Network Management of Predictive Mobile Networks

Stephen F. Bush, Victor S. Frost, Joseph B. Evans

February 27, 1997

Abstract

There is a trend toward the use of predictive systems in communications networks. At the systems and network management level predictive capabilities are focused on anticipating network faults and performance degradations. Simultaneously, mobile communication networks are being developed with predictive location and tracking mechanisms. The interactions and synergies between these systems presents a new set of problems. A new predictive network management framework is developed and examined. The interaction between a predictive mobile network and the proposed network management system is discussed. The Rapidly Deployable Radio Network (RDRN) is used as a particular example to illustrate these interactions.[1]

Keywords: *Prediction Mobility Management*

1 Introduction

Recently proposed mobile networking architectures and protocols involve predictive mobility management schemes. For example, an optimization to a Mobile IP-like protocol using IP-Multicast is described in [1]. Hand-offs are anticipated and data is multicast to nodes within the neighborhood of the predicted handoff. These nodes intelligently buffer the data so that no matter where the mobile host (MH) re-associates after a handoff no data will be lost. Another example [2] [3] proposes deploying mobile floating agents which decouple services and resources from the underlying network. These agents would be pre-assigned and pre-connected to predicted user locations. Finally, this paper will focus on the Rapidly Deployable Radio Networks Project [4] [5] as an example of a predictive mobile network. The Virtual Network Configuration (VNC) algorithm developed as part of RDRN uses the predictive mechanism for every phase of configuration, including location and handoff.

[1]This paper is partially funded by ARPA contract number J-FBI-94-223 and Sprint under contract CK5007715.

Progress is being made in research involving predictive system and network management [6]. This paper develops a variation of the Virtual Network Configuration Algorithm as proposed for the RDRN [5] for a predictive network management system. The predictive capability of such a system can be used to help optimize its own operation by controlling the management of the polling rate. Both VNC and a modification of VNC which results in the predictive management algorithm developed in this paper are modifications of the Time Warp Algorithm [7]. Finally a discussion of how predictive mobile networks and predictive network management systems should interact is presented.

2 Introduction to a Predictive Network Management System

Systems management means the management of heterogeneous subsystems of network devices, processing platforms, distributed applications, and other components found in communications and computing environments. Current system management relies on presenting a model to the user of the managed system which should accurately reflect the current state of the system and should ideally be capable of predicting the future health of the system. System management relies on a combination of asynchronously generated alerts and polling to determine the health of a system [8].

The management application presents state information such as link state, buffer fill and packet loss to the user in the form of a model [9]. The model can be as simple as a passive display of nodes on a screen or a more active model which allows displayed nodes to change color based on state changes, or react to user input by allowing the user to manipulate the nodes which causes values to be set on the managed entity. This model can be made even more active by enhancing it with predictive capability. This enables the management system to manage itself, for example, to optimize its polling rate. The two major management protocols, SNMP [10] and CMIP [11], allow the management station to poll a managed entity to determine its state. In order to accomplish real-time and predictive network management in an efficient manner, the model should be updated with real-time state information when it becomes available, while other parts of the model work ahead in time. Those objects working ahead of real-time can predict future operation so that system management parameters such as polling times and thresholds can be dynamically adjusted and problems can be anticipated. The model will not deviate too far from reality because those processes which are found to deviate beyond a certain threshold will be rolled back, as explained in detail later. The process's messages must obey the rules for consistency in [12]:

Rule 1 *If two events are scheduled for the same process, then the event with the smaller timestamp must be executed before the one with the larger timestamp.*

Rule 2 *If an event executed at a process results in the scheduling of another event at a different process, then the former must be executed before the latter.*

In order to determine the characteristics and performance of this predictive network management algorithm, we will review the research on performance and modeling of other lookahead algorithms and Time Warp in particular. In [13] a comparison of the conservative Chandy-Misra approach and the optimistic Time Warp is presented. This is done using a typical queuing theory approach which assumes exponential service times. There have been several other detailed comparisons between conservative and optimistic methods of simulation. These studies also make simplifying assumptions. In [14], it is shown that in a feed forward network, the time of execution of a message will occur earlier in Virtual Time than its corresponding message in the synchronous parallel algorithm described in [12]. In [15], it is shown that Time Warp can out-perform the conservative technique known as Chandy-Misra by a factor of P, P being the number of processors, but that no such model in which Chandy-Misra out-performs Time Warp by a factor the number of processors used exists. Past work has examined the performance of Time Warp by comparing it to conservative mechanisms [14] or simulating the Time Warp mechanism itself [16]. In this paper the focus is not only on analyzing and optimizing speed of execution but also using the algorithm to maintain network management prediction accuracy.

One goal of this research is to minimize polling overhead in the management of large systems [17]. Instead of basing the polling rate on the characteristics of the data itself, the entity is emulated some time into the future in order to determine the characteristics of the data to be polled. Polling is still required with this predictive network management system in order to verify the accuracy of the emulation.

3 Predictive Standards-Based Network Management Information

Management information from standards-based managed entities must be mapped into this predictive network management system. Network management systems rely upon standard mechanisms to obtain the state of their managed entities in near real-time. These mechanisms, SNMP [10] and CMIP [11] for example, use both solicited and unsolicited methods. The unsolicited method uses messages sent from a managed entity to the manager. These unsolicited messages are called traps or notifications; the former are not acknowledged while the latter are acknowledged. These messages are very similar to messages used in distributed simulation algorithms; they contain a timestamp and a value, they are sent to a particular destination, i.e. a management entity, and they are the result of an event which has occurred.

Information requested by the management system from a particular managed entity is solicited information. It also corresponds to messages in distributed simulation. It provides a time and a value; however, not all such messages are equivalent to messages in distributed simulation and required in a predictive management system. These messages provide the management station with the current state of the managed entity, even though no change of state may have occurred or multiple state changes may have occurred. The design of a management system which requests information on the state of its managed entities at the optimum time has always been a problem in network management. If requested too frequently, bandwidth is wasted, if not requested frequently enough, critical state change information will be missed.

We will assume for simplicity that each managed entity is represented in the predictive management system by a Logical Process (LP). It would greatly facilitate system management if vendors provide not only the standards based SNMP Management Information Base (MIB) as they do now, but also a standard simulation code which models the entity or application behavior and can be plugged into the management system just as in the case with a MIB. Vendors should have models of their devices readily available from product development.

4 Introduction to the Predictive Network Management System Algorithm

Terminology borrowed from previous distributed simulation algorithms has a slightly different meaning in this predictive network management system. In addition, new terminology must be introduced. Thus it is important that the terminology be precisely defined.

The predictive network management system algorithm encapsulates **Physical Processes** (PP) simulating managed network devices within **Logical Processes** (LP). A PP is nothing more than the executing process defined by the program code. An LP consists of the PP and additional data structures and instructions to maintain message order and correct operation as the system executes ahead of real time. An LP contains a **Receive Queue** (QR), **Send Queue** (QS), and **State Queue** (SQ). The QR maintains newly arriving messages in order by their Receive Time (TR). The QS maintains copies of previously sent messages in order of their send times. The state of the LP is periodically saved in the SQ. The LP also contains its notion of time known as **Local Virtual Time** (LVT) and a **Tolerance** (Θ) which is the allowable deviation between actual and predicted values of incoming messages. Also, the **Current State** (CS) of a LP will be the current state of the LP and its encapsulated PP.

The predictive network management system contains a notion of the complete system time known as **Global Virtual Time** (GVT) and a sliding window known as the **Lookahead** time (Λ).

Messages contain the **Send Time** (TS), **Receive Time** (TR), **Anti-toggle** (A) and the actual message itself (M). The TR is the time this message should be received by the destination LP. The TS is the time this message was sent by the originating LP. The A field is the anti-toggle and is used for creating an anti-message to remove the effect of false messages as described later. A message will also contain a field for the current **Real Time** (RT). This is used to differentiate a real message from a virtual message.

A **driving process** is required to predict future events and inject them into the system. For example, in a mobile system such as the Rapidly Deployable Radio Network (RDRN) [5], the Global Positioning System (GPS) is used to provide each node with its current position. The GPS receiver process may run in real-time and inject future predicted location messages as well. In the predictive network management system, the driving process may be the number of expected users and their estimated bandwidth usage. The driving process(es) originate virtual messages via internal prediction. The remaining PPs react to these messages as though they are real messages. A message which is generated and time-stamped with the current time will be called a **real message**. Messages which contain future event information and are time-stamped with a time greater than current time are called **virtual messages**. If a message arrives at a LP out of order or with invalid information, it is called a **false message**. A false message will cause an LP to rollback.

A **rollback** is a mechanism by which a LP returns to a known correct state. The rollback occurs in three phases. In the first phase, the LP state is restored to a time strictly earlier than the time stamp of the false message. In the second phase, anti-messages are sent to cancel the effects of any invalid messages which had been generated before the arrival of the false message. An **anti-message** contains exactly the same contents as the original message with the exception of an anti-toggle bit which is now set. When the anti-message and original message meet, they are both annihilated. The final phase consists of executing the LP forward in time from its rollback state to the time the false message arrived. No messages are canceled or sent between the time to which the LP rolled back and the time of the false message. Because these messages are correct there is no need to cancel or re-send them. This increases performance, and it reduces the number of causing roll-backs. Note that another false message or anti-message may arrive before this final phase has completed without causing any problem.

5 Characteristics of the Predictive Network Management System

There are two types of false messages generated in this predictive network management system; those produced by messages arriving in the past Local Virtual Time (LVT) of an LP and those produced because the LP is generating results which do not match reality. If rollbacks occur for both reasons the question arises as to whether the system will be stable. A stable predictive network management system is one in which rollbacks do not have a significant impact on the system performance. A stable system is able to make reasonably accurate predictions far enough into the future to be useful. An unstable system will have its performance degraded by rollbacks to the point where it is not able to predict ahead of real-time. Initial results shown later indicate that predictive network management systems can be stable.

There are several parameters in this predictive network management system which must be determined. The first is how often the predictive network management system should check the LP to verify that past results match reality. There are two conditions which cause LPs in the system to have states which differ from the system being managed and to produce inaccurate predictions. The first is that the predictive model which comprises an LP is most likely a simplification of the actual managed entity and thus cannot model the entity with perfect fidelity. The second reason is that events outside the scope of the model may occur which lead to inaccurate results. However, a benefit of this system is that it will self-adjust for both of these conditions.

The optimum choice of verification query time, T_{query}, is important because querying entities is something the predictive management system should minimize while still guaranteeing that the accuracy is maintained within some predefined tolerance, Θ. For example, the network management station may predict user location as explained later. If the physical layer attempts spatial reuse via antenna beamforming techniques as in the RDRN project, then there is an acceptable amount of error in the steering angle for the beam and thus the node location is allowed a tolerance. The tolerances could be set for each state variable or message value sent from a LP. State verification can be done in one of at least two ways. The LP state can be compared with previously saved states as real time catches up to the saved state times or output message values can be compared with previously saved output messages in the send queue. In the prototype implemented for this predictive network management system state verification is done based on states saved in the state queue. This implies that all LP states must be saved from the LP LVT back to the current time.

The amount of time into the future which the emulation will attempt to venture is another parameter which must be determined. This lookahead sliding window width, Λ, should be preconfigured based on the accuracy required; the farther ahead this predictive network management system attempts to go past real time, the more risk that is assumed.

5.1 Tolerance and Accumulated Simulation Error

In order to consider the impact that out-of-tolerance rollback will have on the predictive system, consider how simulation error occurs. A predictive management system LP may deviate from the real object because either the LP does not accurately represent the actual entity or because events outside the scope of the predictive network management system may effect the entities being managed. Ignore events outside the scope of the simulation for now and consider error form inaccurate simulation modeling only.

Because of this possibility for prediction error, a method of determining the amount of error in a predicted result needs to be developed. A function of total accumulated error in a predicted result, $AC(\cdot)$, is described by Equations 1 and 2. ME_{dp} is the error introduced by the virtual message injected into the predictive system by the driving process. The error introduced by the output message produced by the computation of each LP is represented by the computation error function, $CE(\cdot)$. The actual time taken by the n^{th} LP to calculate and output the next virtual message is t_{lp_n}. Note that the LP topology may not necessarily be a feed-forward network as described by Equations 1 and 2; it may include a cycle. Note also that $\liminf_{\sum t_{lp_i} \to \tau} \sum_{i=1}^{N} CE_{lp_i}(ME_{lp_{i-1}}, t_{lp_i})$ is the greatest lower bound of all sub-sequential limits of $\sum_{i=1}^{N} CE_{lp_i}(ME_{lp_{i-1}}, t_{lp_i})$ as $\sum t_{lp_i} \to \tau$ approaches τ.

$$AC_n(n) = \sum_{i=1}^{N} CE_{lp_i}(ME_{lp_{i-1}}, t_{lp_i}) \tag{1}$$

$$AC_t(\tau) = \liminf_{\sum t_{lp_i} \to \tau} \sum_{i=1}^{N} CE_{lp_i}(ME_{lp_{i-1}}, t_{lp_i}) \tag{2}$$

The driving process is indicated by lp_0. $AC_n(n)$ is the total accumulated error in the virtual message output by the n^{th} LP from the driving process. $AC_t(\tau)$ is the accumulated error in τ actual time units from generation of the virtual message from the driving process. For example, if a prediction result is generated in the third LP from the driving process, then the total accumulated error in the result is $AC_n(3)$. If 10 represents the number of time units after the initial message was generated from the driving process then $AC_t(10)$ would be the amount of total accumulated error in the result.

5.2 Optimum Choice of Verification Query Times

As previously stated, the prototype system performs the verification based on the states in the state queue.

One method of choosing the verification query time would be to query the entity based on the frequency of the data we are trying to monitor. Assuming the simulated data is correct, query or sample in such a way as to perfectly reconstruct the data, e.g. based on the maximum frequency component of the monitored data. A possible drawback is that the actual data may be changing at a multiple of the predicted rate. The samples may appear to to be accurate when they are invalid.

5.3 Verification Tolerance

The verification tolerance, Θ, is the amount of difference allowed between the LP state and the actual entity state. A large tolerance decreases the number of false messages and rollbacks, thus increasing performance and requiring fewer queries, but allows a larger probability of error between predicted and the actual values will cause rollbacks in each LP at real times of t_{vfail} from the start of execution of each LP.

The error throughout the simulated system may be randomized in such a way that errors among LPs cancel. However, if the simulation is composed of many of the same class of LP, the errors may compound rather than cancel each other. The tolerance of a particular LP, Θ_{lp_n}, will be reached in time $t_{vfail_n} = \{\text{lub } \tau \text{ s.t. } AC_t(\tau) > \Theta_{lp_n}\}$. The verification query period (Υ) should be periodic with period less than or equal to t_{vfail_n} in order to maintain accuracy within the tolerance.

The accuracy of any predicted event must be quantified. This could be quantified as the probability of occurrence of a predicted event. The probability of occurrence will be a function of the verification tolerance, the time of last rollback due to verification error, the error between the simulation and actual entity, and the sliding lookahead window.

Every LP will be in exact alignment with its PP as a result of a state verification query. This occurs every $T_{query} = t_{vfail}$ time units.

5.4 Length of Lookahead Window

The length of the lookahead window, Λ, should be as large as possible while maintaining the required accuracy. The total error is also a function of the chain of messages which lead to the state in question. Thus the farther ahead of real-time the predictive network management system advances, $t_{ahead} = GVT - t_{current-time}$, the greater the number of messages before a verification query can be made and the greater the error. The maximum error is $AC_t(\Lambda)$.

5.5 Simulation Time

Since the verification query time is less than or equal to the current time, $t_{current-time}$, rollbacks due to the verification query will take the LP back to the current time. Thus GVT as defined in [7] is no longer a lower bound on the simulation rollback time. The lower bound is now always $t_{current-time}$. GVT is still required in order to determine how far into the future the predictive network management system has gone.

5.6 Calibration Mode of Operation

It may be helpful to run the predictive network management system in a mode such that error between the actual entities and the predictive network management system are measured. This error information can be used during the normal predictive mode in order to help set the above parameters. This begins to remind one of back propagation in a neural network, i.e. the predictive network management system automatically adjusts parameters in response to real output in order to become more accurate.

This calibration mode could be part of normal operation. The error can be tracked simply by keeping track of the difference between the simulated messages and the result of verification queries.

6 Model and Simulation

An initial test of this concept has been performed in a simulation of a predictive management system implemented with Maisie [18]. Maisie is the simulation environment used here. Its suitability for this has been demonstrated in the RDRN network management and control design and development and in [19] to develop a mobile wireless network parallel simulation environment. The parallel simulation environment shows a speedup over the currently used commercial sequential simulation packages. The environment and a set of modules which have been developed for mobile network simulation are described in [19]. Maisie uses a language which has been influenced by a classic work describing the characteristics of a parallel programming language structure [20]. The programming features developed in [20] are used in many parallel programming languages besides Maisie. Since every Maisie entity has a built-in input queue, each LP is comprised of three additional Maisie entities:

- An entity which represents the PP
- An entity for the LP state queue
- An entity for the LP output message queue

There is also a gvt entity for the calculation of GVT. All three of the above entities work together to implement Virtual Time as described in [7]. The first entity above, representing the PP, contains a delay mechanism in order to implement the sliding lookahead window. The gvt process should notify all processes to cease forward simulation when GVT reaches the end of the window. In this version of the predictive management system, each LP simply compares its LVT to the current time and holds processing until current time is back within the lookahead sliding window.

Determination of Global Virtual Time (GVT) should be done as defined by [21]. This algorithm allows GVT to be determined in a message-passing environment as opposed to the easier case of a shared memory environment. It also allows normal processing to continue during the GVT determination phase. However, in this implementation each output message is sent to the gvt entity as well as to its proper destination. In addition, the gvt entity checks all LPs for their current LVT and chooses the minimum message send time and LVT as the current GVT. The gvt entity is allowed to execute in parallel with the other entities in this simulation, it does not stop the other entities while performing its computation and thus may not always be perfectly accurate. This is because messages may be in transit when the poll takes place, and because the LPs are changing while the GVT computation is taking place. However, the results were close enough for the purpose of these experiments.

6.1 Verification Query Rollback Versus Causality Rollback

Verification query rollbacks are the most critical part of the predictive management system. They are handled in a slightly different fashion from causality failure rollbacks. A state verification failure causes the LP state to be corrected at the time of the state verification which failed. The state, S_v, has been obtained from the actual device from the verification query at time t_v. The LP rolls back to exactly t_v with state, S_v. States greater than t_v are removed from the state queue. Anti-messages are sent from the output message queue for all messages greater than t_v. The LP continues forward execution from this point. Note that this implies that the message and state queues cannot be purged of elements which are older than the GVT. Only elements which are older than real time can be purged.

6.2 The Prototype System Simulation

In order to test the concept, a simple system was simulated to represent the predictive management protocol just described. Note that none of the previous assumptions are made in the simulation. The purpose of this simulation is to determine if the concept is feasible. A key question this simulation attempts to answer is whether the overhead/performance ratio results in a useful system. A small closed queuing network with FCFS servers is used to represent the actual system. Figure 1 shows the real system to be managed and the predictive management model. In this initial feasibility study, the managed system and the predictive management model are both modeled with Maisie. The verification query between the real system and the management model are explicitly illustrated in Figure 1.

The system consists of three switch-like entities, each switch contains a single queue and switches consisting of 10 exponentially distributed servers which must sequentially service each packet. A mean service time of 10 time units is assumed. The servers represent the link rate. The packet is then forwarded with equal probability to another switch, including itself. Each switch is a driving process; the switches forward real and virtual messages. The cumulative number of packets which have entered each switch and queue is the state. This is similar to SNMP [10] statistics monitored by SNMP Counters, for example, the **ifInOctets** counter in MIB-II interfaces [22].

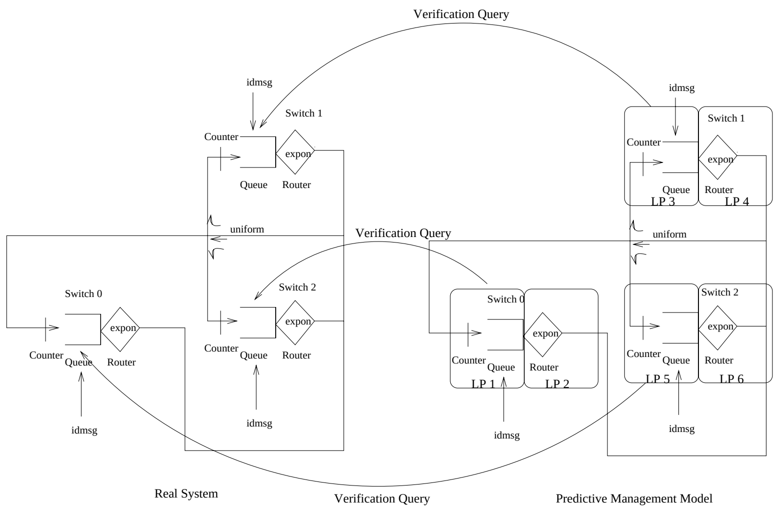

Figure 1: Initial Feasibility Network Model

Both real and virtual messages contain the time at which service ends and a count of the number of times a packet has entered a switch. The switches are fully connected. An initial message enters each queue upon startup to associate a queue with its switch. This is the purpose of the **idmsg** which enters the queues in Figure 1. The predictive system parameters are more compactly identified as a triple consisting of Lookahead Window Size (seconds), Tolerance (counter value), and Verification Query Period (seconds) in the form $(\Lambda, \Theta, \Upsilon)$. The effect of these parameters are examined on the system of switches previously described. The simulation was run with the following triples: $(5, 10, 5)$, $(5, 10, 1)$, $(5, 3, 5)$, $(400, 5, 5)$. The graphs which follow show the results for each triple.

The first run parameters were $(5, 10, 5)$. There were no state verification rollbacks although there were some causality induced rollbacks as shown in Figure 2. GVT increased almost instantaneously versus real time; at times the next event far exceeded the look-ahead window. This is the reason for the nearly vertical jumps in the GVT as a function of real-time graph as shown in Figure 2. The state graph for this run is shown in Figure 3.

In the initial implementation, state verification was performed in the LP immediately after each new message was received. However, the probability that an LP had saved a future state, while processing at its LVT, with the same state save time as the time at which a real message arrived was low. Thus, there was frequently nothing with which to compare the current state in order to perform the state verification. However, it was observed that the predictive system was simulating up to the lookahead window very quickly and spending most of its time holding, during which time it was doing nothing. The implementation was

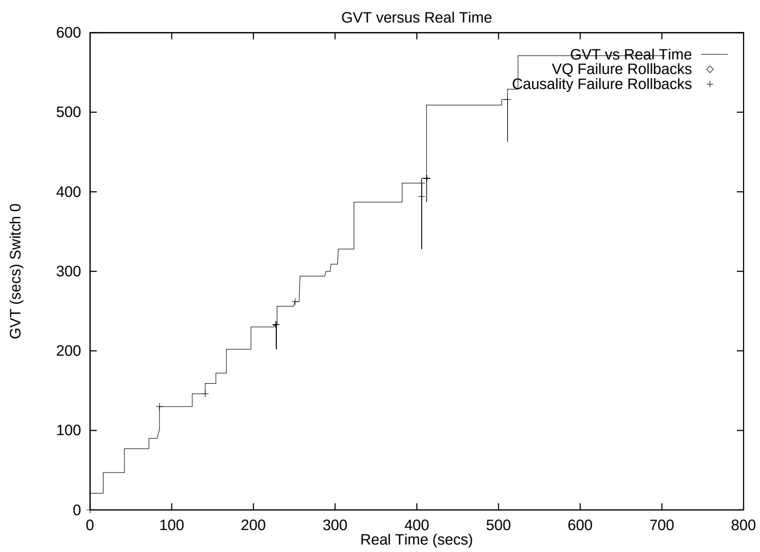

Figure 2: Rollbacks Due to State Verification Failure (5, 10, 5)

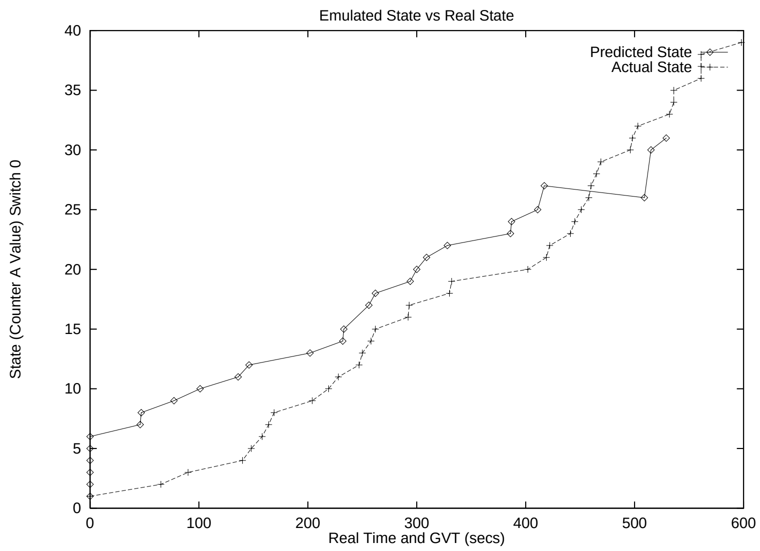

Figure 3: State (5, 10, 5)

modified so that each entity would perform state verification during its hold time. This design change better utilized the processors and resulted in more accurate alignment between the actual and logical processes.

The results for the $(5, 10, 1)$ run were similar, except that the predictive and actual system comparisons were more frequent because the state verification period had been changed from once every 5 seconds to once every second. Error was measured as the difference in the predicted LP state versus the actual system state. This run showed errors that were greater than those in the first run, great enough to cause state verification rollbacks. The error levels for both runs are shown in Figures 4 and 5. The state graph for this run is shown in Figure 6.

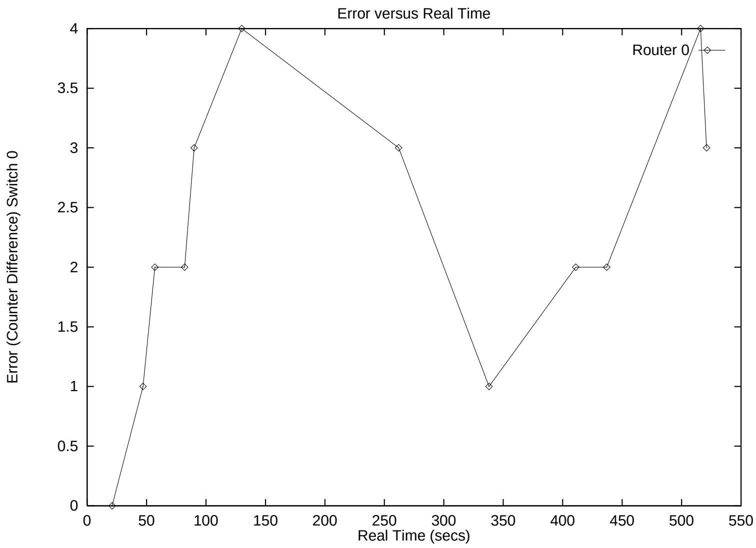

Figure 4: Amount of Error (5, 10, 5)

The next run used $(5, 3, 5)$ parameters. Here we see many more state verification failure rollbacks as shown in Figure 7. This is expected since the tolerance has been reduced from 10 to 3. The cluster of causality rollbacks near the state verification rollbacks was expected. These clusters of causality rollbacks do not appear to significantly reduce the feasibility of the system. The real-time versus GVT plot as shown in Figure 7 shows much larger jumps as the LPs were held back due to rollbacks. The entities had a larger variance in their hold times than the $(5, 10, 5)$ run. The state graph for this run is shown if Figure 8.

A $(400, 5, 5)$ run showed the GVT jump quickly to 400 and then gradually increase as the sliding lookahead window maintained a 400 time unit lead as shown in Figure 9. The LP hold times were shorter here than an any previous run. The state graph for this run is shown in Figure 10.

This set of results is interesting because it shows the system to be stable with

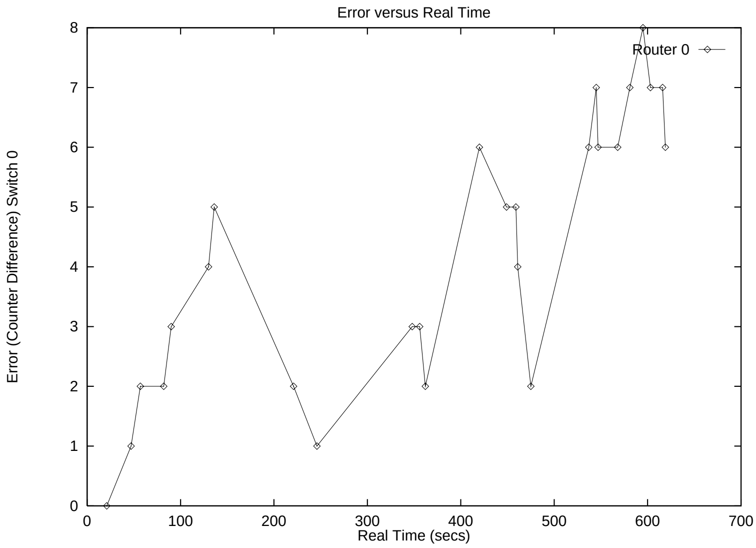

Figure 5: Amount of Error (5, 10, 1)

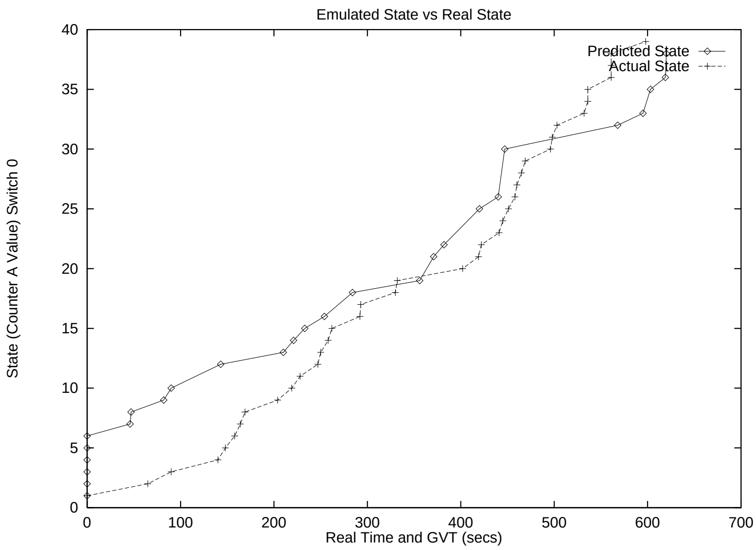

Figure 6: State (5, 10, 1)

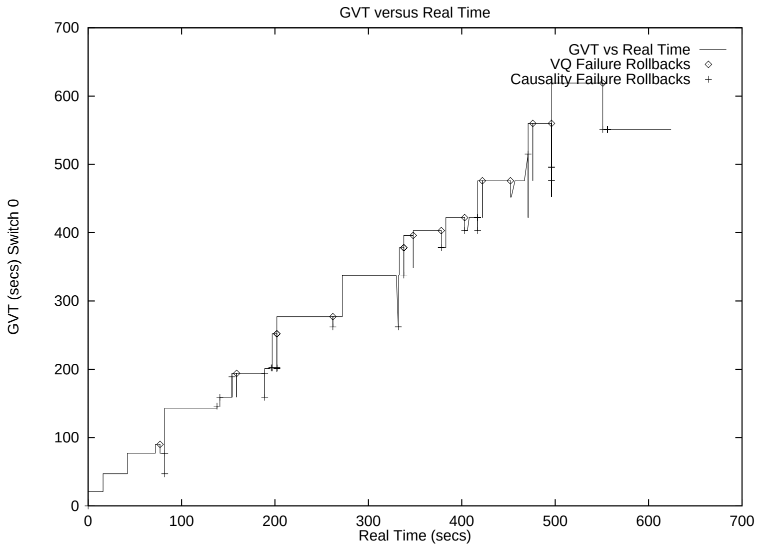

Figure 7: Rollbacks Due to State Verification Failure (5, 3, 5)

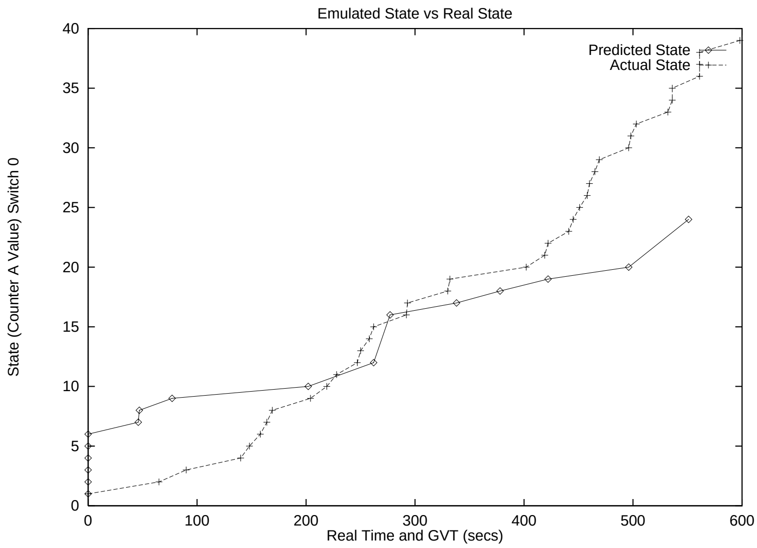

Figure 8: State (5, 3, 5)

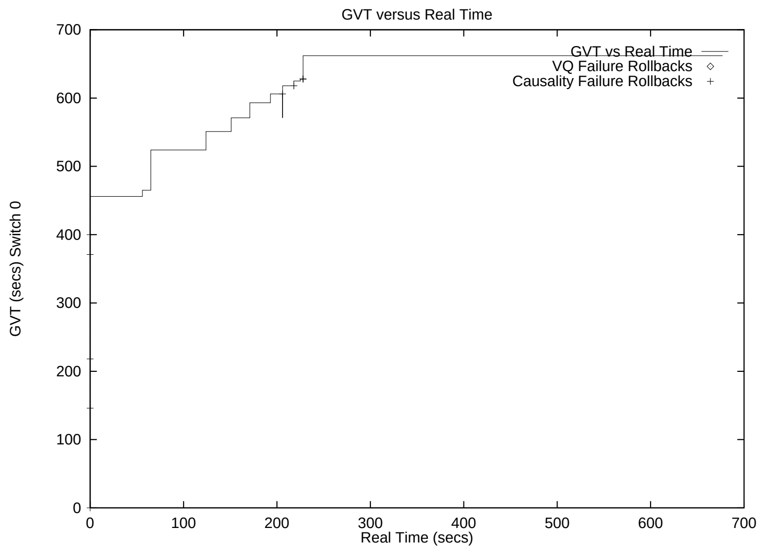

Figure 9: Rollbacks Due to State Verification Failure (400, 5, 5)

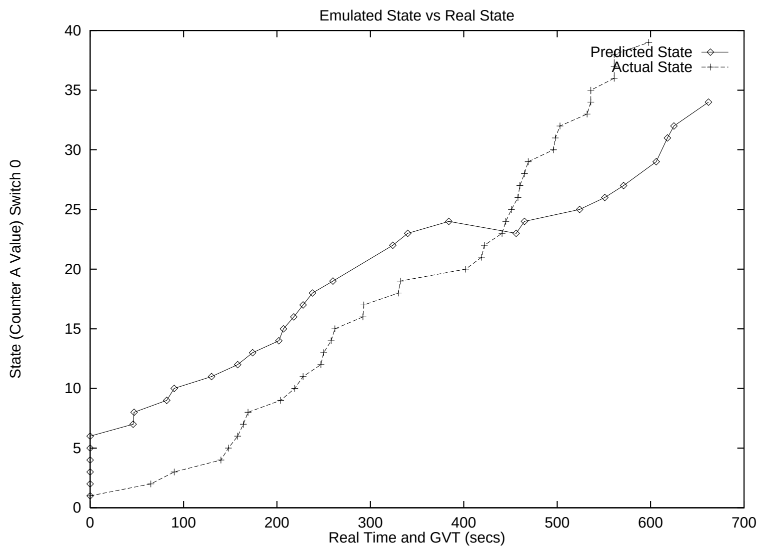

Figure 10: State (400, 5, 5)

the introduction of state verification rollbacks. The overhead introduced by these rollbacks did not greatly impact the performance, because as previously shown in the GVT versus time graphs, Figures 2, 7 and 9, the system was always able to predict up to its lookahead time very quickly.

7 Optimizing Management Polling with the Predictive Manager

Since the predictive network management system provides a good approximation of the future behavior of the data to be managed as shown in the GVT versus real time values of state in Figures 3, 6, 8, and 10, the verification query period can be automatically determined as a function of the look-ahead window and tolerance, with the goal of minimizing the frequency of verification queries thus solving the polling problem in network management.

In most standards based approaches, network management stations are sampling counters in managed entities which simply increment in value until they roll over. A management station which is simply plotting data will have some fixed polling interval and record the absolute value of the difference in value of the counter. Such a graph is not a perfectly accurate representation of the data, it is merely a statement that sometime within a polling interval the counter has monotonically increased by some amount. Spikes in this data, which may be very important to the current state of the system, may not be noticed if the polling interval is too long such that a spike followed by low data values averages out to a normal or low value. Our goal is to determine the minimum polling interval required to accurately represent the data.

From the information provided by the predictive management system, a polling interval which provides the desired degree of accuracy can be determined and dynamically adjusted; however, the cost must be determined as discussed next.

An upper limit on the number of systems which can be polled is $N \leq \frac{T}{\Delta}$ where N is the number of devices capable of being polled, T is the polling interval, and Δ is the time required for a single poll. Thus although the data accuracy will be constrained by this upper limit, taking advantage of characteristics of the data to be monitored can help distribute the polling intervals efficiently within this constraint. Assume that Δ is a calculated and fixed value, as is N. Thus this is a lower bound on the value of $T \geq \Delta N$.

The overhead bandwidth required for use by the management system to perform polling is shown in Equation 3. The packet size will vary depending upon whether it is an SNMP or CMIP packet and the MIB object(s) being polled. The number of packets varies with the amount of management data requested. Let K be the number of packets, L be the bits/packet, N be the number of devices polled, and T be the polling period. BW is the total available bandwidth and BW_{oh} is the overhead bandwidth of the management traffic.

$$\%BW_{oh} = \frac{\frac{NumberofPackets * N * \frac{Bits}{Packet}}{T}}{BW} * 100 \qquad (3)$$

We may want to limit the bandwidth used for polling system management data to be no more than a certain percentage of total bandwidth. Thus the optimum polling interval will use the least amount of bandwidth while also maintaining the least amount of variance due to error in the data signal. All the required information to maintain the cost versus accuracy at a desired level is provided by the predictive network management system.

8 Interaction between a Predictive Management System and a Predictive Mobile Network

There is an interesting interaction between the predictive management system and the predictive mobile network. A predictive mobile network such as the VNC proposal for RDRN [4] will have results cached in advance of use for many configuration parameters. These results should be part of the Management Information Base (MIB) for the mobile network and should include the predicted time of the event which requires the result, the value of the result, and the probability that the result will be within tolerance at that time. Thus there will be a triple associated with each predicted event: *(time, value, probability)*. Network management protocols, e.g. SNMP [10] and CMIP [11], include the time as part of the PDU, however this time indicates the real time the poll occurred and should not be changed.

A predictive management system could simply use LPs to represent the predictive mobile processes as previously described, however, this is redundant since the mobile network itself has predicted events in advance as part of its own management and control system. Therefore, managing a predictive mobile network with a predictive network management system provides an interesting problem in trying to get the maximum benefit from both of these predictive systems.

Combining the two predictive systems in a low level manner, e.g. allowing the LPs to exchange messages with each other, raises questions about synchronization between the mobile network and the management station. However, the predicted mobile network results can be used as additional information to refine the management system results. The management system will have computed *(time, value, probability)* triples for each predicted result as well. The final result by the management system would then be an average of the *times* and *values* weighted by their respective probabilities. An additional weight may be added given the quality of either system. For example the network management system might be weighted higher because it has more knowledge about the entire network. Alternatively, the mobile network system may weighted higher because the mobile system may have better predictive capability for the detailed events concerning handoff. Thus the two systems do not directly interact with each other, but the final result is a combination of the results from both predictive systems. A more complex method of combining results from these two systems would involve a causal network such as the one described in [23].

9 Conclusion

Network management systems capable of not only passive monitoring but also of active prediction capabilities are undergoing research and development. Work on prediction mechanisms for mobile communication networks is also underway. The method used by standards-based network management systems to cope with these two developments have been discussed in this paper.

Characteristics of a predictive network management system have been presented. The Rapidly Deployable Radio Network mobile communications system is used as an example of a predictive mobile communications network. The interaction between the predictive capabilities of these two systems has been discussed.

References

[1] Srinivasan Seshan, Hari Balakrishnan, and Randy H. Katz. Handoffs in Cellular Wireless Networks: The Daedalus Implementation Experience. *Kluwer International Journal on Wireless Personal Communications*, 1996.

[2] George Liu, Alexander Marlevi, and Gerald Q. Maguire Jr. A Mobile Virtual-Distributed System Architecture for Supporting Wireless Mobile Computing and Communications. In *Mobicom '95*, 1995.

[3] George Y. Liu. *The Effectiveness of a Full-Mobility Architecture for Wireless Mobile Computing and Personal Communications.* PhD thesis, Royal Institute of Technology, Stockholm, Sweden, March 1996.

[4] Stephen F. Bush, Sunil Jagannath, Joseph B. Evans, and Victor Frost. A Control and Management Network for Wireless ATM Systems. In *Proceedings of the International Communications Conference '96*, pages 459,463, June 1996. Conference web page http://www-ee.uta.edu/organizations/-commsoc/commsoft.html.

[5] Stephen F. Bush, Sunil Jagannath, Ricardo Sanchez, Joseph B. Evans, Victor Frost, and K. Sam Shanmugan. Rapidly Deployable Radio Networks

(RDRN) Network Architecture. Technical Report 10920-09, Telecommunications & Information Sciences Laboratory, July 1995. Online version available at http://www.tisl.ukans.edu/~sbush .

[6] Analucia Schiaffino Morales De Franceschi, Liuz Fernando Kormann, and Carlos Becker Westphall. Performance Evaluation for Proactive Network Management. In *Proceedings of ICC'96*, pages 22,26, 1996.

[7] D. R. Jefferson and H. A. Sowizral. Fast Concurrent Simulation Using The Time Warp Mechanism, Part I: Local Control. Technical Report TR-83-204, The Rand Corporation, 1982.

[8] L. Steinber, editor. *Techniques for Managing Asynchronously Generated Alerts.* IETF, May 1991.

[9] B. Meandzija and J. Westcott, editors. *Guidelines for Structuring Manageable Entities.* North-Holland, 1989.

[10] Marshall T. Rose. *The Simple Book, An Introduction to the Management of TCP/IP Based Internets.* Prentice Hall, 1991.

[11] ISO. Open Systems Interconnection - Management Protocol Specification - Part 2: Common Management Information Protocol.

[12] L. Lamport. Time, Clocks, and the Ordering of Events in a Distributed System. *Communications of the ACM*, July 1978.

[13] Robert E. Felderman and Leonard Kleinrock. An Upper Bound on the Improvement of Asynchronous versus Synchronous Distributed Processing. In *SCS '90*, January 1990.

[14] Yi-Bing Lin. Understanding the Limits of Optimistic and Conservative Parallel Simulation. Technical Report UWASH-90-08-02, University of Washington, 1990.

[15] Richard J. Lipton and David W. Mizell. Time Warp vs. Chandy-Misra: A Worst-Case Comparison. In *SCS '90*, January 1990.

[16] John Turnbull. A Performance Study of Jefferson's Time Warp Scheme for Distributed Simulation. Master's thesis, Case Western Reserve University, May 1992.

[17] H. Takagi. *An Analysis of Polling Systems.* MIT Press, 1986.

[18] Rajive Bagrodia and Wen-Toh Liao. *Maisie User Manual Release 2.1*, jun 1993.

[19] Joel Short, Rajive Bagrodia, and Leonard Kleinrock. Mobile Wireless Network System Simulation. In *Mobicom '95*, 1995.

[20] C. A. R. Hoare. Communicating Sequential Processes. *Communications of the ACM*, August 1981.

[21] Edward Lazowaska and Yi-Bing Lin. Determining the Global Virtual Time in a Distributed Simulation. Technical Report 90-01-02, University of Washington, 1990.

[22] K. McCloghrie, editor. *Management Information Base for Network Management of TCP/IP-based Internets.* Hughes LAN Systems, May 1990. Online version available at http://ds.internic.net/rfc/rfc1156.txt.

[23] F. Lehmann, R. Seising, and E. Walther-Klaus. Simulation of Learning in Communication Networks. *Simulation Practice and Theory*, 1(1):41–48, July 1993.