計算機アーキテクチャ 117－3
（1996．3．5）

リアルタイム信号処理用マルチプロセッサにおける動的負荷分散方式の評価

高橋正人[†] 青山和弘[††]
高野博行[††] 宮田裕行[†]

近年、リアルタイム信号処理システムにおいて、その発生が事前に予測不可能な数からなるタスクを、実時間処理する技術が求められている。このことを背景に、SCIを利用した分散共有メモリを持つマルチプロセッサシステムにおける、実時間処理を要するタスク数変動時のシミュレーション結果と解析を行なった。結果として、リモートノードでのタスク処理の実施に際しては、資源予約機構の有無が重要であることが判明した。

An Evaluation of Dynamic Scheduling Algorithms of Real-Time Signal Processing on a Multi-Processor System

MASATO TAKAHASHI,[†] KAZUHIRO AOYAMA,[††]
HIROYUKI TAKANO[††] and HIROYUKI MIYATA [†]

This paper describes a performance study of several task scheduling algorithms used in the design of distributed real-time systems. The study is based on a simulation model of a distributed shared memory multiprocessor using Scalable Coherent Interface. The result shows that a shecduling algorithm with a resource reservation mechanism improves system performance significantly.

1. 背景と目的

近年、追尾アルゴリズムの一つである Multiple Hypothesis Tracking (MHT)[1] が、多目標追尾、ロボットビジョン、自動航行などの領域で注目を集めている[2]。MHT は、目標数やクラッタ (不要信号) の状況により、タスク数が急激に増加する特徴を持つ。従来は、計算機能力の制約により、十分なタスク数をサポートしきれずに、MHT の特徴を生かしきることが困難であった。近年、マルチプロセッサ技術の発達等により、MHT の特徴を生かしきる計算機能力の実現が可能となりつつある。

本稿におけるプラットフォームの構成は、数十台の CPU を Scalable Coherent Interface (SCI, IEEE Std 1596-1992)[3] ベースのリングトポロジで結合する分散共有メモリによるマルチプロセッサシステムを選定した。このシステムにおいては、センサからの信号入力が、あるノードに入力されるが、あるフレームタイムにおいて、そのノードのキャパシティ以上の計算負荷が発生することがある。リアルタイム性を確保するためには、他のノードへの負荷分散をいかに適切に行なうかが技術課題となる。

以下本稿では、MHT の説明、我々が設計しているハードウエアの説明、分散プラットフォーム上での動的負荷分散方式の説明、シミュレーションによる評価条件と評価項目、シミュレーション結果とそれに基づく考察、および結論について記す。

2. MHT

本システムにおける目標追尾には MHT アルゴリズムを採用する。MHT アルゴリズムにおいては、目標からの信号やクラッタのあらゆる組み合わせを仮説として発生する。仮説数は指数級数的に増加してゆく。一般的には、仮説にはカルマンフィルタ等による予測との一致性からある確率が持たされ、この確率値が閾値以下になった仮説は破棄される。このようにして仮説数の増加を防いでいる。しかし、破棄される仮説が少ない程 (多くの仮説を保持できる程) MHT の特徴が生かされる。

3. ハードウエアプラットフォーム

前節のようなアプリケーションを駆動させるには、ユニプロセッサでは、計算パワーが圧倒的に不足する。そこで、高速演算のできるプロセッサを数十台結合する並列処理が求められた。

† 三菱電機 (株) 情報技術総合研究所
Mitsubishi Electric Corp. Information Technology R&D Center
†† 三菱電機 (株) 鎌倉製作所
Mitsubishi Electric Corp. Kamakura Works

チップには、プログラマビリティの良さと、コモディティの良さから、DEC 社の Alpha AXP を採用する。相互結合網には (1) 低遅延性 (2) 分散共有メモリの構築が容易 (3)IEEE 規格としての将来性から、SCI を採用する。分散共有メモリシステムが構築できればプログラマビリティも良くなる。OS は、VxWorks for Alpha を並列 OS に改修して使用する。VxWorks は、Alpha との相性と、リアルタイム性から採用する。SCI 結合の際のトポロジとしては、リング構成を採用する。これは、この規模のシステムではリング構成が比較的性能価格比が良好であると判断されたためである[4)5)]。図 1 に全体構成図を示す。

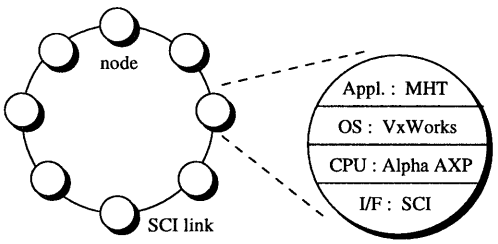

図 1　全体構成図

4. 動的負荷分散方式

負荷分散の一般論として、文献[6)] によれば、負荷分散方式は表 1 のように大きく分類される。(A などの簡略表記は本稿筆者による)、A, B の領域は多くの研究がある。それら静的な負荷分散アルゴリズムの研究は一般には動的な負荷分散アルゴリズムに適用しても最適でないことが多いと述べられている。また、C の領域も比較的研究は進展している。一方、D の領域は、チャレンジングな領域とされている。

表 1　負荷分散方式の分類

	静的	動的
集中型システム構成	A	B
分散型システム構成	C	D

分散したノードを持つシステムにおいて、動的に (実行時に) タスク実行ノードを割りあてる方式を検討するという本稿での研究課題は、この表 1 における D の範疇となる。

4.1 分散型システムでの動的負荷分散

分散型システムでの動的負荷分散は、基本的に二つの機能から構成される。タスクをどのノードに負荷分散するかを決定するグローバルな機構と、実際にタスクが到着した時点でタスクが自ノードで実施した場合、そのタスクのデッドライン前にタスクを完了できるかどうかを判断する機構である。後者を guarantee 機構という。

一般に、分散型システムにおいて、各ノードの負荷情報を取得するためには伝達遅延が発生するため、得られた情報は厳密に言うと古い (現時点の状況を反映してない) 情報となっている。また、できるだけ正確な情報を得ようとすると、頻繁に情報取得を行なうこととなり、ネットワーク負荷を増大させるという欠点も発生する。さらに、MHT においては、ある状況で急激に仮説すなわちタスク数が増大し、システムのキャパシティを超えることも考えられる。そのような状況で、適切に低優先度の仮説を棄却し、高優先度の仮説を生かすことも考えねばならない。これらの要請において、最も適切な動的負荷分散方式を考案、採用することが目標である。

4.2 分散型システムでの動的負荷分散方式の分類

すでに述べたように、負荷分散に関して、分散型システム構成、動的という二つの軸によって範囲を限定してきたが、本副節では、さらに範囲の限定化を行なう。

まず、フォルトトレランス性を重視する観点から、マスター・スレーブ型システムは可能性から除外され、シンメトリ型システムを対象とする。つまりどのノードも同等の機能を持つものである。自ノードで guarantee できないと判断された (リジェクトされた) タスクに関しては、他に適切なノードを探し、そのノードにタスクを渡す。

また、あるノードの負荷が閾値以上になり、負荷を他ノードに与えることを希望した時点で、負荷分散のトリガを引くか、あるいは、あるノードの負荷が閾値以下になり、負荷を引きうけることを希望した時点で負荷分散のトリガを引くかの検討が必要である。前者は、sender-initiate、後者は、receiver-initiate と呼ばれる。前者は、リアルタイム処理に適し、後者は、負荷の均等化に貢献する。今回の我々の目標はリアルタイム処理であるため、前者を採用する。

ここまで、負荷分散に関して、分散型プラットフォーム・動的・シンメトリ・sender-initiate と限定化を行なってきたが、この限定された領域においても、多くの負荷分散方式が考えられる。

そこで、以下に動的負荷分散の主要機能 (資格判定、決定、資源予約) に基づく分類を行なった。

「資格判定」とは、あるノードのプロセッサがそのフレームタイム内での時間的余剰 (surplus) と、割付ようとするタスクの実行の予想所要時間、の比較によって、そのノードが、該当タスクを実行可能かどうかを知ることである。これによって、あるノードが、候補ノードとなりうるかどうかが判定される。

「決定」とは、資格判定により、候補ノードが複数あった場合に、どのノードに実際にタスクを渡すかを決定する処理である。

「資源予約」とは、ネットワーク遅延を考慮して、資格判定時あるいはそれ以前に、割付ける可能性のあるタスク CPU 時間の資源を予約しておく処理である。これに関する説明と分類は次副節でより詳しく行なう。

さて、ここで述べた、「資格判定」も、「決定」も、

いずれかのCPUパワーを使用して行なわれなければならないことは明らかである。これを、

- タスクを割り当てを実施しようとしているノードのCPUパワーを用いて計算する
- タスクが割り当てられようとしてるノードのCPUパワーを用いて計算する

可能性がある。これを表にすると、表2の4方式に分類される。

表2　資格判定機構と決定機構による分類

		資格判定機構	
		集中型	分散型
決定機構	集中型	CC	CD
	分散型	DC	DD

決定を分散型として行なうことは、次のような欠点があると考えられ、考慮からはずされた。決定を分散型として行なうためには、ノード間の相互通信が増大・複雑化し、開発コストの上昇と性能の低下を招くと考えられたためである。ただし、分類そのものは分類の完全性の利点のため、残してある。

一方、資格判定を 集中型として行なうことも、分散型として行なうことも利点を持つと考えられた。前者は、各ノードの負荷情報を採取し、タスク割り当てにおける主体(ノード)が、自分のCPUパワーを使って資格判定とを実施する。構造が単純でわかりやすい。

後者は、タスク割り当てにおける客体(ノード)が、それぞれにCPUパワーを用いて資格判定を実施する。そのため、タスクを他に割り当てたいほどに高負荷であることが明瞭なタスク割り当て主体をより高負荷にすることなく、資格判定が実施される。さらに資格判定が並行に行なわれる利点を享受する可能性があると考えられた。

4.3 予約機構の重要性

表2でのCCとCDの範疇において、さらに予約機構の有無によって、各2方式、合計4方式を検討の対象とした。なぜ予約機構が重要かと判断したかを次に示す。分散型システムプラットフォームにおいては、あるノードAが他ノードBの情報を取得できた時には、実際にはノードBのその情報は既に変化している可能性がある。これはネットワークのデータ転送遅延のためである。さらにその情報に基づいて、ノードAがノードBに何らかの働きかけを実施した場合に、ノードBがその働きかけを認識するまでにも、ネットワーク遅延が生じている。最悪のケースでは、割付けられようとしてタスクが到着したときに、ノードBでは状況が変化していて受けつけれられず(receiver candidate によるリジェクト)、そこからまた割付ノードを探し、さらに、同様のことが繰り返される可能性がある。たらいまわしのような事態である。

このような理由から、「集中型資格判定においては、負荷情報取得時に」、「分散型資格判定においては、資格判定による資格認定時に」、タスクの割り付けが将来なされるものと仮定して資源を予約してしまう、という手法が有効であると考えられた。これによる利点は、ほぼ確実に割付決定先で受け入れられることである。「ほぼ」確実というのは、万が一ネットワーク遅延のため時間がきわめて遅れて、デッドラインを超えた場合は受けつけられないことがありうることを言う。また、この予約方式の欠点は、機構が複雑になることである。これは、予約したものの、実際には、より適切なノードが発見され、先述のノードにはタスクを割付ない場合には、予約された資源は、そのフレームタイムの間に無駄になる。これを避けるためには、使用されない予約を解除する機構も必要となり、機構全体が複雑となる。この複雑化にみあう guarantee ratio (guaranteeされるタスク数と投入されたタスク数の比率)[6] の向上、あるいは、負荷分散完了時間の短縮が観測されるかどうかが関心の一つである。

4.4 実際の動的負荷分散アルゴリズム

下記の4種類の分散型システム上での動的負荷分散アルゴリズムを作成した。2つのアルファベットの綴りで簡略表記する。表記は、左から次の意味を持つ。

- 資格判定機構：Cは集中型資格判定、Dは分散型資格判定を示す
- 資源予約機構：Rは資源予約機構有り、Nは無しを示す

表3　資格判定機構と資源予約機構による分類

		資格判定機構	
		集中型	分散型
予約機構	有	CR	DR
	無	CN	DN

すると、各々のアルゴリズムは、CR、CN、DR、DN、と表記される。対照表を表3に示した。

5. シミュレーション評価

上記のアルゴリズムにつきシミュレータを構築し、各種のパラメータセットにおいて性能評価を行なった。本節では、シミュレーションの概要、シミュレーションのパラメータおよび、評価基準について述べる。

5.1 シミュレーションの概要

離散型シミュレータ構築支援ツールSES/workbench[7]を用いて構築した。並行事象を扱うシミュレーションはデバッグが非常に困難であると考えられるが、SESは、アイコニックなプログラミングが可能なこと、固有のアニメーション機能を持つこと等によりデバッグがしやすく、シミュレータ構築が効率よくできる。シミュレータは4方式につき個別に作成した。図2にSES/workbenchで作成したシミュレータの一部を例示する。

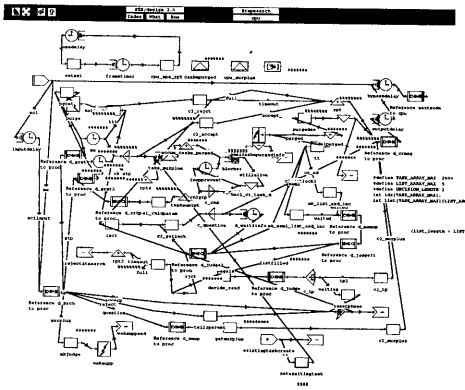

図2　構築したシミュレータの一部

5.2 パラメータ

パラメータは、通信における遅延、各ノードでのメモリアクセス、数理判断、分岐、ロックの取得、メッセージ生成、タスクパラメタ生成、などの実行時間を指定する。また、各周期毎に入力されるタスク数、タスクの実行所要時間の確率分布、タスクの優先順位の混合率等を指定する。

システムのノード数は 4, 8, 16, 32, 64 を対象とする。タスク投入配置として、システムノード中、対称位置にある 2 ノードのみに配置した場合、システムノード中、4 ノードに 1 ノードの割合で配置した場合 (1:4)、同様に 2 ノードに 1 ノードの割合で配置した場合 (1:2)、全ノードに均等に配置した場合 (1:1) を想定した。タスク入力配置を変数とすることにより、アーキテクチャ設計段階で、マルチセンサからの入力ノード配置に設計設計選択の指針が得られると考えられた。同時に、MHT での、仮説 (すなわちタスク) が新たな仮説 (タスク) を生成する特徴を反映させていると見ることもできる。つまり、タスク入力配置が 1:1 では、均等に新規タスク生成が生成されている状況であり、1:4 等では新規タスク生成にノード間の偏りが見られる場合に相当する。

これらの条件のもとで、1 フレームタイムの応答特性を見る。

5.3 評価基準

一般に、リアルタイムシステムにおける性能とは、個々のタスクがデッドラインをどれだけ守れたか、という割合で示される。これを guarantee ratio と呼び、第一の評価基準とする。加えて、今回は、個々の負荷分散方式がどの程度迅速に負荷の分散を実施できたかも観測する。さらに、高優先度のタスクがリジェクトされるという事象が稀に発生することが予想されるが、その割合も観測する。最終的な評価基準はこれらの総合判断である。

6. 結果と考察

本節では結果に基づいて明らかとなった、各負荷分散アルゴリズムの特性とその比較を行なう。

既にシステム内に存在しているタスク数と、その後注入される (あるいは発生すると考えてもよい) のタスク数の割合、に関する条件を比較のために 4 つ想定した。表 4 に示す。

表4　タスクの初期状態と新規投入状況

	初期状態		新規投入	
優先度	高	低	高	低
条件 0	0	0	100	0
条件 1	0	100	50	0
条件 2	50	50	500	0
条件 3	0	100	100	0

(単位 %)

比較の条件として、まず、条件 0 では、空の状態のシステムにシステム能力の 100% の量の高優先度タスクが投入されたときの、負荷分散完了までの所要時間を見る。

次いで、条件 1 では、低優先度タスクでフル定常状態のシステムに、システム能力の 50% の量の高優先度タスクが投入されたときの、負荷分散完了までの所要時間を見る。この条件は、先住タスクの半分がパージされ、後からの高優先度タスクに置換されるシナリオであるが、最終的に低優先度タスクと高優先度タスクが半々になるため、有資格者探索は以降に述べる条件に比べて緩やかな条件であることが予想される。

次いで、条件 2 では、低優先度タスクと高優先度タスク (各々 50% ずつ) でフル定常状態のシステムに、システム能力の 50% の量の高優先度タスクが投入されたときの、負荷分散完了までの所要時間を見る。この条件は、先住タスクの半分がそっくり、後からの高優先度タスクに置換されるシナリオであり、最終的には、全システムが、高優先度タスクで占有されている状態である。負荷分散の最終段階では有資格者探索において、有資格者がなかなか見いだせない状態が発生することが予測される。

最後に、条件 3 では、低優先度タスクでフル定常状態のシステムに、システム能力の 100% の量の高優先度タスクが投入されたときの、負荷分散完了までの所要時間を見る。この条件では、すべての先住タスクはパージされ高優先度タスクにすべて置換されるシナリオであり、負荷分散の最終段階では有資格者探索において、有資格者がなかなか見いだせない状態が発生することが予想される上、パージする量も最も多く、負荷分散に要する CPU 負荷が最も高い条件であると予想される。

動的負荷分散の 4 方式が、上記の各条件において、あるフレームタイム内に負荷分散完了までに要した時間

をグラフ化したものを図3に示す。横軸は各条件、縦軸は負荷分散完了所要時間とした。ここでは、ノード数は、16に固定され、タスク入力配置は、1:4である。シミュレーション評価前には、予約機構を持つ方式は、ネットワークトラフィックの多さから、負荷分散完了までの所要時間が多くかかるとも予測された。また、予約機構ない方式は、たらいまわし的な事態により、最終的にguaranteeされうる受け入れノードを見いだすまで時間がかかるとも予想された。

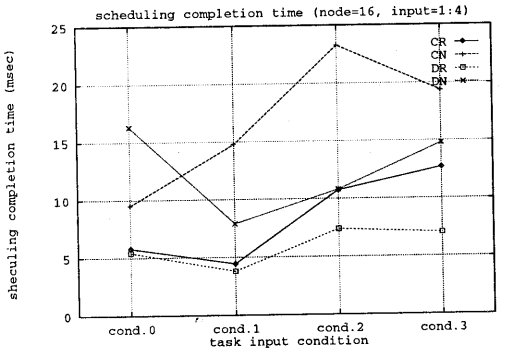

図3 各方式の負荷分散完了時間(16node)

図3の結果は、予約機構を持つ方式の方が予約機構を持たない方式よりも優れる、ことを示した。また、予約の有無が同じ場合は、分散型判定が、集中型判定より優れていることがかわかる。その両方を備える方式としてDR方式が最も良い性能を出していることが判明した。なお、このパターンは、ノード数あるいはタスク投入配置を(5.2節で示した範囲内において)変化させた場合も、常に観測された。

リアルタイムシステムにおいて、負荷分散完了までの所要時間が小さいことは重要である。なぜなら、ノード数が増加した場合にデッドラインを守れなくなるタスクの発生を防ぐからである。実際、32ノード、64ノードにおいては上記の4条件の一部、タスク投入配置一部で、デッドラインを守れなくなる方式が発生した。それは、16ノードにおいても負荷分散完了までの所要時間の比較的大きいCN, DNであった。どちらも予約機構を持たない方式である。リアルタイムシステムである以上、高優先度タスクは可能な限りデッドライン内に完遂されることが意義あることとされるため、予約機構を持つ方式が備えている、負荷分散完了までの所要時間が小さいという傾向は有利な特徴であると考えられる。

予約機構を持たない方式が、負荷分散所要時間が大きいという傾向の原因を検討した。投入タスクについてreceiver candidateによるリジェクトの回数を観測し、投入されたタスク全体に対するタスク数の割合から、度数分布表を作成した。表5は条件0において観測された結果を、表6は条件2において観測された結果を基にしたものである。この結果、予約機構を持たないCN, DNの方式において、receiver candidateによるリジェクトの発生頻度が顕著であり、負荷分散完了までの所要時間に大きな影響を与えていることが明らかとなった。表5によると、DN方式において、投入全タスク数の50%以上が、上述のリジェクトによるたらいまわしを経験しているということがわかる。

表5 receiver candidateによるリジェクト頻度[条件0]

条件0	CR	CN	DR	DN
リジェクト回数0	100.0	75.0	100.0	45.3
リジェクト回数1	0.0	21.9	0.0	29.7
リジェクト回数2	0.0	3.1	0.0	15.6
リジェクト回数3	0.0	0.0	0.0	6.3

(単位%)

同様に、予約機構を持たないCN方式についても、条件2において、receiver candidateによるリジェクトが頻発していることを表6は示している。

表6 receiver candidateによるリジェクト頻度[条件2]

条件2	CR	CN	DR	DN
リジェクト回数0	100.0	75.0	100.0	87.5
リジェクト回数1	0.0	9.4	0.0	6.3
リジェクト回数2	0.0	6.3	0.0	6.2
リジェクト回数3	0.0	0	0.0	0.0
リジェクト回数4	0.0	6.3	0.0	0.0

(単位%)

ここまでの16ノードシステムに関する検討に続いて、CR, CN, DR, DNの各方式を備えるシステムについて、台数効果としての性能を比較した。タスク配置投入は1:4で固定。条件3を用いた。図4にその結果を示す。

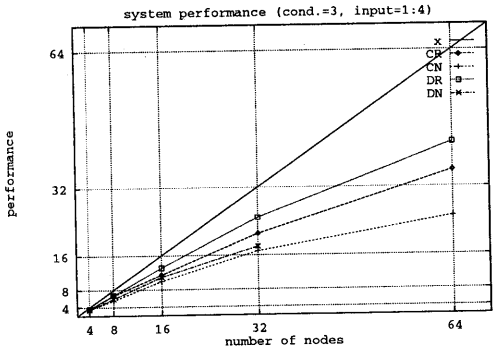

図4 各方式採用システムの台数効果

この結果から、4ノードから64ノードまでシステム構成のいずれにおいても、4方式のなかでは、DR方式が最適と考えられる。その理由は次の2点すなわち、(1) 予約機構のため、再試行がないことがネットワーク負荷に有効に作用する、(2) 分散判定方式のメリットが

生かされた、と考えられた。

なお、予約機構のない二つの方式に関しては、32、64ノードの値は、実は、高優先度タスクのguarantee ratioが100%でなかった。これは、この条件下でのリアルタイムシステムの動的負荷分散機構として不適切であることを示している。DR方式について、32ノード時、98.4%, 64ノード時97.3%, DNについては32ノード時90.6%であった。その他のケースはすべて100%である。

これまでの検討で最適と考えられた動的負荷分散方式DRについて、さらに、タスク入力配置を1:1, 1:2, 1:4, 2-node-only に変えて、システムの拡張性を調べた。ここで採用した条件は条件2である。図5にその結果を示す。タスク入力配置1:1と1:2の結果はほぼ重なっている。

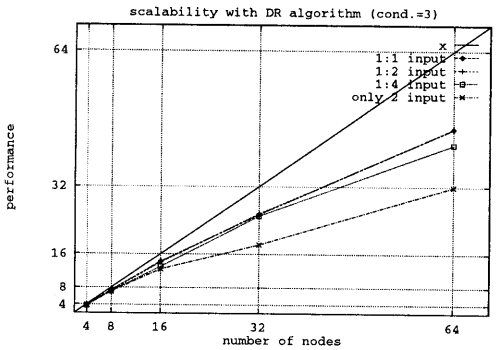

図5　DR方式使用時のタスク入力配置の影響

タスク入力配置に関しては、2-node-only の方式が性能面で劣っている。入力ノードでのシリアライズされた処理がボトルネックになっているものと考えられた。また、マルチセンサという観点からより現実的なタスク入力配置である1:4などを想定した場合には、8あるいは16ノードシステムにおけるDR方式使用システムの有効性が考えられる。このことより、タスク入力配置が1:4の、8ノードから16ノードの単方向リングを基礎とし、それら複数のリングをスイッチ等で結合した構成の実現可能性が大きいことが示唆された。

7. 結　　論

分散型システムをプラットフォームとするリアルタイムシステムにおける動的負荷分散方式はいまだ確立されていない。

本稿では、この領域におけるある条件下での最適な動的負荷分散方式を設計する上で、有効と思われる二つの分類軸(集中型判定 / 分散型判定方式、集中型決定 / 分散型決定および、予約機構の有 / 無)を提案し、それに基づいて分類された各範疇の方式についてシミュレーションによる性能評価を行なった。

その結果、我々の目標であるMHTの実装に関して、現在想定されているハードウエア、OS、アプリケーションに関するパラメータの下で、以下の結果を得た。

- 資格判定方式は、分散型判定が、集中型判定より性能面で優れる。
- 予約機構を持つ方式は、予約機構を持たない方式より、性能面で優れる。
- 単方向リング構成は8ノードから16ノードが適切な上限と考えられる。

以上より、次のことが示唆された。(1)我々の目標とするリアルタイムシステムの動的負荷分散方式として、予約機構を持つ分散型判定方式のアルゴリズムが適している。(2)8ノードから16ノードの単方向リングを基礎とし、それら複数のリングをスイッチ等で結合した構成が適している。

今後は、本稿でのシミュレーション評価結果を、開発中の実機での評価を基に検証するとともに、本稿で、良い評価を得た、予約機構を持つ分散型判定方式のアルゴリズムに関しても、さらなる性能向上をもたらす検討・改良を継続して行なう予定である。

謝辞　本研究に関して、検討会グループのメンバである三菱電機(株)鎌倉製作所の吉岡英明氏、山崎弘巳氏、松本聡氏、山本泰弘氏、宮沢稔氏、花沢徹氏に、貴重な助言を多く頂き、感謝いたします。また、日頃ご指導頂く、三菱電機(株)情報技術総合研究所の下間芳樹情報処理基盤部部長に感謝いたします。

参 考 文 献

1) Reid, D. B., *An algorithm for Tracking Multiple Targets*, IEEE Transactions on Automatic Control, AC-24(6):843-854, December, 1979

2) Miller, M. L., *Implementation Notes for MHT With Multiple target Models*, NEC Research Institute, 1993.

3) IEEE Standard for Scalable Coherent Interface (SCI), IEEE Computer Society, IEEE Std 1596-1992, ISBN 1-55937-222-2, 1992

4) 高橋正人、青山和弘、宮田裕行、菅隆志, SCIを用いたネットワークトポロジの評価, 情報処理学会第50回全国大会講演論文集(6)「ハードウエア・システム」pp.6-25 ～ 6-26 , March 1995

5) 高橋正人、青山和弘、宮田裕行、菅隆志, SCIを構成要素として用いた基本的相互結合網の評価, SWoPP95, 情報処理学会計算機アーキテクチャ研究会研究報告 95-ARC-113-10, Vol.95 No.80 pp.73-80, August 1995

6) Stankovic, J. A., Ramaritham, K., *Tutorial Hard Realtime Systems*, ISBN 0-8186-0819-6, 1988

7) SES/Workbench Reference Manual Release 3.0, SES Inc., Austin TX.