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

○市村 哲; 宇田 隆哉; 伊藤 雅仁; 田胡 和哉; 松下 温
東京工科大学

近年、企業や大学等における情報システムのほとんどが、Web システムとして実装されている。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ユーザに配信するようなサービスに向かないという特徴がある。情報のリアルタイム配信は、同期型グループウェアを設計する上で不可欠な機能であり、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非常に意義があることと思われる。本論文では、任意のアプリケーション画面を、PC、PDA、プロジェクタ機器等の Web ブラウザにリアルタイムに配信できる Web サーバシステム QuickBoard を報告する。

QuickBoard:a realtime slideshow service for generic web browsers

○Satoshi Ichimura; Ryuya Uda; Masahito Ito; Kazuya Tago; Yutaka Matsushita
Tokyo University of Technology

Many 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 creating any realtime groupware due to the lack of realtime communication capabilities of HTTP. It, however, would be obviously meaningful to develop a realtime groupware which takes advantage of the Web server side technology. We developed QuickBoard Web server system which allows presenters to deliver presentation slides of any applications to various Web browsers in realtime manner.

1. はじめに

近年、企業や大学等における情報システムのほとんどが、Web システムとして実装されている。クライアントサーバシステムが主流の時代には、如何にクライアントソフトを配布するかが常に大きな問題となっていたが、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アプリケーションをインストールする必要がない。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 毎に異なる実行モジュールを作成して全ユーザに一斉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よい。以上のような利点から、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。

また、PC 以外のデバイスにおいても、Web ブラウザを搭載するものが増えてきた。例えば、現在一般的に販売されている PocketPC 2002 は、Internet Explorer（以下、IE）を標準で搭載しており、PDA ながらに通常の Web ページの多くをそのままみることができる。この PocketPC に搭載されている IE は、ディスクトップ PC 用 IE のサブセットではあるものの、JavaScript や SSL をサポートしているなど、動的なページを表示する能力や業務アプリケーションに対応する能力を備えている。一方、ネットワーク環境に関しても、コンパクトフラッシュ型の無線 LAN カードが普及しつつあるなど、PDA であっても Web アプリケーションを実行できる環境が整いつつある。前述したトレンドから考えると、現在は未だ専用アプリケーションとして実装されることが普通の PDA ソフトも、サービスの種類よっては、次第に Web サービスアプリケーションへと遷移することが予測できる。

以上のように利点の多い Web システムであるが、反面、機能上の制約が多いというデメリットがある。とりわけ、HTTP の性質上、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特徴があり、システ

ムを設計する際の制約となっている。しかしながら、情報をリアルタイムにユーザ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上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上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、我々は、同期型プレゼンテーションをサポートする Web システムの開発を試み、QuickBoard システムを試作した。プレゼンテーションという行為は、大学の講義、企業内教育の場、セミナー等において情報シェアのための基本的な行為であり、極めて重要なグループ活動の１つと言える[1]。本論文では、任意のアプリケーション画面を、PC、PDA、プロジェクタ機器等の Web ブラウザにリアルタイムで送信できる QuickBoard システムの特徴について説明する。まず、我々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プロトタイプシステムの実装について述べる。さらに、このプロトタイプシステムをユーザの実利用環境に適用した結果について報告する。

2. 現状把握と要求分析

2.1. 実践環境

システムの実用性を高めるため、我々は、実際のユーザが存在する環境を対象として要求分析を行うこととした。今回実践的実験の場として選んだ環境は、我々自身の大学の授業であり、教育の現場である。特に、「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」を具体的な実験対象とした。後述する通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、解決すべき課題が既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、全員が一斉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。学生は、実習プログラムを自分のホームディレクトリに作成することはできるが、基本システムは計算機管理スタッフによって管理されており、学生または教官が自由にソフトをインストールすることはできない。

教官のプレゼンテーションの道具は、書画カメラとホワイトボードのみである。この計算機室は、何年も前に建造された教室であるため、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るのみで、大画面スクリーン等には投影されない(図 1参照)。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。ホワイトボードに書いた文字は教室の一番遠いところからは見ることが難しいため、実質的に書画カメラのみに頼らざるを得ない。

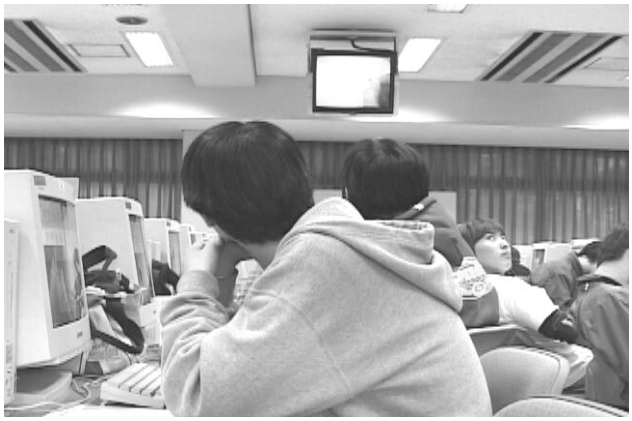

図 1　計算機室の現状

2.2. 現場観察による問題点の分析

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わかった。この時、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。講師は、ペン等でポインティングしながら説明するが、その説明を見るために学生はいちいち自分の PC 画面から視線をはずす必要があり、その都度、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。

事実、長時間観察を続けると、授業の開始時点は多くの学生がテレビを見上げる動作をしているが、授業が進むに従って、次第に、講師の声だけを聞きながら自分の PC 画面を見て理解しようとする学生が増えてくる様子が明確に見て取れた。このとき、当然、講師がポインティングしているところを生徒が注視している確証はなく、しばしば、どこの説明をしているのかわからなくなっている様子も観察された。

以上の観察結果から、我々は、この計算機室での実習において QuickBoard システムが特に有効に機能するであ

ろうと仮説を立て、この授業で起きている現状の問題点を解決することを具体的な目標と設定した。

2.3. 必要要件

前述した授業観察結果に基づき、QuickBoard の設計にあたっては以下の要求機能を実装することを目標とした。

1. 講師が講義に使うアプリケーションを限定しない
   Emacs、VisualStudio、PowerPoint、Web ブラウザ等の画面を共有する
2. Web ブラウザ以外のソフトを必要としない
   OS を特定しない、クライアントソフトを必要としない、PDA でも利用可能
3. 同時に 100 人以上が接続できるように構成する
   １サーバに全員が同時接続できるようにする

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。さらに、汎用性の高いシステムとするためには、UNIX 以外の開発環境（VisualStudio 等）への対応も必要になることが明らかである。加えて、講義には PowerPoint や Web ブラウザが用いられることが非常に多いために、これら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言える。

２に関しては前述した通りである。ユーザ環境は、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。３に関しても、前述した通り、100 名の受講学生を支援対象と設定しているため最低限の要求とした。

3. システムデザインと実装

3.1. 機能概要

我々は、以上に設定した必要要件を満たすべく、以下の機能を実装した(図 2参照)。

1. 講師が用いる PC の表示画面をスナップショットイメージとして転送する
2. 転送する画像領域をアクティブなウィンドウのみとし、ネットワーク負荷を抑える
3. 画像更新が必要な場合にだけイメージデータ転送を実行し、ネットワーク負荷を抑える

1、２、および、３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 Tomcat サーバで動作する Java サーブレットである。以下、それぞれのコンポーネントについて詳細に説明する。

3.2. QuickBoard Capture

QuickBoard Capture は、講師が用いているノート PC の画面のスナップショットを取り、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。

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、イメージファイルの保存形式としては、現在、ネットスケープや IE 等のブラウザに表示できる PNG 画像形式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利点を持っている[2]。さらに、GIF 圧縮が 256 色までしかサポートしないのに比較して、PNG 圧縮はフルカラーに対応している。

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。講師がキーボード操作またはマウス操作にて画像範囲を指定した瞬間に、その画像範囲を PNG 画像ファイルとして出力するようになっている。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（現状は SHIFT/CTRL＋ファンクションキーによる指定）が押された場合に、その時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。

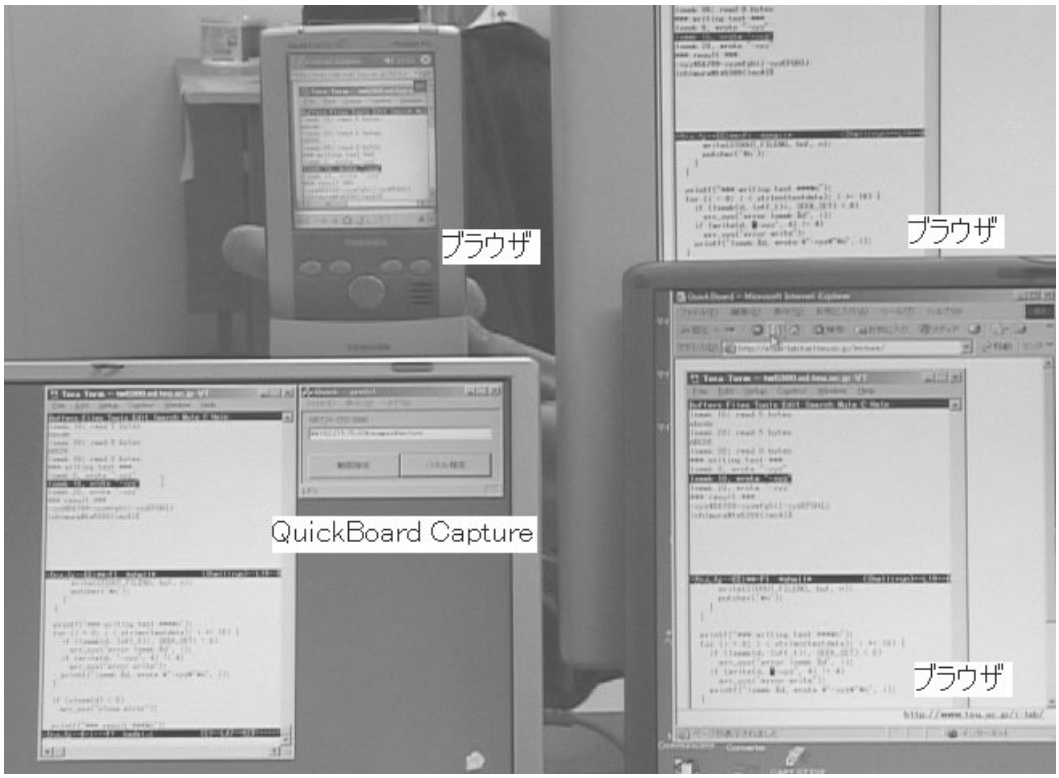

図 2　プロトタイプシステム動作画面

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 サーバ上のある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機能によって講師の PC から書き込み可能となっている。ただし、安全性を確保するために、Web サーバはファイアウォールの内側に設置されており、講師の PC が Web サーバのファイルシステムにアクセスする場合は、VPN(Virtual Private Network)でファイアウォールを越えて Web サーバにアクセスするように構成されている。現在の実装では、VPN として、Windows PC に標準で搭載されている PPTP（Point-to-Point Tunneling protocol）方式を用いており、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。

3.3.　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 PC 画面領域を、画像更新が必要な場合にだけユーザに転送するという機能を備える。

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不可能である。そこで、QuickBoard では、画像更新があったかどうかを Web サーバに定期的に問い合わせる小さなプロシジャを 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という方法をとっている。この Web ブラウザで動作するプロシジャは JavaScript によって記述されている。ネットワーク負荷を軽減するためにはこのポーリングの時間間隔は大きいほうが望ましいが、講師がキーまたはマウス操作によって画面配信指令を発してから学生の PC 画面上に表示されるまでの間が空き過ぎると、講師がストレスを感じるようになる。このため、使用経験に基づいてポーリングの時間間隔は5乃至10秒以下となるように設定した。

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、QuickBoard Servlet は、講師用 PC に公開しているディレクトリを検索し、スナップショット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ようになっている。イメージ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量は、画像データの転送量と比較して極めて小さいデータ量であり、画像が更新された場合のみ、画像データがネットワークを流れるように構成されている。

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リクエストが約 480 バイト(全て HTTP ヘッダ)、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が約 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 480x640 ピクセル程度のスライド画像は、PNG 画像形式で 50KB 程度である。すなわち、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従来の場合と比較して、約 150 分の１程度のトラフィックで済むことがわかった。

4. 実利用実験、評価、今後の課題

実装した QuickBoard のプロトタイプを、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。この実験の目的は、作成したツールを自ら実利用してみて、設計時点ではわからなかった使用感や問題点を、発見および体感することであった。

この実利用実験の結果、書画カメラを使用していた時と比較して以下のような利点を体感した。

1. 学生は、自分のプログラムとソースコードレビューとを画面上で見比べることができるようになった
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画面を見せながら説明でき、学生の理解が高まることが期待できた
4. コンパイル操作やプログラム実行の様子をそのまま表示でき、説明内容が豊富になった
5. エディタの機能を用いて注目させたい箇所をハイライト表示でき、説明箇所を示しやすくなった

また、続いて実施した、学生数名に対する口頭インタビューからも、上記１、３、および４の利点を評価する声を得ることができた。

なお本実験は、書画カメラしか使用できないという、現在ではむしろ珍しい環境で実施されたが、上記利点のうち、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。さらに、QuickBoard は、同じ部屋でしか利用できない PC プロジェクタと比べて、そもそも遠隔での利用が可能という特性があるため、グループウェアとしての応用範囲が広いというアドバンテージを有している。

一方、90 分授業を通してシステムを利用し、利用上の問題点または新たな要求を発見した。発見した項目は、講師側の利便性に関するものと、学生側の利便性に関するものに大別された。

1. (講師側)ソース編集過程を見せたい場合、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
2. (講師側)Emacs 画面（ソースコード）と Web 画面（課題内容等）等、複数画面を同時に提示したい
3. (学生側)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
4. (学生側)カットアンドペーストして文字列を取り出せるようにテキスト形式でも受信したい

１に関しては、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機能を実装することで対処できると考える。具体的な方策としては、講師が指定したウィンドウ領域のピクセル変化を一定時間間隔でチェックする機能を QuickBoard Capture に持たせることを考えている。

２に関しては、必要に応じて複数画面を同時表示できる機能を実装することで対処できると考える。具体的には、領域の同時表示が必要になった場合に QuickBoard Capture プロセスを増やせるようにし、このプロセス数に応じて同時画面表示数を増やせるようにする。

３に関しては、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残しヒストリ表示できるようにすることで対処できると考えている。具体的には、QuickBoard Capture が、スナップショットイメージを生成する度に自動的にファイル名を変えて保存するようにし、また、QuickBoard Servlet が、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て、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のみ画像更新するようにする。

5. 関連研究

アプリケーション画面やディスクトップ画面を共有するソフトウェアは、ネットワークコンピューティングまたは遠隔会議支援システムの発展に伴って増えてきている。これらのソフトウェアを同期対面型、同期遠隔型に大別した場合、前者の先駆けとしては Colab[3]、後者の先駆けとしては MERMAID[4]等が挙げられる。

近年特に、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が増えている。この類のソフトウェアとして最も有名なソフトウェアの 1 つである Netmeeting[5]は、Microsoft Windows 上で動作する、H.323 規格および T.120 規格に準拠した遠隔会議支援システムである。最近では、WindowsXP にリモートアシスタントという遠隔ディスクトップ共有ツールが標準搭載されるようになっている。これらのソフトウェアは高機能である代わりに、使用できる OS が Windows のみに限定されている。また、専用クライアントソフトウェアが必要なことや、同時接続数の上限が極めて小さいために、今回対象としたような使用環境においては用いることができないという問題がある。

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 VNC[6]がある。Java アプレット判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極めて小さく、今回対象としたような使用環境においては用いることができない。また、前述した PocketPC 2002 の IE 上では JavaApplet が動作しないため、PocketPC 2002 で用いる場合には専用のアプリケーションの開発が必要となっているのが現状である。

6. まとめ

現段階は、本プロトタイプの評価結果に基づき、第二プロトタイプの実装を進めている段階である(図 3参照)。第二プロトタイプが完成した際には、本プロトタイプの利用実験のように、実際の現場でツールを使用し、さらにあらたな改善点を発見したいと考えている。

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あった[7]。しかし、Web ベースのグループウェアが成功している現状を鑑みると、当時は、先進的ユーザであればある程、人から強制されたグループウェアソフトを自分の PC にインストールする抵抗が強く、自分の PC 環境を壊してしまうのではないかとの心理的障壁が高かったように感じられる。また、慣れ親しんだユーザインタフェースが使えなくなり、新しい操作を一から覚えなければならないのではないか、という不安感が利用者の声として多かったのも確かである。これらのことがユーザのグループウェア離れを引き起こし、システムとして失敗する原因となっていた可能性がある。Web システムとして構成することで、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 UI が使えなくなるという不安感が低減されるため、グループウェアが成功する確率が向上するのではないだろうか。

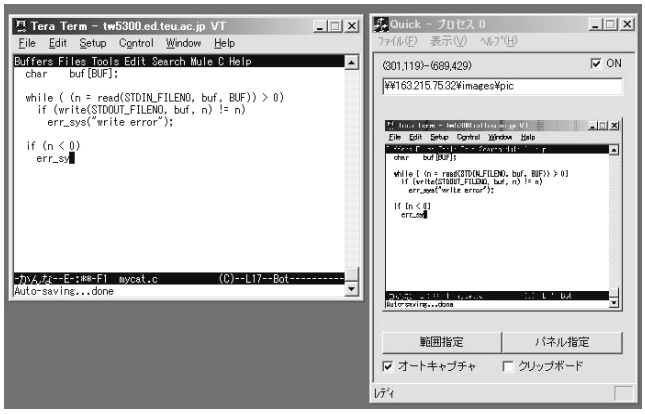

図 3 開発中の第２プロトタイプシステム

7. 参考文献

1. 市村, ネルソン, ペダーセン : CardGear - 紙カードで操る電子プレゼンテーションシステム, 情報処理学会 インタラクション 2000, pp.17-24 (2000).
2. http://www.libpng.org/pub/png/
3. Stefik. M., Foster, G. Bobrow, D.G., et al.: Beyond the Chalkboard: Computer Supported for Collaboration and Problem Solving in Meetings, Communications of the ACM, (Jan 1987).
4. Watabe, K., Sakata, S. Fukuoka, H., et al.: Distributed Multiparty Desktop Conferencing System: MERMAID, Proc. of CSCW'90, pp.27-38 (1990).
5. http://www.microsoft.com/windows/netmeeting/
6. http://www.uk.research.att.com/vnc/
7. Grudin, J.: Why CSCW Applications Fail: Problems in the Design and Evaluation of Organizational Interfaces, Proc. of ACM CSCW'88, pp.85-93 (1988).