通信網ソフトウェア分散処理プラットフォームのカーネル

久保田 稔† 丸山勝己† 田中 聡†

通信網ソフトウェア用分散処理プラットフォーム PLATINA の基本ソフトウェアである実時間分散カーネル（PLATINA カーネル）について述べる．PLATINA は全国規模のパーソナル移動通信等の高度な通信網サービスの実行環境を提供するソフトウェア基盤であり，高い処理能力，機能，信頼性が必要とされる．PLATINAは，並行オブジェクト指向に基づくアプリケーションプログラムの実行環境を提供するPLATINA カーネル，これらのアプリケーションに共通的に必要となる基本機能を提供するサーバからなる．本論文では，PLATINA カーネルの機能，構造およびその性能の評価について述べる．PLATINA カーネルは，複数の論理アドレス空間をサポートし，並行オブジェクトを実現するためのスレッドの実行を制御する．PLATINA カーネルは，分散処理をサポートするとともに，多重度が非常に高いシステムにおける実時間処理要求条件に対応している．さらに論理空間を活用することにより性能と信頼性のバランスを図り，処理時間／所要メモリ量の両方の観点から効率化している．PLATINA カーネルは実時間システムとしての性能を損なうことなくプログラムの信頼性・拡張性を向上させる．今後の発展が予想される通信網ワイドの高度通信処理システムのプラットフォームの構成要素として有効である．

A Kernel of a Distributed Processing Platform for Telecommunication Network Software

Minoru Kubota,† Katsumi Maruyama† and Satoshi Tanaka†

PLATINA is a distributed processing platform for telecommunication software. It provides an execution environment for advanced telecommunication services like nation-wide personal mobile communications, and achieves advanced functionality, high performance and reliability. The PLATINA consists of the real-time kernel, called the PLATINA kernel, and multiple servers. The kernel controls the servers and application programs both of which are based on a concurrent object-oriented model, and the servers provide the common fundamental facilities for the applications, that is, telecommunication service software. In this paper, we describe facilities, structure, and the performance evaluation of the PLATINA kernel. It supports multiple logical memory space management, and controls the execution of threads used to implement concurrent objects. It also supports distributed processing and satisfies the requirements of highly multiple processing systems. Using the logical memory space management, it provides a good balance between performance and reliability, and reduces the processing overhead and the amount of memory required to executed the applications.

1. はじめに

今後，パーソナル移動通信のように，公衆通信網ワイドの分散処理を必要とする通信サービスが広く普及することが予想される．また通信網に対して，従来以上に多種多様なサービスを提供することが求められるであろう．従来，通信網ノード上のソフトウェアは独立に開発され，別個の処理モデルに基づき，各サービスに応じて通信プロトコルを使い分けていた．このため任意のノードで動作可能なソフトウェアを実現することは難しく，拡張性や維持管理性も十分でなかった．さらに各ノードにおけるソフトウェアが分散処理により協調して通信サービスを実現するための機能と性能をもつ基盤ソフトウェアがなかった．

したがって，(1)通信網のソフトウェアのための基本的な処理モデルと，(2)ソフトウェアを効率的に実行するための環境，を提供するソフトウェア基盤の必要性が高まってきた．このソフトウェア基盤を分散処理プラットフォームと呼び，以下の要求条件を満たすことが求められる．

多様なサービスの迅速な導入を可能とするために通信網ソフトウェアの拡張性と保守性の向上が求められている．通信網ソフトウェアは，どのノードでも変更

† NTT ネットワークサービスシステム研究所
NTT Network Service Systems Laboratories

なく動作可能で，どのノードにあるかにかかわらず互いに通信可能でなければならない．分散処理プラットフォームには分散透過性を実現し，通信網におけるソフトウェアバックプレーンの役割を果たすことが求められる．

通信網ソフトウェアは，厳しい実時間条件を満たさなければならず，また同時に多数の処理を行う必要がある．たとえば，代表的な通信網ソフトウェアである交換プログラムでは，同時に数千の呼を扱い，発呼や切断といった多くのイベントに関し，数 10 ミリ秒以内に処理しなければならない．このように厳しい実時間処理要求条件を満たし，高いスループットを達成するため，分散処理プラットフォームには高い実行効率をもつことが要求される．

分散処理プラットフォームは，通信サービスを中断なく提供できるように，信頼性が高く，安全で，頑健である必要がある．多様な要求にこたえるため，基本的な通信サービスのほかに新しいサービスプログラムの導入が通信網ノードで増加している．新たに導入されたソフトウェアが基本サービスに擾乱を与えないよう基本サービスソフトウェアを保護する必要がある．

通信網における分散処理プラットフォームとして既存の分散 OS を適用することも考えられるが，処理効率，高信頼性に関する上記の要求条件を満たすことは難しいと考える．我々は PLATINA (Platform for Telecommunication and Information Network Applications) と呼ぶ分散処理プラットフォームの検討を進めてきた[1),2)]．**図1** に示すように，PLATINA は実時間分散カーネル（以後，PLATINA カーネルと呼ぶ）とサーバからなる．PLATINA とその上で動作するアプリケーションをあわせて PLATINA システムと呼び，通信網の各ノードで動作する．PLATINA で動作するアプリケーションの例としては交換プログラム等がある．

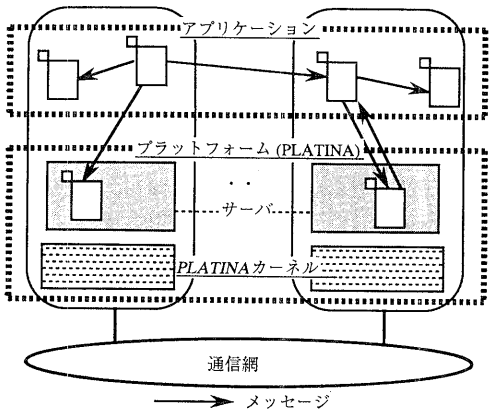

図 1 PLATINA システム
Fig. 1 PLATINA system.

PLATINA は，並行オブジェクト指向処理モデルに基づくアプリケーションプログラムの実行をサポートし，それ自身も並行オブジェクト指向モデルにより作られている．また分散環境においてアプリケーションに対して一様な環境を提供する．たとえばアプリケーションにおける位置透過性をサポートする．これによりノードの特定の物理的な属性に依存せずアプリケーションを開発することができる．これらはソフトウェアの拡張性と保守性の向上を狙いとしている．

最大限の効率と信頼性を同時に達成することは不可能であるため，PLATINA はこれらの最適な平衡点を提供することを目的とする．すなわち，実時間 OS 相当の性能をできる限り維持したまま，汎用 OS 並のプログラム保護機構を実現することを目的としている．

本論文では，PLATINA の概要，PLATINA カーネルの機能，実現法，評価，について述べる．PLATINA カーネルは実時間処理のサポートが主要な目的の一つであるが，以下に述べるような点において，従来の実時間 OS とは異なる．組み込みシステムへの適用を主たる用途とする従来の実時間 OS[3)~6)] は，効率を最優先にしており，論理メモリ空間やカーネルレベルでの分散処理がサポートされていないが，PLATINA カーネルではこれらをサポートする．

一方，代表的な分散処理 OS として，V[7)]，Mach[8)]，Chorus[9)] などがあり，マイクロカーネルを基本として，軽量プロセス（スレッド），分散透過性，実時間処理をサポートする等の点において，PLATINA カーネルにこれらの OS と構造が共通している部分が多い．しかし PLATINA カーネルは，通信網ノードへの適用を前提としていること，また並行オブジェクト指向モデルのカーネルレベルでのサポートを目的とすることから，以下に示すようにこれらの OS にはない特徴をもつ．

通信網ノードにおける処理は極めて多重度が高いこと（数千～数万），またこの多重処理を行ったまま実時間処理を行わなければならないことから，PLATINA カーネルには，所要メモリの削減，高性能，が要求される．このため後述するように，オブジェクトを実現するスレッドの機能を必要最小限にとどめスレッド制

御の処理を高速化し，ユーザ開発のスレッドもカーネルモードで実行可能とすること，等により高性能化をはかっている．またスレッドの周期起床処理，非同期のメッセージ通信とプロトコル階層の単純化，等により，高スループット化をはかっている．分散処理OSをベースに実時間処理機能を導入する試みもあるが[10]，実時間対応スケジューリングを主要な目的としており，高いスループットを達成することについては十分な考慮はされていない．またPLATINAカーネルでは，アプリケーションスレッドにおけるカーネルスタックの共用，長期にわたって中断するスレッドのスタックの解放機能のサポート等により，所要メモリの削減を図っている．

並行オブジェクト指向モデルをカーネルレベルからサポートするため，PLATINAカーネルはスレッドにインスタンス変数を割り付ける機能をサポートしている．これは他OSにない特徴である．これにより，並行オブジェクト指向言語に，より密着した実行環境を提供しており，並行オブジェクト指向言語[11]の実行環境の実装にも用いられている．

2. 通信網ソフトウェア向き分散処理プラットフォーム PLATINA

2.1 処理モデル

2.1.1 並行オブジェクト指向モデル

PLATINAシステムは並行オブジェクト指向モデルに基づいており，その構成要素はオブジェクトとして扱われる．オブジェクトはメッセージ通信によってのみ相互作用し，この制限によりソフトウェアの独立性が高まり，ソフトウェアの保守性と拡張性が向上する．

オブジェクトは独立した自律的な実体であるので，通信網内の各ノードに分散して配置することは容易である．PLATINAでは，メッセージを送るべきオブジェクトを指定するためにグローバルオブジェクト識別子（OID）を用いる．OIDは通信網で一意に定義され，これによりオブジェクトの位置にかかわらず，同じ方法で相手のオブジェクトを指定することができる．

各オブジェクトは識別子のほかに名前を持つことができる．オブジェクトの名前はソースプログラム中に陽に記述される論理的なキー情報であり，通常，文字列として表現される．動的に生成されたオブジェクトについては，システムの状態により識別子の値は変わるが，静的に生成されるオブジェクトはあらかじめ定義された識別子をもつことができる．オブジェクトの名前にはオブジェクトの位置情報が含まれないので，名前だけが分かっているオブジェクトにメッセージを送るためには，オブジェクトの名前をオブジェクト識別子に変換する必要がある．この変換は後述するように名前サーバが行う．

通信網ソフトウェアでは多くのイベントを同時に扱わなければならず，このためオブジェクトは並行に動作できなければならない．たとえば交換プログラムにおける呼制御オブジェクトは，それぞれが呼に対応し，複数の呼を並行して処理する[12]．並行に実行されるオブジェクトを能動オブジェクトと呼ぶ．

すべてのオブジェクトを能動オブジェクトとすることで処理モデルを簡単にできる．能動オブジェクトはプロセスとして実現できるが，プロセスのコンテクスト・スイッチとプロセス間通信はオーバヘッドが大きいので，能動オブジェクトの実行オーバヘッドが大きくなってしまい，PLATINAが想定する性能要求条件を満たすことができない．したがってPLATINAでは，能動オブジェクトの環境下で実行され，オーバヘッドの少ない受動オブジェクトを導入している．受動オブジェクトへのメッセージ通信はプロシージャ呼び出しと等価である．受動オブジェクトは，リモートオブジェクトからメッセージを受けることはできない．

メッセージを送るオブジェクトを送信オブジェクトと呼び，メッセージを受けるオブジェクトを受信オブジェクトと呼ぶ．オブジェクトが存在するノードをローカルノード，他のノードはリモートノードと呼ぶ．同様にローカルノードにあるオブジェクトはローカルオブジェクト，リモートノードにあるオブジェクトはリモートオブジェクトと呼ぶ．

あるメッセージを送った時，それに対する受信オブジェクトの処理の完了を待たずに送信オブジェクトが処理を再開できる場合，そのメッセージを並列メッセージと呼ぶ．並列メッセージの受信オブジェクトは能動オブジェクトでなければならない．もし送信オブジェクトが，受信オブジェクトが送られたメッセージを処理し制御を戻すまで実行を中断する時，このメッセージを直列メッセージと呼ぶ．

オブジェクトの存在期間に関し，永続オブジェクト（persistent object）と動的オブジェクト（dynamic object）の2つのタイプのオブジェクトがある．永続

オブジェクトはシステムの初期設定時に生成され，システムが運用中実行しつづける．動的オブジェクトは必要に応じて生成され，不要になれば削除される．

2.1.2 オブジェクト間通信モデル

従来のコンピュータネットワークにおける分散アプリケーションはクライアントサーバモデルに基づいていることが多い．このモデルではクライアントとサーバが対等でない役割を持ち，並列に走行しない場合が多い．クライアントは，要求メッセージを送ると，サーバが応答を返すまで処理を中断しなければならない．これは処理遅延を増大させ，スループットを低下させる．クライアントサーバモデルにおいて複数のサーバを並列に動作させることにより，並列性を高めることができるが，要求メッセージごとにサーバを生成し，生成されたサーバが1つずつメッセージを処理しなければならない．通常，サーバオブジェクトを生成するコストは高く，多くのサーバがあるとシステムのソフトウェアリソースが無駄に消費されるため，所要メモリおよび性能に関するオーバヘッドが大きい．したがってクライアントサーバモデルだけでは，通信網ソフトウェアのような実時間システムの基本メッセージ通信モデルとして不十分である．

PLATINA では，互いに対等な関係にある複数のオブジェクトが，非同期のメッセージ通信を行うモデルをとる．本モデルは，多くの通信パラダイムに適用可能であり，高い効率とスループットを実現できる．たとえば交換プログラムにおける Caller-Callee モデル[12]に基づく交換サービスの記述に適している．Caller オブジェクトと Callee オブジェクトは互いにメッセージをやりとりしながら呼を制御する．すでに述べたように2つのオブジェクトは異なるノード上にあってもよく，これにより分散交換システムを容易に構築することができる[2]．

2.2 PLATINA の構造

PLATINA を構成する要素について述べる．

2.2.1 プロセッサ

PLATINA が実行されるプロセッサは，**図 2** に示すように3つのアドレス空間と2つの実行モードをサポートし，実時間処理に必須となるタイマ割り込み機能をもつことを想定する．プロセッサは論理メモリアドレス管理機構をもち，論理メモリアドレス空間として，アプリケーション空間，共有空間，カーネル空間，をサポートするものとする．各ノードには複数のアプリケーション空間が存在しうるが，共有空間とカーネル空間は1つずつしかない．アプリケーションプログラムはアプリケーション空間に，共有されるメッセージバッファは共有空間に，また PLATINA カーネルはカーネル空間にそれぞれ割り付けられる．このメモリマップにより，論理空間の不当な操作からプログラムを保護する．性能を優先するならば，実メモリシステムで PLATINA を実行させることも可能である．

PLATINA が実行されるプロセッサは実行モードとして，アプリケーションモードとカーネルモードをもつ．PLATINA カーネルは，カーネルモードで走行し，任意のメモリにアクセスできる．アプリケーションはアプリケーションモードで走行し，それが割り付けられているアプリケーション空間と共有空間にアクセスできるが，カーネル空間に直接アクセスすることはできない．これにより，PLATINA の基本プログラムであるカーネルをアプリケーションの障害から保護する．

PLATINA が実行されるプロセッサは，論理メモリアドレス空間と実行モードを活用することにより，効率を大きく損なうことなくプログラムの保護機能を高めることができる．代表的な 32 ビットマイクロプロセッサ[13)~15)]で PLATINA を実行させることができる．

2.2.2 PLATINA カーネル

PLATINA システムにおいて，各ノードは PLATINA カーネルを1つずつ持つ．カーネルは割り込み処理のようにハードウェア特有の処理を扱い，後述するスレッド（能動オブジェクト）の実行をスケジュールし，論理メモリアドレス空間を管理する．PLATINA カーネルの詳細については，後述する．

2.2.3 PLATINA サーバ

PLATINA サーバはアプリケーションに対し基本的なサービスを提供するオブジェクトである．PLAT-

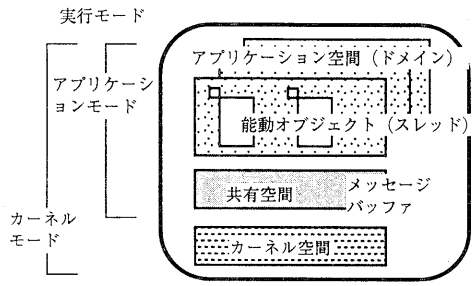

図 2 実行モードとメモリ空間
Fig. 2 Execution modes and memory spaces.

INA カーネルは最低限の機能のみを提供し，汎用 OS ではカーネルがサポートしていた多くの機能（例えばファイルシステム）は，PLATINA ではサーバにより実現される．例えばファイルサーバ，メッセージ通信サーバ，名前サーバ等がある．

同一プログラムのサーバはカーネル空間とアプリケーション空間いずれでも実行可能である．特にカーネル空間にあって，従来の OS の機能を提供するサーバをカーネルサーバと呼ぶ．サーバを置き換えたり，新しいサーバを組み込むことにより，PLATINA の新しい機能を容易に組み込むことができる．例えばユーザが開発した入出力ドライバをサーバとして容易に組み込むことができる．

サーバは，能動オブジェクトであり，通常永続オブジェクトである．サーバは，送信オブジェクトから要求メッセージを受け取り，それに応じた処理を行い，必要ならば応答を送信オブジェクトに返す．

2.2.4 ドメインとスレッド

従来のコンピュータシステムは，プロセス機構により並列性を実現している．プロセスは実行の単位であり，1つの論理空間が割り当てられる．能動オブジェクトをプロセスで実現し，各能動オブジェクトに論理空間を割り当てれば，オブジェクトの独立性と信頼性が高まる．しかし，通信網ソフトウェアにおいては，能動オブジェクトは小規模で，通常数百から数千個同時に存在するために，能動オブジェクトごとに論理空間を割り付けると，論理空間の切り替えを伴うコンテクスト・スイッチによるオーバヘッドが極めて大きくなる．したがって PLATINA では，並列実行の単位とプログラムの実行環境を与える論理空間は区別する．

図2に示すように，個々のアプリケーション空間をドメインと呼び，ドメイン内の並列実行単位（能動オブジェクト）はスレッドと呼ぶ．スレッドは並列処理の実行単位でありドメインのアドレス空間を共用する．ドメインは，スレッドを実行するために必要なリソースを提供する環境である．能動オブジェクトはスレッドにより実現され，1つ以上のドメインがノードに割り付けられる．1つのドメイン内に複数のスレッドすなわち能動オブジェクトがあり，ドメインは1つの仮想ノードとみなすことができる．ドメインは他のノードに移動させることができる．

各アプリケーションをそれぞれドメインに割り付けることにより，あるアプリケーションを他のアプリケーションからの不当なアクセスから保護することができる．さらに，ドメインは他のドメインに影響を与えずに生成することができるので，新しいアプリケーションをドメインとして容易に追加することができる．新しいプログラムは，既存プログラムのオブジェクトの OID を知ることができれば，既存プログラムにアクセスすることができる．これはソフトウェアの拡張性を向上させる．

3. PLATINA カーネルの機能と実現手法

PLATINA カーネルの基本機能をカーネルプリミティブと呼ぶ．主要なカーネルプリミティブを**表1**に示す．アプリケーションはカーネルプリミティブを起動するインタフェースを通して，カーネルの機能を利用することができる．

3.1 スレッド

3.1.1 カーネルスレッドとアプリケーションスレッド

PLATINA カーネルが制御するスレッドは，スレッド制御ブロック，スタック，インスタンス変数領域からなる．スタックはスレッドが実行するプログラムの作業用領域である．インスタンス変数領域は，スレッドの生成時に指定されたサイズの領域が割り付けられるもので，能動オブジェクトのインスタンス変数が格納される．サイズを0とすればインスタンス変数領域は割り付けられない．

図3に示すように，スレッドにはカーネルスレッド（Kスレッドと略記）とアプリケーションスレッド（Aスレッドと略記）の2つがある．Kスレッドはカーネル空間にスタックを持ち，カーネルモードで実行される．したがって，通常のプロシージャ呼び出しによりカーネルプリミティブを起動することができる．Kスレッドは通常 PLATINA カーネルのサーバとして用いられる．

表1 カーネルプリミティブ名
Table 1 Examples of kernel primitives.

プリミティブ名	説　明
thd_create	スレッドの生成
thd_stop	スレッドの終了
mbf_alloc	メッセージバッファ割付
mbf_free	メッセージバッファ解放
msg_send	メッセージ送信
msg_receive	メッセージ受信
msg_reply	応答メッセージの返送

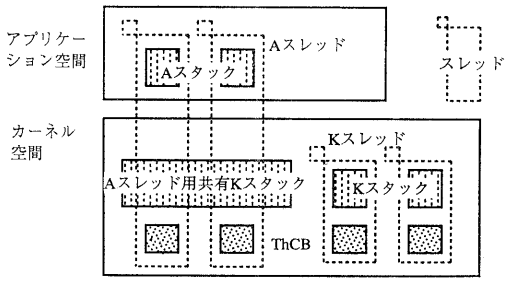

図 3 スレッドの実現
Fig. 3 Thread implementation.

Aスレッドはスタックをアプリケーション空間に持ち，アプリケーションモードで実行される．Aスレッドのスタックを Aスタックと呼ぶ．Aスレッドでは，前述のスレッド単位に割り付けられたスタックに加え，カーネルモードで使用するスタックが必要となる．これをKスタックと呼ぶ．従来の OS では，並列実行単位であるプロセスは，AスタックとKスタックの両方を持っていた．しかし PLATINA では，スレッドの数が膨大になるため，カーネルプリミティブの呼び出しによりAスレッドがカーネルモードで実行される場合，Aスレッドに共通のKスタックを用いる．Kスレッドは個別のKスタックをもつ．

スレッドの実行を中断する場合，カーネルは，スレッドの実行の再開に必要な情報を退避する必要がある．この情報をコンテクストと呼び，レジスタの内容，実行再開番地等が含まれる．カーネルプリミティブが実行されるカーネルモードで，中断するAスレッドのコンテクストを共有されるKスタック上に退避すると，他のAスレッドに退避したコンテクストを破壊されるおそれがある．

PLATINA カーネルは，Aスレッドがカーネル内で中断する場合にコンテクストを含む再開に必要なデータをスレッド制御ブロックに退避し，再開時にこれを戻す．再開に必要な後処理はカーネルプリミティブの種類により異なる．Aスレッドがカーネル内で中断するのはメッセージの受信等の場合に限られ，それぞれの場合に対応した後処理を用意している．中断する可能性のあるカーネルプリミティブごとに再開処理プログラムを用意する方式によりKスタックをAスレッド間で共有することが可能になる．

能動オブジェクトを実現するスレッドがメッセージ受信待ちで中断する場合，そのスレッドのスタックを解放できる場合がある．これは，(1)スレッド固有のデータあるいは履歴情報がインスタンス変数領域に，また再開に必要な情報がスレッド制御ブロックにあり，(2)再開処理あるいは再開後に必要な情報がスタック上にない，場合である．

PLATINA カーネルはユーザの指示により，スタックを一時的に解放する機能を提供する．これは交換プログラムにおける通話中の呼を扱う能動オブジェクトに適用すると，必要なスタック数を 1/10 以下にすることが可能でメモリ削減に効果的である．一時的に解放されたスタックは，メッセージ受信時に再割り付けされる．

高性能が要求される場合，アプリケーションプログラムを変更することなくカーネルモードで実行することも可能である．すなわち頻繁に起動されるAスレッドは，Kスレッドとすることでオーバヘッドを削減することができる．たとえば，交換プログラムにおいて基本サービスの呼制御オブジェクトをKスレッドとして実現することにより実行効率を高めることができる．

3.1.2 スケジューリング

通信ソフトウェアの実時間処理要求を満たすため，PLATINA カーネルは優先度に基づくスケジューラとラウンドロビンスケジューリングを併用する．PLATINA が実行されるプロセッサは周期的に（たとえば 10 ミリ秒）割り込みを発生し，これにより PLATINA カーネルは実行中のスレッドの実行を中断し，より高い優先度をもつスレッドを実行する．PLATINA カーネルは周期的スケジューリングと，一定時間内にイベントが発生しないことを検出するためのタイムアウト機能をサポートする．周期的スケジューリング機能により，スレッドを周期的に起動することができる．周期的に起動されるスレッドは，交換機における加入者線の状態変化のようなイベントを，一定時間内に処理する．この方式は処理すべきイベントが頻繁に発生する場合，割り込み処理に伴うオーバヘッドの削減に効果がある．

実行中のスレッド T1からスレッド T2に切り替える時，T1と T2が異なるドメインに属していると論理アドレス空間の切り替えが必要となり，実行オーバヘッドの増大となる．このため，次に実行可能なスレッドで同一優先度のものが複数ある時は，直前に実行中であったスレッドと同一ドメインのスレッドを優先的に実行することにより，実行オーバヘッドの増大を防止している．

3.2 スレッド間通信

3.2.1 スレッド識別子

スレッドはメッセージ通信により情報を交換する．メッセージを送るため，スレッドは宛先のスレッドの識別子と送るべき情報を格納したメッセージバッファをパラメタとして，メッセージ送信カーネルプリミティブを起動する．

メッセージ送信先を指定するため，各スレッドはグローバル識別子をもつ．これは2.1.1で述べた能動オブジェクトの OID に相当し，通信網で一意に定義され，スレッドの存在するノードの情報とそのノード内で該スレッドを識別するのに必要な情報を含む．

PLATINA システムでは，スレッドが通信網のどこにあっても同一のカーネルプリミティブを用いてメッセージを送ることができる．メッセージを送るカーネルプリミティブプログラムは，まずスレッド識別子を調べ，メッセージがどこに送られるべきかを決定する．受信スレッドが送信スレッドと同一ノードにある場合，PLATINA カーネルはそのメッセージを受信スレッドのメッセージキューに追加する．受信スレッドが送信スレッドとは別のノードにある場合，PLATINA カーネルは受信スレッドがあるノードの PLATINA カーネルにメッセージを送る．

3.2.2 メッセージ種別

PLATINA カーネルは通常メッセージと緊急メッセージの2種類のメッセージをサポートする．通常メッセージはさらに要求メッセージ，応答メッセージに分類される．要求メッセージは，受信スレッドに処理を要求するためのメッセージで，応答メッセージは，その要求に対する処理に関し必要なら返すメッセージである．緊急メッセージは，例外的なイベントを通知するために用いられる．

3.2.3 ノード内通信

スレッド間通信において，送信側から受信側にメッセージの内容をコピーするのはオーバヘッドとなる．**図4**に示すように，PLATINA では，並列メッセージ用のメッセージバッファは，共有空間に割り付けられる．PLATINA カーネルではメッセージ通信において，メッセージバッファのアドレスのみを送るので，コピーによるオーバヘッドを削減できる．

上記のようにメッセージバッファを共有メモリ空間に割り付ける方法では，メッセージバッファに対する不当なアクセスを防止することができない．このため PLATINA では共有メッセージバッファへのアクセスは，ハードウェアによってサポートされるケーパビリティ[16),17)]を用いて行うことを想定している．

ケーパビリティはバッファのアドレスとアクセス権からなり，バッファにアクセスする唯一の方法である．スレッドはケーパビリティがなければメッセージにアクセスできないので，送信側のオブジェクトは，メッセージのケーパビリティを解放し，受信側スレッドがそれを獲得する．メッセージには1つのスレッドだけがアクセスできることを保証することにより，このメカニズムはメッセージ通信における信頼性と効率を向上させる．

3.2.4 ノード間通信

ノード間通信では，高いスループットと短い遅延時間が必要である．通信網ソフトウェアにおいてメッセージは，ファイル転送の場合のような連続した処理とは異なり，多くは要求とその応答のように1つずつ処理される．ノード間で送受される個々のメッセージについて，コネクション設定等のノード間通信を行うための前処理を行っているとオーバヘッドが大きい．したがってノード間通信ではあらかじめコネクションを設定しておき，個々のオブジェクト間のメッセージを多重化して送受する．

図5に示すように，PLATINA におけるノード間通信プロトコルは3つの階層からなる．最下位の階層は物理ネットワーク階層であり，OSI モデルのレイヤ1に相当する．2番目の階層は，ノード間通信レイヤと呼び，ほぼ OSI モデルのレイヤ2から4に相当する．最上位のレイヤは，オブジェクト間通信レイヤと呼ぶ．ノード間通信レイヤはオブジェクト間通信レイ

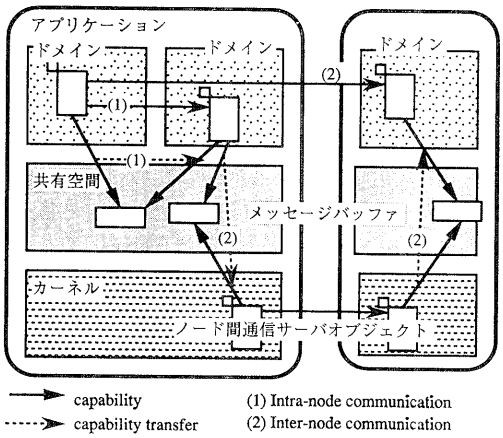

図4 メッセージバッファ
Fig. 4 Message buffer.

ヤのメッセージを多重化し，メッセージの到着を確認し，もし紛失を検出すれば再送することにより，オブジェクト間通信レイヤに対し信頼性の高い効率的な通信機能を提供する．

3.2.5 入出力制御

PLATINA は並行オブジェクト指向モデルに基づいているため，割り込みハンドラを含む入出力ドライバと入出力制御プログラム間のインタフェースとして，メッセージ通信を用いている．入出力ドライバは入力されたデータをメッセージとして，入出力制御プログラムを実行するスレッドに送る．また出力データはメッセージとして入出力ドライバに送られる．入出力ドライバは基本的な入出力処理のみを行うため，入出力ドライバ内で必要となる排他制御のための割り込み禁止区間を短くできるため，実時間応答性を高めることができる．

3.3 カーネルプリミティブ

PLATINA カーネルは同期，非同期の2種類のカーネルプリミティブをサポートする．スレッドが同期カーネルプリミティブを実行すると，カーネルプリミティブが処理を完了するまで，他のスレッドは実行されない．非同期カーネルプリミティブは，システムの実行状態により呼び出しスレッドの実行を中断させる場合があり，その場合，PLATINA カーネルのスケジューラは別のスレッドを起動する．

Aスレッドは，実行モードをカーネルモードにするトラップ命令を使って，カーネルプリミティブを呼び出す．トラップハンドラは，レジタスを実行中のスレッドのアプリケーションスタックに退避し，レジスタあるいは共有メモリ空間を経由してアプリケーション空間から，カーネル空間にパラメタをコピーする．

図6に示すように，アプリケーションは，(1)カーネルプリミティブを直接呼び出す，(2)カーネルサーバにメッセージを送る，のいずれかにより PLATINA カーネルのサービスを利用することができる．(2)で処理結果が必要な場合，呼び出しスレッドは，返答メッセージを待つ．

カーネルサーバを追加したり，更新したりすることで，カーネルの機能の追加・更新を容易にかつ柔軟に行うことができる．また異なるノードにあるカーネルサーバでもそのOIDを知っていれば，メッセージを送ることにより，そのサーバを利用することができる．

KスレッドとAスレッドの両方から同じインタフェースでカーネルプリミティブを呼び出すことを可能にするため，それぞれのスレッドに対応して，PLATINA カーネルはカーネルアダプタ，アプリケーションアダプタの2つのインターフェースプログラムを持つ．アダプタプログラムは，割り込みマスク，パラメタ引き継ぎ，を行い，起動されたカーネルプリミティブが非同期プリミティブの場合は，起動したスレッドのコンテクストを退避する．またカーネルサーバを起動する場合には，メッセージバッファを捕捉し，これにカーネルプリミティブのパラメタをコピーして，カーネルサーバスレッドに送る．

KスレッドとAスレッドは同じカーネルプリミティブを用いるので，同じプログラムをドメイン（アプリケーションモード）とカーネル（カーネルモード）で実行することができる．ドメイン内でプログラムが実行された場合，生成されるスレッドはAスレッドとな

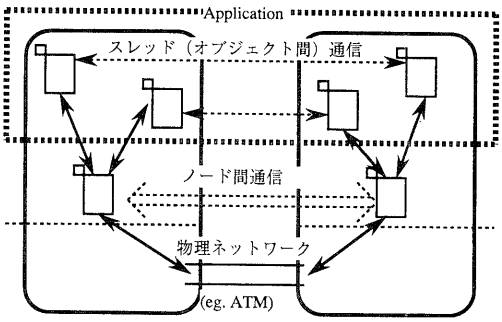

図5 プロトコル階層
Fig. 5 Protocol layer.

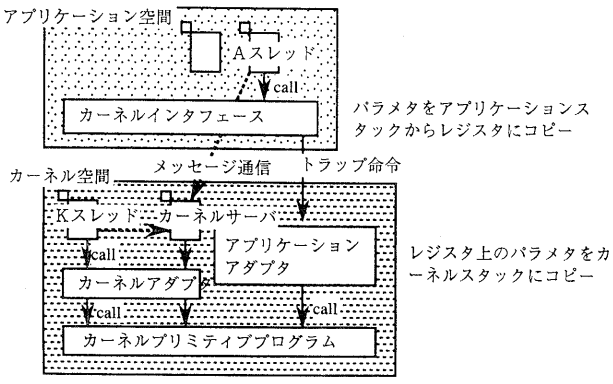

図6 カーネルプリミティブ
Fig. 6 Kernel primitive.

り，カーネルで実行される場合は，Kスレッドとなる．

4. PLATINA カーネルプロトタイプシステムの開発と評価

4.1 プロトタイプシステムの概要

プロセッサとして M68030 (25MHz)，ノード間通信の媒体として Ethernet を用いた実験システムを市販ボードコンピュータ[18]を用いて構築し，その上に PLATINA カーネルのプロトタイプをインプリメントした．ドメインを実現するため M68030 のアドレス変換機構を用いており，アドレス変換テーブルのレベルは2レベルである．

PLATINA カーネル（カーネルサーバを含まない）は，主に C/C++ を用いて記述されており，規模は約 20K ステートメントである．アセンブラ記述部は1K ステートメント以下である．

4.2 性能評価

4.2.1 PLATINA カーネル性能評価

PLATINA カーネルの各機能の実行時間を**表2**に示す．実行時間の測定はボードコンピュータ上のハードウェアタイマを利用した．以下に測定結果について示す．

(a)は，カーネルプリミティブ呼び出しに要する時間を示す．(a-1)はKスレッド，(a-2)はAスレッドがカーネルプリミティブを呼び出した場合を示す．(a-1)と(a-2)の差は実行モードの切り替えの有無による．

(b)は，カーネルサーバへの要求メッセージの送信に要する時間を示す．これにはメッセージバッファの捕捉および送信処理，メッセージ受信処理とメッセージバッファ解放が含まれる．パラメタをコピーする時間は含まない．(b-1)はKスレッド，(b-2)はAスレッドがカーネルサーバに要求メッセージを送信した場合を示す．(b-1)と(b-2)の差は，カーネルプリミティブの実行時間の差による．

(c)はスレッドの生成／削除に要する時間を示す．スレッド生成／削除のためのカーネルプリミティブ呼び出しのための処理も含まれる．(c-1)はKスレッド，(c-2)は，Aスレッドの生成／削除に要する時間を示す．

(d)は，**図7**に示すスレッドスイッチに要する時間を示す．ここに示す評価ではスレッドの優先度を1つだけにしており，スケジューリングの時間は，レディキューからスレッドを1つ取り出すだけの時間である．優先度を多くしたり，計算を伴うスケジュールアルゴリズムを導入すると，処理時間は増加する．切り替え前に実行中のスレッドを T1，切り替え後のスレッドを T2とする．(d-1)は T1，T2 共にKスレッド，(d-2)は T1，T2が同一ドメイン内に含まれるAスレッド，(d-3)は T1，T2が異なるドメイン内に含まれる Aスレッド，の場合を示す．(d-1)と(d-2)の差は，カーネルプリミティブ実行のための実

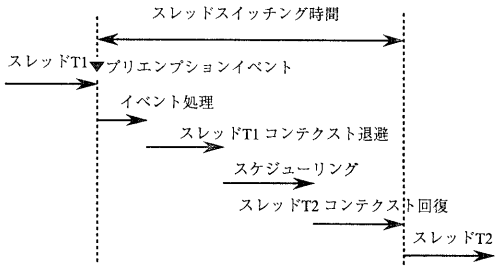

図7 スレッドスイッチング
Fig. 7 Thread switching.

表2 PLATINA カーネル性能評価
Table 2 Performance of PLATINA kernel.

処理内容	処理時間 (μ秒)
(a) カーネルプリミティブの起動	
(a-1) Kスレッドからの呼びだし	1.0
(a-2) Aスレッドからの呼びだし	14.0
(b) カーネルサーバの起動	
(b-1) Kスレッドからの呼びだし	123
(b-2) Aスレッドからの呼びだし	151
(c) スレッドの生成／削除	生成/削除
(c-1) Kスレッド	164/38
(c-2) Aスレッド	175/48
(d) スレッドスイッチ	
(d-1) Kスレッド→Kスレッド	37
(d-2) Aスレッド→Aスレッド (同一ドメイン)	47
(d-3) Aスレッド→Aスレッド (別ドメイン)	93
(e) メッセージ送受信	
(e-1) Kスレッド→Kスレッド	96
(e-2) Aスレッド→Aスレッド (同一ドメイン)	109
(e-3) Aスレッド→Aスレッド (別ドメイン)	156
(e-4) Aスレッド→Aスレッド (別ノード)	1287
(f) メッセージバッファの捕捉解放	
(f-1) Kスレッドからの呼びだし	21
(f-2) Aスレッドからの呼びだし	52

行モードの切り替えの有無による．(d-2)と(d-3)の差は，論理アドレス空間切り替えの有無による．

(e)はメッセージの送受信に要する時間を示す．これにはメッセージバッファ捕捉／解放処理の時間は含まれない．送信側のスレッドをS，受信側のスレッドをRとする．(e-1)はS，R共に同一ドメイン内のKスレッドの場合，(e-2)はS，R共に同一ドメイン内のAスレッド，(e-3)はS，R共に同一ノード別ドメイン内のAスレッド，(e-4)はS，R共に異なるノードのAスレッド，の場合を示す．

(f)はメッセージバッファの捕捉解放処理に要する時間を示す．(f-1)は送信オブジェクトがKスレッドの場合，(f-2)はAスレッドの場合を示す．

PLATINA カーネル評価に用いた同様のプログラムを用いて市販の単機能実時間 OS の性能を評価し，PLATINA カーネルが，市販実時間 OS とほぼ同等か，それ以上の性能をもつことを確認した．メッセージをプロシージャ呼びだしと同程度に使えるようにするためには，さらに効率化が必要であるが，それにはハードウェアサポートが必要である．

4.2.2 交換プログラムへの適用

従来の交換プログラム[19]では，処理効率を最優先しているため論理アドレス変換機構を用いていない．上記の交換プログラムを例に，Aスレッドを用いた際のオーバヘッドを見積もる．プロセッサ処理能力を3 MIPS，1つの呼を処理するのに呼制御スレッドが実行するプログラムの全ステップ数を9万ステップとし，呼制御スレッドが処理を終了するまでにメッセージ送受信に伴い 30 回の中断をするものとする．呼制御スレッドとしてAスレッドを用いたことにより 6% 程度オーバヘッドが増大する．一方，Aスレッドを用いることによりプログラムの保護性が高まる．効率を最優先するのであれば，PLATINA では制御スレッドをKスレッドとすることも可能である．

ここで想定している交換プログラムでは，最繁時には10000個以上のスレッドが存在する．Aスレッドのスタックを共通化することによるメモリ量の削減について見積もる．ページを用いた論理アドレス変換機構を用いるものとし，ページの大きさを4KBとする．Kスタック，Aスタックに1ページずつ割り付けると，スレッドのスタック領域として，80 MB のメモリが必要であるが，Kスタックを共用化することにより，半分に削減することができる．

5. ま と め

通信網ソフトウェア用分散処理プラットフォーム PLATINA の基本ソフトウェアである PLATINA カーネルの機能，構造およびその性能の評価について述べた．PLATINA カーネルは，複数の論理メモリアドレス空間をサポートし，並行オブジェクトを実現するためのスレッドの実行を制御する．実メモリ空間しかサポートしない従来の多くの実時間カーネルと異なり，PLATINA カーネルはプロセッサの論理メモリアドレス空間制御と実行モードを活用することで，オーバヘッドの増大を極力防止しながら，効率的な実時間処理を実現している．さらにネットワークで一意なスレッドの識別子とカーネルレベルでのノード間通信機構のサポートにより効率の良い分散処理を実現している．既存の汎用 OS に実時間機能を追加したものにくらべて高い性能をもつ．

PLATINA カーネルは実時間システムとしての性能を損なうことなくプログラムの信頼性・拡張性を向上させる．今後の発展が予想される通信網ワイドの高度通信処理システムのプラットフォームとして有効である．

謝辞 本研究を進めるにあたって，有益なコメントをいただいた山田茂樹主幹研究員，PLATINA カーネル開発に協力していただいた大崎憲嗣社員に感謝いたします．

参 考 文 献

1) 丸山勝巳，久保田稔，田中 聡，山田茂樹，大崎憲嗣：通信網用分散処理プラットフォーム PLATINA，信学技報，Vol. 91, No. 500, pp. 19-24 (1991).
2) Kubota, M., Maruyama, K., Tanaka, S., Osaki, K. and Yamada, S.: Distributed Processing Platform for Switching Systems: PLATINA, *ISS '92*, Vol. 1, pp. 415-419 (1992).
3) 坂村 健：リアルタイムオペレーティングシステム—i-TRON，情報処理学会オペレーティングシステム研究会，24-10 (1984).
4) VxWorks Reference Manual, Wind River Systems, Inc. (1989).
5) pSOS+/68K User's Manual, Software Components Group, Inc. (1991).
6) VRTX 32 User's Guide, Ready Systems (1992).
7) Cheriton, D. R.: The V Distributed System, *Comm. ACM*, Vol. 31, No. 3, pp. 314-333 (1988).

8) Black, D. L. et al.: Microkernel Operating System Architecture and Mach, *J. Inf. Process.*, Vol. 14, No. 4, pp. 442-453 (1991).
9) Chorus Systems: Overview of the CHORUS Distributed Operating Systems, Chorus Systems, CS/TR-90-25.1 (1990).
10) Tokuda, H. and Mercer, C. W.: ARTS: A Distributed Real-Time Kernel, *ACM Operating Systems Review*, Vol. 23, No. 3, pp. 29-52 (1989).
11) 丸山勝己：並列オブジェクト指向言語 COOL, 情報処理学会論文誌, Vol. 34, No. 5, pp. 963-972 (1994).
12) 丸山勝己, 久保田稔：オブジェクト指向による交換プログラムの構成法, 信学会論文誌, Vol. J74-B-I, No. 10, pp. 757-768 (1991).
13) Motorola, Inc.: *MC 68030 Enhanced 32-bit Microprocessor User's Manual*, Prentice-Hall (1990).
14) MIPS Computer Systems, Inc.: *MIPS RISC Architecture*, Prentice-Hall (1988).
15) INTEL, Inc.: *80386*, Prentice-Hall (1988).
16) Dennis, J. B. et al.: Programming Semantics for Multiprogrammed Computations, *Comm. ACM*, Vol. 9, No. 3, pp. 143-155 (1966).
17) 山田茂樹, 丸山勝己：オブジェクト指向システム用軽量ケーパビリティプロテクション方式, 情報処理学会論文誌, Vol. 34, No. 9, pp. 2037-2047 (1993).
18) Motorola,Inc.: MVME147S MPU VMEmodule User's Manual.
19) デジタル加入者交換機および端末装置特集, 研究実用化報告, Vol. 31, No. 11 (1982).

(平成6年2月7日受付)
(平成6年9月6日採録)

久保田 稔（正会員）

1954 年生. 1978 年東京大学工学部計数卒業. 1980 年同大大学院工学系情報工学専門課程修士課程修了. 同年日本電信電話公社入社. 以来, 電子交換プログラムの実行制御方式, 実時間 OS, 分散処理システム, ソフトウェア開発環境の研究開発に従事. 1987 年～88 年米国マサチューセッツ工科大学客員研究員. 現在 NTT ネットワークサービスシステム研究所, 主幹研究員. 電子情報通信学会, IEEE, ACM 各会員.

丸山 勝己（正会員）

1944 年生. 1968 年東京大学工学部電子卒業. 1970 年同大大学院修士課程修了. 同年日本電信電話公社入社. 以来, 電子交換機の実時間増設方式, 高水準言語, 最適化コンパイラ, 並行オブジェクト指向, 分散処理などの研究実用化に従事. 1985～88 年CCITT 第X研究委員会副議長. 現在, NTT ネットワークサービスシステム研究所, 主席研究員. 工学博士. 1990 年度本学会論文賞受賞. 電子情報通信学会会員.

田中 聡（正会員）

1962 年生. 1985 年慶應義塾大学理工学部管理卒業. 1987 年同大大学院理工学研究科修士課程修了. 同年日本電信電話株式会社入社. オブジェクト指向交換プログラム構造, 通信網用分散処理の研究開発に従事. 現在, NTT ネットワークサービスシステム研究所, 研究主任. 電子情報通信学会会員.