Network Management Game

Engin Arslan, Murat Yuksel, and Mehmet H. Gunes
Department of Computer Science and Engineering
University of Nevada, Reno
1664 N. Virginia Street, Reno, NV 89523, USA
{earslan, yuksem, mgunes}@cse.unr.edu

***Abstract*—Network management and automated configuration of large-scale networks is one of the crucial issues for Internet Service Providers (ISPs). Since wrong configurations might lead to an enormous amount of customer traffic to be lost, highly experienced network administrators are typically the ones who are trusted for the management and configuration of a running ISP network. We frame the management and experimentation of a network as a "game" for training network administrators without having to risk the network operation. The interactive environment treats the trainee network administrators as players of a game and tests them with various network failures or dynamics. To prototype the concept of "network management as a game", we modified NS-2 to establish an interactive simulation engine and connected the modified engine to a graphical user interface for traffic animation and interactivity with the player. We present initial results from our game applied to a small set of players.**

***Index Terms*—network management; network simulation; NS-2; network game; interactive simulation**

I. Introduction

Online management of a running large-scale network poses many challenges that have attracted significant research. Due to the prohibitive costs of building an ISP network, the challenges involved in managing the network resources are exacerbated. Today, vital and critical applications such as VoIP, IPTV and financial markets are converging onto the Internet infrastructure, and thus making the job of provisioning high-performance network services an even more important one. From the technical side, emergence of various substrate networking technologies like 3G wireless and mesh networking is complicating the management tasks to the extent that network operators give up on optimizing their networks' configuration and barely cope with handling default configuration settings of tremendous number of components involved. In most cases, getting the large-scale network to work is the typical target. Highly experienced human administrators are of critical importance as they are typically the only ones who have the gut feelings to quickly find the optimum (or close-to-optimum) response to a major failure, e.g., finding an optimum rerouting for huge amount of traffic on a broken pipe.

Though there have been several tools and outcomes [1] to automate the process of large-scale network management, network operators have found themselves more comfortable with trusting to highly experienced well-trained human administrators. However, the complexity of the management and configuration problem is increasing due to inevitable heterogeneity in substrate technologies as well as applications' demand for more stringent performance targets. Trends in cross-layer design [2] of protocols and more integrated designs of various network components are certainly helping; however, such methods typically further complicate the network configuration due to additional parameters they introduce into the system. Thus, tools to train administrators and to achieve automated ways of managing a running network are vitally needed. In this paper, we propose the concept of "network management as a game" that frames the problem of training network administrators in exploring what-if scenarios as a "game". The fundamental goal of our framework is to establish a game-like environment for trainee network administrators to experiment and play with the networks, without having to risk the large-scale network operation.

Achieving higher utilizations via better load balancing (also known as traffic engineering) is one of the main management problems for network operators. Many algorithms and tools are developed to find optimal or close-to-optimal solutions for high network utilization; however, they are mostly not preferred because of reliability issues. A common implication of load balancing for network administrators is to configure interior gateway protocol (IGP) link weights so that shortest paths give result to a well-balanced traffic load on network links. The IGP link weight setting problem is known to be NP-hard [3]. Several prior studies employed advanced optimization techniques to set the IGP link weights for a given topology and traffic matrix to improve various network performance metrics such as delay, throughput, or congestion [1], [4], [5], [6], [7].

In this paper, we apply our gaming framework to the problem of IGP link weight setting. We leverage existing simulation tools to establish a simulation and build an animator that interacts with the simulation engine in real time. The player (or the trainee) interfaces with the animator and inputs various new IGP link weights as new configurations. The animator, then, conveys these new configurations to the backend simulation engine. The animator interface attempts to make the environment an exciting one for the player. We present snapshots of our initial prototype, and experiment results obtained from 8 players.

The rest of the paper is organized as follows: In Section III, we first describe our network management game (NMG) framework and its details. Then, in Section IV, we present our experimental setup. In Section V, we provide results obtained from NMG experiments on 8 players. Finally, we conclude our

work and discuss potential future directions in Section VI.

II. Related Work

Currently, most of the basic training for a network administrator is performed by means of well-defined certification procedures [8]. The administrators receive several months of education to obtain these certifications to prove that they have the basic skills and knowledge about configuring and administering a network. However, custom skills related to maximizing performance of a particular operational network cannot be attained via generic certifications. Such custom skills require several months of training in work environment where the certified trainee can learn what to do in action from her peers with more experience on that particular network.

Typical customized training of network administrators involves what-if analysis, which is mostly done by in-house tools. What-if analysis is a brainstorming activity that uses extensive questioning to postulate potential failures and issues in a system, and ensure that appropriate safeguards are put in place against those problems. Businesses often use the scenario manager tool of Excel to explore different scenarios such as the decision making process in e-commerce [9]. Although what-if analysis has high impact within business intelligence platforms, its usage is extended for several purposes such as hazard analysis [10], index selection for relational databases [11], Content Distribution Networks (CDNs) [12], and multi-tier systems [13]. For instance, in [12], the authors developed a tool (WISE) that predicts how a deployment of a new server to an existing CDN affects service response time.

Furthermore, various tools have been developed to guide investments and determine how to improve network performance [14] with minimal investments. Though existing what-if analysis tools are pretty successful in helping a network administrator and strategic director make an informed decision about future investments, they cannot train for dynamic events such as demand spikes or failures. A key difference in our approach is the capability of simulating the interactivity and dynamism that might take place in an operational network.

The concept of using a virtualized game-like environment for training is not new. [15], [16] It has been actively used for cases where experimentation with the real system is too costly or risky. Military training involves a lot of such practices, e.g. pilot training [17], commander training [18]. Financial investment training [19] is another venue where a game-like environment can be used for training before deploying money on stock market. To the best of our knowledge, usage of a game for training network administrators was not done before.

III. NMG Framework

The NMG framework has two parts as shown in Figure 1. On the backend, we use a simulation engine to imitate a real network. We interface an animator graphical user interface (GUI) to the backend simulation engine to visualize simulation events and to provide interactivity to the player. The player makes changes on the GUI, which are taken to the simulation

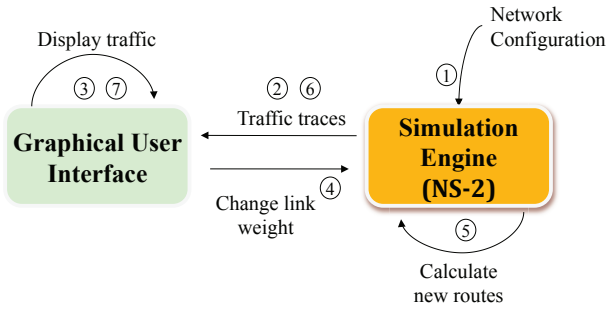


Fig. 1. Block diagram of Network Management Game (NMG) components. Event sequence starts with (1), goes on with (2), (3). Whenever the player/user makes changes, the sequence (4), (5), (6) and (7) repeats.

engine on real-time. In our prototype, we used NS-2 [20] for the simulation engine and developed a custom animator GUI.

NS-2 is a widely used network simulation tool developed in C++ and provides a simulation interface through OTcl/Tcl. The user describes a simulation scenario (i.e., network topology, and traffic) via Tcl scripts, and then NS-2 simulates the scenario. However, NS-2 does not support real-time interactivity. That is, the user has to run the complete simulation before she can observe the animation of the simulation. Thus, to observe effects of a change in the simulation's scenario (e.g., a change in the network configuration), the user needs to run the simulation twice. If the user wants to see the effects of link weight manipulations throughout the time, the number of simulations to be run goes to infeasible numbers. Thus, achieving real-time interactivity is not possible with NS-2 as is. We addressed this issue by synchronizing our custom-designed animator's GUI with the engine.

In order to realize the synchronization of the GUI and the NS-2 engine, we established a two-way pipe via TCP sockets between them. When NS-2 engine starts, it opens a TCP server socket. Then, the GUI process connects and starts receiving the packet traces via the pipe. Briefly, once NS-2 receives the initial configuration file, it starts the simulation and generates traces to describe events taking place in the simulation. Via the pipe, we transfer these traces to GUI, which then displays them to the user. When the user makes a change on a particular link's weight[1], the simulation engine is informed about it through the pipe again. Once the simulation engine receives the changes, it recalculates the routing based on these changes and carries on to generating more traces.

A. The Animator

We designed a new GUI which displays the condition of the simulation on real time and provides interactivity via changing link weights. As the simulation engine runs, event records are generated and sent to the GUI. Then, the GUI processes these events to visualize them. Basically, 3 types of events are taken into consideration: *sending*, *receiving*, and packet *drop*.

[1]Though our current prototype only allows link weights to be changed by the player, it is a straightforward extension to allow other simulation configuration parameters.

In order to show the first 2 of these events on the animator, we adjusted the color of the links based on their load, as shown in Figure 4. However, in our current prototype, we do not animate the packet drops but show the amount of loss that have occurred as a number on the right top corner of the interface. The player's goal in NMG is to maximize the network's throughput by manipulating link weights. We tried to design an interface where user can easily keep track of the ongoing condition of the simulated network and observe how the link weight manipulations affect the throughput. Thus, the coloring of links is used to represent packet-based actions and instantaneous throughput is displayed to inform the player about the effects of changes in link weights.

B. Engine-Animator Interaction

A crucial challenge in designing the NGM framework was to synchronize the simulation engine and the GUI. The simulation engine is able to simulate a given network configuration very fast; however, such speedy simulation is unrealistically fast and the user cannot keep track of the simulation.

One alternative solution is to divide the simulation time into small periods. In each period, the simulation engine simulates the latest network configuration and sends the events traces to the GUI for animation to the player. While the GUI is animating the events of the previous period, the simulation engine runs the next period and waits until the GUI finishes animation. Thus, the simulation engine always runs one period ahead compared to the GUI. If the player changes the weight of a link, the simulation engine is informed about it at the end of the current period. The drawback of this solution is that the synchronization cannot be achieved properly because whenever the player makes a change, it can only be reflected one period later. The amount of interactivity and realism will depend on the length of the periods.

To obtain a more realistic and interactive environment, we decided to reduce the speed of the simulation engine. The simulation engine waits a fixed amount of time before moving to the next event to proces. We tuned the amount of wait time such that the engine runs concurrently with the GUI. By this way, whenever the player changes a link weight, the change is sent to the simulation engine and new routing paths are calculated. Thus, the simulation resumes with new network configurations.

IV. Experimental Setup

The end goal of our NMG framework is to train people in network management and administration. To observe potential benefits of NMG in training, we have designed a sequence of test cases on a difficult network management problem, i.e., intra-domain traffic engineering.

A. Game Goal: Tuning IGP Link Weights for Load Balancing

The Interior Gateway Protocol (IGP) is an intra-domain routing protocol which uses link-state costs (a.k.a. link weights) to determine end-to-end shortest paths. A typical ISP network management problem is to tune the IGP link weights

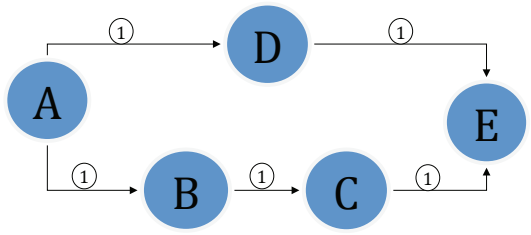


Fig. 2. IGP path selection example. Link weights are same at the beginning. Traffic flows from A to E

so that the end-to-end shortest paths change, and thus the load on individual links are changed. Figure 2 illustrates a simple test case where link weights cause traffic to shift from one end-to-end path to another. When the traffic from A to E is considered, the traffic will follow the path A-D-E by default because its total weight is less than that of the alternative path A-B-C-E. In this case, suppose that available bandwidth on the path A-B-C-E is larger than A-D-E path. Then, the player/user is expected to increase the total weight of the default path such that the traffic flows through higher bandwidth path.

An intuitive way of tuning the IGP link weights for load balancing is to increase the link weights on the links that are highly utilized. However, automatically associating the link weights to the load on the links is known to cause instability in routing, and hence, is avoided in practice. We picked this particular problem of IGP link weight setting for our game. The purpose of such link weight changes may be to reduce other metrics like delay or fast failure recovery. We mainly focus on the utilization of the network in this paper.

Thus, the goal of the player in our current NMG prototype is to maximize the aggregate network throughput by manipulating the IGP link weights. The player is given a network topology with an initial configuration and allowed to increment or decrement each bi-directional link's weight by clicking a red or green button on the corresponding link (see Figure 4). We monitor cumulative and instantaneous network throughput to evaluate the player's performance. The cumulative throughput, τ_{cum}, is mainly used to see how the player performed in the overall test whereas the instantaneous throughput, τ_{ins}, is used for measuring how well the player responds dynamic changes in the network such as failures or demand spikes. We calculated these throughput values as follows:

$$\tau_{cum} = \frac{\sum_{i=0}^{t_{total}} P_s(i)}{t_{total}} \qquad \tau_{ins} = \frac{\sum_{i=t_1}^{t_2} P_s(i)}{t_2 - t_1}$$

where $P_s(i)$ represents the total size of packets which are successfully transferred from source to destination within ith time unit. While all transmitted packets are taken into the consideration for cumulative throughput, packets which are transmitted in a given time period (t_1, t_2) are considered for the instantaneous throughput.

In our training test cases, we used TCP to create traffic flows since it utilizes the maximum available bandwidth on its path. This makes the game more interesting even if there

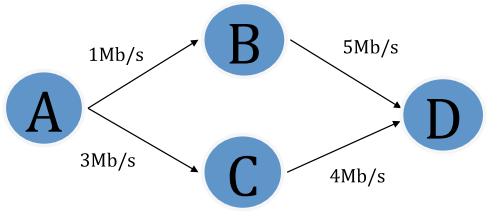


Fig. 3. Network topology with TCP traffic from A to D.

TABLE I
DETAILS OF TRAINING TEST CASES

Test Case #	# of Flows	Time (mins)	Minimum Link BW (Mb/s)	Maximum Link BW (Mb/s)
1	1	2	2	5
2	2	3	1	4
3	2	5	2	6
4	4	6	1	8
5	4	6	2	6
6	4	6	2	6
7	4	6	1	8
6’	4	6	1	6
7’	4	6	1	6

is no failure in the network, as the player has to guide the TCP flows to accumulate higher throughput. For example, in Figure 3, there is a TCP traffic from A to D. If the traffic flows through the path A-B-D, then the throughput would approximately be 1Mb/s. On the other alternative, it almost reaches 3Mb/s if the traffic follows the A-C-D path. We placed many similar alternative paths in our test cases to train the players on discovering better ways of extracting more throughput from the network, and thus maximize the network utilization.

B. Training Mechanism

1) Test Cases: In order to observe the benefit of NMG in terms of training, we have setup five different test cases to train the players in terms of some target skills in the IGP link weight setting. As shown in Table I, the test cases are different from each other in terms of complexity and number of traffic flows. The purpose of the test cases #1-#5 is to train the players with simple topologies and traffic flow scenarios. But, the test cases for Before Training(#6 and #7) and After Training (#6’ and #7’) are relatively more difficult and designed for evaluating how well the players are trained.

The test case # 3 is shown in Figure 4 as an example. There are two TCP traffic flows and bandwidth of the links are different from each other. We increased or decreased the width of the links based on the their bandwidth to make it easier for the player to realize the links with larger bandwidth. When the player moves the mouse on a link, the traffic flows on that link can be observed as well as its current utilization and bandwidth. The links are colored with respect to their current utilization where links with no traffic on it are green colored whereas completely utilized links are red colored.

2) Training Stages: We applied a staged training process to observe efficacy of NMG in training network management skills. As illustrated in Figure 5, we first gave a tutorial to

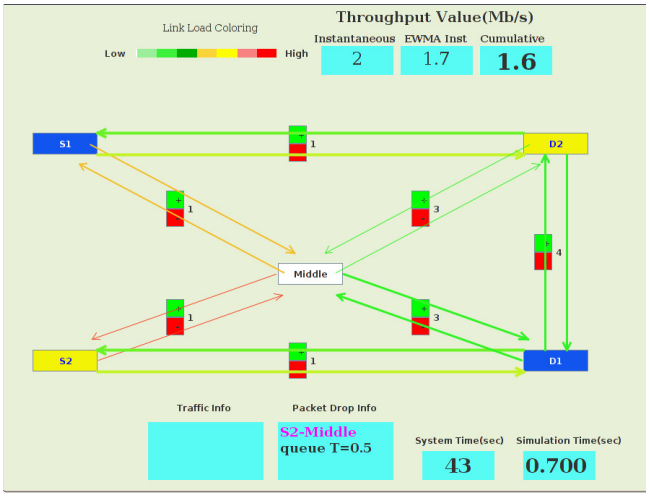


Fig. 4. One of the training test cases (#3) in which there are two flows. Sources and destinations are marked by color and node text.

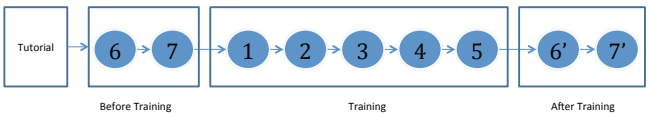


Fig. 5. Training stages: Tutorial, Before Training, Training, After Training

the player about the environment and assured that the player knows what the game is about and how the GUI works. Then, we tested the player with a relatively complex network (i.e., test case #6) to observe how s/he performs before the training. Next, we trained the player with test cases #1-#5 with each case having a fixed amount of time. Finally, after the training stage is over, we exposed the player to test case #6’, which is similar to the test case #6 in terms of difficulty. To see how users react link failures, we applied test cases #7 and #7’ before and after training. To evaluate if the training stage helped the players really learn the skill of IGP link weight setting, we compare the players’ performance after training.

When designing the test cases for the Before and After Training stages, we purposefully made the test cases noticeably more difficult than the test cases in the Training stage (i.e., test cases #1-#5). This allows us to eliminate the dependence of training to the specific test cases and reveal how much the player learned the skill rather than the specific test case. We used the Abilene backbone topology (shown in Figure 6) to design the test cases #6, #7, #6’ and #7’ but with a different configuration in each. In that topology, we placed 4 different TCP traffic flows with link capacities varying from 1Mb/s to 8 Mb/s. In test cases #7 and #7’ link failures are simulated. As an example, in Figure 7, the link between Seattle and Los Angeles is failed and repaired after some time later. During the link failure period, the player is expected to manage the flows traversing the failed link.

V. RESULTS

To observe how well our game environment performs in training people, we experimented with 8 players and monitored

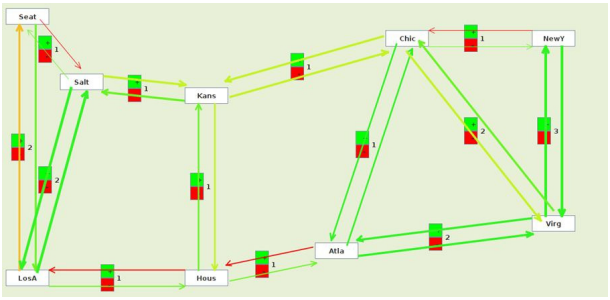


Fig. 6. Topology used in the test cases #6 and #6'.

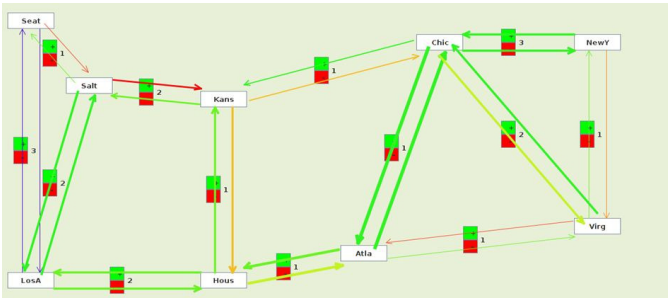


Fig. 7. Topology w/ failures used in the test cases #7 and #7'.

their performances. We kept track of the throughputs the players achieved throughout the game. We compared the performances of the players with the two extreme cases: "Optimal" and "No Player". The "Optimal" is the maximum possible throughput in that particular test cases, while "No Player" refers to the throughput obtained when the initial configuration of the test case is not changed by the player.

We also used two black-box optimization algorithms in after training simulations (i.e., test cases #6' and #7') to compare the player performance with optimization algorithms. We used Genetic Algorithm (GA) and Recursive Random Search (RRS) [21] as the optimization algorithms. GAs are of the global search heuristic algorithms to find a close-to-optimal solution in optimization and search problems. They use evolutionary search technique which is inspired by evolutionary biology. RRS uses random sampling to make use of initial high-efficiency property and restart random sampling to maintain explored high-efficiency. These two algorithms are relevant to problems such as IGP link weight setting since they can find a good solution fast. Comparison with these algorithms allows us to see how well a human player performs against optimization algorithms.

3) The Difference of Training: Figures 8(a) and (b) show the players' performance in the Abilene topology before the Training stage for the test cases #6 and #7, respectively. Clearly, the players do achieve higher throughput than the No Player, but there is still considerable difference to the optimal routing solution. Respectively for the test cases #6 and #7, the optimal throughput was 9 Mb/s and 13 Mb/s while the players' average performance was 7.11 Mb/s and 9.73 Mb/s.

Figures 8(c) and (d) show performance of the players after the Training stage for the test cases #6' and #7', respectively. Compared to the performance before the training, we can see a significant improvement in the players' performance, particularly in the test case without failures, i.e., test case #6. The players' average performance increased to 9.73 Mb/s in the test case #6', which shows a 21% improvement in comparison to their performance prior to training. This also shows that almost all the players obtained the optimum result in the test case without failures (i.e., #6').

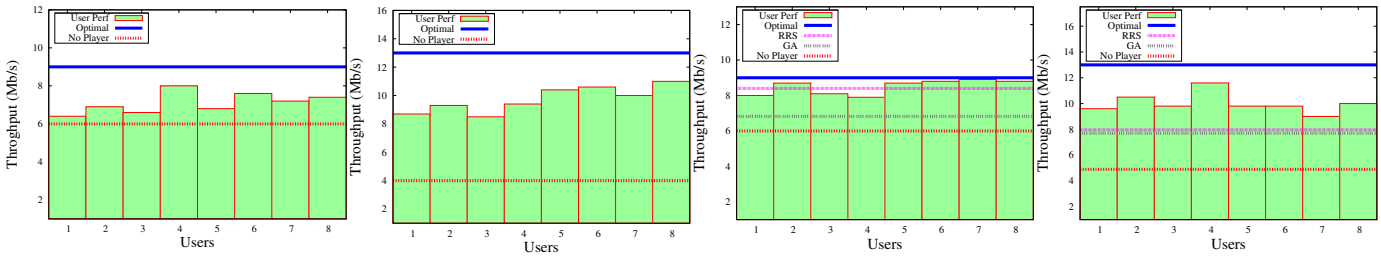
For the case with failures (i.e., #7'), the players achieved 10.01 Mb/s average throughput after they received training, which corresponds to a 2.9% improvement from the training. This is sizably smaller than the improvement observed for the test case without failures. This small improvement in failure case also supports our claim of training since the players were not trained on link failure cases during Training stage and their performances did not improve considerably after training.

Also, we observe that most players improved the network throughput more than the black-box optimization algorithms RRS and GA in no failure case. Furthermore, we can see that the players outperform the black-box algorithms with a much more significant margin when failures are included. This result implies that human players (i) perform better than the automated tools when the system-under-test exhibits dynamism or exceptional situations like failures, and (ii) can handle unexpected behavior better. This outcome also validates our motivation for designing the NGM framework.

4) The Best and The Worst Players: In this part, we zoom in to the performances of the players and plot the performance graphs of the best and the worst players as samples. We plot the instantaneous, τ_{ins}, and cumulative, τ_{cum}, throughput achieved by these players as the game progresses. To comparatively observe how the player's changes affect the achieved throughput, we also plot the instantaneous throughput when no updates to the link weights are done, i.e., the No Player case. Note that the No Player and τ_{ins} lines fluctuate frequently due to the adaptive behavior of the TCP flows.

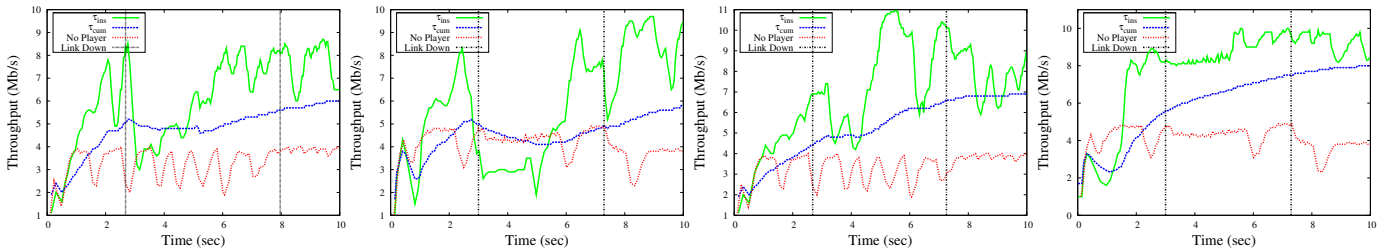
Figures 9(a) and (b) are performance graphs of the worst player for the test cases #7 and #7', respectively. It is evident from the figures that some failures do not affect the throughput as much as others. The difference depends on whether or not the failing link is used for any of the traffic flows. As we only have 4 flows in these test cases, some links are not used for traffic; and, when those links fail, the throughput is not affected, e.g., the second failure in Figure 9(a).

Figure 9(c) and (d) plot the best performing player's behavior over time during the test cases #7 and #7', respectively. After the training, the player clearly responds to the failures faster and starts tuning the link weights earlier with compared his previous performance. Further, the player clearly learns to tune the link weights better to discover more bandwidth in the network to increase the throughput, even before failures happen (see the jumps in τ_{ins} at around time 2s and 5.5s).



(a) Before training (#6) (b) Before training w/ failures (#7) (c) After training (#6') (d) After training w/ failures (#7')

Fig. 8. User performances compared to the best and the worst possible throughput.



(a) Worst player (case #7) (b) Worst player (case #7') (c) Best player (case #7) (d) Best player (case #7')

Fig. 9. Performance of the worst and the best players on the test cases with failures.

VI. Summary and Future Work

In this work, we have designed and developed a game-like environment to train people in terms of network management skills. We focused on the problem of IGP link weight setting and aimed two basic skills that might be needed by network administrators: (i) discovering and selecting paths with high bandwidth, and (ii) decoupling flows to better load balance the network traffic. To see if simple test cases for training on these skills help improving the players' capabilities, we designed five simple test cases and evaluated the players on two relatively complex test cases where link failures may exist. By comparing the players' performance on the complex tests cases played before and after the training, we observed a sizable increase in the players' capability of finding a near-optimal solution to the IGP link weight setting problem.

First step into the future work is to extend the quantity and variety of test cases in our work. We also plan to extend the concept of NMG to large-scale networks as well as more realistic traffic flows. Such an extension involves non-trivial steps such as visualization of large-scale topologies and traffic.Another dimension for future work is to use metrics other than throughput for evaluating the performance of players. Quality-of-service (QoS) metrics such as delay and loss are highly relevant to the practice of network operation, particularly when some customers are promised higher priority service.

Acknowledgments

This work is supported in part by the NSF awards 0721600 and 0721609.

References

[1] T. Ye, H. T. Kaur, S. Kalyanaraman, and M. Yuksel, "Large-scale network parameter configuration using an on-line simulation framework," *IEEE/ACM Trans. Netw.*, vol. 16, pp. 777–790, August 2008.

[2] V. Srivastava and M. Motani, "Cross-layer design: a survey and the road ahead," *Communications Magazine, IEEE*, vol. 43, pp. 112–119, 2005.

[3] B. Fortz and M. T. At, "Increasing internet capacity using local search," *Computational Optimization and Applications*, vol. 29, pp. 13–48, 2004.

[4] B. Gonen, M. Yuksel, and S. Louis, "Probabilistic Trans-Algorithmic search for automated network management and configuration," in *Proc. of IEEE MENS*, Miami, Florida, USA, 12 2010.

[5] B. Fortz, "Internet traffic engineering by optimizing ospf weights," in *Proc. IEEE INFOCOM*, 2000, pp. 519–528.

[6] L. S. Buriol, M. G. C. Resende, C. C. Ribeiro, and M. Thorup, "A hybrid genetic algorithm for the weight setting problem in ospf/is-is routing," *Journal of Combinatorial Optimization*, vol. 6, pp. 299–333, 2003.

[7] A. Riedl, "A hybrid genetic algorithm for routing optimization in ip networks utilizing bandwidth and delay metrics," in *Proc. of IEEE IPOM*, 2002, pp. 166–170.

[8] "Cisco Certifications," "http://www.cisco.com/web/learning/index.html".

[9] H. K. Bhargava, R. Krishnan, and R. Müller, "Electronic commerce in decision technologies: a business cycle analysis," *Int. J. Electron. Commerce*, vol. 1, pp. 109–127, June 1997.

[10] P. Baybutt, "Major hazards analysis an improved process hazard analysis method," *Process Safety Progress*, vol. 22(1), pp. 21–26, 2003.

[11] S. Chaudhuri and V. Narasayya, "Autoadmin what-if index analysis utility," 1998, pp. 367–378.

[12] M. Tariq, A. Zeitoun, V. Valancius, N. Feamster, and M. Ammar, "Answering what-if deployment and configuration questions with wise," in *Proc. of the ACM SIGCOMM*, 2008, pp. 99–110.

[13] S. Chen, K. R. Joshi, M. A. Hiltunen, W. H. Sanders, and R. D. Schlichting, "Link gradients: Predicting the impact of network latency on multitier applications," in *IEEE INFOCOM*, 2009, pp. 2258–2266.

[14] *Mate*, http://www.cariden.com/products.

[15] R. E. Chatham, "Games for training," *Communications of the ACM*, vol. 50, pp. 36–43, July 2007.

[16] M. N. Nicolescu, A. Olenderski, R. Leigh, S. J. Louis, S. Dascalu, C. Miles, J. C. Quiroz, and R. Aleson, "A training simulation system with realistic autonomous ship control," *Computational Intelligence*, vol. 23, no. 4, pp. 497–519, 2007.

[17] "Wikipedia: Flight Simulator," "http://en.wikipedia.org/wiki/Flight_simulator".

[18] "Serious Games by BreakAway," "www.breakawaygames.com".

[19] "The Stock Market Game," "http://www.stockmarketgame.org".

[20] *NS-2*, http://www.isi.edu/nsnam/ns.

[21] T. Ye and S. Kalyanaraman, "A recursive random search algorithm for network parameter optimization," *Proc. of ACM SIGMETRICS*, vol. 32, pp. 44–53, December 2004.