COMPUTER NETWORK PERFORMANCE PREDICTION FOR THE NEAR FUTURE THROUGH FASTER THAN REAL TIME SIMULATION

D. Anagnostopoulos, M. Nikolaidou, G. Philokyprou
Dept. of Informatics
University of Athens
Panepistimiopolis, Athens, 15771, Greece
email: dimosthe@di.uoa.gr

Keywords: Network Simulation, Faster than Real Time Simulation, Object-Oriented Modeling

Abstract

Real time simulation is often used to extend the capabilities of traditional simulation, though not in the computer network domain due to the network operation rate. A rather innovative faster than real time approach that contributes to network performance evaluation through reaching conclusions for the near future is applied for relatively slow (10Base) networks. Since simulation activities are executed in real time and involve both the model and the network under study, modeling and experimentation requirements are drastically increased. Methodological issues for dealing with these requirements are thoroughly examined. Simulation results and conclusions for real time experimentation in the dynamic network environment are also discussed.

1 INTRODUCTION

Simulation contributes considerably to the performance evaluation of networks and individual network entities since, when compared to mathematical and analytical modeling, it provides extended capabilities for the representation of real conditions, the combination of complex entities and the analysis of an overall network architecture.

Numerous simulation tools are active in the network domain, distinguished according to their orientation, modeling decisions and techniques and the degree of automation supported (CACI 1997; Mil3 Inc 1997). Modeling tools usually adopt a layering scheme and are object-oriented when extending to the overall network architecture. Communication-oriented simulators are the most widely accepted simulation software category (Law and McComas 1994), since they reduce program development time and enable automated model construction through built-in modules, closely related to the components of a communications network (CACI 1997).

Network simulation tools extract conclusions for the network performance according to the average or worst case scenario and rely on either hypothetical or previously encountered input data when representing real conditions. The expressiveness of such data is considerably reduced, especially when the state of the network is not steady (e.g. when a network node crashes). Thus, simulation tools only contribute to an “off-line” network performance analysis, since network behavior is intensely dynamic.

In this paper, we present the methodology and results from a prototype application of real time simulation concepts in the network domain, according to the methodological approach introduced by the authors. The application domain is computer networks (local area networks). Selection of this application domain enables dealing with all network reformation types, but makes questionable whether it is always possible to achieve faster than real time simulation. For this reason, an actual network examined is the Ethernet local network of a university campus building, consisting of 10Base segments, which is relatively slow. In these experiments, real data were used to extract conclusions for the network performance in the near future so that an enhanced potential -compared to this of conventional NMSs- would be provided. A prototype simulation environment was also constructed for this objective.

In the following sections, we describe a real time simulation methodology for conducting the real time experiments and the requirements imposed for the various simulation phases. Decisions and techniques adopted for both modeling and experimentation and the prototype environment architecture are then discussed. Finally, faster than real time simulation results indicating both the potential of this approach as well as the dynamic nature of

the network, along with conclusions, are discussed in the last section.

2 SIMULATION METHODOLOGY

The real time dimension imposes requirements as the following for the simulation process (Anagnostopoulos et al 1995a):

- Ensuring that the model runs faster than the real network
- Adapting the model to the current network state
- Ensuring the validity of the model

The execution time of the model and the overall simulation environment is critical, since it relates to the collection and processing of real time data from the network and concluding for the network current and future states. Performing dynamic modifications to the model during experimentation is also required in order to adapt the model to the real conditions. This may be achieved through appropriately integrating new components into the model and disposing components not currently in use. In this way, simulation code recompilation is not required when dealing with cases as the incorporation of additional entity models within the model. Finally, validity of all - either primitive or composite models- must be ensured before models are used.

The network operation scenarios handled by the simulation environment include the initiation of new applications (when these have a significant impact on the overall network load), critical modification of the application load, application termination, node start up and active node crash. When these occur, the simulation environment must be in position to collect and process network data, adapt the model and reach appropriate conclusions.

Handling the complexity encountered is supported through an extended simulation methodology. This methodology is based upon the traditional simulation process, provides extensions to the existing phases due to the increased number of tasks to be accomplished and incorporates new phases. The real time simulation methodology used, as described in (Anagnostopoulos et al 1995b)], consists of the following main phases.

- Modeling
- Experimentation
- Remodeling

The above phases as well as their invocation sequence are depicted in figure 1. A discussion for each phase is about to follow, emphasizing on the individual tasks. To accomplish them, specific techniques and methodological guidelines are adopted.

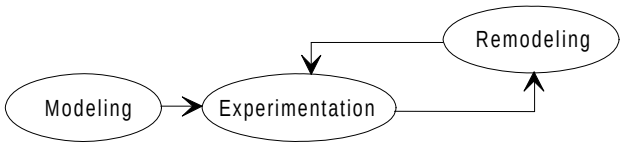

Figure 1: Invocation sequence of simulation phases

Modeling is the initial simulation phase where the composite network model is constructed. During experimentation, both the network and the model are under monitoring. Data depicting their consequent states are obtained and recorded within predetermined time intervals and consequently analyzed. In this way, both systems evolution is audited. In the case where the state of the model deviates from the real one remodeling is invoked, without terminating the real time experiment, since no recompilation is needed. When the model adaptation to the real conditions is completed, experimentation resumes.

Even though not included in the above figure, in case simulation results (predictions for the near future) are considered to be valid, a fourth, independent phase can be invoked to take advantage of these results and take act appropriately. In this paper, we do not emphasize on methodological issues about this phase. However, an in-depth description can be found in (Anagnostopoulos et al 1995b).

3 MODELING FRAMEWORK

The introduced modeling framework aims at providing extended capabilities for conducting network simulation experiments, without imposing limits to the efficient representation of network entities. In the following, we present the modeling decisions and techniques for network entity modeling.

The requirements for handling modifications to the network and ensure model validity are handled through the use of modular models that have a hierarchical structure, according to which components are coupled together to form larger models. This is accomplished based on object-oriented modeling and use of preconstructed model components, which are organized in object hierarchies and reside in model libraries. Either model components or composite models can be preconstructed, representing the key network entities. Since preconstructed models must correspond to all

entities that are likely to participate in the network, all acceptable individual entity combinations must be supported. Preconstruction of primitive and composite models is thus enabled for all higher level entities, corresponding to the accepted primitive entity combinations, and is expected to extend to the level where structural modifications to the network may be encountered (i.e. nodes and applications).

Object oriented modeling is therefore used along with process oriented simulation.

The issue of making the network model run faster than the real network is not discussed in this paper; however, generic approaches towards this direction are already established (Lee and Fishwick 1997).

As key network entities of local networks, modeled in the form of preconstructed components, the following are considered (Cramer and Magee 1992): communication protocols, communication and processing nodes, communication links and applications.

The above classification corresponds to a discrete layering scheme of network operations (Jones and Smythe 1993), as the OSI reference model.

The modeling framework is based on hierarchical layering to depict the functionality of individual network entities. Hierarchical layering enables the construction of more complex models through extending the behavior of existing objects and ensures the uniform manipulation of all equivalent network entities. In this way, common entity models can be accessed through the same public interface, through polymorphism.

Hierarchical layering extends to both primitive and composite network entities. Implementation of this scheme proves rather complex and time consuming. However, the outcome is the pre-construction of all necessary network components, which ensures the availability of all -possibly required- models.

Network models consist of composite and primitive models. Applications and communication links are perceived as primitive entities. Network nodes are decomposed in terms of their communication and processing elements (virtual entities, used to represent the communication and processing properties of the node). In this approach, emphasis is given to communication related issues. The processing element is thus modeled as a primitive entity, while the communication element is composite, formed on the basis of the communication protocol stacks operating on each node. Network nodes can be either processing or communication nodes. The communication elements of processing nodes use identical protocol stacks. Protocol stacks forming the communication elements of communication nodes represent the interconnection mechanisms of these nodes and are more complex. An abstract view of network model composition is depicted in figure 2.

When considering data transmission and receive, two additional requirements for protocol stack modeling must be supported. First, the highest supported layer in local network nodes is not predetermined. The same does not apply for the lowest level. Second, it is possible that intermediate layers are not supported.

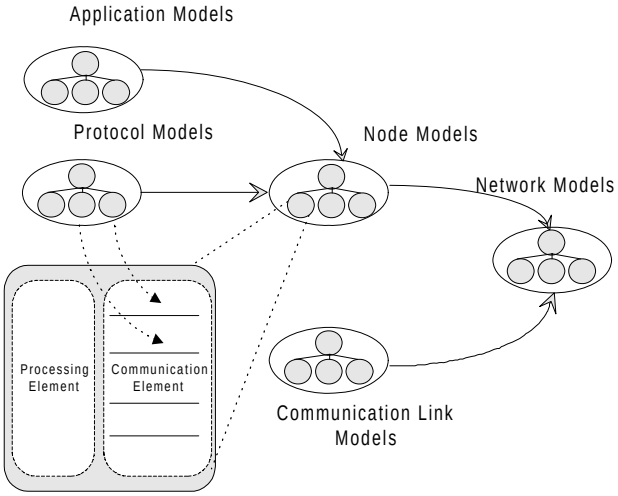

Figure 2: Abstract view of network model composition

To give an example of the preconstruction of all required models for a single entity, application modeling is discussed. Modeling of other network entities, as processing nodes, is performed on the same guidelines.

Although applications are associated with a single node, they operate independently. Their operation involves creation of data units, forwarding them to the highest protocol layer supported and receiving data units in reverse order.

Due to the requirement to support different kinds of applications and the inability to determine the highest supported layer in local networks, application models must be in position to create and forward data units to all candidate highest layers. Application models must then support mechanisms for determining at least the scheduling, size and destination of data units, as already established in commercial tools (CACI 1997). Since different layers handle different data units, application models must be also provided for all layers.

The proposed model hierarchy, focusing on data link layer applications is presented in figure 3.

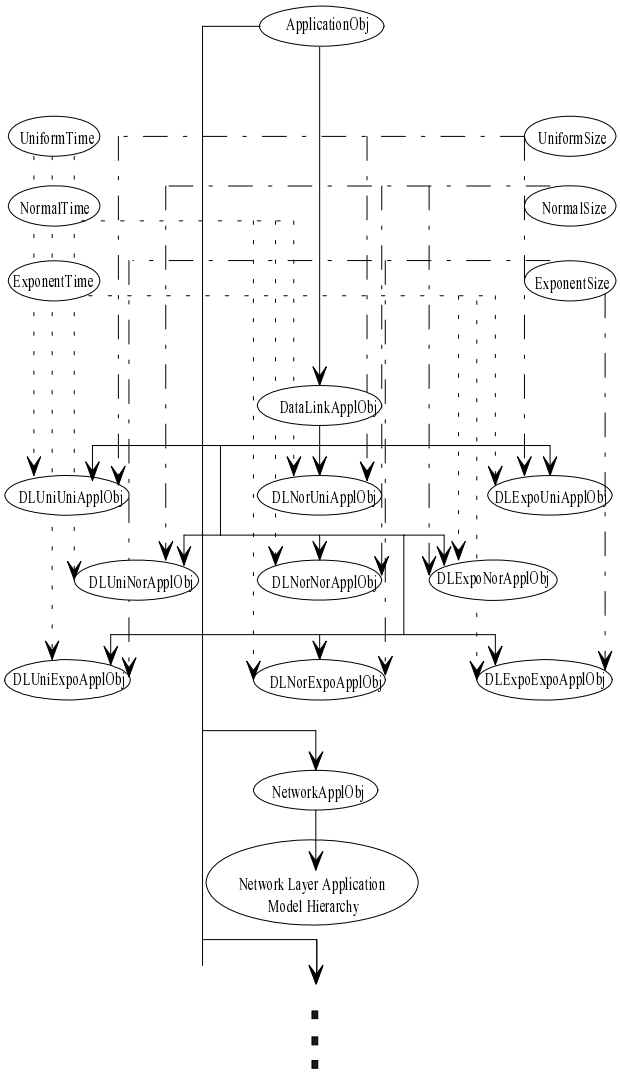

Figure 3: Sample application model hierarchy

Hierarchies are constructed having a base application model as a common predecessor. An abstract model for each supported layer is derived as an ancestor. Additional primitive objects are constructed to provide the required functionality for data unit scheduling and size determination. Customized application models are then derived as ancestors of a) the generic model of the specific layer applications are forwarding data to and b) the primitive objects providing the required functionality.

Multiple inheritance is used to implement complex structures for the required data unit scheduling and size combinations. If, for example, there are five possible ways for scheduling data unit creation and four for determining its size (according to corresponding distributions), nine primitive objects are to be provided and twenty derived ones have to be constructed for the corresponding application models for a single layer. The nine primitive objects can also be used when constructing other layer models. Nevertheless, twenty additional models have to be derived for each other layer supported to provide an equivalent functionality.

Implementation of the proposed modeling scheme proves to be rather complex and time consuming. However, as presented in the next section, the time required when remodeling the overall network using preconstructed application models is minimum and, when objects structures are subjected to insert, update and delete operations, this process can be highly automated.

4 EXPERIMENTATION

Experimentation with the actual network and the simulation model is not trivial, since new activities must be carried out and new parameters must be determined. These issues are discussed in this section. The objective of the simulation environment is to maintain the consistency between the model and the network and to ensure the reliability for the network performance predictions.

Experimentation phase encompasses monitoring and auditing (the activity of examining whether both systems are evolving towards the same direction). Time restrictions are imposed and concern a) the model execution, since it must be faster than the real network speed b) auditing, that must be completed with minimum time overhead, since all other activities are paused prior to concluding for the model evolution and c) remodeling, when restoring consistency is needed, for the same reasons.

Network performance evaluation can only be performed when efficiently handling frequent scenarios that cause major discrepancies to the network, as the crashing of a network node. The model must then be customized to the new conditions and still be in position to ensure the reliability of results. It is therefore essential that the simulation environment is capable of collecting and processing real network data as well as extracting conclusions in real time. Numerous networking issues emerge during the network operation. These can be divided in two main categories: the ones intervening to the network structure and the ones intervening to input data and network parameters. Structural modifications acquire an increased degree of complexity. The following are common networking issues for discussing the effectiveness of the real time simulation approach: critical modification of the application load, node start up and active node crash.

To conclude about occurrence of any of these scenarios, specific measures of both systems are put under monitoring. The variables used to obtain the corresponding values are referred as *monitoring variables*. Auditing examines variable values corresponding to the

same time points (i.e. the current network state and simulation predictions for this time point) and concludes for the evolution of the network and the model.

Assuming that a node crashes, monitoring data are cross-examined to indicate the specific node and applications affected. Auditing indicates that a specific node and applications are no longer active and concludes that structural and input data reformations have occurred. Remodeling is thus invoked, which removes and disposes the corresponding components from the model composition tree.

When all required modifications are accomplished, the resulting model is once more subjected to experimentation, starting from the current real time point. In case that other reformations imposed the integration of additional components, appropriate models would be directly imported from model libraries, initialized and dynamically linked to the network model.

Monitoring is thus performed during predetermined *monitoring intervals* and when being completed, auditing is initiated. Evidently, monitoring variables must be commonly defined for both systems. Monitoring variables as the following can be used for computer networks:

1. Number of active Nodes (*ActiveNodes*)
2. Number of active Sessions (*NumSessions*)
3. Throughput (packets) (*Pthroughput*)
4. Throughput (bytes) (*Bthroughput*)
5. Data units per Session (*PacketSession*$_i$, *1<=i<=NumSessions*)
6. Bytes per Session (*ByteSession*$_i$, *1<=i<=NumSessions*)
7. Delay per Session (*DelaySession*$_i$, *1<=i<=NumSessions*)
8. Number of Collisions (*Collisions*$_i$, *1<=i<=ActiveNodes*)

From the above variables, only the last one is explicitly related to the specific network under study (10Base network).

Auditing examines the values of monitoring variables from the model and the network on the basis of predetermined criteria. These define the range of acceptable deviation between of values of simulation variables and the corresponding ones of the actual network before concluding that deviations are encountered. The algorithm determining how corresponding variable values are compared in order to decide whether remodeling should be invoked is referred as *auditing algorithm.*

For the purposes of this paper, a simplified example using the basic inspection method for comparing real observations and simulation results is presented, whereas a more efficient approach can be established on the basis of confidence intervals. In this example, the bytes transferred through session *i* in the model and the network are noted as *SByteSession*$_i$ and *RByteSession*$_i$, respectively, and the model is considered to deviate when:

$$SByteSession_i \notin [RByteSession_i - RByteSession_i * D_{ni}, RByteSession_i + RByteSession_i * D_{ni}]$$

where D_{ni} is the *deviation parameter* determining the range of the above interval. Deviation parameters must be defined for all monitoring variables.

The values of deviation parameters vary according to the specific variable and the orientation of the simulation experiment, since experiments can emphasize on different aspects. Lower values contribute to the reliability of simulation predictions, but lead more frequently to remodeling thus causing results to be discarded. High values have opposite effects. In the general case, values in the range [0.0, 0.8] prove to be efficient, as indicated through a number of experimental trials.

The lower limit (0.0) is used when no deviation is acceptable, as for the number of active nodes. The upper limit is not as high as it may seem, considering that a) in the general case, more than one monitoring variables must be within the corresponding range and b) use of aggregate variables, as *Pthroughput*, provide a verification mechanism for individual ones, as *PacketSession*$_i$. Use of aggregate variables is also enabled when monitoring of specific measures cannot be directly accomplished.

Simulation results for the examples presented in this paper are produced using the values of Table 1.

N	Variable Name	D parameter value
1	*ActiveNodes*	0.0
2	*NumSessions*	0.1
3	*Pthroughput*	0.5
4	*Bthroughput*	0.5
5	*PacketSession*$_i$	0.8, $\forall i: 1<=i<=NumSessions$
6	*ByteSession*$_i$	0.8, $\forall i: 1<=i<=NumSessions$
7	*DelaySession*$_i$	0.8, $\forall i: 1<=i<=NumSessions$
8	*Collisions*$_i$	0.8, $\forall i: 1<=i<=ActiveNodes$

Table 1: Deviation parameter values for monitoring variables

A significant issue that must be addressed is the length of the monitoring interval. This is the time period where data collection is performed for both systems. Auditing initiates after the completion of the monitoring interval.

A short interval results in the following disadvantages: a) frequent auditing, before a considerable amount of data is collected and b) wasting time, since execution of monitoring and auditing has a significant cost and no other activity can be carried out in parallel. The selection of a 10-sec monitoring interval falls in this category.

On the other hand, if a long interval is used, it is possible that the simulation environment for a considerable time period will not perceive certain events, like a node crash. The selection of a 5-min monitoring interval would fall in this category, which also is not acceptable. Based on these conclusions and a number of initial trials, a 60-sec monitoring interval was finally used.

5 SIMULATION ENVIRONMENT

A prototype implementation based on the above methodology was carried out for a local network environment using Modsim III simulation language (Anagnostopoulos et al 1995c). The implementation domain is the TCP/IP local network of a university campus building, consisting of 10BaseT and 10Base5 segments. It also includes more than 30 computer nodes for students and academic staff.

The real time simulation environment integrates tools supporting the individual simulation phases. The environment architecture is illustrated in figure 4. Connections with arrow ends between modules indicate both information flows and module invocation.

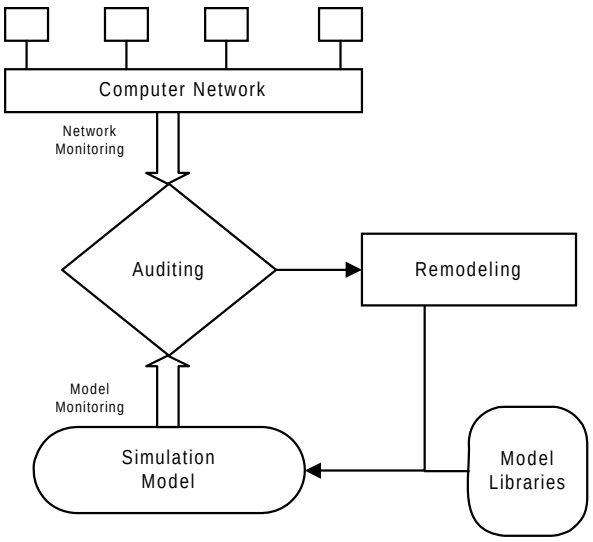

Figure 4: Simulation environment architecture

Model monitoring is performed by an individual module developed in Modsim III so that it remains as close as possible to model implementation. Network monitoring is accomplished through a domain-oriented tool, constructed on the basis of the *etherfind* utility. Network monitoring and auditing tools are implemented in C. Remodeling tool is also developed as part of the overall modeling environment to enhance the degree of interaction with the simulation model.

6 RESULTS AND CONCLUSIONS

The objective of real time simulation is to extend the management capabilities over the Ethernet segments. The selection of 10Base segments as the application domain of this experiment enabled reaching conclusions for the network performance in the near future. Results from two independent experiments are used to describe the experimentation process, express the potential of the real time approach and point out the dynamic network behavior that simulation is dealing with.

Since the model is constantly depicting the current network state, dynamic model modifications are caused upon activation or shutting down of processing nodes as well as the initiation or termination of applications. New model components, as applications and nodes, are integrated within the model after appropriate initialization, while other components, as no longer active applications, are removed. Due to the modeling schema introduced, termination of the experimentation process is avoided when modifying the model.

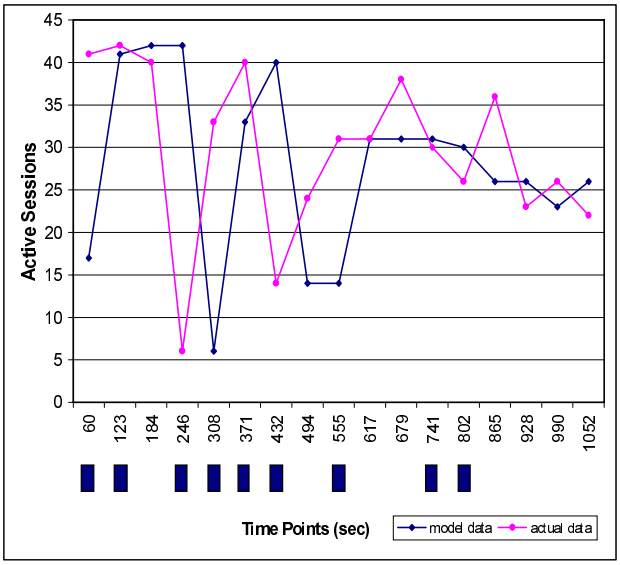

Figure 5: Variation of active sessions

An example of the experimentation process is depicted in figures 5 and 6.

The throughput and number of active sessions in the model and the actual network are plotted for 17 consecutive monitoring intervals of 60 seconds. When deviations are indicated, the model is appropriately modified, as depicted with the marks below the horizontal axon.

It is obvious that simulation results cannot be used when the model is constantly subjected to modifications. Results thus contribute to reaching reliable conclusions only when both systems are moving close enough for a considerable amount of time, as for example at time point 1052 in figure 5. This is also because the network behavior is extremely dynamic (e.g. in time points 184 and 308, there are 40 and 33 active sessions, respectively, while during the intermediate interval only 7 active sessions seem to exchange data).

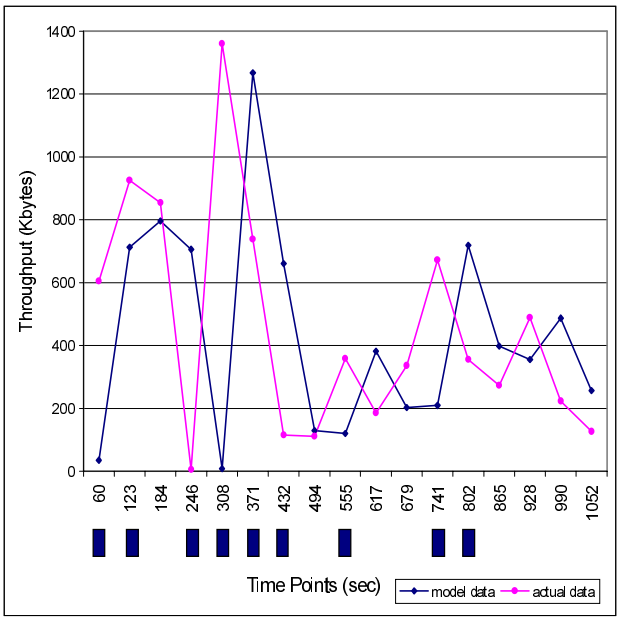

Figure 6: Variation of throughput

The time overhead for examining whether both systems are evolving towards the same direction (auditing) and the - occasional - execution of remodeling is constantly less than 2.5 seconds, which is considered to be acceptable compared to the 60-sec monitoring period.

The above results are obtained with a relatively high deviation interval (high values of *D*). If a closer representation of the network state was required, deviation parameters could be reduced near to 0.3. Results from a second experiment are depicted in figure 7, also for 17 consecutive monitoring intervals.

During the first 10 monitoring intervals, throughput is much smoother compared to the first experiment, thus enabling the model to remain close to the network state. However, due to the low values of the corresponding deviation parameter, the model is almost always subjected to modifications.

The experiment has thus failed, since simulation results are discarded as unreliable, even though the model is relatively close to the real conditions.

This experiment shows that indicating deviations between the network and the model is not a straightforward process and that techniques, as use of aggregate variables (e.g. total number of bytes) must be considered instead of imposing low values for deviation parameters.

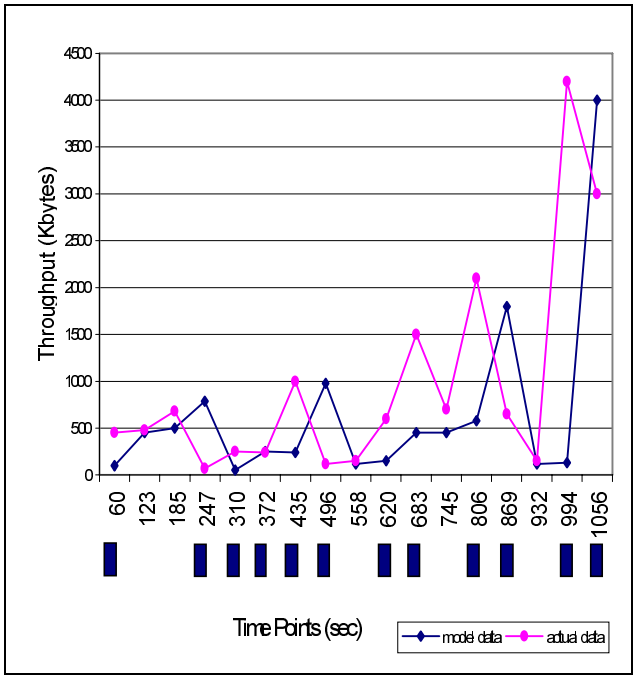

Figure 7: Variation of throughput (2nd experiment)

Analysis of simulation experiment results, as the ones included in this paper, pointed out the following:

1. Network models manipulation was performed during the experiment with minimum time overhead and no recompilation. Simulation results can only be used when considered to be reliable. Appropriate customization and parameterization of the environment is thus critical and may lead to either a success or even total failure of the simulation experiment.

2. Faster than real time network simulation was feasible, since the model actually overcame the network operation speed and provided predictions for the network performance in the near future. Feasibility was one of the primary concerns, when considering this application in the computer network domain. The selection of a relatively slow -10Base- network contributed towards this objective.

   In the worst case, simulation results predicted how the network would perform for at least one interval ahead of real world time, no matter whether predictions were considered to be reliable.

3. Despite the feasibility of this study, the authors consider obtaining real time results from faster networks rather doubtful, since model execution has to overcome transmission rates close to Gbps.

4. The behavior exhibited by computer networks is inherently extremely dynamic. Real time simulation contributes to indicating possible network failures only when these occur as the result of gradient processes and not of instantaneous phenomena.

The overall potential of real time simulation is considered as significant since it enables reaching conclusions for the network performance on the basis of real network data. Nevertheless, taking advantage of these results and exercising administrative authority on the network is not discussed in this paper, viewed as a potential task for network management systems.

REFERENCES

Anagnostopoulos D., P. Georgiadis, M. Nikolaidou, "Applying Real Time Simulation in the LAN Domain: Modeling Considerations", *Systems Analysis Modeling and Simulation*, volume 18, 1995

Anagnostopoulos D., P. Georgiadis, G. Gyftodimos, G. Doukidis, "Establishing the Real time Simulation Concept through a Four-Phase Methodology", In *Proceedings of ESM'95*, Prague, Czech Republic, May 1995

Anagnostopoulos D., P. Georgiadis, M. Nikolaidou, "Network Simulation: A Real Time Simulation Methodology", in *Proceedings of SCSC'95*, Ottawa, July 1995

CACI Products, *COMNET III Reference Manual*, San Diego, 1997

Cramer J., J. Magee, “Configuring Distributed Systems”, In *Proceedings of 5th ACM Workshop on Models and Systems for Distributed Systems Structuring*, Mont St. Michel, September 1992

Jones S., C. Smythe, “A Generic Framework for the Simulation Analysis of Protocol Layered Communication Systems”, In *Proceedings of MASCOTS’93*, Simulation Series, vol. 25, no. 1

Law A., McComas M., “Simulation Software for Communication Networks: The State of the Art”, IEEE Communications Magazine, March 1994

Lee K., P.A. Fishwick, "OOPM: An Object-Oriented Multimodeling And Simulation Application Framework", *Simulation*, vol. 70, no. 6, 1998

Mil3 Inc, *Opnet Modeler Modeling Manual*, Washington, 1997

Zeigler B.P., "*Object-Oriented Simulation With Hierarchical, Modular Models*", copyright by Author, 1995 (originally published by Academic Press, 1990)