Network Management Game (NMG)

A thesis submitted in partial fulfillment of the requirements for the degree of Master of Science in Computer Science

by

Engin Arslan

Murat Yuksel/Thesis Advisor

August 2011
We recommend that the thesis prepared under our supervision by

**ENGIN ARSLAN**

entitled

**Network Management Game**

be accepted in partial fulfillment of the requirements for the degree of

**MASTER OF SCIENCE**

Murat Yuksel, Advisor

Mehmet Gunes, Committee Member

Ramona Houmanfar, Graduate School Representative

Marsha H. Read, Ph. D., Dean, Graduate School

August, 2011
Network Management Game (NMG)

Engin Arslan

University of Nevada, Reno, 2011

Supervisor: Murat Yuksel

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 actual 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 user. We applied our game a small set of users using two different training methods. During the first training method, we observed %2.2 and %16 user performance improvement in cases where link failures are simulated and
not simulated, respectively. We also observed that the users are more successful on network management than automated solutions when the system exhibits exceptional situations like link failures. In the second user experiment, we trained users on two skills which may help the players for better network management. After training on these two skills, the users performed %13-%21 better network management compared to their performance before training.
Acknowledgments

I would like to express my sincere gratitude for my advisors Dr. Murat Yuksel and Dr. Mehmet H. Gunes, whose guidance, inspiration, understanding, patience, and encouragement added considerably to my graduate experience. I would like to thank Dr. Ramona Houmanfar for agreeing to be in my thesis committee despite her busy schedule and for her advices on conducting my experiments.

I also would like to thank all of my colleagues in Computer Networking Lab including Mehmet Bilgi, Hasan Tarik Karaoglu, and Hakan Kardes for their valuable advices during my research.

I also would like to thank my family and my fiancee, Ilknur Yavuzalp, for the support they provided me through my entire life.

Engin Arslan

University of Nevada, Reno

August 2011
Contents

Abstract i

Acknowledgments iii

List of Tables vi

List of Figures vii

Chapter 1 Introduction 1

Chapter 2 Literature Survey 6
  2.1 User Training ........................................ 6
  2.2 IGP Link Weight Setting ............................... 9
  2.3 What-if analysis ................................. 12

Chapter 3 Network Management Game Framework 15
  3.1 The Animator ........................................ 17
  3.2 Engine-Animator Interaction ...................... 19

Chapter 4 Experimental Setup 22
  4.1 Game Goal: Tuning IGP Link Weights for Load Balancing 22
4.2 Training Mechanism ................................................. 25
  4.2.1 Training without Mastery ........................................ 26
  4.2.2 Training with Mastery .......................................... 30

Chapter 5 Experiment Results 41
  5.1 Results of Training without Mastery Method ..................... 41
  5.2 The Difference of Training ......................................... 42
  5.3 The Best and The Worst Players ................................... 46
  5.4 Results of Training with Mastery Method ........................ 49

Chapter 6 Summary and Future Work 52

Bibliography 54
List of Tables

4.1 Details of Training Scenarios of First User Experiment . . . . . . 27
4.2 Details of Training Scenarios of Second User Experiment . . . . . 32
List of Figures

1.1 IGP path selection example. Link weights are same at the beginning. Traffic flows from A to E. 3

3.1 Block diagram of Network Management Game (NMG) components. Event sequence starts with (1), goes on with (2), (3). Whenever the player makes changes, the sequence (4), (5), (6) and (7) repeats. 16

3.2 One of the training test cases in which there are two flows. Sources and destinations are marked by color and node text. 17

4.1 IGP path selection example. Link weights are same at the beginning. Traffic flows from A to E. 23

4.2 Network topology with TCP traffic from A to D. 24

4.3 One of the training test cases (#3) in which there are two flows. Sources and destinations are marked by color and node text. 27

4.4 Training stages: Tutorial, Before Training, Training, After Training. 27

4.5 Topology used in the test cases #6 and #6’. 28

4.6 Topology w/ failures used in the test cases #7 and #7’. 28

4.7 TCP traffic flows from node A to node D. 30
4.8 There are 2 traffic flows from node A to node D and from node B to
node D . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 32
4.9 Test scenario played before training . . . . . . . . . . . . . . . . . 33
4.10 Training Scenarios (#1-#2) . . . . . . . . . . . . . . . . . . . . . . 34
4.11 Training Scenarios(#3-#4). . . . . . . . . . . . . . . . . . . . . . 34
4.12 Training Scenario Level #5 . . . . . . . . . . . . . . . . . . . . . . 36
4.13 Training Scenario Level #6 . . . . . . . . . . . . . . . . . . . . . . 37
4.14 Training Scenario Level #7 . . . . . . . . . . . . . . . . . . . . . . 38
4.15 Test scenario played after training . . . . . . . . . . . . . . . . . . 39
4.16 Training stages: Tutorial, Before Training, Training, After Training . 40
5.1 Before training user performances compared to the best and the worst
possible throughput. . . . . . . . . . . . . . . . . . . . . . . . . . . . . 43
5.2 Before training user performances compared to the best and the worst
possible throughput (Failure included). . . . . . . . . . . . . . . . . . 43
5.3 After training user performances compared to the best, the worst and
two black-box optimization algorithms. . . . . . . . . . . . . . . . . . 44
5.4 After training user performances compared to the best, the worst and
two black-box optimization algorithms (Failure included). . . . . . 44
5.5 Performance of the worst player on the test case with failure (case #7) 46
5.6 Performance of the worst player on the test case with failures (case #7’). 47
5.7 Performance of the best player before training with failure (case #7) 47
5.8 Performance of the best player after training with failure (case #7’) . 48
5.9 Comparison of users’ performance before and after training . . . . . . . 50
Chapter 1

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 a huge amount of traffic on a broken pipe.
Though there have been several tools and outcomes 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 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 thesis, we propose the concept of “network management 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. 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.
Figure 1.1: IGP path selection example. Link weights are same at the beginning. Traffic flows from A to E

In Figure ??, assume that there is a traffic flow from node A to node E. By default, IGP calculates path over node D because of its shortest path selection algorithm. At this point, network administrator may want to change default path of the flow for some reason (e.g. low bandwidth capacity, a failure, or scheduled maintenance). The way to do this is manipulating weights of the links in such a way that the total cost of an alternative path (A-B-C-E in this case) becomes shortest among all possible end-to-end paths. Once the administrator updates the link weights in this manner, the traffic reroutes to this shorter alternative path which allows the administrator to load balance the traffic in an indirect way. Such link weight setting is a common practice in ISP operation, but finding the optimal link weights is known to be a very difficult problem of NP complexity.

In this research, we apply our gaming framework to the problem of IGP link weight setting since it is a complicated enough problem to train someone on. We leverage existing simulation tools to establish a simulation engine for our prototype game and build an animator that interacts with the simulation engine in real time. The
user/player\textsuperscript{1} 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.

To test usability of our framework, we conducted two user experiments to train the players and measured performances of the players before and after training. One of them is “training without mastery” in which players are trained with five different network scenarios for a fixed amount of time on each without any success level requirement prior to proceeding to the next. In the second training method, we focused on frequently used skills in IGP link weight setting and aimed to train the players on those particular skills. We developed seven training scenarios to represent seven levels and the players were required to perform certain success in each level in order to advance next level. We applied both training methods on a small set of players and analyzed the results which are compromising to prove effectiveness of our training framework.

For the first user experiment, eight players participated and training on IGP link weight setting is carried out. The results of training show that the performance of the players improved %16 in scenarios in which link failures are not simulated and %2.2 in scenarios in which link failures are simulated. We also compared the performance of the players with performance of black-box optimization algorithms such as Genetic Algorithm on the failure-included scenarios. The result of this comparison showed that the players obviously outperformed to optimization algorithms.

In the second user experiment we selected five players among those who participated the first user experiment. Since the players got accustomed to the framework

\textsuperscript{1}The word “user” and “player” refer to each other throughout the thesis.
through the first experiment, we were able to eliminate possible performance improvements caused by framework orientation. Moreover, results of the second user experiment indicate that all players become more successful on finding a close-to-optimal IGP link weight setting compared to their performance before training. Although percentage of performance improvement varies from player to player, we observed %13-%21 increase in finding close-to-optimal solution (Figure ??).

In Chapter 2, we present the related work in IGP link weight setting problem, what-if analysis and user training. In Chapter 3, we present the Network Management Game Framework where the design and the working principle of our tool is explained. In Chapter 4, we give details of the user experiments including how we designed the training and test scenarios. Then, in Chapter 5, we present the experimental results and interpret them. Finally, in Chapter 6, we conclude the thesis and provide a brief overview of future work to enhance the Network Management Game.
Chapter 2

Literature Survey

In this chapter, we review the related works about user training, what-if analysis and IGP link weight setting problem.

2.1 User Training

Training is one of the most prevalent methods to promote productivity. Currently, most of the basic training for a network administrator is performed by means of well-defined certification procedures [?]. 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. To the best of our knowledge, usage of a game for training network administrators was not done before.
Thus, our tool would be the first in this area and it will help those who want to be a network administrator and service providers to train their network management operators.

In [?], authors present that the sample weighted effect size of organizational training on these behavioral enhancements was 0.60 to 0.63, which means medium to large effect size for organizational training. The businesses spend an enormous amount of money on end-user training. In 2010, U.S. organizations with 100 or more employees budgeted to spend $52.8 billion [?].

Effectiveness of training method highly depends on active participation, behavior modeling and manuals that encourage exploratory learning and training previews [?,?,?,?]. For these reasons, we tried to make our tool user friendly, easy-to-use and game-like environment. Furthermore, we give a tutorial before carrying user tests where we explained details of the tool; how it works, how interactivity is supported and how user can interpret information related to current simulation state presented on the tool such as throughput values and dropped packet information areas.

Compeau et al. [?] proposes key factors to consider in the management of end-user learning and training. According to the training model they propose, there are three main phases which are \textit{initiation phase}, \textit{formal training and learning phase}, and \textit{post-training phase}. In the initiation phase training, requirements of the training are determined and training materials are developed to meet identified requirements. Training and its evaluation are performed in second and third phase, respectively. In our second user training (training with mastery), we applied proposed training model. That is, we first decided the requirements for training the users on IGP link weight setting and developed training cases. Afterwards, we performed and evaluated training in phase 2 and 3. Figure ?? shows the stages of our second user training.
In behavior science, Skinner claimed that Programmed Instruction (PI) is more efficient and effective than other traditional instructional approaches [?]. According to the research done by Schramm [?], studies incorporating programmed instruction as the method of learning contributed more to trainees on specific skills to learn and perform compared to studies which used other methods. Moreover, in [?], author proposed that programmed instruction does not only lead increase performance on targeted skills, but also increase in performance to handle problems that are not existed in prior training or instruction existed. Similarly, what we aimed to do in this research is training players on sample network topologies so that network management capability of the players improves for any network topologies.

Behavioral fluency is defined as combination of accuracy plus speed of responding that enables competent individuals to function efficiently and effectively in their natural environments [?]. Binder focuses on how learned skills can become second nature and he proposes that complex skills are separated into simple skills and each simple skill is taught until trainee becomes fluent on that skill. Then, skills are combined and taught until user becomes fluent in combined skill. The way we follow in second user experiment is very similar to this method. We first trained players on two skills separately until they become fluent on those skills. Then, the players are trained on scenarios in which targeted skills are combined so that they can experience on more realistic cases where multiple skills are needed at the same time to manage network efficiently.

The concept of using a virtualized game-like environment for training is not new. [?, ?] 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 [?], commander training [?]. Financial investment training [?] is another
venue where a game-like environment can be used for training before deploying money on stock market.

### 2.2 IGP Link Weight Setting

Interior Gateway Protocol (IGP) link weight tuning problem has been of interest to research and Internet Service Provider (ISP) communities [?, ?, ?]. 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 [?, ?, ?, ?]. Due [?] showed that finding optimal solution of this problem is NP-hard, studies use heuristic algorithms. [?, ?, ?] use hybrid genetic algorithms to optimize link weights while [?] uses Recursive Random Search (RRS) and [?] uses both genetic algorithm and RRS at the same time.

In [?, ?], authors target to tune optimal IGP link weight settings while network is in operation. They claim that as scale of network is increasing OSPF ¹ is overloaded with new functions such as traffic engineering ². ISPs generally trust network operators/administrators in order to tune of link weights to meet traffic engineering objectives. Authors alternatively suggest automated solutions to tune IGP link weight settings. Methods of the papers differ in terms of cost function. While [?] takes packet loss rate into consideration, [?] chooses aggregate network throughput as optimization metric. [?] formulates packet loss in terms of link parameters such as bandwidth and then GI/M/1/K queuing model is used to compute the packet drop.

¹OSPF is widely-used Internet Gateway Protocol (IGP).
²Traffic Engineering corresponds the use of different technologies to increase utilization of a given network topology.
probability on a given link in the network. However, [?] offers hybrid method where they benefit from 3 black-box optimization algorithms (Genetic Algorithm, Recursive Random Research and Simulated Annealing). They divide given time period into turns and at the beginning of each turn each algorithm calculates its own optimal link weight settings and let to run for a given time unit. Aggregate throughput of each time unit is then calculated. Afterwards, rest of the turn allocated among algorithms proportional to their performance in aggregate throughput. For the next turn, algorithms make use of link weight setting of the best performed algorithm of previous turn and again calculate their own best solution. This “roulette wheel” approach is able to select the more successful algorithm in the next round.

Moreover, In [?], authors points out how network utilization can be increased by separating unicast and multicast traffics in the same network. The method they offer is setting IGP link weights in such a way that even if unicast and multicast server collocate on same node and target to same destination, routes of these two traffics do not overlap. They make use of difference in path calculating method for unicast and multicast traffics. Unicast routes are obtained by the calculating shortest path from source to destination whereas multicast tree is formed by calculating shortest path from all destinations to source.

[?] proposes 2-phase solution for OSPF flow allocation problem. In Phase 1, demands are allocated using Simulated Allocation and Evolutionary Algorithm to a single paths. Mixed Integer Programming is used in this phase to avoid link weights leading to more than one shortest paths from source to destinations. In Phase 2, Linear Programming is used to find a corresponding weight system that generate the set of single demand allocation paths obtained in Phase 1. Although this method can outperform to heuristic algorithms (e.g. Local Search, Simulated Annealing and
Lagrangian Relaxation), it is not feasible as scale of network exceeds 100 nodes.

Failure of links is another issue in networks that needed to be handled properly in order to avoid performance degradation and service disruption because [?] showed that it occurs as a part of everyday operation. [?] claims that ISPs has to know characteristics of link failure to ensure promised Service Level Agreement. The importance of link failures increased as new applications emerges such as Voice-over-IP (VoIP) due such applications are adversely affected if packets are dropped or delayed due to failures. Hence, we simulated link failure event in our experiments in order to see how users can react such unusual but frequent events.

[?, ?, ?] offers solution in case of link failure occurrence in network. [?] give solution for a single link failure in local networks. The reason that they addresses a single link failure is failures are usually short-lived and recovered in a short time. They come up with a solution in which optimal link weight settings of all single link failure cases are computed and recorded. Whenever a link failures, corresponding optimal setting is found quickly and deployed. [?] similarly focused on single link failure cases. Their solution differs in terms of base point. Instead of deploying same routing paths to all routers in network, they precompute new routing paths for each router separately. Thus, network can continue forwarding packets after constant and small time period and avoid inconsistent forwarding which occurs in usual OSPF/IS protocol within a time span of link fails and new routing is calculated. [?] propose a Tabu-search heuristic for choosing link weights in case of link failure. They also take single link failure into consideration and developed a method to achieve better load balancing within failure interval. Their approach comes with 10% of performance degradation in absence of link failure. However, method achieve 40% performance improvement when a link fails in the system.
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 [?]. Although what-if analysis has high impact within business intelligence platforms, its usage is extended for several purposes such as hazard analysis [?], index selection for relational databases [?], Content Distribution Networks (CDNs) [?], and multi-tier systems [?]. For instance, in [?], the authors developed a tool (WISE) that predicts how a deployment of a new server to an existing CDN affects service response time. They use machine learning techniques to process old dataset and discover the dependencies among system variables. Then, using these dependencies and new dataset which is representative of new deployment, WISE predicts how response time could be affected when deployment changes are done.

Another example of in-house what-if analysis tool is used to estimate network latency in multi-tier systems. In [?], the authors aim to predict change in network latency when location of system components change. They define *link gradient* as an effect of a logical link latency on per-transaction response time. Since response time depends on many factors such as workload in networks and fluctuates too much, delay injection and spectral analysis methods are used to obtain a level of precision. Then using Fast Fourier Transform response time series with delay injection and without delay injection are calculated. Based on two response time series and formula
they used, link gradient of the each logical link is computed for each transaction. When a system component moved to another location, the new response time of the transaction can be computed based on link gradient values of shifted links.

Beyond the what-if analysis tools, there has been massive interest on characterizing performance issues [?, ?, ?, ?]. Performance prediction provide great opportunity to operators in order to manage resources more efficiently. Along with emerging technologies such as IPTV and VoIP, service providers are required to manage the network more carefully because of reliability issues. Thus, they try to minimize uncertainty in network to prevent deterioration in service quality. Uncertainty generally originated from changes in network structure such as relocating a web server or firmware upgrade of routers. Many automated tools are developed to minimize such uncertainties. Li et al. [?] presents a tool, WebProphet, that predicts performance of web services accurately. The method they propose based on timing perturbation to extract web object dependencies then uses these dependencies to predict the performance impact of changes to the handling of the objects. Timing perturbation used to discover the delay relations of web objects and so dependencies. Once dependencies are extracted accurately, it is trivial to predict new scenario. Instead of using server-side to evaluate performance (e.g. [?]), they take advantage of Page Load Time (PLT) due it is more realistic approach in terms of user perceived performance. This method is claimed to predict statistical properties (e.g., median or %95percentile) of the PLT distribution in a new scenario with %16 error probability.

Chun et al. [?] similarly does performance evaluation and prediction process for performance of programs. They use machine learning techniques to identify features that affect program performance. Then, prediction of performance model is formulated as a function of selected features. It finally generates compact code slices
to calculate these feature values for prediction. Mahimkar et al. [?] targets IPTV network to characterize performance issues and similarly make use of old dataset they try to predict and troubleshoot the problems that affect network performance. In [?], authors implemented a tool called MERCURY which identifies network anomalies caused by firmware or operating system upgrade of network element including router, switches etc. The motivation point of this tool is how network upgrade based unexpected behaviors in large operational network can be identified such that system administrators can fix problem before it leads big problems in network performance. Mercury benefits from skewness and rareness properties of configuration to identify anomalies in network caused by upgrades.

Furthermore, various tools have been developed to guide investments and determine how to improve network performance [?] with minimal investments. Though existing what-if analysis and performance prediction 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.
Chapter 3
Network Management Game
Framework

The NMG framework has two parts as shown in Figure ???. 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 engine on real-time. In our prototype, we used NS-2 [?] 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) by writing Tcl scripts, and then NS-2 simulates the scenario. However, NS-2 provides no 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

\(^1\)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.
Figure 3.2: One of the training test cases in which there are two flows. Sources and destinations are marked by color and node text.

3.1 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. In order to show the first 2 of theses events on the animator, we adjusted the color of the links based on their load, as shown in Figure ???. 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. For each link pair (incoming and outgoing), there is a button to increment-decrement link weight. When player changes weight of a link, this change affects the incoming and outgoing link pair, simultaneously. Also, player can see instant and cumulative throughput of simulation as well as maximum possible throughput for that simulation. Instantaneous throughput helps the player to compare his/her current performance by comparing it with the maximum possible throughput. Cumulative throughput helps to observe player’s overall performance. The difference of calculation method for instant and cumulative throughput is explained in Section ??.

Moreover, simulation time and packet loss information are displayed in the left bottom part of the GUI. Simulation time may be useful to observe difference of real time and simulation time in terms of speed which is explained in ?? . As a packet loss information we only considered loss caused by buffer overflow which is caused by burdening a link more than its capacity. Thus, it helps player to see which link is overloaded at a given time so, she may want to step in for better load balancing.

Furthermore, the design of interface is done in a way that player 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 their current utilization where links with no traffic on it are green colored whereas completely utilized links are red colored of a link. Capacity of a link and its current utilization can be viewed by focusing it via mouse. Also, instantaneous throughput is displayed to inform the player about the effects of changes in link weights.
3.2 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 player 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 these events of the previous period, the simulation engine runs the next period and waits until the GUI finishes animating the events of previous period. Thus, the simulation engine always runs one period ahead compared to the GUI. If the player makes a change on the link weights, the simulation engine is informed about it just after the current period finishes. The drawback of this solution is that the synchronization is not achieved completely because whenever the player wants to make 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 chose to reduce the speed of the NS-2 engine. The simulation engine waits a fixed amount of time before moving to the next event to process. We tuned the amount of wait time such that the engine runs concurrently with the GUI. By this way, whenever the player updates a link weight, the change is sent directly to the simulation engine and new routing paths are calculated before engine dequeuing next event from event queue\(^2\). Thus, the simulation resumes with new network configurations.

\(^2\)NS-2 is a discrete event driven simulation tool so events are queued in a list with respect to their event time [?].
We used TCP port in order to provide interaction between animator and engine. Two ports are allocated for this communication. One of them is opened by the animator only to sent traces to GUI. Thus, GUI always listens this port and read traces sent by engine. Second port is opened by GUI to send link weight manipulations to the animator. Similar to the other port, animator always listens to receive messages sent from GUI. Whenever a message is received from GUI, animator executes it before moving next event in event queue. Priority of processing the received message enables real-time interactivity.

In our current framework, there are two types of messages sent from GUI to animator. These are:

- **Set Link Weight**: This message is the only one used for manipulating link weights. When player increments or decrements a link’s weight, corresponding message is prepared in TCL format which provides animator to execute it directly. Then, TCL command is sent to animator to be executed. “\$ns cost $node1 $node2 3” is an instance for this type of message which implies to set cost of the link that lies between node1 and node2 to 3. Right after executing received link weight manipulation command, animator executes the command that calculates routing of flows with new link weight parameters. So, new routes are calculated according to OSPF shortest path algorithm and declared to all nodes.

- **Set Simulation Speed**: This message controls speed of animation. In fact, this message is called only at the beginning of simulation to tune simultaneity. Number of messages to transfer from animator to GUI increases when network scale increases, which causes too much waiting period of animator because animator
waits fixed amount of time after every event process. Hence, this leads too much slowdown of animator. To handle this slowdown, we sent appropriate waiting time for each topology. That is, for smaller topologies waiting time is set to higher values compared to one in bigger topologies.
Chapter 4

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.

4.1 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 so that the end-to-end shortest paths change, and thus the load on individual links are changed. Figure ?? 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 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 ??). We monitor cumulative and instantaneous net-
Figure 4.2: Network topology with TCP traffic from A to D.

work throughput to evaluate the player’s performance. The cumulative throughput, $\tau_{\text{cum}}$, is mainly used to see how the player performed in the overall test whereas the instantaneous throughput, $\tau_{\text{ins}}$, is used for measuring how well the player responds to dynamic changes in the network such as failures or demand spikes. We calculated these throughput values as follows:

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

where $P_s(i)$ represents the total size of packets which are successfully transferred from source to destination within $i$th 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 as traffic type since it utilizes the maximum available bandwidth on its path. This makes the game more interesting even if there is no failure in the network, as the player has to guide the TCP flows to accumulate higher throughput. For example, in Figure 4.2, there is 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.

Player performance is measured by aggregated throughput. It is given in three different formats. Instantaneous throughput shows instant throughput whereas Exponential Weighted Moving Average (EWMA) throughput uses instantaneous throughput and make it more smoother due instantaneous one fluctuate very fast. EWMA throughput is calculated using formula below where it mostly depends on previous value of itself while calculating current value.

\[
EWMA_t = 0.7 \times EWMA_{t-1} + 0.3 \times Inst_t
\]

The throughput values collectively give a good idea of how fast the player finds paths that increase the network utilization.

4.2 Training Mechanism

We carried out two types of training methods in our experiments. First of them is “training without mastery” in which player trained with 5 different network topologies with different difficulty levels, successively. In other words, a player played each level for a pre-defined time period and passed next level regardless of her performance in current level. At the end, we drew player performance graphs based on initial and final performance on Abilene backbone topology. Second training method is “training with mastery” where players trained with 7 different levels however, in order to pass the level mastery criteria is obligated. For this method we focused on two basic skills
as a mastery criteria which are commonly used by system administrators to increase network utilization. These skills are high bandwidth path selection and decoupling of flows.

4.2.1 Training without Mastery

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 ??, 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 ?? as an example. There are two TCP traffic flows and bandwidth of links are different from each other. We increased or decreased the width of the lines based on the links’ 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.
Table 4.1: Details of Training Scenarios of Firs User Experiment

<table>
<thead>
<tr>
<th>Test Case #</th>
<th># of Flows</th>
<th>Time (mins)</th>
<th>Minimum Link BW (Mb/s)</th>
<th>Maximum Link BW (Mb/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>5</td>
<td>2</td>
<td>6</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>6</td>
<td>1</td>
<td>8</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>6</td>
<td>2</td>
<td>6</td>
</tr>
<tr>
<td>6</td>
<td>4</td>
<td>6</td>
<td>2</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td>4</td>
<td>6</td>
<td>1</td>
<td>8</td>
</tr>
<tr>
<td>6'</td>
<td>4</td>
<td>6</td>
<td>1</td>
<td>6</td>
</tr>
<tr>
<td>7'</td>
<td>4</td>
<td>6</td>
<td>1</td>
<td>6</td>
</tr>
</tbody>
</table>

Figure 4.3: One of the training test cases (#3) in which there are two flows. Sources and destinations are marked by color and node text.

Figure 4.4: Training stages: Tutorial, Before Training, Training, After Training
Figure 4.5: Topology used in the test cases #6 and #6'.

Figure 4.6: Topology w/ failures used in the test cases #7 and #7'.
Training Stages

We applied a staged training process to observe efficacy of NMG in training network management skills. As illustrated in Figure ??, we first gave a tutorial to 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 players 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. Also, link failure cases are not taken place in training stage. We used the Abilene backbone topology (shown in Figure ??) 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. We also tested the players with the Abilene topology but include network failures. As an example, in Figure ??, 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.
4.2.2 Training with Mastery

In this training method, we targeted the skills that may be helpful in solving the IGP link weight setting/management problem. Although there are several skills used by network administrators, we just focused on 2 of them. These are high bandwidth path selection and decoupling of flows.

**High bandwidth path selection**

First skill we aimed to train is to flow traffics on links with high capacity in order to maximize overall throughput \(^1\). Since we used TCP traffics in our experiments, throughput of traffics depends on available bandwidth on the path. As in Figure ??, when player manipulates link weights to flow traffic on path A-B-D then throughput would be 1Mb/s. On the other hand, alternative path A-C-D can increase throughput to 3Mb/s. Thus, we expect players to make appropriate link weight changes to flow traffics along the path which maximizes throughput.

\(^1\)Overall throughput is calculated by summing individual throughputs achieved by each TCP flows
Decoupling of flows

Second skill that we targeted is decoupling of flows that are passing over at least one common link. In Figure ??, there are 2 TCP flows from node A and node B directed to node D. It seems that if flows follow path over A-C-D and B-C-D then, throughput for each traffic may reach 4Mbps. However, when both traffic flows pass over link C-D, capacity of this link will be shared which then limits overall throughput to 4Mbps. The skill of decoupling of traffic flows plays an important role in cases where separation of paths of traffic flows may increase overall throughput. Obviously, decoupling of flows may not always be able to increase overall throughput. If the capacity of shared link is too high to become bottleneck, then decoupling of flows will not bring increase in overall throughput. It is also important to note that even high bandwidth path selection and decoupling of flows help to increase utilization \(^2\), network administrators usually face cases where experience of both of the skills are required to increase resource utilization. Hence, we first trained players on each skill separately. Then, we trained the players on cases where they have to experience the both skills to find optimal IGP link weight setting.

Test Cases

In “training with mastery” method we focused on skills and redesigned training levels to target specific skills. Hence, training levels are better designed in this method. We again started with complex network topology and performance of the players are recorded. Afterwards, the players played 7 training scenarios where in each level the players are expected to find optimal solution of routing via setting IGP link weights within a given time period. If a player is unable to find optimal routing for a level

\(^2\)Increase in utilization can be used as meaning of increasing overall throughput in this research
Figure 4.8: There are 2 traffic flows from node A to node D and from node B to node D.

Table 4.2: Details of Training Scenarios of Second User Experiment

<table>
<thead>
<tr>
<th>Scenario#</th>
<th>Number of flows</th>
<th>Time (min)</th>
<th>Average Play Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>2</td>
<td>1.1</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2.1</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>2</td>
<td>2.3</td>
</tr>
<tr>
<td>5</td>
<td>3</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td>4</td>
<td>2</td>
<td>3.1</td>
</tr>
<tr>
<td>7</td>
<td>4</td>
<td>3</td>
<td>4.5</td>
</tr>
</tbody>
</table>

within the given time period, then s/he fails and is required to play again that level until optimal routing is discovered. Whenever the player successes the level, s/he has a right to pass next level. Mastery criteria is obligated since we want players to have enough experience of skills at each level. When the player finished all training scenarios, then s/he played the test scenario which is similar to the one s/he played at the beginning in terms of network topology. Test scenarios that the player played at the beginning and at the end of the training are used to evaluate player performances thus, layers play those for a given time period without any success level requirement. This training method obviously took longer due players mostly played each level more than once. In Table ??, number of flows, time limit and average play count is listed.
The players are tested with Abilene backbone topology as in Figure ?? to see their performance at the beginning of the training session. They are given 4 minutes to manage IGP link weights. Performance of the players in the testing case recorded to compare it with performance of the player after the training session on similar testing case.

Since the design of training scenarios in “training with mastery” method is important to target two skills, we go over them and explain details to show how well they are designed. Figure ??(a) is the first and simplest training topology. In this simple topology, we expect that the players can learn environment and understand how routing path of flows are calculated in IGP. Furthermore, players are required to route flow on a path which increases current overall throughput to target overall throughput. By default, we set equal weight for all links. As can be noticed in Figure ??(a), there is only one flow with three possible paths. Skill of high bandwidth path
In figure ??(b), there is again one TCP flow in simulation. However, finding optimal link weight solution that boosts instantaneous throughput to target one is not as simple as level #1. Skill of high bandwidth path selection is the one that the players have to experience to find optimal routing solution for this topology.

In Figure ??(a) (level#3), there are two TCP flows and many alternative paths for these flows. Due number of flows are more than one and there is possibility
of passing on common link in same direction, skill of decoupling flows has to be experienced. The players are required to perform many link weight manipulations in order to find optimal routing. Directing paths on links with high capacity solely does not give optimal solution because both flows pass over same link which causes bottleneck for each flow. Thus, player has to find a path for each flow in a way that paths do not pass over on a same link as well as they flow on links with high capacity. Since there are only two flows, it is not too hard to achieve decoupling of flows and high bandwidth path selection simultaneously.

Figure ??(b) shows level #4 where simulation contains three TCP traffic flows. Destination and source nodes of the traffics are very close to each other which leads flows to follow similar paths. Thus, players have to experience on decoupling of flows in this level. Furthermore, first(S1) and third(S3) traffic flows are originating from same node (left-top node) and destination nodes are located very close to each other. Thus, decoupling their paths is complicated in this case. The players are required to make calculations to find out total weight/cost of each path in order to identify the link to change its weight which will decouple path of first and third flows. Thus, this level may give idea about complexity of changing link weights in real network topology where there are several traffic flows goes on same link and changing weight of a link may affect many traffic flows simultaneously.

In level #5 (Figure ??), there are again three TCP flows and network topology is more complex than level #4. Since number of alternative paths from sources to destinations are more than the previous levels, it is harder to find optimal solution for this level. High bandwidth path selection is again important part of solution due default paths are generally pass over links with low capacity. Similarly, decoupling of flows plays an important role in finding optimal solution because traffic flows 1
Figure 4.12: Training Scenario Level #5
Figure 4.13: Training Scenario Level #6

(S1) and 2 (S2) are destined to same node which result flowing over a common link so that the link becomes a bottleneck. Furthermore, the player has to recognize the fact that the reverse directional flows on a link does not cause link capacity sharing because optimal solution of this level includes paths where two TCP flows pass over same link but in reverse direction.

Figure ?? shows level #6 where there are four TCP flows. As can be seen in figure, source and destination pairs are scattered all over the topology thus, flow decoupling is key part of the solution. Also, throughput of optimal solution (target throughput) is more than double of default throughput ³ which necessitates many link weight manipulations. When we consider two minutes time restriction to achieve this level, player has to have satisfied level of experience of the two skills.

Figure ?? is the last and the most difficult level of training session. There

³Default throughput stand for throughput of the simulation when it runs with an initial network configuration.
Figure 4.14: Training Scenario Level #7
are four TCP flows and destinations of flows are scattered. Selecting path with high bandwidth and flow decoupling are both important to find optimal link weight setting for this topology. Capacity of links does not differ too much (that can be recognized by their widths)\textsuperscript{4} which complicates optimal routing since it requires intense analysis of paths to find the path with high bandwidth. Player requires to find maximum throughput that each flow can have in order to reach targeted overall throughput.

At the end of the training session, the players are again played Abilene backbone topology as in Figure ?? where they are given four minutes to find close-to-optimal solution for IGP link weight setting. There are four TCP flows similar to the one the players are played at the beginning. This time, however, source and destination pairs of flows and link capacities are different.

\textsuperscript{4}As mentioned in chapter ??, width of a link is proportional to its capacity.
Figure 4.16: Training stages: Tutorial, Before Training, Training, After Training

Training Stages

We applied again a staged training process. As illustrated in Figure ??, we start with a tutorial to the player about the environment and assured that the player knows what the game is about and how the GUI works. Then, player starts with a relatively complex network (i.e., test case #8) to observe how s/he performs before the training. Next, we trained the player with training cases #1-#7 with each case having a fixed amount of time. After the training stage is over, we exposed the player to test case #8’, which is similar to the test case #8 in terms of difficulty.
Chapter 5

Experiment Results

We performed two user experiments with different objectives and methodology. Thus, we present their results separately.

5.1 Results of Training without Mastery Method

To observe how well our game environment performs in training people, we experimented with 8 users and monitored their performances. We kept track of the throughputs the users achieved throughout the game. We compared the performances of the users 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) [?] 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.

We also applied T-test for the first user experiment. Since we tested the players on test cases with and without failure separately (#6-#6’ and #7-#7’), we calculated 2 p-values. P-value for test cases #6 and #6’ is 0.000243. This value is much smaller than threshold 0.05 to be statistically important thus we can conclude that the Training stage affect the user performance in network management. P-value for test case #7 and #7’ is 0.2788157 which is higher than threshold value to be statistically important. Since we did not train the players on link failures in the Training stage, user performances on link failure cases did not improve significantly.

5.2 The Difference of Training

Figures ?? and ?? 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.
Figure 5.1: Before training user performances compared to the best and the worst possible throughput.

Figure 5.2: Before training user performances compared to the best and the worst possible throughput (Failure included).
Figure 5.3: After training user performances compared to the best, the worst and two black-box optimization algorithms.

Figure 5.4: After training user performances compared to the best, the worst and two black-box optimization algorithms (Failure included).
Figures ?? and ?? 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 16% 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 2.2% improvement from the training. This is sizably smaller than the improvement observed for the test case without failures. The reason of smaller improvement can be related to training scenario design since we did not simulate link failure in training stage. Taking to performance increase in test cases without link failures into consideration, we can claim that training stage is successful because we observed 16% performance improvement on test cases that the players are trained on whereas this improvement is limited to 2.2% in test cases that the players are not trained on.

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 user 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.
5.3 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, $\tau_{ins}$, and cumulative, $\tau_{cum}$, 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 $\tau_{ins}$ lines fluctuate frequently due to the adaptive behavior of the TCP flows.

Figures ?? and ?? 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
Figure 5.6: Performance of the worst player on the test case with failures (case #7’)

Figure 5.7: Performance of the best player before training with failure (case #7)
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 ??.

Figure ?? and ?? 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 $\tau_{\text{ins}}$ at around time 2s and 5.5s).

It is interesting to observe in Figure ?? that the worst player’s instantaneous throughput goes below the No Player case. This is due to the fact the player, prior to the first failure, carried some of the traffic flows to the link which failed at the 3rd second. Comparatively, the best performing player in Figure ?? also moves the traffic

Figure 5.8: Performance of the best player after training with failure (case #7’)

\[
\begin{align*}
\text{Throughput (Mb/s)} & \quad \text{Time (sec)} \\
\tau_{\text{ins}} & \quad \tau_{\text{cum}} \\
\text{No Player} & \quad \text{Link Down}
\end{align*}
\]
flows around to achieve a higher throughput before the first failure, but also achieves
to restore the high throughput even after the first failure.

5.4 Results of Training with Mastery Method

In the second user experiment, five users are trained. The users are selected among those who attended first user experiment. Thus, they were aware of framework regarding to what to do, and how to do. It was also important to observe performance improvement of the users by means of training because, in the first user experiment user performance improvement might be related to framework orientation to some extent.

Users’ performance on Abilene network topology at the beginning and at the end of the training is given in Figure ?? in the form of proportion to optimal solution. In other way, if we say $\tau_u$ maximum overall throughput user reached and $\tau_{opt}$ maximum overall throughput that the optimal solution for the given topology could reach, then $Perf$ can be calculated as follows:

$$Perf = \frac{\tau_u}{\tau_{opt}} * 100$$

Regarding to user performance changes, due we used small scale topologies for testing purpose, users generally could go up to %70 of optimal solution before training. However, no one could were able to go over %80 of optimal at the beginning. After the training session, we observed remarkable increase in finding close-to-optimal solution for every user. Every user was able to find solution that goes more than %80 of optimal one. Improvement of user performances varies from %13 to %21 in terms of finding close-to-optimal solution. These results are promising because we only
Figure 5.9: Comparison of users’ performance before and after training
focused on 2 frequently used skills for IGP link weight setting and tested on simple topologies, and traffics. We believe that further improvement in framework and user training methodology would lead to better user performance improvement. The idea that we want to emphasize is trainability of people on IGP link weight setting using interactive simulation engine. When we applied t-test for the user performances on the test cases before and after training, p-value is computed as 0.000013 which is very low compared to threshold value 0.05. Thus, we can claim that “training with mastery” method is effective to train users on IGP link weight setting.
Chapter 6

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 developed tool for training. We carried out two user experiments. In the first one, we trained users on 5 scenarios where scenarios differ in terms of network complexity. We also simulated link failure cases in this experiment to see how users can react in unusual situations. In the second user experiment, we 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 7 scenarios and evaluated the players on two relatively complex test cases. 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. Visualization of large-scale network can be handled by dividing network topology into layers such as AS Level and PoP-layer. Thus, user is able to view and manage whole network by focusing in and out. Further, maintaining the realism in the backend simulation engine for larger networks will be a challenge as well.

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. Such multi-metric conditions require more advanced skills and thus more training, since the metrics might conflict with each other, e.g., decreasing the delay for a very important customer’s traffic might be more valuable than some decrease in the overall network throughput. Also, as a behavioral fluency criteria we may include speed of response when exceptional cases occurs so that players would also be fluent on handling unpredicted events occured in network such as link failures.

Furthermore, this simulation engine can be improved for other purposes. One example can be what-if analyzer tool where link failures can be simulated. Another example could be related to investment based simulations where network managers have a budget and want to add new link(s) to the current architecture.
Bibliography


