RETINA: REal-TIme Network Analyzer *

HaiFeng Zhu[1] Jeffery P. Hansen[3] John P. Lehoczky[2] Ragunathan Rajkumar[1]
Carnegie Mellon University
Pittsburgh, PA 15213

Abstract

In this paper, we present an analytical tool called RETINA for analyzing real-time constraints in complex networks with stochastic traffic. RETINA is based on a statistical method called Real-Time Queueing Theory and can predict the fraction of late tasks in networks of queues given traffic conditions and queue protocol specifications. Upon submitting a profile of the network structure and data flow descriptions, the user is able to query certain performance metrics. Currently RETINA supports two types of real-time scheduling algorithms (EDF and FIFO) and three types of performance queries: the deadline miss rate of a designated flow, the minimum feasible deadline for a particular flow, and whether all QoS requirements of all flows will be met when a new flow is added. Simulation results show excellent agreement with the results predicted by our tool. RETINA has made it possible to analyze larger and more realistic networks and can be used to address network design tradeoffs. This use of RETINA to identify and quantify network bottlenecks is presented in the paper.

1 Introduction

As multimedia traffic ranging from VOIP to high-resolution video consumes an increasing fraction of available network bandwidth, the development of network strategies to meet the needs of such traffic is becoming more critical [1, 3, 10]. Multimedia traffic is generally characterized as having both large bandwidth requirements and end-to-end timeliness requirements. While the end-to-end timeliness requirements are critical to achieving acceptable response times for applications such as video conferencing, in most cases a small portion of the traffic can miss its deadlines without having a significant adverse effect on application quality. While much is known about real-time scheduling of periodic tasks with hard deadlines, new techniques are needed to deal effectively with the combination of soft real-time tasks and stochastic workloads.

Existing analytic tools for real-time systems mostly focus on the hard deadline case in which all deadlines must be met. This approach requires a worst case analysis that can result in extremely low average case utilization when the system must handle tasks with stochastic characteristics. We have developed a theoretical framework to handle networks carrying stochastic traffic. The methodology is presented in [11]. Using this methodology, we are able to decompose a complex network into a simple pipeline (tandem) network with cross-traffic nodes, which, in turn, can be further reduced to a pipeline with bottleneck nodes. The paper by Zhu et. al. [11] presented the theoretical foundation and the sequence of steps that is needed to decompose a complicated network into a simple pipeline structure. RETINA, the tool presented in this paper, automates this process.

As a simple example, suppose the network shown in Figure 1 has two flows. Flow 1 goes from node A to node B passing through a slow node R1, two fast nodes R2 and R3, and then through a slow node, R4. Flow 2 goes from C to D cutting through R4. Assume that R2 and R3 are very high speed with small delay. Either or both of the nodes R1 and R4 will be bottleneck nodes. Given the network capacity and the deadlines of the flows, we want to determine the fraction of pack-

*This research is supported in part by ONR/DARPA MURI under contract N00014-01-1-0576.

1. Department of Electrical and Computer Engineering, *{haifengz,raj}@ece.cmu.edu*.
2. Department of Statistics *jpl@stat.cmu.edu.*,
3. Institute for Complex Engineered Systems *hansen@cmu.edu.*

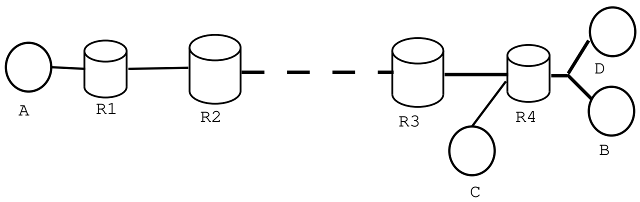

Figure 1. A real-time networked system

ets from each flow that will be late, i.e. the "Deadline Miss Rate" of each flow. Alternatively, we might specify the maximum fraction of packets that can be late, and compute the *Minimum Feasible Deadline*, the minimum end-to-end deadline that satisfies the deadline miss rate bound. Finally, we can improve performance by upgrading some of the links. The analysis can be used to determine which links should be upgraded and what increase in performance will result from the upgrade.

Such analysis is very useful for designing a real-time network. However, when the network grows larger, the complexity increases and hand computation is no longer reasonable. RETINA was designed to automate these computations and can also be used to discover new effects in networks.

In RETINA, the user is able to define the profiles of the network structure and the input task flows. There are seven parameters associated with each task flow. interarrival times (mean and variance), service (execution) times (mean and variance), end-to-end deadline, route and QoS (allowable deadline miss rate).

Once the network structure and the profiles of each of the flows has been specified, the tool can be used to query certain performance metrics. Currently RETINA supports two types of real-time scheduling algorithms (EDF and FIFO) and three types of performance queries. First, for the given specification one can query the deadline miss rate of any flow and check whether it meet its QoS specification. Second, one can determine the minimum feasible deadline for a certain flow, i.e. the minimum deadline this flow must have to ensure that it will meet its QoS specification, holding the other parameters constant. Third, assuming that all QoS specifications are currently being met for a set of flows, one can add an additional flow and determine whether the resulting set of flows will continue to meet their QoS requirements. The tool is interactive in that one can adjust the parameter values and determine the resulting performance characteristics. Simulation results show excellent agreement with the results predicted by RETINA, thus making it a very useful tool for network engineering design.

The rest of the paper is organized as follows. Section 2 briefly describes the underlying methodology that is implemented by RETINA. Section 3 describes the RETINA usage model. Section 4 illustrates a series of experiments performed using RETINA and compares them with simulation results. Finally, Section 5 presents our conclusions.

2 RETINA: Underlying Methodology

RETINA is designed to handle general networks with N nodes carrying a set of K independent acyclic soft real-time flows (i.e. flow routes include any station at most once along their source to destination path). We want to determine if, for a given network and set of flows, any particular flow will meet its quality-of-service requirement, here taken to be the fraction of tasks (packets) in a flow that can miss their deadline. Even though the tasks' routes can be very different, we can consider any flow to have a route that is a pipeline with cross-traffic. Consequently, if we can determine the fraction of late tasks for a basic pipeline with cross-traffic, we can determine the whether or not any particular flow will attain its desired QoS. Moreover, we can determine the minimum feasible deadline for the flow for which it will attain its desired QoS. Finally, we can determine if adding one more flow will cause any of the existing flows to fail to meet their desired QoS. In this section we provide a brief intuitive discussion of the calculations underlying RETINA. The mathematical foundations are given by Kruk et. al. [5], and the specific formulas used in RETINA are contained in Zhu et. al. [11].

We begin the discussion for the single node case, then extend to the general pipeline with cross-traffic. Suppose there are K independent flows passing through this single node. Each flow consists of a sequence of tasks or packets. We assume that the arrival of packets forms a renewal process, namely the interarrival times are independent random variables whose common mean and variance are specified. For periodic flows, the variance is 0. Each task at each node is assumed to have a computation requirement, the mean and variance of which must be specified. Each flow contributes to the overall traffic intensity at each node, and the aggregation of the flows passing through a node creates a workload for that node. For the light to moderate traffic intensity case, there is a very small queueing delay, and the total delay at the node is dominated by the task computation times. In the heavy traffic case, the queueing delay is substantial and dominates the computation time. One can develop an asymptotic analysis of the workload process, modeling it as a Brownian motion process with reflection at the origin (since workloads cannot be negative). The equilibrium distribution of the workload process is given by an exponential distribution with some mean value that we denote by $1/\theta$. The means and variances of the flows passing through the node can be combined to determine the parameter θ. If a flow with deadline D passed only through a single node, then Real-Time Queueing Theory (RTQT) can be used to determine the fraction of tasks from that flow that would miss their deadline, and it would be given by $e^{-\theta D}$, see Kruk et. al. [5].

We next turn to the case of a basic pipeline with cross-traffic. We can again reduce consideration to the nodes that have heavy traffic as the nodes with light or moderate traffic will contribute little more than the task service times to the end-to-end delay. Suppose that the pipeline consists of N nodes, each in heavy traffic. Just as in the single node case, we can model the workload process at a node by a Brownian motion. This process will have an exponential equilibrium distribution having a mean value $1/\theta_i$, $1 \leq i \leq N$, where θ_i is computed from the means and variances of the interarrival times and the service times of the tasks passing through node i. Consider the flow through this pipeline, and assume its end-to-end deadline is D. RTQT indicates that the fraction of tasks in this flow that will not meet their deadline is given by $P(W_1 + \cdots + W_N > D)$ where W_i is a random variable representing the equilibrium workload at node i. We assume that the random variables $W_1, \cdots, W_N$ are independent with exponential(θ_i) distribution, $1 \leq i \leq N$. It follows that the random variable $W_1 + \cdots W_N$ has a generalized gamma distribution with parameters $(\theta_1, \cdots, \theta_N)$. The required probability can be calculated exactly, using closed-form expressions when the θs are equal or when N is small. One can easily compute the Fourier transform of the sum and use a fast-Fourier inversion formula to compute the probability. Finally, if several of the θs are significantly smaller than the rest of the θs, then those correspond to bottleneck nodes, and the non-bottleneck nodes can be ignored, see Zhu et. al. [11]. This can be done using a closed-form expression for the small N case. These techniques are the ones used in the calculation module of RETINA.

3 RETINA Usage

RETINA has two interfaces: a text interface and a graphical user interface (GUI).

3.1 Text Interface

The text interface is useful for using retina from a customized script. It takes two input files, a network profile and a query command file. The profile is used to describe the network structure and task flow characteristics. The query file is used to query a particular performance metric. Currently, the tool supports three types of queries: deadline miss rate, minimum feasible deadline and admissions control for all the flows injected into the network. A simple language was defined to describe the network structure, task flows and queries.

A network profile consists of a set of node declarations, a set of link declarations and a set of flow declarations. Each link is connected to exactly one output port on one node and exactly one input port on another node. Each output port is considered to be a queue with a default queue discipline of EDF. It is also possible to specify an internal processing delay for nodes.

Link declarations specify an output port, a destination node and a link bandwidth in Mbps. Links are considered point-to-point one way connections. This allows us to model connections in which the upstream and downstream link bandwidth can differ.

Flow declarations specify a packet interarrival time distribution, a service time distribution, a deadline distribution, a statistical lateness constraint and a path. The interarrival time, service time and deadline distributions can be either constant, uniform, normal, nonnegative normal, exponential or shifted exponential.

3.2 Graphical Interface

The graphical interface is implemented as a separate program which transparently invokes the textual-based

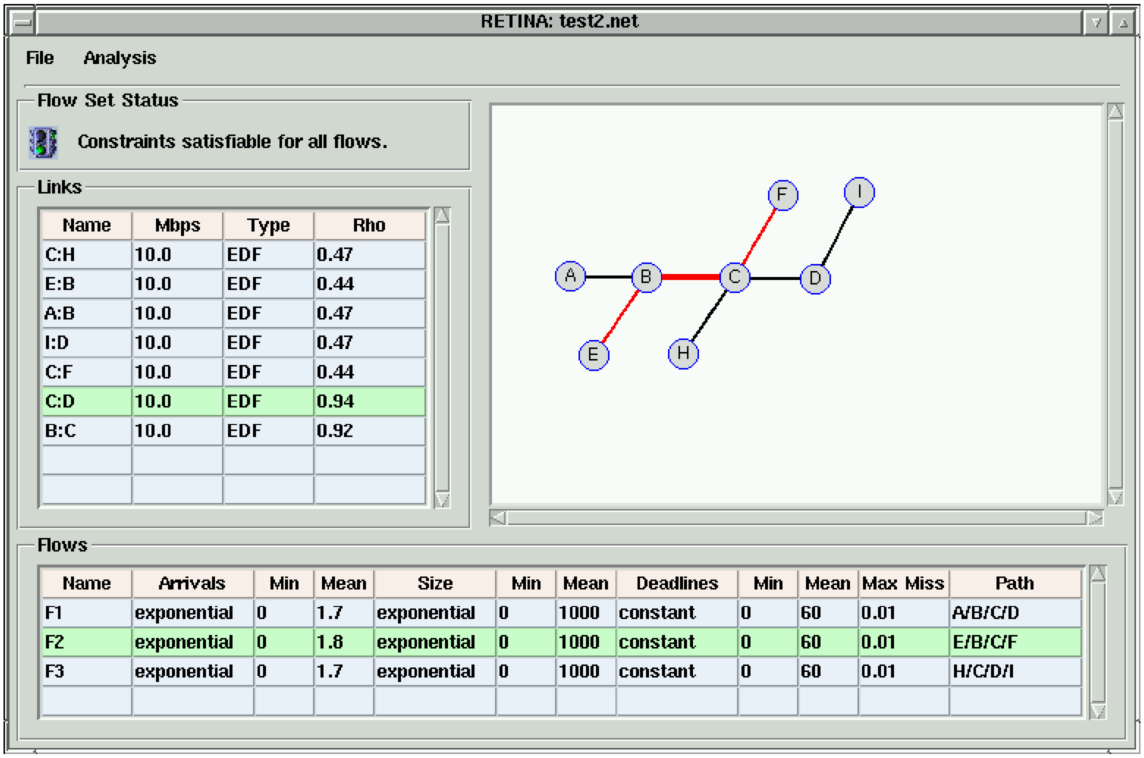

Figure 2. RETINA Main Window

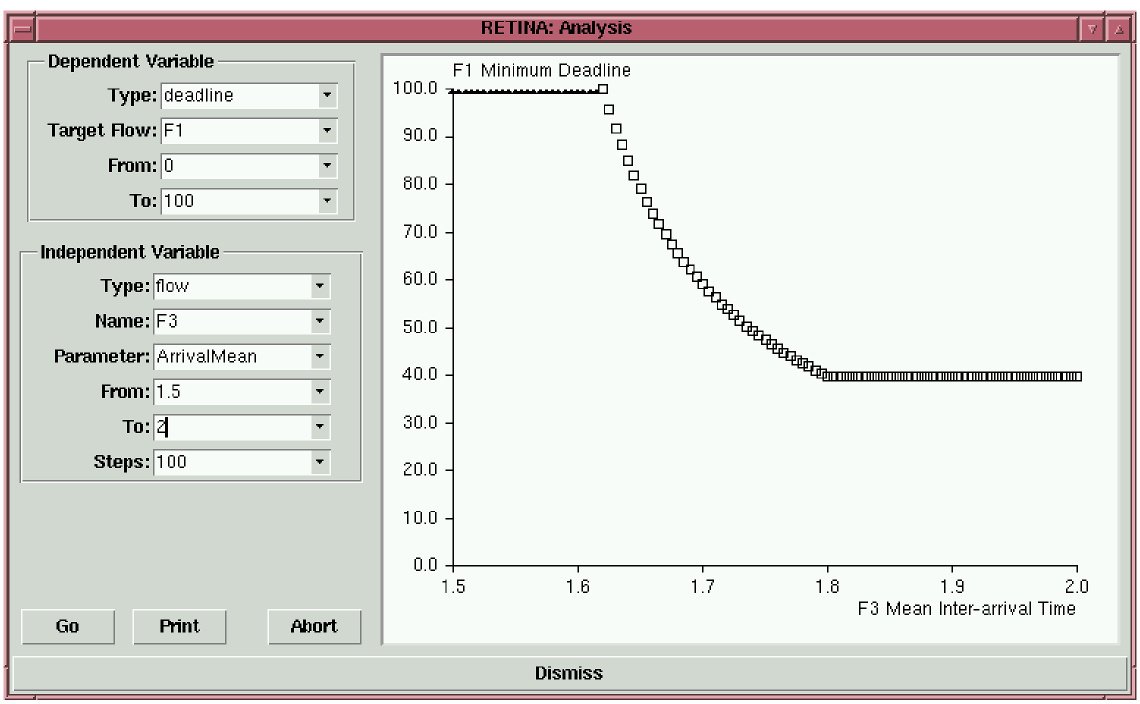

Figure 3. RETINA Analysis Window

tool and displays results. The main window is shown in Figure 2. The network shown in the screenshot has a "primary" flow from node A to node D, and two cross-traffic flows, one from node E to node F and one from Node H to node I. This results in two bottlenecks, one on the link from B to C and one on the link from C to D. Using the interface, one can create or modify a network and use the spreadsheet areas to set link and flow properties. Any time a parameter value is changed, the feasibility of the flow set is dynamically evaluated and displayed in the upper left with either a green light to indicate a feasible flow set or a red light to indicate an infeasible flow set. In addition, flow properties such as the traffic intensity at each link is also dynamically updated.

The primary output from the tool is derived from an additional analysis window (Figure 3). Multiple analysis windows can be opened so as to compare the results from multiple experiments. The analysis window has areas in which to set the dependent and independent variables for the experiment and an area to show the results. In the dependent variable (y-axis) selection area we can choose to show either the miss rate, or the minimum feasible deadline for the selected flow. In the independent variable selection area we can choose to plot the change in the dependent variable as either the specified flow or link parameter changes. For flows we can choose from among the interarrival mean, the packet size mean, the deadline mean or the required miss rate.

In the example shown in Figure 3, we have plotted the minimum feasible deadline for Flow F1 as the mean interarrival time between packets for the cross traffic Flow F3 changes under the conditions shown in Figure 2. In this plot we see that when the interarrival time for Flow F3 packets is below about 1.62ms, the minimum feasible deadline for Flow F1 goes to infinity due to the fact that the traffic intensity on the C-D link goes above one. We also see that as the Flow F3 interarrival time goes above about 1.7ms, the Flow F1 minimum deadline levels out at about 58ms. This flattening of the curve is a result of the bottleneck effect when the bottleneck link shifts from C-D to B-C as the dominating cross traffic flow shifts from F3 to F2.

We can change the point at which the bottleneck crossover occurs by changing the mean interarrival time for Flow F2. Figure 4 shows the results of the exact same experiment as was presented in Figure 3 except that the Flow F2 mean interarrival time was increased from 1.7ms to 1.8ms. This results in the bottleneck crossover occurring at a higher Flow F3 mean interarrival time than in the Figure 3 example.

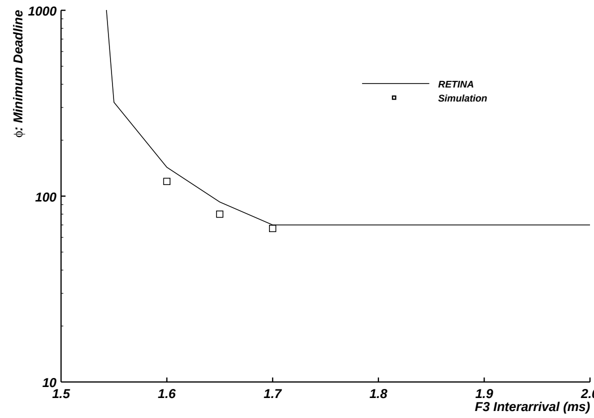

Figure 4. Minimum Feasible Deadline with Multiple Cross-Traffic Flows

4 Experiments

In the following section we will perform a series of simulation experiments to demonstrate the validity of the RETINA computed curves. We built a discrete event simulator similar to ns-2 (network simulator) in which packets are generated at a source nodes and forwarded through the network. At the sink node, we calculate the deadline miss rate as the number of packets that missed their deadlines divided by the total number of packets that were sent. We then compared this experimental miss rate with the RETINA predictions. Unless otherwise stated, figures throughout this paper use symbols to represent experimental results and lines to represent RETINA generated theoretical results. Also, unless otherwise stated, for each simulation data point we performed five runs of five million events each with differing random seed values. The plotted points represents the average of the five runs and the confidence bars represent a 90% confidence interval. A Student T distribution was used to compute the confidence interval from the five sample runs. In comparing theoretical and experimental results, we consider a 1/3 order of magnitude difference between simulation and theory to be good for miss rates below about 0.01, and a 1/6 order of magnitude difference good for miss rates

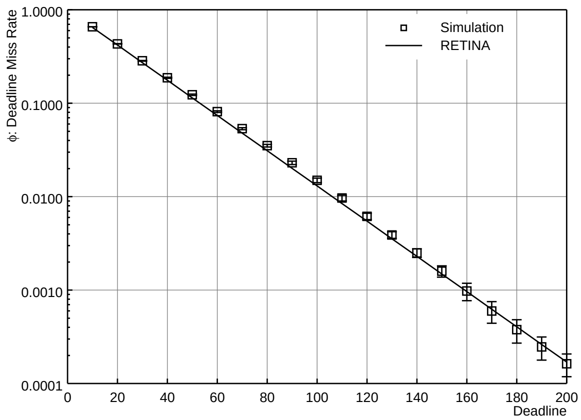

Figure 5. EDF Single Node Single Flow

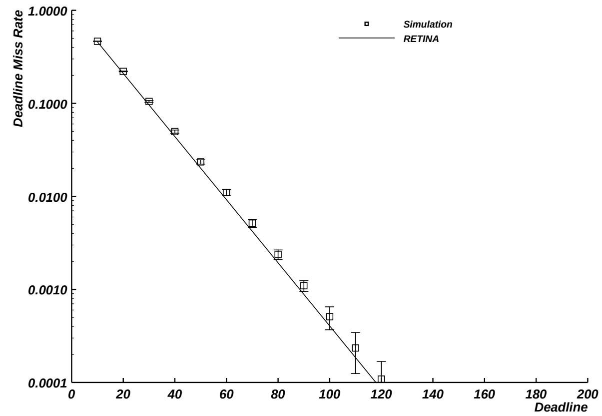

Figure 6. EDF Single Node Multiple Flows

above 0.01.

4.1 Single Node

In our first experiment we test the miss rate prediction made by RETINA. We assume a 1Mbps link, packet sizes that are exponentially distributed with a mean of 1200 bytes and a packet interarrivals that are exponentially distributed with a mean of 10ms. A comparison of the miss rate predicted by RETINA and the simulation results as a function of the deadline (assuming constant deadlines) are shown in Figure 5. As can be seen, there is a very close match between the prediction and the measured result.

A second single-node miss rate experiment with two flows was also performed. In this experiment we have two flows with exponentially distributed packet sizes with means of 400 bytes and 800 bytes respectively. The packet interarrival times are exponentially distributed with a mean of 10ms. The results of this experiment are shown in Figure 6. Here we show the miss rate of Flow 1 as we vary the deadline of Flow 1. Both flows have the deadline of 10 ms. Again we see a good match between the RETINA prediction and the simulation results.

4.2 Network

A comparison of RETINA with simulation results in a two node pipeline network is shown in Figure 7. In this experiment we assume exponentially distributed interarrival times with a mean of 10ms and exponentially distributed packet sizes with a of 1187 bytes, but

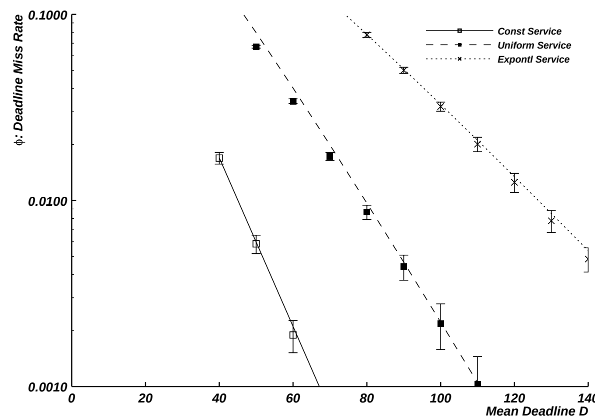

Figure 7. Basic Pipeline Network

look at three different distributions for the packet size: constant, uniform and exponential. As expected, the higher the packet size variance the higher the miss rate even when the mean is identical [11]. We see that RETINA accurately predicted this phenomenon.

Figure 8 presents the results of a second network experiment to show the effect of traffic intensity in a two node pipeline in which one node is made a bottleneck node by slightly reducing its link bandwidth. We assume packet sizes are exponentially distributed with mean 1125 bytes and packet interarrival times are exponentially distributed with a mean of 10ms. We assume the non-bottleneck link has a bandwidth of 1Mbps and perform experiments for bottleneck link bandwidths of 0.928 Mbps, 0.918 Mbps and 0.909 Mbps as shown in the figure. Since one of the nodes is a bottleneck, RETINA applies the bottleneck node effect to elimi-

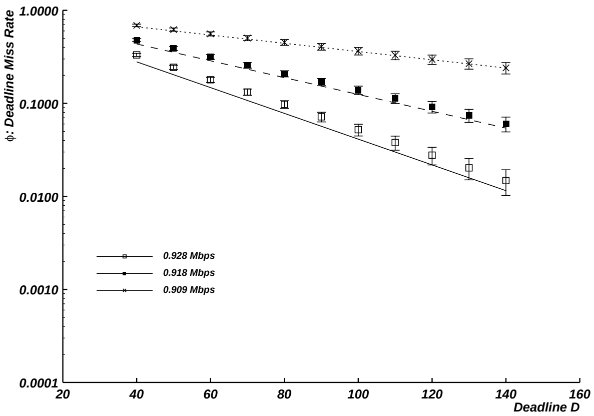

Figure 8. Bottleneck Pipeline Network

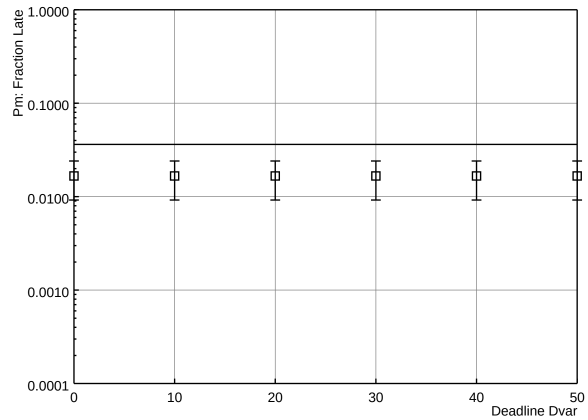

Figure 9. Cross-Traffic Pipeline Network

nate all but the bottleneck node before computing a miss rate. Despite the simplification we still see that the RETINA predictions are close to the simulation result for a variety of traffic intensities.

Unlike variance in the service time, variance in the deadline distribution did not significantly affect the miss rate[11]. This can be seen in the results from the experiment shown in Figure 9. This figure shows the miss rate as a function of the deadline variance in a three node pipeline network in which the first node carries cross traffic. Both the primary flow and the cross traffic flow have exponentially distributed packet interarrivals with a mean of 10ms and exponentially distributed packet sizes with a mean of 1200 bytes. The link bandwidth on the bottleneck node was set to 2Mbps, and bandwidth on the other links was set to 1Mbps. Deadlines are uniformly distributed with a mean of 135ms. We can see that the curve is flat indicating that deadline variance does not affect the miss rate. The same result was obtained when we moved the cross traffic to either of the other two nodes. We also note that the RETINA prediction (solid line) is reasonably close the simulation results.

Figure 10 presents results for an experiment performed with the network setup shown in Figure 1. In this experiment, the output port of R1 has a bandwidth of 0.929 Mbps, the output port of R4 has a bandwidth of 2.751 Mbps and all other links have a bandwidth of 1 Mbps. The primary flow from A to B has exponentially distributed packet interarrivals with a mean of 1ms, and exponentially distributed packet sizes with a mean of 105 bytes. The cross traffic flow from C to D also has exponentially distributed packet interarrivals with a mean of 1ms, but exponentially distributed packet sizes with a mean of 620 bytes. The theoretical and simulation miss rates are plotted as a function of the primary flow end-to-end deadline D. The simulation results represent five runs of twenty million events. Notice the excellent agreement between the simulation and theoretical approximation. Consequently, this simple example shows how we can use RETINA to analyze and design networks of this type to meet real-time specifications.

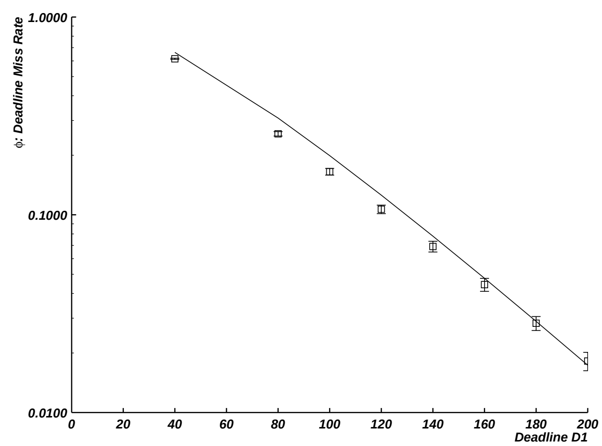

Figure 10. Experimental and Theoretical Comparison in Example Network

5 Conclusion

In this paper, we presented an analytical tool called RETINA for analyzing timeliness under stochastic traffic in complex networks. RETINA includes capabilities for predicting deadline miss rates, minimum feasible deadlines and conducting admission control tests. The tool was verified using simulation, and we obtained close agreement between the RETINA predictions and the simulation results. By applying RETINA we have been able to analyze larger networks than previously practical through hand analysis and gained new insights on other network effects. RETINA will also act as a framework for future research on additional issues such as the impact of path selection and QoS optimization on end-to-end timeliness.

References

[1] Chandra, P., Fisher, A., Kosak, C., Ng, T.S.E., Steenkiste, P., Takahashi, E., Zhang, H., "Darwin: Resource Management for Value-Added Customizable Network Service", From *6th IEEE International Conference on Network Protocols (ICNP'98)*, October 1998.

[2] Doytchinov, B., Lehoczky, J.P. and Shreve, S.,"Real-time queues in heavy traffic with earliest deadline first queue discipline," *Ann. of Appl. Prob.*, **11**, 2001, 332-378.

[3] Hoover,C.L., Hansen, J.P., Koopman, P., Tamboli, S., "The Amaranth Framework: Policy-Based Quality of Service Management for High-Assurance Computing", *Int. Jour. of Reliability, Quality, and Safety Engineering, 2001.*

[4] Kruk, L., Lehoczky, J., Shreve, S., and Shu-Ngai Yeung, "multiple-input heavy-traffic real-time queues," *Annals of Applied Probability*, **13(1)**, 2003, 54-99.

[5] Kruk, L., Lehoczky, J., Shreve, S., and Shu-Ngai Yeung, "Earliest-deadline-first service in heavy-traffic acyclic networks," *Annals of Applied Probability*, **14**, 2004, 1306-1352.

[6] Lehoczky,J.P., "Real-Time Queueing Theory", In *Proc. Real-Time Sys. Symp.*, pg. 186-195, Dec. 1996

[7] Lehoczky, J.P., "Scheduling communication networks carrying real-time traffic", In *Proc. Real-Time Sys. Symp.*, Dec. 1998, 470-479.

[8] Peterson, W.P., "A heavy traffic limit theorem for networks of queues with multiple customer types," *Math. of Operations Research*, **16**, 1991, 90-118.

[9] Yeung, S-N., Lehoczky,J., "End-to-end Delay Analysis for Real-Time Distributed Networks" , to *Proc. Real-Time Sys. Symp.,* London, 2001

[10] Zhang, L., Deering, S., Estrin, D.,Shenker, S., Zappala, D., "RSVP: A New Resource ReSerVation Protocol", From *IEEE Network*, vol. 7, no. 5, p. 8-18, September 1993.

[11] Zhu, HF., Lehoczky, J.P., Hansen, J.P., Rajkumar, R.," Design Trade-Offs for Networks with Soft End-to-end Timing Constraints", *Proc. Real-Time and Embedded Tech. and Appl. Symp. (RTAS)*, Toronto, May 2004.