社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2009－DPS－138 (10)
2009－CSEC－ 44 (10)
2009／3／5

Tender におけるネットワーク透過なプロセス間通信機構の設計

佐伯 顕治 † 田端 利宏 † 谷口 秀夫 †
† 岡山大学大学院自然科学研究科

Tender オペレーティングシステムは，独自インタフェースのプロセス間通信機能を実現している．ここでは，プロセス間通信機能にネットワーク透過性を実現し，分散環境を生かしたサービスの実現を可能にする．このプロセス間通信機能にネットワーク透過性を実現するため，分散共有メモリ機能を利用する方式について述べる．具体的には，***Tender*** 独自のプロセス間通信の資源「コンテナ」,「コンテナボックス」及び「イベント」にネットワーク透過性を実現する方式について述べる．さらに，コンテナの送受信について述べ，コンテナの共有モードで分散共有メモリ機能を利用する方式について述べる．

Design of Network-Transparent InterProcess Communication on *Tender*

Kenji Saeki † Toshihiro Tabata † Hideo Taniguchi †
† Graduate School of Natural Science and Technology, Okayama University

The ***Tender*** operating system has original InterProcess Communications. We realize Network-Transparent InterProcess Communication which can use in distributed system. In this paper, We describe Network-Transparent original InterProcess Communication with “container”, “container box” and “event” on ***Tender***. Furthermore,We describe method of send and receive container ,and shared container which used distributed shared memory.

1. はじめに

Tender オペレーティングシステム[1] は，独自インタフェースのプロセス間通信機能を実現している[2]．また，***Tender*** は分散 OS であり，プロセスをネットワーク透過に利用できる[3]．そこで，プロセス間通信機能にネットワーク透過性を実現し，分散環境を生かしたサービスの実現を可能にする．利用者はローカル計算機内でのプロセス間通信と同様に複数計算機間にまたがるプロセス間通信を扱うことができ，他計算機への負荷分散も可能になる．

なお，***Tender*** は，分散共有メモリ機能を実現しており[4]，他計算機のメモリを利用できる．一方，プロセス間通信機能は単一計算機内での共有メモリ機能に相当する機能を持つ．このため，プロセス間通信機能と分散共有メモリ機能を融合した機能の実現が可能である．

そこで，本稿では，プロセス間通信機能にネットワーク透過性を実現する．具体的には，***Tender*** 独自のプロセス間通信の資源「コンテナ」,「コンテナボックス」，および「イベント」にネットワーク透過性を実現する方式について述べる．さらに，コンテナの送受信について述べ，コンテナの共有モードで分散共有メモリ機能を利用する方式について述べる．

2. *Tender* オペレーティングシステム

2.1 資源の分離と独立化

Tender は，プログラム構造を重視し，OS の操作対象を資源として分離し，独立化している．

資源には，図 1 に示す資源識別子と資源名を付与する．資源識別子は，資源の場所と種類と同一種類内の通番を情報として有する数字である．資源名は，場所名と種類と固有名からなる文字列である．場所とは，資源が存在する計算機を指す．場所を示す数字 (以降，計算機番号と呼ぶ)，ならびに名前 (以降，計算機名と呼ぶ) は，システム内の計算機との対応関係とともに，計算機の場所情報表としてシステム内の特定計算機上で管理している．この計算機番号と計算機名の中には，ローカルを示す番号 (0 番)，ならびに名前 (”tender”) を確保しており，これらを用いることで，資源を明示

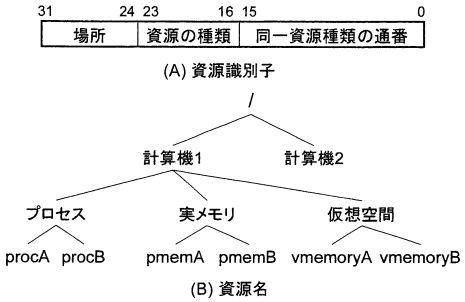

図 1 資源識別子と資源名

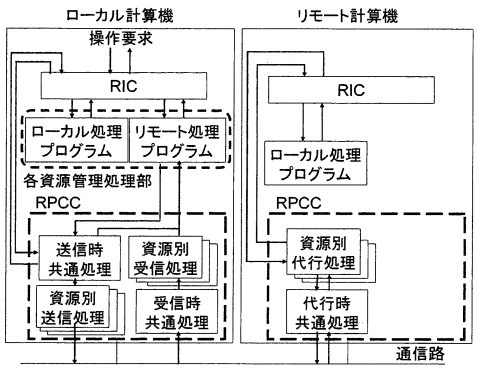

図 2 ***Tender*** における資源操作処理の様子

的にローカル指定できるようにしている．種類については，OS で数字や名前を規定している．同一種類内通番は OS が資源生成時に決定する．固有名は応用プログラムが指定する．

2.2 資源操作方式

Tender における資源操作処理の様子を図 2 に示す．資源操作を行うときは，ローカル計算機の資源インタフェース制御 (以降，RIC: Resource Interface Controller と略す) に処理を依頼する．RIC では，引数の資源名，もしくは資源識別子を調べて，当該の要求が陽にローカルを指定しているのか否かの判別を行う．陽にローカルを指定している場合は，要求された処理を行う資源管理処理部のローカル処理プログラムを呼び出し，その戻り値を返す．逆に，陽にローカルを指定していない場合は，要求された処理を行う資源管理処理部のリモート処理プログラムを呼び出す．リモート処理プログラムは，リモート計算機に対して処理の依頼をするため，遠隔手続呼出制御 (以降，RPCC: Remote Procedure Call Controller と略す) を呼び出す．この RPCC では，共通処理において，引数の資源名，もしくは資源識別子と，計算機の場所情報表との整合をとることにより，処理の依頼先計算機を判別する．この際，依頼先がローカル計算機である場合は，その処理要求をローカルに指定し直して，再度，ローカル計算機の RIC に処理の依頼を行う．逆に，依頼先がリモート計算機である場合は，共通処理と資源別送信処理で処理依頼パケットを生成し，通信路を介して，当該のリモート計算機に対して処理の依頼をする．リモート計算機では，共通処理にてパケット分解し，資源別に代行処理を実行する．代行処理は依頼された処理要求を，陽にローカルを指定するように変更して，自計算機の RIC に依頼する．RIC では，当該の要求は陽にローカルを指定していると判別し，要求された処理を行う資源管理処理部のローカル処理プログラムを呼び出し，その戻り値を返す．この戻り値は，資源別代行処理によって，依頼元のローカル計算機に返される．ローカル計算機では資源別に受信処理を行って，RIC に戻り値を返す．

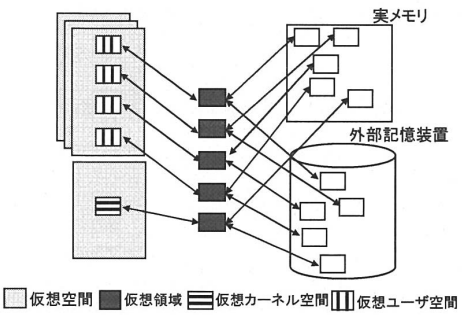

図 3 メモリ管理機構

2.3 メモリ関連の資源

Tender のメモリ管理機構を図 3 に示す．資源「仮想空間」は，仮想アドレスの空間であり，仮想アドレスを実アドレスに変換する変換表に相当する．

資源「仮想領域」は，メモリイメージを仮想化した領域であり，仮想領域の実体は，実メモリ，または外部記憶装置に上に存在する．なお，外部記憶装置上の領域の種類として，資源「永続ユニット」と資源「仮想ユニット」の 2 種類がある．資源「永続ユニット」とは，外部記憶装置上の永続化領域の管理単位である．永続化領域は既存 OS のファイルシステムに相当する．資源「仮想ユニット」とは，外部記憶装置上の仮想化領域の管理単位である．仮想化領域は既存 OS のスワップ領域に相当する．

資源「仮想カーネル空間」や「仮想ユーザ空間」は，資源「仮想空間」と「仮想領域」からなる．この時，

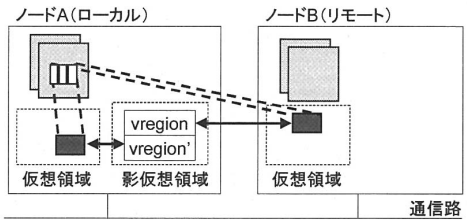

図 4　分散共有メモリの実現方式

前者はカーネルモードのみ，後者はカーネルモードとユーザモードでアクセス可能である．両者は共に，仮想空間が持つアドレス変換表に，当該の仮想領域のデータ格納情報を設定する (以降，貼り付けると呼ぶ) ことで生成することができる．逆に貼り付けた際に設定したデータ格納情報を解放する (以降，剥すと呼ぶ) ことで，削除することができる．

2.4　プロセス間通信

2.4.1　コンテナ

資源「コンテナ」は，プロセスが利用可能なある大きさを持つメモリ空間で，仮想空間に存在する仮想ユーザ空間や仮想カーネル空間を用いて生成され，ユーザモードまたはカーネルモードでアクセス可能である．プロセスは，この資源「コンテナ」をやり取りすることでプロセス間通信を行う．プロセス間での資源「コンテナ」を用いた通信方式として，移動，共有，および複写の 3 つのモードがある．

2.4.2　コンテナボックス

資源「コンテナボックス」は，プロセス間での資源「コンテナ」の受け渡しを仲介する．コンテナボックス管理処理部は，コンテナの送出要求を受けると，送出時に指定されたコンテナボックスのキューの最後に，送出要求のあったコンテナを格納する．コンテナの受信要求があった場合には，コンテナの送出時の貼り付け希望アドレスと受信側の貼り付け希望アドレスを比較し，一致する場合にコンテナを受信プロセスに渡す．

2.4.3　イベント

資源「イベント」は，イベント発生時に資源「イベント」に割り当てられた関数を実行する．また，資源「イベント」に関数が割り当てられていない場合は，資源「イベント」はカウンタセマフォの役割を果たす．

2.5　分散共有メモリ

Tender における分散共有メモリの実現方式を図 4 に示す．分散共有メモリは，遠隔計算機に存在する仮想領域の複製をローカル上に保持する方式で実現している．このローカル上に保持された仮想領域を影仮想領域と呼ぶ．影仮想領域は遠隔計算機上の仮想領域に関連付けられた仮想ユニットからデータを読み出し，仮想ユニットへデータを書き戻す．

なお，***Tender*** では，SC，RC，LRC，HLRC といった既存の一貫性保持方式を幅広くサポートするために，OS 核では必要最低限の以下の機能のみを提供している．

(1) アクセス権の予約，予約解除

(2) 仮想ユニットへの読み出し，書き戻し

これらの機能を OS 核の上位層で組み合わせることで，様々な一貫性保持方式を実現することを提案している[4]．

3.　ネットワーク透過なプロセス間通信の実現

3.1　観　　点

ネットワーク透過なプロセス間通信機能の実現においては，コンテナボックスとコンテナの位置を考慮する必要がある．送信において，プロセスはコンテナを指定して送信を行う．このため，当該コンテナは当該プロセスの仮想空間上に貼り付いていることが好ましい．他仮想空間上に張り付いているコンテナを送信できるためには，コンテナの所有権審査が必要になってしまう．受信においても，コンテナの所有権を明らかにする必要がある．以上のことから，単一計算機内のプロセス間通信機能（ローカル機能）に加え，次の二つの機能を実現する．一つは，送信プロセスの仮想空間上に貼り付いているコンテナを，リモート計算機に存在するコンテナボックスへ送信する機能である．もう一つは，リモート計算機に存在するコンテナボックスから，受信プロセスの仮想空間上にコンテナを受信する機能である．

3.2　資源操作のリモート化

プロセス間通信で利用される資源操作の一覧およびリモート化の対処内容を表 1 に示し，以下に説明する．

○と◎のプログラム部品の違いは，図 2 に示したリモート処理プログラムおよび資源別の送受信処理と代行処理の違いである．○のプログラム部品は，RIC の呼び出し，パケットの生成と分解，および単純な代行処理を行う．これに対し，◎のプログラム部品は，処理内容に合わせた制御処理を行う．これについては，後節で述べる．

×のプログラム部品は，実現が不要なものである．コンテナの送信において，コンテナは送信プロセスの仮想空間上に貼り付いている．したがって，container_send への新たな機能追加はない．コンテナの受信では，リモート計算機に存在するコンテナボックスから，受信プロセスの仮想空間上にコンテナを受信

表 1　プロセス間通信で利用される資源操作の一覧

資源操作名	L	R	送信	代行	受信
cbox_create	-	○	○	○	○
cbox_delete	-	○	○	○	○
cbox_write	-	○	◎	◎	◎
cbox_read	-	○	○	◎	◎
cbox_seek	-	○	○	○	○
cbox_check	-	○	○	○	○
container_create	-	○	○	○	○
container_delete	-	○	○	○	○
container_send	-	×	×	×	×
container_get	△	○	×	×	×
container_unshare	△	○	◎	◎	○
container_check	-	○	○	○	○
container_set	新規	○	○	○	○
event_create	-	○	○	○	○
event_delete	-	○	○	○	○
event_send	-	○	○	○	○
even_receive	-	○	○	○	○
event_attach	-	○	○	○	○
data_copy	新規	○	○	○	○

L ：ローカル，R：リモート，送信：RPCC 送信処理
代行：RPCC 代行処理，受信：RPCC 受信処理
○：パケット生成処理のみ，◎ ：制御処理も含む
- ：実現済みプログラム，新規：新規に実現するプログラム
△：修正の必要なプログラム，×：実現不要なプログラム

する．したがって，コンテナのローカル処理プログラムからコンテナボックスのリモート処理プログラムが呼び出される．このため，container_get の送信/代行/受信での新たな機能追加はない．一方，container_get の処理は，従来，コンテナの存在がローカルに閉じていたため，コンテナの管理情報やデータの参照や更新を直接行っていた．これをリモートに対応させるために，新たに次の三つの機能を実現し，container_get で利用する．一つ目は，ネットワーク透過にコンテナの管理情報を取得できるように container_check をリモート化する．二つ目は，ネットワーク透過にコンテナの管理情報を更新できる container_set を新たに実現する．三つ目は，ネットワーク透過にデータを複写できる data_copy を新たに実現する．

3.3　移動モードと複写モードの送受信

移動モードと複写モードでのコンテナ送信の処理の流れを図 5 に示す．cbox_write の送信処理は，コンテナを貼り付け，送信領域にコンテナのデータを複写する．cbox_write の代行処理は，コンテナを生成し，ローカルのコンテナのデータを複写する．また，コンテナを剥がした後に，コンテナボックスにコンテナを格納する．cbox_write の受信処理は，不要なコンテナを削除する．

移動モードでのコンテナ受信の処理の流れを図 6 に

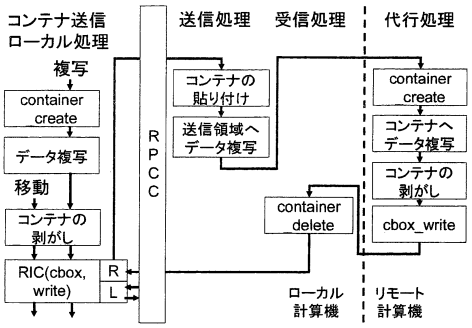

図 5　送信時移動モードと複写モードの処理の流れ

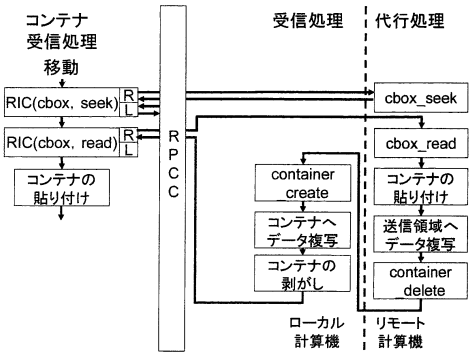

図 6　受信時移動モードの処理の流れ

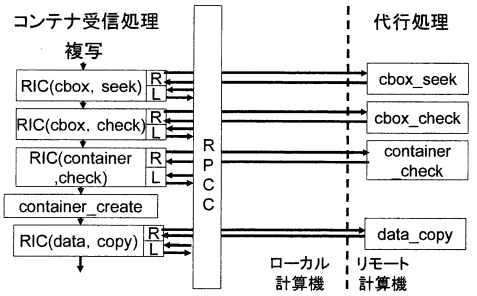

図 7　受信時複写モードの処理の流れ

示す．cbox_seek は指定されたコンテナがコンテナボックスに存在するか確認する．cbox_read の代行処理は，コンテナボックスからコンテナを取り出し，送信領域にコンテナのデータを複写した後に，コンテナを削除する．cbox_read の受信処理は，コンテナを作成し，そのコンテナにデータを複写した後に，コンテナを剥がす．

また，複写モードでのコンテナ受信の処理の流れを図 7 に示す．cbox_check はコンテナボックスから指定のコンテナの情報を取得する．container_check は

コンテナの構成資源の情報を取得する．複写では，移動先のコンテナが作成されてからデータの複写を行うため，data_copy が必要である．

3.4 共有モードの送受信

3.4.1 コンテナ共有モードの一貫性保持

Tender の分散共有メモリは一貫性保持のために必要最低限の機能しか実現していない．このため，コンテナの共有モード時のための一貫性保持方式が必要である．そこで，***Tender*** では，コンテナの共有を行う際に，以下の特徴を持つ一貫性保持方式を採用する．

(1) コンテナボックスのある計算機を Home ノードとする．

(2) 更新の反映はアクセス権の予約解除時に行い，最新版の保持はアクセス権の予約時に情報を確保する．

(3)diff は取らず，単一書き込みとする．

(1)(2) により，通信負荷を減らす．(3) により，計算機処理の増加を抑制する．多重書込みができないため，大きく負荷分散を可能にするとは言えない．しかし，計算機処理の負荷が小さく，通信負荷の低い一貫性保持を可能にする．

一貫性保持を行うにあたり，ローカルの 3 つのプロセス A,B,C がリモート上にあるコンテナを共有する事例を考える．A,B は read only 属性で，C は read/write 属性で共有しているとする．単一書き込みのみを可能としているため，C のみ read/write 属性を持つことができる．この場合，一つの影仮想領域を A と C で共有してしまうと，アクセス権の予約解除以外で更新の反映が発生してしまう．A と B で 1 つの影仮想領域を共有する場合は，A が影仮想領域へ読み出しても B は一貫性保持の観点から影響を受けない．この問題を解決するために，影仮想領域は read only と write/read の 2 つの属性を持つ影仮想領域を作成できることとする．read only の影仮想領域を vregionR とし，read/write の影仮想領域を vregionW とする．vregionR は read only 属性で予約された時に生成され，予約解除時に vregionR の参照数が 0 になる時に削除される．この際，vregionW の参照数が 0 であれば，VregionW も削除する．vregionW は read/write 属性で予約された時に生成され，予約解除時に vregionR の参照数が 0 であれば削除される．

3.4.2 共有モードの送信

リモートに存在するコンテナボックスへコンテナの共有を行う場合の処理の前後関係を図 8 に示す．共有モードでは，コンテナボックスのある計算機を home ノードとするため，読み出しや書き戻しの対象となる

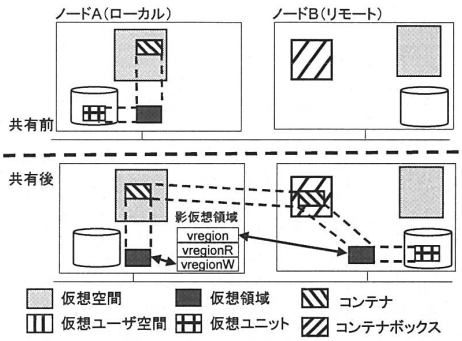

図 8 送信時共有モード

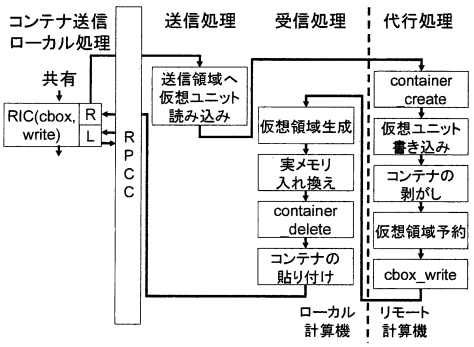

図 9 送信時共有モードの処理の流れ

仮想ユニットのみをリモートへ移動させなければならない．コンテナボックスへ送信されたコンテナの仮想領域は，リモートで利用されていないため，実メモリを持たず，仮想ユニットのみを持つ．また，コンテナを共有モードで送信する際，コンテナは予約した状態，もしくは予約解除した状態でコンテナを送信している．このため，コンテナの送信の前後でコンテナの所持するデータが解放されたり，変わってはいけない．つまり，送信前のコンテナが最初に保持していた内容は，影仮想領域として登録されなければならない．

共有モードでのコンテナ送信の処理の流れを図 9 に示す．cbox_write の送信処理は，仮想ユニットのデータを送信領域へ読み込む．cbox_write の代行処理は，コンテナを生成し，新たに生成したコンテナの持つ仮想ユニットへデータを書き込む．さらに，コンテナを剥がして，仮想領域のアクセス権の予約を行った後に，コンテナをコンテナボックスに格納する．cbox_write の受信処理は影仮想領域となる仮想領域を生成し，実メモリ入れ換えにより，コンテナのデータを仮想領域へ移動させ，コンテナを削除する．また，影仮想領域

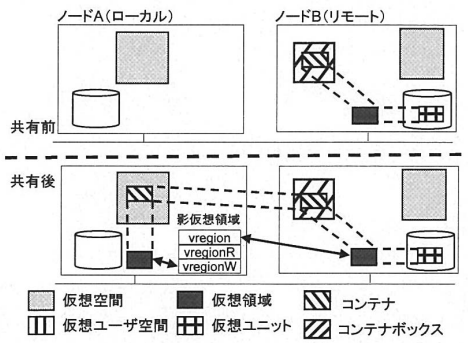

図 10　受信時共有モード

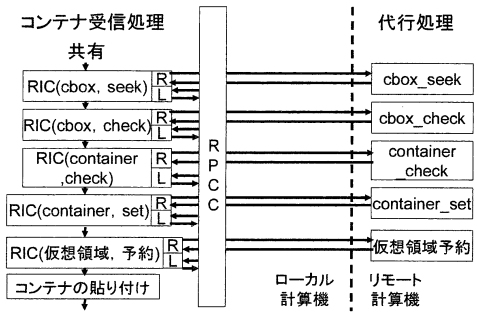

図 11　受信時共有モードの処理の流れ

を空間に貼り付けて利用可能とする．

3.4.3　共有モードの受信

リモートに存在するコンテナボックスからコンテナの共有を行う場合の処理の前後関係を図 10 に示す．影仮想領域の内容はリモートの仮想ユニットから読み込まれる．

共有モードでのコンテナ受信の処理の流れを図 11 に示す．共有モードでのコンテナ受信はリモートに存在するコンテナを特定し，そのコンテナの仮想領域を貼り付ければよい．この処理後に，貼り付けられたコンテナに対し，利用者が読み書きを行うと，ページフォルトを契機として影仮想領域が生成され，リモートのメモリを利用可能になる．共有モードは，他の受信モードと違い，コンテナをローカルに移動させないため，管理情報を更新する container_set が必要である．

3.4.4　共有の解除

コンテナ共有の解除の処理の流れを図 12 に示す．container_unshare の送信処理は，コンテナを剥がし，影仮想領域を削除する．container_unshare の代理処理は，仮想領域の予約解除を行い，リモートのコンテナの管理情報を更新する．

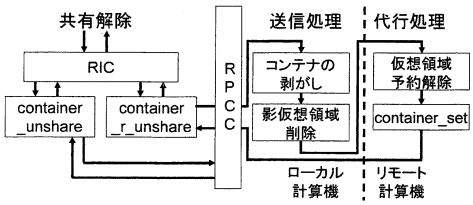

図 12　コンテナ共有の解除の処理の流れ

3.5　各モードの組み合わせ

ローカルに閉じたコンテナの送受信では，各モードを組み合わせることができた．しかし，コンテナの送受信可能な範囲を分散環境に拡張したため，組み合わせることができるモードは移動モードと複写モードのみになった．共有モードと移動モード，共有モードと複写モードは組み合わせることができない．これは移動モードと複写モードの実体が実メモリであるのに対し，共有モードは仮想ユニットであるためである．また，共有されたコンテナを移動すると，第 3 の計算機から当該コンテナを参照できなくなるためである．

4.　おわりに

Tender 独自のプロセス間通信であるコンテナ，コンテナボックス，およびイベントにネットワーク透過性を実現する方法について述べた．具体的には，コンテナの送受信について，各モード毎の実現方式を述べた．また，共有モードでの一貫性保持方式についても述べた．残された課題として，本方式の実現と評価，コンテナの送受信時における共有モードと他のモードの共存がある．

謝辞 本研究の一部は，科学研究費補助金 若手研究 (B)(課題番号：18700030) による．

参　考　文　献

1) 谷口秀夫，青木義則，後藤真孝，村上大介，田端利宏，“資源の独立化機構による ***Tender*** オペレーティングシステム，”情報処理学会論文誌，Vol.41, No.12，pp.3363–3374，2000.
2) 田端利宏，谷口秀夫，“***Tender*** オペレーティングシステムにおけるプロセス間通信機能の実現と評価，”情報処理学会研究報告，Vol.99，No.32, pp.95–100, 1999.
3) 石井陽介，谷口秀夫，“位置透過に利用可能な構成要素を用いたプロセス変身機能，”情報処理学会論文誌， Vol.44, No.7 pp. 1666-1679，2003.
4) 下崎 誠，谷口秀夫，“***Tender*** オペレーティングシステムにおける分散共有メモリの実現と評価，”コンピュータシステム・シンポジウム，pp.161–168, 1999.