QuickBoard：任意のアプリケーション画面をリアルタイム配信するWebシステム

市村 哲[†] 中村亮太[†] 伊藤雅仁[†]
宇田隆哉[†] 田胡和哉[†] 松下温[†]

現在，Webシステムが情報システムの主流となっている．しかしながらWebシステムは，HTTPの特性から，情報をリアルタイムに配信するようなサービスに向かないという性質を有している．著者らは，Webシステムの利点を活かしつつリアルタイムに情報を配信できるシステムを構築することが重要と考え，任意のアプリケーション画面を，PC，PDA等のWebブラウザにリアルタイム配信できるQuickBoardシステムを開発した．本開発においては，Webベースのリアルタイム情報共有を必要としている教育の現場に対し，構築したシステムを実際に投入して評価を行うという方法を実践した．そして，その評価から得られたユーザフィードバックを反映させてシステムをリデザインし，システムを再構築した．また，システムの実用性向上を狙い，ネットワーク帯域の使用を抑えるための工夫を施した．本論文では，実際の教育現場の要求を観察した結果について報告し，この観察結果に基づいて作成した第1プロトタイプについて述べる．さらに，このシステムの実利用実験から得られたフィードバックを反映させた，第2プロトタイプについて議論する．

QuickBoard: A Web System for Realtime Delivery of Computer Screen

Satoshi Ichimura,[†] Ryota Nakamura,[†] Masahito Ito,[†]
Ryuya Uda,[†] Kazuya Tago[†] and Yutaka Matsushita[†]

Most of the information systems for business use have been developed recently as web-based applications. Unfortunately the web technology has been considered not to be suitable for the basis of any real-time groupware due to the lack of real-time communication capabilities of HTTP. It, however, would be obviously meaningful to develop a real-time groupware which takes advantage of the web-server-side technology. We developed QuickBoard web server system which allows users to deliver presentation slides of any applications to multiple web browsers in real-time. In the development process of the system, we took the approach of user-centered iterative design. We first studied the existing problems by observing real educational scenes, and developed prototype-1 to solve these problems. We then re-designed the system and developed prototype-2 based on feedback from the users of prototype-1. We were also interested in how to reduce network traffic on web-based real-time groupware, and implemented several methods for this purpose. This paper describes the implementations and evaluations of both prototypes.

1. はじめに

Webシステムが現在の情報システムの主流となっている．クライアントサーバシステムが主流の時代には，いかにクライアントソフトをユーザに配布するかがつねに大きな問題となっていた．しかしWebシステムにおいては，サービスを利用するユーザの環境にはWebブラウザがあるだけでよく，特定のクライアントソフトを必要としない．さらに，システムをバージョンアップする場合，クライアントサーバシステムであれば，修正したクライアントソフトが，ユーザの使用している様々なOSやミドルウェアの上で正常動作するかどうかを漏れなく点検し，かつ，OSごとに異なる実行モジュールを作成して全ユーザにいっせいにインストールさせる必要があったが，Webシステムであれば，サーバ側のプログラムを変更するだけで済む．以上のような利点を背景に，Webサーバサイドコンピューティングが，現在のネットワークサービスのトレンドとなっている．

† 東京工科大学
Tokyo University of Technology

以上のように利点の多い Web システムであるが，HTTP の特性上，情報をリアルタイムにユーザに配信するようなサービスに向かないという性質がある．たとえば，Web サーバから複数の Web ブラウザに向けて情報をプッシュ配信することができず，このことがリアルタイム型のシステムを設計する際の大きな制約となっている．しかしながら，情報をリアルタイムに配信するという機能は，同期型グループウェアを設計するうえで不可欠な機能である．このことから，Web システムの長所を活かしつつ，リアルタイム型グループウェアを構築することは，実用的なグループウェアを開発するうえで非常に意義があることと思われる．

以上のような現状において，著者らは，Web ベースのリアルタイム情報共有システムの開発を試み，QuickBoard システムを構築した．QuickBoard は，任意のアプリケーション画面を，PC や PDA 等の Web ブラウザにリアルタイム送信する機能を有している．ユーザ環境には Web ブラウザのみが必要で，特殊なネットワーク環境や特別なクライアントソフトが必要ないことが特徴である．また，リアルタイム配信型 Web システムにおけるネットワークトラフィック削減方法の開発に注力し，「必要なタイミングで，必要な部分だけをネットワーク転送する」ための機能を提供した．

本論文では，まず，著者らが行った要求分析の結果を報告し，この要求分析に基づいて作成した第 1 プロトタイプについて述べる．さらに，構築したシステムの実運用から得られたフィードバックを反映させた，第 2 プロトタイプについて述べ，その評価について議論する．

2. 要求分析

2.1 現状把握

システムの実用性を高めるため，著者らは，実ユーザが存在する環境を対象として要求分析を行った．具体的には，著者らが所属する大学の「計算機室」と呼ばれる教室で行われる「UNIX システムプログラミング演習」を今回の実践的実験の場として選んだ．後述するとおりこの授業においては，教室および使用機器の物理的な制約による問題点が顕在化しており，解決すべき課題がすでに明確に存在していたからである．

この授業の受講学生数は 100 名弱であり，全員がいっせいに同じ教室で授業を受ける．教室には，FreeBSD + XFree86 がインストールされた PC が 100 台余り配置されており，学生はこの PC を用いて UNIX プログラミングを演習形式で学習する内容となっている．講師のプレゼンテーションの道具は，教室の端に設けられた黒板と書画カメラである．書画カメラの映像は，教室の天井から吊るされた大型テレビ数台に表示され，学生は，自分に近いテレビを選択して見るようになっている（**図 1** 参照）．黒板に書いた文字は教室の端からは見ることが難しいため，実質的には書画カメラのみが用いられている．

図 1 計算機室の風景
Fig. 1 A scene at the computer room.

授業観察の結果，講師が書画カメラを利用するのは，あらかじめ用意したサンプルプログラムを，部分的に拡大して表示する場合が一番多いことが分かった．このとき，学生は同じサンプルプログラムを各自の PC にダウンロードしており，自分の PC 画面上でも見ることができるが，講師の説明を見るためには，自分の PC 画面から視線をはずし，大きく顔を上に向けてテレビを見なければならない．このため，しばしば自分の PC 画面と今説明されている箇所との対応関係を見失ってしまうことが問題となっていた．

さらに，テレビモニタの画面サイズ，および，書画カメラの撮影解像度の制約から，テレビモニタに一度に表示できる情報量はきわめて少なく，多くても 10 行程度であることが分かった．プログラムソースコードレビューでは，前後のソースコードとの関係について説明することがきわめて重要であり，10 行程度しか表示できない状況では説明が困難であることは明らかである．一度に表示できる情報量が少ないという問題は，PC プロジェクタを用いて教室の前方に大きく投影するような場合にも共通して発生する問題ではあるが，天井から吊るされたテレビモニタを用いた場合はこの問題が顕著となった．

2.2 必要要件

以上に述べた，学習環境に関わる物理的制約事項，および，授業観察から明らかとなった要求事項に基づき，本システムの設計にあたっては以下の機能の実現を目標とした．

1. 講師が使うアプリケーションを限定しない．
2. 受講者は Web ブラウザ以外必要としない．
3. 同時に 100 名以上が接続できる．
4. 受講者の PC 画面に講師の説明画面を表示する．

1 に関し，計算機実習のような授業では，特に，講師の使用するツールを限定してしまうことは危険であると考えた．事実，同じ UNIX 上のプログラミングを教える際にでも，ある講師は Emacs を用い，ある講師は vi を使うことが観察された．加えて，講義には PowerPoint や Web ブラウザが用いられるために，これらのアプリケーションも利用できることが必須である．このことから，1 は非常にクリティカルな要件といえる．

2 に関し，受講者の環境においては，OS やデバイスを特定せず，かつ，Web ブラウザだけで動作するように構成する必要があった．3 に関しても，前述したとおり，約 100 名の受講学生を支援対象と設定しているため必須要件である．

4 に関し，著者らは，同期遠隔型グループウェアとしても利用できるシステムを開発することを指向しており，「テレビモニタを PC プロジェクタに置き換える」というような同期対面でしか使えない解決策を選択することはしないようにした．また前述したように，PC プロジェクタを用いた場合にでも一度に表示できる情報量はきわめて少ないため，受講者の PC 画面に講師の説明画面を並べて表示するのが適当と考えた．

なお，同期遠隔型の場合には，音声や映像を別途受講者に伝達する手段が必要となるが，たとえば，一般電話，PHS 電話，ISDN テレビ電話等で音声または講師の顔映像を 2 つの遠隔教室間で共有し，各学生は各自 Web ブラウザで，QuickBoard サービスにアクセスできるようにする等の形態が考えられる．

3. 従来技術の問題点

近年，インターネットを介して通信が可能な，同期遠隔型のリアルタイム画面共有ツール[1] が増えている．これらのツールを用いれば，任意のアプリケーション画面を遠隔に配信することが一部可能である．この類のソフトウェアとして有名なソフトウェアとしては，Netmeeting[2] や，Windows XP に標準搭載されたリモートアシスタント[2] という遠隔デスクトップ共有ツールがある．しかしながら，これらのソフトウェアは使用できる OS が Windows のみに限定されているうえに，専用クライアントソフトを利用する必要があるため，今回の用途には用いることができない．また，企業や学校に設置されたファイアウォールは，セキュリティ確保上 HTTP 以外の通過を許さない設定がなされていることが多く，現状，広域においてこれらのシステムを実利用するには問題が多い．

ストリーミングビデオ技術を応用して画面共有を実現するツールも存在している．Windows Media Screen と Windows Media Encoder とを組み合わせて利用[3] すれば，1 方向にデスクトップ画面をネットワーク配信できる．しかし，受講者の全 Web ブラウザに ActiveX を導入する必要があるという問題がある．ActiveX は，実質的には Windows 実行ファイルそのものである．このため ActiveX を，FreeBSD のような，Windows 以外のプラットフォーム上で動かすことは不可能であり，著者らの目的のために用いることはできない．

複数の OS に対応し Java アプレットとしても動作する画面共有ソフトウェアとしては VNC[4] がある．Java アプレット版のクライアントを用いれば，特定のクライアントアプリケーションをユーザ環境にインストールしておく必要がない．ただし，前述の Netmeeting やリモートアシスタントと同様，基本的に 1 対 1 通信のためのツールであることから，同時接続数の上限がきわめて小さく，今回対象とする使用環境においては用いることができない．また，通信には HTTP 以外の専用プロトコルが使用されているという問題がある．

このほかに，任意のアプリケーション画面を遠隔地に配信する手段としては，PC の VGA 出力をデジタル映像として遠隔地にネットワーク転送する方法が考えられる．たとえば，MPEG2-TS over IP 技術[5] を用いれば，高画質な映像を IP ネットワーク経由で転送できる．しかしながら，これらはたえず画像が書き換わる映像の圧縮を対象とした技術であるために，プレゼンテーションスライドの転送には適さない部分がある．たとえば，MPEG2-TS over IP 装置を用いて，一般的なプレゼンテーションスライドの文字が読める程度の画質の映像を伝送する場合，受講者 1 名あたり 4–6 Mbps のネットワーク帯域が必要となる．仮に百名に同時配信したとすると，400 Mbps から 600 Mbps の帯域を占有する必要がある．また，動画映像を受信するためには，現状では専用のアプリケーション，または，OS に依存した専用のプラグインをユーザの環境に導入する必要があり，今回対象とする使用環境においては利用できない．

4. 第 1 プロトタイプ

4.1 基本機能

前述したシステムの必要要件に基づき，著者らは，リアルタイム型 Web システム「QuickBoard」を開発した．QuickBoard が提供する基本機能は以下のとおりである．

1. 講師が用いる PC のデスクトップ画面のスナップショットイメージをネットワーク転送する．
2. 転送する画像領域を特定し，アクティブな領域のみをネットワーク転送する．
3. 画像更新が必要な時点を特定し，必要なタイミングでのみ画面イメージをネットワーク転送する．
4. サーバと Web ブラウザは HTTP のみで通信する．

1 から 4 の機能は，講師が用いるノート PC 上で動作する「QuickBoard Capture」アプリケーションと，Web サーバ上で動作する「QuickBoard Servlet」とが協調動作することによって実現されている．QuickBoard Capture は Windows PC 上で動作するアプリケーションであり，QuickBoard Servlet は Windows，Linux 等の Web サーバで動作する Java サーブレットである（本実装には Apache Tomcat サーバを用いた）．QuickBoard Servlet が稼動する Web サーバは著者らの研究室内に設置された．研究室はファイアウォールにより防御されており，一般教室や計算機室からは VPN（Virtual Private Network）を用いないとアクセスできないようになっている．以下，**図 2** のシステム全体構成図を参照しながら，それぞれのコンポーネントについて説明する．

4.1.1 QuickBoard Capture

QuickBoard Capture は，講師が用いているノート PC のデスクトップ画面のスナップショットをとり，イメージファイルとしてファイルに書き出すアプリケーションである．講師が送出したい画面範囲を指定すると，その画面範囲のイメージがファイルとして書き出される．作成されたイメージファイルは，Windows のネットワークファイル共有の機能によって，即座に Web サーバに配置される．

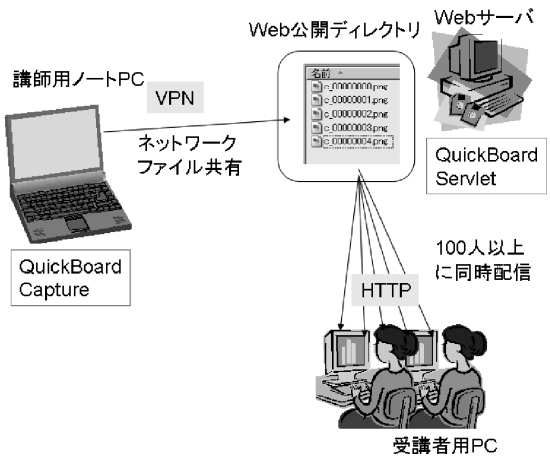

図 2 システム全体構成図
Fig. 2 The system overview.

講師が画面範囲を指定する手段としては，キーボードショートカットで指定する方法と，マウスクリックで指定する方法とを提供した．キーボードショートカットで指定する場合は，あらかじめ設定したショートカットキーが押された場合に，そのときマウスポインタが置かれているウィンドウを特定し，そのウィンドウ領域のスナップショットイメージをファイル保存するようにした．一方，マウスクリックで指定する場合は，ウィンドウをクリック指定する方法と，任意矩形領域をラバーバンド指定する方法の両方を提供した．

本プロトタイプの構成において，イメージファイルが作成されるディレクトリは，Web サーバの公開ディレクトリである．Web サーバ上の特定の公開ディレクトリが Windows のネットワークファイル共有によって講師の PC から書き込み可能となっている（FTP による方法または HTTP のファイルアップロードによる方法で代替することも可能）．ただし講師の PC が Web サーバのファイルシステムに書き込みをする場合は，VPN で Web サーバにアクセスする．これによって，講師以外のユーザは Web サーバのファイルシステムに書き込みできなくなっている．

なお，イメージファイルを Web ブラウザに送信する場合，イメージファイルサイズがほぼそのままネットワーク転送量に比例するため，圧縮率の高いイメージファイルフォーマットを用いることが必要となる．そこで，QuickBoard Capture が作成する画像ファイル形式としては，ネットスケープや Internet Explorer 等の Web ブラウザに表示できる PNG（Portable Network Graphics）画像形式[6] を用いることとした．PNG 圧縮は，画像の質を低下させない可逆圧縮でありながら，ZIP 圧縮技術を用いているため，PC 画面のスナップショットのような画像データであれば，JPEG 圧縮よりも小さなファイルサイズで保存できるという特徴がある．

図 3 は第 1 プロトタイプの実装画面である．講師がノート PC で説明を行い，受講者が PDA の Web ブラウザでその説明を見ている例である．QuickBoard はきわめて標準的な方法のみで Web ブラウザに情報を配信し，かつ，専用プラグイン等を必要としないた

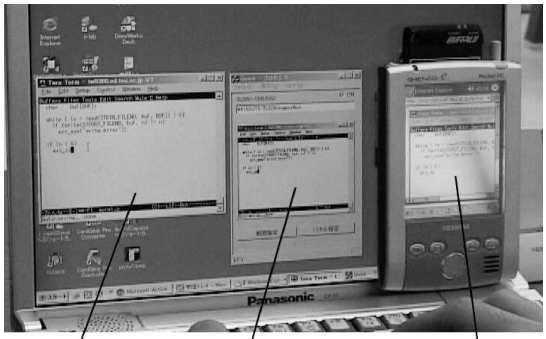

図 3 第 1 プロトタイプ
Fig. 3 Prototype-1.

め，PDA に搭載されている簡易な Web ブラウザにもリアルタイムな情報配信が可能である[7]．

4.1.2 QuickBoard Servlet

QuickBoard Servlet は，イメージデータをリアルタイムにユーザに送信する Web サーバアプリケーションであり，Java サーブレットとして実装されている．このサーブレットは，前記 QuickBoard Capture を用いて指定したデスクトップ画面領域を，ユーザの Web ブラウザにリアルタイムで配信する機能を備える．

ただし実際は，HTTP プロトコルがプッシュ型配信をサポートしていないために，QuickBoard Servlet が能動的に複数の Web ブラウザにデータを送りつけることはできない．そこで QuickBoard では，画像更新があったかどうかを QuickBoard Servlet に定期的（3 ないし 5 秒間隔程度）に問い合わせる小さな JavaScript プロシジャをユーザの Web ブラウザにダウンロードさせ，そのプロシジャによって Web ブラウザにポーリングさせるようになっている．

具体的な処理の流れは次のとおりである．Web ブラウザにダウンロードした JavaScript が，画像更新があったかどうかを Web サーバに問い合わせると，当該 Web サーバ上の QuickBoard Servlet は，QuickBoard Capture が作成したイメージファイルが更新されたかどうかを，ファイル更新日時によってチェックする．ファイルが更新されたことを検出すると，Web ブラウザにこの更新されたイメージファイルを読み込むための小さな JavaScript プロシジャをダウンロードさせ，Web ブラウザがこのプロシジャを実行することで，更新された画面イメージ（PNG 画像ファイル）が Web ブラウザにダウンロードされて表示される．

すなわち，定期的にネットワークを流れる転送データは，画像データと比較してきわめて小さいデータ量の JavaScript プロシジャのみであり，実際の画像データがネットワークを流れるのは画像が更新されたタイミングに限られている．ネットワークパケットをモニタリングするツールを用いてトラフィック調査した結果，画像更新があったかどうかを Web サーバに問い合わせる前記 JavaScript プロシジャが発する HTTP リクエストは約 480 バイト（すべて HTTP ヘッダ）であった．また，画像更新がなかったことを Web ブラウザに返答するレスポンスは約 350 バイト（HTTP ヘッダ約 50 バイト ＋ JavaScript 約 300 バイト）であった．なお，標準的な 640 × 480 ピクセル程度の講義スライド画像は，PNG 画像形式で 50 KB 程度であるため（無地部分が大部分を占める場合は 5 KB 程度に収まる場合もある），画像更新がなかった場合には，PNG 画像を毎回ダウンロードする方式と比較して約 140 分の 1（350 Byte/50 KByte）のトラフィックで済むことになる．

以上のような条件で，1 台の PC あたり 40 前後の Web ブラウザプロセスを動作させ（Web ブラウザのローカルキャッシュ機能はオフに設定），5 台の PC を用いて QuickBoard Servlet に同時アクセスする実験を行った．用いた PC はすべて Pentium4 2.4 GHz マシンである．そして，合計 200 の Web ブラウザプロセスが，同一 QuickBoard Servlet から更新画像を取得して正常に表示できることを確認した．

4.1.3 設計方針

第 1 プロトタイプで提供した機能の設計方針について述べる．まず著者らは，「サーバサイドに特殊なソフトウェアを用いても，クライアントサイドには汎用 Web ブラウザ以外のソフトウェアを必要としない」という方針でシステムを設計した．これは，サーバシステムは 1 つ用意すれば済むが，多数存在するクライアントに専用ソフトウェアを導入するとなると手間とコストが非常にかかるという理由によるところが大きい．QuickBoard では，サービス提供者である講師（1 名）には「Windows PC を利用しなければならない」または「QuickBoard Capture をインストールしなければならない」という制約が課せられるが，受講者（約 100 名）にはそのような制約がまったく課せられないようになっている．なお，前記演習授業では，講師は Windows の TELNET ソフトを用いてリモートの UNIX ホストにログインして Emacs や vi を利用している．この意味では，TTY ベースの UNIX アプリケーションも利用できているわけであるが，Windows で動く X 端末エミュレータソフトを用いれば，ほぼすべての UNIX アプリケーションが講師端末で利用

可能となる．

また，通信プロトコルについても，「サーバシステム内の通信には特殊なプロトコルを用いても，クライアントとの通信には HTTP 以外のプロトコルを必要としない」という方針で設計した．講師用端末と Web サーバとは SMB プロトコルや FTP プロトコルで通信するが，Web サーバと Web ブラウザとは HTTP のみで通信するようになっている．現状では，企業内からインターネットにアクセスする場合に，HTTP による接続しか許可されていないことは珍しくない．このようなユーザ向けのインターネットサービスを提供する場合，HTTP 以外のプロトコルを用いないことは重要である．

次に，システムの性能要件について議論する．本システムでは，各 Web ブラウザは，画像更新があったかどうかをあらかじめ定められたインターバル（たとえば 3 秒間隔）で Web サーバに問合せ（ポーリング）を行い，画像更新があった場合には次のポーリングタイミングで更新画像を Web サーバから取得するようになっている．この仕組み上，同一インターバル内に，講師が 2 回以上続けて画面キャプチャ操作を行った場合には，Web ブラウザに配信されない画面が存在する可能性がある．しかしこの状況が発生することを防ぐために画像更新チェック間隔をきわめて短くする必要はないと考えている．なぜなら，少なくとも次のインターバルのタイミングには確実に全員の Web ブラウザに同じ内容が表示され，通常の使用においては問題が生じないからである．見せたい画面であれば，講師は，通常，数十秒以上同一スライドを継続的して表示するため，3 秒程度の遅れは，講義資料配信という目的に与える影響はほとんどないと考えられる．

また著者らは，画面キャプチャしてから何秒以内にその画面が Web ブラウザに配信される必要があるかという性能要件について実験を行った．ソースコードの重要箇所にマーキングしながら学生に解説するという形態の講義を配信し，このとき Web ブラウザの画像更新チェックタイミングを，3 秒，5 秒，7 秒，9 秒と変化させた．学生に対するアンケート調査の結果，画像更新インターバルが 5 秒以下の場合は，ほぼ全員が，講師が話している内容と表示画面とが一致していたと感じたことが分かった．これに関し，学生から「講師が説明対象箇所について話し始めたその 1 文が終了する前に，説明箇所が表示されると，話しと画面とが一致していると感じられる」という意見を得た．そこで授業を観察した結果，講師が説明対象箇所について説明する文は，確かに平均 5 秒長程度であった．

4.2 評価と考察

4.2.1 ヒアリング調査とユーザ観察

QuickBoard の第 1 プロトタイプを，前述した「UNIX システムプログラミング演習」の授業で実際に利用する実験を行った．この実験実施後に，従来の書画カメラを利用した場合との比較について講師にヒアリングを行ったところ以下のような回答を得た．

1. ソースコードレビュー画面を参照しながら演習を進める学生が大幅に増えた．
2. 一度に多くの情報を学生に提示することができ，ソースコードレビューがしやすくなった．
3. 実際のソースエディティング過程や，プログラム実行結果を表示でき，説明内容が豊富になった．

以上の回答は，前述した授業観察において抽出された問題点に関係する事項であり，既存の問題点に対策がなされたことに対しておおむね好評であったと判断できる．特に，1 および 2 に関しては，現在主流である PC プロジェクタを利用した環境と比較しても有利な点であると考えられる．3 に関しては，PC プロジェクタを用いれば同期対面環境では達成可能であるが，QuickBoard は遠隔での利用が可能という違いを有している．

次にユーザ観察結果について記す．個々の学生がどのように QuickBoard 画面を扱ったかについて観察した．まず，学生個々にウィンドウ配置の仕方にかなり差があることが観察できた．1 つの作業ウィンドウ（Emacs 画面）と 1 つの Web ブラウザ（QuickBoard 画面）以外のウィンドウをすべて閉じている学生，Web ブラウザウィンドウを複数オープンし他の Web ページも表示している学生等様々であった．ただし，ほとんどの学生が，他のウィンドウの下に隠れないように QuickBoard 画面を配置していることも観察できた．講師からの伝達事項が表示される重要なウィンドウであるという意識が学生側にあるためと思われる．

4.2.2 評価実験

さらに上記 1 と 2 に関し，QuickBoard の主要な特徴を評価するための評価実験を実施した．1 つ目の実験 P1 は，PC 画面から視線を外すことなく講師説明画面を見られるという特徴を評価する実験であり，2 つ目の実験 P2 は，一度に多くの講師説明画面を表示できるという特徴を評価する実験である．いずれの実験も現役大学生 6 名 + 教官 1 名の計 7 名を被験者とした．

実験 P1 において，被験者らには，手元の PC 画面に表示された C 言語のソースコード 10 行と，天井から吊るされたテレビモニタに表示された類似ソースコー

表 1 評価実験 P1 および P2 の結果
Table 1 The result of the experiment P1 and P2.

実験	制限時間	被験者数	表示装置	制限時間内未完人数	差異部分 1 箇所発見に要した平均時間
P1	2 分	7 人	テレビモニタ	3 人	26.4s
			QuickBoard	0 人	13.5s
P2	3 分	7 人	テレビモニタ	0 人	36.0s
			QuickBoard	0 人	20.4s

ド 10 行とを比較してその差異部分を見つけ出すという作業と，手元の PC 画面に表示された C 言語のソースコード 10 行と，同じ PC 画面上の QuickBoard に表示された類似ソースコード 10 行とを比較してその差異部分を見つけ出すという作業とが，順次課せられた．なお，公平な実験結果を得るため，被験者 7 名は 3 名と 4 名から成る 2 つのグループに分られ，3 名のグループが被験者の際はテレビモニタにソースコード A，および，QuickBoard にソースコード B を表示し，4 名のグループが被験者の際はテレビモニタにソースコード B，および，QuickBoard にソースコード A を表示するようにした（ソースコード A を用いた実験でも，ソースコード B を用いた実験でも差異部分は 5 カ所ずつ存在した）．そうして，ソースコードの差異部分 1 つを見つけ出すのに要した時間の平均を算出して比較した（制限時間は 2 分に定めた）．実験 P1 の結果は**表 1** のとおりである．なお，QuickBoard 使用時においては，全員が制限時間内に作業を完了したが，テレビモニタ使用時においては制限時間内で作業を完了できない被験者が 3 名存在した．この結果に基づけば，PC 画面から視線を外す必要のない QuickBoard 使用時は，テレビモニタ使用時に比べてはるかに短い時間で「間違い探し作業」を終えることができたことが分かる．これにより，視線を外す必要がないことの効果が確認できた．

実験 P2 においては，微妙に異なる 2 つのイラスト画から差異部分を見つけ出すという作業を被験者に課した．実験 P1 のときと同様に，被験者 7 名には，手元の PC 画面に表示された内容と，手元のテレビモニタまたは QuickBoard に表示された内容とを比較し，差異部分を見つけるよう指示した．イラスト画としては，前記テレビモニタに一度に表示すると細部が見えなくなる程度の，やや複雑な画像を選定して用いた（イラスト画には 7 つの差異部分が存在した）．そのため，QuickBoard に表示する場合は一度に全体を表示するが，テレビモニタに表示する場合は，イラスト画を 4 つの画像領域に分割して，1 つの画像領域あたり 15 秒間ずつ表示するようにした．制限時間を 3 分と定めたため，各画像領域はテレビモニタに 3 回表示されることになる．そうして，差異部分 1 つを見つけ出すのに要した時間の平均を算出して比較した．実験 P2 の結果は表 1 のとおりである．この結果から，内容全体を一度に表示できないテレビモニタ使用時は，内容全体を見ることができる QuickBoard 使用時と比較して，約 1.8 倍もの時間がかかったことが分かる．これにより，一度に全体が見えることの効果が確認できた．

4.2.3 新規課題抽出

一方，著者らは，本システムを利用した講師および学生に対するヒアリングから，新たな問題点または新たな要求を発見した．それらを以下に記す．

1. ソースコード編集操作の説明等では両手がふさがっていることが多く，画面キャプチャのためのキーまたはマウス操作が煩わしい（講師）．
2. 見ているスライドが突然切り替わるため，前のスライドをもう少し見たい場合がある（学生）．

次章では，これらの問題点に対策を施した第 2 プロトタイプについて述べる．

5. 第 2 プロトタイプ

5.1 基本機能

第 1 プロトタイプの実利用実験によって明らかとなった課題を解決することを必要要件と設定し，第 2 プロトタイプをデザインした．第 2 プロトタイプに新たに実装した機能は以下のようなものである．

1. 指定したウィンドウ領域に変化があった場合に，自動的にスナップショットファイルを作成する（オートキャプチャ機能）．
2. スナップショットを更新した場合にでも，以前のスナップショットを参照できるようにする（スライドヒストリ表示機能）．

1 に関し，任意のデスクトップ画面領域を講師があらかじめ指定しておけば，QuickBoard Capture がこの画面領域のピクセル変化を一定時間間隔（1 秒間隔程度）でチェックし，変化があったと検出された場合に自動的にスナップショットイメージファイルを作成するようにした[8]．なお，Emacs 等のエディタを用いた場合，文字入力位置を指し示すカーソルが点滅しており，その点滅によるピクセル変化を無視する必要があった．これに関しては，閾値（500 ピクセル程度）を設け，この閾値以下のピクセル変化は無視するようにして対処した．

2 に関しては，QuickBoard Capture は，スナップ

ショットイメージを生成するたびに自動的にファイル名を変えて保存するようにした．これにより Web ブラウザは，以前に表示されたスライドにも必要に応じてアクセスできるようになった．この機能の追加にともない，Web ブラウザから画像更新チェック要求が到着すると，Web 公開ディレクトリに保存されているスナップショットイメージファイルの中からファイル更新時刻の最も新しいものを探し出し，このファイルが以前配信したものより新しいファイルであった場合に，画像更新された旨の返答を返すように仕様変更した．

なお，1 および 2 に関し，画像更新チェック要求が到着するたびに，サーバ内のファイルシステムをサーチするのは，サーバの I/O 負荷を著しく増加させてしまう恐れがある．さらに，この I/O 負荷は Web クライアント数に比例する．そこで，Web ブラウザから更新チェック要求が到着した場合にでも，前回のファイルシステム検索時刻から一定時間（1 秒間程度）が経過しないうちは，ファイルシステム検索を実行しないようにした．これによって，Web クライアント数に関係なく I/O 負荷を低く抑えることができた．

5.2 追加機能

第 2 プロトタイプの実装に際しては，第 1 プロトタイプの実践利用実験によって明らかとなった課題を解決することを主目的としたが，加えて，第 2 プロトタイプではさらに効率的なネットワーク転送を目的とした「オートクロップ機能」を実装した（**図 4** の実装画面を参照）．

オートクロップ機能は，説明箇所周辺の冗長な余白または背景を自動的に切り捨てて伝送する機能である．第 2 プロトタイプにおいて新規実装したオートキャプチャ機能を使う際，講師は，画面更新検出対象となるデスクトップ画面領域を最初に指定する必要があるが，

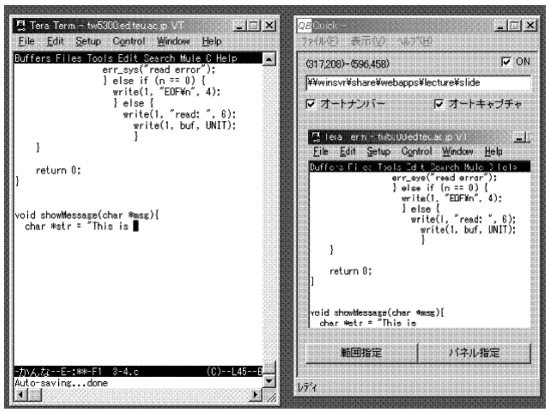

図 4 第 2 プロトタイプ（オートクロップ）
Fig. 4 Prototype-2 (auto crop activated).

「テキストエディタ全体」というような画面領域を指定することが普通である．この場合，エディタに数行しか文字列が表示されていなくても，エディタ全体のスナップショットイメージが作成されるため，白地の部分等が多く含まれてイメージファイルサイズが大きくなってしまう．そこで，背景と認識できる色（画面領域全体に多く分散する色）や背景と認識できるパターン（画面領域の末端部分に均等に存在する繰返し模様）のみから成る冗長領域を特定し，それらを含まないようにして矩形領域のスナップショットイメージを作成するようにした．

白無地背景のテキストエディタ（640 × 480 ピクセルで使用）にソースコード 10 行を表示してテストしたところ，オートクロップ前に 4,750 Byte であったスナップショットイメージのファイルサイズが，オートクロップ後には 330 × 150 ピクセルにリサイズされて 3,244 Byte に減少した．このテスト結果からは，画像サイズが約 62%に減少したときに，ファイルサイズが約 70%に減少したことが分かる．著者らは，同様なテストを実際の講義用教材を使用して繰り返し行い，冗長な無地背景部分を切り落とすだけで PNG 画像のファイルサイズを有意に減少できることを確認した．

5.3 評価と考察

開発した第 2 プロトタイプを，前述した「UNIX システムプログラミング演習」の授業で実際に利用する実験を行い，第 1 プロトタイプと第 2 プロトタイプの違いについてヒアリング調査を実施した．この結果，オートキャプチャ機能について講師からは，「画面キャプチャ処理が自動的に行われるようになったため，授業内容自体に集中できるようになった」という好意的な意見を得ることができた．一方，ヒストリ機能については，著者らが期待したほどには，学生は利用していないようであった．理由としては，本演習授業の授業ペースが比較的ゆっくりしており，前のスライドに戻って確認する必要があまりなかったことが考えられる．ただし，スライドヒストリが授業後の講義録として用いられる可能性があり，このような利用用途において有効な機能と思われる．

効率的なネットワーク転送を目指して第 2 プロトタイプではオートクロップ機能を実装したが，さらに同時接続数が増えるような状況に備え，著者らは，個々の Web ブラウザへの画像配信タイミングを自動的に分散させる「タイミング分散配信機能」の実装について検討を行っている．現在の実装では，新しいスナップショットイメージファイルが QuickBoard Capture によって作成されたことを QuickBoard Servlet が検

知した際に，QuickBoard Servlet は，そのスナップショットイメージをただちに取得して表示するための JavaScript プロシジャを各 Web ブラウザに送信していた．しかしながら，多数の Web ブラウザがいっせいにスナップショットイメージを要求した場合には，ネットワークトラフィックが極度に増大してしまうという問題がある．

通常 Web サーバには，スレッド同時実行数の上限を制御する機能（最大スレッド数制限機能）が備わっており，最大スレッド数を超えた要求が到着した場合は，その要求は待ち行列に入るようになっている．この機能を用いれば同時刻に実行されるスレッドの数は制御できるが，ネットワークトラフィックや Web サーバの負荷を考慮する場合には，ある時刻一点の負荷だけでなく，たとえば「0.5 秒間」というような単位時間あたり（局所的時間内）のトータル負荷を考慮する必要がある．負荷が局所的に増加すると，Web サーバが提供している他のサービスや，Web サーバが存在するネットワーク上の他のネットワークサービスに障害を発生させる可能性が高まる．単位時間内に実行されるスレッドの数は，前記スレッド同時実行数の上限を増減させてもあまり変化しない．これは，同時に実行できるスレッド数が少ない場合は，個々のスレッドの処理が早く終了するため，待ち行列に入っているスレッドの処理が早く開始されるためである．

そこで，新しいスナップショットイメージファイルが作成されたことを QuickBoard Servlet が検知した場合に，個々の Web ブラウザごとに画像取得タイミングを変えるように指定して，各 Web ブラウザはその指定された時刻（数秒間後）まで待ってイメージファイルを取得するように構成すれば，急激なトラフィック増大を防ぐことができると考えられる．Web ブラウザからの画像取得タイミングを，数秒間の範囲で均等に散らすようにすることで，0.5 秒というような局所的時間内にネットワークトラフィックや Web サーバの負荷が集中することを防ぐ．この機能についは現在実装を計画中である．

5.4 研究の位置づけ

遠隔講義支援システムとして本システムをとらえた場合の，本研究の位置づけについて議論する．遠隔講義支援システムとしては，同期型のものと非同期型のものとが存在する．非同期型の例としては，重野らの遠隔教育システムがある．重野らのシステム[9] は，同期対面講義を収録し，その収録した講義風景や講義コンテンツを遠隔非同期で利用可能とすることで，講義の疑似体験ができるようにしている．ただし，QuickBoard のような，講義資料や PC 操作画面のリアルタイム配信機能は備えていない．

同期型の例としては，田中らの遠隔講義支援システム[10] がある．田中らのシステムは，PowerPoint ファイル等をサーバにアップロードし，サーバ上で画像ファイルに変換して各 Web ブラウザに配信している．アップロードした資料を配信するという性質上，PC 操作画面をリアルタイムで配信することや，任意のアプリケーションの画面を配信することには対応していない．また，QuickBoard のオートキャプチャ（画面変化があったときだけ画面キャプチャを実行する機能を含む）およびオートクロップに相当するような，ネットワークトラフィック削減に関する機能は提供していない．

講義支援システムの範疇ではないが，PC 画面をキャプチャし，一定間隔で Web ブラウザにそのキャプチャ画像を配信するデスクトップ画面配信ツールとして Web-Picture[11] がある．しかし，Web-Picture は QuickBoard のオートキャプチャおよびオートクロップ等に相当するような，ネットワークトラフィック削減に関する機能を提供していない．また，Flash プラグインまたは Java アプレットが Web ブラウザに導入されていなければならず，必ずしも汎用 Web ブラウザで動作しないという問題がある．

このほかにも，文字コンテンツを Web ブラウザに配信するツールがある．NEWSTAG[12] は，あらかじめ Web サーバに蓄積したコンテンツを，Web ブラウザ上の Flash プラグインや Java アプレットを用いて表示する．PC 画面を配信する機能や，ネットワークトラフィック削減に関する機能は提供しない．

6. ま と め

本論文では，リアルタイム型プレゼンテーションをサポートする Web システム QuickBoard を提案した．Web システムが普及する以前，「グループウェアはなぜ失敗するか」という議論がなされることがしばしばあった[13]．しかし現状において，幸いにも Web ベースのグループウェアは大きな成功を収めている．新しいアプリケーションの利用を強いられるという抵抗感や，慣れ親しんだユーザインタフェースが使えなくなるという不安感が低減されるという Web システムのメリットが，グループウェアの成功に貢献した可能性がある．このことからも，著者らは，Web システムの利点を活かしつつリアルタイムに情報を配信できるグループウェアを構築することは意義のあることと考えている．

参 考 文 献

1) Stefik, M., Foster, G., Bobrow, D.G., et al.: Beyond the Chalkboard: Computer Supported for Collaboration and Problem Solving in Meetings, *Comm. ACM* (Jan. 1987).
2) http://www.microsoft.com/windows/netmeeting/
3) http://www.microsoft.com/japan/windows/windowsmedia/
4) http://www.uk.research.att.com/vnc/
5) Rao, K.R. ほか：デジタル放送・インターネットのための情報圧縮技術，pp.200–237, 共立出版 (1999).
6) http://www.libpng.org/pub/png/
7) 市村，宇田，伊藤，田胡，松下：QuickBoard：汎用 Web ブラウザのためのリアルタイムスライド配信サービスの試作，情報処理学会グループウェアとネットワークサービス研究会報告，GN-46-7, pp.37–42 (2003).
8) 市村，中村，宇田，伊藤，田胡，松下：任意のアプリケーション画面をリアルタイム配信する Web サービスの提案，日本 VR 学会研究会報告，CSVC 2003-1, pp.15–20 (2003).
9) 重野，間下，楢原，松下：講義イベントに着目した XML ベース遠隔教育システム，情報処理学会論文誌，Vol.42, No.9, pp.2319–2327 (2001).
10) 田中，勅使河原：Web ページやその部分構成要素のリアルタイム共有・記録機能の設計と開発，情報処理学会第 65 回全国大会第 4 分冊，pp.21–22 (2003).
11) http://www.teleboard.jp/wp-general.htm
12) http://www.newstag.net/
13) Grudin, J.: Why CSCW Applications Fail: Problems in the Design and Evaluation of Organizational Interfaces, *Proc. ACM CSCW '88*, pp.85–93 (1988).

(平成 15 年 5 月 27 日受付)
(平成 15 年 11 月 4 日採録)

市村　哲（正会員）

1966 年生．1989 年慶應義塾大学理工学部計測工学科卒業．1994 年同大学大学院理工学研究科博士後期課程修了．博士（工学）．同年富士ゼロックス（株）入社．1997 年〜1999 年富士ゼロックスパロアルト研究所駐在．2002 年 4 月より東京工科大学助教授．グループウェア，ネットワークサービスの研究に従事．『IT TEXT 基礎 Web 技術』，『IT TEXT 応用 Web 技術』（オーム社）．DICOMO 2003 優秀論文賞受賞．ACM 会員．

中村　亮太（学生会員）

1981 年生．2002 年東京工科大学工学部情報通信工学科卒業．現在，同大学大学院工学研究科博士前期課程に在学中．無線通信工学，ヒューマンインタフェースに興味を持つ．

伊藤　雅仁（正会員）

1998 年慶應義塾大学理工学部計測工学科卒業．2000 年同大学大学院理工学研究科修士課程修了．2003 年同大学院理工学研究科後期博士課程修了．博士（工学）．現在，東京工科大学コンピュータサイエンス学部専任講師．モバイル・コンピューティング，情報家電，デジタル放送の研究に従事．2000 年情報処理学会高度交通システム研究会優秀研究報告賞受賞．電子情報通信学会員．

宇田　隆哉（正会員）

1998 年慶應義塾大学理工学部計測工学科卒業．2000 年同大学大学院理工学研究科計測工学専攻前期博士課程修了．2002 年同大学院理工学研究科開放環境科学専攻後期博士課程修了．2003 年 4 月より東京工科大学コンピュータサイエンス学部専任講師．博士（工学）．ネットワークセキュリティの研究に従事．2002 年 IFIP/SEC2002 Best Student Paper Award 受賞．電子情報通信学会会員．

田胡　和哉（正会員）

1986 年筑波大学工学研究科博士課程修了．工学博士．筑波大学電子情報工学系助手，東京大学工学部助手，日本 IBM 東京基礎研究所を経て，2003 年より東京工科大学コンピュータサイエンス学部助教授．オペレーティングシステムの構成方式に興味を持つ．1984 年情報処理学会論文賞受賞．ACM 会員．

松下　　温（フェロー）

1963 年慶應義塾大学工学部電気工学科卒業．1968 年イリノイ大学大学院コンピュータサイエンス専攻修了．1989 年より 2002 年 3 月まで慶應義塾大学理工学部教授，2002 年 4 月より東京工科大学コンピュータサイエンス学部長，教授および慶應義塾大学理工学部客員教授．マルチメディア通信，コンピュータネットワーク，グループウェア等の研究に従事．情報処理学会理事，同学会副会長，マルチメディア通信と分散処理研究会委員長，グループウェア研究会委員長，電子情報通信学会，情報ネットワーク研究会委員長，MIS 研究会委員長，バーチャルリアリティ学会サイバースペースと仮想都市研究会委員長等を歴任．現在，情報処理学会 ITS 研究会委員長，郵政省，通産省，建設省，農水省，都市基盤整備公団，行政情報システム研究所等の委員長，座長，委員を多数歴任,『やさしい LAN の知識』（オーム社），『201x 年の世界』（共立出版）等著書多数，1993 年情報処理学会ベストオーサ賞，1995 年および 2000 年情報処理学会論文賞，2000 年 10 月 20 日情報処理学会 40 周年記念 90 年代学会誌論文賞，2000 年 10 月 2 日電子情報通信学会フェロー，2000 年 10 月 VR 学会サイバースペース研究賞，2001 年 5 月情報処理学会功績賞，2002 年 3 月情報処理学会フェロー，電情報通信学会，人工知能学会，ファジイ学会，IEEE，ACM 各会員．