計算機アーキテクチャ　99－1
（1993. 3. 11）

TRONプロジェクトにおける実時間処理

坂村 健　　高田 広章

東京大学 理学部 情報科学科
〒113 東京都文京区本郷 7-3-1

あらまし　TRONプロジェクトは、近い将来に高度にコンピュータ化された社会（電脳社会）が到来することを想定し、コンピュータシステムのあるべき姿を総合的に研究するプロジェクトである。コンピュータシステム構築の基礎パーツとなるCPUや目的別に設計された3種類のオペレーティングシステムの研究・開発を行なう基礎プロジェクトと並行して、電脳社会における問題点を検討するために、電脳社会のプロトタイプの構築を行なう応用プロジェクトを進めている。本稿では、TRONプロジェクトの現状とプロジェクトの目指すところについて、TRONプロジェクトで重視している実時間処理の面を中心に紹介する。

和文キーワード

Real-Time Processings in TRON Project

Ken Sakamura　　Hiroaki Takada

Department of Information Science, Faculty of Science, University of Tokyo
7-3-1 Hongo, Bunkyo-ku, Tokyo, 113, Japan

Abstract　TRON is a project to study an ideal computer architecture totally viewing the future computerized society. In the project, various parts of computer systems are investigated and developed including a CPU and three operating systems for different requirements of applications. Prototype systems of the computerized society are also being developed to find real problems in the society. In this paper, we introduce the current activities and the goals of the TRON Project, focusing on real-time processing which is laid importance in the project.

英文 key words

1 はじめに

TRON (The Real-time Operating system Nucleus) プロジェクトは、近い将来に高度にコンピュータ化された社会が到来することを想定し、コンピュータ化社会におけるコンピュータシステムのあるべき姿を総合的に研究するプロジェクトである。

TRON プロジェクトにおいては、コンピュータシステム構築の基礎パーツとなる CPU やオペレーティングシステムの研究・開発を行なう基礎プロジェクトと並行して、コンピュータ化社会における問題点を検討するために、そのプロトタイプの構築を行なう応用プロジェクトを進めている。

本稿では、TRON プロジェクトの現状とプロジェクトが目指しているものを、TRON プロジェクトで特に重視している実時間処理の面を中心に紹介する。さらに、機器組み込み用のリアルタイムカーネル仕様である ITRON について、その設計方針と拡張仕様について述べる。

2 TRON の目指すもの

2.1 インテリジェントオブジェクトと HFDS

TRON プロジェクトが想定するコンピュータ化された社会 (以下では、電脳社会と呼ぶ) においては、生活のあらゆる局面においてコンピュータシステムが人間の活動を支援する。身の回りのあらゆる設備・家具・家電・道具等にコンピュータが内蔵され、さらにそれらの機器がネットワークによって接続され、相互に協調しながら人間によりよい活動環境を提供する。コンピュータを内蔵し、ネットワーク接続することができる機器を、インテリジェントオブジェクトと呼び、非常に多数のインテリジェントオブジェクトがネットワーク接続され、相互に協調して動作するシステムを、HFDS (Highly Functionally Distributed System、超機能分散システム) と呼ぶ (図 1)。HFDS 構築のための技術の研究・開発が、TRON プロジェクトの最も重要なテーマとなっている [1, 2]。

HFDS の特徴の 1 つに、想定しているノードの単位が小さいことが挙げられる。従来の分散システムにおいては、ノードとしてワークステーションないしはパーソナルコンピュータを想定している場合が多いが、HFDS においては家電、家具など生活環境を構成するあらゆる機器がインテリジェントオブジェクトとしてノードとなりうる。そのため、各ノードには極めて限定された CPU パワーしか期待できない。また、ノードの数の面からも、ワークステーションないしはパーソナルコンピュータで構成される分散システムのノード数がユーザ数と同等程度であるのに対して、HFDS におけるノードの数はユーザ数の 10 〜 1000 倍程度になる。各インテリジェントオブジェクトは別々のベンダによって開発されるため、ノード間の相互接続性が問題となる。ノードの追加 / 削除 / 移動の頻度も高い。

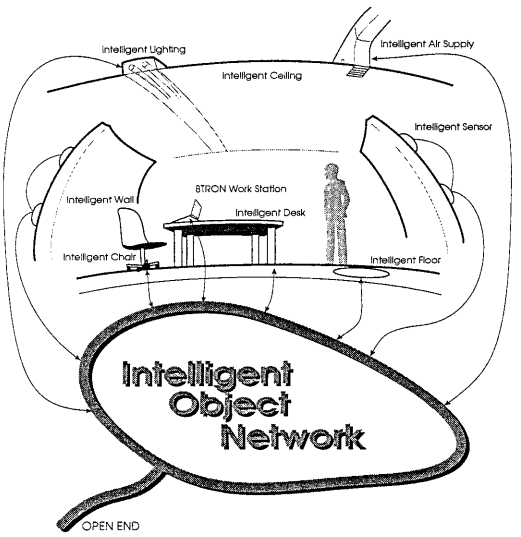

図 1: HFDS 環境

2.2 応用プロジェクト

TRON プロジェクトでは、電脳社会においてコンピュータに求められる機能・性能を探り、電脳社会の小規模なプロトタイプの構築を通じて HFDS の実現を目指すいくつかの応用プロジェクトを進めている。TRON プロジェクトでは現在、以下の応用プロジェクトに取り組んでいる [3]。

1. TRON 電脳住宅プロジェクト

   コンピュータ化された住宅を作り、将来の住環境の模擬実験を行なうプロジェクト。空調、照明、窓などあらゆる設備・機器がコンピュータ制御される。実験のためのパイロット住宅を 1989 年に建設し、各種の実験を行なった。

2. TRON 電脳ビルプロジェクト

   電脳住宅での成果をベースに、高度にコンピュータ化されたオフィスビルについて研究するプロジェクト。コンピュータシステムの支援により、人間中心の快適なオフィス環境の実現を目指す。間もなく、パイロットビルディングの建設が始まる予定。

3. TRON 電脳都市プロジェクト

   住宅、オフィスビル、情報通信システム、交通システム、物流システム、エネルギー供給システムなど、コンピュータ化された未来都市を構築するための要素技術を総合的に研究し、そのプロトタイプの実現を目指すプロジェクト。HFDS の最終的な実験環境となる。

4. TRON 電脳自動車プロジェクト

   コンピュータ化サポートされた自動車および道路システムについて研究するプロジェクト。

5. TRON マルチメディア通信プロジェクト

ISDN 網を用いてマルチメディア通信を行なう方法について研究するプロジェクト。

以上の応用プロジェクトの目標からわかるように、HFDS の最大の目的は人間の生活環境を支援することであり、HFDS を構成する各ノードはいろいろな意味で実世界との関連で動作することになる。そのため各ノードには、人間を相手にするレベルのリアルタイム性からハードリアルタイムまで、それぞれのレベルに応じたリアルタイム性が求められていることになる。

3 TRON のアプローチ

HFDS を構成する各種のノードに用いる OS を設計する際に、TRON プロジェクトでは、必要とされる性能 (リアルタイム性能) や機能に応じて複数の OS を設計するというアプローチを採っている。以下では、HFDS を構成するノードの分類を行ない、それらの構築に用いる OS について述べる。さらにその基礎となる研究について紹介する。

3.1 HFDS におけるノード

HFDS を構成するノードは、必要とされる性能や機能に応じて、以下の 3 つに分類できる。

1. インテリジェントオブジェクト

   前節で述べたような、生活環境を取り巻く各種の機器。必要とされるリアルタイム性は機器の種類によって大きく異なるが、コスト制約が厳しいために、ハードウェアの持つ性能を最大限に発揮することが求められる。

2. HMI (Human Machine Interface) マシン

   人と HFDS のインタフェースを受け持つマシン。具体的には、高度な HMI 機能を持つワークステーションないしはパーソナルコンピュータ。

   HMI マシンに求められるリアルタイム性としては、主として人を相手にするため、絶対速度としては 1/10 秒程度の反応性があればよい。しかし、現在のワークステーション上の GUI (Graphical User Interface) システムでは、1/10 秒程度の反応性は必ずしも実現できていない。これは、従来のワークステーションの OS で採用されていた TSS ベースのスケジューリング方法に原因の一端がある。1 人が 1 台のマシンを専有する状況では、すべてのタスクに計算資源を等分配するのではなく、ユーザが求める反応性に応じて計算資源を配分すべきである[1]。このような状況から、HMI マシンについても、リアルタイム性を考えた実装が必要であると考えられる。

   また、continuous media を含むマルチメディア処理が注目されているが、マルチメディア機能を持つ HMI マシンにおいては、より厳しいリアルタイム性が求められることになる。

3. サーバーマシン

   データベースサーバ、計算サーバ、通信サーバなど、バックエンドとして働くサーバマシン。実世界との直接のインタフェースは持たないが、他のノードから依頼された処理を行なうために、クライアントノードの時間制約を継承することになる。

[1] バックグラウンドでコンパイルが始まるとエディタの動きが遅くなるという状況はしばしば経験するが、1 人が 1 台のマシンを専有する状況では、ユーザがエディタを操作する手を止めたタイミングでコンパイラが動けばよいわけで、操作中はエディタの動きを優先すべきであると考えられる。

3.2 OS のシリーズ化

前節で述べた 3 種類のノードを実現する際に用いるために、TRON プロジェクトでは以下の 3 つの OS 仕様を作成し、それに基づいた実装を行なっている。

ITRON (Industrial TRON)

インテリジェントオブジェクトに組み込む制御用のリアルタイムカーネル仕様。組み込みシステムは、制御する機器の種類によって、安価で低機能な CPU (例えば、8bit MCU) から高機能な CPU まで使われるため、ITRON 仕様の中でもさらにシリーズ化を行なっている。低機能な CPU 上でも効率良く動作させるために、ハードウェアの仮想化を極力避け、カーネル仕様に CPU 依存部分を多く残している。ITRON 仕様の設計ポリシーと検討中の拡張仕様については、4節で詳述する。

すでに、各種の CPU 上に多くの ITRON 仕様カーネルが実装されており、登録されているものに限っても、10 種類以上の CPU 上で、20 以上の実装例がある [4]。

BTRON (Bussiness TRON)

人と HFDS とのインタフェースを受け持つワークステーションないしはパーソナルコンピュータ向けの OS 仕様。実身 / 仮身モデルと呼ばれるハイパーテキストベースのファイルシステムを特徴とする独自のウィンドウシステムを持つ [5]。ウィンドウマネージャやメニューマネージャ、パーツマネージャなど GUI を実現するための各種のマネージャの仕様も、BTRON 仕様に含まれている。

BTRON では、前節で述べたような GUI におけるリアルタイム性を実現するために、優先度を 3 つのクラスに分割し、クラス間および最高優先度クラス内では優先度ベースのスケジューリング、下位の 2 つのクラス内ではラウンドロビンベースのスケジューリングを行なっている。

BTRON の OS 仕様としては、比較的低性能なパーソナルコンピュータの上に実装するための BTRON1 仕様と、高性能なワークステーションを想定した BTRON2 仕様が設計されている。BTRON の本質は、実身 / 仮身モデルをベースとしたユーザインタフェースと、データ互換

性(後で紹介するTAD)にあり、BTRON1とBTRON2で内部の構成はかなり違ったものになっている。

このうちBTRON2仕様OSのカーネルでは、コンピュータ内の資源をすべて可変長レコードの列である実身であると抽象化し、そこへのリンクを仮身とするモデル化を行なっている。例えば、メモリ、ファイル、デバイス、タスク等が実身となる。また、実身の状態変化を知らせるための仕組みとして、イベントを導入している。BTRON2カーネルにおけるイベントは、従来のOSのシグナルや、ウィンドウシステムにおけるイベントの概念を、一般化して拡張したもので、複数のタスクが種々のデータを共有する場合に、データの変化をタイムリに伝えることを可能にしている。例えば、デーモンタスク(バックグランドで常に動いているタスク)は、その動作を設定するファイルが変更されたというイベントを受けとることで、新しい設定ファイルを即座に読み込むことができる[6]。またBTRON2では、ウィンドウを一種の共有メモリと見なす、新しいウィンドウシステムアーキテクチャの採用を検討している[7]。

BTRON仕様OSの実装としては、86系のCPUで動作するBTRON1仕様のOSがある。さらに現在、TRON仕様チップで動作するBTRON2仕様のOSの実装が、ITRON仕様のカーネルをマイクロカーネルに用いて行なわれている[8]。

CTRON (Central/Communication TRON)

サーバマシンで用いるためのOS仕様。CTRONカーネルとITRONとの最大の違いは、CTRONではソフトウェアの流通性を重視しているために、ITRONに比べてハードウェアを仮想化する度合が高く、ハードウェアに依存した規程が少ないことである。またCTRONカーネルが、マルチユーザをサポートする機能を備えていることも、ITRONとの大きな違いである。

CTRON仕様では、カーネルと入出力制御からなる基本OSインタフェースに加え、情報蓄積制御、実行制御、OSIの各プロトコルに対応した通信制御などからなる拡張OSインタフェースについても、標準化を行なっている[9]。

CTRONでは、リアルタイム性に加えて、ソフトウェアの流通性を重視しているため、CTRON仕様に準拠したOSの互換性を保証するために、インタフェースの検定を行なっている。現在までに各種のCPU上で動作する10余りの製品が検定を通過している。また、CTRON上で開発されたソフトウェアの流通性を検証するために、大規模なソフトウェア移植実験を行ない、CTRON仕様がソフトウェアの流通性に貢献することを実証した[10]。

3.3 基礎技術の研究

さらに我々は、上で述べたOSを構築するベースとなる技術として、以下のような研究を進めている。

TRON仕様チップ

HFDSを構成するノードに用いるために設計された汎用のCPU仕様。ITRON仕様カーネルやBTRON仕様OSを高速に実行するための機能を持つ。

ITRON仕様のカーネルを高速に実行するための機能としては、タスクスイッチのための命令や、カーネル中の各種のキュー構造を操作するためのキュー操作命令、遅延割込機能が挙げられる。ITRONやCTRONを使って作られる制御用のシステムの場合、タスクスイッチの性能がシステム全体の性能に大きな影響を与える。また近年、OSの実現にマイクロカーネル方式が採られることが多いが、タスクスイッチ性能はマイクロカーネル方式で実装されるOSの性能にも大きな影響を持つ。

TRON仕様チップが持つこれらのOSサポート機能により、6～7MIPS程度の性能を持つTRON仕様チップ上に実装されたITRON仕様カーネルでは、割込禁止時間5ないしは8μsec.、タスク切替え時間13ないしは25μsec.の性能を得たという報告がなされている[11, 12]。

現在までに、6種類のTRON仕様のCPUが実装されており、その中で最も高性能なGmicro/300は、40MHzのクロックで動作時に40MIPS弱の性能を発揮する。さらに現在、次世代のTRON仕様チップとして、100MIPS程度の性能を目標に、スーパスケーラ方式を採用したCPUの実装が進められており、間もなく発表される予定である[13]。

TRON標準システムバス

TRON仕様チップの性能を活かすバスとして、TRON標準システムバスの研究を行なっている。非同期式バスの仕様であるTOBUSと、同期式バスの仕様であるTOXBUSの仕様を公開している。

このうちTOXBUSは、スプリット転送方式をサポートする同期式バスで、フォールトトレラント支援機能やコピーバックキャッシュコヒーレンス制御もサポートしている[14]。

μBTRONバス

小規模なインテリジェントオブジェクトを接続するためのLANとして、リアルタイム性を持ち、一般ユーザにも手軽に扱えることを目指して、μBTRONバス仕様の設計および実装を行なっている[15]。

μBTRONバスは、4Mbpsの速度を持つトークンリング方式を採用しており、フレームに4レベルの優先度を持つ。トークンリング方式のLANは、リアルタイム性に優れている反面、一般ユーザが手軽に扱えるようにするにはいろいろな工夫が必要である。そのためμBTRONバスでは、1本に信号の往復路を含んだケーブルを用いてバス状に接続することを可能にする、コネクタを抜くだけで信号が折り返されるようなコネクタを採用する、ノードの電源を切ると信号を通過させるようにする、コネクタの

形状を変えることでどのように接続しても正しい接続を保証するといった工夫を行なっている。

トークンリング方式の LAN を設計する際に、最大パケット長を短くすると、リアルタイム性は向上するがヘッダ転送等に伴うオーバーヘッドは大きくなるというトレードオフがある。μBTRON バスにおいては、低優先度のパケットの最大長を長めに設定し、高優先度のパケットは低優先度のパケットの転送中に割り込めるような仕様とし、このトレードオフの解決をはかっている。また、各ノードの時計を同期させるためのフレーム形式や、単純なセンサーなど CPU を用いる必要性の少ないデバイスをノードとして接続するためにネットワーク制御 LSI から直接 I/O ポートの制御を行なうためのフレーム形式などを用意している。

μBTRON バス仕様に従ったネットワーク制御 LSI は、最初のバージョンの実装が完了し、今後各方面へ応用することを予定している。

TAD (TRON Application Databus) と TULS (TRON Universal Language System)

HFDS において、別々のベンダによって開発されるインテリジェントオブジェクト間での協調動作を可能にするには、あらゆる階層の通信プロトコルの標準化が必要になる。

TRON プロジェクトにおいては、通信プロトコルのうち、最も上の階層にあたるデータ交換フォーマット[2]の標準化に重点を置いている。TRON プロジェクトで標準化作業を行なっているデータ交換フォーマットを TAD (TRON Application Databus) と呼ぶ。

TAD は、実世界のあらゆる情報に対して、標準的なデータ形式を定義することを目指している。具体的には、文書データや図形データに加えて、音声や動画などの時間変化を伴うデータ、各種の制御用データ[3]、手続きの記述形式といった分野で標準化作業を行なっている。

このうち、音声や動画などの時間変化を伴うデータの表現形式をリアルタイム TAD と呼んでいる。リアルタイム TAD は、音素や静止画データをベースに、時間的変化を定義するためのデータ形式である。データ相互間の同期条件や、分岐・繰り返しといったデータの時間構造を、データ形式の中に取り込んでいることを特徴としている [16]。

手続きの記述方法としては、TULS (TRON Universal Language System) と呼ばれる言語システムについての研究を行なっている。TULS は、TAD 中において手続きを記述する枠組であると同時に、TAD の表現力を拡張することを可能にする。例えば、繰り返し出てくるデータを TULS によってマクロ定義することで、データの抽象化が可能になり、データをまとまりとして扱うこと[4]やデータ量の圧縮が可能になる。言い換えると、TAD を TULS が扱うデータ構造と見なすと、TULS が実世界のあらゆるデータを表現する言語になっていることになる [17][5]。

TAD にリアルタイムデータが含まれているため、TULS もリアルタイム性を保持して動作しなければならない。TULS のリアルタイム言語としての側面については、今後の研究課題である。

TULS は、TAD というデータ交換のためのインタフェースを拡張する言語であったわけだが、TRON プロジェクトでは、コンピュータシステムを構成するあらゆるインタフェースをプログラム拡張することを考えている。この考え方を、プログラマブルインタフェースと呼ぶ [18]。

HMI の標準化

HFDS 環境においては、生活環境を取り巻くあらゆる機器が電子化されるため、一般のユーザがそれらを有効に利用できるようにするためには、各種の機器の操作作法の標準化が必要である [19]。

TRON プロジェクトにおける HMI 標準化の特徴は、コンピュータの画面上の GUI の標準化だけではなく、一般の電子機器の操作パネルまで含めた標準化を狙っていることである。そのために、電子機器に使われているスイッチやボリュームを分類整理し、それぞれの使用方法および組み合わせ方にガイドラインを与えている。一般の電子機器のインタフェースを、GUI と対比して、SUI (Solid User Interface) と呼んでいる。また、HFDS 環境をあらゆる人が利用できるようにするために、多国語 HMI やイネーブルウェアと呼ばれる身体障害者のための HMI についても検討を行なっている。

Life-critical な応用では、HMI の設計自身にもリアルタイムシステムの視点を持ち込むことが必要である。ユーザが情報を処理できる能力には限界があり、コンピュータ内の資源と同様、スケジューリングの対象としなければならない[6]。そのためには、ユーザの性能モデルを作り、ユーザの処理能力まで含めたスケジューリングモデルを構築する必要がある。

4 ITRON 仕様カーネル

ITRON 仕様カーネルは、HFDS 環境におけるあらゆる組み込みシステムへ適用できることを目指している。そのために我々は、ITRON 仕様の設計にあたってなるべく

---

[2] データ交換フォーマットは、OSI の 7 階層参照モデルでは、アプリケーション層のさらに上に位置付けられる。

[3] 例えば、長さ、体積、圧力、流量、温度、色彩、明るさ、電力、磁力、エネルギー量、時間などを表すためのデータ形式。

[4] 例えば、TAD の図形データ形式の規程では、矢印を表す形式は用意されていないが、TULS によって矢印をいくつかの直線からなるものと定義することで、標準形式のデータ中でも矢印という情報が保存でき、TAD に矢印を表す形式が導入されたのと同様の処理が可能になる。

[5] 類似の性格をもった言語としては、プリンタ出力イメージを記述するための言語であるポストスクリプトが挙げられる。

[6] スリーマイル島の原発事故の際には、非常に多くの警告がメッセージが出されたために、オペレータには何が起こったかわからず、これが事故を大きくした原因の 1 つであると言われている。

広い範囲のシステムに適用できるよう配慮するとともに、その拡張仕様の検討を続けている。本節では、まずITRON仕様を設計するにあたっての基本ポリシーについて述べる。さらに、ITRON仕様を分散システムおよびマルチプロセッサシステムに適用可能にするための拡張仕様について紹介する。HFDS環境の性質上、ネットワーク接続された分散システムをサポートする拡張は特に重要なものと位置づけている。

4.1 ITRON仕様の設計ポリシー

ITRON仕様を設計にあたって、基本的に考慮すべき要求事項としては、以下ようなものがある[20, 21]。

1. ハードウェアの持つ性能を最大限に引き出せること。
2. リアルタイム性が保てること。ここでいうリアルタイム性とは、実行時間の予測可能性と高速な応答性のことをいう。
3. 広い範囲のハードウェアの上で動作すること。
4. プログラムの生産性向上に役立つこと。

一般に、プログラムの実行効率を上げることと、生産性の向上とはトレードオフの関係になることが多い。例えば、実行効率を上げるためには、実装依存の部分を多くし、ハードウェア毎に最適な仕様を採用できるようにする方法が考えられる。しかし、この方法はプログラムの移植性を下げることになり、ひいてはプログラムの生産性を悪くすることになる。TRONプロジェクトの各OSは、このトレードオフの決め方で性格付けされていると言うこともできる。その中でITRONは、実行効率を最重要視するカーネル仕様である。標準化の度合を下げ、実装依存の部分を増やしている。

以上の要求を実現するために、ITRON仕様の設計あたっては以下のアプローチを採っている。

1. ハードウェアの過度の仮想化を避けることにより、ハードウェアの性能を最大限に引き出し、リアルタイム性を持たせる。

   標準OS仕様を作成する場合、ハードウェア間で共通な仮想アーキテクチャを想定し、それをベースに仕様を定義する方法が考えられる。しかし、この方法では、想定されたアーキテクチャとハードウェアの実際のアーキテクチャの間のギャップを埋めるために、カーネルのオーバーヘッドが大きくなり、システムの性能低下につながる。また、プログラマが実行にかかる時間を把握することが難しくなり、リアルタイム性を持ったプログラムの作成に支障が出る。

   そこでITRON仕様では、何を標準化すべきかを検討し、標準化する部分と実装依存として残す部分を分離する。これにより、ハードウェアを過度に仮想化することを避け、ハードウェアの性能を最大限に引き出すことが可能になる。ハードウェアの仮想化を避けている例として、割込ハンドラの起動方法が挙げられる。ITRONカーネル仕様では、外部割込が発生した際に、カーネルが関与することなく直接ユーザが定義した割込ハンドラが起動される。

2. アプリケーションに対する適応化を考慮する。

   アプリケーションに対する適応化とは、アプリケーションが用いるカーネルの機能や必要とする性能に応じてカーネル内部の実現方法を変え、システム全体としての性能向上を計ることをいう。組み込みシステムの場合、カーネルのオブジェクトコードはアプリケーション毎に生成されるため、アプリケーションに対する適応化が特に有効に働く。

3. ハードウェアに対する適応化を考慮する。

   ハードウェアに対する適応化とは、ハードウェアの持つ性能や性質に応じて、カーネルの内部の実現方法を変え、システム全体としての性能向上を計ることをいう。

4. プログラマの教育の容易さを重視し標準化を行なう。

   プログラマの教育を重視するため、標準化を通じて一度覚えた事が広く活用できるよう配慮する。また、仕様における用語の使い方や、システムコール名の決め方などは、できる限り consistent になるよう配慮する。また、教育用テキストの作成にも力を入れている[22]。

5. 仕様のシリーズ化やレベル分けを行ない、広い範囲のハードウェアに適合できるようにする。

   多種多様なハードウェアへの適用を可能にするため、仕様のシリーズ化やレベル分けを行なう。すでに作成した仕様のうち、μITRON仕様は主に8bitプロセッサなど小規模なシステム向けに作成されたもので、ITRON2仕様は32bitプロセッサなど大規模システム向けのものである。また、これらの仕様の中でも、さらに機能毎の必要度に応じたレベル付けを行なっている。

6. 豊富な機能を提供する。

   カーネルが提供するプリミティブを少数に限定するのではなく、性質の異なる豊富なプリミティブを提供するというアプローチを取る。ハードウェアやアプリケーションの性質に適合したプリミティブを利用することで、実行性能やプログラムの書きやすさの向上が期待できる。

4.2 ITRON仕様の拡張

ITRON仕様を、分散システムやマルチプロセッサシステムといった複数のプロセッサを用いたシステムをサポートするよう拡張するにあたって、複数のプロセッサを用いたシステムのハードウェア構成の分類を行ない、それを元にして仕様をシリーズ化する方針をとることにした。

複数のプロセッサを用いたシステム上にリアルタイムカーネルを実装するにあたっては、共有メモリを持つかどうかが極めて大きな差になる。例えば、他のプロセッサ上で実行されているタスクの状態を参照する際に、そのタスクのコントロールブロックが共有メモリ上に置かれている場合には、相手のプロセッサの動作を大きく妨げることなく状態を参照できるのに対して、共有メモリを持たない場合には、相手のプロセッサに処理を依頼する必要がある。処理の依頼を行ない結果を受けとるまでに時間がかかる場合には、システムコール内で他のプロセッサからの結果を待っている状態を考えなければならない。さらに、処理を依頼される側のプロセッサで、依頼された処理が待ち状態に入る場合の処置を考慮する必要も出てくる。

以上の考察の結果、共有メモリアーキテクチャをとることが可能なマルチプロセッサシステムと、プロセッサを含むノードが物理的に分散していることが必要で、ノード間を LAN 等のコンピュータネットワークで接続した分散システムの２つの場合に分けて ITRON 仕様の拡張を行なっている。

分散システムへの拡張

ITRON を分散システムに対応させることは、HFDS 構築のためには最も基本的な仕様となるため、μITRON 仕様のバージョンアップという形で分散システムのサポートを取り込むことにした。この拡張仕様を μITRON 3.0 と呼び、仕様検討を行なっている。

ITRON を分散システムに拡張するにあたって、我々は２段階の拡張を考えた。最初の段階として、ネットワークの構成はシステム開発者が指定し、一度システムが立ち上がると、ネットワークの構成がほとんど変化しないシステムへの適用を考える。この様なシステムの代表例としては、単一の組み込みシステムを分散システムとして構築する場合がある。具体例としては、自動車の制御システムやロボット制御、プラント制御があげられる。μITRON 3.0 では、この段階の拡張を行なう。

これに対して、HFDS 環境においては、ネットワークの使用目的が多種多様で、ネットワーク構成がシステム動作中に変化することがある。その様なシステムへの適用は、さらに次の段階として仕様検討を行なう計画である。これを IMTRON と呼ぶ (図 2)。

μITRON 3.0 仕様では、各ノードは、プロセッサ、メモリ、I/O およびネットワークコントローラを含む自立したシステムであり、ノード間にはメッセージが届く仕組みがあることのみを必要とする。各ノードの上では ITRON カーネルが動作しており、タスクや同期・通信用の各種のカーネルオブジェクトを持つ。

μITRON 3.0 仕様では、他のノードのオブジェクトを、自ノードのオブジェクトと同様に操作できるような拡張を行なっている。具体的には、オブジェクト ID を拡張し、従来の ITRON 仕様と同様のシステムコールインタフェースを用いて、分散システムのプログラミングを可能としている。他ノードのオブジェクトを操作する機能を接続機能と呼ぶ。

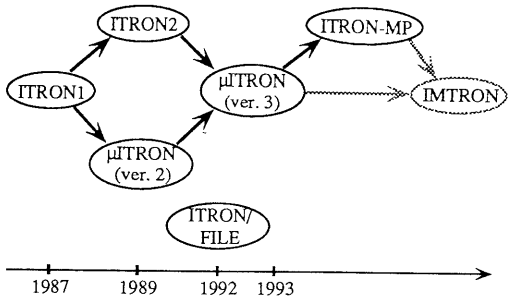

図 2: ITRON 仕様の発展

また、複数のベンダによって開発された μITRON 3.0 仕様カーネル間での相互接続を可能にするために、ノード間通信のパケット形式についての標準化を行なっている。この際に、小規模なノードと大規模なノードで用いられる CPU の語長が違うこともありえるため、各種のデータについても語長の違いを考慮して標準化を行なっている。

マルチプロセッサシステムへの拡張

ITRON 仕様をマルチプロセッサシステムのために拡張する主な目的は、性能向上とフォールトトレラントシステムへの適用を可能にすることである。この拡張仕様を ITRON-MP と呼ぶ。

ITRON の主な応用分野である組み込み用のシステムにおいては、最終製品のコストパフォーマンスを追求するために、システムのハードウェアはアプリケーションごとに最適になるよう設計されることが多い。マルチプロセッサシステムの場合、最適なアーキテクチャ (とりわけプロセッサ間の接続方法) はアプリケーションによって大きく異なるため、特定のアーキテクチャを想定して設計されたカーネル仕様では、標準とはなり得ない。バックプレーンバスが標準化され、そのバスに対応した各種のボードが提供されている現在では、それらのボードを組み合わせて独自のシステムを構築することは容易であり、同様のことは組み込み用に限らず特定目的専用のマルチプロセッサシステム一般に成り立つ。

そこで、ITRON-MP 仕様の設計においては、特定のアーキテクチャを想定せず、どのようなアーキテクチャにでも適用可能な仕様となることを目指す。具体的には、ITRON-MP 仕様では多様なアーキテクチャに対応するのに十分なカーネル機能を定義し、実際のシステムに適用する場合には、そのシステムのアーキテクチャに適合した機能のみを選択してカーネル仕様が決めることになる。

さらに、システム開発のコストを下げるために、アーキテクチャの記述とカーネル構成の記述から、そのアーキテクチャに適応したカーネルを生成できるような実装も可能にする [23]。

5 おわりに

本稿では、TRONプロジェクトの現状とプロジェクトの目指すところについて、実時間処理に対する考慮点を中心に紹介した。以上ではふれなかったが、TRONプロジェクトでは、作成した仕様はすべてオープンつまり誰もが自由に利用できるようにするというオープンアーキテクチャの考え方を採っている。オープンアーキテクチャは、HFDSを実現する上で不可欠な考え方である。

TRONプロジェクトは、近い将来に社会のコンピュータ化が進行した場合のことを考え、今から研究しておくべきことは何かを洗いだし、研究活動を進めている。解決すべき課題は数多く残っているが、それらを1つづつ解決していこうと考えている。

参考文献

[1] K. Sakamura, "The objectives of the TRON project," in *TRON Project 1987*, pp. 3–16, Springer-Verlag, 1987.

[2] K. Sakamura, "The computerized society," in *TRON Project 1989*, pp. 3–14, Springer-Verlag, 1989.

[3] K. Sakamura, "TRON application projects: Gearing up for HFDS," in *Proc. of the Eighth TRON Project Symposium*, pp. 2–14, IEEE CS Press, 1991.

[4] K. Saitoh, "ITRON standards," in *Proc. of the Eighth TRON Project Symposium*, pp. 16–24, IEEE CS Press, 1991.

[5] K. Sakamura, "BTRON: The bussiness-oriented operating system," *IEEE Micro*, vol. 7, pp. 53–65, Apr. 1987.

[6] K. Sakamura, "Design policy of the operating system based on the BTRON2 specification," in *TRON Project 1990*, pp. 103–118, Springer-Verlag, 1990.

[7] N. Koshizuka, H. Takada, and K. Sakamura, "Window shared data pool: A new window system for BTRON2," in *Proc. of the Eighth TRON Project Symposium*, pp. 151–168, IEEE CS Press, 1991.

[8] N. Osawa, "2B: An operating system based on BTRON2 specification," in *Proc. of the Eighth TRON Project Symposium*, pp. 141–150, IEEE CS Press, 1991.

[9] (社) トロン協会 編, *CTRON*概説 - 入門・共通規程編 -. 新版 原典 CTRON 大系 1, オーム社, 1992.

[10] T. Ohta, T. Terazaki, T. Wasano, and M. Hanazawa, "Software portability in CTRON," in *Proc. of the Eighth TRON Project Symposium*, pp. 86–92, IEEE CS Press, 1991.

[11] A. Shimohara, T. Minohara, K. Kudou, and H. Ito, "REALOS/F32: Implementation of ITRON2 specification on Gmicro F32," in *TRON Project 1989*, pp. 33–43, Springer-Verlag, 1989.

[12] S. Yamada, K. Horikoshi, T. Shimizu, and H. Takeyama, "HI32: An ITRON-specification operating system for the H32/200," in *TRON Project 1989*, pp. 77–97, Springer-Verlag, 1989.

[13] S. Matsui, M. Yamamoto, I. Kawasaki, S. Narita, F. Arakawa, K. Uchiyama, and K. Hashimoto, "Gmicro/500 microprocessor: Pipeline structure of superscaler architecture," in *Proc. of the Ninth TRON Project Symposium*, pp. 56–62, IEEE CS Press, 1992.

[14] K. Okada, M. Itoh, S. Fukuda, T. Hirosawa, T. Utsumi, K. Yoshioka, Y. Tanigawa, and K. Hirano, "Performance evaluation of TOXBUS," in *TRON Project 1990*, pp. 347–374, Springer-Verlag, 1990.

[15] K. Sakamura, K. Tamai, K. Tanaka, S. Tsunoda, K. Tsurumi, and M. Kaneko, "The μBTRON bus: Functions and applications," in *TRON Project 1989*, pp. 101–112, Springer-Verlag, 1989.

[16] Y. Ueshima, M. Zeze, Y. Yamamoto, and K. Sakamura, "A fundamental study on picture motion TAD on BTRON specification operating system," in *Proc. of the Eighth TRON Project Symposium*, pp. 169–176, IEEE CS Press, 1991.

[17] K. Sakamura, "TULS: TRON universal language system," in *TRON Project 1988*, pp. 3–20, Springer-Verlag, 1988.

[18] K. Sakamura, "Programmable interface design in HFDS," in *TRON Project 1990*, pp. 3–22, Springer-Verlag, 1990.

[19] K. Sakamura, "Human interface with computers in everyday life," in *Proc. of the Ninth TRON Project Symposium*, pp. 2–13, IEEE CS Press, 1992.

[20] H. Monden, "Introduction to ITRON, the industry-oriented operating system," *IEEE Micro*, vol. 7, pp. 45–52, Apr. 1987.

[21] K. Sakamura, "ITRON: An overview," in *TRON Project 1987*, pp. 29–34, Springer-Verlag, 1987.

[22] (社) トロン協会 ITRON 専門委員会 編, *ITRON*標準ガイドブック *'92-'93*. パーソナルメディア, 1992.

[23] H. Takada and K. Sakamura, "ITRON-MP: An adaptive real-time kernel specification for shared-memory multiprocessor systems," *IEEE Micro*, vol. 11, pp. 24–27,78–85, Aug. 1991.