Simulated Annealing for Throughput Optimization in Communication Networks with Window Flow Control

Ian F. Akyildiz and Ron Shonkwiler***

* School of Information and Computer Science
** School of Mathematics
Georgia Institute of Technology
Atlanta, Georgia 30332
U. S. A.

ABSTRACT

The queueing model of the multichain network is presented where the network is window flow controlled on its virtual channel of communication network. Its performance is optimized based on Simulated Annealing. The solution space is assumed to be the set of all admissible service rates of all stations with the nonlinear cost constraint. The optimum total throughput of the network is determined for different chain packets.

Key Words: Computer Networks, Network Design, Window Flow Control, Queueing Network Modeling, Simulated Annealing, Optimization.

1. Introduction

Most packet networks nowadays provide virtual channels that are end-to-end flow controlled. Flow controlled virtual channels may be maintained between data sources and sinks or between pairs of source destination nodes. In some networks, both types of flow controlled virtual channels are maintained. In the queueing network models to be considered, only one type of flow controlled virtual channel is assumed. Such channels may be interpreted as being maintained either between packet sources and sinks or pairs of source-destination switching nodes. An important function of end-to-end flow control protocols is the synchronization of the data source input rate to the data sink acceptance rate. All of them work by limiting the number of packets that a virtual channel can have in transit within the network. Hence they also provide, to some extent, a congestion control capability for the network as a whole. We consider the modeling of flow controlled virtual channels as multiple closed chains. The chain population size corresponds to the maximum number of packets that can be in transit within the virtual channel. This number will be referred to as the *virtual channel window size.* The effect of virtual channel window sizes on the throughput-delay characteristics of the network can be studied using closed queueing networks with multiple chains.

Figure 1 illustrates a queueing network model of packet switching network with R virtual channels. Each virtual channel has a source and a sink both of which are also modeled as FCFS servers with exponentially distributed service times. Packets in the same virtual channel follow a fixed route which may be chosen probabilistically from a finite set of routes between source and sink.

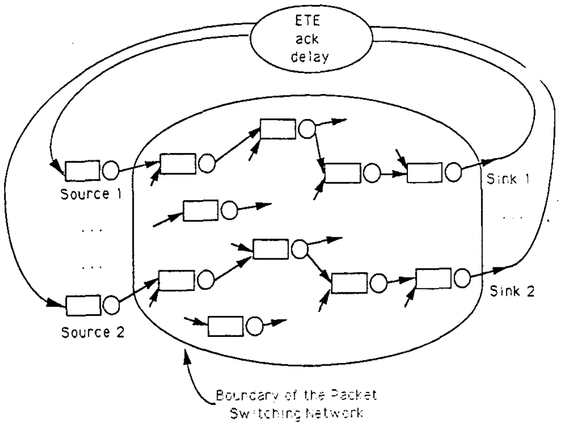

Figure 1. Modeling Flow Controlled Virtual Channels

The delay for the return of an end-to-end (ETE) acknowledgement (ACK) from the sink to the source indicating receipt of a packet is modeled by an independent random variable, the distribution of which may be different for different virtual channels. This delay is modeled by an IS (infinite server) node that joins the sink to the source to yield a closed chain in the queueing network model. It is not really important to model the route of an ACK explicitly because ETE ACKs are typically either sent piggybacked in data packets, or, if sent separately, very short. Thus, they consume relatively small amounts of buffer and channel resources in the network, which may be accounted for separately. In [15], Reiser suggested that the ACK traffic may be accounted for by reducing the channel capacities by amounts equal to the throughputs of ACK packets.

The flow control window size of a virtual channel is the maximum number of packets that it can have in transit within the communication network at the same time. Let K_r denote the window size of virtual channel r for $r = 1,...,R$. If the number of packets in transit within a virtual channel is equal to its window size, then the source server is "blocked". A blocked source server is later unblocked when an ETE ACK returns from the sink indicating the receipt of a packet. The blocking behavior is naturally modeled in a queueing network by a closed chain with a fixed number of circulating packets. Each packet corresponds to an "access token". Initially, K_r tokens are placed at the source server of virtual channel r. Each packet admitted into the network carries a token with it. When there is no more token at the source server, it is blocked. A packet arriving at the sink node of the virtual channel releases its token which is then carried back by the ETE ACK to the source server to be reused again. Thus, the K_r circulating tokens of a virtual channel corrspond to the K_r circulating packets of a closed chain. When the source server is not blocked, it generates a new packet for input into the network at the rate γ_r. The physical interpretation of the rate γ_r depends upon the loading on the virtual channels. For a lightly loaded network γ_r may be interpreted as the external arrival rate of packets to the user of virtual channel r. For a heavily loaded network such that the queue of data packets at the source is nonempty most of the time, γ_r may be interpreted as the reaction speed of the user to a signal (or message) from the network interface of the virtual channel authorizing new input. Several researchers investigated the flow control in communication networks in recent years. Reiser [15] introduced a tandem closed queueing network model for the evaluation of computer communication networks. He assumed losses in the model, i.e, arriving messages facing a full window are discarded. Varghese, Chou and Nilsson [19] assumed that such messages are not discarded; instead they wait for their turn to enter the network in a global queue. Georganas [6] studied a message-swiched network with three level flow-control, namely End-to-End, Local and Global window flow controls. He assumed the service times of each server to be exponential. Akyildiz [1] analyzed similar networks with local and global window controls and servers with nonexponential service times. In some other analysis of window flow control protocols the ACK delay is explicitly or implicitly assumed to be a random variable and follow a particular distribution [12,17] or congestion freedom [13,14].

2. Queueing Network Modeling

In the queueing network model considered, FCFS servers are used to model communication channels and IS servers are used to model random delays associated with acknowledgements and timeouts. We assume that there are N servers in the network. Service times at each server i for packet of chain r have mean values $1/\alpha_{ir}\mu_i$ where μ_i is the nominal service rate and α_{ir} characterize the relative service rates for different packet classes in the same server for $i = 1,...,N$ and $r = 1,...,R$. Packets belong to different (routing) chains, indexed by $r = 1,2,...,R$. Chains are characterized by different routing behaviors. The routing behavior of packets in chain r is specified by a first order Markov chain with transition probabilities $p_{ij;r}$, i.e., a packet of chain r after receiving service at server i proceeds to server j. Multiple chains are needed between the same source-destination node pair to correspond to different virtual channels connecting data sources in the same source node and data sinks in the same destination node.

Bard/Schweitzer [3,16] algorithm provides the following solution for the analysis of networks with the above assumptions.

The *mean delay of a chain r* packet at server i (for $i = 1,2,...,N$ and $r = 1,...,R$) is:

$$\bar{t}_{ir} = \begin{cases} \dfrac{1}{\alpha_{ir}\ \mu_i}[1 + \dfrac{K_r - 1}{K_r}\bar{k}_{ir}] + \sum\limits_{s \neq r}^{R} \bar{k}_{is} & \text{for } \neq IS \\ \dfrac{1}{\alpha_{ir}\ \mu_i} \quad \text{for } IS \end{cases} \tag{1}$$

The *throughput* of the network for chain r packets is obtained by Little's law:

$$\lambda_r = \frac{K_r}{\sum\limits_{i=1}^{N} e_{ir}\ \bar{t}_{ir}} \tag{2}$$

The *mean number of chain r packets* at the i-th server (for $i = 1,2,..,N$ and $r = 1,2,...,R$) is also obtained by Little's law:

$$\bar{k}_{ir} = \lambda_r\ e_{ir}\ \bar{t}_{ir} \tag{3}$$

where e_{ir} is the mean number of visits that a packet of chain r makes to server i and is computed from

$$e_{ir} = \sum_{j=1}^{N}\sum_{s=1}^{R} e_{js} \cdot p_{ji;s} \qquad \text{for } i = 2,...,N \ ; \quad r = 1,..,R.$$

We start with $\bar{k}_{ir} = \dfrac{K_r}{N}$ for $i = 1,...,N$ and $r = 1,...,R$ and iterate sequentially through equations (1) - (3) until convergence is observed. The convergence is observed if the $\bar{k}_{ir}$ values, equation (3), of the current iteration deviates less than a threshold value (e.g., $\varepsilon = 10^{-4}$) from $\bar{k}_{ir}$ values of the previous iteration.

Several optimal routing problems may be formulated depending upon the nature of the incremental flow and optimization objective. For example, the objective of ARPANET routing algorithm [11] is to minimize the (estimated) delay of an individual packet from its source node and to its destination node. However, it has been observed that routing algorithms with the objective of individual-optimization do not necessarily lead to network optimization, i.e., minimizing the mean delay of all packets in the network or accordingly maximizing the throughput of the network.

Performance optimization is an important step in the design and planning of communication networks. Several authors have discussed the issue of optimization in recent years [2,5,8,10,18]. Most of the optimization studies regarding the communication networks are based on the selection of the routing of packets in the network. However, in virtual circuit networks the routes are fixed. Thus, in this work the optimization question is to determine the optimal service rates of each server as decision variables subject to cost constraints such that the throughput of the network will be maximum.

3. Simulated Annealing

In this section we describe our application of simulated annealing to the task of locating the service rates for optimal or near optimal throughput under fixed cost. Initially, more traditional optimization methods for problems subject to constraints were tried but these had to be rejected. In an approach by Lagrange multipliers, the Lagrangian proved too difficult to solve in general due to its non-linearities. Next a gradient ascent numerical method was tried. However, due to the presence of multiple local maxima, this method was unable to locate the global maximum. When there are numerous local maxima finding the global maximum by an ascent method is a matter of pure chance depending on the algorithm's starting point. Unfortunately, a starting point chosen without good knowledge of where the global optimum is located will in all probability be near a local maximum and "get stuck" at it.

Thus an aproach incorporating random elements is indicated. Modern Monte Carlo optimization methods such as simulated annealing exploit random search but in a controlled way based on natural phenomena. Moreover, there is now a substantial body of experience and theory underlying simulated annealing. The method has been applied to numerous problems from practical ones such as cell placement and interconnection for VLSI circuit design [9] to well-known fundamental ones of combinatorial optimization such as the traveling salesman problem [4]. Excellent results have been reported in many cases. Normally, a disadvantage of Monte Carlo methods is their long run times. However, on the suite of example problems we used here, run times are in seconds on workstations. Since our example problems were not specially selected, we feel the methods will work well in general on this class of optimization problem.

The physical analogy on which simulated annealing is based is that of thermodynamic systems where internal molecular energy is to be minimized, for example, the formation of crystals from a liquid melt. If the system is cooled sufficiently slowly, low internal states of energy can be reached, the formation of a perfect crystal in our example. If cooling is too fast, then only metastable states of much greater internal energy will be reached. The function of temperature is to allow the system to pass from one state to another. When the temperature is zero, no transitions are possible and the system is frozen in some particular configuration with its corresponding energy. However, at non-zero temperatures and especially at high ones, transitions from states of low energy to states of high energy can occur. In this way it is possible for the system to climb out of a local energy well and ultimately freeze in the globally minimal energy state.

The frequency with which possible thermodynamic states are actually observed is governed by the Boltzmann energy distribution law, thus the probability of observing a state having energy E at temperature T is

$$\frac{1}{Z}\ e^{-E/BT}$$

where B is the Boltzmann constant and Z, the partition function, is the sum

$$Z = \sum e^{-E/BT}$$

over all states of the system.

Consequently, it is less likely to observe a high energy state when the temperature is low than when it is high. Put differently, a transition from a state of low energy to a state of high energy is less likely to occur at low temperatures than at high temperatures. In fact, the probability of such a transition is given by

$$e^{-\Delta E/BT}$$

where ΔE is the energy difference between the two states.

Consider now an abstract implementation of this physical process. An ensemble of particles with their attendant positions and velocities is replaced by a data structure containing values sufficient to describe a possible solution to the abstract problem, in short a *state* $\underline{\mu}$. The energy of the physical system becomes in the abstract system the variable λ to be optimized. Values of this all important variable are defined for each state and are calculated by an *objective function* w, $\lambda = w(\underline{\mu})$. Thermal kinetic energy as a function of temperature in the physical system brings about transitions between states. In the abstract implementation a suitable *transition process* between states must be concocted. Finally, just as a physical system cooled sufficiently slowly freezes in states of low energy, work on simulated annealing theory conducted by several researchers, [7,9,20] and others, has yielded the required *temperature management* to guarantee convergence of the abstract process to a globally optimum objective value. We state the result due to Hajek [7]:

Theorem 1. If an annealing process is irreducible and reversible, then as running time t tends to infinity $t \to \infty$, the probability that the process is in a globally optimum state at time t tends to certainty provided temperature T is lowered no faster than

$$T = \frac{d}{\log(t+1)}.$$

In this the parameter d is a constant at least as large as the depth of all non-global minima (or the height of all non-global maxima). The process is *irreducible* if every state eventually leads by the transition process to every other state. The process is *reversible* (minimization formulation) if whenever a state μ_2 can be reached from a state μ_1 along a sequence of intermediate states all of whose objective values are less or equal to λ, then there should be a sequence of states leading back from μ_2 to μ_1 all of whose intermediate objective values are less or equal to λ as well.

The salient difference then between a traditional deterministic minimizer and simulated annealing is that the latter permits uphill transitions in a controlled manner as well as downhill ones governed probabilistically by the auxiliary parameter of temperature and

Boltzmann's law. This difference enables the Monte Carlo method to succeed in finding the global optimum provided temperature is controlled properly during the course of the run.

4. Throughput Optimization

As explained above a numerical implementation of simulated annealing consists of a data structure for the state or solution space of the problem, a probability distribution on the transition between states, a temperature variable and an optimization function defined on states.

We now detail our assignments of these ingredients to the problem of maximizing multi-class network throughput. Since our application is one of maximization, the sense of optimization of the previous section is reversed. Thus, it is with probability

$$e^{+\Delta \lambda/T}$$

that transitions will be made from high *throughput* states to low ones. We have taken the Boltzmann constant $B=1$ here as it merely serves to scale objective values. According to Theorem 1 these may be scaled once and for all by the choice of the parameter d.

State Space

Our solution space X will be the set of all N-tuples of admissible service rates

$$\underline{\mu} = (\mu_1, \mu_2, \ldots, \mu_N) \qquad \text{for} \qquad \mu_i \geq 0$$

with the constraint

$$\sum_{i=1}^{N} c_i \ \mu_i^{\beta} = COST$$

where $COST$ and the c_i's are given constants. When $\beta = 1$, X is the portion of a hyperplane lying in the positive cone of N-dimensional Euclidean space. The hyperplane being determined by $c_1, c_2, \ldots, c_N$ and $COST$. When $\beta \neq 1$ the hyperplane is replaced by a curved surface instead.

Objective Function

Associated with every service rate vector $\underline{\mu}$ in X is its corresponding network throughput $\lambda = \sum_{r=1}^{R} \lambda_r$ where λ_r is the throughput for packet chain r. These latter quantities for $r = 1,2,\ldots,R$ are determined from equations (2) which must be solved iteratively using $\bar{t}_{ir}$ and $\bar{k}_{ir}$ from equations (1) and (3).

Transition Probability Distribution

The transition from state to state is affected by means of two processes. The first, the generation process, generates new trial states as a function of the known present state. Annealing theory has little to say about the generation process outside of the general requirement of irreducibility and reversibility. Thus it is here that experience and problem specific information is brought to bear.

On the basis of experience we use a generation process in which the trial state does not differ much from the present state. The process is made more complicated by having to satisfy the constraint. Fortunately a hyperplane constraint is easy to incorporate. In the first step one randomly selected component of the given rate vector is chosen to be modified and that component is then modified by a random amount up to a maximum, DMUMAX, which is taken as a parameter of the implementation. Following this step the modified rate vector will no longer satisfy the constraint. However, an adjustment factor is easily calculated and multiplying it into all components of the modified rate vector results in a new rate vector satisfying the constraint and still only slightly perturbed from the original. Expressed algorithmically this is

```
generate $(\mu_1, \mu_2, \ldots, \mu_N)$;
    $k \leftarrow (int)(1 + N*rand())$;
    $s \leftarrow (c_k/COST)\mu_k^{\beta}$;      /* this scales terms in 0 to 1*/
    $l \leftarrow \max(0, s - DMUMAX)$;
    $r \leftarrow \min(1, s + DMUMAX)$;
    $n \leftarrow l + (r-l) * rand()$;
    $f \leftarrow 1/(1 + n - s)^{1/\beta}$;
    for $i = 1,2,\ldots,N$
        if $(i \neq k)$   $\nu_i \leftarrow f * \mu_i$
        else
            $\nu_i \leftarrow f * (COST*N/c_k)^{1/\beta}$ ;
        fi
    endfor
    return $\nu_1,\ldots,\nu_N$
```

In the linear case, $\beta = 1$, the generation process may be visualized geometrically as follows. As previously noted, valid service rate vectors $\underline{\mu} = (\mu_1,...,\mu_N)$ are those on a hyperplane in the positive cone of N-dimensional Euclidean space. Perturbating a single component of $\underline{\mu}$ by amount $S = N-s$ results in moving off this hyperplane in the coordinate direction of the chosen component. Finally, the readjustment by factor f corresponds to shrinking or expanding the perturbed rate vector until the constraint is again satisfied.

When the exponent differs from one, $\beta \neq 1$, the only change in the aforegoing analysis is that the surface is not a hyperplane but rather is curved slightly. Otherwise geometrically the procedure is the same.

Having generated a trial rate vector $\underline{v} = (v_1,...,v_N)$ as a perturbation of the present rate vector $\underline{\mu}$, it is accepted or rejected as the next state of the process probabilistically in accordance with the Boltzmann distribution law. The throughput λ_v for the trial state is calculated and compared with the throughput λ_μ of the present state. If λ_v exceeds λ_μ then v does replace μ. Even if $\lambda_v < \lambda_\mu$ it still replaces μ with probability

$$p = e^{\frac{(\lambda_v - \lambda_\mu)}{T}}$$

This is conveniently affected by use of a random number generator, *rand*(), returning values in [0,1). Then v does replace μ if *rand*() $< p$. In the event that v does not replace μ, then μ is again the service rate vector for the next iteration. Algorithmically

```
replacement ((μ,v),Δλ)
    if $\Delta\lambda \geq 0$ return v;
    if (rand() < exp((Δλ)/T)) return v;
    return μ;
```

Temperature Management

Temperature control during the course of an anneal, known as the cooling schedule, is a much investigated subject in simulated annealing theory. Convergence to a globally optimum objective value is guaranteed eventually if temperature is decreased log hyperbolically as a function of iteration count k,

$$T = \frac{d}{\log(k+1)}$$

The parameter d should be at least as large as the height of all non-global maxima. In practice, this value is not known exactly but any estimate larger than the largest submaximal peak will do. In our work we have found that more rapid cooling performs equally well and so we use hyperbolic cooling

$$T = \frac{d}{d+k}$$

where d and k are as above.

Summarizing, in a simulated anneal potential solutions are generated and compared over and over and either accepted or rejected probabilistically consistent with the Boltzmann distribution law involving a temperature T. During the course of the anneal T is reduced according to a cooling schedule. When finally a suitably low temperature is reached the algorithmn terminates on an optimal or near optimal solution. The algorithmic structure is given as follows:

```
T←1;
μ←Random_Initialization();
λ = evaluate (μ);
k←1;
do
    v←generate (μ);
    Δ λ←evaluate (v) − λ;
    μ←replace ((μ,v) Δλ);
    λ←evaluate (μ);
    $T \leftarrow \frac{d}{(d+k)}$;
    k←k+1;
while (T > exit_T);
```

5. Example

The topology of a communication network model is shown in Figure 2. This network has 5 nodes and 12 half duplex links, so it is regarded as the multichain queueing network.

Window flow controlled network is modeled as a closed queueing network model based on ACKs between source destination pairs. FCFS queues are used to represent forward data path and IS (infinite servers) are used to model random delays (ACKs). The queueing model is shown in Figure 3. Note that servers 1-6 represent FCFS queues and 7-11 represent IS servers.

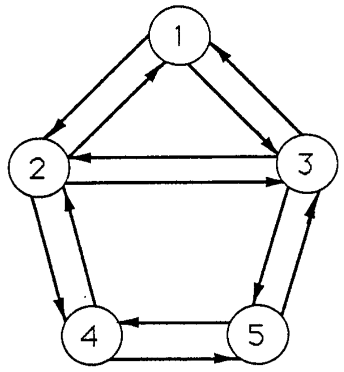

Figure 2. Communication Network

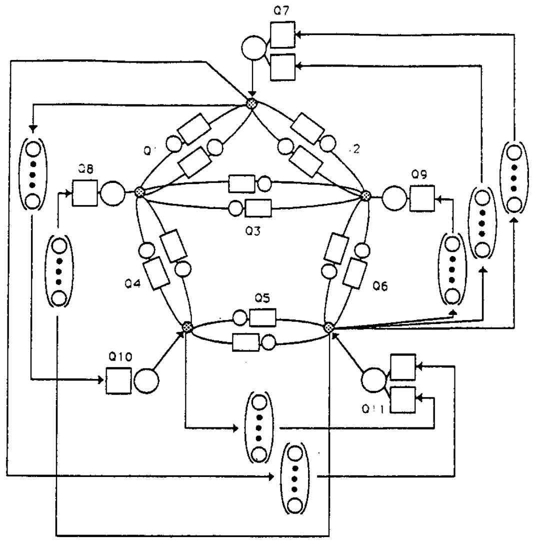

Figure 3. Multichain Queueing Model

It is assumed that routing strategy is fixed and seven routing chains exist. Mean service times of each queue and routing chains are specified in Tables 1 and 2 as input parameters. We assume the uniform service rate and uniform service demand at each queue for convenience.

server	rate c_i	server type
1	500	$\neq$ IS
2	250	$\neq$ IS
3	250	$\neq$ IS
4	100	$\neq$ IS
5	100	$\neq$ IS
6	200	$\neq$ IS
7	400	= IS
8	250	= IS
9	100	= IS
10	300	= IS
11	250	= IS

Table 1. Cost Rates

e_{ir}							
	chain						
server	1	2	3	4	5	6	7
1	1.0	0.0	0.0	0.0	1.0	1.0	0.0
2	0.0	1.0	0.0	0.0	0.0	0.0	0.0
3	0.0	0.0	1.0	1.0	0.0	1.0	0.0
4	1.0	0.0	0.0	1.0	1.0	0.0	0.0
5	0.0	0.0	0.0	0.0	0.0	0.0	1.0
6	0.0	1.0	1.0	0.0	0.0	1.0	0.0
7	1.0	1.0	0.0	0.0	0.0	0.0	0.0
8	0.0	0.0	1.0	0.0	0.0	0.0	0.0
9	0.0	0.0	0.0	1.0	0.0	0.0	0.0
10	0.0	0.0	0.0	0.0	1.0	0.0	0.0
11	0.0	0.0	0.0	0.0	0.0	1.0	1.0

Table 2. Visit Ratios

a_{ir}							
	chain						
server	1	2	3	4	5	6	7
1	1.0	3.0	1.0	2.0	0.7	0.5	0.3
2	0.5	3.5	2.5	2.3	1.9	0.65	4.2
3	0.75	2.0	4.0	4.0	3.6	1.0	5.1
4	2.0	1.5	6.5	1.0	4.2	8.7	5.8
5	2.5	1.75	8.2	0.5	1.0	0.4	6.9
6	1.0	2.0	2.5	3.0	1.0	3.1	1.0
7	2.5	1.0	1.0	2.1	4.0	3.5	4.3
8	1.0	1.0	1.0	3.0	2.0	1.0	1.0
9	2.1	2.0	1.0	4.0	2.5	1.0	2.0
10	1.4	1.5	2.5	2.0	2.6	2.5	1.0
11	1.0	1.5	3.0	2.5	2.1	1.0	1.0

Table 3. Service Rate Factors

We assume that chain population is $2L$, i.e., two times of hop number. So we set population vector to $\underline{K}$ = (4,4,4,4,4,6,2). For the total cost we assume $COST$ = 2700. The results are tabulated in the following Table.

Solutions								
	exponent β							
	1.0	1.2	1.4	1.6	1.8	2.0	2.2	2.4
throughput	9.43	7.31	5.98	5.41	5.26	4.96	4.69	4.60
μ_1	0.036	0.019	0.223	0.207	0.215	0.377	0.251	0.431
μ_2	0.011	0.243	0.685	0.446	0.404	0.458	0.453	0.504
μ_3	2.838	1.844	1.597	1.407	1.410	1.471	1.340	1.302
μ_4	9.976	5.816	3.941	2.702	2.545	2.596	2.265	2.302
μ_5	1.532	0.929	0.660	1.742	1.412	1.160	1.218	1.291
μ_6	0.041	0.631	0.896	0.916	1.122	1.154	1.307	1.101
μ_7	0.036	0.294	0.487	0.391	0.693	0.693	0.672	0.721
μ_8	0.061	0.457	0.704	1.350	1.274	1.130	1.238	1.071
μ_9	3.041	1.982	1.591	1.411	1.252	1.194	1.486	1.400
μ_{10}	0.041	0.033	0.245	0.367	0.358	0.402	0.425	0.462
μ_{11}	2.162	2.438	1.865	1.651	1.411	1.113	0.967	0.992

Table 4. Optimal Throughput and Optimal Service Rates

APPENDIX

$N = 4$ servers, $R = 5$ chains, $K_r = 20$ packets per chain, $COST = 1100$

server	rate c_i	server type
1	500	$\neq$ IS
2	250	$\neq$ IS
3	250	$\neq$ IS
4	100	$\neq$ IS

e_{ir}					
	chain				
servers	1	2	3	4	5
1	1.0	1.0	1.0	1.0	1.0
2	1.0	0.5	1.0	0.333	0.444
3	1.0	0.25	1.0	0.333	0.333
4	1.0	0.25	1.0	0.333	0.666

α_{ir}					
	chain				
servers	1	2	3	4	5
1	6.0	3.0	3.0	4.0	3.75
2	3.0	2.4	2.0	3.0	2.4
3	1.0	1.0	1.0	1.0	1.0
4	1.5	2.0	2.4	2.0	1.5

Solutions								
	exponent β							
	1.0	1.2	1.4	1.6	1.8	2.0	2.2	2.4
throughput	3.21	3.19	3.15	3.11	3.11	3.07	3.04	3.01
μ_1	0.957	0.954	0.913	0.921	0.913	0.881	0.897	0.947
μ_2	0.773	0.733	0.731	0.730	0.743	0.706	0.725	0.726
μ_3	1.410	1.415	1.454	1.384	1.425	1.434	1.350	1.312
μ_4	1.037	1.044	0.963	0.972	0.919	1.025	1.047	0.917

6. Conclusion

The queueing model of the multichain network has been presented where the network is window flow controlled on its virtual channel of communication network. Its performance has been optimized based on Simulated Annealing. In particular, we assumed our solution space to be the set of all admissible service rates of all stations with the nonlinear cost constraint and determined the optimum total throughput of the network for chain r packets. From our experiments we conclude that mean delay time decreases as window size decreases while throughputs change little. Mean delay time increases as window size increases while throughputs change little. Mean delay time reduces as mean service times are shortened while throughputs increase. Finally, the most desirable method against congestion in any region is to speed up the transmission rate of that channel; but this costs too much, so alternative methods can be considered which reduce the window size within the tolerable limit of throughputs.

REFERENCES

1. I. F. Akyildiz, "Performance Analysis of Computer Communication Networks with Local and Global Window Flow Control", *Proc. of the IEEE INFOCOM'88*, March 1988, pp. 401-410.
2. I. F. Akyildiz and G. Bolch, "Throughput Maximization and Response Time Minimization In Queueing Network Models of Computer Systems", *Proc. of Int. Seminar on Distributed and Parallel Systems*, North-Holland, December 1988, pp. 241-259.
3. Y. Bard, "Some Extension to Multiclass Queueing Network Analysis", *Proc. of the Performance'89*, Feb. 1979, Vol. 1.
4. V. Cerny, "Thermodynamical Approach to the Traveling Salesman Problem: An Efficient Simulation Algorithm", *Journal Opt. Theory Appl.*, 45, 1985, pp. 41-51.
5. K. M. Chandy, J. Hogarth and C. H. Sauer, "Selecting Capacities in Computer Communication Systems", *IEEE Transactions on Software Engineering*, Vol. 4, July 1977, pp. 290-295.
6. N. D. Georganas, "Modeling and Analysis of Message Switched Computer-Communication Networks with Multilevel Flow Control", *Computer Networks and ISDN*, North-Holland, Vol. 4, 1980, pp. 285-294.
7. B. Hajek, "Cooling Schedules for Optimal Annealing", *Math. of Operations Research*, 1986.
8. J. R. Kenevan and A. von Mayrhauser, "Convexity and Concavity Properties of Analytic Queueing Models for Computer Systems", *Proc. of Performance' 84 Conf.*, North-Holland, 1984, pp. 361-375.
9. S. Kirkpatrick, C. D. Gelatti and M. P. Vecchi, "Optimization by Simulated Annealing", *Science Journal*, 220, 1983, pp. 671-680.
10. H. Kobayashi and M. Gerla, "Optimal Routing in Closed Queueing Networks", *ACM Transactions on Computer Systems*, Vol. 1, No. 4, Nov. 1983, pp. 294-310.
11. S. S. Lam and J. W. Wong, "Queueing Network Models of Packet Switching Networks: Part 2: Networks with Population Size Constraints", *Performance Evaluation Journal*, North-Holland, Vol. 2, 1982, pp. 161-180.
12. S. R. Lee, W. H. Cho, S. B. Lee and M. Park, "Performance Analysis of Multichain Queueing Networks for Window Flow Control", *Proc. of TENCON 87*, Seoul, Korea, August 1987, pp. 321-325.
13. D. Luan and D. Lucantoni, "Throughput Analysis of a Window Based Flow Control Subject to Bandwidth Management", *Proc. of INFOCOM 88*, March 1988, pp. 411-417.
14. M. Nassehi, "Window Flow Control in Frame-Relay Networks", *IEEE GLOBECOM 88*, Nov. 1988, pp. 1784-1790.
15. M. Reiser, "A Queueing Network Analysis of Computer Communication Networks with Window Flow Control", *IEEE Transactions on Communications*, Vol. 27, No. 8, August 1979, pp. 1199-1209.
16. P. J. Schweitzer, "Approximate Analysis of Multiclass Closed Networks of Queues", *Int. Conf. Stochastic Control and Optimization*, Amsterdam, 1979.
17. J. B. Suk and C. Cassandras, "Analysis and Optimization of Pacing Window Flow Control with Admission Delay", *Proc. of IEEE INFOCOM 88*, March 1988, pp. 391-400.
18. K. Trivedi and R. A. Wagner, "A Decision Model for Closed Queueing Networks", *IEEE Transactions on Software Engineering*, Vol. SE-5, No. 4, July 1979.
19. G. Varghese, W. Chou and A. Nilsson, "Queueing Delays on Virtual Circuits Using a Sliding Window Flow Control Scheme", *Proc. of the ACM Sigmetrics Conference*, Aug. 1983, pp. 275-281.
20. M. P. Vecchi and S. Kirkpatrick, "Global Wiring by Simulated Annealing", *IEEE Transactions on Computer Aided Design*, 2, 1983, pp. 215-222.