを用いて本フレームワークを説明する．他の種類のインタフェースについては 2.5 節で述べる．

本インタフェースでは，**図 1** のように，各機器にプラグボタン（P）とソケットボタン（S）を設ける．機器 A（たとえば携帯音楽プレーヤ）と機器 B（たとえばヘッドフォン）を接続したい場合には，まず，機器 A のプラグボタンを押すことによってそれが接続元機器であることを指示し，その後，機器 B のソケットボタンを押すことによってそれが接続先機器であることを指示する．プラグをソケットに差し込むというメタファにより，ユーザはこの操作を容易に理解することができる．

ここで，複数のユーザが独立に操作する状況を想定し，以下のような順番でボタンが押された場合を考える．

(1) ユーザ 1 が機器 A のプラグボタンを押す．
(2) ユーザ 2 が機器 C のプラグボタンを押す．
(3) ユーザ 2 が機器 D のソケットボタンを押す．
(4) ユーザ 1 が機器 B のソケットボタンを押す．

この場合，システム側はだれがボタンを押したのかを認識できないため，ユーザ 1 は機器 A と B の，ユー

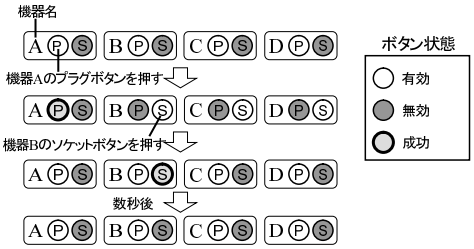


図 2 ボタンの状態遷移例
Fig. 2 Example sequence of button state.

ザ 2 は C と D の接続を意図しているにもかかわらず，A と D，C と B が接続されてしまう可能性がある．

無線ネットワークでは，このように他人の操作との干渉が発生する可能性がある．すぐ近くの他人の場合は，コミュニケーションをとって解決することも可能であるが，無線電波は壁をつきぬけるため，見えない隣の部屋の接続操作が干渉する場合もある．また，会議室などで多数の人間がいっせいに接続を行う状況では直接のコミュニケーションには限界がある．このような誤接続を防止する方法の 1 つとして，ボタンを押しているユーザを認識する方法が考えられる．しかし，ユーザの識別のためには，たとえば ID を持ったデバイスをユーザが携帯する必要があったり，ID を読み取るために特別なセンサが必要であったりする．

本手法では，ユーザの操作間の排他制御，すなわち，あるユーザが操作をしている間，他のユーザの操作を禁止することにより，ユーザを識別することなく誤接続を防止する．このために，状態を表示可能なボタンを用いたロックメカニズムを導入する．各ボタンは「有効」,「無効」,「成功」のいずれかの状態を表示できる．ユーザは，ボタンが有効状態のときにのみ押すことができ，押した後，成功状態になったことを確認する．この成功状態により，ユーザは自分の操作の成功を確認できる．また，あるユーザが接続操作を行っている間は，他のすべての機器はロックされ，プラグボタンが無効状態となる．これにより，誤接続の原因となる他人の操作の開始を防止できる．先ほどの例の場合，ユーザ 1 の操作中は，A 以外の全機器のプラグボタンが「無効」となり，ユーザ 2 は操作を開始できない．この場合のボタンの状態遷移例を**図 2** に示す．

また，操作中に手動または自動で接続指示をキャンセルする機能がある．(1) プラグボタンを押して接続元を指定した後，再度プラグボタンを押すことにより，接続先の指定をキャンセルすることができる．(2) ロックしたまま放置された場合に対応するため，プラグボタンを押してから一定時間経過後，自動的にキャンセルされる．

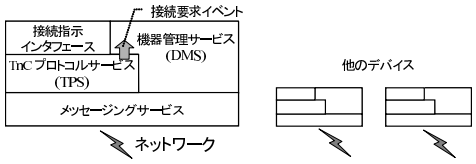


図 3 システム構成
Fig. 3 System architecture.

また，接続元専用・接続先専用の機器も考えられる．この場合，片方のボタン（プラグボタンまたはソケットボタンのどちらか）のみを機器に取り付ければよい．

これらの機能を実現するプロトコルの詳細は，2.4 節に示す．

2.2 システム構成

図 3 に本フレームワークのシステム構成を示す．本フレームワークは，メッセージングサービス，接続指示インタフェース，Touch-and-Connect プロトコルサービス（TPS: Touch-and-Connect Protocol Service），機器管理サービス（DMS: Device Management Service）の 4 つのモジュールからなる．

メッセージングサービス

メッセージングサービスは，様々なネットワーク媒体やネットワークプロトコルの違いを隠蔽する．また，各ソフトウェアサービスに対してユニークなアドレスを割り当て，ソフトウェアサービス間の非同期メッセージ通信機能を提供する．DMS と TPS は，他の機器と通信するために，このメッセージングサービスを使う．

接続指示インタフェース

接続指示インタフェースは，本フレームワークにおいて，ユーザが機器を指定するためのヒューマンインタフェース，およびそれを管理するソフトウェアサービスである．前述したボタン 2 つ版インタフェースだけでなく，ボタン 1 つ版のインタフェースなども存在する（2.5 節参照）．

Touch-and-Connect プロトコルサービス（TPS）

Touch-and-Connect プロトコルサービス（TPS）は，2.4 節で述べるプロトコルを実装しており，本フレームワークの中心となるサービスである．TPS は，接続指示インタフェースの状態（ボタンの色など）の管理や，ユーザからの入力の処理を行い，また接続操作が行われた際には DMS に接続要求イベントを通知する．

機器管理サービス（DMS）

機器管理サービス（DMS）は，本フレームワークのアプリケーションレイヤである．DMS は機器固有の

機能を管理し，その実装は機器の種類によって異なる．

本フレームワークが DMS に対して要求する機能は以下のものである．

- DMS は**機器情報**を提供する．機器情報はデバイスの性質を表したデータ構造であり，詳細については 2.3 節で述べる．
- DMS は**接続要求**イベントを受け取り，それに応じた適切な動作を実行する．接続要求イベントは，ユーザが接続を指示した際に TPS から通知される．

一般に，DMS は機器間の接続状態を管理する機能を持つが，どのように接続状態を管理するかは本フレームワークでは規定していない．たとえば，各機器が接続相手のアドレスのリストを持ち，接続要求イベントを受け取った際に，接続相手のアドレスをリストに追加する．すでにアドレスが存在する場合には削除すれば，切断操作も可能になる．また，接続状態を不揮発性メモリなどに記憶しておき，再起動した際に接続状態を復元（同じ機器に自動的に接続）することにより，ユーザの利便性を向上させることもできる．

接続要求イベント

ユーザが接続指示インタフェースにより 2 つの機器間の接続を指示した場合，接続する両機器各々において，接続要求イベントが TPS から DMS へ通知される．接続要求は以下の情報からなる．

- 接続相手の**機器情報**
- **接続元フラグ：**自分が接続元（1 番目に指定された機器）であるかどうか．この情報は，接続の方向により動作を変更する場合に用いる．

2.3 機器情報

本フレームワークでは，各機器は機器情報というデータを持つ．機器情報は，アドレス，機器の Type，グループ ID，拡張情報からなる．機器情報は，TPS において，機器間の接続可能性の判断するために，また DMS において，接続要求時に適切な連携動作を自動的に選択するために用いられる．

機器の Type と接続可能性

本フレームワークは，様々な種類の接続に対して一般的に用いることができる．接続する 2 つの機器が指定されて接続要求イベントが発生した際に，DMS において適切な連携動作（機器間連携に用いるアプリケーションプロトコルなど）を自動的に選択するために，各機器に Type（機器の種類）を定義する．ここでは単純な Type として，light（ライト），video-player（ビデオプレーヤ），music-player（音楽プレーヤ），display（表示装置），speaker（スピーカ），volume-controller（ボリュームコントローラ）などを考える．接続要求時には，接続する 2 つの機器の Type に基づいて適切な動作が自動的に選択される．たとえば，video-player と display を接続する場合には映像伝送が，video-player と speaker を接続する場合には音声伝送が選択される．

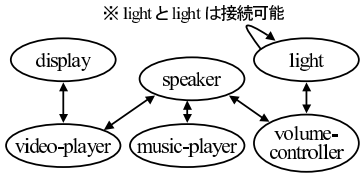


図 4 接続可能性の例
Fig. 4 Example of connectability-relation.

使いやすさを考慮すると，接続操作中にロックを行う機器はできる限り少ないことが望ましい．本手法では，存在しうる機器の Type の集合とそれらの接続可能性を事前に限定することにより，ロックが必要な機器の Type を制限する．これにより，プラグボタンが押されたときに，すべての種類の機器をロックするのではなく，必要最小限の種類の機器のみをロックできる．ここで,「A から B へ接続可能」とは，機器 A から機器 B への接続要求（A のプラグボタンを押し，B のソケットボタンを押すこと）に対して，適切な動作が定義されていることを意味する．接続可能性は，2 つの Type 間の関係として定義する．接続可能性の定義例を有向グラフに表したものを**図 4** に示す．なお，ここでの「接続可能性の方向」は,「ボタン操作の順番（どちらの機器のボタンを先に押したか）」であり，通信の方向（実際の連携動作時に，情報がどちら方向に伝えられるか）とは関係ない．

接続操作中，プラグボタンが押された機器（接続元機器）から接続可能な機器のみが，ソケットボタンが有効状態となり接続先として指定できる．また，この「接続可能な機器」へ接続可能な機器のみがロックの対象となる．たとえば，先ほどの定義の場合，light のプラグボタンが押された場合は，light から接続可能な light，volume-controller のプラグボタンのみが有効となり，またこれらに接続可能な light，speaker，volume-controller のみがロックの対象となる．

Type 間の接続可能性を判断できるようにするために，接続可能性を記述した**図 5** のような設定ファイルを各機器に持たせる．名前の衝突を防止するために，機器の Type 名は，この図の例のように Type を定義した組織のドメイン名を含むべきである．また，将来作成されたデバイスとの接続性を確保する方法として，設定ファイルを動的に更新できるような仕組み，たと

```
bidirectional-link: org.cogma.tnc.display, org.cogma.tnc.video-player
bidirectional-link: org.cogma.tnc.speaker,
                    org.cogma.tnc.volume-controller
bidirectional-link: org.cogma.tnc.speaker, org.cogma.tnc.music-player
bidirectional-link: org.cogma.tnc.speaker, org.cogma.tnc.video-player
self-link: org.cogma.tnc.light
```

※self-link は，同一タイプ間のリンクを意味する

図 5 接続可能性を記述した設定ファイル
Fig. 5 Configuration file of connectability-relation.

えば，ネットワーク内に更新情報をブロードキャストしたり，インターネットに接続可能であったりすれば，サーバで更新情報を配布したりするような仕組みの導入も考えられる．

グループ ID

本手法では，プラグボタンを押した後，悪意のある他人にソケットボタンを押されることにより，意図しない接続が行われてしまう可能性がある．また，多数の接続が行われる場合には，ロックが解除されるまで長時間待たされる可能性がある．

これらの問題に対応するため，本フレームワークでは，グループを作成して接続可能な機器を制限することができる．同じグループに属する機器間のみが接続可能，およびロックが必要となる．グループを実装するために，各機器はグループ ID を持つ．同一のグループ ID を持つ機器が同じグループに属すると見なされる．

機器は初期状態では「パブリックグループ」に属する．機器の属するグループを変更したい場合は，機器のグループ ID を変更する必要がある．グループ ID を設定するためのユーザインタフェースとしては，ボタンなどにより直接 ID を入力する方法のほかに，より簡単な方法としてグループチェンジャの利用が考えられる．グループチェンジャは，グループ ID を変更する目的で使用する特別な機器であり，「設定用 ID」を持っている．設定したい機器からグループチェンジャへの接続指示を行うと，グループチェンジャの設定用 ID が接続元機器のグループ ID として設定される．したがって，グループ ID を入力することなしにグループ ID の設定を行うことができる．グループチェンジャの設定用 ID は，初期状態でユニークな値が割り当てられている．何らかの UI を設けてこの設定用 ID を変更可能としてもよい．

拡 張 情 報

拡張情報は機器の Type に依存した情報である．接続要求時に DMS に通知される接続要求イベントのパラメータに，接続相手の拡張情報が含まれており，DMS において機器間の連携を実現するために用いられる．拡張情報の具体的な内容はアプリケーション依存であるが，たとえば，アプリケーションプロトコル通信用のアドレス・ポート番号等などである．

2.4 プロトコル

本節では，Touch-and-Connect プロトコルサービス（TPS）で実装されるプロトコルについて述べる．本プロトコルは，必要に応じて一時的に構築されるアドホックネットワークでも利用可能とするため，管理サーバの不使用，および動的な端末の入退出の考慮を要件としている．サーバでの集中管理の代わりに，ブロードキャスト通信を用いた状態の分散管理を行う．また，ブロードキャストメッセージの送信を定期的に繰り返すことにより，動的な端末の入退出に対応する．

本プロトコルのシーケンス図を**図 6** に示す．プラグボタンが押され接続元として指定された機器（リクエスタ）は，接続可能な他の全機器に要求（**リクエスト**）を行い，ソケットボタンが押され接続先として指定された機器（リプライヤ）が，このリクエストに応答（**リプライ**）することにより，接続操作が確定する．またこの際，リクエスタは，他の機器を**ロック**してプラグボタンを無効とし，他人の操作開始を禁止することにより，誤接続を防止する．

各機器はリクエストリストとロックリスト（初期状態は空）を持つ．リクエストリストには，現在自分にリクエストしている機器のアドレスが格納される．ロックリストには，現在自分をロックしている機器のアドレスが格納される．

以下，プロトコルのルールを述べる．時間パラメータとして，request-wait，cancel-wait，success-time（デフォルト値は後述）がある．ルールを 2 つの並列状態遷移図で表したものを，**図 7** に示す．

リクエスト

プラグボタンが押された機器（リクエスタ）は，Request メッセージをブロードキャストする．接続可能性を判断するために，このメッセージにはリクエスタの機器情報（拡張情報を除く）が含まれる．Request メッセージを受信した機器は，それが自分へ接続可能な機器からのメッセージであれば，そのメッセージをリクエストリストに追加する．

ロック

Request メッセージを受信した機器は，それがロックされるべき機器からのメッセージであれば，そのメッセージをロックリストに追加する．

リ プ ラ イ

ソケットボタンが押された機器（リプライヤ）は，リ

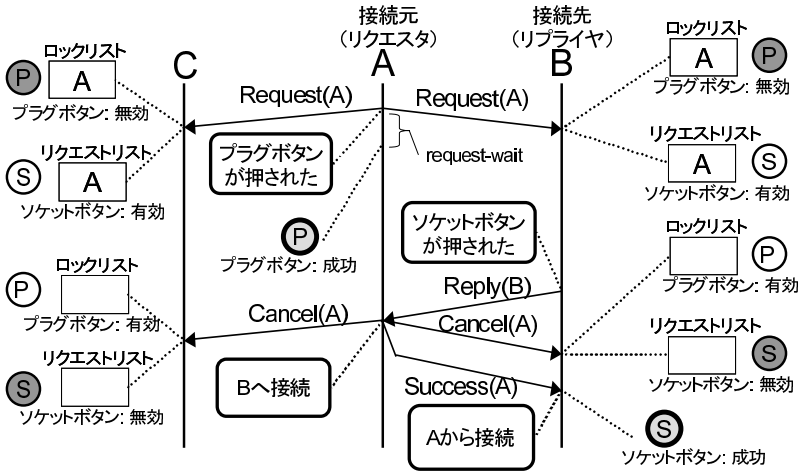
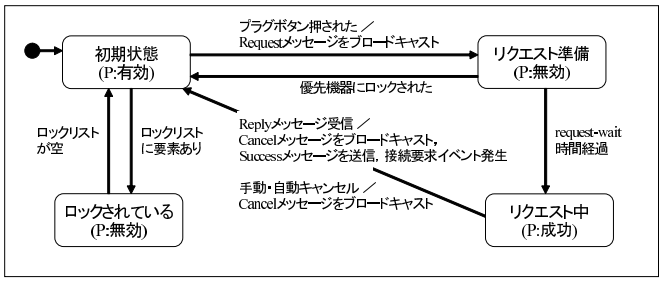
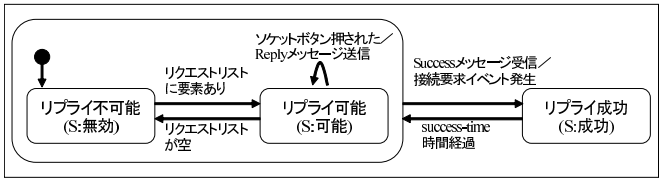


図 6 プロトコルのシーケンス図
Fig. 6 Protocol sequence.



状態遷移1（P:はプラグボタンの状態）



状態遷移2（S:はソケットボタンの状態）

図 7 プロトコルの状態遷移図
Fig. 7 State transition diagram.

クエストリストの先頭要素の送信元に対して，Reply メッセージを送信する（リプライ）．

ロックが正しく行われていれば，リクエストリストの要素は 1 つである．パケットが届かなかった場合や，ネットワーク構成が動的に変化した場合などは，リクエストリストの要素が複数になる可能性がある．要素が複数の場合は，どれにリプライすればよいのかがあいまいとなり，誤接続が発生する可能性がある．

接続操作の完了

リクエスタが Reply メッセージを受信すると，リプライヤと接続することを認識する．リクエストおよびロックをキャンセルするために，Cancel メッセージをブロードキャストする．また，接続することを通知するためにリプライヤへ Success メッセージを送信する．

Cancel メッセージを受信した機器は，リクエストリストおよびロックリストから，送信元機器の要素を削除する．

Reply，Success メッセージには，送信元機器の機器情報が含まれている．機器がこれらのメッセージを受信した際に，この機器情報を含んだ接続要求イベントが，TPS から DMS へ通知されることにより，接続（連携動作）を実現する．

手動キャンセル

ユーザによる手動キャンセルを実現するために，リクエスタは，リクエスト中に再度プラグボタンが押された場合は，Cancel メッセージをブロードキャストして初期状態に戻る．

自動キャンセル

ロックしたまま放置された場合に対応するため，リクエスタは，リクエスト後 cancel-wait 経過した場合は，自動的に Cancel メッセージをブロードキャストして初期状態に戻る．

リクエストの衝突への対応

複数の機器のプラグボタンが同時に押された場合，ロックが行われる前に複数のリクエストが発生する可能性がある（リクエストの衝突）．これは，リクエストリストに複数の要素が存在することになり，誤接続の原因となるため望ましくない．これに対処するため，機器にはアドレスにより優先順位が決めてあり，最も優先する機器からのリクエストのみを有効とする．

リクエストの衝突を検出するために，リクエスタは，リクエスト後 request-wait 時間の経過を待つ（リクエスト準備状態）．その後，はじめてリクエストが有効となりプラグボタンが成功状態となる．この待ち時間中に，より優先する機器からロックされた場合は，そのリクエスタは Cancel メッセージをブロードキャストし，初期状態に戻る．

ボタン版 2 つインタフェースの状態表示

ロックリストが空の場合，新たなリクエストの発生が可能であり，プラグボタンは有効状態となる．要素がある場合，リクエストの発生が禁止されているロック状態であり，プラグボタンは無効状態となる．また，リクエストリストが空の場合，リプライが不可能なため，ソケットボタンは無効状態となる．要素がある場合，リプライが可能なため，ソケットボタンは有効状態となる．ボタンが無効状態の場合は，押すことはできず，押されても無視する．ただし，リクエストリストの要素が複数の場合は，そのままソケットボタンを押すと誤接続が発生する可能性があるため，何らかの表示でユーザに警告することも考えられる．

接続元指定の成功を表示するために，リクエスタは，リクエスト後 request-wait 時間が経過してからキャンセルする（Cancel メッセージをブロードキャストする）までの間，プラグボタンが成功状態となる．また，接続先指定の成功を表示するために，リプライヤは，Success メッセージを受信してから success-time 時間の間，ソケットボタンが成功状態となる．

送信の繰返し

動的な端末の入退出やパケットの損失に対応するため，Request メッセージには，有効時間が定められている．リクエスト中，Request メッセージは何度も送信を繰り返す．また，Cancel メッセージも，パケットの損失に対応するため数回送信する．本フレームワークによるネットワークトラフィックの増加を抑えるため，現在発生中のリクエストの数が増加するにつれ，これらのメッセージの送信間隔も長くする．

セッション ID

A を接続元とした接続操作の直後に，再度連続して A を接続元とした接続操作が行われる場合を考えると，パケットの到着順が変わり，新しい Request(A) が，直前の Cancel(A) によって無効になってしまう可能性が考えられる．そこで，リクエストからキャンセルまでの一連の流れ（セッション）に，毎回異なるセッション ID を割り当て，セッション ID を各メッセージに含める．現在有効なリクエストとセッション ID が異なる Cancel メッセージは無視される．

時間パラメータのデフォルト値

時間パラメータのデフォルト値は，request-wait = 300 ms，cancel-wait = 30,000 ms，success-time = 2,000 ms とする．

2.5 他の接続指示インタフェース

本フレームワークでの最も標準的な接続指示インタフェースは，2.1 節で示したボタン 2 つ版インタフェースである．しかし，他の形態の接続指示インタフェースも考えられる．ここでは，ボタン 1 つ版インタフェースについて述べる．

ボタン 1 つ版インタフェース

ボタンの状態表示を工夫することにより，各機器に対してボタン 1 つのみで本手法を実現することができる．従来の 2 つのボタンの状態を，1 つのボタンで表現するために，ボタンには**表 1** にあげる 5 つの状態を設ける．ユーザは，まず，接続元機器を指示するために，接続元指定状態のボタンを押し，ボタンが接続元指定成功状態になったことを確認する．この際，ボタンが接続元指定状態ではない場合はそうなるまで待つ．その後，接続先機器を指示するために，接続先指定状態のボタンを押し，ボタンが接続先指定成功状態になったことを確認する．

ボタンの状態変化と動作は，基本的には表 1 のルールに従う．ただし，複数のユーザがほぼ同時に操作を開始するような場合への対処が必要である．あるユーザが操作を開始した直後，ボタンが接続元指定状態から接続先指定状態に変わったことに気づかずに，別の

表 1 ボタン 1 つ版インタフェースの動作（より上のルールが優先）
Table 1 Behavior rules of the one-button interface (the higher rule has higher priority).

条件	ボタン状態（とその意味）	押されたときの動作
リクエスト中（自分のリクエストが有効な間）	接続元指定成功（接続元機器の指定が成功）	手動キャンセル
リプライ成功（Success メッセージ受信後一定時間）	接続先指定成功（接続先機器の指定が成功）	
リプライ可能（リクエストリストに要素がある）	接続先指定（接続先機器の指定が可能）	リプライ
ロックされている（ロックリストに要素がある）	無効（押すことはできない）	
デフォルト（初期状態）	接続元指定（接続元機器の指定が可能）	リクエスト

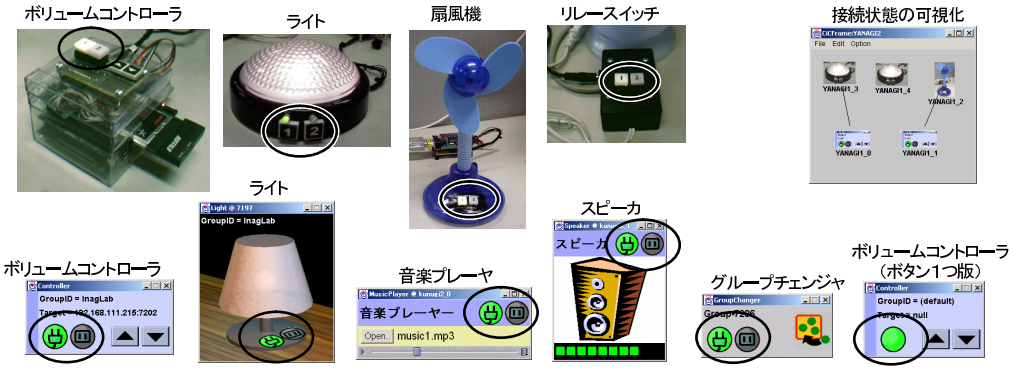


図 8 実装した機器
Fig. 8 Implemented devices.

ユーザが誤ってボタンを押してしまい，誤接続が発生する可能性がある．したがって，接続元指定状態から接続先指定状態へ切り替わる際に，一定時間ボタンを無効状態（操作不可能）とすることにより，状態の切替わりをユーザに認識させる．すなわち，3 行目のルールにおいて，リプライ可能となった後，change-mode-wait（デフォルト値は 1,000 ms）経過するまでの間は無効状態とし，その後に接続先指定状態とする．

ボタン 2 つ版と比較すると，ボタン 1 つ版は，インタフェースの部品コストや実装面積を削減できるという利点がある．一方，ユーザが change-mode-wait の間にボタン状態の切替わりを認識できなかった場合に，誤操作による誤接続が発生する可能性がある．

3. 実　　装

本手法に基づくプロトタイプシステムを Java（Personal Java 1.1.3）を用いて実装した．端末間の通信には無線 LAN（IEEE802.11b）を使用した．本プロトタイプシステムは，ノート PC，PDA（SHARP SL-C750），組み込み用の小型 Linux PC（LAMB-EM-01）などで動作を確認した．

接続指示インタフェースを内蔵した機器として，**図 8** に示すような，複数のソフトウェアによるエミュレータ，およびハードウェアを作成した．図中の円の部分が接続指示インタフェースである．上下ボタンによる操作が可能なボリュームコントローラを，対象物（ライト，扇風機，リレースイッチ）に接続し，明るさ，風速，電源スイッチなどを制御することができる．ボリュームコントローラの対象物どうしを接続することにより，ボリューム値の変化（明るさの変化など）を連動させることができる．音楽プレーヤとスピーカを接続することにより，音楽プレーヤで再生された音楽を，スピーカ（の起動している端末）で再生することができる．また，任意の機器をグループチェンジャへ接続することにより，容易にグループ ID の設定を行うことができる[6]．

Touch-and-Connect のボタンインタフェースの状態は色によって表示した．ボタン 2 つ版のインタフェースでは，緑が有効，赤が成功，黒（消灯）が無効を表す．ボタン 1 つ版のインタフェースでは，緑が接続元，赤が接続先を表し，点滅により成功を表す．

また，DMS に対して拡張を行い，接続状態を可視化できるようにした．図 8 右上の接続状態の可視化ソフトウェアは，ネットワーク上に存在する機器と，そ

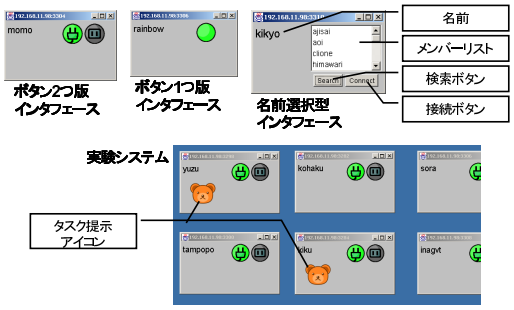


図 9 実験システム
Fig. 9 Experiment system.

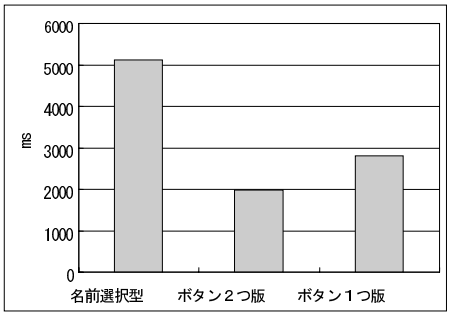


図 10 平均操作時間
Fig. 10 Average operation time.

れらの機器間の接続状態を表示する．

4. 被験者実験による評価

接続指示手法は，ユーザにとって使いやすいこと，また多数のユーザや機器がネットワークに存在する環境を考慮していることが望ましい．我々は，以下のような被験者実験を行い，これらの性質を評価した．

4.1 実験 1：使いやすさの評価

接続先の機器を選択する場合は，たとえばBluetooth Software Suite [7] の接続ウィンドウなどに見られるように，機器の名前を選択するインタフェースが一般的である．一方，提案手法では，機器そのもののボタンを押すという直接的な指定が可能である．本実験では，従来手法である名前選択インタフェースとの比較実験を行い，提案手法の使いやすさを評価した．

提案手法の使いやすさを定量的に評価するために，まずは操作時間の計測を行った．また，実際の使いやすさには，操作時間だけではなく他の要因（操作方法の分かりやすさなど）も影響すると考え，アンケートによる主観的評価も行った．被験者は 6 人の大学生・大学院生である．実験システムを**図 9** に示す．典型的な名前選択型インタフェース，および Touch-and-Connect フレームワークの 2 種類の接続指示インタフェース（ボタン 2 つ版，ボタン 1 つ版）を比較した．

名前選択型インタフェース：これは接続を指示するための最も典型的なインタフェースであり，以下のように操作する．まず，ユーザが接続元機器の検索（Search）ボタンを押すと，接続用ウィンドウを見立てたメンバリストボックスに，ネットワーク上のすべての機器の名前の一覧が表示される．ユーザはこの中から接続先機器の名前を選択し，接続（Connect）ボタンを押す．

メンバリストボックスには一度に 4 つの名前が表示でき，ネットワーク上には 24 台の機器が存在するとした．リストボックスの表示内容はスクロールバーでスクロールでき，名前の一覧は辞書順にソートされている．

ボタン 2 つ版インタフェース：これは，Touch-and-Connect フレームワークの最も標準的な接続指示インタフェースである（2.1 節参照）．

ボタン 1 つ版インタフェース：これは，Touch-and-Connect フレームワークの接続指示インタフェースの 1 つである（2.5 節参照）．

まず，各インタフェースを使った場合に，接続の操作にかかる時間を計測した．図 9 のような実験システムを，タッチパネルディスプレイを装備したノート PC で動作させた．ディスプレイ上には 12 個のソフトウェアエミュレータ機器が存在する．実験システムは，これらの機器のうち 2 つをランダムに選択し，タスク提示アイコンによってユーザに提示する．ユーザのタスクは，ディスプレイ上をペンで操作し，この 2 つの機器間の接続を指示することである．接続の方向（どちらが接続元機器か）は任意である．30 回のタスクを連続して与え，各タスクの操作時間を計測した．操作時間とは，タスクを提示してから，接続の操作を行いタスクを完了するまでの時間である．

各インタフェースを使った場合の平均操作時間を，**図 10** に示す．この結果は，提案手法の操作の簡潔さを示している．Touch-and-Connect フレームワークの 2 つのインタフェースは，典型的な名前選択型インタフェースの約半分の操作時間しか必要としていない．ボタン 1 つ版インタフェースが，ボタン 2 つ版インタフェースよりも平均操作時間が約 0.8 秒長くなっているのは，change-mode-wait（2.5 節参照）の影響だと考えられる．

提案手法の使いやすさを評価するためには，複数のユーザの操作期間が重なり，ロックの解除を待たされるような状況の再現も必要である．そこで，被験者に本システムを他人と並行して使ってもらい，最後に，アンケートにより主観的評価を行った．アンケートの

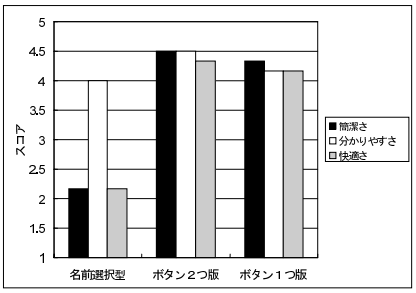


図 11 使いやすさの主観的評価
Fig. 11 Subjective evaluation of usability.

項目は，簡潔さ（指示に必要な手間の少なさ），分かりやすさ（インタフェース使用方法の習得の容易さ），快適さ（インタフェース使用時の心地良さ）に関する 5 段階評価（1–5）である．**図 11** に，各インタフェースのスコアの，全被験者に対する平均値を示す．この結果は，提案手法が，従来の名前選択型インタフェースと比較して，簡潔かつ快適であることを示している．

本手法は，ケーブル接続による機器間の関連付けが行えない無線機器においても，物理的なケーブル接続と同じような感覚で，直接的な指示を行えることを目指したものである．2 秒〜3 秒程度で操作ができ，操作方法も簡潔である提案手法は，物理的なケーブルと同程度の手間で接続が指示でき，インタフェースとして十分に使いやすいといえる．

4.2 実験 2：ロックメカニズムの有効性の評価

Reactive environments[5] や AOSS[8] のボタンインタフェースのような実世界インタフェースでは，複数のユーザの操作が干渉して誤接続が発生する可能性がある．一方，提案手法では，状態表示可能なボタンを用いてユーザの操作を排他制御（ロック）することにより，誤接続を防止できる．本実験では，それらの従来手法を仮定した「ロックなしのボタンインタフェース」との比較実験を行い，提案手法のロックメカニズムが，複数のユーザの独立操作によって発生しうる誤接続を防止できることを評価した．

ロックメカニズムの有効性を示すためには，複数のユーザの操作期間が重なるような状況の再現が必要である．実生活での自然な接続要求に基づいて，長時間の実験を行うことが理想的であるが，被験者の実験時間には限りがあるため，アイコンでタスクを提示することにより，多くの接続要求が発生する状況を意図的に作り出して実験を行った．

被験者は 8 人の大学生・大学院生であり，2 つの 4 人グループに分割して実験を行った．実験システムは実験 1 の場合と基本的に同様である．ただし，タスク提示間隔（タスクが完了してから，次のタスクを提示するまでの間）は，平均 10,000 ms の指数分布に従うランダム時間とした．

この実験システムを，4 人の被験者に同時並行して使用させた．ネットワーク接続された 4 台のノート PC を，それぞれ各被験者が操作する．被験者は，他の被験者と間隔をあけて座り，他の被験者の操作を見ることはできない．

Touch-and-Connect フレームワークのボタン 2 つ版インタフェース（実験 1 と同様），および以下のようなロックなしインタフェースとを比較し，10 分間に発生する誤接続の回数をカウントした．

ロックなしインタフェース：これは，ボタンにより接続を指示する場合の，最も平凡なインタフェースである．各デバイスは，状態表示ができない 1 つのボタンを持つ．最初に押したボタンが接続元機器を意味し，次に押したボタンが接続先機器を意味する．この操作により，この 2 つの機器間が接続される．その後，システムは初期状態に戻り，3 回目に押したボタンは再び接続元機器を意味する．

ロックなしインタフェースの実装は，Touch-and-Connect フレームワークのボタン 1 つ版インタフェースを基にして行った．ただし，ロックを無効（ロックリストがつねに空）とし，ボタンの状態表示をなくし，手動のキャンセルを無効とし，change-mode-wait を 0 とした（2.5 節参照）．

表 2 に，各グループおよび各インタフェースに対して，10 分間に発生した誤接続の回数と，失敗タスクの数を示す．タスクの遂行中（タスクを提示してから，次のタスクが提示されるまで）に，少なくとも 1 回の誤接続が発生した場合に，タスクは失敗したと見なす．2 人の被験者の間に誤接続が発生した場合，その両方のタスクが失敗する．タスク失敗率とは，全タスク数に占める失敗タスク数の割合である．

この結果は，ロックなしインタフェースでは頻繁に誤接続が発生し，提案手法のロックメカニズムにより，これらの誤接続の大部分を防止できたことを示している．

5. 関 連 研 究

Touch-and-Connect フレームワークに関連した研究としては，以下のようなものがあげられる．

5.1 接続先機器の自動決定

STONE[9]，AMIDEN アーキテクチャ[10] では，機器の種類（Type）などに基づき，ネットワーク上から自動的に適切な機器を検索する．また，Follow-me

表 2 10 分間に発生した誤接続の回数と，失敗タスクの数
Table 2 The count of incorrect connections and failure tasks during 10 minutes.

インタフェース	ボタン 2 つ版		ロックなし	
グループ	グループ 1	グループ 2	グループ 1	グループ 2
誤接続の回数	2	3	244	295
失敗タスク数	2	4	70	65
全タスク数	163	168	123	142
タスク失敗率	1.2%	2.4%	56.9%	45.8%

application[4] は，ユーザの位置をセンサにより取得し，自動的にユーザに最も近いディスプレイが作業環境として選択される．しかし，多数の機器が存在する環境では自動決定には限界があり，Touch-and-Connect のように，ユーザが望んでいる機器を直接指示できる手段も必要となるだろう．

5.2 直接操作技法

Reactive environments[5] では，接続したい両機器のボタンを押すことにより直接的に接続を指示できる．しかし，複数のユーザが独立に操作を行う状況を想定しておらず，複数のユーザの操作が干渉して誤接続が発生する可能性がある．

Pick-and-Drop[11] では，端末の画面をペンでタップすることにより，コンピュータ間をまたがるデータの移動を，アドレスなどを入力することなく直接的に行うことができる．コンテキストメニューに GUI のボタンインタフェースを表示することにより，Touch-and-Connect を同様の目的に使うことができる．Pick-and-Drop は，ユーザ識別のためのユニークな ID を持ったペンや管理サーバを必要とするが，Touch-and-Connect はこれらを必要としない．

FUI（Fingerprint User Interface）[12] は，コンピュータ間をまたがるデータの移動を，Pick-and-Drop と同様に直接的に指示することができる（fingermemo）．ユニークな ID を持ったペンの代わりに，指の指紋を ID として用いる．この FUI を，Touch-and-Connect と同様に，機器間の接続指示の目的に使用することもできる．しかし，一般に指紋認識モジュールは高価であり，より安価に実現できる手法が望まれる．

InfoPoint[13] は，ID 情報を記述した 2 次元マトリックスバーコードを各機器に貼り付け，カメラを搭載した携帯端末でこれを撮影することにより，ユーザが指し示している機器を特定する．

gesturePen[14] では，赤外線通信モジュールを内臓したタグを各機器に取り付け，指向性の赤外線通信モジュールを内臓したペンでタグを指し示して通信することにより，機器のアドレスを取得する．

これらの既存のシステムと比較して，提案手法である Touch-and-Connect は以下のような特徴がある．まず，複数のユーザによる独立操作を考慮しているにもかかわらず，ユーザの ID を識別する必要がなく，ユーザは特別なデバイスを持ち運ぶことなくシステムを利用することができる．接続指示インタフェースは，状態表示が可能なボタンのみで実装できるため安価であり，また GUI として実現することもできる．また，既存のホームネットワークシステムは一般に中央サーバにより集中管理されており，あらかじめ準備された環境での運用を前提としている．一方，Touch-and-Connect はサーバを用いずにインプリメントすることができ，モバイル環境や必要に応じて一時的に構築されるアドホックネットワークでの利用も可能である．

また，提案手法はボタンを押す必要があるため，手の届く範囲の機器間の接続に用いられることを想定しているが，別の実世界インタフェースとは共存でき，互いに補間関係にある．たとえば，手元のデバイスは提案手法のボタンによって指示し，遠くのデバイスは他の実世界インタフェース（赤外線リモコンなど）で遠隔指示を行うといった利用形態も可能である．我々は方向センサを搭載した携帯デバイスを用い，ユーザの位置とデバイスの方向から，ユーザが携帯デバイスで指し示している対象物を特定する実世界インタフェース Azim を提案した[15]．Azim のプロトタイプシステムでは，提案手法のボタンインタフェースを，携帯デバイス上で遠隔操作することができる．

5.3 セキュリティ

Resurrecting Duckling[16] は，無線アドホックネットワーク環境における情報機器のためのセキュリティモデルである．機器を新規購入した際にその機器を所有者のものとして Imprinting（刷り込み）し，以後その機器は他人の機器との通信が制限される．

提案手法におけるグループチェンジャによるグループ ID の指定（2.3 節参照）は，Imprinting と類似しており，グループチェンジャに秘密情報を持たせることにより，同様の役割を果たすことができる．

AOSS[8] では，無線 LAN のアクセスポイントへの接続に必要な暗号鍵などの情報を，クライアント端

末とアクセスポイント各々のボタンを押すことにより容易に設定できる．これはグループチェンジャによるImprinting に類似しているが，AOSS は複数のユーザが独立に操作を行う状況を想定しておらず，多くのユーザが同時並行して操作を行うと誤接続の可能性がある．

6. ま と め

接続したい両機器のボタンを押すことにより，機器間の接続を直接的に指示できるフレームワーク Touch-and-Connect を提案した．本手法では，状態を表示可能なボタンを用いてユーザの操作を排他制御し，複数の人間が独立に操作を行う状況においても誤接続を防止する．ブロードキャスト通信を用いた本手法のプロトコルは，管理サーバを必要とせず，動的な端末の入退出も考慮しているため，必要に応じて一時的に構築されるアドホックネットワークでも利用可能である．また，セキュリティと使いやすさの向上のため，グループを作成して機器間の接続可能性を制限できる．本フレームワークのプロトタイプシステムを実装してその実現可能性を示し，また，被験者実験により，インタフェースとしての使いやすさ，および本手法により誤接続が防止できることを示した．

6.1 今後の課題

今後の課題としては以下のようなものがあげられる．

拡張性・柔軟性の向上

現在，本フレームワークでは，2 つの機器間の接続可能性や通信プロトコルをあらかじめ定めておく必要がある．機器間の連携のためのフレームワークは，事前に想定していない機器とも連携が行えるような，拡張性や柔軟性を持っていることが望ましい．このような拡張性のあるフレームワークを実現するためには，プログラムコード移動技術が適している[17]．アドホックネットワーク環境を想定したモバイルエージェントシステム cogma[18] を用いて，機器間連携のためのより拡張性のあるフレームワークを実現することを考えている．

隠れ端末問題

現在，本手法のプロトコルでは，ネットワーク上の全ノードが互いに直接通信可能であると仮定している．しかし，広範囲の無線アドホックネットワークでの利用を考えた場合，隠れ端末問題を考慮する必要がある．現状では，電波到達範囲の境界付近において，リクエストリストの要素が 2 つになってしまい，誤接続を引き起こす可能性がある．この問題を解決するためには，リクエストが行われる領域に対して，その 2 倍以上の大きさの領域がロックされる必要がある．

ProxNet[19] では，無線 LAN の電波強度を端末間の距離を測る尺度として利用している．本手法でも同様に，無線メッセージの電波強度を用いてリクエストとロックの範囲を制限することにより，隠れ端末問題の解決が期待できる．具体的には，2.4 節のプロトコルにおいて，Request メッセージの電波強度が閾値 p_r 以下の場合はリクエスタの Type によらず接続不可能（リクエストされない）とし，また閾値 p_l 以下の場合はロックされなくする．ただし，p_r は，本手法の適用を想定している “手の届く範囲”（5 m 程度）に対応した電波強度とし，p_l は，前者の 2 倍以上の距離に対応した電波強度とする．

謝辞 本研究の一部は平成 16 年度産業技術研究助成事業費助成金（「安全なユビキタス社会を実現する組み込み機器用アドホックネット基盤ソフト」）からの支援を受けて行ったものである．ここに記して謝意を表す．

参 考 文 献

1) Weiser, M.: The computer for the 21st Century, *Scientific American*, Vol.265, No.3, pp.94–104 (1999).
2) Microsoft Corporation: Universal Plug and Play Device Architecture. http://www.upnp.org/resources/
3) ECHONET (Energy Conservation and Homecare Network) Consortium: エコーネット規格書. http://www.echonet.gr.jp/
4) Harter, A., Hopper, A., Steggles, P., Ward, A. and Webster, P.: The Anatomy of Context-Aware Application, *Proc. 5th Annual ACM/IEEE International Conference on Mobile Computing and Networking* (*MOBICOM'99*), pp.59–68 (1999).
5) Cooperstock, J.R., Fels, S.S., Buxton, W. and Smith, K.C.: Reactive environments, *Comm. ACM*, Vol.40, No.9, pp.65–73 (1997).
6) Iwasaki, Y., Kawaguchi, N. and Inagaki, Y.: Touch-and-Connect: A connection request framework for ad-hoc networks and the pervasive computing environment, *1st IEEE Annual Conference on Pervasive Computing and Communications* (*PerCom 2003*), pp.20–29 (2003).
7) Digianswer: Bluetooth Software Suite. http://www.btsws.com/
8) 株式会社バッファロー：AOSS (AirStation One-Touch Secure System). http://buffalo.jp/
9) 澤畠康仁，杉田 馨，南 正輝，森川博之，青山友紀：ネットワークサービスシンセサイザにお

ける機能接続・管理機構の設計と評価，マルチメディア，分散，協調とモバイル（DICOMO 2001）シンポジウム論文集，pp.103–108 (2001).
10) Minoh, M. and Kamae, T.: Networked Appliances and Their Peer to Peer Architecture AMIDEN, *IEEE Communications Magazine*, Vol.39, No.10, pp.80–84 (2001).
11) Rekimoto, J.: Pick-and-Drop: A Direct Manipulation Technique for Multiple Computer Environments, *Proc. 10th Annual ACM Symposium on User Interface Software and Technology* (*UIST'97*), pp.31–39 (1997).
12) Sugiura, A. and Koseki, Y.: A User Interface Using Fingerprint Recognition: Holding Commands and Data Objects on Fingers, *Proc. 11th Annual ACM Symposium on User Interface Software and Technology* (*UIST'98*), pp.71–79 (1998).
13) Kohtake, N., Rekimoto, J. and Anzai, Y.: InfoPoint: A device that provides a uniform user interface to allow appliances to work together over a network, *Personal and Ubiquitous Computing*, Vol.5, No.4, pp.264–274 (2001).
14) Swindells, C., Inkpen, K.M., Dill, J.C. and Tory, M.: Use that there! Pointing to determine device identity, *ACM Symposium on User Interface Software and Technology* (*UIST 2002*), pp.151–160 (2002).
15) Iwasaki, Y., Kawaguchi, N. and Inagaki, Y.: Azim: Direction Based Service using Azimuth Based Position Estimation, *The 24th International Conference on Distributed Computing Systems* (*ICDCS 2004*), pp.700–709 (2004).
16) Stajano, F. and Anderson, R.: The Resurrecting Duckling: Security Issues for Ad-hoc Wireless Networks, *Proc. 3rd AT&T Software Symposium* (1999).
17) Edwards, W.K., Newman, M.W., Sedivy, J. and Izadi, S.: Challenge: recombinant computing and the speakeasy approach, *Proc. 8th Annual International Conference on Mobile Computing and Networking* (*MOBICOM 2002*), pp.279–286 (2002).
18) 河口信夫，稲垣康善：cogma：動的ネットワーク環境における組み込み機器間の連携用ミドルウェア，情報処理学会コンピュータシステム・シンポジウム論文集，pp.1–8 (2001).
http://www.cogma.org/
19) 暦本純一，味八木崇，河野通宗：ProxNet：近接センシングによるワイヤレス環境制御のための直接操作技法，第 11 回インタラクティブシステムとソフトウェアに関するワークショップ（WISS 2003），pp.103–108 (2003).

(平成 16 年 4 月 1 日受付)
(平成 16 年 10 月 4 日採録)

岩崎 陽平（学生会員）
平成 13 年名古屋大学工学部電気電子情報工学科卒業．平成 15 年同大学院工学研究科情報工学専攻修了．現在，同大学院情報科学研究科情報システム学専攻博士後期課程所属．ユビキタスコンピューティング，位置情報サービスの研究に従事．

河口 信夫（正会員）
平成 2 年名古屋大学工学部電気工学科卒業．平成 7 年同大学院情報工学専攻博士課程修了．同大学助手，講師，助教授を経て，平成 14 年より同大学情報連携基盤センター助教授．工学博士．平成 16 年より大学発ベンチャー企業（有）ユビグラフ取締役兼務．モバイルコミュニケーション，マルチモーダルインタフェースの研究に従事．電子情報通信学会，日本音響学会，日本ソフトウェア科学会，人工知能学会，IEEE，ACM 各会員．

稲垣 康善（正会員）
昭和 37 年名古屋大学工学部電子工学科卒業．昭和 42 年同大学院博士課程修了．同大助教授，三重大学教授を経て，昭和 56 年名古屋大学工学部教授，平成 15 年より名古屋大学名誉教授，愛知県立大学情報科学部教授．工学博士．コンピュテーションとコミュニケーションの理論，オートマトン言語理論，ソフトウェア基礎論，自然言語処理に関する研究に従事．電子情報通信学会，日本ソフトウェア科学会，人工知能学会，言語処理学会，IEEE，ACM，EATCS 各会員．