高速なレスポンスを実現した携帯電話対応シンクライアント

関口真良[†] 中島 隆[†] 奥村康行[†]

いつでもどこでもネットワークにつながるユビキタス時代の到来により，ユーザはその場にある端末を利用して自分の環境へアクセスできるようになる．この際，ヘテロジーニアスな環境から一様にアプリケーションを操作できるシステムが必要である．本論文では，サーバベースコンピューティングを利用し，単一なユーザ環境を提供するための，ユビキタスな各端末で動作するシンクライアントについて述べる．特に，ユビキタス端末の中でもスペックの厳しい携帯電話上で動作するシンクライアントを実現することを目標とする．携帯電話の狭帯域・高遅延な通信特性を克服するために，UI コンポーネントベースの軽量プロトコルと，ユーザイベントの予測に基づいた投機的実行の 2 つの手法を提案し，携帯電話上でも高速なレスポンスを実現できることを述べる．また，プロトタイプシステムと，その上で動く 2 つのアプリケーションについても述べる．

A Thin Client for Cellular Phones Which Can Achieve Quick Response

Masayoshi Sekiguchi,[†] Takashi Nakashima[†]
and Yasuyuki Okumura[†]

In the coming ubiquitous age, people can browse their own information space anytime and anywhere. In that age, it is important to provide the unique interface for users whenever and wherever they are. In this paper, thin client system is proposed, which provides unique interface, operates on the several terminals, and is based on Server-Based Computing technology. To achieve comfortable operation even on a cellular phone, two techniques are proposed. One is UI component based lightweight protocol. The other is speculative execution mechanism based on user event prediction. These two proposals can achieve quick response even on a cellular phone. In addition, two applications implemented on the prototyping system are described.

1. はじめに

近年，携帯電話は高機能化・高性能化し，単なる電話としてではなく，いつでもどこでも利用可能なユビキタス端末として認識されつつある．また，PDA・店頭端末・セットトップボックスなどの組み込み端末も生活の中に普及してきている．これらの端末上に同一のアプリケーション環境を提供できれば，ユーザは端末を持ち歩くことなく，その場にある身近な端末を利用して，いつでもどこでも自身の環境へアクセスすることが可能になる．これにより，家庭内の PC でメールを書いていた続きを携帯電話で書いたり，家庭の TV で取得したコンテンツを出先の端末で再生したりといった応用が可能となる．

多くの端末が混在するヘテロジーニアスなクライアント環境において，ユーザがアプリケーションを一様に利用できる環境を提供するためには，アプリケーションの状態や保存されているデータといったユーザのコンテキストを各端末間で共有できる仕組みが必要となる．本論文では，サーバベースコンピューティング（SBC）モデルを適用することでこの問題を解決する．SBC では，ネットワークに接続されたサーバ上でアプリケーションやデータを管理し，ユーザはシンクライアントと呼ばれるネットワークを介して接続されたクライアント端末から，サーバ上のアプリケーションを操作する．シンクライアントは，ユーザからキーボードやマウスの入力情報を受け取り，サーバ上のアプリケーションへ通知するとともに，サーバからアプリケーションの実行結果を受信し，ユーザに表示する．

ヘテロジーニアスなクライアント環境では，各端末の性能や入出力装置の形状に大きな違いがある．これらの違いを吸収し，多くの端末で動作する SBC システムを実現する必要がある．現在，ユビキタスなクライアント端末の中で，最もスペックの厳しい端末の 1 つが携帯電話である．携帯電話の処理能力は飛躍的に

† NTT アクセスサービスシステム研究所
NTT Access Network Service Systems Laboratories

向上しているが，PC の処理能力には遠く及ばない．また，通信速度は第三世代携帯電話でも有線 LAN や無線 LAN に比べてきわめて遅い．これらの問題を解決し，携帯電話上でも動作する SBC システムを実現できれば，他の多くのクライアント端末でも動作することが期待できる．

携帯電話上で動作する SBC システムを実現するうえで大きな課題となるのが，低処理能力・狭帯域・高い通信遅延という携帯電話の特性である．既存の SBC システムの多くは，帯域を節約するために高度な圧縮手法を利用してデータサイズを抑えているが，携帯電話の処理能力では圧縮・伸張にかかる処理時間がボトルネックとなるため，高度な圧縮手法は利用できない．また，通信遅延による影響は避けられない．よって，複雑な処理を行わずに狭帯域・高遅延な通信環境上で SBC システムを実現する必要がある．なお，携帯電話では入出力インタフェースが貧弱という課題もあるが，本論文が解決しようとしている問題点ではないため，本論文内ではこの解決策について言及しない．

現行の携帯電話で利用されている通信方式の 1 つである W-CDMA の帯域幅は，理論値で上り 64 Kbps/下り 384 Kbps である．しかし，実際のスループットはこれより低下する．特に，SBC システムのように細かいデータを頻繁に送受信する場合，スループットは大幅に低下する．しかし，帯域が狭いからといって貧弱なユーザインタフェースしか実現できなければ，ユーザは利用しなくなってしまう．

また，W-CDMA では，非常に大きな往復通信遅延が存在することが知られている．通信遅延そのものをなくすことは不可能なため，通信遅延の影響をユーザに対して隠蔽できる仕組みが必要である．既存の SBC 製品の多くは業務での用途を前提としているため，レスポンスタイムの要件が厳しくない．しかし，一般ユーザをターゲットとする場合は使い勝手が非常に重要となる．レスポンスタイムはユーザがストレスを感じない程度に高速である必要があり，そのために許容される遅延時間は非常に小さい．

携帯電話上の SBC システムを実現するためには，これらの厳しい要件を満たす必要がある．

2. 全体アーキテクチャ

本論文で提案する SBC システムの全体アーキテクチャを**図 1** に示す．

本 SBC システムでは，様々な端末から利用できるように，端末の種類によらず統一されたプロトコルを利用する．そのため，各端末に固有の情報は各端末にインストールされるミドルウェア内に記述し，サーバから送られてきたデータを各端末ごとにカスタマイズして表示する．

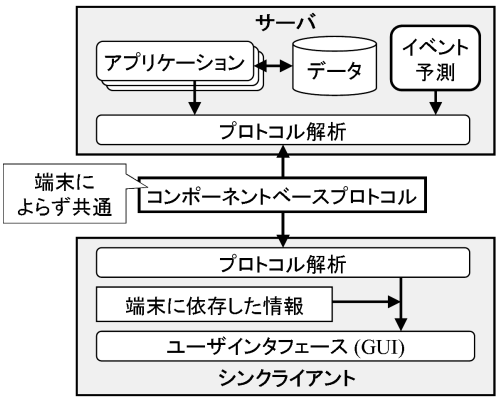

図 1 全体アーキテクチャ
Fig. 1 An architecture of our system.

本 SBC システムでは，携帯電話から PC まで様々な端末で共通のプロトコルを用いるため，最も帯域が狭い携帯電話のスペックでも問題なく動作する一方で，PC などのリッチなクライアント環境でもユーザインタフェースが貧弱にならない表現力を持ったシンクライアントプロトコルが必要である．本論文では，狭帯域な通信環境に対応したシンクライアントプロトコルとしてコンポーネントベースの軽量プロトコルを提案する．

また，SBC システムでは，ユーザイベントが発生してから結果が表示されるまで，サーバ上での実行時間をゼロと仮定しても往復通信遅延時間の間ユーザは待たされることになる．高速な有線 LAN や無線 LAN の環境では通信遅延時間が小さいため問題にならないが，W-CDMA の場合 1,000 ミリ秒程度の往復通信遅延時間が存在するため，レスポンスに問題が生じる．この携帯電話の高い通信遅延の影響を隠蔽しレスポンスを向上させるための，ユーザイベントの予測に基づいた投機的実行手法を提案する．

以降の章では，これら 2 つの手法の詳細について述べるとともに，携帯電話および PDA 上に実装したプロトタイプと，その上で動作するアプリケーションについても述べる．

3. 軽量プロトコル

携帯電話の狭帯域な環境で高速に動作する軽量プロトコルの要件として，

(1) ユーザインタフェースの表現力を損なわないレベルでデータサイズが小さいこと，

(2) 携帯電話から PC までの幅広いすべての端末で動作すること,

があげられる．以降では，これらの要件を満たす軽量プロトコルについて述べる．

3.1 UI コンポーネントベースの軽量プロトコル

市販の SBC 製品の中で最も多く利用されているプロトコルが，画素に基づいた情報を送信するプロトコルである．このプロトコルでは，サーバ側で生成された画面更新情報を画素単位に分割してクライアントに送信する．クライアント側では受信したデータを基に画素を復元し画面に表示する．画素をベースとしたプロトコルでは，通信量が大きいだけでなく，色数や解像度によって通信量が大きく変化するという問題がある．PC 向けの SBC 製品では高度な圧縮手法により，通信量を非常に小さく抑えているが，処理能力の小さい携帯電話では，圧縮・伸張に要する時間が長くなるため適さない．

また，グラフィック API（Application Program Interface）を送信するプロトコルも存在する．このプロトコルでは,「線を引く」「丸を描く」といったグラフィック描画に関する命令をクライアントに送信する．クライアント側では，受信した命令に従って画面上に描画を行う．この方式では，色数や解像度によってデータサイズは変化しないものの，複雑な UI を実現しようとすると，呼ばれる API の数が多くなるため，通信量が大きくなる問題がある．

これらの問題点を解決するために，UI（User Interface）コンポーネントに基づいたコマンドベースのプロトコルを提案する．UI プログラミングを行う場合，API として提供されている「Button」や「TextField」といった UI コンポーネントを利用して GUI を構築する方法が一般的である．UI コンポーネントが画面上にどのように描画されるかは API の提供者側が決めている．プログラマは設定可能ないくつかのプロパティ値を変更することにより，背景色を変更するといった UI コンポーネントのカスタマイズを行うことができる．各 UI コンポーネントがどのように描画されるかは API の内部に記述されている．つまり，既存の SBC システムの場合では，UI コンポーネントの描画情報はサーバ側に保持されており，画素またはグラフィック API の形でシンクライアント側へ転送する必要があり，これがデータサイズを大きくする要因となっている．

提案プロトコルでは，各 UI コンポーネントの描画情報をシンクライアント側であらかじめ保持しておくことで，通信量を削減する．また，プロパティ値のデフォルト値もシンクライアント側で保持しておき，UI コンポーネントの種類とその変更されたプロパティ値のみを，サーバからシンクライアントへ送信することにより，アプリケーションの画面をシンクライアント側で再現することができる．このとき，各 UI コンポーネントのプロパティ値は一般にそのほとんどがデフォルト値から変更されないため，通信量を非常に小さく抑えることができる．また，画面解像度，画素数，色数，UI コンポーネントのグラフィックの複雑さなどにも通信量が左右されない特徴がある．なお，アプリケーションから変更されたプロパティ値を抜き出す作業は，サーバ上のミドルウェアによって行われるため，プログラマがアプリケーションに改変を加える必要はない．また，すべての通信は HTTP（Hyper Text Transfer Protocol）のようなテキストベースのデータではなく，バイナリ化されたコマンド列として送信される．

3.2 サーバからシンクライアントへの通信

サーバからシンクライアントへ送信される項目は，UI コンポーネントのインスタンスを一意に識別できる ID と，UI コンポーネントの種類，そして変更のあったプロパティ値である．シンクライアント側では，受信したコマンド列からコンポーネントの種類と変更のあったプロパティ値を取得し，シンクライアント内に保存されている描画情報の中から，コンポーネントの種類をキーに対応する描画情報を検索し，プロパティ値の変更を反映し，画面に表示する（**図 2**）．

3.3 シンクライアントからサーバへの通信

シンクライアントからサーバへ送信される項目は，発生したユーザイベントの種類，イベントの対象となった UI コンポーネント，そしてイベントパラメータである．サーバ側では受信したコマンド列からこれらのパラメータを復元し，サーバ上で管理しているアプリケーションに対してイベント内容の反映を行う．

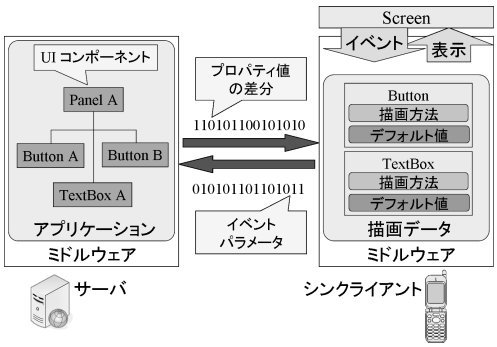

図 2 プロトコル概略図
Fig. 2 An overview of our protocol.

その結果生じたプロパティ値の変更を再びコマンド化し，クライアント端末へ通知する（図 2）．

3.4 不要な通信の除去

通信量を削減するために，シンクライアント側で発生したユーザイベントをすべてサーバへ送信するのではなく，サーバ上のアプリケーションの内部状態が変化しないイベントに関してはサーバへの送信を行わない．日本語変換やカーソルの移動，そして，そのイベントをトリガとするイベント処理が設定されていないすべてのイベントについてイベントパラメータの送信を行わない．

日本語変換については，クライアントがローカルに備える日本語変換機能を利用することでサーバへの送信は行わない．その他のユーザイベントについては，以下の手順でイベント処理の有無を把握し，不要なイベントパラメータの送信を避ける．まず，サーバ上のミドルウェアで，UI コンポーネントに対してその UI コンポーネントに対するイベントをトリガとするイベント処理の有無を表すプロパティ値を設定し，他のプロパティ値と同様にシンクライアントへ送信を行い，シンクライアント内に保持しておく．実際にイベントが発生した際に，シンクライアント内に保持されているイベント処理の有無に関するプロパティ値を参照し，発生したイベントをトリガとするイベント処理の記述がない場合は，該当イベントに関するパラメータは送信しない．これによって無駄な通信が発生することを防ぎ，通信回数を減らすことができる（**図 3**）．

たとえば，Java 言語において実装を行った場合には以下のようになる．Java 言語のイベント処理では，プログラマがあらかじめ java.util.EventListener インタフェースのサブインタフェースを実装し，トラップするイベントに対してそのイベントが発生した際に行う処理の内容を記述しておく必要がある．そのため，

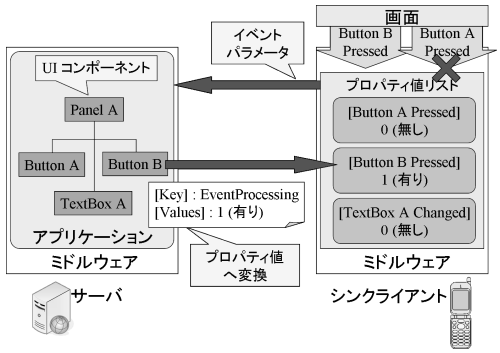

図 3 イベント処理のプロパティ化による通信量の削減
Fig. 3 Reduction of traffic by using property values.

EventListener のサブインタフェースが実装されているかどうかを調べ，プロパティ値としてクライアントに送信しておくことで，イベント発生時にイベントパラメータを送信するか判断することができる．

4. ユーザイベント予測に基づく投機的実行

携帯電話を対象とした SBC システムを実現するうえで，狭帯域とともに大きな問題となるのは通信遅延である．通信遅延の大きさに比例してレスポンスが悪化するため，ユーザビリティを向上させるためには，ユーザにネットワークを意識させない程度のレスポンスを実現することが重要である．

この問題を解決するために，本論文では，ユーザイベント予測に基づく投機的実行手法を提案する．本手法は，まず，ユーザが起こすイベント内容を事前に予測し，予測に基づいてあらかじめアプリケーションを実行するとともに，その実行結果をシンクライアントに送信し，シンクライアント端末内に保存しておく．これにより，実際にユーザがイベントを起こした際，保存されている予測結果を読み出すことで，更新された GUI を表示することができる．このとき，サーバとシンクライアントとの間で通信がまったく発生しないため，通信遅延の大きさにかかわらず通信遅延の影響を隠蔽することができる（**図 4**）．

ユーザが実際に操作を行うまでに予測結果を送信でき，かつその予測が的中した場合は，更新された GUI を表示するまでの時間はクライアント端末内での描画処理時間だけとなり，ほぼゼロとなる．

なお，既存のウェブブラウザには，リンク先の HTML の先読みを行う機能が搭載されているものがあるが，本システムは，静的な HTML より表現力に優れたシンクライアントの U I を対象としているため，HTML を先読みするための技術では同等な機能を実現できない．HTML は文章を読むために設計されているため，リンク先を読み込むための十分な時間があ

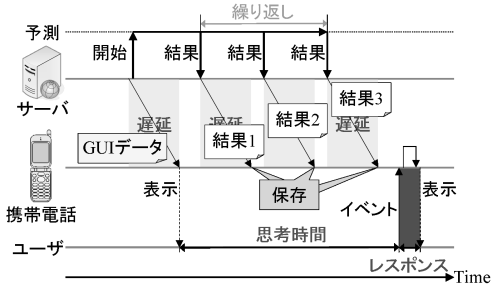

図 4 イベント予測に基づく投機的実行
Fig. 4 Speculative execution based on event prediction.

り，先読みのアンカータグの選択肢は少なく，先読みされるコンテンツが静的である．一方で，本システムは動的に生成される UI で先読みを行うため，ユーザがインタラクティブに操作を行う短い時間の中で，多くの選択肢の中から効率的な予測で絞り込み，投機的実行を行う必要がある．また，先読みの結果表示される新しい UI は静的ではなく動的である．

4.1 予測可能性

予測に基づく投機的実行を実現するためには，ユーザのイベント内容を予測できる必要があるが，これは以下の理由から予測することが可能である．

まず，GUI 上でユーザが起こすことができるイベントの数は限られている．特に，携帯電話のような画面解像度の低い端末の場合は，配置される UI コンポーネントの数が少ないため，生じる可能性のあるユーザイベントの数も少ない．また，イベント処理が記述されているイベントの数はさらに少ないため，予測対象イベントは十分予測可能な数となる．なお，画面解像度の高い端末の場合は，有線 LAN などの通信遅延の小さい回線を使用可能である場合が多いため，このような予測手法を用いる必要がない．

次に，ユーザは表示された GUI を見て即座に何らかの操作を行うわけではなく，画面を認識・判断・操作するために 1 秒〜数秒程度の時間が存在する．Raskin は，ユーザが GUI を認識し実際に行動を起こすまでの心理的準備期間として 1,350 ミリ秒が必要だとしている[1)]．この時間は，サーバ上でユーザのイベントを予測するのに十分な時間である．また，ユーザが操作を行うまでの時間が長ければ長いほど多数のパターンを予測でき，予測の的中確率を高めることができる．

4.2 UI の特徴を利用した予測アルゴリズム

ユーザが起こすイベントを高精度で予測するために UI 独自の指標を利用した予測手法を提案する．以下では，予測手法の詳細について述べる．

イベント予測のために UI から取得できる情報としては，

- イベント種別
- イベントパラメータ
- イベント処理の実装の有無
- UI コンポーネントの配置順序
- 初期フォーカスの位置

がある．これらの情報を以下の手法により数値化する．

(1) イベント種別は，「Button が押された」「TextBox の内容が変更された」「キーが押された」といったイベントの種類である．イベント種類別の発生確率の統計をとり，利用される確率が高いイベント種別に高い数値をつける．設定数値は，一般的な発生確率に基づいて決定したり，ユーザの行動履歴を基に動的に変化させたりすることも可能である．

(2) イベントパラメータは，キーが押されたイベントにおける押されたキーの値といった，各イベント種別ごとに定められたパラメータである．イベントパラメータについては，イベント種別によっては存在しなかったり，TextBox の編集内容のような予測に適さなかったりするパラメータもあるため，予測に適用可能なパラメータのみ数値化を行い利用する．イベントパラメータについても，設定数値は，一般的な発生確率に基づいて決定したり，ユーザの行動履歴を基に動的に変化させたりすることが可能である．

(3) イベント処理の実装の有無は，サーバ側で当該のイベントに対するイベント処理がプログラマによって実装されているかどうかである．イベント処理が実装されていないイベントに関しては，予測の対象外となるので数値として 0 を与える．

(4) UI コンポーネントの配置順序は，画面上に各 UI コンポーネントが配置される順番である．利用頻度が高いと思われる UI コンポーネントを画面の上の方に配置する傾向があるため，上に配置された UI コンポーネントに対して発生するイベントに高い数値を与える．たとえば，ボタンが上から 3 つ並んでいた場合，最も利用頻度が高いと期待されるボタンは最も上に配置されたボタンである．

(5) 初期フォーカスの位置は，プログラマが意図的に設定した初期フォーカスをもつ UI コンポーネントのことである．プログラマが何も指定しない場合，最も上にある UI コンポーネントに自動的にフォーカスが設定されるが，明示的に初期フォーカスを設定した場合は，その UI コンポーネントが最も利用される確率が高いと判断することができる．よって，その UI コンポーネントに対して発生するイベントに高い数値を与える．

以上の手法で，予測対象となる各イベントに対して各要素の数値を計算し，その相乗平均をとり，最も数値の高かったイベントが最も発生する確率が高いイベントと予測する．

4.3 予測に基づく投機的実行手法

下記の手順で投機的実行を行う（**図 5**）．

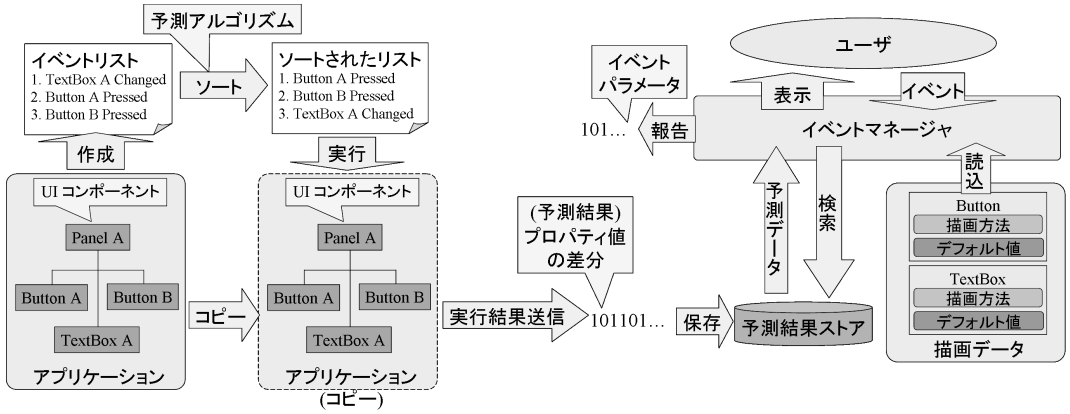

図 5 投機的実行プロセス
Fig. 5 A process of speculative execution.

(1) サーバ上の予測エンジンが，起こりうるすべてのユーザイベントのリストアップを行う．この数は前述のとおり有限で大きな数ではない．
(2) リストアップしたイベントを，前述の予測アルゴリズムにより数値化する．
(3) リストアップしたイベントを (2) で数値化された値が高い順にソートする．
(4) (3) でソートした順に，予測結果に基づいてアプリケーションを実行する（投機的実行）．この際，予測が外れる場合に備えて，実アプリケーションに対して予測結果を直接反映するのではなく，アプリケーションのコピーを作成し，コピーに対して実行を行う．また，外部データベースなど，ユーザインタフェース外へアクセスするものに関しては，予測が外れた場合に整合性がとれなくなるため，投機的な実行を行わない．ただし，データベースにアクセスする場合などでは，データベースアクセスにある程度の時間がかかるため，ユーザインタフェースで高速なレスポンスを実現する必要がない．なお，外部にアクセスするかどうかの判断はミドルウェア上で行うため，ユーザがアプリケーションに改変を加える必要はない．
(5) 実行した結果生成されるプロパティ値の差分をクライアント端末へ送信する．
(6) クライアント側で，サーバ側から送信された予測結果（イベント種別とプロパティ値の差分の組）を受信し，クライアント端末内に保存する．
(7) ユーザが実際に操作を行うまでに (4)〜(6) を繰り返し，できる限り多くの予測を行う．
(8) ユーザが実際に操作を行った際，クライアント端末上のイベントマネージャがイベントを受け取り，イベント種別をキーとして保存されている予測結果を探索する．
(9) 当該のイベント種別に対する予測結果が存在した場合，その予測結果であるプロパティ値の差分を取り出し，クライアント端末内で保持している描画情報に反映し，ユーザに表示する．
(10) サーバ側へ予測が的中したことを表すコマンドとイベントパラメータを送信する．
(11) サーバ側で受信したイベントパラメータをアプリケーション本体に適用する．

なお，予測結果がクライアント端末内に存在しなかった場合は，イベントパラメータをサーバ側に送信し，アプリケーション本体に反映する．そして，実行結果であるプロパティ値の差分をクライアント端末に送信し，そのデータに基づいて画面を描画する．この場合，通信遅延の影響は隠蔽されない．

5. プロトタイプ実装

前述した軽量プロトコルおよびユーザイベントの予測に基づく投機的実行機能を備えた SBC システムを試作し，NTT DoCoMo の携帯電話機上および Java 搭載 PDA である PocketCosmo 上で動作させた．

シンクライアント側，サーバ側の実装はともに Java 言語で行った．携帯電話では，NTT DoCoMo が採用している携帯向け Java 仕様である DoJa Profile 上に実装を行った．PDA では，PocketCosmo に標準搭載されている Personal Java1.1 上に実装を行った．サーバ側は，Servlet 上に実装を行った．

本プロトタイプで実装した UI コンポーネントは，AnchorButton，Button，Image，ImageButton，ImageLabel，Label，ListBox，Panel，TextBox，Ticker である．これらは，DoJa で提供される UI コ

図 6 プロトタイプシステム
Fig. 6 Prototyping.

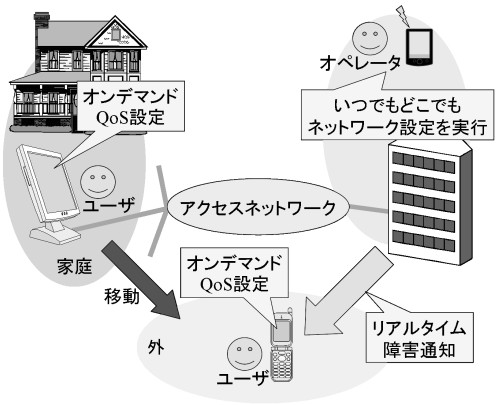

図 7 アクセスネットワークオペレーション
Fig. 7 Access network operation.

図 8 オペレーション画面
Fig. 8 Operation screen.

ンポーネントと同一のクラス名であり，同一のインタフェースを持つ．よって，本プロトタイプシステムでは，DoJa 用のアプリケーションを改変することなくシンクライアントアプリケーションとすることができる．今回，DoJa の UI コンポーネントに合わせて実装したのは，規模が小さいため実装が容易だったためであり，将来的には Java の Standard Edition で提供される UI コンポーネントに合わせて実装することを考えている．なお，このことは Java 以外のアプリケーションでは動作しないことを意味しているのではなく，他のプログラム言語で実装されたアプリケーションを移植する場合は，本システムで提供する UI コンポーネントに合わせてアプリケーションを改変する必要があるということを示している．

本プロトタイプにより，異なる端末環境で利用者の環境を共有できることを確認した（**図 6**）．

また，本プロトタイプ上で 2 種類のアプリケーションを試作した．1 つは「アクセスネットワークオペレーション」への適用，もう 1 つは「情報提供サービス」への適用である．以降では，それぞれのアプリケーションについての詳細を述べる．

5.1 アクセスネットワークオペレーション

通信事業者間の競争激化により，QoS の設定や障害対応といったアクセスネットワークのオペレーション操作をリアルタイムに行うことが求められている．一方で，人件費削減のため，ネットワークの設定を行うオペレータを多数配置することは難しい．そこで，少ないオペレータで迅速なオペレーションを実現するために，オペレータがいつでもどこでもオペレーションを行える環境が必要である．また，オペレータだけでなくユーザ自身が必要なときに自分自身で QoS などの設定を行えるようになれば，さらなる利便性の向上につながる．さらに，リアルタイムに障害通知をユーザに知らせることにより，より高度なカスタマケアを提供できる（**図 7**）．

オペレーションの誤操作は加入者に与える影響が大きいため，端末の誤作動や故障による影響を受けない必要がある．また，設定情報には個人情報が含まれるため，個人情報漏洩防止の観点からもクライアント端末内に個人情報を保存しないことが望ましい．以上のことから，いつでもどこでも誰でもがオペレーション操作を行うためには，本 SBC システムが有効である．

今回，本 SBC システムを利用した帯域制御・遅延保証などの QoS 設定や障害のリアルタイム報告などのアクセスネットワークオペレーションを行うアプリケーションを試作し，その動作を確認した（**図 8**）．

5.2 情報提供サービス

2 つ目のアプリケーションとして，あらゆる人々が，いつでもどこでも簡単に必要な情報を得られる情報提供サービスを試作した．

本アプリケーションは，電気，水道，ガス，電話といったライフライン系の情報や，自治会などの回覧板，

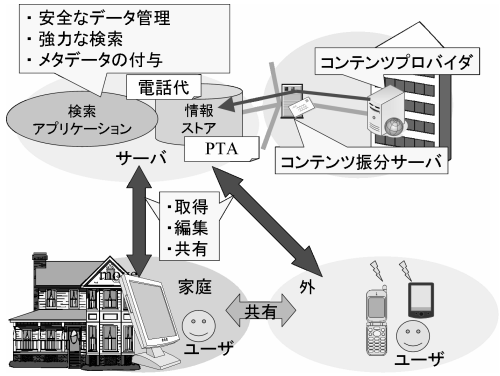

図 9 情報提供サービス
Fig. 9 Information provider service.

図 10 スケジューラ
Fig. 10 Scheduler.

PTA からのお知らせといったコミュニティ情報，さらには，選挙の投票や各種申請など行政に関する情報などを効率的に提供・管理するアプリケーションである．

本 SBC システムを利用することにより，上記の膨大な情報をサーバ上で安全に管理するとともに，サーバの強力な資源を利用して高速な検索が可能となる．また，利用する端末の種類やユーザのシチュエーションにかかわらずつねに同じ情報を共有することができる（**図 9**）．

サーバ上では，膨大な情報を管理する際に，各情報に対して「名称」「場所」「時間」の 3 つのメタデータを設定し，ユーザはこれらの切り口から検索を行うことができる．これにより，ユーザの断片的な記憶から当該の情報を素早く検索することができる．基本画面はスケジューラの形をとっており，情報を受信した日時および情報の対象日（イベント発生日）の 2 つの「時間」から検索を行うことができる（**図 10**）．

6. 評　　価

携帯電話上のプロトタイプで，軽量プロトコルの性能の評価のための通信量の測定と，予測に基づく投機的実行機能の評価のためのレスポンスタイムの測定，および投機的実行時の CPU 負荷の測定を行った．評価に用いたシステムの構成を**図 11** に示す．また，評価に用いたシステムの諸元は**表 1** のとおりである．

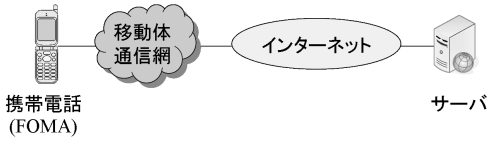

図 11 システム構成
Fig. 11 Evaluation environment.

表 1 評価システムの諸元
Table 1 Specification of evaluation system.

シンクライアント	NTT DoCoMo F900i（FOMA） i アプリ DoJa-3.5 プロファイル
サーバ	Sun Blade 2000 CPU:UltraSPARCIII 1.2 GHz メモリ：1 GB OS:Solaris9

表 2 UI コンポーネント数
Table 2 Number of UI components.

画面	Label	Button	TextBox	ImageLabel
(1)	0	3	0	1
(2)	4	3	0	0
(3)	12	3	2	0
(4)	9	2	3	0
(5)	8	1	3	0

6.1 通　信　量

前述したアクセスネットワークオペレーションのアプリケーションにおいて，QoS の設定を完了するまでの操作を 1 回実行する間に，シンクライアント端末とサーバとの間でやりとりされたデータ量を測定した．なお，アプリケーションの画面遷移は以下のとおりで，各画面に含まれる UI コンポーネントの数は**表 2** のとおりである．

(1) タイトル画面の表示
(2) QoS サービスの選択
(3) QoS の詳細設定（確保する帯域幅など）
(4) 確認画面
(5) 設定完了画面

表 2 の画面遷移において，画面ごとに表示に必要なバイト数を測定した（**表 3**）．軽量プロトコルに関する評価のみを行うため，予測データに関しては除いている．また，本プロトコルの効果を比較するために，商用の携帯電話対応シンクライアントである Bullant [2] との比較を行った．

表 3 のとおり，画面 (1) に関しては差が小さいが，

表 3 通信量（バイト）
Table 3 Traffic (byte).

画面	本システム	Bullant
(1)	2,151	2,396
(2)	259	327
(3)	591	1,055
(4)	591	1,055
(5)	620	1,155

表 4 レスポンスタイム
Table 4 Response time.

投機的実行	レスポンスタイム（ミリ秒/画面）		
(On/Off)	平均	最大	最小
On	206	700	120
Off	1,438	2,010	1,080

これは 1,989 バイトの画像を含むためである．画像に関してはそのまま送信するため圧縮効果が小さい．なお，1 度表示された画像に関してはシンクライアント端末内にキャッシュするため，2 度目以降では送信する必要がない．それ以外の画面については総じて Bullant に比べて非常に少ないデータ量を実現している．

6.2 レスポンスタイム

通信量の測定と同様に，アクセスネットワークオペレーションのアプリケーションにおいて，QoS の設定を完了するまでの操作を 1 回実行する間の，1 画面を表示するのに要したレスポンスタイムの測定を行った．ここでのレスポンスタイムとは，ユーザが実際に操作を起こしてから GUI が表示されるまでの時間である．その結果を**表 4** に示す．なお，画面遷移や各画面の構成については，通信量の測定と同様である．ユーザの操作は，QoS の設定を行う操作を最短で行うものとし，ボタンを間違えたり後戻りしたりするケースは存在しない．表 4 では，予測に基づいた投機的実行を行った場合と，行わなかった場合とを測定し，その効果を比較している．また，投機的実行を行う場合，画面が表示されてからユーザが操作するまでの時間を 1,000 ミリ秒に設定した．なお，計測は 5 回の画面遷移を 1 サイクルとし，10 サイクル，計 50 回の画面遷移を行い，そのレスポンスタイムの最小値，最大値，そして平均値を表示している．計測結果に大きな揺らぎはなかったため試行回数は妥当であると考えられる．

表 4 のとおり，本システムでは，ユーザイベントの予測に基づいた投機的実行を用いることで，ユーザの体感時間を大幅に削減できている．206 ミリ秒というレスポンスタイムは，人間がストレスを感じない十分小さな値である[1)]．

表 5 予測的中率
Table 5 Predict result.

画面	予測的中
(1)	○
(2)	○
(3)	×
(4)	○
(5)	○

表 6 CPU 利用率
Table 6 CPU availability.

同時アクセス数	CPU 利用率 (%)
1	3.4
2	5.8
4	9.6
8	17.0
16	29.3

今回は 1,000 ミリ秒という時間の間で予測を行ったが，すべての画面遷移においてすべてのパターンについての予測結果を送信できた．よって，予測的中率は 100%である．なお，本システムでは効率的に通信するために，複数の予測結果を 1 回の通信にまとめて送信している．今回は，すべての画面遷移において通信回数 1 回ですべての予測結果を送信できた．

なお，予測アルゴリズムの精度を確認するために，最も発生確率が高いと予想したイベントが的中したかどうかについても調べた．その結果を**表 5** に示す．表 5 のとおり，高い確率で予測を的中させることができた．

6.3 CPU 負荷

サーバ上のミドルウェアは，予測の実行回数分だけサーバ上でアプリケーションのコピーと投機的実行を行う．投機的実行時のサーバ側のスケーラビリティ検証のため，投機的実行を行った際のサーバの負荷に関する評価を行った．

計測には vmstat コマンドを利用し，同時アクセス数を変化させながら CPU 利用率を測定した．測定した結果を**表 6** に示す．vmstat は 1 秒おきに測定し，表の数値は 20 回測定した平均値である．なお，20 回の計測中に，数値に大きな変化はなかった．

表 6 から，今回利用したワークステーションの場合で，数十ユーザのスケーラビリティは十分確保できるものと考えられる．なお，今回はすべてのユーザがつねに 1 秒ごとに動作させるという設定で測定したが，実際にユーザが利用する場合は操作ごとのインターバルがより大きくなることが予想されるため，より多く

のユーザを収容できると考えられる．

今回は，アクセスネットワークオペレーションのアプリケーションを用いて評価を行ったが，本アプリケーションは，図 8 に示すとおり，携帯電話の画面をいっぱいに利用しており，配置されるコンポーネントの数も十分である．本アプリケーションで，通信量，レスポンスタイム，CPU 負荷という点で許容できる結果を得られたことで，他の多くのアプリケーションにおいても十分な性能を発揮できると考えられる．

7. ま と め

本論文では，ユーザがいつでもどこでも同じ環境へアクセスするために，様々な端末で利用可能な SBC システムの実現を目標にあげ，多くのユビキタスな端末の中でも処理能力による制約が厳しい携帯電話を対象とした SBC システムを実現するための手法を提案した．携帯電話の低処理能力・狭帯域・高遅延な環境を克服するための軽量プロトコルとイベント予測に基づく投機的実行手法により，携帯電話でもユーザがストレスを感じない SBC システムを実現することができた．今後は，ユーザの行動履歴を利用したより精度の高い予測手法の検討を行うとともに，本システムを利用した実アプリケーションの検討を行っていく予定である．

8. 関 連 研 究

SBC 製品のうち現在主流のものとして，Citrix 社の「MetaFrame」[3] や Sun Microsystems 社の「Sun Ray」[4]，GraphOn 社の「Go-Global」[5]，などがあげられる．しかし，現時点ではこれらうち携帯電話に対応した製品はない．

また，RealVNC 社の「VNC（Virtual Network Computing）」[6] を利用した携帯電話対応 SBC 製品として，日立システムの「μ VNC for BREW」[7] やオープンソースの「J2ME VNC」[8] などがある．さらに，独自のプロトコルを利用した携帯電話対応 SBC 製品として，Gravana 社の「Bullant」[2]，東芝の「ユビキタスビューワ」[9]，といった製品がある．いずれもビジネス用途としては十分なパフォーマンスがあるが，本論文で目標にしている一般ユーザがストレスなく利用できるレスポンスまでは実現できていない．

参 考 文 献

1) Raskin, J.: *The Humane Interface: New Directions for Designing Interactive Systems*, Addison-Wesley Professional, Germany (2000).
2) http://www.bullant.com.au/
3) http://www.citrix.com/
4) http://jp.sun.com/products/desktop/infoappliances/sunray1/
5) http://www.graphon.com/
6) http://www.realvnc.com/index.html
7) http://www.hitachi-system.co.jp/mVNC/
8) http://j2mevnc.sourceforge.net/
9) http://www.toshiba.co.jp/about/press/2005_01/j1801/shiryo.pdf

(平成 17 年 6 月 9 日受付)
(平成 18 年 1 月 6 日採録)

関口 真良（正会員）
昭和 52 年生．平成 14 年早稲田大学大学院理工学研究科情報科学専攻修士課程修了．同年日本電信電話株式会社入社．アクセスネットワークサービスに関する研究開発に従事．

中島 隆
昭和 33 年生．昭和 58 年大阪大学大学院工学研究科通信工学専攻修士課程修了．同年日本電信電話公社（現，日本電信電話株式会社）入社．光ファイバ中の非線形効果，ATM 伝送装置，オペレーションサポートシステムの研究に従事．現在アクセスネットワークに関する研究開発に従事．電子情報通信学会，IEEE 各会員．

奥村 康行
昭和 31 年生．昭和 56 年早稲田大学大学院電気工学専攻修士課程修了．同年日本電信電話公社（現，日本電信電話株式会社）入社．主に，ISDN ユーザ・網インタフェース，HDTV 伝送システム，ATM アクセスシステム，ブロードバンドアクセスの企画立案ならびに研究開発に従事．現在，NTT アクセスサービスシステム研究所光アクセスシステムプロジェクトマネージャー．電子情報通信学会会員．