Network Management and Control Using Collaborative On-line Simulation

Authors: Tao Ye, Shivkumar Kalyanaraman, Biplab Sikdar, Bin Mo,
Hema Tahilramani, Ken Vastola and Boleslaw Szymanski

Contact: Shivkumar Kalyanaraman
Department of Electrial, Computer and System Engineering
Rensselaer Polytechnic Institute
Troy, New York 12180
Phone: 518-276-8979 Fax: 518-276-2433
Email: shivkumaecse.rpi.edu

Abstract

The complex, dynamic feature of the current Internet requires a scalable, effective network control. This paper proposes a dynamic network control scheme, which performs the parameter tuning of the underlying network algorithm based on collaborative on-line simulation, thus achieves second order control on the network. We have used *ns* as the simulation platform and introduced some methods to tackle its drawbacks in traffic generation. To efficiently search the concerned parameter space for "good" points, we propose an increasingly-improving search procedure and a hybrid search algorithm. We also present some methods to speed up the network simulation. In particular, we find that topology decomposition can achieve very high speed-up if decoupling the execution of different decomposed parts. In addition to the network control, the on-line simulation scheme can also be used in many other areas. The applicability of this scheme to routing algorithms is investigated in this paper. Extensive simulation and experiments have been carried out for the proof of concepts.

Submitted to: Global Internet2000, General Conference Topics–Network Operations & Management

1 Introduction

Network management and control has been the area receiving considerable attention recently. A large variety of algorithms and schemes have been proposed to improve the network performance (RED[2], ERICA[3]), provide better-than-best-effort services (CBQ[4], differentiated-service[5]), or perform other control functions. These algorithms operate directly over the network architecture and produce immediate effects on the network. However, their effectiveness is mostly dependent on the interaction of their parameter setting and specific network conditions, and the canonical parameter setting may not be the best under current network conditions. There is hence opportunity for new, scalable approaches to network management and control, which can dynamically adapt to current network conditions.

We propose in this paper a collaborative on-line simulation scheme, which uses cooperative simulation to find out the suitable parameters for the underlying network algorithms and attempts to perform the dynamic network management by applying second-order control over algorithm operation. By second-order control, we mean that on-line simulation merely prescribes parameters required for the operation of algorithms and does not interfere with their normal operation in any other way. A presumption of our work is that since we do not understand how to build stationary models of Internet traffic patterns, a reasonable approximation of "current" traffic behavior could be characterized by quasi-stationary models, and such models can be constructed on-line. As the nature of these quasi-stationary traffic models changes over time, the algorithm performance could be improved by an alternative choice of algorithm parameters. The purpose of on-line simulation is therefore to derive the appropriate quasi-stationary traffic model, and given an approximation of the underlying topology and algorithm implementations, search the parameter space to determine "good" parameter settings which suit the current conditions. Observe that the emphasis is not on "full" optimization, but on a best-effort strategy to move the system towards a "better" operating point. In other words, one of the major benefits of on-line simulation would be to continuously tune underlying operation (albeit at a larger time-scale than their normal operation). And in this sense, on-line simulation equips the network management infrastructure with "proactive" management capabilities. Each simulator is simple because it represents a local, parameterized view of the network, and hence can execute quickly. And these simulators exchange information and cooperate with each other to optimize the network usage from a broader point of view.

In section 2, we describe the overall architecture of collaborative on-line simulation scheme. Section 3 introduces our work in traffic generation of on-line modeling. Section 4 describes our approach to the efficient parameter search. Section 5 discusses two ways to speed up the process of the network simulation. In particular, our approach for topology decomposition shows a high speed-up in large-scale simulations. Section 6 and section 7 present the simulation results under *ns* and the validating experiment results under Linux. Section 8 discusses the applicability of on-line simulation to routing algorithms. Section 9 concludes this paper and points out areas for further research.

2 Infrastructure of Collaborative On-line Simulation

Our collaborative on-line simulation scheme introduces a simulator into each local network domain.An on-line simulator incorporates such function modules: *data collection and on-line modeling*, *experiment design*, *management interface* and *experiment execution*. Figure 1 shows the structure of an on-line simulator.

Data Collection and Online Modeling continually collects all kinds of information about the local network topology, traffic conditions, etc., and tries to build an up-to-date network model for use in simulation according to this information.

Management Interface is the control center of the on-line simulator. It controls and synchronizes the operation of all the other modules. Meanwhile, it is also the interface of the on-line simulator to the outside.

Experiment Designer is responsible for setting up simulation experiments with appropriate search techniques, and then *experiment execution* will execute these experiments and collect the results. *Experiment designer* in turn analyzes these results to perform further searches and try to find "good" parameter settings.

The on-line simulator also communicates with other on-line simulators and exchanges the relevant network information, such as network traffic models, good algorithm settings. Thus, a collaborative on-line simulation network can be formed. Through this, the local simulator can have a global view of the network and perform a better network simulation. Another advantage is that collaborative network control can be easily applied to this simulation network, therefore greatly improve the effectiveness of the resource management.

As described above, our scheme is based on network simulation; and consequently, an efficient and reliable simulation platform is the prerequisite of this scheme. We have chosen the network simulator *ns* [1] developed by UCB/LBNL and the VINT project as our simulation platform because of its wide acceptance in the networking community and its open design suitable for modification.

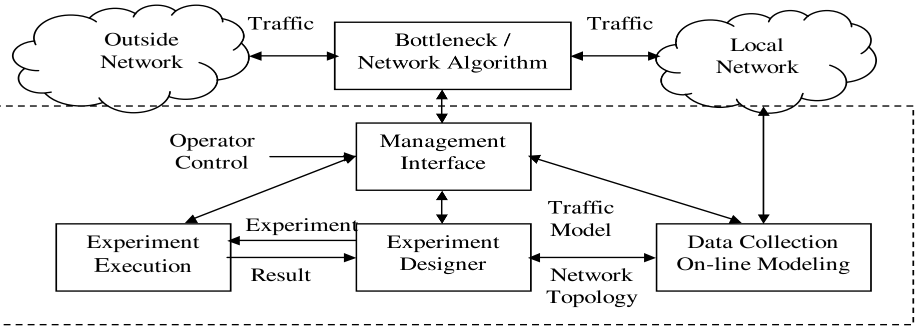


Figure 1: Structure of on-line simulator

3 On-line Modeling: Workload Generation

On-line modeling is greatly complicated by the complexity and heterogeneity of the Internet. At this time, we are mainly concerned about the proof of the on-line simulation concept. To avoid overly complicating the problem, currently our effort in this area focuses on generating the workload consistent with the present study on Internet traffic.

The first issue we are going to address is maintaining the proper traffic composition in the simulation, which is essential to capturing the behavior of wide area networks. For the Internet, the dominating applications are WWW, Telnet, FTP, SMTP, and NNTP and we use the empirical distributions in [7] to characterize the underlying protocols for these applications. We implement these application-specific traffic generator under *ns* and deploy some measures to maintain the proportion of these traffics according to the empirical distribution [6] in the simulation.

Another important issue with traffic generation is self-similarity in wide area and Ethernet traffic[7]. Generation of aggregate traffic which is self-similar in nature is of utmost importance in simulation scenarios as Poisson models grossly underestimate the queuing delays and overflow probabilities. We have implemented in ns two self-similar traffic sources: Application/Traffic/SupFRP and Application/Traffic/SS, respectively based on the algorithms proposed in [8] and [9].

All the protocol specific, application specific and self-similar traffic generators described above have been validated through extensive simulation and experimentation. The detailed results and analyses are presented in another paper[10].

4 Experiment Design

The goal of the experiment design is to use as few experiments as possible to find as good a setting as possible. Note that here the emphasis is not put on seeking the optimum setting; instead a best-effort strategy is used to find a better point within the limited time frame. Based on this principle, it would be desirable for the experiment design to execute an iterative search in the parameter space, that is, the search result is increasingly improved step by step. Thus, it is possible to make a compromise between the result goodness and the search time, whereas the non-iterative methods have to wait for finishing all experiments to get the search result.

The basic procedure of our approach is first probing the search space roughly and trying to find the important parameters which have greater effects on the system performance. After pruning part of the search space, we will explore the search space in more detail. We have designed a new hybrid search algorithm to perform the fast and efficient search in the concerned parameter space.

4.1 Roughly Probing of Search Space

As described above, we start our search process with the rough probing of the search space. For this purpose, simulations will be conducted in portions of the parameter space, specifically the boundaries of the space. These simulations will be based upon 2^k full factorial experiment design ideas that are well known in performance analysis[11].

2^k full factorial design tries all possible combinations of the parameter boundaries and fits the results into a linear regression model to analyze the importance of different parameters. It is not a non-iterative method. To achieve our increasingly-improving goal, we carefully order the experiments to form a series of subsets and the first subset is generated by applying 2^{k-p} fractional factorial design on the parameter space. Here, p is the minimum integer satisfying $2^{k-p} \geq k$, which is required by the regression analysis[11]. 2^{k-p} fractional factorial design just tries part of the experiments in 2^k full factorial design. By carefully selecting the experiments, the analysis of the parameter importance can still be executed at the expense of some accuracy. After finishing this subset of experiments, we analyze the simulation results and then step into the next larger subset which is obtained by using 2^{k-p+1}fractional factorial design, and so on until all 2^k experiments are finished. During this process, if the simulator receives an interrupt signal, the simulation is stopped at once, and the analysis result based on the last subset of experiments is returned as the "best so far" result.

4.2 Hybrid Search Algorithm

Once the high level pruning is complete, the next task is to search the remaining parameter subspace in detail with state space search techniques. Basically, the state space search algorithms known in the literature include two important components: *exploration* and *exploitation*, and a balance strategy between these two components. *Exploration* encourages the search process to examine unknown regions. In comparison, *exploitation* attempts to converge to a maximum or minimum in the vicinity of a chosen region.

The "No Free Lunch" theorem[15] cautions us that there is no general most efficient algorithm as measured by both exploration, exploitation and balance. In other words, the best search algorithms would aggressively incorporate the maximum domain-specific and surface-specific features into the search strategy at all times. This necessitates that the search algorithm be extremely flexible and adaptive to different kinds of specialized information.

Our hybrid search algorithm is based on the hillclimbing technique[12] with TABU technique[14] included to avoid revisiting the previous region and speed up the exploration process. In comparison with the others, the hybrid method maintains a dynamic balance between *exploration* and *exploitation* by accepting the bad move with a probability of *exp(-current gradient feature / average gradient feature)*. The idea here is that in the promising areas whose gradient is much steeper than the average gradient, the algorithm mostly performs *exploration*: random walk, and otherwise, *exploitation*: hill-climbing to quickly converge to the local optimum.

The hybrid algorithm also automatically adjusts its step size to suit the current gradient features. When the current gradient is large relative to the average, the step size is reduced so as to explore this promising area carefully. Otherwise, the step size is increased to quickly get through this area. Additionally, with the increase of the step size, the rugged microscopic features of search space are smoothed out and the major features are exhibited. These macroscopic features can be used to speed up the search process.

The hybrid algorithm is very efficient for the search space with relatively regular gradient structure. It combines the advantages of various search techniques, and attempts to apply a dynamic balance strategy in the search. Note that this balance strategy is different from that of Simulated Annealing[13]. We do not apply any concepts of statistical mechanics, such as *heat equilibrium*, and *cooling scheme*, in our method; whereas in SA, *temperature* decreases gradually according to a certain cooling scheme, and for each temperature level, the search is repeated for a number of times to achieve *heat equilibrium*. As a result, SA will take a long time to converge.

5 Speeding Up On-line Simulator

Since in the on-line simulation system, the execution of the experiments is the most time-consuming task, we have designed a few methods to speed up this process.

5.1 Farmer-Worker Infrastructure

We have designed a parallel farmer-worker scheme to distribute many single-machine experiments across a bunch of workstations. In fact, this scheme can be used not only in our on-line simulation, but also in any other research areas which require executing a lot simulation experiments. The basic infrastructure of this scheme is shown in Figure 2.

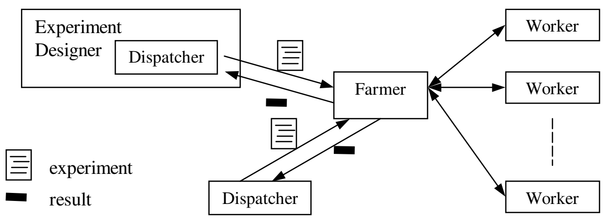


Figure 2: Structure of on-line simulator

In this scheme, the dispatcher is the interface between this distributed simulation performer and the outside simulation designer. All the experiments have to go through this interface to be distributed among the workers. The farmer is the center of this scheme, which controls and synchronizes the operations of dispatchers and workers. The workers are the actual experiment performers. We can use multiple workers to work for the same experiment designer so as to speed up the simulation process. All the communication in this scheme is through TCP connections. Therefore, the dispatchers, farmer and workers can be located anywhere in the network. That means that we can even distribute the experiments over the whole network and maximize the utilization of the computing resources.

The extensive tests have been performed on the current farmer-worker implementation to ensure its effectiveness and robustness.

5.2 Topology Decomposing

After a lot of experiments, we have found that the running-time of a simulation is not linearly proportional to the simulation size. Figure 3 shows the simulation results on a network with a simple star topology. Using

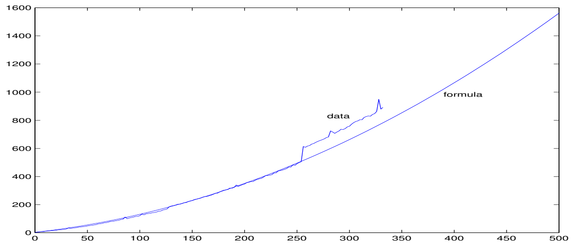


Figure 3: Execution time vs. simulation size

the least squared method to fit the experiment results of execution time and network size, we can get the following approximate formula:

$$T(n) = 3.49 + 0.8174 \times n + 0.0046 \times n^2$$

Here T is the execution time of the simulation, and n is the number of nodes in the simulation. The approximate formula estimation is also shown in Figure 3.

From the above, we can see that the execution time of a network simulation has a quadratic relation with the network size. We can thus presume that it is possible to speed up the network simulation more than linearly by splitting up the simulation into smaller pieces and paralleling the execution of these pieces.

Traditional decomposion only splits up the network topology, but the simulation is still executed as a whole. Therefore, the decomposed parts have to exchange a lot of information to keep them synchronized with each other. Our approach is to first execute these splitted simulations independently, next repeat each of these with the output of the other parts as the input, and then repeat again and again until there is no significance difference between the results of two sequential iteration. Decomposed simulation greatly simplifies synchronizations between parallel parts, and can significantly speed up the simulation of large networks. Some simulations have been done and the results show a very fast convergence.

6 An Experiment with the On-line Collaborative Simulator

To test the validity of our on-line collaborative simulation model, we first implement a simple on-line simulator under *ns*. We use it to control some network algorithm and observe how this can affect the performance of the network. We have chosen Random Early Drop queueing management algorithm as the underlying network algorithm to be adjusted because of its sensitivity to the parameter setting. And it also has been indicated that deciding the parameter setting of RED for different network conditions is not a trivial task[2], and the automation of its parameter adjustment is very meaningful. A simple network topology with one bottleneck link has been used for the test, which is shown as Figure 4. The router on the source side is equipped with

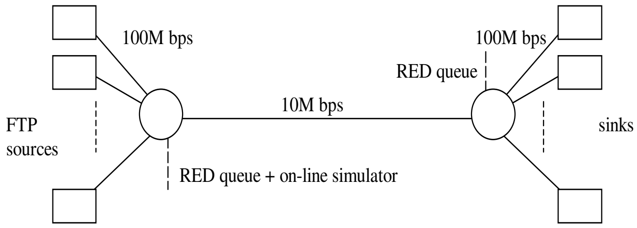


Figure 4: Test network topology

a RED queue, and an on-line simulator is used here to control the RED queue. The simulation varies the traffic conditions on the bottleneck link: 4 FTP traffics in the first 4 second period, 8 FTP traffics in the second one, 4 FTP traffics in the third one, and 16 FTP traffics in the last one. Our optimization objective is to achieve the best throughput. The simulation results are shown in Figure 5(without on-line simulation) and Figure 6(with on-line simulation). We can see from these figures that on-line simulation tunes up the setting of RED queue so that there is less oscillation in the queue size and a more stable queueing status is obtained. As a result, the utilization is better than the experiment without on-line control, in which we can find that when the traffic conditions change, the utilization drops dramatically for a while.

7 A Simple Implementation of On-line Simulator for Improving RED Performance in Real World

To verify the simulation results of our OCS scheme under real network, we have built a Linux testbed with 20+ machines and 3 subnets. The network topology is basically like the one shown in Figure 4. We adopted Linux as our test platform because of its open source policy and wide popularity. Furthermore, a variety of traffic management algorithms have already been implemented in Linux kernel, such as RED, CBQ, etc. Although it is still in the experimental stage, we found it quite stable in our experiments. A software package *iproute2* is provided in Linux to operate on these traffic management components with a great ease. We ported our OCS and farmer/worker into Linux platform and made an interface so that our OCS can interact with the traffic management algorithm in Linux kernel. All the configuration and assumption are almost the same as those in last section. We have used two traffic generators to generate

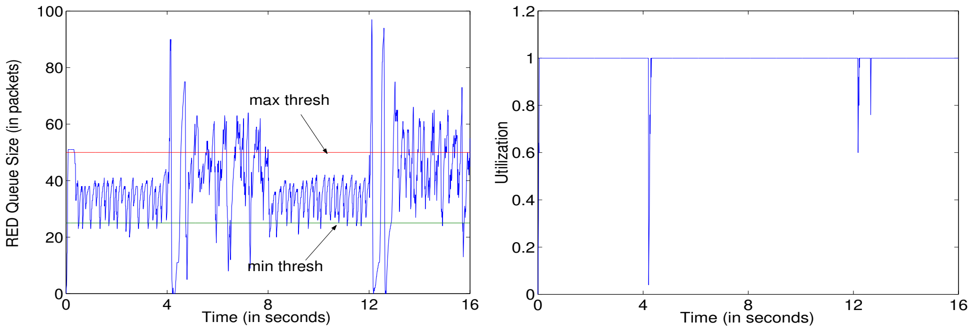


Figure 5: Simulation results without on-line simulation

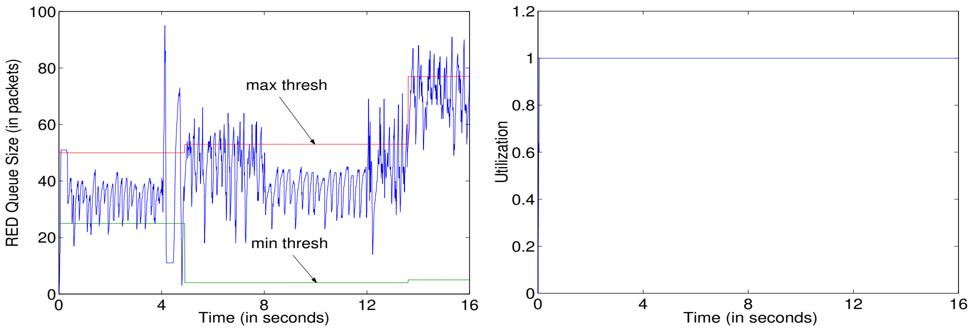


Figure 6: Simulation results with on-line simulation

massive TCP traffic from one side of the bottleneck link to the other side and observe the performance of RED on the bottleneck when the on-line simulator has been applied for dynamic control.

The simulation results are shown in Figure 7. We can see that there is a large oscillation in the average queue length of RED since we have chosen its parameter at some random in the beginning. Then in the middle of the simulation, we started our on-line simulator to search for the settings with low average queue length. Very quickly, the simulator found a much better setting and applied back to RED queue. This results in a dramatic decrease of the average queue length, and at the same time, the oscillation looks much smaller than before and the link utilization is not affected.

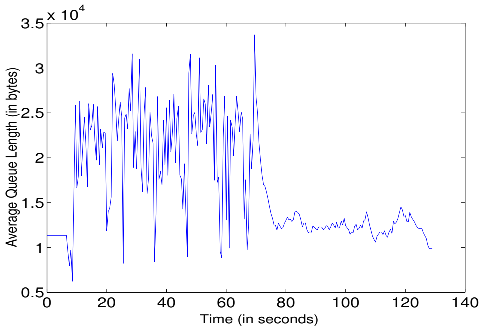
8 Online Simulation for Improving Routing Stability

As described in section 7, the on-line simulation has been demonstrated to improve the performance of the network management algorithm RED. The results show that the average queue lengths and hence the end-to-end delay can be significantly reduced by tuning parameters of RED using on-line simulation. A similar approach can be applied to routing, and an improvement in the end-to-end performance can be achieved by tuning the parameters of routing algorithm [16].

Adaptive routing is known to improve the network performance by increasing throughput and lowering the end-to-end packet delay. But it has been largely abandoned in the Internet due to the problems associated with routing oscillations. We have identified various parameters of an adaptive routing scheme which affect its performance and stability. The simulation based study indicates that these parameters may be tuned using online simulation to achieve a stable routing with improved performance. Moreover, the adaptive routing scheme may be deployed in the existing routers using OSPF. The parameters in this algorithm which we consider for tuning are *uFactor* and *bFactor*, which represent the weights associated with the link utilization and buffer utilization when the link cost is given by

$$linkCost = defaultCost \times (1 + uFactor \times Util\_)$$

where, $Util\_$ may be the exponentially averaged link or buffer utilization. These parameters are representative of the adaptiveness of the routing to the congestion. Another parameter that we considered is *Threshold* which reflects the minimum change in the cost to trigger a Link State Advertisement with updated metric value. Also, *Interval* between the routing updates is another parameter that we consider as it represents the tradeoff between the responsiveness of the routing algorithm and the maximum computational and bandwidth



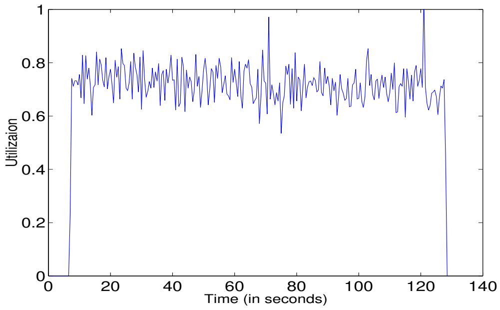


Figure 7: Simulation results

overheads associated with frequent route changes. We have demonstrated tunability of the parameters $uFactor$ and $bFactor$ to achieve significantly better end-to-end throughput and delay performance. The tuning of parameter *interval* was found to increase the throughput at the same time when it minimizes the number of route changes. However, the parameter *threshold* does not affect the network throughput or the end-to-end delay, but may be tuned to minimize the route changes. This will achieve a TCP-friendly routing as frequent route changes may lead to out-of-order packets, timeouts and result in a poor performance due to the flow control mechanism of TCP. However, testing on experimental testbed network with on-line simulation to tune the routing parameters is a goal for future work.

9 Conclusion

In this paper, we have proposed a collaborative on-line simulation scheme to perform the dynamic, scalable, and effective network control and management. To generate more realistic traffic, various protocol-specific, application-specific and self-similar traffic generators have been implemented in *ns* and some measures have been deployed to maintain the appropriate mix of traffic composition. A best-effort strategy has been used for searching for the good parameter setting within the limited time frame. To make highly efficient search in the parameter space, we combine multiple search techniques to perform an increasingly-improving search process. We also propose a new hybrid search algorithm, which, distinguished from the others, maintains a dynamic balance between *exploration* and *exploitation* and aggressively utilize the domain-specific information of the concerned parameter space.

To speed up the execution of the simulation, farmer-worker scheme has been designed to achieve a linear speed-up by distributing multiple experiments and executing them on more than one machines. Furthermore, we also use topology decomposition method to obtain higher speed-up by splitting up a large simulation into smaller pieces.

Various software components have been developed in *ns* and Unix/Linux. Extensive experimentation and simulation have been executed for the demonstration of our collaborative on-line simulation concept. These softwares and results are available on line.[17]

In addition to traffic management, we also investigated the feasibility of applying the on-line simulation into other areas, such as routing algorithms.

Our current work is limited to proof-of-concept stage. There is still a lot of work to be done for realizing the realistic, scalable and effective on-line control. Further work needs to address the problem of how to collect data from the network and build a good model from the data. Further study should also consider the scalability and cooperativity of the on-line simulation. How to make our simulator work with thousands of flows is still an open question.

References

[1] (1997) NS(*network simulator*). http://www-mash.cs.berkeley.edu/ns.

[2] S. Floyd, V. Jacobson, "Random Early Detection Gateways for Congestion Avoidance," *IEEE/ACM Transaction on Networking*, vol. 1, pp. 397-413, August 1993

[3] S. Kalyanaraman, R. Jain, S. Fahmy, R. Goyal, and B. Vandalore, "The ERICA Switch Algorithm for ABR Traffic Management in ATM Networks," *IEEE/ACM Transactions on Networking*, To appear, February 2000. Available

[4] S. Floyd, V. Jacobson, "Link-sharing and Resource Management Models for Packet Networks," *IEEE/ACM Transaction on Networking*, vol. 3 No. 4, pp. 365-386, August 1995.

[5] IETF, Differentiated Services(diffserv) working group. http://www.ietf.org/html.charters/diffserv-charter.html

[6] J. Apisdorf, K. Claffy, K. Thompson, and R. Wilder, “Oc3mon: Flexible, affordable, high performance statistics collection”, *Proceedings of INET'97*, June 1997.

[7] V. Paxson, and S. Floyd, “Wide area traffic: The failure of poisson modeling,” *IEEE/ACM Transactions on Networking'*, 3 (3), 226–244, June 1995.

[8] B. Ryu, “Fractal network traffic: From understanding to implications,” *Ph. D. thesis*, Columbia University, New York City, 1996.

[9] A. Andersen, and B. Nielsen, “A markovian approach for modeling packet traffic with long-range dependence,” *IEEE Journal on Selected Areas in Communications*, 16 (5), 719–732, June 1998.

[10] M. Yuksel, B. Sikdar, K. S. Vastola and B. Szymanski, "Workload generation for ns Simulations of Wide Area Networks and the Internet," Accepted for publication in the *Proceeding of Communication Networks and Distributed Systems Modeling and Simulation Conference*, San Diego, CA, 2000

[11] R. Jain, *The Art of Computer Systems Performance Analysis*, John Wiley & Sons, 1991.

[12] H.-P. Schwefel, *Evolution and Optimum Seeking,* New York: Wiley, 1995.

[13] S. Kirkpatrick, D.C. Gelatt and M.P. Vechhi, “Optimization by simulated annealing,” *Science*, vol. 220, pp.671-680, 1983.

[14] F.Glover, “Tabu Search I,” *ORSA J. Comput.*, vol. 1, pp. 190-206, 1989.

[15] D. H. Wolpert and W. G. Macready, “No free lunch theorems for optimization”, *IEEE Trans. On. Evolutionary Comput.*, vol. 1, pp. 67-82, April 1997.

[16] H. Tahilramani and K. Vastola, “ The Tunability of Network Routing using Online Simulation,” *Submitted to SCS-Symposium on Performance Evaluation of Computer and Telecommunication Systems*-2000.

[17] “Network Management and Control Using Collaborative On-line Simulation” website, http://networks.ecse.rpi.edu/ olsim/internal.