D－SSQにおける先行規制方式の検討

肥塚貞男　渡辺尚　佐藤圭　中西暉　真田英彦　手塚慶一
（大阪大学工学部）

1．まえがき

待ち行列網シミュレーションは本来高い並列性を有するため、これに注目して処理能力向上を目的とした分散型待ち行列網シミュレータの構成が可能である。しかし待ち行列網シミュレーションは確率的に変動する要素を含むため事前に処理計画を立てることができず、プロセッサの実行をいかに制御するかが重要な問題となる。我々は既にこの実行制御方式として先行制御方式を用いたノード分散型待ち行列網シミュレータD-SSQ (Distributed System Simulator for Queueing Network) を提案した。[1] さらにD-SSQの処理能力はプロセッサ数に対し線形に向上するが、先行し過ぎによる無効処理量の増加等のためその絶対値は大きくないことを示した。[2][3]

ここでは、先行に適当な規制を加えることにより無効処理の増加を抑え処理能力向上を図る先行規制方式について述べ、さらにtime-driven方式と先行制御方式との比較、および先行規制方式を拡張したtime+event-driven型シミュレータの提案を行う。

2．先行制御方式

分散型シミュレータの実行制御には、シミュレーション時刻の更新方式とプロセッサ間の時刻同期方式が必要である。時刻更新方式にはtime-driven方式とevent-driven方式の2つがある。[4]

time-driven方式は、シミュレーション管理プロセッサからの時刻更新指示により全プロセッサが一斉にシミュレーション時刻を一定値ずつ更新していく方式である。この方式は、時刻更新方式と時刻同期方式が一致するため、制御が容易になる利点がある。しかし、シミュレーション精度と利用できる並列性は時刻更新幅に依存し、それらの間にはトレードオフの関係が存在する。

event-driven方式は、シミュレーション時刻を次に処理するeventの時刻まで進める方式である。この方式は並列性をより有効に利用しうると考えられるが、その反面、時刻同期アルゴリズムを必要とする。時刻同期方式には矛盾発生を許容する方式と矛盾発生を許容しない方式がある。[5]

D-SSQでは時刻更新方式としてevent-driven方式を、また時刻同期方式として矛盾発生を許容する先行制御方式を採用している。[5] 先行制御方式は並列性を最も有効に利用できると考えられる。

先行制御方式とは、各プロセッサが他のプロセッサとの時刻関係を無視して独立に処理を行い、プロセッサ間通信の際にパケットの到着時刻が受信プロセッサのシミュレーション時刻よりも過去であるという時刻矛盾が発生すると、パケットの到着時刻以降の処理をキャンセルし、シミュレーション時刻をパケットの到着時刻まで戻すことにより時刻矛盾を解消する方式である。

先行制御方式を行うためには次の3つの機能が必要である。

(i) 処理履歴の保存
(ii) キャンセル処理
(iii) 確定時刻の検出

各プロセッサはeventの処理を実行する前に履歴を保存しておき、時刻矛盾が発生した時にはこれをもとにキャンセルを実行する。確定時刻とは、すべて

のプロセッサが少くともこの時刻まで到達したと決定できる時刻である。確定時刻以前に各プロセッサの時刻は戻らないため、確定時刻以前の処理履歴は不要となり、統計収集後消去することができる。

また、D-SSQ における主な分散化オーバーヘッドとしては、処理履歴の保存、キャンセル処理、確定時刻の検出、キャンセルによる無効処理の発生などが挙げられる。

3、先行規制方式

D-SSQ では、先行に適当な規制を加えないと先行し過ぎによる無効処理量が多く、処理能力（1プロセッサシミュレータと分散型シミュレータで同じモデルのシミュレーションを行った時の処理速度の比）の大幅な向上は期待できない。[2] これに対し本章では、時刻矛盾の発生する確率が高くなると先行に適当な規制を加える先行規制方式を提案する。

先行規制は、シミュレーション時刻が先行規制時刻 T_b をこえた場合、ノードの擬似動作処理を中断することによって行われる。擬似動作中断中は統計収集、ガーベジコレクション等の時刻の更新を伴わない処理を行う。

先行規制時刻 T_b を決定するには次の2つの方法がある。[3]

(i) グローバル方式

(ii) ローカル方式

グローバル方式は各プロセッサのシミュレーション時刻を確定時刻から一定範囲内に抑える方式である。すなわち T_b を、

$$T_b = (\text{確定時刻}) + T \qquad (1)$$

とする方法である。ただし T は定数である。

ローカル方式は隣接プロセッサから次に受信するパケットの到着時刻を予想し、その時刻を先行規制時刻 T_b とする方式である。図1を例に上げて説明する。図1では、簡単のため1つのプロセッサに1つのノードが割り当てられているものとする。プロセッサiは、シミュレーションモデル上の出回線 l_{ji} に待ちがある場合には、パケット P_{j1} を送信する時に次のパケット P_{j2} の到着時刻 T_{j2} をのせて送信する。待ちがない場合には、T_{j1} に予想値を加えて送信する。回線に待ちがないことから、予想値は次の到着時刻までの平均時間と伝送遅延の和である。すなわち P_{j1} にのせて送信する次のパケット到着時刻 T_{pj} は

$$T_{pj} = \begin{cases} T_{j2} & (l_{ji} \text{ に待ちがある}) \\ T_{j1} + (1/\lambda_s + 1/\mu_s) & (l_{ji} \text{ に待ちがない}) \end{cases} \qquad (2)$$

となる。ただし、λ_s および μ_s は、シミュレーションモデルにおけるパケット平均到着率、パケット平均処理率をそれぞれ表わす。プロセッサiは、P_{j1}、P_{k1} を受信した時に T_b を

$$T_b = \min(T_{pj}, T_{pk}) \qquad (3)$$

と決定する。

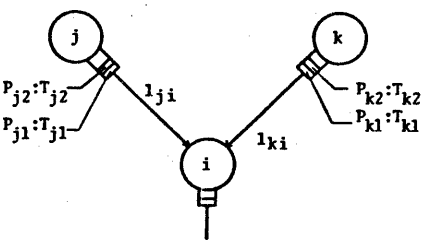

図1　ローカル方式の一例

3、実験及び近似解析による検討

先行規制方式の処理能力を検討するために、実験及び近似解析を行った。

実験に用いたシステムは、マンマシンインターフェース、確定時刻の検出、通信バスの管理等を行うマスタプロセッサ（ＭＰ）１台と、ノードの擬似動作、キャンセル処理等を行うコンピューティングプロセッサ（ＣＰ）５台がIEEE-488 インターフェースにより結合された構成である。システム構成を図２に示す。

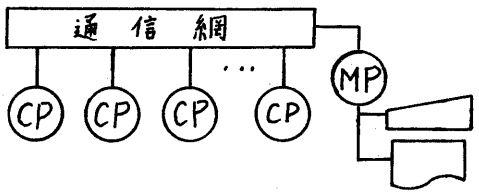

図２　実験システムの構成

このシステムによりシミュレーションを行い、１プロセッサシミュレータに対する処理能力を測定した。

図３に、グローバル方式を用いた場合、及びグローバル方式とローカル方式を併用した場合の、式(1)の規制値Tに対する処理能力を示す。これより、グローバル方式を用いた先行規制を行った場合、処理能力が最も高くなる最適規制値が存在することがわかる。この理由は、規制値Tが大きすぎると、時刻矛盾発生率の増加による無効処理量の増加、及びキャンセル処理時間の増加により処理能力が低下し、一方規制値が小さすぎると、先行規制による遊休時間の増加により処理能力が低下するためである。次に、グローバル方式のみを用いた場合の方がグローバル方式とローカル方式を併用した場合より処理能力が高いことがわかる。さらに、無規制時（T=∞）の値より、ローカル方式の場合、先行規制を行わない場合より処理能力が低下していること

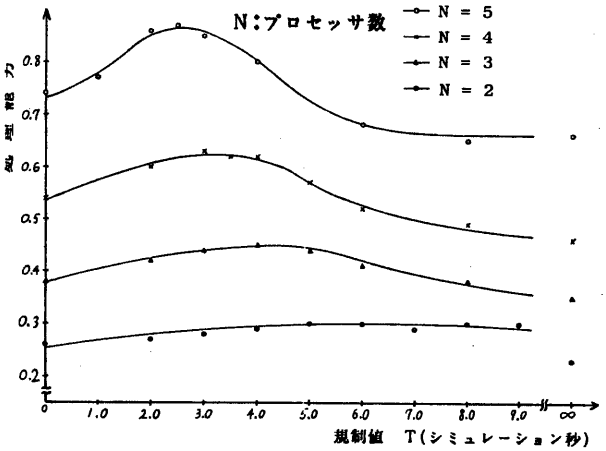

図４　規制値に対する処理能力(実験値)

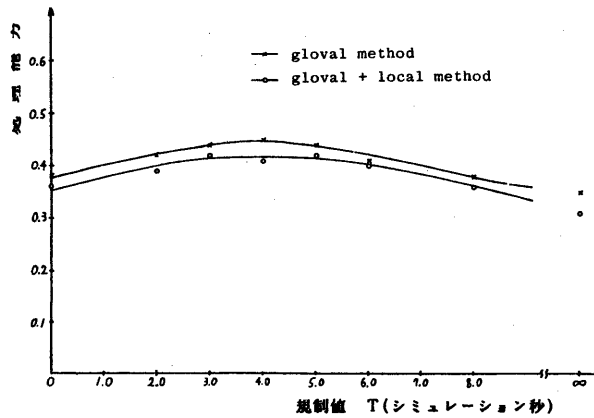

図３　規制値に対する処理能力の変化

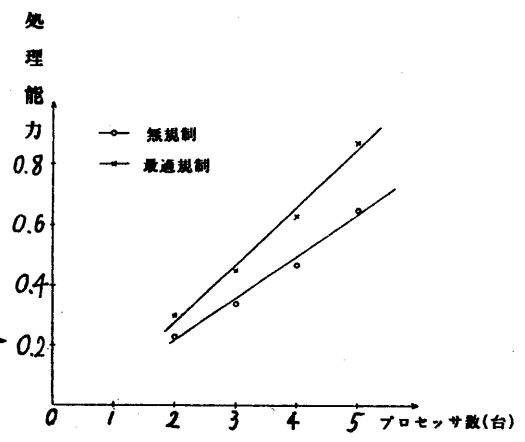

図５　処理能力に対する規制の効果

がわかる。この理由は、ローカル方式を用いて規制値を予想することによるオーバーヘッドが規制による処理能力向上の効果を上回るためである。

図4に、グローバル方式を用いた場合の規制値Tに対する処理能力の変化をプロセッサ数をパラメータとして示す。これより、グローバル方式を用いた場合、先行規制による処理能力改善の効果はプロセッサ数が増加するほど大きくなること、最適規制値はプロセッサ数が増加するほど小さくなることがわかる。

図5に、グローバル方式を用いた先行規制において最適規制値を与えた場合の処理能力を示す。これより、最適規制値を与えた場合の処理能力は、プロセッサ数に対し線形に向上すること、その傾きは無規制の場合より大きくなることがわかる。

以上の実験によりグローバル方式の先行規制が処理能力向上に有効であることがわかったので、グローバル方式に対し近似解析を行う。

近似解析を行うために次の仮定を設けた。

(i) 確定時刻はシステム全体のプロセッサのシミュレーション時刻の最小値に等しい。

(ii) 各プロセッサの確定時刻からの時間幅は各プロセッサにおいて互いに独立で、指数分布に従う。

(iii) キャンセル処理に要する実時間は、キャンセルによって戻るシミュレーション時刻の戻り時間に比例する。

(iv) プロセッサ間の通信時間は無視する。

図6に、プロセッサ数ごとの規制値Tに対する処理能力を示す。これより、実験結果と同様に、プロセッサ数が増加すると先行規制による処理能力向上の効果が大きくなること、最適規制値はプロセッサ数が増加すると小さくなることがわかる。

図7に最適規制値を与えた場合の処理能力を示す。これより、最適規制値を与えた場合の処理能力はプロセッサ数が増加するとほぼ線形に向上することがわかる。

以上より、近似解析の結果は実験値とほぼ同じ傾向を示している。しかし、図4、図6よりわかるように、同じプロセッサ数（N=5）に対する最適規制値は一致しない。これは以下の理由によるものである。

(i) 実際のシステムでは、確定時刻はMPによって一定時間ごとに更新されるため、システム全体のプロセッサのシミュレーション時刻の最小値と一致しない。

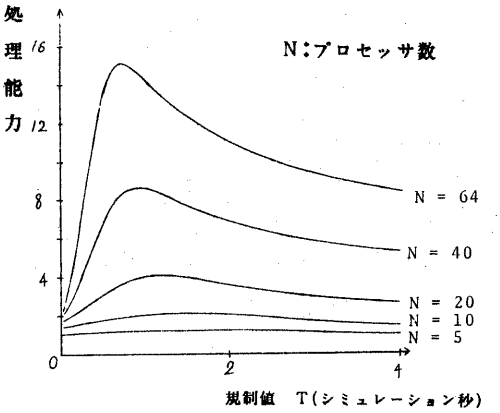

図6 規制値に対する処理能力(解析値)

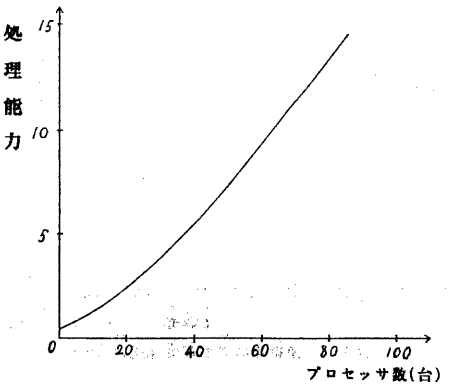

図7 最適規制値を与えた場合の処理能力

(ii)各プロセッサの確定時刻からの時間幅は、実際のシステムでは完全な指数分布とはならず、また独立でもない。

しかし、図4、図6よりわかるようにプロセッサ数が増加すると、規制値Tが最適規制値からずれた場合、処理能力は大きく低下する。最適規制値はシミュレーションモデルによって変化すると考えられるので、最適規制値を正確に予想することが重要である。このため、近似解析を改良し、最適規制値の正確な予想が必要となる。

4. D-SSQ と time-driven 方式の処理能力の比較

time-driven 方式は時刻更新方式と時刻同期方式が一致するという利点から、種々のシステムで用いられている。[5][6] time-driven 方式においてシミュレーション時刻の更新は次のようにして行われる。

まず、シミュレーション管理プロセッサが、シミュレーション時刻を一定時刻幅（1クロック）進める時刻更新指示を全プロセッサに送信する。シミュレーション実行プロセッサは、時刻更新指示を受信すると、現在のシミュレーション時刻から1クロック進んだ時刻までの処理を実行し、処理が終了するとシミュレーション時刻を1クロック進め、シミュレーション管理プロセッサへ実行終了報告を送信する。シミュレーション管理プロセッサは全シミュレーション実行プロセッサより実行終了報告を受信するとシミュレーション時刻をさらに1クロック進める。以上をシミュレーション終了時刻まで繰り返す。ここで、管理プロセッサにおいて決定されるクロック長が小さいと、1クロック中に実行できる処理数が少くなり、並列性を有効に利用できなくなる。また、クロック長が大きいと、結果の精度が悪くなる。

D-SSQ の処理能力と、time-driven 方式を用いたシミュレータの処理能力を比較するために、図2の実験システムを用いて time-driven 型シミュレータを構成した。ただし、シミュレーション管理プロセッサをMPに、シミュレーション実行プロセッサをCPに割り当てる。

この time-driven 型シミュレータとD-SSQにおいて、以下に述べるシミュレーションを行い、その処理能力を比較した。シミュレーションに対して以下の仮定を設ける。

(i)網はプロセッサ数と等しいノード数の完全網である。
(ii)パケットの発生分布は平均発生間隔 $1/\lambda$ sec の指数分布に従う。
(iii)パケット長は平均 $1/\mu$ bit の指数分布に従う。
(iv)回線容量は c bit/sec である。
(v)ノード内の処理時間は無視する。
(vi)ノード内のバッファ容量は十分大きい。

なお、回線利用率は $\rho=\lambda/\mu c=0.4$ 、また、$\lambda=0.4$ である。また、time-driven 型シミュレータでは1クロック=0.1 secとおいた。(結果の誤差=±4%)

図7に D-SSQ と time-driven 型シミュレータの処理能力を示す。これより、time-driven 型シミュレータの処理能力は、プロセッサ数に対し線形に向上すること、その傾きは先行規制を行わない D-SSQ より高いが、最適規制値を与えた D-SSQ とほぼ同じであること、最適規制値を与えた D-SSQ の処理能力は time-driven 型シミュレータよりも高いことがわかる。

ただし、この実験では、クロック長が比較的大きく、十分な精度を得ようとすると、クロック長を実験で用いた値よりも小さくしなければならない。

この場合、time−driven 型シミュレータの処理能力は実験で求めた値よりもさらに低下する。

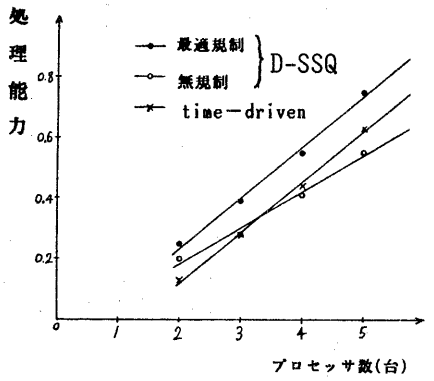

図8　シミュレータの処理能力比較

5. time+event−driven 型シミュレータの提案

D-SSQ においてグローバル方式を用いた先行規制を行う場合、確定時刻の更新が、速かに効率よく行われないと、先行規制による待ち時間の影響が大きくなり、処理能力が低下する。この確定時刻の更新方式には、集中コントロール方式と論理リング方式が既に提案されている。[3] しかし、両方式とも、プロセッサ数の増加とともに確定時刻の更新に時間がかかり、また、通信システムが階層化された場合、効率が悪くなる。これは、基本的に両方式とも、システム中の各プロセッサのシミュレーション時刻を収集し、最小時刻を求める方式であるため、処理が複雑になり、効率が悪化するためである。

そこで、確定時刻の更新を time−driven 方式で行うことにより、確定時刻の更新を効率よく行う time+event−driven 型シミュレータを提案する。

time+event−driven 型シミュレータにおいては、確定時刻の更新はMPにより行われる。各CPは先行規制時刻Tbまでの処理を実行すると、MPに到達報告を送信する。ここで、MPに到達報告を送信した後、キャンセルが起こり、シミュレーション時刻がTb以前に戻ると、MPに既に送信した到達報告の取り消しを要求する取り消しパケットを送信する。MPは、システム中の全CPからの到達報告がそろうと、確定時刻を先行規制時刻Tbに更新し、全CPに新しい確定時刻を送信する。このようにしてシステム全体のシミュレーション時刻としての確定時刻は time−driven 方式により更新される。シミュレーションを実行するCPでは、シミュレーション時刻は event−driven 方式により更新される。

ここで、あるCPにおいてキャンセルが起こり、MPに取り消しパケットを送信する必要が生じた時を考える。この取り消しパケットがMPに送信される前に、キャンセルを発生させる原因となったパケットを送信したCPがMPに到達報告を送信した場合には、以下のような現象が発生する可能性がある。すなわち、MPが到達報告を受信し、確定時刻を更新した後に取り消しパケットを受信するという現象である。これを防ぐため、各CPはパケットを受信すると送信CPに対してACKを返す。ACKを送るタイミングは、取り消しパケットを送信する必要がある場合は、取り消しパケット送信後、そうでない場合は、パケット受信直後である。各CPは送信したパケットすべてに対するACKを受信している場合のみ、到達報告を送信することができる。

本方式には、ACKを送信するためにプロセッサ間の総通信回数は増加するが、ACKはメッセージ長が非常に短いので、通信システムに対する負荷は大きくないと考えられる。

図9，図10に time+event−driven 型シミュレータのMPおよびCPのソフト

ウェア構成を示す。両図においてTはグローバル方式における規制値であり、time+event−driven 型シミュレータにおけるクロック長を表わす。

ＭＰは初期パラメータを送信した後、ＣＰからパケットを受信するのを待つ。ＣＰから到達報告を受信すると、全ＣＰから到達報告が来ているかどうかを判定し、全ＣＰから到達報告が来ている場合は、確定時刻を更新する。また、取り消しパケットを受信した場合には、送信ＣＰからの到達報告を取り消し、終了パケットを受信した場合には、統計量の編集、出力を行った後停止する。

ＣＰは初期パラメータを受信した後、シミュレーションを実行する。確定時刻を受信すると、確定時刻と先行規制時刻Tbを更新する。また、他のＣＰからシミュレーションパケットを受信すると、送信プロセッサにACKを送信する。また、時刻矛盾が発生した場合には、キャンセル処理を行う。シミュレーション時刻がTbに達した場合、全送信パケットに対しACKを受け取っておれば、ＭＰに到達報告を送信し、統計収集を行う。その他、シミュレーション時刻がTbに達してから確定時刻が更新されるまでの間、ガーベジコレクション等を行い、プロセッサの遊休時間を減少させ処理能力向上を図る。

さらに、time+event−driven 型シミュレータの付加的利点としてはシステム全体のシミュレーション時刻がTbずつ進んでいくため、ＭＰによりシミュレーションの中断、再開、パラメータの変更等の制御を容易に行うことができる。これより、会話的シミュレーションが現在のD-SSQ より容易に実現できると思われる。

また、time+event−driven 型シミュレータは、event−driven 型シミュレータはもちろん、取り消しパケットに関する処理やACKの送信を取りやめることによりtime−driven 型シミュレータに変更可能である。したがって、シミュレーションモデルの性質を考慮して、最も有利なシミュレータを構成できる点で。

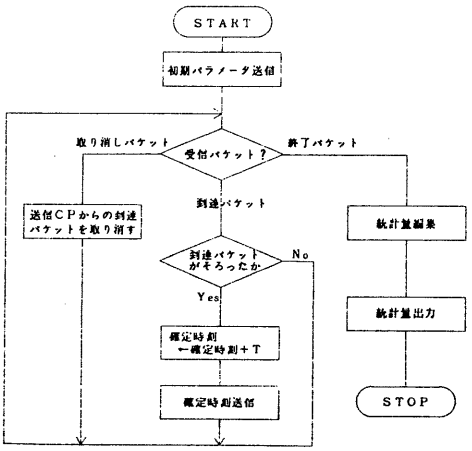

図９　time+event−driven方式のＭＰのフローチャート

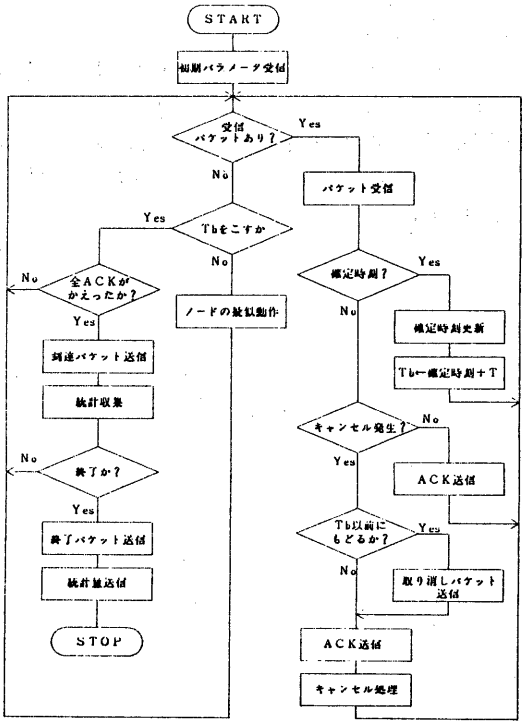

図１０　time+event−driven方式のＮＰのフローチャート

time+event−driven 型シミュレータは柔軟性に富んでいると言える。

6. まとめ

ノード分散型待ち行列網シミュレータ D-SSQ の処理能力改善のため、2つの先行規制方式について述べた。このうちローカル方式は、処理能力改善の目的を結果的には達成できなかった。一方、グローバル方式は最適規制値を与えれば、処理能力を改善できることを示した。

次に time−driven 型シミュレータと D-SSQ の処理能力の比較を行った。

さらに、確定時刻を time−driven 方式によって更新し、各プロセッサのシミュレーション時刻を event−driven 方式で更新する time+event−driven 型シミュレータを提案した。

今後の課題としては、グローバル方式の最適規制値の予測、及びシミュレーション中に規制値Tを変更し、最適規制値に近づけていくダイナミック先行規制方式の検討、time+event−driven 型シミュレータの処理能力の検討が挙げられる。

[参考文献]

(1) 佐藤、中西、真田、手塚:"先行制御方式ノード分散並行処理事象型待ち行列シミュレータ D−SSQについて"、情報理論とその応用研究会資料 1982−10

(2) 佐藤、中西、真田、手塚 : "待ち行列網シミュレータD−SSQについて"、待ち行列理論とその応用研究会報告集1983

(3) 渡辺、佐藤、中西、真田、手塚: "並行処理事象型待ち行列網シミュレータD−SSQについて"、信学技報,CS83−102(1983−8)

(4) Peackok,J et al.:"Distributed simulation Using a Network of Processors",ComputerNetworks 3 (1979)pp.44−56

(5) 佐藤、中西、真田、手塚:"プロセスレベルの並列性を引出すマルチプロセッサシステムの実行制御について"、信学技報,CS83−151,(1984−1)

(6) 稲森、清水、今井、畑田、竹之内、宇留賀:"複合マイクロプロセッサによる並列処理形通信網シミュレータ "、情報処理 EC79−78

(7) 佐竹、上月、西田、宮原、高島: "分散型待ち行列網シミュレータ"、信学技報EC83−40

(8) 中川、長谷川、近江谷、隅田、相磯:"待ち行列システム・シミュレーションにおける並列処理"、情処 計算機アーキテクチャ 34−38(1979)

(9) 渡辺、佐藤、真田、中西、手塚: "シミュレータD−SSQにおける先行制御方式の検討"、信学技報,SE83−166