計算機アーキテクチャ 99－2
(1993. 3. 11)

マルチメディア統合環境プロジェクトにおける リアルタイム処理技術†

徳田 英幸[1,2]　斎藤信男[1]

[1] 慶應義塾大学環境情報学部
[2] カーネギーメロン大学計算機科学部

あらまし　本稿は、マルチメディア統合環境プロジェクトにおいて研究開発している分散マルチメディア環境を実現するためのリアルタイム処理技術に関して述べている。我々が開発した連続メディアブジェクト、サービスの質(QOS)、QOSクラスの概念などの紹介とともに、研究開発している分散リアルタイムカーネル、Real-Time Mach、リアルタイム通信プロトコルなどの現状および今後の課題について述べる。

キーワード　マルチメディアシステム、リアルタイムシステム 、分散リアルタイムカーネル、リアルタイムプロトコル、マイクロカーネル

Real-Time Computing Technology for Integrated Multimedia Computing Environment

Hideyuki Tokuda[1,2]　Nobuo Saito[1]

[1]hxt@sfc.keio.ac.jp, Keio University, 5322, Endo, Fujisawa-shi, Kanagawa, 252 Japan,
[2] hxt@cs.cmu.edu, Carnegie Mellon University, Pittsburgh, PA 15213 USA
[3] ns@slab.sfc.keio.ac.jp, Keio University, 5322, Endo, Fujisawa-shi, Kanagawa, 252 Japan,

Abstract.　In this paper, we describe real-time computing technology which has been developed for building an integrated multimedia environment. We also introduce the notion of continuous media objects, quality of services (QOS), and QOS classes for multimedia data such as digital video and audio. The current status and future issues of our distributed real-time kernel, Real-Time Mach, and real-time communication protocols are discussed.

Keywords:　Multimedia System, Real-Time System, Distributed Real-Time Kernel, Real-Time Communication Protocol, Micro Kernel

†この研究は、情報処理振興事業協会(IPA)が実施している開放型基盤ソフトウェア研究開発評価事業「マルチメディア統合環境基盤ソフトウェア」プロジェクトのもとに行なわれました。本稿に含まれている見解や結論は、著者自身のものであり、IPA自身の見解や結論を表すものではありません。

1 はじめに

慶應義塾大学環境情報学部におけるマルチメディア統合環境プロジェクトは、1992年からスタートし、参加企業11社とともに、分散マルチメディア環境のソフトウェア基盤に関する基礎技術を研究し、その環境を実現するためのプロトタイプを開発し、評価実験を行なうことを目的としている。特色は、従来までの基盤ソフトウェア技術では確立されていない種々のマルチメディアオブジェクトの取り扱いを、分散環境下で行うことのできる新しい基盤ソフトウェアの基礎技術の研究開発にある。基本プラットフォームを構成する分散リアルタイムオペレーティングシステム、実時間通信プロトコル、マルチメディア・サーバ、サーバツールキット、分散マルチメディアオブジェクトシステム／ファイルシステムなどを研究テーマとし、その統合された環境を実現できる基盤ソフトウェアの開発を目指している。また研究成果物を PDS とし、より広域な普及もめざしている。

今年度は、共通基盤ソフトウェアのためのカーネルの評価対象として、筆者らがカーネギーメロン大学と慶應義塾大学で共同研究開発してきたリアルタイム・マイクロカーネル **Real-Time Mach 3.0** [20, 21] をもとに研究開発評価を進めている。Real-Time Mach 3.0 は、Mach 3.0 マイクロカーネル [1] を ARTS カーネル [18] で開発したリアルタイムスレッド、リアルタイムスケジューラ、同期操作、リアルタイムプロトコルモジュールなどを取り入れてリアルタイム処理用に拡張したカーネルである。

本稿では、マルチメディア統合環境プロジェクトにおいて研究開発しているリアルタイム処理技術に関し、我々が開発した連続メディアブジェクト、サービスの質 (QOS) と QOS クラスの概念の紹介とともに、分散リアルタイムカーネル、実時間通信プロトコルなどの現状および今後の課題を述べる。

2 基本的な課題

リアルタイム処理技術は、マルチメディア統合環境を実現する共通ソフトウェアプラットフォームを構築する上でもっとも基本的な技術である。従来からのハードリアルタイムとソフトリアルタイムというタスクのクラス分けからマルチメディアオブジェクトに関するタスクをみると、その多くは、ソフトリアルタイム処理であるといえる。しかし、デジタルビデオ情報などを利用しての高度なテレプロセッシングやバーチャルリアリティなどのアプリケーションでは、時間的制約に対してハードな要求を満たすことも必要となってきている。

ソフトウェアプラットフォーム上で、従来からの UNIX などに代表されるタイムシェアリング方式に基づく資源管理を行った場合に、画像表示が一時的に止まってしまったり、音声が途切れてしまうといった不都合が起きている。さらに、分散マルチメディア環境下においてはリアルタイム処理が、オペレーティングシステムにおけるプロセッサスケジューリングだけではなく、同期操作、メモリオブジェクトの管理、高速な I/O 処理、実時間メッセージ通信などのサービスモジュールにおいて時間的制約に基づいておこなわれることが必要であり、予測できない遅延をできるだけ排除することが重要となってきている。

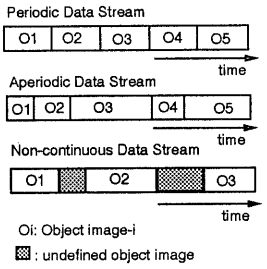

図 1: Continuous Media Objects

ここでは、まずこれらの時間的制約をもったマルチメディアデータとしての連続メディアオブジェクト (Continuous Media Object) の概念を紹介し、これらの連続メディアオブジェクトを使った分散アプリケーション上での課題について述べる。

2.1 連続メディアブジェクトの概念

連続メディアオブジェクトは、通常のオブジェクトと異なり、デジタル化された音声やビデオデータなどのように、時系列で連続的に変化するデータを呼び、オブジェクトの正当性が、値の正当性だけでなく時間的な正当性にも依存している。

ここでは、連続メディアオブジェクトをデータストリームで表し、ストリームの構成要素である個々のデータオブジェクトに時間的制約が付随しているものと定義する。個々のデータオブジェクトに与えられた時間的制約が同一の場合、そのストリームを周期的データストリームと呼び、そうでない場合を非周期的データストリームと呼ぶ。また、データストリームが、連続的に定義されていない場合を非連続データストリームと呼ぶ。これらのデータストリームを、図 1 のように表すことができる。

例えば、、ビデオ、動画、音声データなどは、周期的データストリームで表わせ、オンラインのデスクトップ・プレゼンテーションは、非周期的データストリームで表

図 2: Spatial resolution of video image

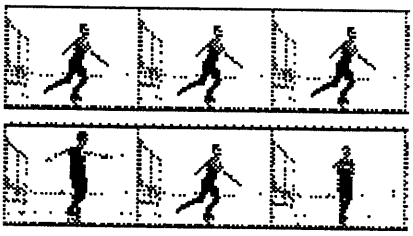

図 3: Temporal resolution of video image

すことができる。また、手動によるスライドなどのプレゼンテーションは、非連続データストリームとなる。

2.2 サービスの質 (QOS) とクラス

連続メディアオブジェクトを扱うシステムにおいて、そのシステムが提供するサービスの質は、連続メディアオブジェクトに対する空間的解像度 (spatial resolution) と時間的解像度 (temporal resolution) によって表現することができる。例えば、ビデオや音声データなどの場合、時間的な解像度は、フレーム数/秒 (fps) やサンプリングの速度で決定され、空間的な解像度は、データサイズやデータ圧縮率に依存する。

空間的解像度の違いの例を図 2 に、時間的解像度の違いの例を図 3 にそれぞれ示す。図 2 でわかるように、高い空間的な解像度をもつデータオブジェクトの方がより精密にデータを表現できている。また、図 3 に示されているように、時間的な解像度が高いほど、より細かく連続的な動きを表すことが可能である。

また、実際のシステムにおいては、このような空間的、時間的解像度をユーザが細かく指定する代りに、いくつかの代表的な解像度を設定しておき、それらのクラス (QOS クラスと呼ぶ) を選択することを行っている。例えば、ビデオデータなどの場合、ユーザは、単に 1, 2, 3, 5, 10, 15, 30 fps (frame per second) といった QOS クラスから時間的解像度を選択したり、8, 16, 24 bpp (bits per pixel) といった QOS クラスから、空間的解像度を選択している。

ユーザから与えられたこれらの QOS クラスをもとに、システム資源の利用状況に応じて、動的にビデオセッションなどの QOS をコントロールする方式も開発している [22]。

2.3 いくつかの課題

分散マルチメディア環境を実現するシステムにおいて、さまざまな技術的課題が未解決であるが、ここでは、以下にあげたリアルタイム処理に関する問題について考える。

- リアルタイムスケジューリング問題
- リアルタイム同期問題
- リアルタイム通信問題

代表的なスケジューリング問題は、もっとも基本的なプロセッサスケジューリングで発生する。例えば、ビデオ、画像、音声などの連続メディアオブジェクトをワークステーション上で多重処理する際に起きる。あるプロセスが、音声データや MIDI データなどの処理中に他のユーザ・プロセスの起動によってプロセッサが横取りされてしまうと、ビデオや音の再生が正しく行なうことができなくなってしまう。このような場合、従来のタイムシェアリングシステムで使用されているスケジューリング機能では、周期的なアクティビティの制御が難しいだけでなく、一時的な過負荷な状況 (transient overload) の下では、どのアクティビティがデッドラインを満足できなくなるかなどの予測が全くできず、時間的制約を持つマルチメディアサーバなどの多重処理が著しく困難となってしまっている。また、分散環境では、同期やメッセージ通信など、他のスケジューリングドメインを考慮しないと、プロセッサスケジューリングの効果が、ネットワークを伝わって他のホストでのアクティビティに反映することができなくなってしまう。

リアルタイム同期問題においては、優先度の高いリアルタイムタスクが何らかの原因で優先度の低いタスクによって、実行が遅れてしまう状況をプライオリティインバージョン (priority inversion)[15] といい、クリティカルセクションなどを複数のタスクが共有している際に、優先度の高いタスクがデッドラインを満たせなくなる問題が生じてしまう。実際、プライオリティインバージョンは、タスク間の同期に関してだけでなく、横取りできないサービスを提供しているサーバ、ネットワーク・アクセスプロトコール、プロセッサ・バスアーキテクチャ上

のアクセスプロトコールなど、あらゆるシステムレベルで起きる基本的な問題である。

また、マルチメディアシステムにおいては、よく言われる“リップ・シンク (lip- synching) 問題”[8] [1]のような、二つのデータストリーム間での同期 (interstream synchronization) と、ビデオフレーム間での同期のような一つのデータストリーム内での同期 (intrastream synchronization) の両方の問題がある。例えば、2つの独立した通信チャネルからネットワークを介して送られてくる音声とビデオストリームに対して、確実に同一時間軸上で同期させることを保証することは大変困難である。また、アプリケーションからの音声やビデオ転送に対するサービスの質に対する要求をもとに、それに伴うデータ転送に関する時間的制約 (遅延 (delay) やジッタリング (jittering) の上限) やスループットに対する制約を満足できるようどのようにネットワークやプロセッサの資源を管理するかといった問題もある。

リアルタイム通信問題に関しては、単にメッセージを高速に転送したり、スループットを向上させるといったことだけでは解決できない。メッセージ転送に対するデッドラインの保証や、連続したメッセージ間の到着時間差の上限などを保証できるように、プロトコル処理に必要なCPUやメモリ資源、ネットワーク資源を確保できるような資源管理ができなくてはならない。また、プロトコル処理モジュールにおいても、優先順序の低いメッセージが何等かの理由により、高いメッセージの転送や到着を遅延させてしまうといったプライオリティインバージョンの状況をできる限り回避しなければならない [19]。

3 分散リアルタイムカーネル

ここでは、我々が共通ソフトウェアプラットフォームの評価対象として使用している Real-Time Mach 3.0 マイクロカーネルを中心に、新しく開発されたリアルタイムスレッドモデル、スケジューリング技法、リアルタイム同期プロトコルなどの概要を述べる。

3.1 リアルタイムスレッドモデル

従来から用いられている計算のモデルは、プロセス・モデルである。多くの商用リアルタイム・エグゼクティブでは、単純なマルチ・プロセスのモデルを採用しており、プログラムの動作に対する時間的制約は、プログラム上には明示されていなかった。従って、実行時のプロセッサスピードが数倍に速くなった場合など、アニメーションといった連続データの表示などが、プロセッサスピードに比例して速くなってしまうという問題も起きていた。

一方、より“予測可能な”リアルタイムシステムを目指す、新しい分散リアルタイムシステムでは、プログラムの動作に対する時間的制約をプログラム上で明示する方法[2]が取り入れられてきている。

また、マルチ・プロセスモデルの欠点のひとつであったコンテキスト・スイッチングのオーバヘッドを減少させるためや、並列プロセッサとの整合性をよくするために、並行プロセスの動作を“スレッド”で記述し、スケジューリングの単位として採用してきている計算のモデルが開発されてきている [7, 18]。

Real-Time Mach では、ARTS カーネルで開発したリアルタイムスレッドの概念と Mach の “C-Thread” パッケージ [5] を融合させた RT-Thead モデルが提供されている。Real-Time Mach では、次のようなリアルタイムスレッドを制御するシステムプリミティブがマイクロカーネルに実装されている。

```
rt_thread_create(task, thread, thread_attr)
        スレッドを生成する。
rt_thread_exit(thread)
        スレッドを終了する。
thread_get_attribute(thread, flavor, old_attr, old_attrCnt)
        スレッドの属性を読み出す。
thread_set_attribute(thread, flavor, new_attr, new_attrCnt)
        スレッドの属性をセットする。
```

RT-Thread のインタフェースは、基本的には、C-Thread の拡張であるが、リアルタイムタスクにおいて、より正確に周期的スレッドを記述ができ、かつデッドライン等の時間的的制約を明示できることが特徴である。

Real-Time Mach では、時間的制約をリアルタイムスレッドの時間属性として、動的に与えることができる。例えば、周期的なスレッドの場合、開始時刻、周期、周期内のオフセット、デッドライン、アボートタイム、最悪実行時間、スレッドの重要度などを指定する。また、非周期的なスレッドの場合、デッドライン、アボートタイム、最悪実行時間、スレッドの重要度などとともに、最悪到着待ち時間を指定する。これらのリアルタイムスレッドの時間属性データを参考に、ハード／ソフトリアルタスクの両方をサポートでき、Mach Timesharing (TS), FP/RR(Fixed Priority Preemptive/RR), FP/FIFO(Fixed Priority Preemptive/FIFO), RM(Rate Monotonic), DM(Deadline Monotonic), EDF(Earliest Deadline First) などのスケジューリングポリシィをアプリケーション側から選択できるリアルタイムスケジューラが実現されている。

[1]一般に、話している人の唇の動きと音声を同期させること。実際、どの程度の時間差まで理解できるかは、ビデオ画面の大きさ、や話者の話し方に大きく依存する。

[2]従来の方法を、プログラムと時間的制約の implicit binding と言い、明示的な方法は、explicit binding という。

3.2 スケジューリング技法

一般に、リアルタイムOSで採用されているスケジューリングの技法は、そのシステムが対象としているリアルタイムタスクの性質、すなわち、静的／動的なタスクセット、周期／非周期的なタスクなどの状況によって大きく左右される。

従来からの組み込み型システムでは、周期的なタスクを静的にスケジューリングする方法として、サイクリック・エグゼクティブモデル (Cyclic Executive Model) [6] が多く使われている。サイクリック・エグゼクティブモデルでは、タスクセットを、まず主サイクルと幾つかのマイナ・サイクルに分け、各リアルタイムタスクをその実行時間を考慮しながら、それぞれのマイナ・サイクルに静的に割り当てていく方式である。タスクセットが、非常に単純なシステムの場合は、タスクの割り当ても可能であるが、すこしでもタスクセットが複雑になったり、タスクの変更が頻繁に起こる場合は、割り当て作業を手動で行ない、再構成をしなければならないなど多くの問題を含んでいる。

一方、最近注目されている手法として、レート・モノトニック (Rate Monotonic) 方式 [10, 9] がある。レート・モノトニック方式は、周期の一番短いタスクに最高の優先順序を与える横取り可能なスケジューリング方式であり、従来のモデルと違いスケジューラビリティの解析システムなどとの整合性が非常に良いのが特徴である。システムに与えられたタスクセットの周期の値が、静的に決定されているものであれば、簡単に、周期データの値から、整数の優先順序に変換でき、従来の Fixed Priority Preemptive スケジューリング方式を代用して使うことが可能である。しかし、タスクが動的に生成され、かつ周期も外界の状況により決定され、前もって整数の優先順序に変換できないような場合には、周期の値をそのまま優先順序とする必要がある。

特に、マルチメディア・データを扱う分散アプリケーションでは、タスクが動的に生成されるので、Real-Time Mach では、周期の値をそのまま優先順序して、スケジューリングポリシィモジュールが実現されている。

また、従来のレート・モノトニック方式の長所としては、各周期タスクが必要とする最悪 CPU 消費時間 (WCET, Worst Case Execution Time) にもどづき、各タスクが必要とする CPU ユーティライゼーション、を算出し、それらの総和と理論的なブレークダウン・ユーティライゼーションと比較できる点である。特に、タスクが比較的単純なモデルでは、スケジューラビリティ解析ツールなどにも適用されている [17]

しかし、この周期的タスクのモデルでは、非周期的タスクに対しては、WCET をもとに必要とする CPU ユーティライゼーションを各非周期タスクごとに算出することがうまくできなかった。このような欠点を克服するため、非周期的タスク・サーバ (aperiodic server) の方式 [16] が開発され、ART に実装されている。この方式では、非周期的タスク・サーバに一定の CPU ユーティライゼーションの持ち時間をある周期に対して与え、その持ち時間が残っている限りにおいて、非周期的タスクを実行させるという方式である。この持ち時間を正しく設定することにより、従来のバックグランド方式や、ポーリング方式などよりも、非常によいレスポンスタイムを非周期的タスクが得られることが分かっている。

3.3 リアルタイム同期プロトコル

従来の並行プロセスにおける同期モデルは、セマフォアを使って実現されたクリティカルセクションに代表されるように、共有資源を利用するタスクをすべて平等に扱い、順番を待っているタスクを FIFO の順にキューイングすることによって、タスクのスタベーション (starvation) を防いでいた。

しかし、このような FIFO のキューイングでは、2.3 節で述べたプライオリティインバージョンが起きてしまうので、クリティカルセクションの問題に関しては、二つの新しい方法を採用している。一つは、無制限のプライオリティインバージョンを回避するための、一連のプライオリティ継承プロトコル [13] を使う方法である。もう一つは、クリティカルセクション自体を、再開始可能とするような RCS(Restartable Critical Section) プロトコル [21] を使う方法である。

プライオリティ継承プロトコルでは、低い優先順序のタスクがクリティカルセクションに既に入っている時に、もしもそれよりも高い優先順序のタスクがクリティカルセクションに入ろうと試みた場合に、単に待ち行列にキューイングするだけでなく、この高い優先順序を既にクリティカルセクションに入っているタスクに継承させ、クリティカルセクションを実行させるようにする。さらに、キューイングに関しても、FIFO の順でなく、デッドラインの順にキューイングを行う。一時的に高い優先順序をもらった優先順序の低いタスクが、クリティカルセクションの実行を完了すると、もとの低い優先順序にもどり、待ち行列のなかで一番優先順序の高い (もっともデッドラインの近い) タスクが起動されて、クリティカルセクションの実行を開始する。

一方、RCS プロトコルは、比較的大きなクリティカルセクションの制御に適している。プライオリティ継承プロトコルでは、最悪時に、少なくとも、低い優先順序のタスクがクリティカルセクションを終了するまで、待たなければならない。この RCS プロトコルでは、高い優先

順序のタスクがクリティカルセクションに入る際、もし低いタスクがクリティカルセクション内にいた場合、そのタスクをアボートし、再び待ち行列の中に優先順序にそって再キューイングし、すべての共有変数をクリティカルセクションの開始状態に復帰させて、高いタスクの実行を開始しようとする方法である[3]。この方式の場合、最悪待ち時間は、低いタスクのアボート、再キューイングに要する時間、共有変数の状態を復帰させる時間だけに比例し、算出することが可能となる。

また、これらの基本的なプライオリティ継承プロトコルは、単一プロセッサ上でプライオリティインバージョンを解決できる方法であるが、一般のマルチプロセッサ上や複数ホスト間でのプライオリティインバージョン問題に関しては、これらのプロトコルをさらに改良する必要がある。

4 リアルタイム通信プロトコル

リアルタイム通信プロトコルに対する関心は、マルチメディア・データを扱う分散アプリケーションの要求から、非常に高くなってきている。新しいリアルタイム通信プロトコルとして開発されてきたトランスポートレベルの代表的なプロトコルとして、VMTP (Versatile Message Transaction Protocol)[3], XTP (Express Transport Protocol)[4], ST II (Stream Protocol II)[23], SRP (Session Reservation Protocol)[2], CBSRP (Capacility-based Session Reservation Protocol)[22], などがある。

VMTP は、スタンフォード大学で開発された V-Kernel 用に作られたプロトコルで、転送効率やプロトコルエンジンの実装効率が良くなることをめざして設計されている。トランザクション指向、マルチキャスト、選択可能再転送機能 、8-ビットのパケット優先順序等といった特徴をもっている。

また、VMTP よりさらに転送効率の向上、高速なプロトコルエンジンの VLSI 実装をめざした XTP プロトコルでは、ウインドウ・フロー制御、レート制御、自動繰り返し制御、選択可能再転送機能などが付加されている。また、パケット内の制御情報の配置なども、VLSI での制御が簡単になるように工夫されている。現在までのところ、XTP 3.6 版の C 言語による実装がテストされている。

一方、マルチメディアのコンファレンスシステムにおけるビデオや音声データの転送を、広域ネットワークや LAN 上で時間的制約を満たすように開発されたのが ST II, SRP, CBSRP などのプロトコルである。ST II は、BBN 社が ARPA ネット上で開発した ST(Internet Stream) プロトコルを TWBNet(Terrestrial Wideband Network) 用に拡張したもので、T1-専用回線 (1.5Mb/秒) で接続されているネットを DQDP(Distributed Queue Dual Bus) プロトコルを使って転送制御をしている。

ST II は、コネクションオリエンテッドなプロトコルで、ビデオ、音声などの他に、共有ワークスペース等の通常データもマルチキャストできる機能を提供している。アプリケーションは、メディア転送をする際、ストリームを設定し、そのストリームに対して、データの平均サイズ、バースト時のスループット、ラウンドトリップの最大許容遅延時間、遅延時間の分散、エラー率などを指定することができる。

SRP は、カリフォルニア大学バークレイ校の ICSI 研究所 (International Computer Science Institute) で開発されているプロトコルで、ST II と同様に、コネクションオリエンテッドなプロトコルであるが、通常の IP ネットワーク上での、マルチメディアデータ転送をめざしている点が特徴である。

CBSRP は、CMU で開発されているプロトコルで、ST II、SRP と同様に、コネクションオリエンテッドなプロトコルであるが、単一の FDDI ネットワーク [14] 上で、FDDI プロトコルにおける同期と非同期転送モードをうまく利用し、ART 上に実装され [18]、現在、Real-Time Mach に移植中である。

このような単一 LAN 上のプロトコルでは、ST II や SRP などで行なわれているゲートウェイ上の資源確保などを保証することが必要でない。また、新しいセッションを開始する際、もし他のセッションの QOS を低化させても不都合が起きなければ、CPU、ネットワーク資源などの再配分を行なって、新しいセッションに対する QOS を提供できるよう動的に QOS を協調制御する機能がある。

以上、いくつかの新しいプロトコルを述べたが、いずれのプロトコルに関しても、時間的制約に関したメッセージ転送の解析がタスクスケジューリングの解析と統合して行えるまでには達していないのが現状である。

5 まとめ

我々は、マルチメディア統合環境プロジェクトにおいて、分散マルチメディア環境を提供できる共通ソフトウェアプラットフォームを構築するためのカーネルとして Real-Time Mach の研究開発評価を進めている。Real-Time Mach には、いくつかの新しいリアルタイム処理の試みがなされているが、まだまだ多くの課題が残されている。特に、時間的制約に基づく資源管理の方式を採用しているが、次のような問題はシステムデザイナにとって、興味ある根本的な課題である。

- Sharing vs. Isolation

[3] トランザクションシステムにおける、アボートとアンドウの概念に似ているが、アボートしたタスクを待ち行列に戻す点が異なる。

- Lazy vs. Eager Evaluation
- Fair vs. Deadline- (or Time-) driven
- Fixed vs. Programmable Policy

現在、いろいろな商用リアルタイムシステムが使われているが、分散マルチメディア環境や分散リアルタイムアプリケーションを構築するには、十分なOS機能や、解析・予測可能性を持っていない。一方、多くの実験研究用リアルタイムシステムが研究されてきているが、まだまだ、実際の分散システムで利用するまでには、多くの課題が残されているのが現状である。しかし、これからの高度分散システムの構築に関しては、リアルタイム処理技術はもっとも重要な技術のひとつであり、今後とも新しい研究成果が期待される。

6 謝辞

本プロジェクトを遂行しするにあたり、協力して頂いたカーネギーメロン大学のARTグループ、Machグループの皆様、慶應義塾大学の開放型基盤ソフトウェア研究開発評価事業「マルチメディア統合環境基盤ソフトウェア」プロジェクトの皆様に感謝致します。

参考文献

[1] M.J. Accetta, W. Baron, R.V. Bolosky, D.B. Golub, R.F. Rashid, A. Tevanian, and M.W. Young, "Mach: A new kernel foundation for unix development", *In Proceedings of the Summer Usenix Conference,*, July, 1986.

[2] D. Anderson, R. Heritwich, C. Schaefer "SRP: A Resource Reservation Protocol for Guaranteed-Performance Communication in the Internet", TR-90-006, ICSI, Berkekey, February 1990.

[3] D.R. Cheriton, "VMTP: Versatile Message Transaction Protocol", Computer Science Department, Stanford University, 1987.

[4] Protocol Engines Inc., "XTP Protocol Definition, Rev. 3.5", PEI-90-120, September 1990.

[5] E. C. Cooper, and R. P. Draves, "C threads", Technical report, Computer Science Department, Carnegie Mellon University, CMU-CS-88-154, March, 1987.

[6] P. Hood and V. Grover, "Designing real time systems in ADA", Tech Report 1123-1, SofTech, Inc., January, 1986.

[7] Y. Ishikawa, H. Tokuda, and C. W. Mercer, "Object-Oriented Real-Time Language Design: Constructs for Timing Constraints", In *Proceedings Joint ACM OOPSLA/ECOOP'90 Conference on Object-Oriented Programming: Systems, Languages, and Applications*, October 1990.

[8] C. Nicolaou, "An Architecture for Real-Time Multimedia Communication Systems", *IEEE Journal of SAC*, Vol. 8, No. 3, April 1990.

[9] J. P. Lehoczky, L. Sha, and Y. Ding, "The rate-monotonic scheduling algorithm: Exact characterization and average case behavior", Department of Statistic, Carnegie Mellon University, 1987.

[10] C. L. Liu and J. W. Layland, "Scheduling algorithms for multiprogramming in a hard real time environment ", *Journal of the ACM*, Vol.20, No.1, 1973.

[11] A. K. Mok, "Fundamental Design Problems of Distributed Systems for the Hard-Real-Time Environment", *PhD thesis*, Massachusetts Institute of Technology, May 1983.

[12] S.J. Mullender, G.V. Rossum, A.S. Tanenbaum, R. Renesse and H. Staveren, "Amoeba: A Distributed Operating System for the 1990s", *IEEE Computer* Vol.23, No.5, May, 1990

[13] R. Rajkumar, "Task Synchronization in Real-Time Systems", Ph.D. Dissertation, Carnegie Mellon University, August, 1989.

[14] F. Ross, "FDDI - A Tutorial", *IEEE Communication Magazine*, May, 1986.

[15] L. Sha, R. Rajkumar, and J. P. Lehoczky, "Priority inheritance protocols: An approach to real-time synchronization", Technical Report CMU-CS-87-181, Carnegie Mellon University, November 1987

[16] B. Sprunt, L. sha and J. P.Lehoczky, "Aperiodic Task Scheduling for Hard-Real-Time Systems", *The Journal of Real-Time Systems*, Vol.1, No.1, 1989.

[17] H. Tokuda and M. Kotera, "A real-time tool set for the ARTS kernel", Proceedings of 9th IEEE Real-Time Systems Symposium, December, 1988.

[18] H. Tokuda and C. W. Mercer, "ARTS: A distributed real-time kernel", *ACM Operating Systems Review*, Vol.23, No.3, July, 1989.

[19] H. Tokuda, C. W. Mercer, Y. Ishikawa, and T. E. Marchok, "Priority inversions in real-time communication", In *Proceedings of 10th IEEE Real-Time Systems Symposium*, December, 1989.

[20] H. Tokuda, T. Nakajima, and P. Rao, "Real-Time Mach: Towards a Predictable Real-Time System", In *Proceedings of USENIX Mach Workshop*, October, 1990.

[21] H. Tokuda and T. Nakajima, "Evaluation of Real-Time Synchronization in Real-Time Mach", In *Proceedings of USENIX Mach Workshop*, October, 1991.

[22] H. Tokuda, Y. Tobe, S.T.-C. Chou and J. M. F. Moura, "Continuous Media Communication with Dynamic QOS Control Using ARTS with and FDDI Network," In *Proceedings of ACM SIGCOMM '92*, August, 1992.

[23] C. Topolcic, "ST II", In *Proceedings of 1st International Workshop on Network and Operating System Support for Digital and Audio and Video*, UCB-ICSI TR-90-062, November 1990.