Predicting Behavior Patterns using Adaptive Workload Models

J.Srinivasan
Department of Computer Science
University of California
Santa Barbara, U.S.A
jsrini@cs.ucsb.edu

N.Swaminathan
Department of Computer Science and Engineering
Indian Institute of Technology, Madras
Chennai, India
swami@cs.iitm.ernet.in

Abstract

Workload characteristics in a modern networking environment are very dynamic. In order to maximize performance continuously, it is natural to explore the possibility of intelligent systems which can take cognizance of the dynamics in workload and adapt themselves for future control applications. In this paper, we propose a mechanism which represents the previous state of the system as a string. User is allowed to define relevant information for better management as substrings. The adaptive workload model, called the SS[1] *model, predicts the short term future as a string in which the information content conveyed as a substring reflects the future. We illustrate the applicability of the SSR model through web traffic generation and ATM bandwidth management. We use genetic algorithms as the vehicle to address the learning aspects of the model.*

1. Introduction

Performance optimization is one of the major goals in the design and operation of any communication system. In general, resource management will be effective if the requirements are known *a priori*, which implies prediction. Prediction of resource requirements is all the more relevant in systems which exhibit dynamic workload characteristics.

Several models were proposed to describe the workload characteristics of applications [1, 2, 3, 4, 5, 6]. Early researchers have proposed a string based representation of the state of the networked computing environment [7]. In fact, the inherently hierarchical nature of such systems has also been captured using such an approach in the form of Context Free Grammars [8, 9]. The most recent of these models [10] uses Probabilistic Attributed Context Free Grammars (PACFGs) to generate a string representing the state of the system in such a way that the distribution of symbols in a generated string follows the probability distribution corresponding to that of the different classes of events that can occur actually in the system. In a dynamic system, the string representing the state of the system keeps changing. Hence, the workload model has to be adaptive to accurately represent the system by learning some attributes of the substring(s) of interest, embedded within the string. The model thus dynamically keeps pace with the changing system workload behavior. The adaptive workload model makes an implicit assumption that the workload behavior of the system under observation can be learned from the trend in the attributes of the substring(s), based on which, the pattern of behavior in the future can be predicted.

The workload behavior of the system can be defined as the disposition of resource usage over a period of time. The disposition allows us to characterize the behavior of the system during that period of time. From the knowledge of the behavior of the system, reasonable predictions could be made about the upcoming behavior of the system. The prediction that can be made is based on learning some aspects of behavior which we call as *control information*. The user specifies a set of parameters belonging to different dimensions along which the system performance needs to be optimized. The control information then is the pattern of occurrence of the user specified parameters in the behavior. In other words, if the workload behavior of the system can be represented by a string of symbols belonging to an alphabet, the control information is the statistics associated with the pattern of occurrences of the substring parameters specified by the user. Any substring specified by the user, represents a sequence of events that can occur in the system. For instance, we would be interested in knowing about sudden changes in bandwidth demand in a virtual path of an ATM network which could be a parameter specified by the user who is interested in dynamic bandwidth allocation for a virtual path. The control information in a string representing behavior of a virtual path could then be the pattern of sudden changes in demand.

The adaptive workload model needs to learn the work-

[1] <u>S</u>rinivasan-<u>S</u>waminathan

load behavior of the system under observation continuously in order to predict the future behavior. Learning techniques using hidden markov models, artificial neural networks, genetic algorithms, or some regression methods could possibly be employed for learning the behavior. Of these, the genetic algorithms have been used in a variety of optimization problem domains, but they have rarely if at all, been used for dynamic problem domains due to long convergence time. *In this paper, we choose genetic algorithms to learn and predict the workload behavior patterns in a dynamic system* primarily due to the simultaneous learning and predicting ability of the technique unlike other techniques such as artificial neural networks and hidden markov models which require separate training and operational phases. We choose application domains where behavior changes are slow and hence sufficient time is available for the genetic algorithm to converge and provide satisfactory results.

The rest of the paper is organized as follows: Section 2 gives a formal representation of the adaptive workload model. Section 3 describes the genetic algorithm implementation of the adaptive workload model. Sections 4 and 5 discuss applications of the adaptive workload model in web traffic generation and virtual path bandwidth management. Section 6 concludes the paper with a brief description on the scope of the adaptive workload model in other domains such as video traffic management, security, and performance improvement of high-end workstations.

2. The SS Model

The SS model is capable of predicting the future behavior of a system whose state could be represented by a string. The model keeps pace with the changing system behavior by learning some attributes of the substrings of interest that are embedded within the string. An important attribute of a substring is to be able to quantifying the presence of the same which is better illustrated using a formal example as given below:

Example 1: Let us consider a binary string produced by the PACFG of length $2m$, we could define R as the number of 1's just succeeding a 0 in the sequence. For instance, if the string is 010010, R has the value 2. Now consider the set of strings S={ w | w has equal number of 0's and 1's }. (In other words S represents all the strings of length $2m$ which have equal number of 0's and 1's). The set S can be partitioned into $(m+1)$ subsets by R values ranging from 0 to m. At the extremities, we have the sets $1^m 0^m$ with $R = 0$ and $(01)^m$ with $R = m$. The proportion of substring '01' present in 'S' is given by the ratio R/m.

In general, we would like to call the order of alphabets i.e., the substring to be learned as the *concept* to be learned by the learning model. The concept that is chosen should be such that it should have maximum control information. For instance, in an Unix environment at the command level of the workload hierarchy, one may be interested in the number of times the user uses *vi*[2] followed by *latex*[3] followed by *xdvi*[4]. Based on the history of command usage, the learning model can learn the occurrences of the concept and can then predict the future occurrences of the concept based on what has been learned. Occurrences of a concept is the order in which some symbols (representing the user commands) of the alphabet occur in a string. In other words, this order is equivalent to the presence of a substring in a given string. Hence, if we have a count of the number of occurrences of a particular substring in a given string, then the learning model has simply to learn that count. To compare the string the model has learned and the string which the model is attempting to learn, we need a normalized comparison metric. We introduce a normalizing factor to provide a uniform metric for comparison by taking the ratio of the number of occurrences of the substring in a given string to the maximum number of times the substring *could* have occurred in that string. Such a ratio was defined as the r-value of the given string in [11]. The r-value enables us to compare two strings of different lengths to estimate the difference in the concept that each string possesses. Thus, two strings can be compared by comparing their respective r-values.

A good learning model produces strings with r-value very close to the r-value of the string it tried to model. Whenever the r-value of the predicted string does not lie within a predefined range around the r-value of the actual string, we say that the *workload characteristics have changed*. The learning model must now learn the new workload characteristics and adapt itself to generate strings with r-value closer to the new workload characteristics. Such a learning ability of the model is gauged by the *Degree of Learning* (DoL) associated with a concept. The DoL in effect is the comparison of the r-values of the *concept* in the generated command sequence and the actual command sequence. Once, the comparison in terms of the DoL is poor i.e., the value of DoL is low, then it means the learning model needs to learn and adapt itself to the new behavior. Now, the key to the whole issue is the choice of the concept. The concept to be learned is domain specific and is closely related to the dimensions along which one would like to optimize the performance of the system. In this paper, we focus our attention on how to learn the given concept(s), rather than define methodologies for identifying the concept(s) in a given environment.

In a typical networked computing environment one can find one or more concepts of interest to be learned to maximize the performance. When more than one concept is to be learned the relationship among themselves will affect the learning process of these concepts. If the concepts to be learned are related to one another, we consider them to

[2]Text Editor
[3]Text formating software
[4]DVI file viewer

share an OR relationship among themselves. The concepts could also be unrelated to one another, (i.e., they pertain to entirely different dimensions along which one would like to optimize the performance) in which case we consider them to share an AND relationship. The OR and AND relationships among the concepts are illustrated through Examples 2 and 3 and also in Fig.1.

Example 2: Let us consider an environment where we want the model to learn the general trend in occurrences of the substrings $abcd$ and da. In addition, we can suppose that both the strings are equivalent in concept. For instance, the user in a Unix machine may use *vi* to write his mail followed by the program *mail* or he may directly use *elm* or *pine*[5] for the purpose. The former command sequence is equivalent to the latter command in terms of concept. Both imply that the user wrote a mail and sent it. This situation is modeled by the many-to-one-mapped r-value or simply the OR criterion r-value.

Example 3: Suppose we are interested in studying the user behavior in using the *pine* command to send mail and the user behavior in using the *xfig* drawing program. We will expect the learning model to learn the user behavior in using these commands which are not related. But, we can relate them at the logical level with an AND relationship, i.e., the learning model should learn the user behavior in using *pine* AND the user behavior in using *xfig*. This concept related to this situation is captured by the AND criterion r-vector.

In order to bring out the relationship between the OR and AND criterion schemes, let us consider different classes of users in a computing environment as shown in Fig.1. Consider that one class of users are trying to edit, compile and run some programs, while another class is trying to edit a document, draw some figures, compile and view the document. We could consider many such classes of users in the system. The sequences that may be followed by the users are shown in Fig.1. The user sequence workload associated with each class is captured by the OR criterion r-value and the overall workload is captured by the AND criterion *r-vector*.

One can come up with more complicated situations which use a combination of AND and OR relationships. The mathematical formalism for this will be a simple extension of the criteria we propose (the boolean relationship in terms of AND and ORs can always be converted to conjunctive normal form. Then an r-vector whose components consist of OR criterion r-values can be used to model the situation). In the remaining part of this section we develop formal representations for the OR criterion r-value and AND criterion r-vector.

2.1. OR Criterion r-value

The OR criterion r-value is computed by counting the number of occurrences of each of the substrings. Then we normalize each component individually and sum the normalized components.

Let,

S	:	Any string
S_a	:	Actual string
S_p	:	Predicted string
L	:	Length of the given string
s_1 .. s_k	:	Substrings 1 to k (*concepts*)
l_1 .. l_k	:	length of s_1 to s_k respectively
R_i	:	Number of occurrences of s_i in S
M_i	:	$\lfloor L/l_i \rfloor$
$r(S)$	:	r-value i.e., Normalized index of order information contained in S

We follow the convention that if we find two occurrences of the same substring which overlap, then we count only one of them, i.e., we count only disjoint occurrences of a substring. For instance, in the string "1111" if the the substring of interest is "11", then we consider the number of occurrences of this substring as two, i.e., R = 2, and the maximum number of possible occurrences of this substring in the given string also to be two, i.e., M = 2. Thus M_i is equal to $\lfloor L/l_i \rfloor$ according to this convention. Then, the OR criterion r-value is given by:

$$r(S) = \sum_{i=1}^{k} R_i/M_i \tag{1}$$

the normalized difference in control information in S_p and S_a is given by:

$$r_{normalize} = \sqrt{r(S_p)^2 + r(S_a)^2} \tag{2}$$

$$r_d(S_p, S_a) = \frac{\mid r(S_p) - r(S_a) \mid}{r_{normalize}} \tag{3}$$

the Degree of Learning for OR criterion learning is given by:

$$DoL = 1.0 - r_d(S_p, S_a) \tag{4}$$

This r-value is independent of the lengths of the substrings. Other modifications suitable to the application at hand can be made. Some such criteria are discussed in the subsequent sections. The thumb rule to be followed is:

1. Choose the substrings which contain information.
2. Count the occurrences of each substring.
3. Normalize the vector such that the values of the individual elements lie between 0 and 1.
4. Add the individual components to compute the OR criterion r-value.

[5]Both elm and pine are user side email management softwares

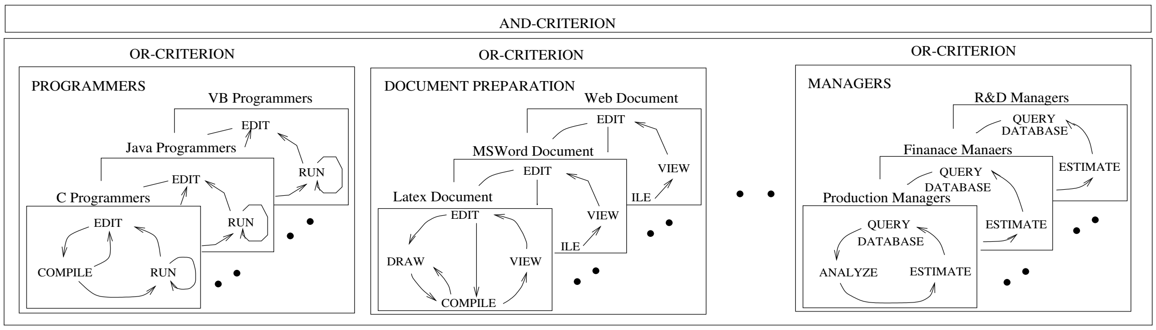

Fig. 1. Relationship between OR criterion r-value and AND criterion r-vector

2.2. AND Criterion r-vector

The next parameter of learning that we define is the *r-vector*. This index comes into play when the model is interested in learning more than one concept, which are mutually exclusive. The AND criterion r-vector is computed by counting the number of occurrences of each of the substrings and treating them as components of a vector. In other words, if there are 'k' substrings we can visualize a k-dimensional vector space in which each substring represents a dimension. For each substring s_i, we shall assume a unit vector $\hat{s_i}$. Now given this vector of occurrences we can normalize it to get the r-vector.

$$\vec{R}(S) = R_1\hat{s_1} + R_2\hat{s_2} + ... + R_k\hat{s_k} \tag{5}$$

$$r_{normalize} = \sqrt{R_1^2 + R_2^2 + ... + R_k^2} \tag{6}$$

$$\vec{r}(S) = \frac{R_1\hat{s_1} + R_2\hat{s_2} + ... + R_k\hat{s_k}}{r_{normalize}} \tag{7}$$

for notational convenience (7) can be represented as:

$$\vec{r}(S) = r_1(S)\hat{s_1} + r_2(S)\hat{s_2} + ... + r_k(S)\hat{s_k} \tag{8}$$

the normalized difference in control information in S_p and S_a is given by:

$$\vec{r_d}(S_p, S_a) = a_1\hat{s_1} + a_2\hat{s_2} + ... + a_k\hat{s_k} \tag{9}$$

where, $a_1, a_2, ..a_k$ are $\mid r_1(S_p) - r_1(S_a) \mid$, $\mid r_2(S_p) - r_2(S_a) \mid$ and $\mid r_k(S_p) - r_k(S_a) \mid$ respectively. The Degree of Learning for AND criterion learning is given by:

$$DoL = 1.0 - \mu \tag{10}$$

where, μ is the mean of $a_1, a_2, ...a_k$. In the definition of DoL for an AND criterion, *the difference of r-vectors* is mapped to a scalar value in the range 0 to 1. The DoL which represents the similarity of the concept can therefore be computed from this scalar value as shown in (10). The simplicity of the mathematics involved in constructing the r-values and r-vectors contributes to the elegance of this formalism.

3. Implementation of the Adaptive Workload Model Using Genetic Algorithms

In this section, an implementation of the proposed adaptive workload model using genetic algorithms for learning is discussed. A genetic algorithm is a heuristic approach which applies the natural genetic ideas of natural selection, mutation and survival of the fittest. The algorithm uses a set of offered solutions called a *population*. Each solution called an *individual*, can be any solution in the solution space represented by a string called *chromosome*. The solution space is explored in order to find an optimal solution which satisfies the constraints posed by the problem. A solution is judged as optimal based on a criterion called the *fitness value*. The current population is evolved creating a new population with higher fitness. The fitness of each individual of the new population is evaluated. If an optimal individual is found then the process is terminated, else the next generation of individuals is evolved. The same process continues until a solution is found.

Coding: In nature, an allele[6] is encoded in an alphabet composed of four symbols. Similarly, in genetic algorithms we use strings composed of alphabets. We define a string composed of such alphabets as a *loadgene* (load + gene). In certain problem domains of networking, the state of the system can be represented as a string of alphabets. Table 1 shows the alphabet used in two such domains - Virtual Path bandwidth management in an ATM network and Networkload modeling of WWW traffic (at the level of methods). The method-prediction problem has an alphabet composed of 5 symbols. These symbols correspond to different classes of methods. In the bandwidth demand pattern prediction prob-

[6]An allele is a string representing a characteristic of an organism

Web Proxy		VP Bandwidth Demand	
Method	*Symbol*	*Bandwidth Range*	*Symbol*
GET	1	0-64kbps	1
POST	2	64-128kbps	2
ICP_QUERY	3	128-192kbps	3
HEAD	4	.	.
CONNECT	5	.	.
		1472-1536kbps	24

Table. 1. Alphabets used for WWW Methods and VP Bandwidth Changes

lem, the alphabet corresponds to discrete ranges of bandwidth demands. Thus a loadgene in method prediction represents a sequence of methods. In the latter case, a loadgene represents the bandwidth demand sampled at regular intervals of time. The problem, in general is to predict the next loadgene given the previous few loadgenes. The predicted loadgene reflects the state of the system as it will be in the near future. The methodology adopted is to generate a population of sample loadgenes, each obeying the distribution of symbols in the previous loadgene. The next loadgene is then predicted by evolving the population using a suitable fitness criterion. The fitness criterion incorporates the information relevant to the user in terms of better management of the system.

Fitness Criterion: The fitness of a living organism corresponds to its ability to survive the stress factors in its environment. In natural selection, the best or the fittest organism is selected to live and propagate further. The aim of our problem is to *learn* from the relationship among previous loadgenes and predict the future loadgene. The range of difference in the control information (represented by the r-value of the loadgenes) between k successive loadgenes is observed and set as a parameter for prediction. The difference in control information between the predicted loadgene and the current loadgene should lie within this observed range in order to keep pace with the trend in behavior. The difference in control information of a possible pattern and the current loadgene when compared with the limits of the observed range (of differences over k previous loadgenes) provides us with a measure of fitness for the candidate loadgene. Candidates with high fitness values are preferentially chosen to propagate and reproduce further. Thus natural selection is incorporated.

The absolute difference in the r-values of two loadgenes: S_1 and S_2 is denoted by $r_d(S_1, S_2)$. If $S_k, S_{k-1}, ..., S_1$ represent the k previous loadgenes that have been sampled where S_1 is the most recent loadgene, and m and M represent the minimum and maximum respectively, of $r_d(S_k, S_{k-1}), r_d(S_{k-1}, S_{k-2}), ..., r_d(S_2, S_1)$ then the fitness value, F, of candidate loadgene S can be formally defined as in Fig.2.

if $(r_d(S, S_1) < m)$ then
    $F = 1.0 - (m - r_d(S, S_1))$
else if $(r_d(S, S_1) > M)$ then
    F =1.0 $- (r_d(S, S_1) - M)$
else
    F =1.0

Fig. 2. Fitness Criteria

Clearly, a candidate with higher fitness is one which lies close to the permissible range. The entire algorithm hinges on the fact that the next loadgene will share a relationship with the current loadgene which is very similar to the relationship that existed among the previous k loadgenes. The DoL is given by the following equation.

$$DoL = 1.0 - r_d(S_{predicted}, S_{actual}) \qquad (11)$$

Thus, if the r_d value of the predicted loadgene is small, then the DoL is near 1.0. A high value of DoL signifies that the predicted loadgene has control information very similar to that of the actual loadgene that arrives.

The Substring Parameter: In the previous sections, a mathematical formalism was developed which mapped the control information of a string to a number or vector. The control information is a user defined parameter and corresponds to what the user believes is important for managing the system. This information is encoded as a string of alphabets using the same procedure for converting the state of the system into a string. Then, by gauging the amounts in which this information is present in the system (by counting the number of occurrences of the substrings and then normalizing the count) in the recent past an estimate is made of the amount in which it will be present in the near future using the genetic algorithm. The genetic algorithm thus learns the *relationship* between the amounts of information present in the previous few loadgenes and predicts a future scenario in which the amount of information (i.e., the number of occurrences of the substring) shares the same relationship with the previous loadgenes.

4. Application of the Adaptive Workload Model in Web Workload Generation

In this section, we discuss a simple application of the proposed SS model in web workload generation. In particular, we consider *method* generation as seen in a typical web proxy server. The log files of the web proxy server at Indian Institute of Technology, Madras, India, were used by the adaptive workload model to learn the pattern of arrival of method requests at the proxy. A sequence of 100 method requests was treated as a loadgene. Three successive loadgenes from the proxy server log file were used to predict the next loadgene, i.e., the next 100 requests. The r-value of the predicted loadgene was then compared with the r-value of the next 100 requests in the log file to obtain the DoL. Since we cannot predict the exact sequence of requests, we would at least like to predict which method occurs after a particular method. This guided our choice of the substring parameters discussed in Section 3. The substrings chosen were: 11, 12, 13, 14, 15, 22, 23, 24, 25, 33, 34, 35, 44, 45 and 55. If any of these substrings occurred, they contributed to the control information of the string. The proportion of symbols representing the methods generated by the SS model is shown in Table 2. The proportion of symbols generated is well within 0.01% of the symbols actually generated.

Symbol	1	2	3	4	5
Actual (15200)	4026	3180	2739	2421	2834
Predicted (15200)	4024	3191	2738	2417	2830

Table. 2. Proportion of Symbols in Actual and Predicted Sequence

We also tried to see if the model is able to learn the pattern if the symbols are a function of the previous few symbols. To change behavior, the function was changed at intervals. Fig.3 shows the functions used and the number of symbols generated by each function while Fig.4 shows the DoL of the model when it tried to learn these patterns. Except the second function which is a random function, the DoL is found to be above 0.8 showing that the adaptive model is able to follow the changes in behavior.

Function 1:
Loadgene[i] = (Loadgene[i-2]*Loadgene[i-1]) % 5 + 1
Amount of Data = 2000

Function 2:
Loadgene[i] = (rand() % 5) + 1
Amount of Data = 4500

Function 3:
Loadgene[i] = (Loadgene[i-4]*Loadgene[i-1] + Loadgene[i-3]) % 5 + 1
Amount of Data = 3000

Function 4:
Loadgene[i] = (Loadgene[i-4]*Loadgene[i-1]*Loadgene[i-3]) % 5 + 1
Amount of Data = 3000

Function 5:
Loadgene[i] = (2*Loadgene[i-4] + Loadgene[i-1])% 5 + 1
Amount of Data = 3000

Fig. 3. Functions used to Generate Data

The application described in this section was quite straight forward and simple and shows how to define sub-

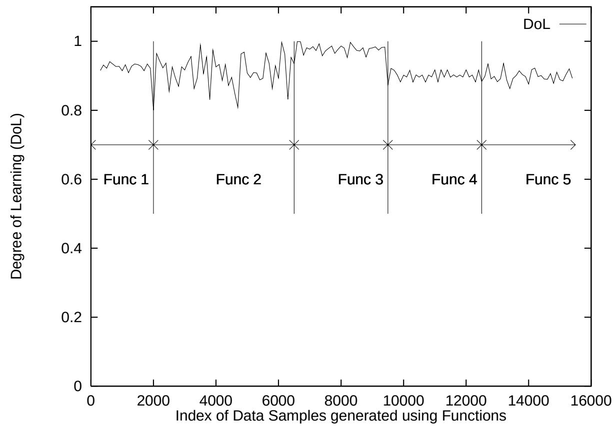

Fig. 4. Performance of Adaptive Workload Model with Input Data Generated using Functions

string parameters for a given context and also shows how the model can dynamically adapt to changing system behavior. In the next section, we discuss a more realistic application of the SS model.

5. Application of the Adaptive Workload Model in VP Bandwidth Management

In this section, an application of the adaptive workload model in virtual path bandwidth management in ATM networks is described. In an ATM network, transmission efficiency can be improved by dynamically changing the bandwidths of the virtual paths (VP) according to the changing traffic [12, 13]. The bandwidth demand within a VP depends on the traffic pattern of the call arrivals experienced by the VP. The aggregate bandwidth demand of the individual calls within a VP can be mapped to discrete bandwidth ranges. These ranges in turn can be mapped to symbols of an alphabet as shown in Table 1. Thus, the characteristics of the VP bandwidth demand sampled at regular time intervals can be encoded as a string composed of the symbols of the alphabet. The string of symbols now characterizes the bandwidth demand pattern within a VP.

The call level throughput of an ATM network could be improved if the VP bandwidths are altered in accordance with the VP bandwidth demand patterns. The VP bandwidth control mechanism should respond to drastic changes in demand in a VP. The drastic change in demand in a VP could be recognized by the adaptive workload model by dynamically changing the threshold value for the difference between two successive samplings of bandwidth-demands. If the difference is greater than the current value of the threshold, then the change in demand is deemed serious enough to warrant a contribution to the control information of the loadgene. The threshold value is dynamically changed by observing the minimum and maximum differences in successive symbols present in the previously arrived loadgene. The threshold is then set as a linear function of the minimum and maximum differences. Simulation study was done on VP bandwidth management using the adaptive workload model for up-coming bandwidth demand prediction in a VP. A description of the simulator and the results are discussed in the next section.

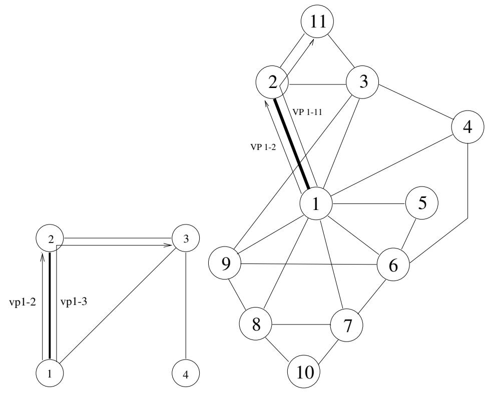

Fig. 5. The Networks used in the Simulation

Description of the Simulator: A call level ATM simulator was developed in C language for the purpose of simulating the call level traffic in a typical ATM network. Two physical network topologies were considered for the simulation as shown in Fig.5. All the VPs are assumed to be unidirectional. Call arrivals are assumed to be Poisson and the call holding times to be exponentially distributed. The calls are assumed to be of three types based on their bandwidth-demand and duration of calls, the details of which are given in Table 3. The proportion of calls of type A, B and C generated are 90%, 9%, and 1% respectively. The traffic from any node to any other node in the network is uniformly distributed. The traffic in the VPs named vp1-2, vp1-3 of the 4-node network and vp1-2, vp1-11 of the 11-node network were monitored. The physical links which these VPs share are shown in thick line in Fig.5. The bandwidth-demand in these VPs were sampled at intervals of one second and one day's worth of data was simulated and used for the analysis.

The maximum bandwidth of the physical link for the simulation purposes was intentionally chosen to be small in order to deal with less alphabets. However, the approach is scalable because a higher maximum bandwidth will only result in the cardinality of the alphabet set becoming larger. We consider T1 (1.5 Mbps) type of physical links and symbols as defined in Table 1, hence the cardinality of the alphabet set was 24.

Prediction of Bandwidth Demand: The bandwidth demand in a VP is sampled at 1 second intervals. The bandwidth samples are collected for an observation period of 10 minutes. The bandwidth demand sampled every second form the symbols of the loadgene. These symbols are presented in Table 1. The set of symbols for the entire 10 minutes is a loadgene. Two previous actual loadgenes and the predicted loadgene for the current 10 minute interval are used to predict the loadgene for the next 10 minutes. This ensures that a loadgene is predicted before it is actually sampled. From the population, the loadgene whose fitness value is 1.0 is predicted as the solution. If a solution is not found in the current population, next generation is evolved and searched for solution. Upto 100 generations are evolved to search for highest fitness solution. The trade-off between the time for convergence and solution fitness are discussed in detail in [14]. The actual bandwidth demand is determined from the predicted loadgene by considering the mean, standard deviation, maximum and minimum of the demands within the predicted loadgene.

Call type	*Bandwidth-demand per call in 64kbps units*	*Mean Call Duration*	*Call Blocking Prob. Acceptable*
A	1	3 min	0.03
B	3	30 min	0.03
C	6	15 sec	0.03

Table. 3. Call Characteristics used in the Simulation

Simulation Results: The effectiveness of the VP bandwidth management using the adaptive workload model is validated by means of simulation. The simulation results shown are with a confidence interval of 95%. The value of the bandwidth predicted by the proposed model for the 4-node and 11-node networks are shown in Fig.6. The demand during the period of 36000 to 40000 was raised by 25% to test the adaptability of the proposed model. The bandwidth allocation is able to follow the demand for the 4-node network very closely while for the 11-node network during the period from 36000 to 40000 seconds the allocation is far below the demand. The low bandwidth allocation for VP1-11 in the 11-node network is due to the sharing of physical link bandwidth by many VPs flowing through it. More detailed study on the VP bandwidth management using the adaptive workload model such as performance comparison with constant peak assignment, performance with Poisson and self similar call arrivals, and the cost, effect and applicability of genetic algorithm are reported in [14].

6. Conclusion

Creating adaptive workload models that can learn and adapt themselves to any environment has been the *ultimate*

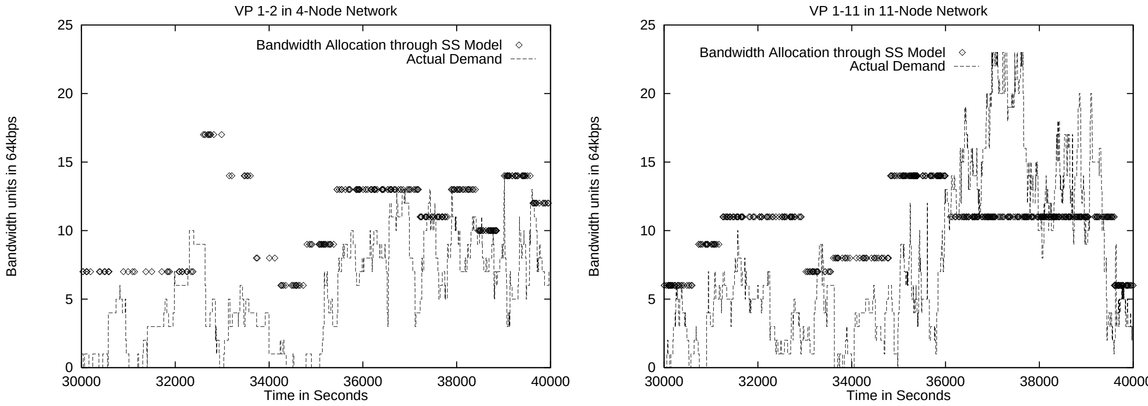

Fig. 6. VP Bandwidth Allocation for the 4-Node and 11-Node Networks through the SS Model

goal of any workload modeler. Attempts such as dynamic models [3] and time-based models [4] are endeavors in this direction. The primary concern of a modeler working with learning and adaptive models is to quantify the ability of the model to learn, thereby quantifying the intelligence associated with the model. In this work, we have addressed these concerns as well as successfully applied our solution to real environments such as Web traffic generation and Virtual Path bandwidth management in ATM networks. The work may be extended to other application areas such as *MPEG Video Stream Modeling* where the scene change behavior could be characterized and predicted in order to improve video transmission quality by advance network resource reservations. Another potential area where the proposed model could be applied is in *Intrusion Detection* where intruders logging into a secure computer system can be detected by comparing their behavior with usage pattern of authorized users. *Intelligent Prefetching* is another application domain for the proposed model to improve the performance of high-end systems by characterizing the IO operations.

7 Acknowledgements

We sincerely thank all the reviewers for their suggestions for improving the readability of this paper. We also wish to thank Dr. Anurag Acharya of Univ. of California, Santa Barbara for his comments and suggestions for extending the scope of this work.

References

[1] Ferrari D. Workload Characterization and Selection in Computer Performance Measurement. *Computer*, 5(4):18–24, 1972.

[2] Ashok K Agrawala, Mohr J M and Bryant R M. An Approach to the Workload Characterization Problem. *IEEE Computer*, 9(6):18–32, June 1976.

[3] Haring G. On State-Dependent Workload Characterization of Software Resources. *Proc. ACM Sigmetrics Conf.*, pages 51–57, 1982.

[4] Calzarossa M and Serazzi G. A Characterization of the Variation in Time of Workload Arrival Patterns. *IEEE Trans. on Computers*, C-34(2):156–162, 1985.

[5] Calzarossa M and Serazzi G. Work-load Caracterization: A Survey. *Proc. of the IEEE*, 1981(8), 1993.

[6] Ashok K Agrawala, Leyuan Shi and Paola Rossaro. A Workload Model for Multi-Window Environments. *Technical Report: TR3246, University of Maryland, U.S.A*, 1994.

[7] Winfried Dulz and Stefan Hofmann. Grammar-based Workload Modeling of Communication Systems. *Proc. TOOLS'91 Conference, Torino, Italy*, pages 16–30, 1991.

[8] Raghavan S V, Vasukiammaiar D, and Gunther Haring. Generative Workload Models for Single Server Environments. *Proc. of ACM SIGMETRICS '94, Conference on Measurement and Modeling of Computer Systems, Nashville, Tennesse, USA.*, pages 118–127, May 1994.

[9] Raghavan S V, Vasukiammaiar D, and Gunther Haring. Hierarchical Approach to Building Generative Workload Models. *Computer Networks and ISDN Systems*, 27:1193–1206, 1995.

[10] Raghavan S V, Prabhakaran B, and Satish K Tripathi. Synchronization Representation and Traffic Source Modeling in Orchestrated Presentation. *IEEE Journal of Selected Areas in Communication*, 14(1):104–113, Jan 1996.

[11] Raghavan S V, Guenter Haring, Srinivasan V and Vishnu Priya N. Learning Generators for Workloads. *Proc. of Workshop on Workload Characterization in High Performance Computing Environments, MASCOTS'98, Montreal, Canada.*, pages 1–81 to 1–88, July 1998.

[12] Ohta S and Sato K. Dynamic Bandwidth Control of the Virtual Path in an Asynchronous Transfer Mode Network. *IEEE Trans. on Communications*, 40(7):1239–1247, July 1992.

[13] Burgin J and Dorman D. Broadband ISDN Resource Management: The Role of Virtual Paths. *IEEE Communications Mag.*, 29(9):44–48, Sept 1991.

[14] Swaminathan N,Srinivasan J and Raghavan S V. Bandwidth-Demand Prediction in Virtual Path in ATM Network Using Genetic Algorithms. *Computer Communications*, 22(12):1127–1135, July 1999.