資源情報流通サーバ SSS-Server

亀 沢 寛 之　　松 本　　尚　　平 木　　敬

汎用並列オペレーティングシステム SSS-CORE [1][松本,'94] の資源情報流通機構を元に汎用 OS 上に実装した SSS-SERVER について述べる。 SSS-CORE の情報収集機構の特徴として、 (1)eagar な情報収集 (2) 階層的な情報流通機構による効率的な情報収集 / 配布、などがあげられる。本論ではこれらの特徴を基本とした情報流通サーバ、 SSS-Server、の実現のためのアルゴリズム、実装を述べる。 Server として汎用 OS 上に実装することによって異なる OS 間で情報の共有を行うことが可能にし、アプリケーションの開発を支援する。サンプルアプリケーションとして rwho を示す。

Server for gathering remote resource's status SSS-Server

Hiroyuki Kamesawa , Takashi Matsumoto and Kei Hiraki

SSS-Server is shown ,in this paper which is based on information delivering system of SSS-CORE [1][Matsumoto,94],generally purposed parerell operationg system, and implemented on generary used operating system.Information delivering system of SSS-CORE has characteristics as (1) eagar information gathering (2) efficient information delivering with stratified structture of network system. This paper shows algorythms and implementation of infromation delivering server named SSS-Server ,which shares those characteristics.As is implemented on generary used operaring system,SSS-Server enables hosts with various operating systems to share infromation, and supports development of applications with remote resources.And "rwho" is shown as a sample.

1. 背　　景

コンピュータ環境のネットワーク化の進展に伴い、非ローカルな資源を有効に利用して効率良く動作可能なアプリケーション / 環境への要求が高まっている [2],3]。(前者には、並列計算アプリケーション、並列 make, 後者には NFS,RPC など) 特にネットワーク上の資源ステータス情報の効率的な収集は資源利用を効率良く行なうという観点において非常に重要である。平木研究室では、非ローカルの資源情報をユーザアプリケーションに低コストで提供する機能を持つオペレーティングシステム (OS) として、汎用並列オペレーティングシステム,SSS-CORE を研究している。 SSS-CORE は (1) あらかじめ収集したネットワーク上の資源情報及びスケジューリング情報をユーザに低コストで開示する、 (2) ユーザに OS の資源割り当てに対して要求を行なうことを許す、という機能の実現を目的として開発されている。本論で示される SSS-Server は、 SSS-CORE の資源情報流通に関する部分をサーバとして独立し、汎用 OS 上に実現するものであり、 SSS-CORE と資源情報流通機構における基本概念を共有する。また、 OS によらないサーバとして実装することによって異なる OS 間での情報共有を可能とし、アプリケーション開発の枠を広げるものである。

SSS-CORE の情報収集機構の特徴として (1)eagar な情報収集よるユーザアプリケーションへのレスポンス向上 (2) 階層的な情報流通機構による効率が良い情報収集ということを挙げることができる。 SSS-Server は上記特徴 (1),(2) を実現し、汎用 OS 上で単体のソフトウェアとして動作する。 SSS-Server は (1) あらかじめ集めておいたネットワーク情報をユーザに低コストで開示 (2) ユーザに自由な流通データの作成を許す。 (3) パケットの流れを整理しネットワークへの負化を低く抑えるといった特徴を持つネットワーク情報流通サーバである。本報告における SSS-Server は SunOS-4.1.3 上に実装され、更に、サンプルアプリケーションとして rwho を実装した。

2. 既存のサーバ (rwhod,ewhod) について

現在、ネットワーク上のホスト情報を流通させるシステムとしては、 rwhod とその改良版である ewhod が

† 東京大学情報科学科平木研究室
Hiraki laboratory at Information science dept. of Tokyo Univ.

広く用いられている。☆ ewhod は、今までの比較的遅いネットワーク (10Base-T) 上では正常に動作していたが、高速なネットワーク (100Base-T) 上ではうまく動作しない。その原因として (1)ewhod は、ネットワークを越えるパケットを一気に流す,(2) データの圧縮・パケットのコンパクト化を行なわないのでパケット量が多い,(3) ewhod は排他制御を行なわず一定の時間経過によってパケットを流す, といった点が挙げられる。SSS-Server は上記 3 点を踏まえて設計されており、問題点はとり除かれている。

3. SSS-Server の基本動作

3.1 SSS-Server の機能

SSS-Server は、同一ネットワーク内の他の SSS-Server と互いに排他制御をしながらローカルなホスト情報を自分の属するサブネットワークにブロードキャストする。データパケットを受けとった SSS-Server はそれを共有メモリ上に格納する。サブネットワークを越える通信は TCP を用いて行なわれ、得られた外部ネットワークの情報はローカルネットワーク内にブロードキャストされる。 SSS-Server はコンフィグ情報及び NIS を用いてネットワークに疑似的な階層性を与え、パケットの流通経路を形成する。この流通経路は、同一のデータパケットが同じ SSS-Server に 2 度以上受けとられないことを保証する。また、異常が発生した際には、柔軟にネットワーク構造を再構成してネットワーク構造を維持する。

3.2 SSS-Server の階層化

SSS-Server はネットワーク全体で互いに協調してその機能を果たすが、パケットの流通経路を一本化しパケット流量を抑えるために、そのネットワーク構造に階層性を導入している。 SSS-Server の階層化は、各サブネット (ネットマスク) 単位で行なわれ、各階層間はゲートウェイと呼ばれる SSS-Server によって互いにデータを交換する。この通信には TCP が用いられる。
各階層には「親」サーバが存在し階層構造の維持を担当する。また、「親」以外の SSS-Server は「子」と呼ばれる。また、これらの中からゲートウェイが「親」によって選択され、他のネットワークとの通信を担当する(図**1**)。「親」は各階層に唯一つ存在し、新規加入サーバの初期化、他のネットワークとの交渉によるゲートウェイの配置、稼働しているゲートウェイの稼働チェックを行なう。「子」は、データを受けとり、ローカルデータをブロードキャストする。

☆ ewhod は rwhod に (1)NFS クライアントの負担軽減 (2) ネットワークを越えたデータのやりとり (3) 情報の流量制御 の機能拡張を施したもの

3.3 Lamport の論理クロック [4],5]

パケットの再送などによる混乱を避けるために、SSS-Server では Lamport の論理クロックを変形したものを用いている。 Lamport の論理クロックはネットワーク上の各 event に対して唯一のタイムスタンプを与えるアルゴリズムであり、 SSS-Server では 各サーバの変数 clock とパケットに押される timestamp によってネットワークに単一のクロックを作る。 SSS-Server は、 timestamp と clock を以下のように利用する。

(1) timestamp $<$ clock ならパケットを棄却
(2) timestamp $=$ clock かつ「パケットの送出元が直前のパケットと同じ」なら受理
(3) timestamp $<$ clock なら受理、 clock $=$ timestamp とする。

また、 timestamp $= 0$ なら無条件で受理する。☆☆
SSS-Server は、この論理クロックによって階層内でのイベントの時間的整合性を保つ。☆☆☆

3.4 「親」SSS-Server

整合性を維持するため SSS-Server では、その全体を通して 1 つのコンフィグファイルしか許していない。このため、新しく起動する SSS-Server はすでに起動している他の SSS-Server からデータを受けとって自分自身を初期化する。また、ネットワーク間の情報流通システムの状態をチェックし必要に応じてネットワーク構造の最構築を行ない、全体の構造を維持することも必要である。各階層の「親」SSS-Server はこれらのことを行なうために各階層に 1 つだけ設定される。通常、その階層で最初に立ち上がった SSS-Server がその階層の「親」になるが、もしこの「親」が何らかの原因で止まってしまった場合以下のアルゴリズムによって新しい「親」を決定する。各 SSS-Server にはそれぞれ一意に決まる番号が振られているものとする。

(1) 「親」の異常を検出した「子」SSS-Server は、自分の名前を記録した ELECTION メッセージを発行して次の SSS-Server に渡す。
(2) ELECTION メッセージを受けとった SSS-Server はメッセージの名前の優先順位と自分の優先順位とを比較し、優先順位の高い側の名前を記録した ELECTION メッセージを次のサーバに回す。
(3) ELECTION メッセージの名前が自分の名前と一致したら自分を「親」にする。自分の名前を記した COODINATOR メッセージ発行してを次の SSS-Server に渡す。
(4) COORDINATOR メッセージを受けとった

☆☆ ネットワーク外との通信、初期化前のサーバは同期機構外となっている。

☆☆☆ 例えば、ネットワーク上でのイベント同期が実現可能なグループ通信用ツールキットとしてコーネル大学の Isis システム [6]~8] が挙げられる。 (SSS-Server はグループ通信システムでなく、情報流通サーバを目的とする。)

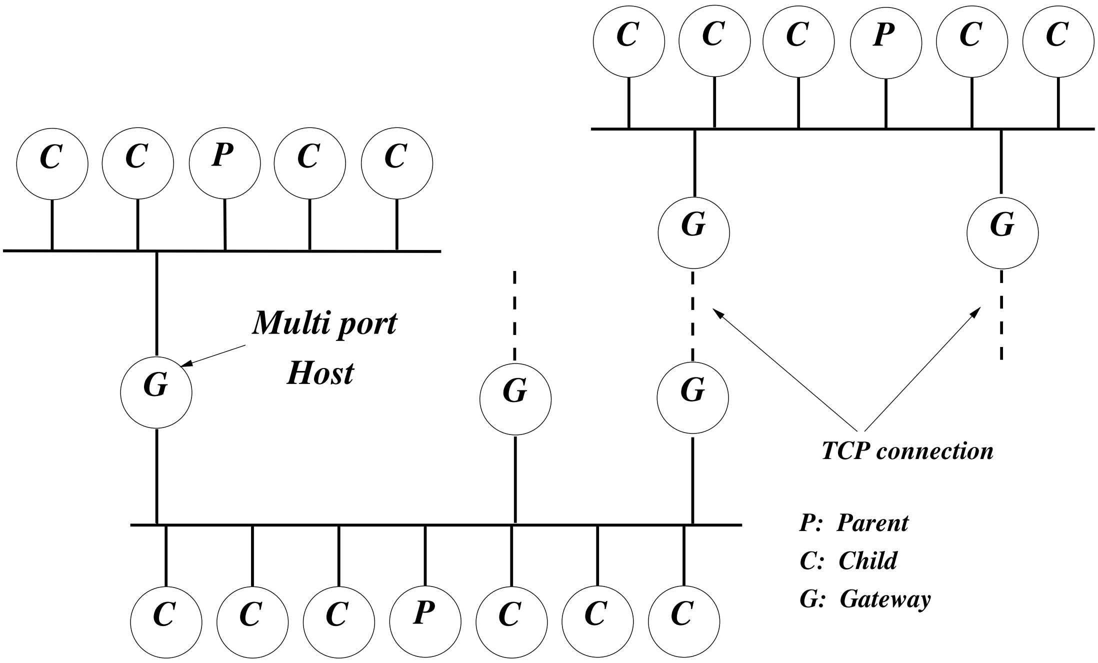

図1　SSS-Server の階層構成
Fig. 1　Structure of SSS-Servers

SSS-Server は親を設定する。メッセージの名前が自分と一致した場合、アルゴリズムを終了する。

このアルゴリズムは 1 対 1 通信のみで親を決定可能なので TCP を用いて信頼性のある親の決定ができる。

3.5　SSS-Server の排他制御

SSS-Server は、各階層内でパケット送信時の排他制御を実現するために以下のようなアルゴリズムを採用している。

(1) 最初に、SSS-Server は BOOKING(予約) メッセージをブロードキャストとして発行する。timestamp は 0 とする。予約メッセージには自分の名前を記載しておく。
(2) BOOKING メッセージを受けとった SSS-Server は予約キューに予約を enqueue する。
(3) 予約キューの先頭が自分自身ならば送信フェイズに入る。送信フェイズを終了したら RELEASE メッセージをブロードキャストする。RELEASE メッセージには使用された予約の timestamp と自分の名前を記載しておく。
(4) RELEASE メッセージを受けとった SSS-Server は、timestamp が一致する予約までを消去し、メッセージの発行者とそのタイムスタンプで予約キューの末尾に予約を付け加える。
(5) 一定時間 RELEASE が発行されなければ予約キューを POP する。

このアルゴリズムは集中型排他制御と異なり単一故障点を持たない。また、トークンリング・アルゴリズムのようにトークン紛失の検出が必要ではないのでネットワーク全体がストールしてしまう時間が短いという利点を持つ☆。また、論理クロックによる管理が、パケットの紛失に対してのこのアルゴリズムの耐性を保証する。

3.6　ネットワーク構造の形成

SSS-Server はデータ流通経路の一元化のため、ネットワークに仮想的な階層構造を与える。階層構造の決定は各階層を点、階層間のゲートウェイを枝としたグラフ上で重みつき最小木決定アルゴリズムを用いて行なわれる。

各枝の重みは各ゲートウェイのレベル表**1**によって与えられる。ネットワーク間のゲートウェイはコンフィグで明示的に指示することもできるが、階層間にゲートウェイがなかった場合はレベルを ADD にして適当なゲートウェイを付け加える。ゲートウェイがマルチポートを利用するならばレベルを ALIAS にする。ALIAS のゲイトウェイは互いにパケットを運ぶ必要はなく、共有メモリ上で情報を共有する。また、一つのホストにゲートウェイが集中することはない。DEFAULT のゲートウェイにはユーザの設定した重み、ALIAS には 3、ADD には 1 が与えられ、重みが最大になるように最小木が決定される。

この構造の決定は (1) 新しい階層がネットワークに加わ

☆ ethernet なら token を回した場合よりも必要なパケット量が減少する。

レベル	意味
DEFAULT	コンフィグで望ましいとされる。
ALIAS	マルチポートマシン。[☆1]
ADD	自動生成された。

表1　ゲートウェイのレベル
Table 1　level of gateways

る、(2) ある階層全体が応答しなくなる、たびに行なわれる。

3.7　交渉によるネットワーク構造の維持

ある階層とある階層の間のゲートウェイが使用不能になった場合、それぞれの「親」の間で "交渉" が行なわれ、使用不能になったものの代わりに適当な(使用可能な)ゲートウェイが割り振られる。交渉の結果は「親」から各々の階層にブロードキャストされる。交渉はなるべくネットワーク構造全体を変化させないように行なわれるが、「ある階層が死んでしまった場合」などには構造全体が変化する。この場合、交渉の結果は全てのネットワークに伝えられる。
この階層構造データの配信は「親」の送信フェイズを利用して行なう。

3.8　パケットのタイプ

以上のアルゴリズムに対する必要性から、SSS-Server の発行するパケットのタイプには表2のようものが定義されている。
REPLY を除けば UDP を使用するパケットには全てブロードキャストが用いられている。「親」のみが、DETECT PARENT,NEGOTIATE 及び INITIALIZE に返答する。

4. SSS-Server の実装

4.1　SSS-Server の構成

SSS-Server は、基本となる 3 つのプロセスと 1 M の共有メモリから構成される。また、TCP による通信を受信する際にはプロセスが生成される。3 つのプロセスは各々 (1)Receiver (2)Storer (3)Sender と呼ばれているが、(1)Receiver が他の 2 つを統括する方式で動作する[☆2]。3 つのプロセスに分割する利点は (a) 自分が出したパケットを受けとれる[☆3]。(b) 作業の分割によって、Receiver は常にパケット待ちの状態を維持できるという点が挙げられる。また、各プロセス間の通信はメッセージ・システムコールを用いて行なっている。[☆4]共有メモリの大きさは主に SunOS4.1.3 で一般ユーザが一度にとれる最大の共有メモリの大きさによっている。また、BUDDY システムをメモリ管理に用い、1 セグメント 256byte である[☆5]。セグメントサイズはデフォルトの rwho 用のデータのみならば通常 256byte 以内に収まる、メモリの alignment が良いということから用いている。また、指定されたマルチポートマシン上では、RECEIVER,SENDER がそのポートの数だけ起動し、共有メモリ上のデータを互いに共有する。このようなホストは LEVEL が ALIAS のゲートウェイとして自動的に登録される。図2 はこのようなマシン上での SSS-Server の構成を示している。

4.2　アクセスの制限

最初に起動される SSS-Server はコンフィグファイルを読み込み、SSS-Server に参加する予定のすべてのネットワークとホスト名を手に入れ、ネットワーク構造を構成する。コンフィグにないホストからのアクセスは受け付けない。SSS-Server はユーザに対して自由な情報の作成を許している。ユーザは、データを作成し SSS-Server に渡すことでサーバの動いているネットワーク全体でそのデータを受けとることができる。[☆6]デフォルトでは TTY の状況、workload(rwho フォーマットで) を運んでいる。

4.3　パケットの欠落

Fast-Ether(100base-T) 上では、rwhod はかなり頻繁にパケットの欠落を起こす。
本学科[☆7]で使用されているネットワーク上での実験の結果を表3に示す。
簡単な実験ではあるが、この実験の結果、速いネットワーク上でも同一のネットワーク上ならほとんどパケットを落さないが、他のネットワークとの通信や、遅いネットワークを用いる場合にはウェイトをかけてやるのが良いとわかる。
これらの実験結果は比較的ホストの負荷が軽い場合に軽いプログラムによって得られたものであるから、受け取り手が負荷の重い処理を行なう場合は必ずしも表は参考にならないと考えられる。
SSS-Server は、パケットの送出には 1 パケットあたり 1ms のウェイトをかけてある。

4.4　データ圧縮の効果

SSS-Server では、Lempel-Ziv のアルゴリズム [9] によってデータの圧縮を行なっている[☆8]。デフォルトで流通させている rwho 用のデータは、1 ホストあたり最大 1.3kbyte だが、圧縮によって 1/4 程度にまで小さくなる。各ゲートウェイはパケット数を最小にするようパッケージングするのでネットワークを越える際の 1 階層あたりのパケット量は 1 階層のホスト数の 4 分の 1 程度になる。これは、ewhod などがホスト数と同数の

[☆2] (2),(3) は通常眠っていることになる。
[☆3] 論理クロックの更新に必要
[☆4] (1) ソケットの使用数を極力抑える。(2) 厳密な同期は必要ではない。(3) メッセージはソケット、パイプに比べて割り込みに対処しやすい。といった点でメッセージを採用している
[☆5] 4096 個のセグメントが確保され、メモリ管理のために共有メモリ中の 16kbytes が割り当てられる。
[☆6] 現在、データの流出入制御はない。
[☆7] 東京大学理学部 6 号館
[☆8] 変更の可能性は高い。

タイプ名	目的	プロトコル
DATA	データパケット	UDP/TCP
DETECT PARENT	「親」を探す。	UDP
INITIALIZE	初期化情報を「親」に要求する。	TCP
NEGOTIATE	ゲートウェイを決定するための交渉を行なう	TCP
GATECONF	ネットワーク構造の変化を伝える	UDP
BOOKING	パケット送信権の予約を行なう	UDP
RELEASE	パケット送信の終了を伝える。	UDP
ELECTION	新しい親の選挙を行なう	TCP
COORDINATOR	新しい親の決定を伝える	TCP
ASK ACTIVE	ACTIVE かどうかチェックする。	TCP
REPLY	パケットが REPLY であることを示す。	UDP

表2 SSS-Server のパケットタイプ

Table 2 Types of SSS-Server's packets

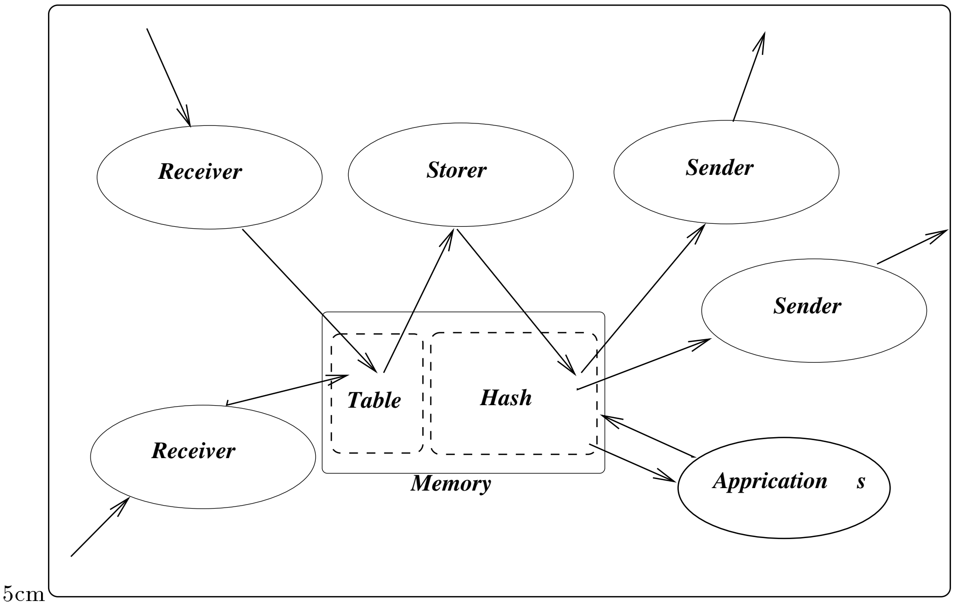

図2 Multi port マシン上の SSS-Server の構成

Fig. 2 Structure of a SSS-Server on multi port machine

Wait	10Mbps	100Mbps	2nets(fast to fast)
0	52.2	0.0	53.3
1usec	0.0	0.0	0.0
10usec	0.0	0.3	0.3

表3 ネットワークとパケットの欠落

Table 3 loss packet on various network

- UDP を用いた実験。パケットのサイズは 1300byte として 100 パケットずつ１０回送信された。
- １パケット毎のウェイトタイムを増やしていきながら、実験を行なった。数字は欠落したパケットの百分率。
- 左から 10Mbps,100Mbps,
  ネットワークを越える場合 (100Mbps → 100bps)

パケットを必要としていたことと比べるとかなりネットワークの負担を軽くする。(1) １階層当たり４０ホスト (2) 全部で１０のネットワークを接続 (3) １分で送信フェイズが１週する (4)rwho 用のデータのみとすると、$(40*9)/4+40=130$ パケットの DATA パケットが１階層内にブロードキャストされる。これに、１ホスト毎に RELEASE パケットがあるので この様な条件では約 170 パケットが１分間にブロードキャストされる事になる。(3) の値はコンフィグによって調整可能なので各々のネットワークに適した値を設定しなければならない。

4.5 外部アプリケーションとの通信

SSS-Server と外部アプリケーションとの通信は、共有メモリを介して行なわれる。

ユーザアプリケーションは書き込みたいデータに LABEL を割り当ててデータを書き込み、ホスト名と

LABEL を指定してデータを読み出すことができる☆。
ただし、ユーザアプリケーションによる共有メモリ上への直接的な malloc は提供されない。ユーザやグループによる認証は行なわないので、データの保護に関してはユーザは自分自身で責任を持たなければならない。
また、データは SSS-Server のコンフィグで示された全ての範囲に届けられる。
すなわち、 SSS-Server はデータの流通のみを保証する。
各ホストのデータはネットワーク毎のハッシュテーブル上に格納される。

4.6 排他制御に関する問題点の回避

SSS-Server は起動時に 有効な RELEASE パケットがエントリを消去するまで「予約」をとり続け、その後 BOOKING パケットを送信して SSS-Server の環の中に入る。 RELEASE パケットが一定時間受け取られなかった場合、予約キューは POP されるが、「次」の予約を持つもののタイムアウト時間を小さく設定することで混乱を避けている。一定時間送信フェイズに入れなかった SSS-Server は自分の予約キューをチェックし自分のエントリを探す。もしエントリがなければ BOOKING パケットを配信する。

4.7 Sample application “ rwho “

SSS-Server の流通する各ホストの TTY STATUS のデータを用いる rwho(以下 SSS-rwho) を作成した。
通常の rwho は、 rwhod が書き込む whod.hostname というファイルを読み込んで動作するが、 SSS-rwho は、 SSS-Server が共有メモリ上に作成したデータベースからデータを取り出してそれを開示する。 SSS-rwho は、メモリ上のデータベースからデータを取り出すため、 rwho とは違い、ネットワーク単位にソートするなどの処理を (NIS を介さずに) 行なうことができる。

5. ま と め

ネットワーク上に情報を流通させるサーバ SSS-Server を開発した。 SSS-Server は既存のサーバ、 ewhod などと比べて次のような特徴を持っている。

- 協調動作による排他制御
- UDP 送信時の ウェイトによる確実性の向上。
- TCP によるネットワーク間のパケット交換
- ネットワークの構造化によるパケット流通経路の一元化
- データ圧縮によるパケット量の減少

これらの特徴によって SSS-Server はネットワークへの負荷とパケットを落す確率を減少する。更に、動的なネットワーク構造の再構成によりシステム全体のクラッシュを避けている。共有メモリによるデータアクセスを可能にし、ユーザアプリケーションに扱いやすいデータを提供する。

謝辞 様々な助言を与えてくれた平木研究室のメンバー、実験に協力して下さった情報科学科の皆さんに感謝の意を表します。

参 考 文 献

1) 松本尚, 古荘進一, 平木敬: 汎用超並列オペレーティングシステム SSS–CORE, 日本ソフトウェア科学会第 11 回大会論文集, pp. 13–16 (1994).
2) Theimer, M. M., Lantz, K. A. and Cheriton, D. R.: Preemptable Remote Execution Facilities for the V-System, *Proceedings of the Tenth ACM Symposium on Operating System Principles*, pp. 2–12 (1985).
3) Nichols, D. A.: Using Idle Workstations in a Shared Computing Environment, *Proceeding Eleventh Symp. on Operating System Principles, ACM*, pp. 5–12 (1987).
4) Lamport, L.: A New Solution to Dijkstra’s Concurrent Programing Problem, *Communications of the ACM*, Vol. 17, pp. 453–455 (1974).
5) Lamport, L.: Time,Clocks,and the Ordering of Events in a Distributed System, *Communications of the ACM*, Vol. 21, pp. 558–564 (1978).
6) Birman, K. P. and Joseph, T. A.: Reliable Communication in the Presence of Failures,, *ACM Transactions on Computer Systems*, Vol. 5, pp. 44–76 (1987).
7) Birman, K. P. and Joseph, T. A.: Exploiting Virtual Synchrony in Distributed System, *Proc. Eleventh Symp. on Operating Systems Principles ACM*, pp. 123–128 (1987).
8) Birman, K. P., Schiper, A. and Stephenson, P.: Lightweight Casual and Atomic Grup Multicast, *ACM Transactions on Computer Systems*, Vol. 9, pp. 272–314 (1991).
9) Ziv, J. and Lempel, A.: A Universal Algorithm for Sequential Data Compression, *IEEE Transactions on Information Theory*, Vol. 23, pp. 337–343 (1977).

☆ LABEL=”1” は rwho data 用に割り当てられる。