マイクロコンピュータと
ワークステーション 52－1
（１９８８．９．２７）

Ｐｒｏｌｏｇを利用した待ち行列網解析ツールの構成について

渡辺　尚，三好　浩司，島田　良作
徳島大学工学部

計算機ネットワークの性能評価には，一般に，待ち行列網解析ツールが用いられる．本研究では，高機能マンマシンインターフェイスを持つモデル記述部と高速にシミュレーションを行なう実行部から成る解析ツールを提案する．モデル記述部には，事象の発生条件をホーン節論理によって記述する論理型待ち行列網シミュレーション言語ＳＩＬＱを用いる．シミュレーション実行部は，ＳＩＬＱで記述されたモデルを手続き型言語Ｃに変換して実行する．このツールの構成といくつかのネットワークモデルの評価を行なった例を示す．

CONSTRUCTION OF QUEUEING NETWORK ANALYZER USING A LOGICAL LANGUAGE PROLOG

Takashi WATANABE , Hiroshi MIYOSHI and Ryosaku SHIMADA

Information Science and System Engineering ,
Faculty of Engineering Tokushima University

2-1 Minamijyousanjimacho
Tokushima City Tokushima

Queueing network analyzer has been used to analyze computer networks. This paper provides an analyzer which consists of a model description part with high level man-machine interface and a high speed simulation engine. With the description part a user describes a model in SILQ(SImulation language based on Logic for Queueing networks) by declaring event occurrence condition with horn clauses. The simulation engine converts a model written in SILQ into one in procedual language C to perform high speed simulation. We show the structure of the analyzer and evaluate some examples of network models.

1．まえがき

近年，ＬＡＮやＴＳＳなどに代表される計算機ネットワークの構築が盛んになってきている．計算機ネットワークを構築する場合，ネットワーク性能すなわちデータの入力から出力までの待ち時間やＬＡＮの送信権獲得までの時間などを実システムによって測定することは困難である．そこで，システム内の特定の機能に注目して，待ち行列網モデルを構築し，このモデルの動作を分析することで実システムの性能評価を行うことが考えられる．

待ち行列網を解析するツールは，いくつか既に提案されており，これらは大きくわけて，近似解析を行なうもの，シミュレーションを行なうもの，その両方を行なうものに大別できる[1]-[6]．

理論解析だけを行なうツールは，比較的高速ではあるものの，対象にできるモデルの範囲が狭い欠点をもつ．一方，シミュレーションを含んだツールは，現実に近いモデルを対象にできるため有効である．これには，記述性の高いモデル記述部と高速なシミュレーション実行部が必要となる．

本研究で開発する待ち行列網解析ツールはシミュレーションを行なうツールであり，そのモデル記述にはＰｒｏｌｏｇを利用した待ち行列網専用論理型言語ＳＩＬＱ[4](Simulation Language Based On Logic For Queueing Network)[7]を用いる．ＳＩＬＱは，特に待ち行列網モデルに対して高い記述力を持つように開発された言語であり，事象の発生条件などをホーン節論理で表現することによってモデルを記述する．

また，シミュレーション実行部は，ＳＩＬＱで記述されたプログラムを手続型言語に変換し，シミュレーションを実行することによって高速な評価を可能とする．

本稿では，まず，ＳＩＬＱの概説を行なう．次にツールのシミュレーション実行部について述べる．最後にいくつかの例についての評価を示す．

2．モデル記述部

2．1　シミュレーションモデルの構築

シミュレーションモデルを構築するためには，まず対象とするシステムのモデル化を行なわなければならない．モデル化には基本的に３つの考え方がある．[3][7]

①客中心のモデル化
　モデル化しようとするシステム内を客がどのように動くかを記述

②事象中心のモデル化
　事象発生時，すなわち，システムの状態になんらかの変化をもたらす出来事が発生したときに，システムの状態がいかに変化するかを記述

③プロセス中心のモデル化
　いくつかの事象の連なった処理をプロセスとみなし，プロセスを組み合せてシミュレーションモデルを記述

客中心のモデル化は，客の流れを追って行くと自然にモデルができるので考えやすいという利点がある．代表的なシミュレーション言語として，GPSS[8]，SLAMⅡ[10]などが挙げられる．一方，事象中心のモデル化は，汎用性は高いが具体性に欠けモデル構成が複雑になり，ソフトウェアの記述性が難しくなる傾向がある．これらには，SIMSCRIPT[9,11]，SLAMⅡなどがある．

本ツールのモデル記述部に用いるＳＩＬＱはプロセス中心のモデル化を行なう言語である．

2．2　シミュレーションソフトウェアの記述性

プロセス中心のモデル化を行なった場合，モデルの構造，動作などを定義するために以下の記述を必要とする．

（1）各プロセス間の接続関係
（2）各プロセスの定義を記述するプロセスルーチン
　これは以下の２つの内容からなる．
　（a）プロセスに含まれる事象の論理時刻上での順序関係
　（b）各事象の発生条件
（3）モデルの初期設定

従来のシミュレーション言語でこれらを記述した場合，(2)の記述に多大な労力を費やす．

ＳＩＬＱでは，シミュレーションの対象を待ち行列網システムに限定することにより待ち行列網モデルを構成するプロセスが数種類に分類可能である点に注目し，言語の機能に(2)(a)の内容を包含させることでユーザからこの記述を開放する．

また，(2)(b)については”条件”を記述しやすい言語が適していると考え，Ｐｒｏｌｏｇを基礎とした．

(3)の記述については，定常状態における待ち行列網システムの振舞を扱うため考慮していない．

2．3　ＳＩＬＱの概要

ＳＩＬＱは，待ち行列モデルが論理的な表現によって記述できる部分が多いことから，論理型言語 Prolog を記述形式としている．

Ｐｒｏｌｏｇは一階述語論理の中のホーン節をプログラムとみなし，宣言的記述が可能な言語である．しかし，Ｐｒｏｌｏｇで記述できる世界は同一時刻に発生するイ

ベントのみからなり時刻概念を扱うことはできない．そこで，Ｐｒｏｌｏｇに時刻概念を導入した言語として，Ｔ－Ｐｒｏｌｏｇ[14]などがある．しかし，その時刻概念導入の方法は，ホーン節の内部に処理の実行順序を示す表記を取入れており，結局プロセス間の時刻関係をユーザが記述するため，Ｐｒｏｌｏｇの宣言的記述が可能な利点を十分に生かしきれていない．

ＳＩＬＱは，Ｐｒｏｌｏｇの記述性を損うことなく，プロセス内部での時刻消費を規則化することによって待ち行列網システムをモデル化することで時刻概念を陰に導入している．以下にＳＩＬＱの特徴を列挙する．

（1）同じような事象から成るプロセスをまとめたタイムオブジェクトによって視覚的かつ具体的なモデルの構築が可能．

（2）一種の組み込み述語であるプリミティブによって待ち行列網モデルを論理的に定義可能．

（3）個々の論理式は，その実行状態を考慮することなくプログラミング可能．

２．４　タイムオブジェクト

タイムオブジェクト(Arrival,Queue,Server,Branch,Joint,Departure)とはオブジェクト指向プログラミングにおけるオブジェクトの概念にオブジェクト内部での論理時刻の消費を導入したものであり，待ち行列網モデルの構造は，これら６つのタイムオブジェクトを組み合わせることによって記述される．以下にその性質を示す．

（1）各タイムオブジェクトはそれぞれに内部変数を持つ．

（2）外部からメッセージを送ることによってのみタイムオブジェクトを起動しオブジェクトの内部変数の操作が可能である．

（3）タイムオブジェクトは一定の規則のもとに，下記の３状態を循環的に遷移する．

（a）待機状態

メッセージを待っている，あるいは，ある条件が満たされるのを待っている状態．

（b）実行状態

受け取ったメッセージに対する処理を実行している状態．必要な場合には他のオブジェクトに対してメッセージを送信する．

（c）停止状態

ある時刻がくるまで処理を停止している状態．

２．５　プログラミングプリミティブ

ＳＩＬＱでは，約３０個のプログラミングプリミティブを提供しており，ユーザはこれらを使って，モデルの構造，及び，動作といったモデル定義，及びシミュレーション制御，出力統計の指定をＰｒｏｌｏｇ形式のホーン節の形で記述できる．さらに，プリミティブはシステムプリミティブとユーザプリミティブとに分類できる．

＊システムプリミティブ

システム内部で定義されているプリミティブであり，ユーザはホーン節の本体部のみで用いることができる．例えば，現在の待ち行列長を示すsys_customers_in_queueや，システムの時刻を示すsys_timeなどがある．

＊ユーザプリミティブ

ユーザが定義を行うプリミティブである．ユーザプリミティブはホーン節の頭部のみで用いることができ，ユーザは引き数もしくは本体部を記述することによってその定義を行う．またこれらには，ユーザがプログラムの中で必ず定義を行わなければならないものと，定義を省略すればデフォルト値によって補われるものとがある．ユーザプリミティブの例としては，客が待ち行列からでることができる条件を示すqueue_out_conditionや，プロセス間の接続関係を示すconnectなどがある．

３．シミュレーション実行部

３．１　実行部の概要

実行部は，パーサモジュール，初期化モジュール，シミュレーション制御モジュール，タイムオブジェクト実行モジュール，シミュレーション結果出力モジュールの５つのモジュールから構成されている．[15]

実行部におけるシミュレーションを行う過程は次のようである．まず，パーサモジュールにより論理型言語で記述されたユーザプログラムを手続き型言語Ｃのプログラムに変換する．そして，このＣのプログラムを本システムで用意したシステムプログラム中のタイムオブジェクト実行モジュール内に埋め，初期化モジュール，シミュレーション制御モジュール，タイムオブジェクト実行モジュール，シミュレーション結果出力モジュールと共にコンパイルする．これによりシミュレーションの実行が可能となる．（図１）

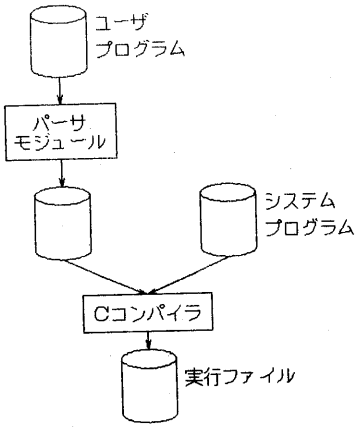

図1　実行ファイルの作成

シミュレーションの実行は，データやプロセスの初期化を行った後，まず，シミュレーション制御モジュールが起動することによって開始される．このモジュールは，これから起きるイベントの中で，現在のシミュレーション時刻に最も近いイベントを選出し，そのイベントの時刻にシミュレーション時刻を更新する．そして，そのイベントを処理するために必要なタイムオブジェクト実行モジュールを呼び出す．

呼び出されたタイムオブジェクト実行モジュールは，プロセスに対応する処理を対象となる客に施し，プロセスの状態等を更新する．この様にシミュレーション実行モジュールがタイムオブジェクト実行モジュールをコールすることによってシミュレーションを進行させる．

シミュレーション実行中には，随時統計データが収集される．そして，シミュレーション終了時刻に達すると，シミュレーション結果モジュールが起動し，統計データとモデルの状態を出力する．以上によってシミュレーションが実行される．（図2）

なお，実行部はＩＢＭ　ＲＴ－ＰＣ上に作成されている．

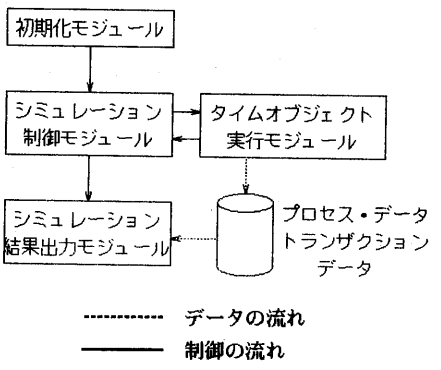

図2　実行部のフローチャート

３．２　モジュールの機能

［１］　パーサモジュール

パーサモジュールは，ＳＩＬＱで記述されたプログラムをＣ言語に変換する．

ＳＩＬＱで記述されたプログラム中のｆａｃｔは主にパラメータに，またｒｕｌｅはｉｆ文に変換される．プロセス間の接続は以下に述べるイベント番号とそのイベントが起きるプロセスを示すデータに変換される．

例えば，プロセスの接続関係などは，客に対する次の８つのイベントを考え，それぞれを０－７の番号で表す．

０・・Ａｒｒｉｖａｌからモデルへ到着する
１・・Ｑｕｅｕｅに入る
２・・Ｑｕｅｕｅから出る
３・・Ｓｅｒｖｅｒに入る
４・・Ｓｅｒｖｅｒから出る
５・・Ｂｒａｎｃｈに行く
６・・Ｊｏｉｎｔに行く
７・・Ｄｅｐａｒｔｕｒｅからモデル外へ退去する

そして，１人の客が引き起こすイベントの系列を上記番号で示し，配列(tran_way)に格納する．また，どのプロセスで起こるイベントかを表すために配列(no_way)にプロセス識別子を格納する．

例　客が０番の入口から到着する→待ち行列３に並ぶ→サーバ５でのサービスを受ける→０番の出口から退去するといった道順を通るときそれぞれの配列は以下のようになる．

```
tran_way[][]={{0,1,2,3,4,7}};
no_way[][]  ={{0,3,3,5,5,0}};
```

［２］　初期化モジュール

初期化モジュールは，プロセス・データ及びトランザクション・データの初期化を行う．また，最初の客の到着時刻を決定する．

［３］　シミュレーション制御モジュール

シミュレーション制御モジュールは，シミュレーション時刻に最も近いイベントが起こる客を取り出し，その時刻を求めシミュレーション時刻の更新を行う．そして，その客に対応するタイムオブジェクト実行モジュールを呼び出す．

［４］　タイムオブジェクト実行モジュール

タイムオブジェクト実行モジュールは６つのタイムオブジェクトに対するサブモジュールから構成されている．サブモジュールは，各タイムオブジェクトの機能に相当する一連の処理を実行する．

［５］　統計出力モジュール

統計出力モジュールはシミュレーション実行中に，採取したデータを算出し得られたデータを画面及びファイルに出力する．

4．評価例

4．1　タクシースタンドモデル

タクシー乗り場のような二つのトランザクションの同期を取るようなモデルを考える．タクシーと客の待ち行列をそれぞれ別のプロセスと考え，二つの待ち行列から出てきた客とタクシーをＪＯＩＮＴによって結合すると考えると図3のようにモデル化できる．

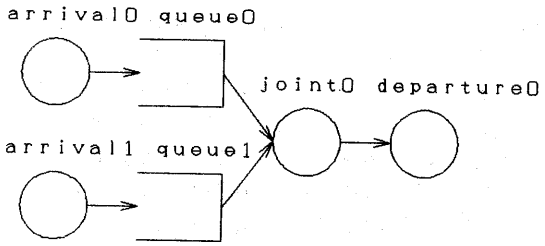

図3　タクシースタンドモデル

図4に示すモデルを，客の平均到着間隔の分布がそれぞれ平均２０秒，１５秒，の指数分布であるという仮定のもとにシミュレーションを行なう場合のＳＩＬＱによるプログラム記述を図4に示す．

また図5に，シミュレーション実行部のパーサによってＳＩＬＱで記述されたプログラムを変換した例を示す．

```
start_time(0).
end_time(1000).
buffer_capacity(Id,10000).
queue_out_condition(Q_Id):-
        sys_customers_in_queue(0,[Cust1|R1]),
        sys_customers_in_queue(1,[Cust2|R2]).
connect(1,[arrival  ,0],[queue    ,0]).
connect(2,[arrival  ,1],[queue    ,1]).
connect(3,[queue    ,0],[and_joint,0]).
connect(4,[queue    ,1],[and_joint,0]).
connect(5,[and_joint,0],[departure,0]).
```

図4　ＳＩＬＱによるタクシースタンドプログラム

```
#define LANE_NUM 3
#define QUEUE_NUM 2
#define JOINT_NUM 1
#define SERVER_NUM 0
#define BRANCH_NUM 0
tran_way[][50]={{0,1,2,6},
                {0,1,2,6},
                {6,7}};
no_way[][50]  ={{0,0,0,0},
                {1,1,1,0},
                {0,0}};
start_time=0;
end_time=1000;
queue[0].buffer_capacity=10000;
queue[1].buffer_capacity=10000;
queue 0 :if(queue[1].len > 0)
queue 1 :if(queue[0].len > 0)
```

図5　変換後のタクシースタンドプログラム

4．2　ウィンドウ・フロー制御モデル

本項では，複雑なネットワークモデルの実行例として計算機網におけるトラヒックの一時的な増大によるデッドロックを回避するために用いられるウィンドウ・フロー制御を取り上げる．

ウィンドウ・フロー制御は，パケット交換網において一般に用いられている制御方式であり，各発着信局間に設定される論理的な１本の伝送路である論理チャネル（ＬＣ）ごとに，網内に入力できる着信未確認パケット数を一定数（ウィンドウ・サイズ）以下に規制し，網内の混乱を緩和する制御方式である．ここでは，図7に示す様な3局直列網を取り上げる．

各局の処理時間に比べて無視できると仮定すれば，図6のモデルは回線を処理施設と見なして，２個のサーバが直列に接続された待ち行列モデルであると考えられる．このモデルを記述したＳＩＬＱによるプログラムを図7に示す．

またＣ言語に変換されたプログラムを図8に示す．

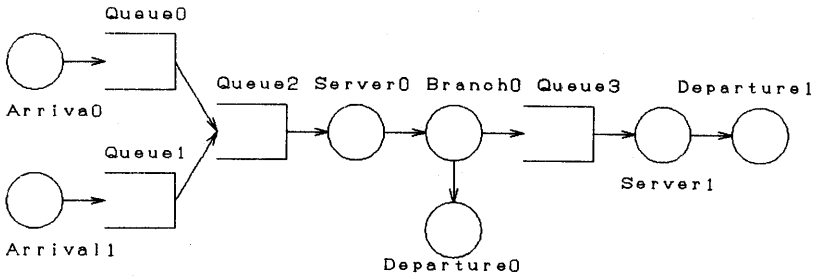

図6　ウィンドウ・フロー制御モデル

```
queue_oue_condition(K):-member_of(K,[1,2]),
      sys_customers_record([queue,1],X),
      sys_customers_record([departure,K],Y),
      X=Y+C.
      window_size(K,M).
      C<M.
queue_out_condition(3):-
      sys_customers_in_server(1,[]).
queue_out_condition(4):-
      sys_customers_in_server(2,[]).
branch_selection(1,7):-
      sys_customers([branch,1],[X,[1]]).
branch_selection(1,8):-
      sys_customers([branch,1],[X,[2]]).
member_of(X,[X|Y]).
member_of(X,[Y|Z]):-member_of(X,Z).
connect(1,[arrival,1],[queue,1]).
connect(2,[queue  ,1],[queue,3]).
connect(3,[arrival,2],[queue,2]).
connect(4,[queue  ,2],[queue,3]).
connect(5,[queue  ,3],[server,1]).
connect(6,[server ,1],[branch,1]).
connect(7,[branch ,1],[departure,1]).
connect(8,[barnch ,1],[queue,4]).
connect(9,[queue  ,4],[server,2]).
connect(10,[server,2],[departure,2]).
```

図7　ＳＩＬＱによるウィンドウ・フロー制御モデル

```
#define LANE_NUM 2
#define QUEUE_NUM 4
#define JOINT_NUM 0
#define SERVER_NUM 2
#define BRANCH_NUM 0

Queue0:
if(arr_num[0].-dep_num[0]<window_sizw[0]);
Queue1:
if(arr_num[1].-dep_num[1]<window_sizw[1]);
Queue2:
if(server[0].busy_num<server[0].window_num);
Queue3:
if(server[1].busy_num<server[1].window_num);

tran_way[][50]={{0,1,2,1,2,3,4,8}
                {0,1,2,1,2,3,4,1,2,3,4,8}};
no_way[][50]  ={{0,0,0,2,2,0,0,0}
                {0,1,1,2,2,0,0,3,3,2,2,0}};
```

図8　変換後のウィンドウ・フロー制御モデル

５．実行部の評価

５．１　精度の評価

Ｍ／Ｍ／１（窓口１のサービス施設を１つ持ち，客の到着，及びサービス時間が共に指数分布に従う）モデルに対しては，理論解析により，厳密解を得ることができる．そこでＭ／Ｍ／１モデルを用いて利用率をいくつか変えてシミュレーションを行ない理論値との比較を行なった．表１にその結果を示す．対象とした統計量としては，平均システム時間，平均待ち行列長，並びに平均待ち時間である．なお，サンプル数は約１０００００である．図９，図１０に利用率の違いによる平均待ち行列長，平均待ち時間のグラフを示す．その結果理論値との誤差は最大約２％であり，十分高い精度によってシミュレーションを実行できることが分った．

表１　Ｍ／Ｍ／１モデルのシミュレーション結果

平均到着時間	5	6.6	10	20	40	80
平均サービス時間	4	4	4	4	4	4
平均待ち時間	15.98	6.30	2.73	1.00	0.44	0.20
平均待ち行列長	3.22	0.96	0.28	0.05	0.01	0.003
平均システム時間	19.98	10.29	6.77	5.00	4.44	4.20
サーバ利用率	0.806	0.609	0.405	0.201	0.101	0.050

５．２　処理速度の評価

ＲＴ－ＰＣ上で，Ｍ／Ｍ／１モデルを実行したところ約１５０秒で１０００００のサンプルを得ることができた．また，従来のＳＩＬＱ処理系[7]は，Ｓｕｎ３上に作成されたシステムであり，タイミングコントロール等のシミュレーション制御や統計処理については，手続き型言語のルーチンを用いて処理している．論理型言語で記述された待ち行列網モデルに依存する部分は，ホーン節処理モジュールによって通常のＰｒｏｌｏｇインタプリタと同様に処理される．

本研究で提案した解析ツールの実行部は，ホーン節で記述されたモデルを手続き型言語に変換することによって高速化を図っている．

実行部の速度に関する評価を行なうために，従来のシステムが作成されている同じパソコン上に実行部を移植し，同じモデルのシミュレーションを行なって比較した．表２にM／M／１待ち行列モデル，表３にウィンドウ・フロー制御モデルに対するの処理速度の比較を行った結果を示す．なお，M／M／１モデルについては，客の到着間隔を平均１０秒の指数分布，サービス時間を５秒の指数分布とし，シミュレーション時刻０～５００００秒までのシミュレーションを行った．また，ウィンドウ・フロー制御モデルに対しては，Arrival0,Arrival1の到着間隔を平均３０秒，２０秒の指数分布，全てのサーバのサービス時間分布を平均１０秒とし，シミュレーション時刻０～１５０００秒までのシミュレーションを行った．

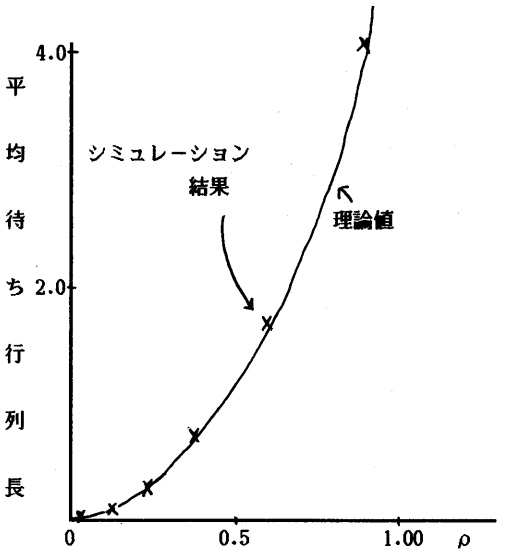

図９　理論値との比較（平均待ち行列長）

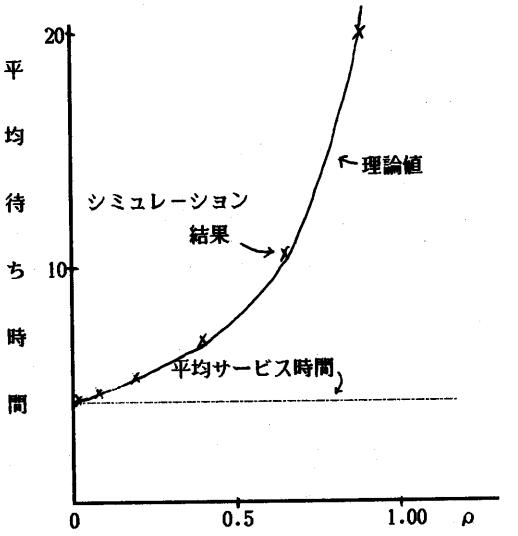

図１０　理論値との比較（平均待ち時間）

表２　処理速度の比較（M／M／１）

	退去客数	所用時間
本システム	５０７０	２２　秒
従来のシステム	４８２９	１０２５　秒

表３　処理速度の比較（ウィンドウ・フロー制御）

	退去客数	所用時間
本システム	(LC1)　４９５	１０　秒
	(LC2)　６９３	
従来のシステム	(LC1)　４７９	５６１　秒
	(LC2)　７３７	

６．まとめ

計算機ネットワークを評価するために，モデル記述部

とシミュレーション実行部から成る待ち行列網解析ツールを構成した．まず，モデル記述部ではＳＩＬＱによって事象の発生条件などを論理式によって記述する．また，シミュレーション実行部は，ＳＩＬＱによって記述されたプログラムを手続き型に変換し，高速にシミュレーションを実行する．いくつかのモデルの評価を行なって，ツールの解析精度と処理速度について検討した．その結果，十分高い精度で満足しうる速度によって評価可能であることが分った．

今後は，視覚的インターフェイスを持ったモデル記述部を構成し，ツールの強化を図る予定である．

参考文献

[1]小島,米田,田中,春木:"蟻塚とその利用環境,"情処研報 Vol.86,No12(1986.2)

[2]木村:"QNA:Queueing Network Analyzerについて,"ｵﾍﾟﾚｰｼｮﾝｽﾞﾘｻｰﾁ学会誌(1984.4)

[3]D.Potier and M.Veran:"The Markovian Solver of QNAP2 and Examples,"International Seminar on Computer Networking and Performance Evalution,(1985.9)

[4]C.Sauer,E.MacNair and S.Salza: "A Language for Extended Queueing Network models,"IBM J. Res. Develop Vol.24,No6(1980.11)

[5]紀，守田，小林:"BCMP型待ち行列網による性能評価ﾂｰﾙQM-X,"情処第２４回全国大会,p.271,(1982)

[6]K.G.Ramakrishnan,D.Mitra:"An Overview of PANACEA,a Software Package for Analyzing Markovian Queueing Networks,"The Bill system Technical Journal,Vol.61,No10(1982.12)

[7]巽，渡辺，井上，中西，手塚：”離散型シミュレションン言語ＳＩＬＱの処理系の構成”，情報処理学会プログラミング言語研究会(1987)

[8]中西　俊男：”シミュレーション言語”，情報処理，Vol.22,No.6(1981)

[9]山本，浦：”離散型シミュレーション言語の現状と将来の展望(1)～(2)”，情報処理学会誌vol.22,No.9,No.11(1981)

[10]森戸，相沢：”ＳＬＡＭⅡによるシステム・シミュレーション入門”，構造計画研究所(1986)

[11]森戸　晋：”離散系シミュレーションの最近の動向”，シミュレーション　第７巻第２号(1988)

[12]荒木，渡辺，井上，手塚：”論理型言語による待ち行列網モデルの仕様記述とその作成支援環境”，情報処理学会第３５回全国大会，４ｗ－９(1987)

[13]荒木，渡辺，中西，真田，手塚：”論理型言語による待ち行列網シミュレーションについて”，電子通信学会 IN86-79 (1986)

[14]渡辺，中西，真田，手塚：”タイムオブジェクトを用いた待ち行列網シミュレーション専用論理型言語”，電子情報通信学会論文誌 （Ｄ） Vol.J70-D No.5(1987)

[15]三好，渡辺，島田：”待ち行列網シミュレーション言語ＳＩＬＱの高速処理系について”，信学技報 IN88-65 (1988)

[16]Ivan F., Janos S."A Discrete Simulationsystem based on Artificial Intelligence methods"

[17]中西　俊男：”コンピュータシミュレーション”，近代科学社(1977)