マルチメディア通信と分散処理　59－10
（１９９３．１．２９）

リアルタイムCSCWシステムにおける アプリケーション間データ交換方式

石崎 健史[1]　森 賢二郎[1]　亀田 正美[2]

[1](株)日立製作所システム開発研究所　[2](株)日立製作所ソフトウエア開発本部

複数の参加者がアプリケーションプログラム（AP）を共同実行するリアルタイムCSCWシステムにおいてローカルAPと共同実行APとが共存できる場合，両AP間でデータ交換ができると便利である．ところがローカルAPと共同実行APとの間で，カットアンドペーストなどの手段を用いてデータ交換を行うと，場合によっては参加者の期待に反する結果が得られるという問題がある．リアルタイムCSCWシステムにおいてデータ交換を行う際に生じる種々の問題点を検討し，その一解決案としてAP集中実行方式における親サーバ動的切り替え方式を示す．

Data Exchange between Application Programs in Real-time CSCW Systems

Takeshi ISHIZAKI[1]　Kenjiro MORI[1]　Masami KAMEDA[2]

[1]Systems Development Labolatory, Hitachi, Ltd.
({ishizaki, morike}@sdl.hitachi.co.jp)
1099 Ozenji, Asao-ku, Kawasaki-shi, Kanagawa 215, Japan

[2]Software Development Center, Hitachi Ltd.
1 Horiyamashita, Hadano-shi, Kanagawa 259-13, Japan

When using real-time CSCW (Computer-Supported Cooperative Work) systems, by which more than two participants can see the same displays of application programs, it is convenient if any participant can exchange data between a local application program and a shared application program in both directions. In some cases, a participant might get an unexpected result when he/she tries to cut data in local application program and paste it into shared application program. We focus on various problems in data exchange in real-time CSCW systems, and show a way to solve one of those problems by changing the parent server dynamically.

1. はじめに

ネットワーク接続されたコンピュータを用いてグループによる共同作業を支援するシステム，いわゆるCSCWシステムの研究が活発に行われている．共同作業の形態や，使用するメディアなどにより，さまざまなタイプのプロトタイプシステムが開発されている[1][2]．

筆者らは，複数の共同作業参加者が同一のアプリケーションプログラム（AP）を共同実行し，その出力画面を共有することによって共同作業を支援するリアルタイムCSCWシステムの開発を行ってきている[3][4][5]．本稿では，リアルタイムCSCWシステムにおける，アプリケーション間データ交換の際に生じる問題点を考察する．

現在主流となっているマルチウィンドウシステムでは，複数のウィンドウ上で異なるアプリケーションプログラムを実行させ，ウィンドウ上でのカットアンドペースト操作によって，アプリケーション間でのデータ交換が容易に可能である．これによって，作成したデータを有効に活用し，作業の効率を大幅に改善することができる．

一方，リアルタイムCSCWシステムを使ってアプリケーションプログラムを共同実行している場合について考えると，共同作業中にも，個人作業中と同様にアプリケーションプログラムでのデータ交換が自由にできることが望ましい．たとえば，グループである資料を共同作成しているときに，ある参加者が自分の持っている資料を開いてその中のデータをカットアンドペーストによって共同作成中の資料に組み入れることができると便利である．逆に，共同実行中の資料の一部を切り取って保管しておきたいような場合もあるだろう．

一般に，リアルタイムCSCWシステムはアプリケーションプログラムとウィンドウサーバの間で送受されるメッセージを利用して実現されることが多い．そのため，共同実行アプリケーションプログラムとローカル実行アプリケーションプログラムとの間でデータ交換を行うと，参加者の期待に反する結果が得られる場合がある．参加者は必ずしも，アプリケーションプログラムの実行モデルを熟知しているわけではなく，個人作業時と同じ感覚で操作を行ってしまうからである．

そこで本稿は，リアルタイムCSCWシステムの制御プログラム側に何らかの対策を講じることによって，共同作業中のデータ交換時にユーザの期待している結果が得られるようにすることを目的とする．そのために，共同作業実行時のデータ交換の際に生じる問題について考察し，その一解決法として，親サーバ動的切り替え方式について基本的なアイディアを提案する．

2. リアルタイムCSCWシステム

まず，本稿で対象とするリアルタイムCSCWシステムをモデル化する．

リアルタイムCSCWシステムにおけるAP共同実行方式には，大きく分けて分散実行方式と集中実行方式とがある．

前者は，接続されたすべてのコンピュータ上に同一のAPを配置し，ユーザからの入力のみを同報する方式である（図1(a)）．

後者は，1つのコンピュータ上で実行されるAPの出力をすべてのコンピュータに同報し，逆にユーザからの入力を，APが実行されているコンピュータに集中させる方式である（図1(b)）．

いずれの方式にも一長一短の特徴があるが，本稿では集中実行方式のリアルタイムCSCWシステムに限定して考えることにする．

図2に集中実行型リアルタイムCSCWシステムの実行モデルを示す．

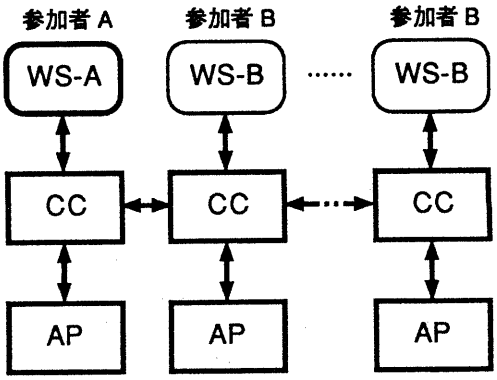

図1(a). 分散実行型システム

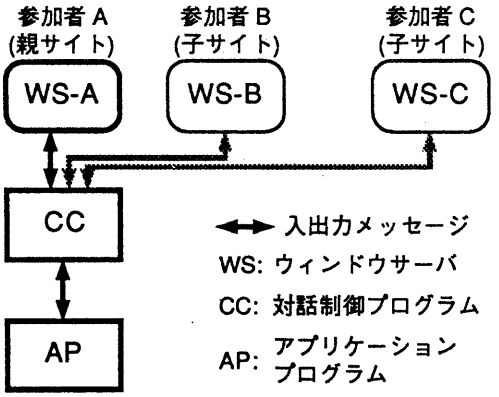

図 1(b). 集中実行型システム

共同実行される AP は，ウィンドウサーバに直接接続するのではなく，CSCW システムが提供する対話制御プログラムに接続する．対話制御プログラムは全参加者のコンピュータのウィンドウサーバと接続する．同時に複数の AP を共同実行するときは，それぞれの共同実行 AP ごとに同様な接続を行う．図では簡単のため共同実行 AP がひとつだけの場合の接続を示す．

AP からのリクエストメッセージは対話制御プログラムによってすべてのサーバに送られる．このとき，リクエストメッセージに含まれるサーバ依存情報（たとえばウィンドウ識別子など）は，送信先サーバに応じて適切な値に変換される．

サーバから AP へは 2 種類のメッセージが送れる．

1 つは，ユーザによるキーボードやマウスの操作に対応して発生するイベントメッセージである．もう 1 つは，AP からの問い合わせリクエストに対するリプライメッセージである．

これらのうち，イベントメッセージについては，すべてのサーバからのメッセージが AP に送られる．このとき，リクエストメッセージと同様にメッセージ中のサーバ依存情報は適切な値に変換される．この結果，接続されているどの参加者からも自由に AP を操作することができる．さらに，必要な場合には，特定の参加者からのイベントだけを AP に渡すように制御することにより，AP の排他的な操作も可能である．このような制御は，操作権制御と呼ばれることもある．

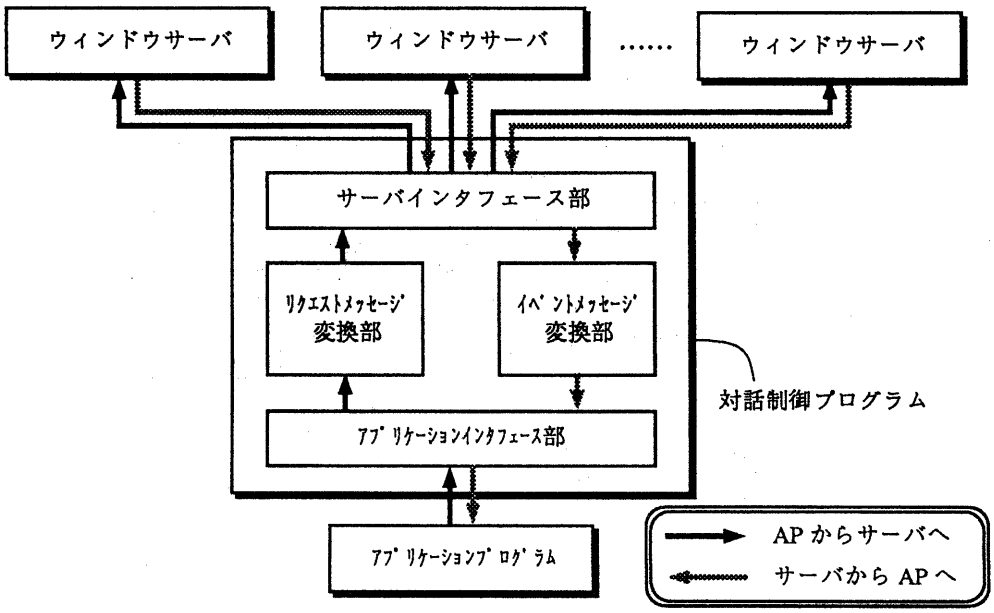

図 2. 対話制御方式

一方，リプライメッセージは各種の資源（ウィンドウなど）の属性など，ウィンドウサーバが管理している情報をAPが問い合わせるときに帰されるメッセージである．共同実行しているAPから見れば，1つのサーバと接続しているのと同じ状態にあるため，1回の問い合わせに対してはただ1つのリプライメッセージが帰されなければならない．しかし，上で述べたように，問い合わせかどうかにかかわらず，すべてのリクエストメッセージは接続されているすべてのサーバに送られるため，1回の問い合わせに対して，サーバの数だけのリプライメッセージが送られてくる．そこで，対話制御プログラムは，特定の1つのサーバからのリプライメッセージのみをAPに送る．他のサーバからのリプライメッセージは対話制御プログラムが破棄する．

この特定のサーバのことを親サーバと呼ぶ．他のサーバのことは子サーバと呼ぶ．通常，APが実行されるコンピュータのウィンドウサーバを親サーバとすることが多いが，他のコンピュータ上のサーバを親サーバとすることももちろん可能である．

3. データ交換と問題点

マルチウィンドウシステムでは異なるウィンドウ上で複数のAPを同時に実行し，さらに，あるAPで作成したデータをコピーして，他のAPにペーストすることによって容易にAP間のデータ交換が可能である．

図3にカットアンドペーストによるAP間データ交換の例を示す．この例では，表計算プログラムなどで作成したグラフをワープロの文書に貼込んでいるところである．

現在主流となっているクライアント・サーバ型のウィンドウシステムにおいては，こうしたAP間のデータ交換を，ウィンドウサーバを経由したアプリケーション間通信によって実現しており，そのためのプロトコルも定められている．本稿では最も基本的なカットバッファによるデータ交換を例に，データ交換の問題点を考える．

図4は2つのAP間でデータを送受するときのメッセージの流れを模式化したものである．データ送信側APはユーザがカット操作を行うと，リクエストメッセージをサーバに送って，サーバが管理するカットバッファ領域にカットされたデータを格納する．データ受信側APでユーザがペースト操作を行うとAPはサーバに対してカットバ

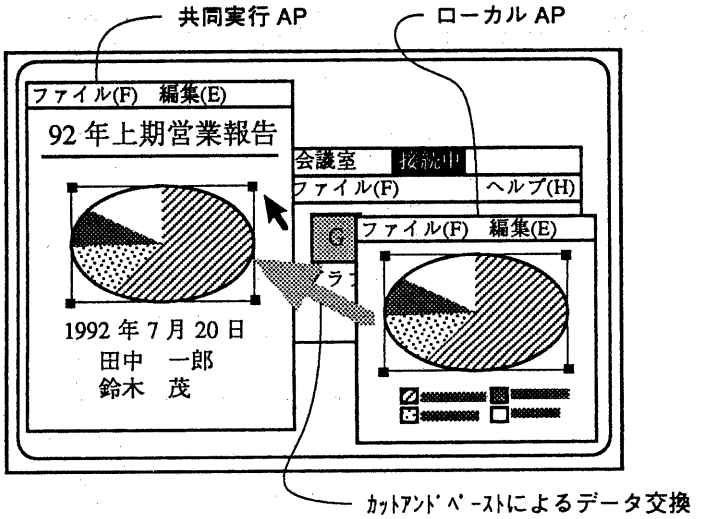

図3. カットアンドペーストの例

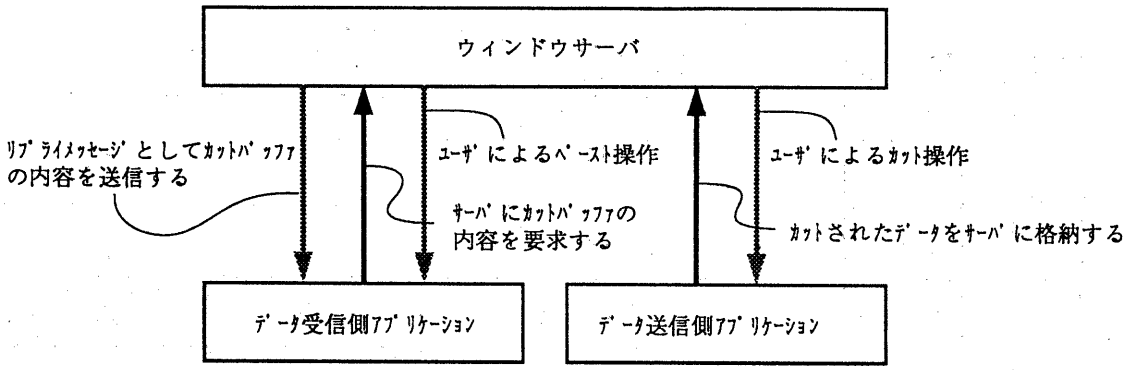

図 4. ｶｯﾄｱﾝﾄﾞﾍﾟｰｽﾄにおけるﾒｯｾｰｼﾞの流れ

ッファ領域のデータを要求する問い合わせリクエストを送る．サーバはデータをリプライメッセージとして AP に送る．

これに対し，リアルタイム CSCW システムによる AP 共同実行時の失敗例を図 5 に示す．この例は，子サイトの参加者がローカル AP でカットしたデータを共同実行 AP にペーストしようとした場合である．

ローカル AP でカットされたデータは子サーバにのみ格納される．つぎに，共同実行 AP でペースト操作を行うと，両方のサーバに問い合わせリクエストメッセージが送られる．すると両方のサーバからリプライメッセージとしてデータが返されるが，子サイトからのリプライは対話制御プログラムが破棄するので，親サーバのカットバッファの内容が共同実行 AP に渡される．この結果，子サイトの参加者は自分の操作に対する期待に反する結果を得ることになる．

どのような場合にこのような不都合が起こるのかを表 1 に示す．

表 1 によれば，共同実行 AP でカットを行った場合にはいずれも問題ない．これは，共同 AP でカットされたデータは全参加者のサーバに送られるためである．

異なるサーバ間でローカル AP どうしのカットアンドペーストを行ったときは問題が生じるが，実際には，他サイトのローカル AP のデータをカットしようとすることは考えられないため，無視

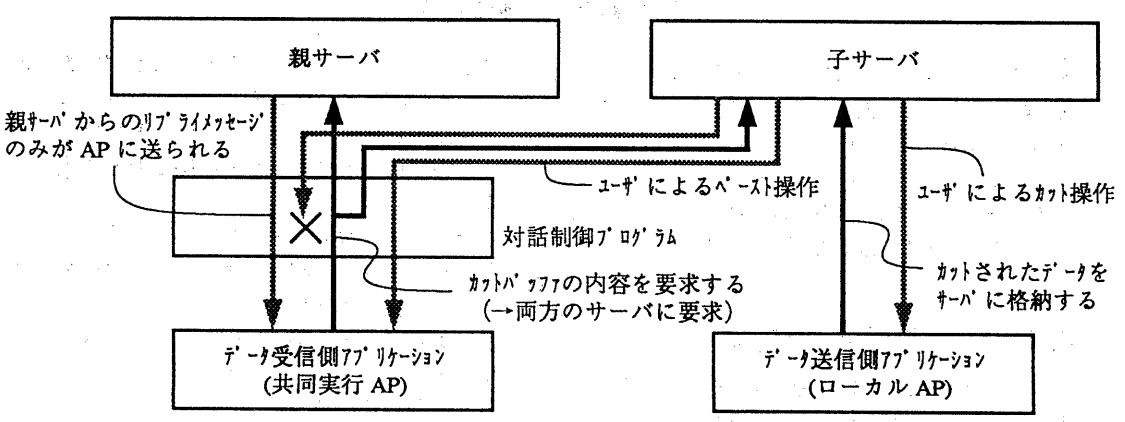

図 5. 共同実行中のﾒｯｾｰｼﾞの流れ

表 1. カットアンドペーストの分類

送信側(カット) ＼ 受信側(ペースト)		共同実行 AP		ローカル AP	
		親サイト	子サイト	親サイト	子サイト
共同実行 AP	親サイト	○	○	○	○
	子サイト	○	○	○	○
ローカル AP	親サイト	○	○	○	×
	子サイト	×	×	×	○

○…成功する
×…失敗する（▒▒…無視可能）

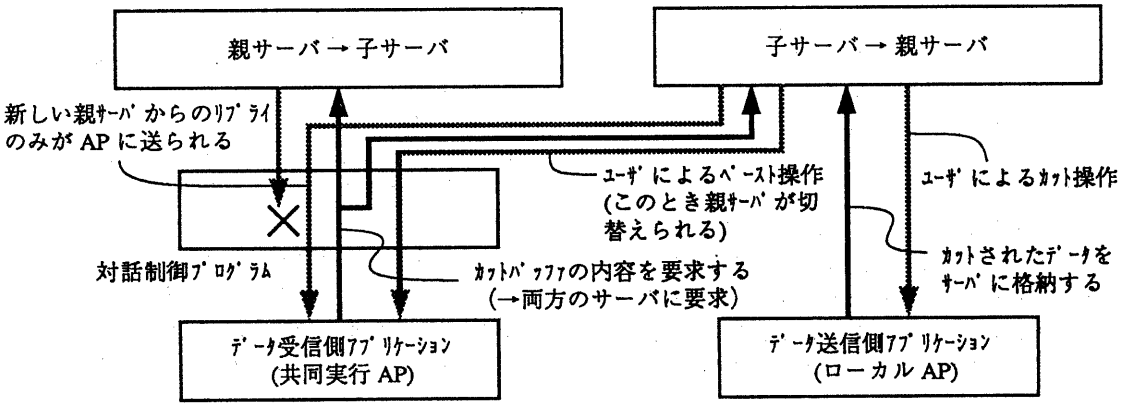

図 6. 親サイト動的切替方式

することができる．同様に子サイトのローカル AP でカットして，親サイトの共同実行 AP でペーストする場合も考えなくてよい．

したがって，子サイトのローカル AP でカットして，子サイトの共同実行 AP でペーストするときの問題に対する対策が必要であることがわかる．

4. 親サーバ動的切替方式

3 章で述べた問題を解決するための方式として親サーバ動的切り替え方式を提案する．

従来，親サーバは起動時に決まる特定のサーバに固定されていたが，本方式では，必要に応じて親サーバを動的に切り替えることができるようにする．本方式を用いると，子サイトでのローカル AP から共同実行 AP へのカットアンドペーストはつぎのようにユーザの期待どおりに実現できる．

まず，ローカル AP でのカット操作によって，子サーバのカットバッファにデータが格納される点までは従来と同様である．つぎに，子サイトの参加者がペースト操作を行ったとき，親サーバを切り替える．すると，カットバッファの内容を要求する問い合わせリクエストに対するリプライメッセージは，今度はカットを行った（元は子サーバであった）サーバからのものが共同実行 AP に

渡される．これはまさに，子サイトの参加者が直前にカットしたデータなので，データ交換が参加者の期待どおりに行われたことになる．

実際にこの方式を適用するための問題点は，共同実行APに対するペースト操作がAPによって異なるという点である．すなわち，ペースト側のAPでユーザがペースト操作を行ったことを対話制御プログラムが検出するには，APによって異なる処理をしなければならなくなる．

したがって，これに代わる方式として，

1）共同実行APに操作権を導入し，操作権のある参加者のサーバを親サーバとする

2）最後に入力が行われたサーバが常に親サーバとなるようにする

などの方式が考えられる．

5. おわりに

リアルタイムCSCWシステムを用いたAP共同実行時に，ローカルAPと共同実行APとの間のデータ交換で生じる問題点について検討した．

解決策として，親サーバ動的切り替え方式を提案した．本方式を用いると，共同作業中にも個人的に利用しているローカルAPを特に意識することなく活用することができる．

今後は本方式を電子対話システムASSOCIA[1]に適用して，さまざまなAPで実験を行い，さらに検討・評価を行っていく．

参考文献

[1] Ahuja, S. R. el al.: The Rapport Multimedia Conferencing System, *Proc. Conf. on Office Information Systems* (1988)

[2] Ohmori, T. et al.: Distributed Cooperative Control for Sharing Applications Based on Multiparty and Multimedia Desktop Conferencing System: MERMAID, *Proc. ICDCS'92* (1992).

[3] 石崎他：多者間電子対話システムASSOCIAにおけるアプリケーションプログラム実行方式の拡張，*第44回情報処理学会全国大会* (1991).

[4] 中山，森他：多者間電子対話システムASSOCIA，*情報処理学会論文誌*，Vol. 32, No. 9 (1991).

[5] Hoshi, T., et al: An Integrated Multimedia Desktop Communication and Collaboration Platform for Broadband ISDN, *Proc. IEEE MULTIMEDIA'92* (1992).