Capturing a Qualitative Model of Network Performance and Predicting Behavior

by

Sule O. Ibraheem [*]

Mieczyslaw M. Kokar [**]

Lundy Lewis [***]

ABSTRACT

This paper describes a method for constructing behavior models of communication networks. The method utilizes archived quantitative performance data created by a network management platform to create a Quantitative/Qualitative (Q^2) Dynamic System representation. The Q^2 representation captures the predominant qualitative (symbolic) states of the network, qualitative input events and transitions among the states resulting from these events. This symbolic model allows the network manager to understand the current system behavior, and predict future possible behaviors. We evaluated the method on two sets of archive data. The method shows promise for use in network management, including network monitoring, fault detection, prognostication and avoidance.

Keywords: network performance, qualitative model, network behavior, behavior prediction

* Digital Equipment Corporation, Shrewsbury, MA 01545

** Northeastern University, 360 Huntington Avenue, Boston, MA 02115; ph: (617) 373-4849, fax: (617) 373-2501; e-mail: kokar@coe.neu.edu (all correspondence should be directed to this author)

*** Cabletron Systems, Inc., Nashua, NH 03063

1. INTRODUCTION

Modern communication networks are very complex dynamic systems. The high complexity of networks has many causes: complex network topology (many subnets, routers, nodes), different communication protocols used by subnets (e.g., Token Ring, Ethernet), congestion due to collisions and bottlenecks, varying demands imposed by the users, and different transmission speeds on various nodes. A typical view of network topology (using the Cabletron's Spectrum network management platform representation) is shown in Figure 1. The subnets are represented by the oval shaped icons while the routers are represented by the rectangular shapes. The diamond shaped icons are the links that connect router's external ports. This is a portion of the Cabletron Corporate Network, which includes subnets contained in Rochester, Durham, Merrimack, and Nashua, New Hampshire. Figure 1 shows the Nashua facilities (on the left) and the Merrimack facility (on the right).

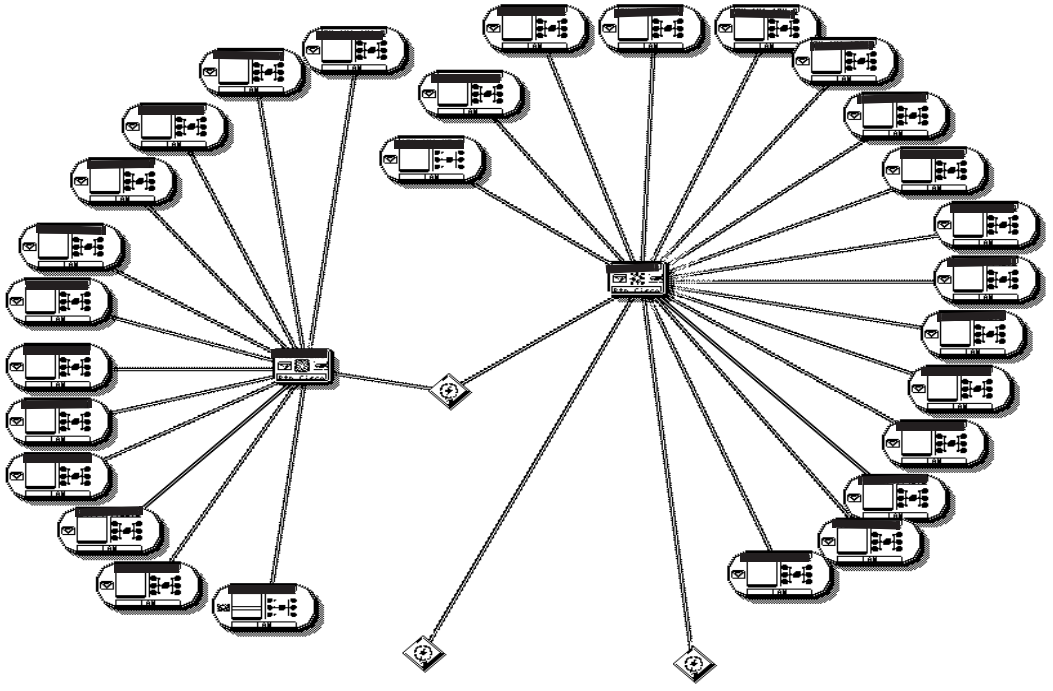

Figure 1: Topology of a Typical Network Model

Communication networks need to be managed very carefully and in a timely fashion in order to achieve optimal performance. Network management consists of monitoring and control. Monitoring consists of information gathering and information interpretation in terms of a network model. Problems identified during network monitoring are addressed by determining and taking appropriate control steps in order to avoid significant decrease in network throughput [1].

Network information can be classified into three categories [2]: *static* (information on current configuration of a device, such as port identification and number of devices on the network), *dynamic* (event-related information, such as the transmission of a packet on a network), and *statistical* (information that may be derived from dynamic information, such as the average number of packets transmitted per unit of time). The static information parametrizes network models. The statistical information describes global behaviors of the network. The dynamic information collected from each device allows the network manager to monitor the current performance of devices on the network.

Each device on a network can be configured to report certain status information. These status reports can quickly accumulate, overwhelming the processing resources of the system manager. We require a method to identify *qualitative states* (based on this data) of interest to network managers. Typical managers' questions related to various qualitative states (events) are: "Is there *excessive traffic* on certain areas of the network?", "Are there *bottlenecks* in any part of the network?", "Is network *response time increasing or decreasing*?", "Is network *error* or *collision rate increasing* or *decreasing*?", "What is the level of network *capacity utilization*?" and "Has *network throughput* been reduced to unacceptable levels?" For each of these questions, the manager would like to know when the event occured, whether the event will occur in the future, and how the input could be changed to prevent the occurrence of one of these undesirable qualitative states.

These qualitative states are not the same states as quantitative states described, for instance, in [3]. These qualitative states are aggregates of the quantitative states, and as a result, the qualitative dynamic model of the network will consist of a much smaller number of states so that the model is amenable to human interpretation and analysis. The network manager can combine his/her knowledge with the model in the process of network analysis. Such a functionality is not available in

any of the existing network management tools. For instance, tools like OPNET, COMNET III, BONeS DESIGNER, and others, employ exact mathematical (quantitative) models for network simulation (cf. [4],[5]). Several of these products have been integrated with network management platforms (NMPs) such as Cabletron's Spectrum, HP's Openview and, IBM's Netview, in order to take advantage of the NMP's knowledge of network topology and network traffic data ([6],[7]), but they do not include qualitative models that can be used for network management and capacity evaluation and planning. These tools are useful in controlled situations, i.e., when an exact mathematical model is known and when the amount of data is relatively small. But in practice, the amount of processing time of all the quantitative data gathered by a monitoring tool and the storage requirements are often exorbitant and thus prohibitive. Also, the required models of vendor-specific network devices, such as hubs and routers, are hard to get because of business concerns.

Lacking such models, network management decisions are usually made by experienced network managers who have the knowledge of the network model. Consequently, they know how to monitor the network and what control actions to apply when such events occur. The managers, however, perform all these actions based on their *internal mental model*, which is not explicitly formalized. A manager who manages thousands of nodes with this approach would be difficult to replace with either another person or a computer program. Clearly, there is a definite need to assist the manager with a computer tool that can learn and analyze explicit computational network models [8]. The methodology proposed in this paper can be used in the development of a strategic tool of traffic evaluation with the objective of anticipation.

The explicit computational model of the network must account for the network's dynamic behavior. It must relate the future network behavior to the current network input and the state. The current input is measured using a network monitoring tool and the current state is estimated based upon current and past measurements. The measurements give quantitative values of the parameters characterizing the network behavior, for example number of packets, collision rate, error rate, and load. These quantitative measurements can be analyzed in terms of a quantitative model of the network. However, purely quantitative models are difficult for human network managers to interpret, which makes such models inappropriate for use as a network manager's tool. Instead, a *hybrid model* is needed that can abstract symbolic qualitative information from the quantitative

data and give qualitative interpretation that is easily understood.

Models can be either derived analytically or constructed empirically using the data collected from the modeled system. For such complex systems as communication networks, analytical derivation of models is not possible, and thus the empirical approach needs to be used. Empirical model derivation can be accomplished in a number of steps: selecting model attributes (variables), selecting classes of functions and relations to be used in the model, gathering data on the selected attributes, model fitting (adjusting model parameters based upon the collected data so that the model captures the relationships represented by the data). The selection of the model variables depends on the goal for which the model is to be used. In our case we are interested in network performance and thus we are interested in those variables that are relevant to performance.

The network performance data that vendors must make available to network managers are determined by the ISO standard on the Management Information Base (MIB) [9], [10]. The MIB data is collected by a monitoring application and stored in the archive files for all the objects that are being managed. The following are some performance related attributes: Soft Error Rate, Hard Error Rate, Hard Error Count, LAN Ethernet Collisions, LAN Ethernet Packets, Ethernet Collision Packets, HUB Actual Packets, HUB Actual Collisions, HUB Total Collisions, HUB Total Packets, HUB Board Packets, Packet Rate, Total Packets, Load, Contact Status, Device Contact Status, IP Forward Datagrams, IP IN Delivers, and IP IN Receives. Most of these attributes are common to all devices and come directly from the MIB variables. Derived attributes are also common, but their derivation may vary from device to device and from vendor to vendor.

We are interested in modelling both quantitative and qualitative aspects of the network, and thus we require a *hybrid dynamic system*. We model the quantitative behavior of the network using *general dynamic systems (GDS)* and the qualitative counterpart with *qualitative dynamic systems (QDS)*. The main idea of the approach is based on partitioning the space of the *GDS* into classes (regions) and treating each of the regions as a qualitative variable. Since the ultimate goal for us is reasoning, we pay special attention to the requirement of *consistency of reasoning* with the qualitative model. We interpret “consistency” in the logical sense: the representation is consistent if the qualitative model of the network always correctly classifies the network's current and/or future

qualitative behavior.

Consistent representations must have the ability to capture all the system parameters (input, state, and output) and must always derive correct conclusions. Unfortunately, these requirements are not satisfied by many approaches presented in the literature. For instance, the two most typical approaches to qualitative representation, landmark points and fuzzy sets, do not fulfill these requirements. Both of these methods derive qualitative representations through partitioning of the state space, but they do not guarantee consistency. The landmark points approach does not guarantee a unique correct classification of the qualitative behavior, and the fuzzy set approach is always non-unique.

The Q^2 representation, as described in [11], employs *critical hypersurfaces* to delineate boundaries of qualitative regions in the system space. The critical hypersurfaces can be derived when the quantitative dynamic model of the system (analytical form) is known. Also, the logical formulation of consistency holds only when the system is deterministic. For communication networks, such analytical models are not known and also networks cannot be modelled as deterministic systems. Therefore, to deal with these problems, we use a clustering technique to approximate critical hypersurfaces and a non-deterministic finite state automaton to model the state transitions. We also simplified the Q^2 by predicting the next state based only on the current state and input; the system does not take into account the delta-time that it will remain in a state before switching to the next state.

In Section 2 we give an overview of related research.The Q^2 representation is described in Section 3. In Section 4 we show steps that we have taken to develop a network behavior model using this representation. Section 5 describes methods (greedy and random) used for predicting next states. We tested our Q^2 representation on two sets of archived data: a five week set and an eighteen week set. Results of our experiments with predicting network's next qualitative states are presented in Section 6. Conclusions and suggestions for future research are presented in Section 7. All the details are described in [12].

2. RELATED RESEARCH IN NETWORK PERFORMANCE ANALYSIS

Numerous research studies have been completed in the area of network performance analysis. This section briefly summarizes some of the relevant research.

Braden in [19] introduces a Packet Monitoring Program (PMP). PMP analyzes traffic patterns and gathers network statistics in real-time. One advantage of PMP over other network analyzers is that it can be optimized to filter certain parameters and to check for certain redundancy based on the programming criteria. PMP analyzes incoming packets and saves only "interesting" statistical events and keeps a frequency count and cumulative statistics of individual protocols. However, since it does not time stamp events, a network administrator cannot determine when events occur. A disadvantage of PMP is that it can only analyze one branch or one point-to-point connection between two nodes. Therefore, it has to be placed at the main trunk where most traffic passes to get the full benefit of analyzing a whole network node. Also, since PMP is an on-line network analyzer, it provides no functionality for analyzing archived data.

Amer and Cassel [20] describe several network measurement programs that provide the network managers with real-time notifications of network irregularities based upon a statistical model of a network. Effective utilization of these tools required knowledge of which statistics to monitor and characteristics of interesting events. Since data collection and analysis are done concurrently, it is important to have efficient algorithms for both of these functions in order to avoid loss of information. Lost packets can result in inaccurate estimates of network behavior. Alternatively, instead of analyzing all packets, the information can be selected using a random sampling method. Amer and Cassel [20] discuss two types of sampling methods. While the Simple Random Sampling method provides a low variance estimate of the sampled population, the Systematic Random Sampling method provides less precise estimates but with minimum computational overhead. Amer and Cassel have also experimented with a moving window approach, which was tested in conjunction with Simple and Systematic Random Sampling. The authors suggest the following guidelines when sampling network data [20]: (1) select traffic frames at random with no preference for, or avoidance of, any particular class of traffic; (2) select traffic frames as often as possible without interfering with time allotted for CPU calculations; and (3) contribute minimal

overhead to the operation of the monitor.

Fowler and Leland [21] investigated network characteristics based on analysis of archived data and suggested ways to manage traffic congestion for the LAN. They also investigated the difficulties involved in capturing a model of network performance. They tried to reproduce and determine the causes of network congestion based on different components of the network, like the multiplexer, network packet switching, and different protocols.

Histon [22] investigated an approach to understanding the behavior of TCP/IP protocols. He used a knowledge-based system called Knowledge-Based Network OBServer (KNOBS/TCP). The rules for the expert system were developed by collecting data from several network experts. The input to the expert system consists of network topology and statistics. KNOBS is good for answering generic questions about a network's global behavior and explanation of possible causes of failure. KNOBS, however, cannot capture and predict network behavior, since its rules are not specific enough to capture varying statistical and dynamic information for each network node. Human experts cannot provide this kind of detailed knowledge.

Most of the research done in the area of network management uses live network data and a manually constructed model of the network to determine network performance. The use of archive data to study network performance is not widespread. Among different approaches to network monitoring we were able to identify a statistical approach and an expert system approach. The statistical approach captures some of the regularities and patterns in network behavior, but it does not account for the dynamic behavior of the network. It does not treat the network as a dynamic system. Also, the quantitative character of the statistical model makes it difficult to interpret and manipulate by most human managers. The current expert system approach, while much more user-friendly, does not provide flexible and adaptable mechanisms for capturing quantitative knowledge about the network. The Q^2 approach proposed in this paper lies in between the two approaches and capitalizes on the advantages of both. Our Q^2 method examines quantitative archive data and constructs a symbolic representation of network behavior in form of a finite state automaton.

3. MODELING NETWORK BEHAVIOR WITH Q^2

A communication network is a dynamic system. Methods for modeling dynamic systems have been studied primarily in systems science and general systems theory (cf. [13], [14]). However, these methods lead to quantitative dynamic models, which, as we stated earlier in this paper, are not easy for human network managers to deal with. Our main objective is to take this infinite state dynamic system and abstract a simpler finite-state qualitative dynamic system that preserves consistency, meaning that the result of reasoning within the qualitative structure must hold in the underlying quantitative dynamic system.

We model the network as the quantitative/qualitative (Q^2) dynamic system [11]. This method formalizes a quantitative structure in terms of a *general dynamic system* (*GDS*) and the qualitative structure in terms of a *qualitative dynamic system* (*QDS*). The qualitative structure is represented by a finite-state automaton (*FSA*) and *qualitative abstractions* that map the *GDS* onto the *QDS* space.

The structure of the Q^2 representation is shown in Figure 2. The bottom part of this figure represents the quantitative dynamic structure. *P* is the input process, $P: T \rightarrow X$, where *T* is time and *X* - input. *Q* represents the state of the system. The state of the system changes according to the state transition function $f: Q \times X \times T \rightarrow Q$. The output of the system is represented by *W*; it varies according to the output (readout) function $g: Q \rightarrow W$. The upper part of the figure represents the qualitative counterpart; it is implemented as a (non-deterministic) finite state automaton. Particular symbols of Figure 2 have the following meaning: Θ - states of the automaton, Λ - qualitative input events, Ω - qualitative outputs, ϕ - qualitative state transition function, $\phi: \Theta \times \Lambda \rightarrow \Theta$, and γ - qualitative output function, $\gamma: \Theta \rightarrow \Omega$. The two representational substructures are connected through *qualitative abstraction functions* χ_P, χ_Q, χ_W . The abstraction function χ_W translates the quantitative outputs of the system into qualitative outputs; χ_Q translates quantitative states into qualitative states, and χ_P translates the Cartesian product of quantitative input, state, and time into qualitative input events. Qualitative outputs represent intervals on the output variable of the dynamic system. Qualitative states represent regions in the state space of the *GDS*. Qualitative input events are regions in the Cartesian product of quantitative input, state and time.

These regions are delimited by *critical hypersurfaces*.

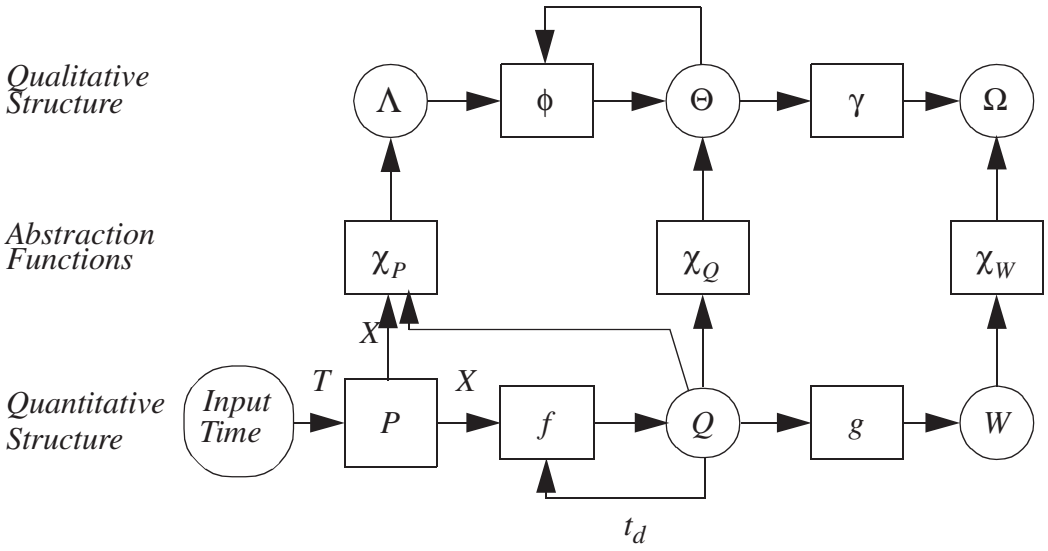

Figure 2: Quantitative/Qualitative (Q^2) Dynamic System Model

The main idea behind the Q^2 representation is that reasoning can be done in the *QDS* structure and the reasoning is consistent, or in other words, the results of the reasoning in the *QDS* hold in the *GDS*. Mathematically this is expressed as:

$$GDS \propto QDS$$

which reads: "*GDS* is a model for *QDS*".

Examples of reasoning using the *QDS* representation are: (1) determine the qualitative output (interval on the quantitative output), given the current qualitative state (region in the quantitative state space); (2) determine the next qualitative state, given the current qualitative state and qualitative input event (region in $Q \times X \times T$); (3) determine the qualitative input event that would cause the system to change from one state to another.

In our conceptualization of the network as a dynamic system, we selected the load as the state variable $Q=(Q_1, ... Q_r)$, and the packet rate as the input variable $X=(X_1, ..., X_n)$. The output W is defined as a variable that is derived from a set of state variables by taking the maximum value over the state variables. The function f is the *local* state transition function that uses the process input and current state to predict its next quantitative state. After a fixed time delay t_d, the new state becomes the current state and the prediction process is repeated. The function g is the output function that associates an output with each state.

In [11] this function is derived analytically from the (known) model of the dynamic system. Unfortunately, in our application we do not have knowledge of the quantitative model of the dynamic system and therefore we derive the abstraction function from the historical network data. Consequently, in this research we are not using the full version of the method presented in [11], but an approximation of this method in which the critical hypersurfaces are learned rather than derived from the underlying quantitative model. We used the Centroid Clustering Method [15] of the SAS modeling tool and found that it provides a better result with this type of network data than the Complete, Ward, Density, and Average methods [15]. The clustering is done by the standard k-means clustering algorithm [16]. The algorithm first randomly selects k items to represent k clusters. Then centroids are computed for each cluster and each item is assigned to one of the clusters based upon the geometric distance from the centroid. The process of computing centroids and cluster assignment is repeated until no re-assignments are made in one pass.

.

According to Q^2 methodology, first the quantitative output is translated to qualitative output structure by clustering the output into several output intervals (see Figure 3). The second step is to partition the state space into clusters so that each of the clusters is mapped through the output function g into only one cluster in the output space. Usually, there will be more state clusters than output clusters. The third step repeats the second step to partition the input event space, i.e., the Cartesian product of (previous) state, time, and input. The arrows in Figure 3 indicate that several input event space regions can be mapped to one state region and several state regions can be mapped into one output region. Finally, after all of the above steps are complete, the input and state space partitions can be used to build a Non-Deterministic Finite State Automaton. An example of this is given in Section 4.

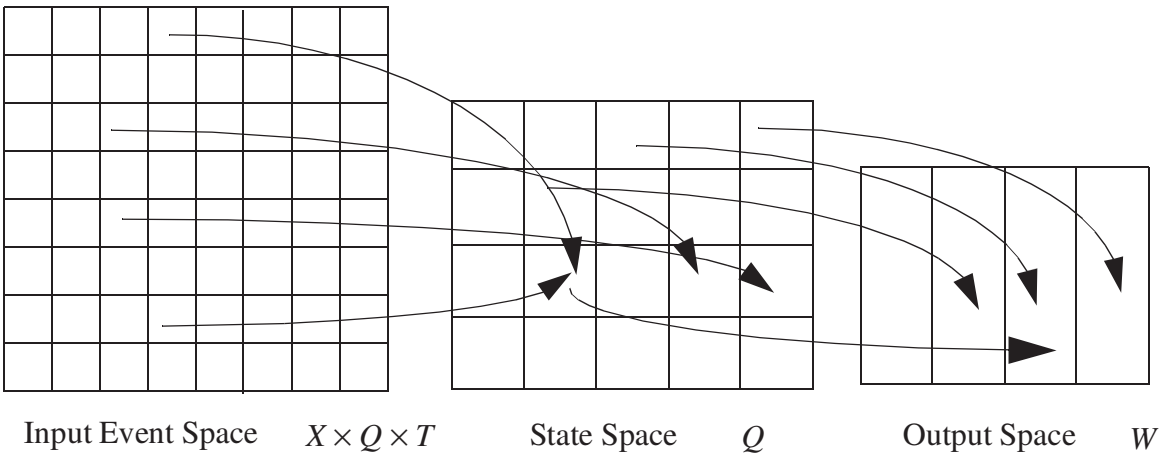

Figure 3: Input, State, and Output Space Partitions

4. STEPPING THROUGH THE Q^2 METHOD

Step 1: Information Gathering and Search for Appropriate Attributes

To fully capture network behavior, we must first identify relevant network parameters. Specifically, we have to determine which attributes can provide concrete information about network behavior. There are hundreds of attributes, of which only a handful are relevant to network performance. After interviewing many network managers [17], we concluded that *Load, Packet Rate*, and *Packet Breakdown* are the attributes that are most informative for understanding of network behavior. These derived attributes depend on many other *MIB* variables and need to be computed from the *MIB*. *Packet Rate* represents the number of packets arriving within the time interval; it was used as the input to the system. *Load* represents the percentage of utilization of the network bandwidth; it was used as the state variable. The network managers chose these attributes because they most directly identify bottlenecks and best capture the dynamics of network behavior.

Step 2: Data Collection and Formatting

Network performance data was collected from the Cabletron Corporate Network represented in Figure 1 using the Nashua router (the left rectangle). A data collection tool was built on top of the

Spectrum network management platform. The tool collected the quantitative state variables *(Load)* and the quantitative input event variables *(Packet Rate)* from all the ports on the router except the external ports. The data were stored in separate files for each experiment. Each set of data was formatted in a table (we call it *Data-Table*) as shown below:

Day Time Pkts -0 Pkts -1 ... Pkts -n Load-0 Load-1 ... Load-n Load-Max Port-Max

The first column is the day of the week, represented by numbers one through seven, where one represents Sunday. The second column is the time, each entry representing a 10-minute interval. Since experiments cover periods of twenty four hours, the time is represented in numbers ranging from 1 to 144, where 1 stands for 12:10 am, 2 stands for 12:20 am, and so on. The port *Packet Rate* is represented in columns 3 through n+2, where n is the number of subnets on the router (see Figure 1). This is followed by columns representing the load for particular nodes *(Load-i)*. The second to the last column is the *Load-Max*, referred to as the output column. The output column is derived from all subnet ports on the router, defined as the maximum value of all ports at that particular time. The last column represents the port where *Load-Max* occurs for that particular time. Only the maximum load is being considered as the output column in this project because the *Port-Max* column in both load and packet rate ports are usually the same. In the Q^2 structure, *Load-Max* is considered as the system's quantitative output (W = *Load-Max).*

Step 3: Data Reduction

Some of the ports appear in the output column very infrequently, meaning that, in a long run, they do not have a significant impact on network behavior. To reduce our search for the relevant network attributes we heuristically excluded from further analysis all those ports that appear less than five percent of the time in the output column. We denote the remaining state variables after data reduction as C and the remaining input variables after data reduction as D. Columns of C represent state variables: C = *(Load-0, ... , Load-s),* and columns of D represent the corresponding input variables: D = *(Pkts-0, ... , Pkts-s).* The data table *Data-Table* is reduced appropriately to include only those columns represented by C and D.

Step 4: Determining Qualitative Output

Clustering of *Load-Max* groups the output values into subsets that will be treated as qualitative outputs $\Omega = (\Omega_1, \Omega_2, \ldots, \Omega_n)$. Each cluster Ω_i is represented by a triple (minimal value, max-

imal value, and mean value). Geometrically, since *Load-Max* is one-dimensional, each Ω_i is an interval on *Load-Max*. After experimenting with the data we concluded that we need only from five to seven clusters to achieve best results. After clustering, *Data-Table* is appended by one more column representing the cluster number (qualitative output) associated with each table row.

Step 5: Determining State Variables

In this step, we select a subset of state variables from among all the variables *C*. We consider all subsets of these variables and select the subset that is best at mapping the resulting clusters, which are potential candidates for qualitative states, into qualitative outputs. Note that this mapping is through the output function *g*. We don't know the function *g*, but for the data points we have collected we know the values of the output function for each state that we measured. We are interested in mapping each of the qualitative states into only one qualitative output. To accomplish this, we formulate a clustering criterion that counts the percentage of times the quantitative states within any cluster C_i are successfully mapped into a single qualitative output Ω_i. The criterion that we minimize is:

$$\frac{1}{n}\sum_{l=1}^{n}\frac{N_l - N_{li}}{N_l}$$

where l - index over state clusters, n - number of state clusters. N_l - number of items in cluster l, and N_{li} - maximum number of items in cluster l mapped through the output function g into one qualitative output Ω_i. The subset of variables selected through this procedure constitutes the state variables *C = (Load-0, ... , Load-s)* that we will use to characterize the network.

Step 6: Determining Qualitative States

The clusters of states generated by the previous step are potential candidates for qualitative states. However, we also want to minimize the number of clusters with small numbers of elements. For this reason, we apply the following postprocessing heuristic: *the number of clusters with less than five elements should be less than five.* In order to satisfy this requirement we first perform clustering with a high limit on the number of clusters (number of clusters is a parameter in the SAS clustering algorithm), and then, if the resulting number of clusters with less than five elements is five

or more, we reduce the maximum number of clusters by one and repeat the clustering procedure again. This procedure is repeated until the heuristic criterion is satisfied. The final clustering gives the qualitative states $\Theta = (\Theta_1 \ldots \Theta_s)$. Each qualitative state Θi is represented by *m* triples, i.e., the minimal value, the maximal value, and the mean value for each of the state variables Q_1, ..., Q_m. Geometrically, since the state vector *Load*, represented by *C*, is *s*-dimensional, each Θi is an *s*-dimensional hypercube. *Data-Table* is appended by the column representing qualitative states so that each state in the table is classified into its qualitative counterpart.

Step 7: Determining Qualitative Input Events

We select as the input variables the packet rates for each port corresponding to *C*. We denote them as *D = (Pkts-0, ... , Pkts-s)*. The space $Q \times X \times T$ includes also the state variables *C*, and time *T*. Therefore, the qualitative input event variables are represented as: *E = {C; D; T}*. To determine the qualitative input events, we cluster the qualitative input event data in a similar way as described in Steps 5 and 6 along all the variables in *E*. For each "new" qualitative state Θ_i, we cluster those combinations of input, time and "initial" state that lead to this new state Θ_i. The same type of the quality criterion in Step 5 is used in this clustering, but this time we are interested in clustering such that each qualitative input (cluster) is mapped through the state transition function *f* into one "new" qualitative state. This step is followed by the same heuristic procedure for minimizing the number of clusters with less than five elements as described in Step 6. As a result of this step, we obtain qualitative input events $\Lambda = (\Lambda_1 \ldots \Lambda_e)$, where each qualitative input event Λ_i is an *m+e+1*-dimensional hypercube represented by *m+e+1* triples (minimum, mean and maximal values). Each row of *Data-Table* is annotated with an appropriate qualitative input event from Λ.

Step 8: Building a State Transition Table

Since *Data-Table* is ordered by the time of the event, pairs of consecutive rows in this table represent state transitions. Some of these transitions move the system into a new qualitative state and some others leave the system in the same qualitative state. The qualitative input event that causes the transition is the event associated with the first element of the pair. In one set of data (one data table) multiple occurrences of the same qualitative state and of the same qualitative input event exist. The system is not deterministic, which means that for the same initial state and for the same

qualitative input event, various transitions take place. To find the transition probability p_{ijk} of moving from the initial state Θ_i to the new state Θ_j, given the event Λ_k occurs, we count how often this transition took place for that state and that qualitative input. The probabilities for a given initial state and a given input add up to one:

$$\sum_j p_{ijk} = 1.$$

The resulting state transition table and the associated non-deterministic automaton may look as shown in Table 1 and Figure 4 respectively. Note that two different qualitative inputs, Λ_2 and Λ_3, can lead from state Θ_1 to state Θ_2. Also, the same qualitative input Λ_3 can lead from state Θ_1 to state Θ_2 with probability 0.3, and to state Θ_4 with probability 0.7.

Table 1: State Transition Table

.

Initial State	Input	New State	Probability
Θ_1	Λ_1	Θ_1	1.0
Θ_1	Λ_2	Θ_2	1.0
Θ_1	Λ_3	Θ_2	0.3
Θ_1	Λ_3	Θ_4	0.7
Θ_2	Λ_3	Θ_3	1.0
Θ_3	Λ_1	Θ_3	0.2
Θ_3	Λ_1	Θ_1	0.8
Θ_4	Λ_2	Θ_3	1.0

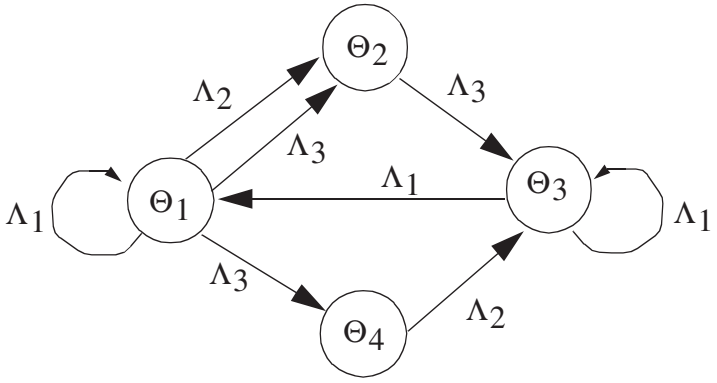

Figure 4: Example of a Non-Deterministic Finite State Automaton

5. PREDICTING NEW STATE

Two possible approaches for predicting new states using the non-deterministic finite state automaton are *random* and *greedy*. In the random approach, for a given qualitative input, all possible transitions are considered; they are selected randomly with the frequency determined by the probability of a given transition. In the greedy approach, only the transitions with the higher probability for a given input are considered. There are also many strategies for considering more than one, but less than all transitions. In our experiments, we considered the strategies listed below.

Greedy:	Only the transition with the highest probability for a given input.
Rand_Top2:	Only two highest-probability transitions.
Rand_Top3:	Three highest-probability transitions.
Rand_Top4:	Four highest-probability transitions.
Rand_>5%:	All transitions whose probability is greater than 0.05.
Rand_>10%:	All transitions whose probability is greater than 0.1.
Rand_>15%:	All transitions whose probability is greater than 0.15.
Rand_All:	All transitions (random method).

6. EXPERIMENTAL RESULTS

We tried to answer the following questions: (1) Which of the new state prediction methods gives better results? (2) How much data do we need to train the system? (3) Can we derive only one global qualitative model to predict network behavior, or do we need a number of local models?

In each of the experiments described below we had two sets of data: the training set and the test set. The training set was different from one experiment to another, but the test set was the same for all the experiments. In other words, we used different sets or amounts of data to derive the qualitative model, but we used the same set of data to evaluate the quality of the model. This allowed us to compare different models and different prediction methods.

6.1 Comparison of Prediction Methods

In the first set of experiments we compared all the prediction methods listed in Section 5 using two models, one derived from a two-week set of data and the other one derived from a five-week set of data. The average predictions for these two models and for all the methods are shown in Figure 5. As we can see, the random method gives the worst prediction accuracy. The other methods predict very closely, with the greedy method scoring slightly above the others. The best performance of the greedy method is in accordance with the expectation (c.f. [18]). This method is the optimal method of prediction. We can also see that for all the methods the quality of prediction was better for the model that was derived from the five-week data set.

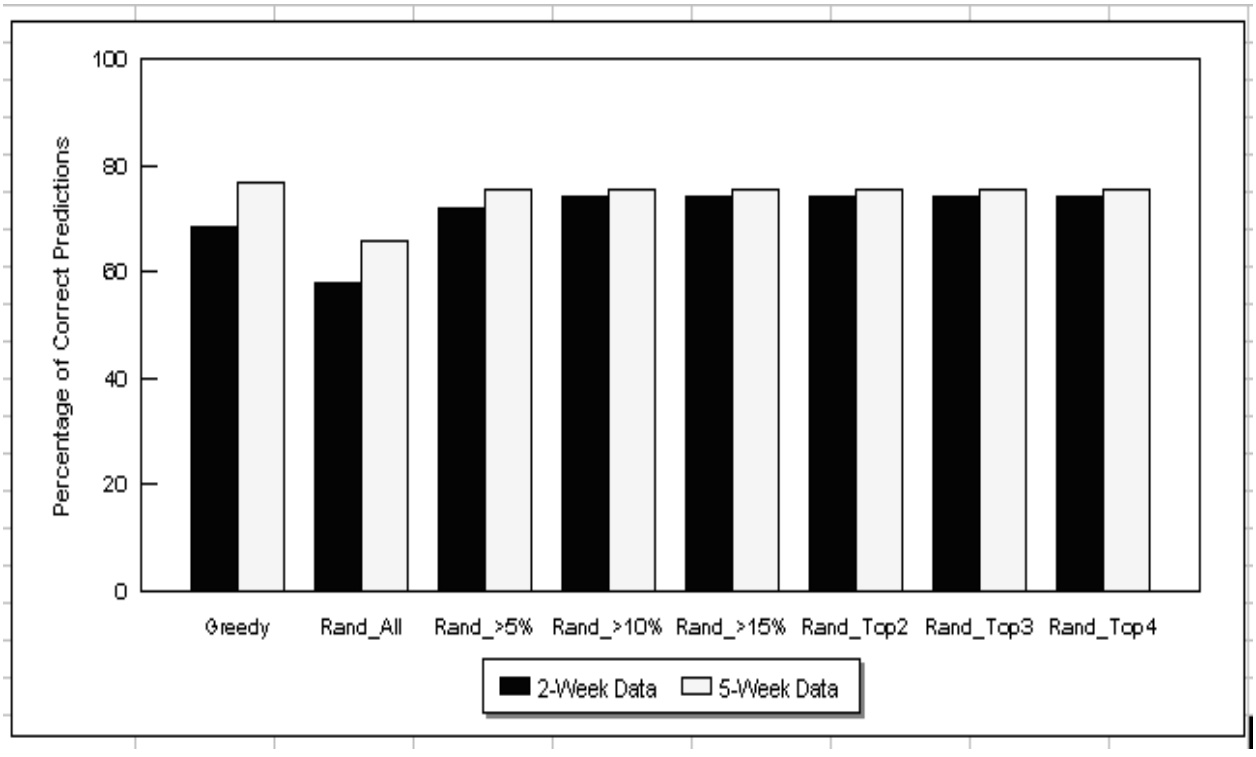

Figure 5: Comparison of Prediction Methods

6.2 Prediction Accuracy for Days and Weeks

In the second set of experiments we used only the greedy and the random methods of prediction. We also used only the five-week data for deriving one qualitative model of network behavior. We estimated the precision of prediction for each day separately. The result of this experiment is presented in Figure 6. Then we calculated the average accuracy of prediction for each day of the week. These results are shown in Figure 7.

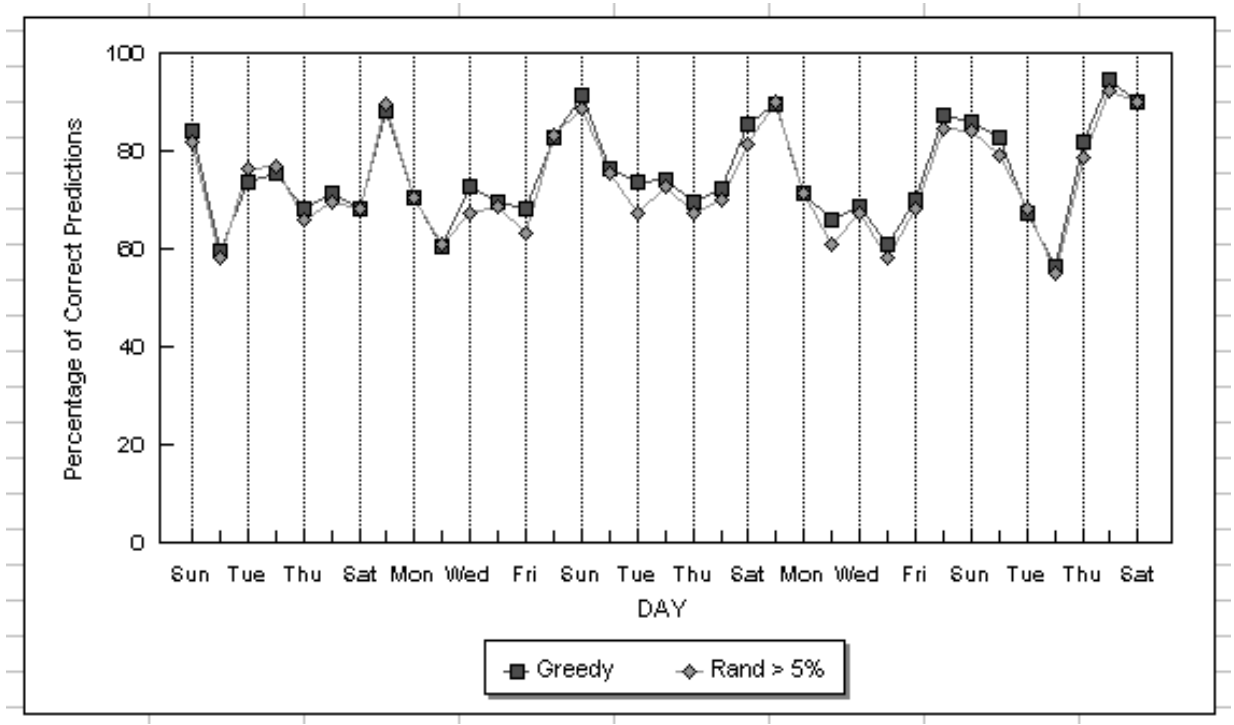

Figure 6: Daily Prediction

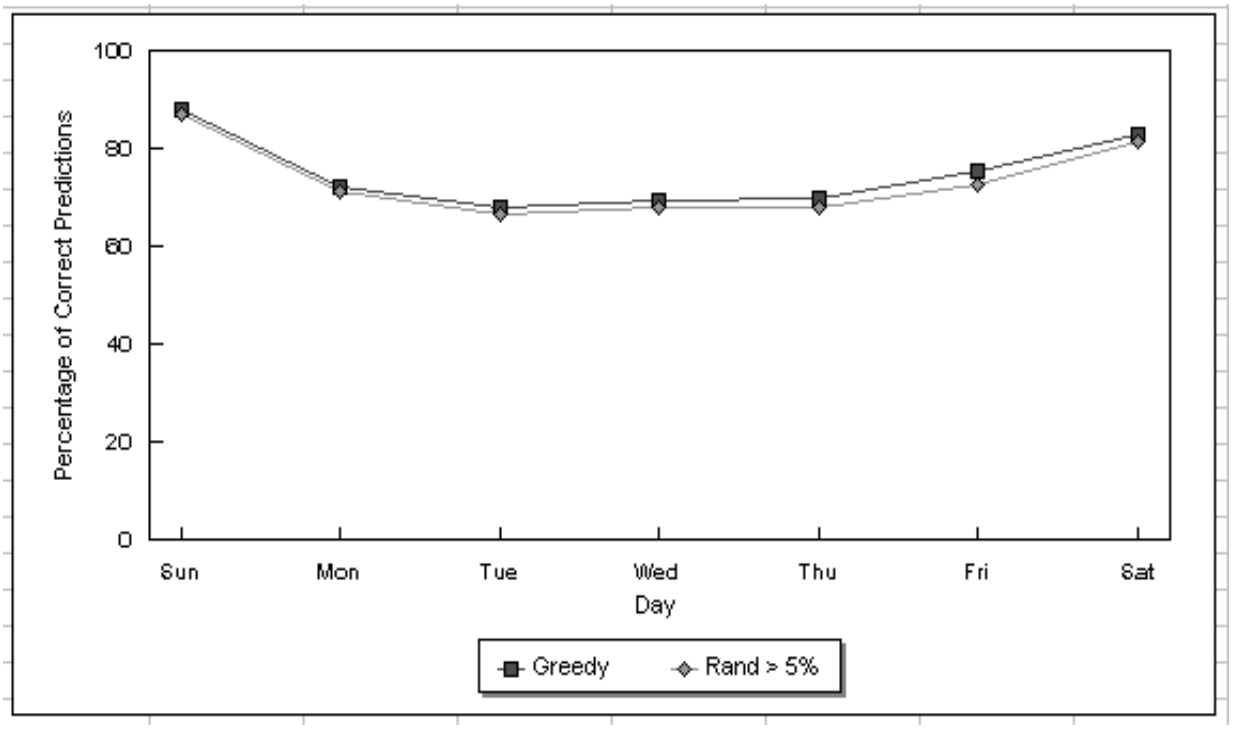

Figure 7: Average Daily Prediction

As we can see in Figure 6, there is a pattern in the prediction accuracy: the accuracy of prediction is higher for weekends than for weekdays. This is even more clear in Figure 7. One explanation for this is that because of more variability in the network activity during the weekdays, it is more difficult to have an accurate prediction. This may suggest that we should derive different models for the weekdays and for the weekends. In our experiments we did not observe any significant variation of the prediction accuracy from week to week.

6.3 The Day-of-the-Week Models

In this experiment we derived separate models to capture the behavior of each day of the week. Eighteen weeks worth of data was used to derive this model. We adopted this daily model when we noticed the variability of prediction accuracy for different days (weekdays vs. weekends), and that the five-week data gave better results than the two-week data. Figure 8 shows the results of (greedy) daily prediction. Figure 9 shows the average prediction accuracy for each day of the week.

The results of this experiment were not exactly what we expected. We expected to achieve better prediction accuracy by using more data to train the network model. It appears that more data is not necessarily better. Our conclusion was that the network model changes over time and by using more data we derive a model which is better on the average, but not necessarily better for a specific period of time. To model the model change over time, a number of models that better capture time-local behaviors might be better suited than one model for each day of the week.

Figure 9 shows a different result in average daily prediction than a five-week result (Figure 6). When using eighteen weeks of data we have a better prediction for Sunday through Thursday. With five weeks of data, we have a better prediction for weekends and a less accurate prediction on weekdays. It is hypothesized that the model derived from the five-week data captured the behavior of the network better during weekends than during weekdays, while the eighteen-week model was better for other days. This indicates that network behavior model changes over time. We were not able to conclude that this change is periodic. The weekly prediction accuracy was almost the same for each of the eighteen weeks.

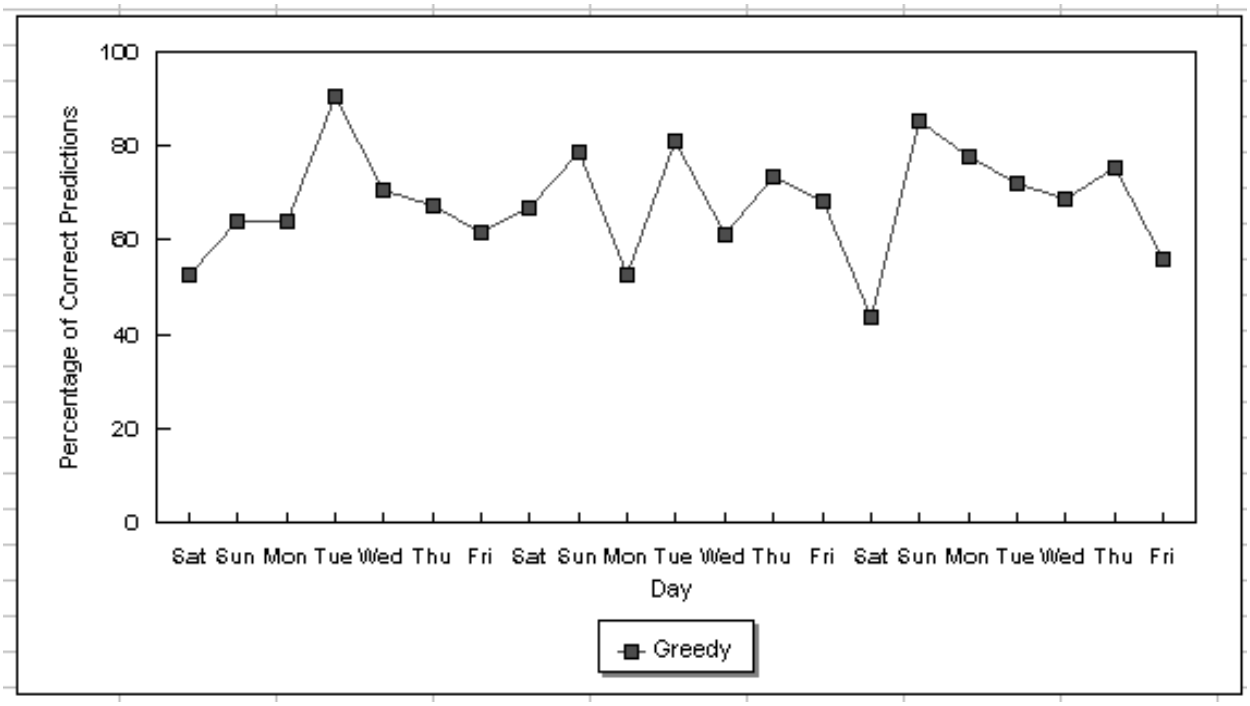

Figure 8: Daily Prediction (18 Week Data)

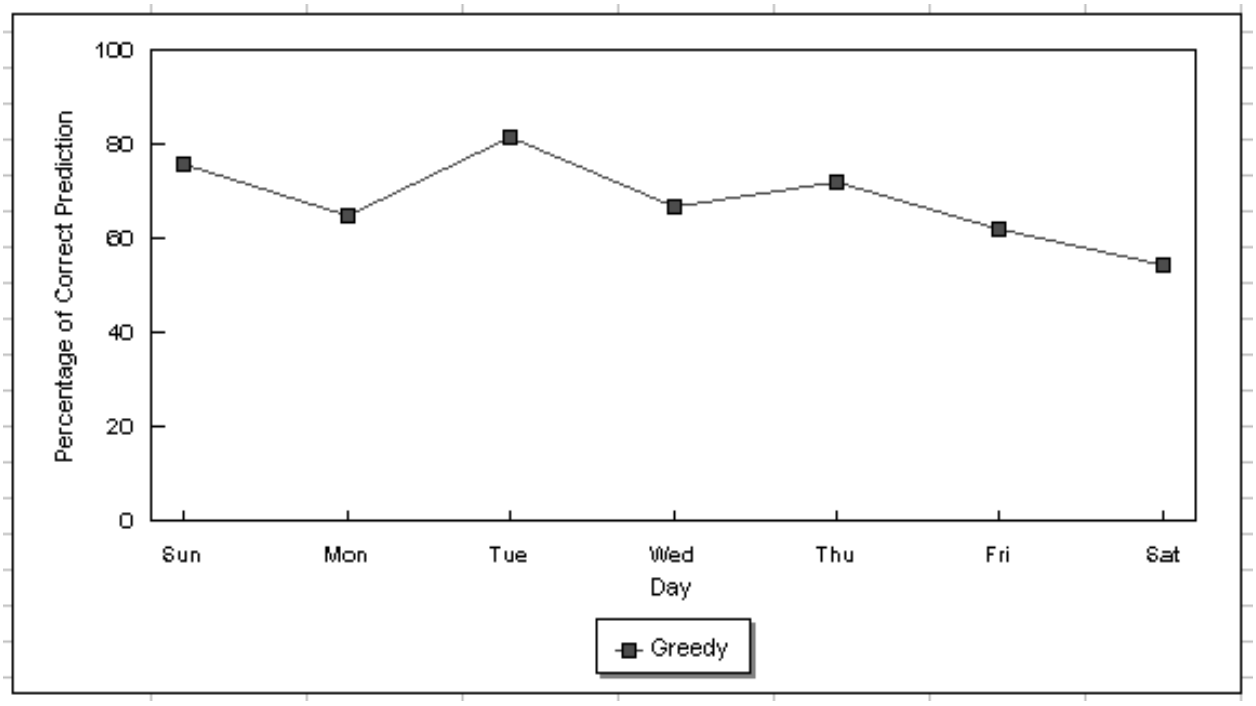

Figure 9: Average Daily Prediction (18 Week Data)

7. CONCLUSIONS AND FUTURE DIRECTIONS

This paper describes the Q^2 computational method for capturing qualitative network models and for predicting future network behavior. A manager can utilize the Q^2 method to detect trouble spots in the network before they become problems. The *(Q^2)* [11] approach we utilized uses archive network data to derive a qualitative model of network behavior, called a Qualitative Dynamic System *(QDS). QDS* consists of a finite state non-deterministic automaton and an abstraction function that maps regions in the underlying dynamic system (modeled as *GDS)* to states, outputs and input events in the automaton. The central idea of the Q^2 approach is partitioning of the *GDS* spaces in such a way as to guarantee consistency of reasoning with the combined *GDS/QDS* structure. Predicting network behavior consists of collecting current network data, mapping the data through the abstraction function to the *QDS*, and then using the automaton to determine the network next qualitative state.

To assess the validity of the proposed approach to predicting of network behavior we tested the method on three sets of data. We also evaluated random and greedy methods of predicting new states using the non-deterministic finite state automaton. The greedy method gave slightly better accuracy, but the generation of the automaton was a difficult process, since the automaton had to be checked for consistency; i.e., that there exists only one transition from each state and that parts of the automaton are not disconnected. Also, considering only one (most probable) transition, and forgetting all others, limits the adaptability of the automaton should a need for such a process arise. The random method is less vulnerable to this kind of problem.

Overall, the method as described in this paper, gave the expected prediction accuracy of 82% for our experimental scenario. In this calculation we used the probabilities of particular qualitative input events, the probabilities of particular state transitions, and the probabilities of qualitative outputs. The mean value of the prediction accuracy was 70%. This result is intuitive since the automaton was better tuned to predict those points that are more probable in the data. We believe it is a relatively good result for our initial experiments, and more importantly, it justifies expenditures for further research.

In our experiments we considered using a different model for each day of the week, rather than one global model. But this hypothesis was rejected since this approach did not lead to better results. The final conclusion was that we need to maintain multiple models, monitor the behavior of the network for compliance with the current model and select a best-match model when the current model does not provide accurate behavior prediction.

In order to simplify the problem of capturing and predicting network behavior, several assumptions were made: that the archive data collected from the network over several weeks is sufficient to capture its behavior, that we can find patterns in these data by developing a method that completely analyzes all the data, that learning these patterns and determining their transition points from pattern to pattern is sufficient for predicting network behavior, and that future behavior will resemble past behavior. Also, to simplify the experiments, we selected a scenario in which a single router was able to monitor the whole network. Clearly, not all networks will be amenable to such tests if such an omniscient router is not in place.

Our application of Q^2 methodology makes two simplifying assumptions. First, we did not preserve the time history of events. By ignoring time history we greatly reduce the amount of data collected but we sacrifice the ability to predict more than one state ahead. We did not want to make this complex step before obtaining initial results that would justify further study. Second, we used clustering instead of hypersufraces for partitioning the *GDS* spaces. This was because, unlike the example described in [11], we did not know the analytical form of the network. The two ways of dealing with the lack of an analytical model are either to do system identification first, and then derivation of the hypersurfaces, or identification of the hypersurfaces directly. Other possibilities include the use of machine learning clustering algorithms and neural networks [23] in conjunction with the Q^2 method described in this paper.

Acknowledgments

Cabletron Systems, Inc. sponsored and funded Sule Ibraheem in this project. The authors thank Bill Tracy and Jeff Ghannam for providing the equipment necessary to complete this research; John Smart, Dave Dickson, and Tom Donovan for sharing fundamental experience and knowledge about the functionality of network managers; Sue Avery, Carol Steele, John Bartlett and Mary Helander for reviewing the manuscript; Utpal Datta for monitoring the progress of the thesis; and Stephen Linder for valuable comments on this manuscript.We also thank the anonymous reviewers for their constructive comments on our paper.

8. Bibliography

[1] B. Alderson, Network Analysis: 10 Steps to Fine-Tuning Network Performance*, Network Computing,* pp. 90-94, May 1991.

[2] W. Stallings, *SNMP SNMP V2 and CMIP: The practical guide to network management standard,* Adison Wesley, 1993.

[3] J. Filipiak, Real Time Network Management, North-Holland, 1991.

[4] A. Law and M. McComas, Simulation Software for Communications Networks: The State of the Art, IEEE Communications Magazine, Vol.32, No.3, 1994.

[5] U. Herzog, Network Planning and Performance Engineering, in Networks and Distributed Systems Management, M.Sloman (Ed.), Addison-Wesley, 1994.

[6] R. Cahn and H. Liu, Network Management and Network Design, in Network Management and Control (Volume 2), I.Frish, M.Malek and S.Panwar (Eds.), Plenum Press, 1994.

[7] L. Lewis, How to Integrate Cabletron's SPECTRUM and CACI's COMNET, Technical Note lml-95-03, Cabletron Systems, Rochester, NH, 1995.

[8] L. Lewis and G. Dreo, Extending Trouble Ticket Systems to Fault Diagnostics, *IEEE Network Magazine*, pp. 44-51, November, 1993.

[9] *Request for Comments: 1213 (MIB II)*, Network Working Group, K. McCloghrie, Huges LAN Systems, Inc. and M. Rose, Performance System International, (eds.), March 1991.

[10] *Request for Comments (Draft)*, Cisco Systems, Inc., 1525 O'Brien, Menlo Park, CA 94025.

[11] M. M. Kokar, On Consistent Symbolic Representations of General Dynamic Systems, *IEEE Transactions on Systems, Man and Cybernetics,* Vol.25, No.8, pp. 1231-1242, 1995.

[12] S.O. Ibraheem, *A Method for Capturing Qualitative Network Performance and Predicting Behavior*, M.S. Thesis, Computer Systems Engineering, Northeastern University, Boston, MA, June 1995.

[13] M.D. Mesarovic and Y. Takahara, *Abstract Systems Theory,* Springer Verlag, 1989.

[14] G. Klir, *Architecture of Systems Problem Solving,* Plenum Press, 1985.
[15] *SAS/STAT User's Guide (The Cluster Procedure)*, Vol. 1, Version 6, Fourth Edition, pp. 520, 1990.
[16] M. R. Andberg, Cluster Analysis for Applications, Academic Press, New York, 1973.
[17] J. Smart, T. Donovan and D. Dickson, Personal Communications, Cabletron System Network Managers, 1993.
[18] A. G. Barto and S. J. Bradtke, Learning to Act using Real-Time Dynamic Programming, *CMPSCI Technical Report 93-02*, March 1993.
[19] R. T. Braden, *A Pseudo-Machine for Packet Monitoring and Statistics*, USC Information Science Institute, Marina del Rey, California, Association for Computing Machinery, Inc. pp. 199-208, 1988.
[20] P. D. Amer and L. N. Cassel, Management of Sampled Real-Time Network Measurements, *Proceedings 14th Conference on Local Computer Networks*, pp. 62-68, 1989.
[21] H. J. Fowler and W. E. Leland, *L*ocal Area Network Traffic Characteristics, with Implications for Broadband Network Congestion Management, *IEEE Journal on Selected Areas in Communications,* Vol. 9. No. 7. pp. 1139-1148, 1991.
[22] B. L. Histon, Knowledge-Based Monitoring and Control: An approach to Understanding the Behavior of TCP/IP Network Protocols, Association for Computing Machinery, Inc. pp. 170-181.
[23] L. Lewis, Doing Network Capacity Evaluation/Planning with Neural Network Clustering Algorithms, Technical Note lml-96-01, Cabletron Systems, Rochester, NH, 1996.