

CONTENTS: 携帯電話上での同期型ホワイトボードの実現

太田 雅敏[†] 吉滝 幸世[†] 坂根 裕[‡] 石原 進^{*} 水野 忠則[†]

[†]静岡大学大学院情報学研究所 [‡]静岡大学情報学部 ^{*}静岡大学工学部

CONTENTS: Synchronous Whiteboard System on Mobile Phones.

Masatoshi OHTA[†] Sachiyo YOSHITAKI[†]

Yutaka SAKANE[‡] Susumu ISHIHARA^{*} Tadanori MIZUNO[†]

[†]Graduate School of Information, Shizuoka University

[‡]Faculty of Information, Shizuoka University, ^{*}Faculty of Engineering, Shizuoka University

1 はじめに

遠隔地のユーザと共同で絵が描ける仮想ホワイトボードシステムが、PC 上のアプリケーションとして多く実装されている。このようなホワイトボードは、待ち合わせ場所の地図を共有したり、仕事の打ち合わせ等に利用でき便利である。しかし、これらのシステムは、ユーザが PC を使用できる状況にあることを前提としており、タクシーによる移動中や、駅のプラットフォームにいる時に利用することは困難である。近年、携帯電話の普及に伴い、DYNAMIX[1] や Rajicon[2] といった、無線ネットワークを利用した携帯電話用アプリケーションが数多く公開されている。

筆者らはこれまでに、携帯電話を持つ複数のユーザが、共同して絵が描ける仮想ホワイトボードシステム CONTENTS(Canvas ON mobile TElephone with NeTwork Synchronization)[3] を提案している。PC 上のホワイトボードと異なり、携帯電話では、通信時の遅延が大きいことや、通信プロトコルに HTTP しか使えないため、サーバからのプッシュが使用できないという制約が、実装上の問題点として存在する。CONTENTS は、クライアントがサーバと通信を行う間隔を逐次変更することにより、これらの問題を解決する。本稿ではこのシステムのシミュレーション及び、実環境での運用テストによる評価について述べる。これらの評価により、本システムの通信効率、ユーザ間の描画内容の整合性、スケーラビリティを明らかにした。

2 同期式共有ホワイトボード CONTENTS

2.1 概要

CONTENTS は携帯電話を用いた仮想のホワイトボードを実現するシステムである。図 1 に CONTENTS のシステム構成を示す。このシステムは、描画情報やセッション情報を管理する 1 つの描画データ同期用サーバと、それに接続する各クライアントから構成される。各クライアントは 1 つの仮想的なキャンバスを共有し、同期用サーバと描画操作イベントを HTTP により送受信することで、キャンバス内容を一致させる。CONTENTS の入力インタフェースは、携帯電話によるお絵かきソフト CONTE[4, 5, 6] と同様のものを採用しており、数字キーによる 16 方向のカーソル移動を実現している。描画機能として、単純なペンによる描画の他、矩形や円(楕円)、テキストなどの入力をサポートする。

作成した画像は、共同作業が終了すると自動的にサーバに保存される。CONTENTS では、全てのユーザが作業を終了したというメッセージを明示的に送信するか、セッションタイムアウトにより全てのユーザが離脱したとみなされた時に、共同作業が終了したと判断する。作成された画像は Web ブラウザから参照できる形式で保存され、他のアプリケーションから利用することができる。

2.2 描画データの同期手順

携帯電話網を介した通信は遅延が大きいため、複数のユーザが描画した図をリアルタイムに同期させることは難しい。サーバ上で管理されている描画順序通りに描画操作を画面に反映しようとすると、

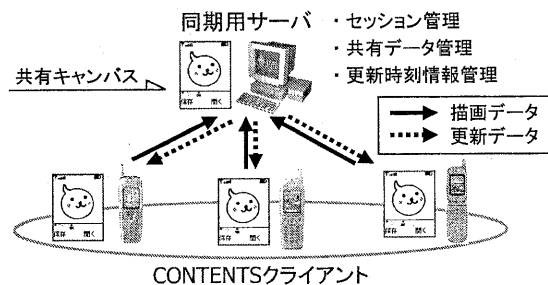


図 1: CONTENTS の構成

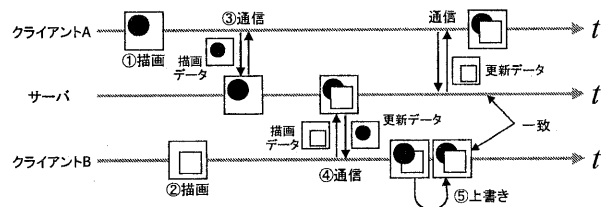


図 2: キャンバス内容の同期手順

描画操作を行った後に、一度サーバにアクセスする必要があるため、ユーザは快適な描画を行えない。そこで CONTENTS は、ユーザの描画操作をすぐに画面に反映させ、次のサーバとの通信が終わってから再描画処理を行うことにより、キャンバス内容の整合性を保持している。グループ内で基準となるキャンバス内容は、サーバが管理しており、クライアントからデータを受信した順にその内容を更新する。この同期手順を図 2 に示す。各クライアントは描画操作(①②)を行っても、すぐにはサーバにアクセスしない。ただし、ローカルのキャンバスには描画内容を反映する。サーバと通信する際に、それまでにバッファリングされた描画データを送り、その応答として他のユーザによるキャンバス更新データを受信する(③④)。クライアント B は受信した更新データでローカルのキャンバス内容を更新し、次に自分が通信前に行った描画操作を再生することにより、サーバ上のデータとの整合性を保つ(⑤)。

一般的に、画像を利用するアプリケーションが扱うデータは、ラスタ形式とベクタ形式の 2 種類のデータに分類される。ラスタデータはピクセルごとの色情報を保持し、ベクタデータは線や矩形の始点座標や終点座標、ペンの色、太さなどの情報で表現される。CONTENTS は必要に応じて両方の形式のデータを使い分ける。現状の携帯電話はメモリ容量と処理速度の制約が大きいため、クライアントではラスタデータのみを保持している。サーバとの通信の際には通信コスト削減のため、ベクタ形式で描画データをやりとりする。この描画データは、ユーザが描画操作を行っている間に逐次生成される。例えば、線分は 1 つのベクタデータ、曲線は複数のベクタデータとして作成され、クライアントのバッファに格納される。

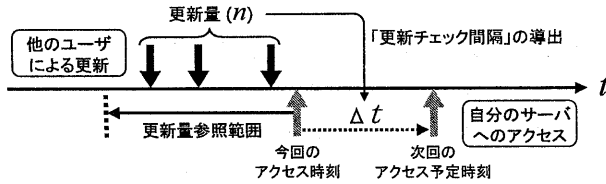


図 3: 更新チェック間隔の導出

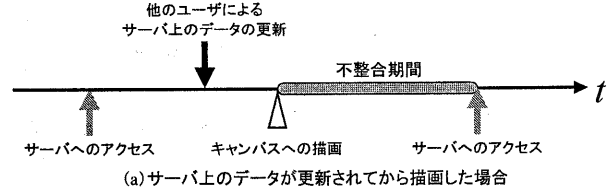
2.3 更新チェック間隔の動的変更

本システムを実装するプラットフォームでは、通信プロトコルに HTTP を使用するため、サーバからのプッシュによる通信が利用できないと仮定している。サーバからの応答を意図的に遅延させることにより、擬似的なプッシュを実現することが可能であるが、コネクションを維持しているゲートウェイに負荷がかかるなどの問題があり汎用的ではない。そこで、クライアントはサーバ上のデータが更新されているかを確認するために、定期的にサーバへアクセスする必要がある。クライアントがサーバにアクセスする間隔を「更新チェック間隔」と呼ぶ。この間隔を極端に短くすると、無駄なアクセスが増加し、逆に長くしすぎるとキャンパス内容を同期するまでに時間がかかる。CONTENTS ではこの更新チェック間隔を動的に変更することにより、サーバへの効率的なアクセスを実現している。以下、その詳細について述べる。

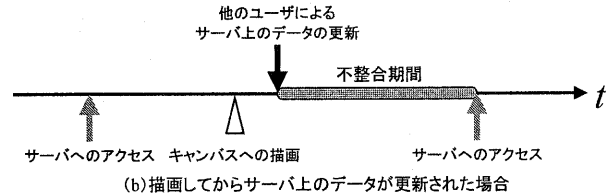
通信コストの削減と応答性の向上 CONTENTS は、新たな画面更新を伴わない意味のない通信を避け、他のユーザの描画情報をなるべく早く取得するために、サーバ上のデータの更新頻度に基づいて、次のアクセス予定時刻を決定する。このアルゴリズムを図 3 に示す。図中の下向き矢印が他のユーザがサーバ上のデータを更新した時刻を表しており、上向き矢印は自分がサーバへアクセスする時刻を表している。各クライアントの次のアクセス予定時刻は、サーバにアクセスした時点でサーバが算出し、応答としてその情報を返す。サーバは、クライアントがサーバにアクセスした時刻を知ると、アプリケーションによって指定される「更新量参照範囲」の時間だけさかのぼった時刻から今回のアクセス時刻までの間に、他のユーザによってどれだけサーバ上のデータが更新されているかを調べる。更新量はサーバ上の共有データを更新したベクタデータの数を表しており、図 3 の n で示される。この更新量によって次のアクセス予定時刻が求められる。更新チェック間隔は、図 3 中の Δt であり、式 (1) により算出する。ここで、 α は「更新間隔係数」であり、共有データの更新量が、どれだけ更新チェック間隔に影響するかを決定するパラメータである。更新間隔係数は 0 以上で 1 より小さい実数値である。また、 Δt_{min} と Δt_{max} はそれぞれ更新チェック間隔の最小値と最大値であり、これらにはアプリケーションごとに適切な値が設定される。

$$\Delta t = \max(\Delta t_{min}, \Delta t_{max} - \alpha n (\Delta t_{max} - \Delta t_{min})) \quad (1)$$

キャンパス内容の一貫性の保持 CONTENTS では、描画中のクライアントが負荷を上げない範囲内で、なるべく早くサーバにアクセスすることにより、各クライアントが表示しているキャンパス内容をできるだけ早く一致させようとする。(1) 式で示した、共有データの更新量により次のアクセス間隔を決定する方法は、アクセスしたユーザが現在描画中であるか、そうでないかを考慮していない。ある時点で描画していないユーザは、サーバから更新データを受信するのみなので、キャンパス内容の整合性がくずれることがはない。逆に現在描画中のユーザは、自分の描画中に他のユーザがキャンパス内容を更新している可能性があるため、キャンパス内容の整合性がくずれることがある。これらをもふまえると、現在描画中でないユーザよりも、描画中のユーザの更新チェック間隔をより短くする必要がある。そこで、ユーザが描画動作を行った場合は、サーバ



(a) サーバ上のデータが更新されてから描画した場合



(b) 描画してからサーバ上のデータが更新された場合

図 4: 不整合期間の定義

表 1: シミュレーション条件

パラメータ	値
シミュレーション時間	3600 (sec)
クライアント数	3, 6
最小更新チェック間隔	5 (sec)
最大更新チェック間隔	15 (sec)
描画イベント送信待ち時間	5 (sec)
更新量参照範囲	15 (sec)
更新間隔係数	0.05
ユーザが描画動作を行う平均間隔	5, 10, 20, 30 (sec)

から示された更新チェック間隔を経過しなくても、一定時間経過後に強制的にサーバにその描画情報を送信する。この時間を「描画イベント送信待ち時間」と呼ぶ。この値は少なくとも最小更新チェック間隔 Δt_{min} 以上の値を設定し、サーバへの過剰なアクセスを防止する。

3 シミュレーションによる評価

CONTENTS の通信方式についての評価をシミュレーションにより行った。更新チェック間隔の動的変更アルゴリズムを用いることで、サーバへの無駄なアクセスがどれだけ減少し、整合性の保持にどの程度有効なのかを検証した。

3.1 評価基準

シミュレーションによる評価基準を設けるために「不整合期間」と「無駄なアクセス」を定義する。

不整合期間 サーバ上のデータとクライアントが保持しているキャンパスデータの一貫性がくずれている間隔。具体的には 2 つのパターンがある。1 つは、サーバ上のデータが更新されてから自分が描画した場合で、自分が描画した時刻からサーバアクセスまでの時間を不整合期間とする (図 4(a))。もう 1 つは、自分が描画してからサーバ上のデータが更新された場合で、サーバ上のデータの更新からサーバアクセスまでの時間を不整合期間とする (図 4(b))。

無駄なアクセス サーバへアクセスする際に、送信する描画データが存在せず、かつレスポンスとして受信すべき更新データも存在しない場合のアクセスを、無駄なアクセスとする。

3.2 シミュレーションモデル

表 1 に示す条件下で、更新チェック間隔を 5~15 秒の間で動的に変更した場合と、これを固定にした場合の、無駄なアクセスの割合と不整合率の変化を調べた。実環境では、サーバ、クライアント間

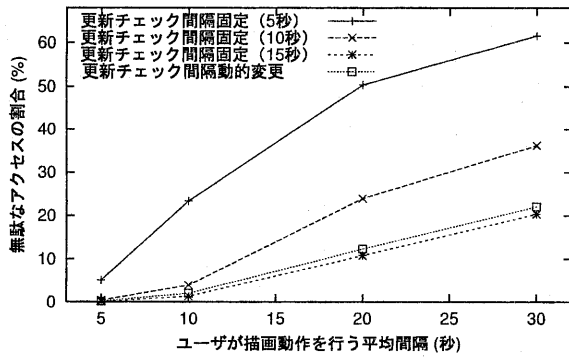


図 5: 無駄なアクセスの割合 (クライアント数 3 の場合)

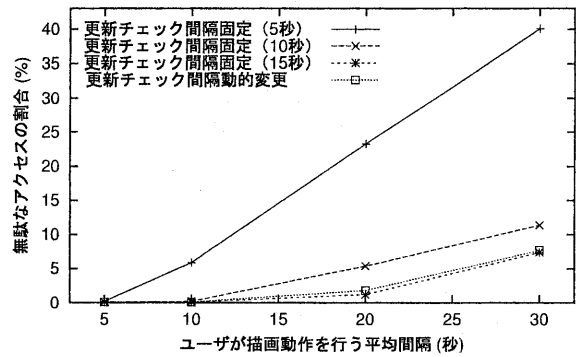


図 7: 無駄なアクセスの割合 (クライアント数 6 の場合)

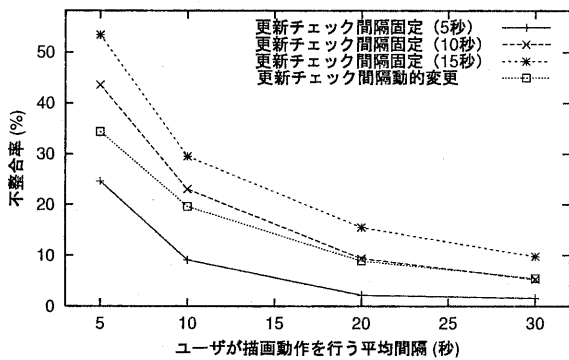


図 6: 不整合率 (クライアント数 3 の場合)

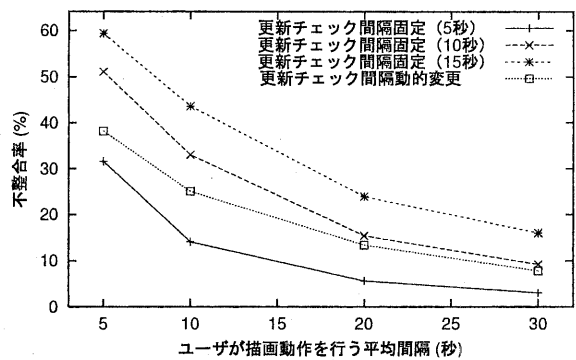


図 8: 不整合率 (クライアント数 6 の場合)

の通信遅延が存在するが、シミュレーションではこの値を 0 とした。ユーザの描画動作はポアソン到着に基づく頻度で行われるとした。

3.3 シミュレーション結果

クライアント数が 3 の時の無駄なアクセスの割合 (無駄なアクセス数/全アクセス数) の変化を図 5, 不整合率 (不整合期間の合計/シミュレーション時間) の変化を図 6 に示す。更新チェック間隔を 15 秒と長くした場合は、当然無駄なアクセスの割合は低くなるが、サーバ上のデータとクライアントが保持しているデータの一貫性を保ちにくくなる。一方、更新チェック間隔を動的に変更した場合は、更新チェック間隔を 15 秒で固定した時と同程度まで無駄なアクセスを抑えつつ、不整合率も比較的強く抑えられている。図 7 と図 8 はクライアント数が 6 の場合の結果である。この場合もクライアント数が 3 の時とほぼ同じ結果が得られた。

共有ホワイトボードの使用用途として、複数ユーザのうち 1 人だけが書き込みを行い、他のユーザはその書き込みを見るだけという状況も考えられる。このような状況を再現するため、1 人だけ描画頻度が高い場合をシミュレートした。図 9 と図 10 は、ユーザ 3 人のうち 2 人の描画動作間隔を平均 30 秒で固定し、残り 1 人の描画頻度を高くした場合の結果を示している。このような状況でも、これまでの結果と同様に、更新チェック間隔を動的に変更した方が効率的な通信を行えることが分かった。これらの結果から、更新チェック間隔の動的変更アルゴリズムは、クライアント数が変化したり、描画頻度変化しても有効に機能するといえる。

4 実環境での運用による評価

CONTENTS クライアントとサーバをそれぞれ i モード Java(NTT DoCoMo N2002), Java Servlet により実装し、実環境での運用テストを行った。

4.1 実験環境

実験では、更新チェック間隔を 5~15 秒で動的に変更するバージョンと、10 秒で固定したバージョンを用いた。手本となる絵を用

意し、それを 2 人で協力して描いてもらい、それぞれのバージョンを用いた場合にどのような違いが現れるのかを調査した。実験は大学生と大学院生、計 10 名に 4 パターンの絵を描画してもらうことにより行った。実験条件は次の通りである。

- 合図により 2 人同時に作業を開始する
- 会話をしてはならない (描画以外のコミュニケーションを禁止する)
- 通常のペンによる描画の他、矩形や円 (楕円) の描画が可能
- 消しゴム機能は使えない (やり直しはできない)

完成した画像の、どの部分をどちらのユーザが描画したかを判断しやすくするために、それぞれ黒と赤のペンのみで描画してもらった。

4.2 実験結果

共同作業によって作成された画像の一部を図 11 に示す。被験者は CONTENTS の操作に十分に習熟していなかったが、1 分半から 7 分程度で 1 枚の絵を完成させることができた。表 2 は、任意の絵を構成する要素 (例えば、雪だるまの帽子や手など) 全体のうち、いくつかの要素を重複して描画してしまったかを示している。パーセンテージが大きいほど、重複して描画してしまった部分が多いことを表す。アクセス間隔を動的変更にした方が、若干ながら重複して描画した割合が少なかった。更新チェック間隔が固定の場合は、相手が頻繁に描画している場合でも、一定の間隔でしかサーバにアクセスしないので、相手の描画した絵が一度に多く表示されてしまう。このため、2 人が同じ場所を描画してしまうことが多く、同じ場所から描画を開始してしまうと、続けて同じ場所を描画してしまう可能性が高いと考えられる。

実験後のアンケートの結果を表 3 に示す。更新チェック間隔を動的に変更した場合の方が良い評価を得られたが、どちらの場合も相手がどの辺りを描画しているのか分からないと、線を重ねて引いてしまうことが多いという意見を得られた。なお、CONTENTS の通信にかかるパケット料金は、選択する料金プランにも依存するが、

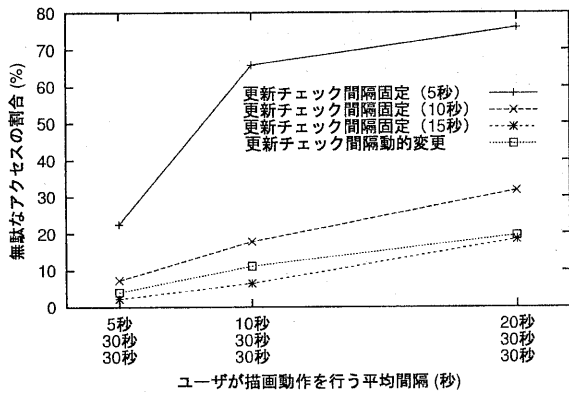


図 9: 無駄なアクセスの割合 (3人中1人だけ描画頻度が高い場合)

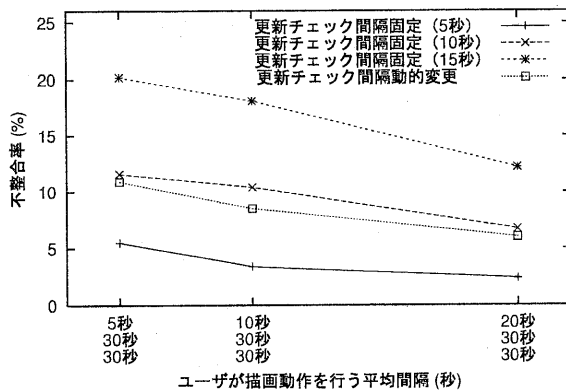


図 10: 不整合率 (3人中1人だけ描画頻度が高い場合)

1分あたり1円～数円程度である。今回の実験のように連続して描画操作を行った場合、どちらのバージョンを用いてもポケット料金にほとんど差は生じない。

4.3 考察

実環境での運用テストにより、相手がどのくらい描画を行っているかという情報よりも、相手が現在描画中であることを示す情報や、相手がどの辺りを描画しているのかを示す情報が必要とされていることが分かった。しかし、CONTENTSの通信形態では、自分が現在描画中であることを他のユーザに伝えることは難しい。もし、自分と共同作業を行っているユーザが、現在描画中であるか描画中でないかを予測することができれば便利である。クライアントがサーバと通信する時に、ユーザがキー入力を行っていた場合は、次の通信までの間に何らかの描画操作を行う可能性が高いと考えられる。例えば、サーバへアクセスする時に、ユーザがカーソルを移動させていた場合は、次の描画のためにカーソルを移動させていると予測できる。このような情報を通信時に同時に送信することにより、他のユーザへ自分が描画中であることを知らせる手がかりとする。具体的には、サーバへのアクセスが発生した時に、次のような状況であると、次のアクセスまでにユーザが描画操作を行う可能性が高い。

- カーソルを移動している時
- 矩形や円 (楕円) の描画モードで始点座標を決定した後で、終点座標を決定する前
- テキスト描画のためにテキストを入力している時

この性質を利用して、通信時のカーソル位置や、キー入力情報を他のユーザに知らせるようにすれば、より円滑に共同作業を行えると考えられる。この仕組みは、複数のアプリケーションを切り替えて使用することが少なく、マウスのようなポインティングデバイスが存在しない携帯電話の特性を活かしたものである。



図 11: 共同作業によって作成した絵

表 2: 絵の構成要素を2人が重複して描画してしまった割合

手本	固定	動的可変
① 家と木	3.25/13 (25%)	2.25/13 (17%)
② 雪だるま	4.75/13 (36%)	3.00/13 (23%)
③ 鳥	3.00/12 (25%)	1.75/12 (14%)
④ 女の子	2.75/18 (15%)	2.75/18 (15%)

平均重複要素数/全要素数 (重複比率)

表 3: アンケート結果 (10点評価)

アンケート項目	固定	動的
相手の描画位置が分かったか?	3.3	6.9
協力して絵を描くことができたか?	4.8	7.5

5 まとめ

携帯電話における共有ホワイトボードシステム CONTENTS とその評価について述べた。CONTENTSでは、キャンパス内容の同期をとる際に、サーバへのアクセス間隔を動的に変更することで、効率的なアクセスを実現している。シミュレーションと実環境での運用テストによる評価より、本システムのアルゴリズムは、クライアント数やユーザの描画頻度が変化しても有効であること、実環境での使用に関しても有効であることが分かった。しかしながら、他のユーザと同じ領域を重複して描画してしまうことが多かったため、今後は共同作業ユーザの状況を把握するための仕組みをさらに検討する必要がある。また、携帯電話以外の端末との相互通信も今後の課題である。

参考文献

- [1] 川口 他: PC と携帯電話の混在環境における電子会議システム, 情報処理学会研究報告, Vol. 2002, pp. 7-12, 2002-GN-42.
- [2] Norman SU 他: Rajicon: A System for Remote PC Access through a Cellular Phone, DICOMO 2001, No. 59, (2001).
- [3] 太田 他: Java 搭載携帯電話における同期式共有ホワイトボード, 情報処理学会第 64 回全国大会, 5A-02 (2002).
- [4] 吉滝 他: CONTE: 携帯電話を利用したお絵描きツール, 情報処理学会研究報告 SST2001-157 (2002).
- [5] 吉滝 他: 携帯電話を利用したお絵描きツールの提案, 電気系学会東海支部連合大会, No. 653 (2001).
- [6] CONTE WebPage, <http://www.mizulab.net/conte/>.