Proceedings of the IEEE International Conference on Communications 2001, Helsinki, Finland. vol. 1, pp. 204-209.

Traffic Management and Network Control Using Collaborative On-line Simulation

Tao Ye[1], Shivkumar Kalyanaraman[1], David Harrison[2], Biplab Sikdar[1], Bin Mo[2],
Hema Tahilramani Kaur[1], Ken Vastola[1] and Boleslaw Szymanski[2]
[1]Department of Electrical, Computer and System Engineering
[2]Department of Computer Science
Rensselaer Polytechnic Institute
Troy, New York 12180
{yet3, shivkuma, bsikdar, hema, vastola}@networks.ecse.rpi.edu
{harrisod, mob, szymansk}@cs.rpi.edu

Abstract

The complex and dynamic feature of the Internet requires scalable and effective network control. In this paper, a collaborative on-line simulation scheme is proposed to provide the automated and pro-active control functions for networks. This scheme introduces autonomous on-line simulators into local networks, which continuously monitor the surrounding network conditions, collect the relevant information, communicate with other simulators and execute collaborative on-line simulation. Based on the simulation results, the on-line simulators keep tuning the network parameters to the better operation point to fit the current network conditions. In this paper, we describe the basic concepts and investigate the solutions to the challenges faced in the realization of this scheme, particularly in the areas of network modeling, on-line simulation and parameter search. We also discuss the applicability of this scheme, and present the simulation results under *ns* and the test results of a preliminary implementation on a real network.

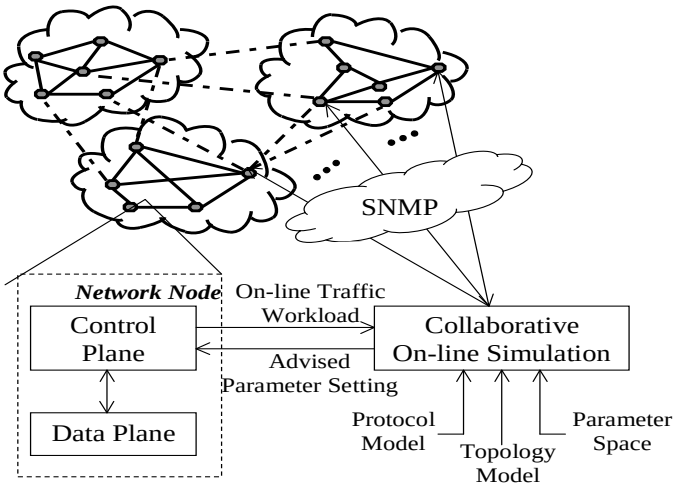

Figure 1: Collaborative on-line simulation scheme

1 Introduction

With the Internet expanding at an explosive speed, there is an urgent need for scalable and reliable network management and control. We propose in this paper a collaborative on-line simulation scheme to achieve the scalable, reliable and automated network management. The basic idea of this scheme is as illustrated in Figure 1.

s shown in the Figure 1, autonomous on-line simulators are introduced into local networks, which continuously monitor the surrounding network conditions, collect the relevant information and exchange information with other simulators. Based on all the information, on-line simulations are executed and parameter search methods are used to evaluate the results of the simulations and search for better network parameters. By applying these parameters into the network, the on-line simulators are able to keep tuning the network to the better operation point to fit the current conditions. Thus, a dynamic and automatic network control can be achieved. Note that the on-line simulation scheme only interact with the network control plane, therefore, it actually accomplishes a second-order control over the network. By second-order control, we mean that on-line simulation merely prescribes parameters required for the operation of network protocols and does not interfere with their normal operation in any other way.

The on-line simulation scheme uses a best-effort strategy in its second-order control, whose emphasis is not on "full" optimization, but on continuously and increasingly moving the system towards a "better" operating point. In other words, one of the major benefits of on-line simulation would be to continuously tune up under-

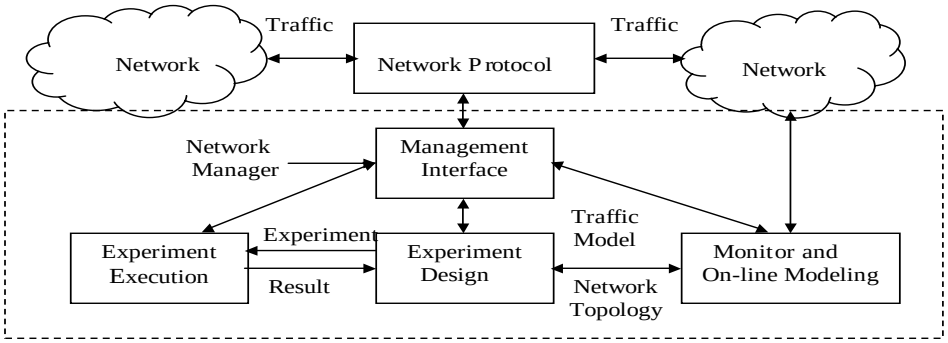

Figure 2: Structure of on-line simulator

lying operation (albeit at a larger time-scale than their normal operation). And in this sense, on-line simulation equips the network management infrastructure with "pro-active" management capabilities.

The on-line simulation scheme requires fast and reliable network simulation and parameter search. In the following, we will discuss the challenges faced in the areas of network modeling, on-line simulation and parameter search, and investigate the solutions to these problems. In section 2, we describe the basic structure of on-line simulator. Section 3 introduces our work in on-line modeling. Section 4 describes our approach to the efficient parameter search. Section 5 discusses two ways to speed up 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 validation 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 Structure of On-line Simulator

In the collaborative on-line simulation scheme, each local network domain is introduced with an on-line simulator module. Figure 2 shows the basic structure of an on-line simulator.

The simulator is composed of the following function units: *monitor and modeling*, *experiment design*, *management interface* and *experiment execution*.

Monitor and Modeling unit continually collects all kinds of information about the local network, such as network topology, traffic conditions, and tries to build the most updated network model for use by on-line simulation.

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

Experiment Design unit is responsible for setting up simulation experiments with appropriate search techniques, and analyzes these results to perform further searches and try to find "good" parameter settings.

Experiment Execution unit executes the experiments received from *experiment design* unit and return the results to the experiment designer. The network simulator *ns* [1] developed by UCB/LBNL and the VINT project has been chosen as our current simulation platform for its reliability and wide acceptance. However, this is not the only choice, and any other good simulation software can also be used as execution unit.

Besides interacting with the local network, the on-line simulator also communicates with other simulators and exchanges the relevant network information, such as network traffic models, good network parameters. Thus, a collaborative, scalable on-line simulation network is formed. Through this, the local simulator acquires a global view of the network and perform better network simulation and control.

3 On-line Modeling: Workload Generation

On-line modeling is to create traffic models, topology models, protocol models, etc. for use in the simulation to reconstruct the network conditions. On-line modeling, especially traffic modeling, is greatly complicated by the complexity and heterogeneity of the Internet. We will first address the problem of on-line traffic modeling. 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.

The first issue in generating the realistic traffic 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 [5] 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 [4] in the simulation.

Another important issue with traffic generation is self-similarity in wide area and Ethernet traffic[5]. 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 [6] and [7].

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[8].

4 Experiment Design

Since the network conditions keep changing all the time, the on-line simulation scheme also requires the fast experiment design method to quickly finish the simulation experiments and find the "good" network parameters before the underlying network information become dated. 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 on seeking the optimum setting; instead a best-effort strategy is adopted to find a better point within the limited time frame. Based on this principle, it would be desirable for the experiment design to deploy an iterative method, which leads the search to the optimum point with an increasingly improving manner. Thus, the search can be interrupted at any time and still produce a result better than the starting point. This provides us the possibility to make a compromise between the goodness of the result and the search time.

The basic procedure of our approach is first probing the search space roughly and find the important parameters which have greater effects on the network performance. After pruning part of the search space by ignoring less important parameters, we explore the promising 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 Probing the 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[9].

2^k full factorial design tries all possible combinations of the parameter boundaries and fits the results into a non-linear regression model to analyze the importance of different parameters. It is not a iterative method, which is opposite to our search strategy. To achieve the 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 described in [9]. 2^{k-p} fractional factorial design is a technique which just execute 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[9]. After finishing this subset of experiments and analyzing the simulation results, we 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 search is interrupted, 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 space in detail with state space search techniques. Basically, the state space search algorithms 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[13] 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 hill-climbing technique[10] with TABU technique[12] included to avoid revisiting the previous region and speed up the exploration process. Different from the other algorithms, the hybrid method maintains a dynamic balance between *exploration* and *exploitation* by accepting the bad move with a probability of $\exp(-\textit{current gradient feature}/\textit{average gradient feature})$. The idea here is that in the promising areas whose gradient is much steeper than the average gradient, the algorithm mostly performs *exploration*, i.e. random walk, and otherwise, *exploitation*, i.e. hillclimbing, to quickly converge to the local optimum. Note that this balance strategy is different from that of Simulated Annealing[11]. 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 may take a long time to converge.

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.

5 Speeding Up On-line Simulation

In the on-line simulation scheme, the execution of the simulations is the most time-consuming task. We have designed a few methods to speed up this process.

5.1 Farmer-Worker Infrastructure

The first method to achieve the speedup is to parallel the execution of the simulations. We have developed a farmer-worker infrastructure to distribute many single-machine experiments across multiple 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 large quantity of simulation experiments. The infrastructure is shown in Figure 3.

The dispatcher is the interface between this distributed simulation executer and the outside experiment designer. All the experiments have to go through this interface to be distributed among the workers. The farmer is the center of this infrastructure, which controls and synchronizes the operations of dispatchers and workers. The

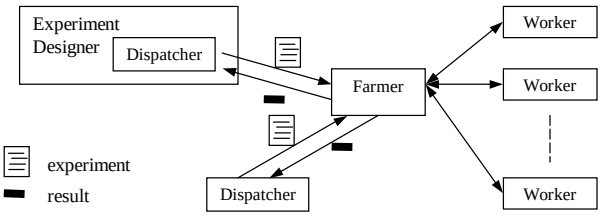

Figure 3: Farmer-Worker structure

workers are the actual experiment executers. We can use multiple workers for the same experiment designer 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 evenly distribute the experiments over the whole network and maximize the utilization of the computing resources.

5.2 Topology Decomposition

The farmer-worker method is effective only when the experiment designer sends out a batch of experiments every time and then wait for the results. Its speedup level is dependent on the size of each batch. On the other hand, topology decomposing method is used to expedite the execution speed of a single simulation experiment. Through tests, we have found that the running-time of a simulation is not linearly proportional to the simulated network size. Figure 4 shows the simulation results on a network with a simple star topology. Using the

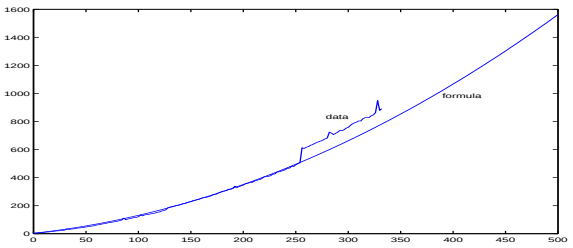

Figure 4: Execution time vs. simulation size

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 4.

From the above, we can see that the execution time of a network simulation holds a quadratic relationship 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 decomposition 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 split 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 Experiment with the On-line Simulator

To test the validity of our on-line simulation model, we first implemented 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. In this test, we use Random Early Drop(RED) queueing management algorithm as the underlying network algorithm to be adjusted because of its sensitivity to the parameter setting. And it has also 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. However, the applicability of the on-line simulation scheme is not limited to RED or other queueing management algorithms. Any other network protocol or algorithm, which is sensitive to the parameter setting, can be controlled with our on-line simulation scheme.

A simple network topology with one bottleneck link has been used for the purpose of proof-of-concept, which is shown as Figure 5. The router on the source side is

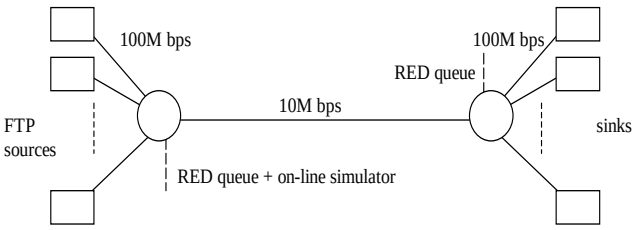

Figure 5: Test network topology

equipped with 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 connections in the first 4 second period, 8 FTP connections in the second period, 4 FTP connections again in the third period, and finally 16 FTP connections in the last period. Our optimization objective is to achieve the best throughput. The simulation results are shown in Figure 6 (without on-line simulation) and Figure 7 (with on-line simulation). We can see from these figures that when the FTP connections change from 4 to 8 and from 4 to 16, the on-line simulation tunes the setting of RED queue to reduce oscillation in the queue size and achieve a more stable queueing status. 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 Real World Implementation of On-line Simulator

To verify the simulation results of our 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 5. We adopted Linux as our test platform because of its open source policy and great popularity. Furthermore, a variety of traffic management algorithms have already been implemented in Linux kernel, such as RED, CBQ, etc. Although these traffic control elements are still in the experimental stage, we found it quite stable in our experiments. In addition, a software package *iproute2* is provided in Linux to operate on these components with a great ease. We ported our on-line simulator into Linux platform and created an interface so that the simulator can interact with the traffic management algorithm in Linux kernel and adjust the algorithm parameters automatically. All the configuration and assumption are almost the same as those in last section. We have used *netperf*[15] traffic generators to generate 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. For testing purposes, we assume the simulator already has all the information about the network conditions. Our carefully designed traffic generator are applied in the simulation to reproduce the realistic traffic under *ns* and the previously described speedup measures are also used to boost up the speed of the simulations. The simulation results are shown in Figure 8.

We can see in the figures that there is a large oscillation in the average queue length of RED since we have chosen its parameter at random in the beginning and this parameter setting may not fit the current network conditions at all. 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 reduction of the average queue length, as shown in Figure 8 after the

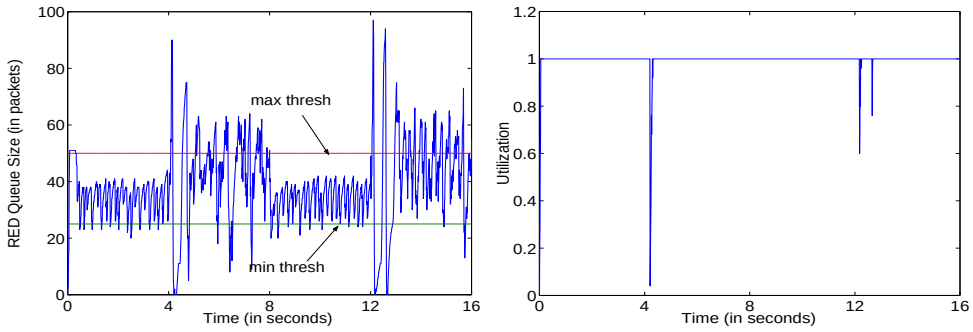

Figure 6: *ns* simulation: RED average queue size and link utilization (without on-line simulation control)

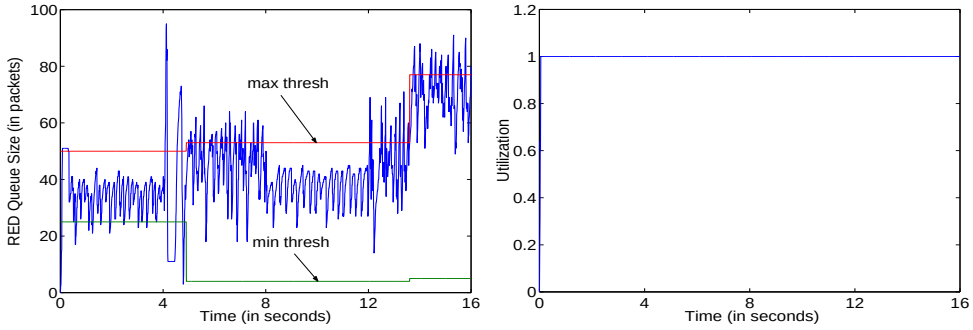

Figure 7: *ns* simulation: RED average queue size and link utilization (with on-line simulation control)

test starts for about 70 seconds, 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 [14].

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 on-line 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 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.

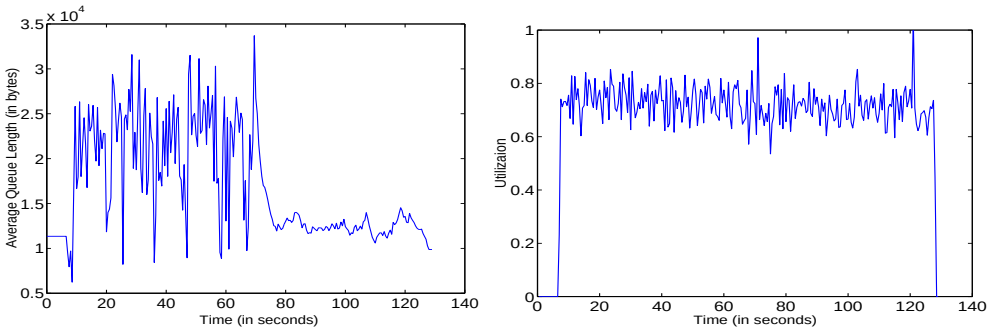

Figure 8: Real network test: RED average queue length and link utilization

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 realize this scheme, the problems faced in the areas of network modeling, network simulation and parameter search are addressed and some solutions are presented. For traffic modeling, we designed and implemented various methods to maintain the appropriate mix of traffic composition and generate realistic traffic. To speed up the execution of the simulation, we use farmer-worker scheme to distribute experiments and parallel their execution on multiple computing resources. In addition, we also use topology decomposition method to expedite the execution of a single simulation by splitting up the large simulation into smaller pieces. In parameter search, we use a best-effort, increasingly improving strategy to search for better parameter setting within the limited time frame. We also proposed a new hybrid search algorithm, which, different from the others, maintains a dynamic balance between *exploration* and *exploitation* and aggressively take advantage of the known information of the parameter space to perform highly efficient search.

Various software components have been developed in *ns* and Unix/Linux. Preliminary experimentation and simulation have been executed on traffic management algorithms for the demonstration of our collaborative on-line simulation concept. These tests produce very promising and encouraging results. All the softwares and results are available on line[16]. In addition to traffic management, we also investigated the feasibility of applying the on-line simulation into other areas, such as routing algorithms. The experiments on this aspect is under way.

The 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 and more complex network scenario 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] IETF, Differentiated Services(diffserv) working group. http://www.ietf.org/html.charters/diffserv-charter.html

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

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

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

[7] 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.

[8] M. Yuksel, B. Sikdar, K. S. Vastola and B. Szymanski, "Workload generation for ns Simulations of Wide Area Networks and the Internet," *Proc. of Communication Networks and Distributed Systems Modeling and Simulation Conference*, pp 93-98, San Diego, CA, USA, 2000.

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

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

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

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

[13] 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.

[14] H. T. Kaur and K. Vastola, " The Tunability of Network Routing using Online Simulation," *Proceedings of the Symposium on Performance Evaluation of Computer and Telecommunication Systems*, July 16-20, 2000 Vancouver B.C. Canada.

[15] *netperf* traffic generator, http://www.netperf.org.

[16] "Network Management and Control Using Collaborative On-line Simulation" website, http://networks.ecse.rpi.edu/ olsim.