Journal of Network and Computer Applications 53 (2015) 14–23

Contents lists available at ScienceDirect

Journal of Network and Computer Applications

journal homepage: www.elsevier.com/locate/jnca

Training network administrators in a game-like environment[☆]

Engin Arslan[a,1], Murat Yuksel[b], Mehmet Hadi Gunes[b,*]

[a] University at Buffalo, 12 Capen Hall, Buffalo, NY 14260-1660, USA
[b] University of Nevada - Reno, 1664 N. Virginia Street, Reno, NV 89557, USA

ARTICLE INFO

Article history:
Received 25 December 2012
Received in revised form
24 July 2014
Accepted 10 March 2015
Available online 21 March 2015

Keywords:
Network operations
Network management
NS-2

ABSTRACT

Management and automated configuration of large-scale networks is one of the crucial issues for Internet Service Providers (ISPs). Since incorrect configurations may lead to loss of an enormous amount of customer traffic, highly experienced network administrators are typically the ones who are trusted for the management and configuration of a running ISP network. In this paper, we present an online management and experimentation system with a "game" interface to train network administrators and explore what-if scenarios without having to risk the large-scale network operation. The interactive environment treats the trainee network administrators as players of a game and tests them with various failures or dynamics for real-time management and configuration of large-scale networks. 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 also conducted two user experiments using different learning methods and observed sizable improvement in the capabilities of trainees on the problem of interior gateway protocol link weight setting.

1. Introduction

Online management of a running large-scale network poses many challenges that have attracted significant research. Due to the significant costs and risks (Srinagesh, 1997) of building and operating a large-scale ISP network, the challenges in managing the network resources are exacerbated (Kerravala, 2004) because the margin for error or inefficiency is slim (Enck et al., 2009; Oppenheimer et al., 2003). Today, critical applications such as VoIP, IPTV and financial markets migrating 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.

Quick and effective response to system dynamics like failures and demand spikes is a key problem in network operation. Link or node failures are not rare events for a large-scale network of thousands of devices. Even though most network components have pre-installed backups (e.g., backup links implemented using MPLS (Rosen et al., 2001)), multiple failures or major failures/events on critical links or nodes need fast and effective response. A significant portion of the cost associated with such unwanted network dynamics pertains to the detection of the event. Though standards like SNMP (Presuhn et al., 2002) have been in place for many years, swift and low-cost collection of network management data for event detection becomes tedious for larger networks (Habib et al., 2004). Yet a more significant portion of network event handling goes to the time needed to figure out how to respond, which is mostly done manually in current practice. Seeking the optimal response is mostly impractical, but even settling on a "good" response is very hard as well (Ye et al., 2008a).

Highly experienced human administrators are of critical importance as they are typically the only ones who can quickly find the optimum (or close-to-optimum) response to a major failure, e.g., finding an optimum rerouting for a large volume of traffic on a broken pipe. Answers to these questions for an online operational network necessitate meta-tools so that the network administrator can learn and characterize the network.

Good characterization and modeling calls for well-planned design of experiments on the system-at-hand. However, such experiment design on a network quickly becomes prohibitive as the number of parameters increases, also known as "curse of dimensionality"

[☆] This work was supported in part by the NSF awards 0721600, 0721609, and 1321069.

[*] Corresponding author.
E-mail addresses: enginars@buffalo.edu (E. Arslan), yuksem@unr.edu (M. Yuksel), mgunes@unr.edu (M.H. Gunes).

[1] Mr. Arslan was with the University of Nevada, Reno when this work was performed.

http://dx.doi.org/10.1016/j.jnca.2015.03.005
1084-8045/© 2015 Elsevier Ltd. All rights reserved.

(Tenebaum et al., 2000; Törn and Žilinskas, 1989). Thus, there is a need for heuristic optimization algorithms to search for the minimal number of experiments that reveal the most information about the system-at-hand. In order to characterize and understand the system-at-hand well enough and manage it efficiently, there is a heavy need for meta-tools that facilitate experimentation and training capability without harming the real system. The need for such meta-tools that perform efficient experiment design is emphasized when the fact that the system-at-hand is an operational network and the cost of each experiment can be quite high.

Although there have been several tools and outcomes (Ye et al., 2008b) to automate the process of large-scale network management, network operators have found themselves more comfortable with trusting highly experienced administrators. Those well-trained administrators are of critical importance as they are typically the only ones who can quickly find the optimum (or close-to-optimum) response to a major failure, e.g., finding a reroute for a large volume of traffic on a broken pipe (Anderson and Anderson, 2003). However, the complexity of the management and configuration problem is increasing due to the growing heterogeneity in substrate technologies as well as more stringent performance targets demanded by applications (Sun et al., 2012; Benson et al., 2009). Trends in cross-layer design (Srivastava and Motani, 2005) of protocols and integration of various network components are certainly helping the overall performance; however, such methods typically complicate the configuration due to additional parameters they introduce. Thus, tools to train administrators and achieve automated management of a running network are vitally needed.

In this paper, we propose the concept of "network management game" (NMG) that frames the problem of training network administrators in exploring what-if scenarios as a "game". *The NMG aims to establish a game-like environment for trainee administrators to experiment and play with the networks. Such a "game" will enable trainee network administrators to explore what-if scenarios, without having to risk the large-scale network operation – in a way similar to what Flight Simulator provides to trainee pilots.*

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 may not be completely relied upon for handling extreme cases or large scale failures. 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 (Ye et al., 2008b; Gonen et al., 2010; Fortz, 2000; Akyildiz et al., 2014).

In this paper, we apply our network management game (NMG) framework to the problem of IGP link weight setting. We modified the simulator NS-2 and built an animator that interacts with it in real time. The player interfaces with the animator and inputs new IGP link weights as new configurations. The animator, then, conveys these new configurations to the simulation engine. We present screen snapshots of our NMG prototype while it is running and results of user experiments.

2. Related work

Currently, most of the basic training for a network administrator is performed by means of well-defined certification procedures (Cisco Certifications). The administrators receive 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 long 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.

Customized training of network administrators involves what-if analysis (Mate), which is done by in-house tools according to two backbone carriers we have contacted. What-if analysis is a brainstorming activity that uses extensive questioning to guess potential failures and issues in a system, and ensure that appropriate precautions are taken against those problems. For a network, it typically involves a comparison of the network's current performance and "would-be" performance under certain scenarios such as some particular links/nodes fail, and a traffic spike occurs or a new pipe is installed between two points. Network managers regularly use such analysis to quantify robustness or riskiness of the network at hand, in order to answer questions like "Is my network robust enough for potential failures?" or "Where should I put more capacity in my network?". Investment decisions, planning and dimensioning on a network are heavily guided by technical what-if analysis. Businesses often use the scenario manager tool of Excel to explore different scenarios such as the decision making process in e-commerce (Bhargava et al., 1997). Although what-if analysis has high impact within business intelligence platforms, its usage is extended for several purposes such as hazard analysis (Baybutt, 2003), index selection for relational databases (Chaudhuri and Narasayya, 1998), and multi-tier systems (Chen et al, 2009). For instance, in Tariq et al (2008), 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.

Furthermore, various tools have been developed to guide investments and determine how to improve network performance (Mate) with minimal investments. Though existing what-if analysis tools help a network administrator and a 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 (Chatham, 2007; Nicolescu et al., 2007). 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 (Wikipedia), commander training (Serious Games by BreakAway). Financial investment training (The Stock Market Game) is another venue where a game-like environment can be used for training before deploying money on stock market. To our knowledge, using a game for training network administrators was not tried before.

Simulation for training is widely used in many areas including pilot training, computer games, mock markets for businesses (Carmen, 2013), learning nursing (Berragan, 2014), and statistics simulation (Novak, 2014). Carmen (2013) claims that simulations help overcome the major obstacle to the effectiveness of classroom-based training in which the context is different from day-to-day work context. Based on the experience of people on different areas (e.g., pilot training, computer games, and businesses), simulations serve to compress and speed up the learning experience at a fraction of cost and risk. Moreover, simulations serve as a more enjoyable learning activity in comparison to reading a textbook, listening to a lecture, or being part of a team work (Berragan, 2014).

Simulation for training is also helpful to increase effectiveness of teaching computer programming (Iosup and Epema, 2014; Fitz-Walter et al., 2011; Jiang et al., 2011). Jiang et al. (2011) analyzed the effect of computer game-like environment in teaching computer programming to students with no computer science background. Students' performance increased from 40% to 90% in passing computer tests with the help of proposed tool. Iosup and Epema (2014) used “gamification” to increase the success rate of students in higher education. Based on the conducted experiments, gamification helped to increase course completion rate from 65% to 75% as well as increase the student satisfaction.

3. NMG framework

The NMG framework has two parts as in Fig. 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 engine on real-time.

In our prototype, we used NS-2 for the engine and developed a custom 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.

In order to synchronize the GUI and the NS-2 engine, we established a two-way pipe via TCP sockets between them. Briefly, once the engine receives the initial configuration file, it starts the simulation and generates the event traces. Via the pipe, we transfer these traces to GUI, which then displays them to the player. When the player makes a change on a particular link's weight,[2] the engine is informed about it through the pipe again. Once the engine receives the changes, it recalculates the routing based on these changes and carries on to generating more traces.

Algorithm 1. Slow the simulation engine and send traces to GUI.

```
void BaseTrace::namdump()
1: n ← length(trace)
2: if n > 0 then
3:     Client.send(trace,n){ Send the trace to GUI}
4:     sleep(getSimulationSpeed()){ Wait a fixed amount of time}
5: TCLWrite(trace, n) {Write traces to log file}
6: end if
```

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. The player's goal in NMG is to maximize the network's throughput by manipulating link weights. We tried to design an interface where 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 packet-based actions and instantaneous throughput is displayed to inform the player about the effects of changes in link weights.

[2] Though our current prototype only allows link weights to be changed, it is a straightforward extension to allow updating of other simulation configuration parameters.

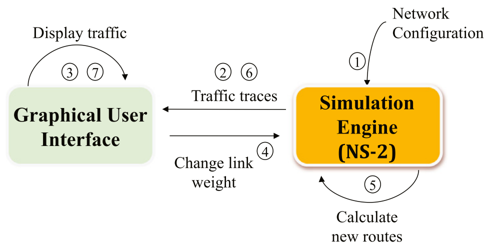

Fig. 1. Block diagram of Network Management Game (NMG) components. The sequence (4)–(7) repeats whenever the player makes changes.

3.2. Engine–animator interaction

A crucial challenge in designing the NGM framework was to synchronize the NS-2 simulation engine and the GUI. The 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. To obtain a more realistic and interactive environment, we reduced the speed of the simulation engine by adding a *sleep* command to its tracing function as detailed in Algorithm 1. The engine waits for a fixed amount of time before processing the next event. We tuned the wait time to so that the engine runs concurrently with the GUI. When the player changes a link weight, the change is sent to the engine and new routing paths are calculated.

We used two TCP sessions to implement the interaction between the GUI and the engine. One of them is opened by the engine to send traces to the GUI, as detailed in Algorithm 2. The GUI listens to this TCP session, reads traces sent by the engine, and animates the changes to the player.

Algorithm 2. Set link weight and simulation speed within the simulation engine scheduler.

```
// Original simulation engine scheduler
void Scheduler::Run()
1: while nextEvent ← EventQueue.dequeue() do
2:     dispatch(nextEvent, nextEvent.time) {process the next
       event}
3: end while
// Revised simulation engine scheduler
messageReceived ← FALSE
message ← emptyString
void listenPort()
1: socket ← ServerSocket(portNumber) {Open a TCP port}
2: socket.connect() {Listen for an incoming connection}
3: while TRUE do
4:     socket.receiveMessage(message)
5:     if message=“Set-Link-Weight” then
6:         STATE messageReceived ← TRUE
7: else if message=“Set-Simulation-Speed” then
8:         setSimulationSpeed(message) {simulation speed is set
       in log file}
9: end if
10: end while
11: createThread(newThread, listenPort) {Create a new thread
       to execute listenPort()}
12: while nextEvent ← EventQueue.dequeue() do
13:     if messageReceived=TRUE then
14:         TCLInstance.eval(message)
15         TCLInstance.computeRoutes()
16:     end if
17:     dispatch(nextEvent, nextEvent.time)
18: end while
```

The second TCP session is opened by the GUI to send link weight changes to the engine. The engine listens to this TCP session with a dedicated thread and processes messages from the GUI before processing the next item in the event queue. As detailed in Algorithm 2, this kind of priority processing of the messages from the GUI enables real-time interactivity with the player. In our current implementation, the GUI sends two types of messages to the engine:

Set-Link-Weight: This message is used for manipulating link weights. When the player changes a link's weight, the GUI prepares this message as a TCL command and sends it to the engine which will directly execute it. “$ns cost $node1 $node2 3” is an example *Set-Link-Weight* message implying to set the cost of the link between *node*1 and *node*2 to 3. Right after executing the *Set-Link-Weight* message, the engine recalculates shortest routes with the new link weights.

Set-Simulation-Speed: This message is utilized only at the beginning of the game to synchronize the GUI and the engine. It controls the engine's speed by setting the fixed wait time between each event (line #4 in Algorithm 1). Since the frequency of calls to BaseTrace::namdump() in Algorithm 1 increases with the network size, the engine slows too much for a larger network if the wait time is not tuned according to the network size. To resolve this synchronization issue, we tuned the wait time to be larger for smaller topologies.

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 scenarios 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 is changed. Figure 2 illustrates a simple test scenario 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 scenario, 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 link loads is known to cause instability in routing, and hence, is avoided in practice (Anderson and Anderson, 2003; Wang and Crowcroft, 1992). We

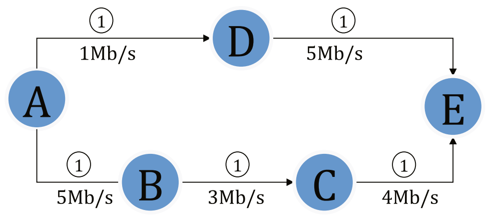

Fig. 2. IGP path selection example. Initial link weights are the same as shown in the circles. Traffic flows from A to E.

picked this particular problem of IGP link weight setting for our game. The purpose of such link weight changes may also be to reduce other metrics like delay or fast failure recovery along with the network utilization. In this work, however, we solely focus on increasing the utilization.

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/scenario 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 Fig. 4). As the game progresses, the player can change the link weights as many time as he/she desires. We picked the link weights in the game since the network administrators dealing with the IGP link weight setting in practice do use the link weights as well. They do not use any other major parameter to solve the problem. Please note that other parameters, such as link bandwidths or failures as detailed in Section 5, are part of our experimental scenarios but they are not user configurable.

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 to changes in the network such as failures or demand spikes. We calculated these throughput values as follows:

$$\tau_{cum}(t) = P(0,t)/t$$

$$\tau_{ins}(t) = P(t-0.1,t)/0.1$$

where t is the current time in seconds from the start of the game and $P(t_1, t_2)$ is the total size of packets successfully transferred to destination(s) within the time period $[t_1, t_2]$. While all transmitted packets are taken into the consideration for τ_{cum}, we considered only the packets transmitted during the last 0.1 s for τ_{ins}.

In the experiments, 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 is no failure in the network, as the player has to guide the TCP flows to accumulate higher throughput. For example, in Fig. 2, there is a TCP traffic from A to E. If the traffic flows through the path A–D–E, then the throughput would approximately be 1 Mb/s. On the other alternative, it almost reaches 3 Mb/s if the traffic follows the A–B–C–E path. We placed many similar alternative paths in our test scenarios to train the players on discovering better ways of extracting more throughput from the network.

4.2. Stages: tutorial, training, and testing

We applied a four-stage process to observe efficacy of NMG in training network management skills: *Tutorial*, *Testing* (Before Training), *Training*, *Testing* (After Training). Figure 3 illustrates these stages for one of the user experiments. We first gave a tutorial to the player about the environment to make sure that the player knows what the game is about and how the GUI works. Then, we tested the player with a relatively complex network scenario (e.g., #6 in Fig. 3) to observe how he/she performs before the training. Next, we trained the player with a sequence of scenarios (e.g., #1–#5 in Fig. 3). Finally, after the training stage is

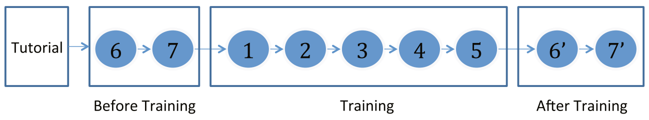

Fig. 3. Training stages: Tutorial, Before Training, Training, After Training.

over, we exposed the player to a test scenario (e.g., #6' in Fig. 3), which is of similar difficulty to the test scenario before the training stage.

By comparing the player's performance before and after the training stage, it is possible to make an observation on the effectiveness of the training. To observe if the comparative results are statistically significant, we applied the T-test for the performance results before and after training.

We used different training and test scenarios. The training scenarios are different from each other in terms of complexity and number of traffic flows. The purpose of the training scenarios (e.g., #1–#5 in Fig. 3) is to train the players with simple networks. The training scenario #3 is shown in Fig. 4 as an example. There are two TCP traffic flows and bandwidths of the links are different from each other. We increased or decreased the width of the links based on 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.

The amount of load on each link is shown by its color, i.e. green for lighter load and red for heavier load. For instance, Fig. 5(a) shows congestion on the links between Seattle and Kansas. Similarly, we visualize the failure of a link on this interface, as seen in Fig. 5(b). The player responds to failure or congestion events by changing the link weights and thus guiding the traffic flows to paths other than those failed or congested.

When designing the testing stages (Before and After Training), we purposefully made the scenarios noticeably more difficult than the scenarios in the Training stage. This allows us to reduce the

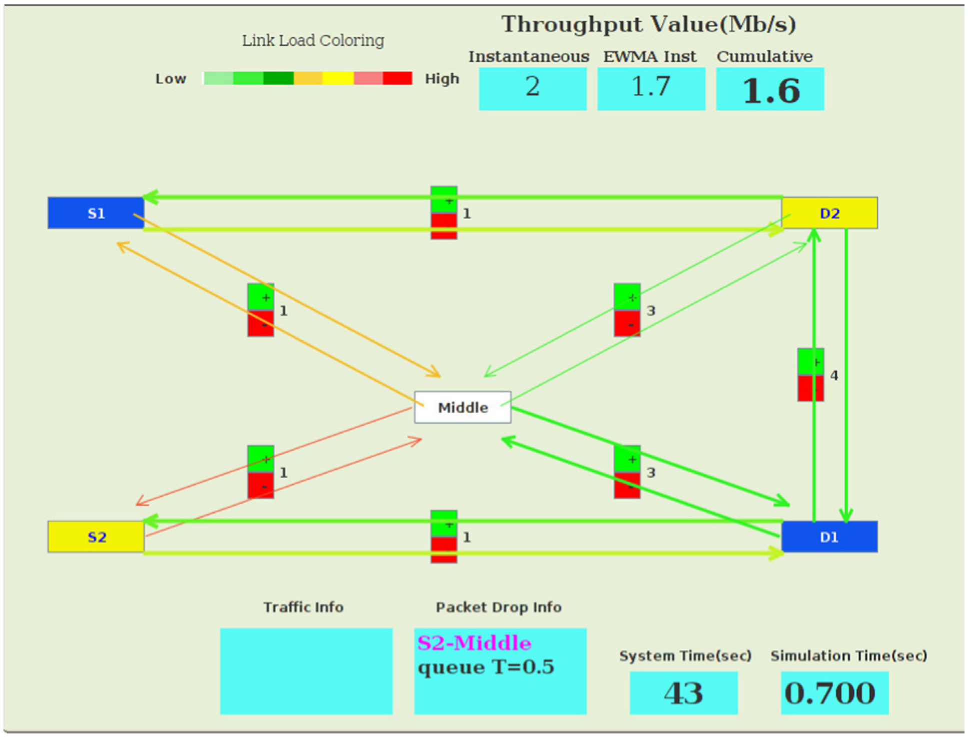

Fig. 4. One of the training scenarios (#3) in which there are two flows. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.)

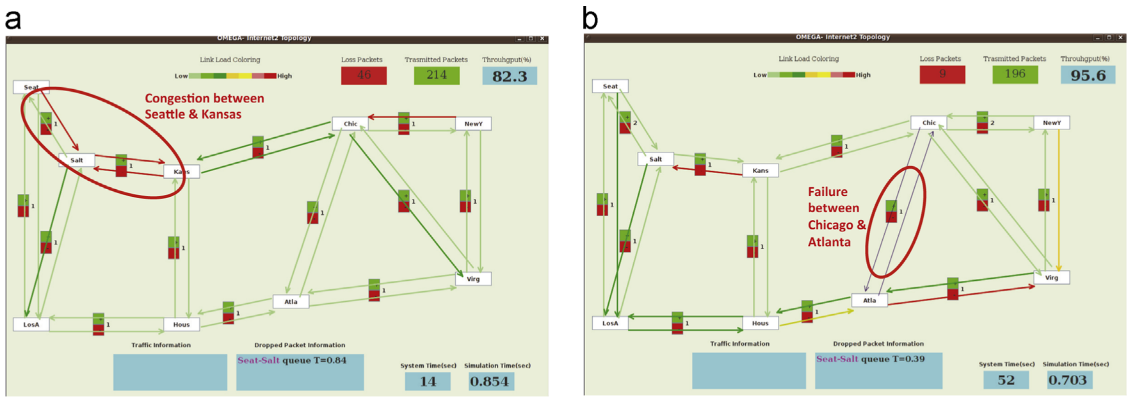

Fig. 5. Sample Internet2 backbone. (a) A congested spot, and (b) A link failure

dependence of training to the specific scenarios and reveal how much the player learned the skill rather than the specific scenarios or networks.

5. User experiments and results

We carried out preliminary user experiments under two different methods to train the players. First of them is "training without mastery" (Sullivan and McIntosh, 1995) in which the players are trained on different network topologies with different difficulty levels. The player played each training scenario/level (e.g., #1–#5 in Fig. 3) for a pre-defined time period and passed to the next level regardless of how well she performed. It is analogous to traditional educational method in terms of being time-based approach. Second training method is "training with mastery" (Sullivan and McIntosh, 1995; Competency-based learning,) where players are required to achieve mastery before advancing to the next level.

We picked volunteer players from our lab fellows in the Computer Networking Lab at University of Nevada, Reno. This ensured that the players are already very knowledgeable about computer networking as they had taken at least one graduate level networking course prior to our experiments. We had 8 volunteers for the first user experiment which took 2 h per player. Among the 8 volunteers who participated in the study, 6 were male, 2 were female; and their age range was 20–30. Instead of finding new volunteers, we proceeded with 5 of those 8 that are willing to continue to the second user experiment, which took 4 h per player. Thus, this selection of players makes the results of the second experiment less susceptible to false-favorable results that may arise due to the player's unfamiliarity to the NMG framework. To assure that our preliminary results are statistically significant, we applied the T-test and will report the p value whenever possible in the subsections below.

5.1. Training without mastery

In this training method, we applied the stages in Fig. 3 where the player is exposed to each training scenario #1–#5 for a fixed amount of time. We used the Abilene topology to design the testing scenarios #6, #7, #6' and #7' but with a different configuration in each. We placed 4 different TCP flows with link capacities varying from 1 Mb/s to 8 Mb/s.

To see how players react to link failures, we applied testing scenarios #7 and #7' before and after training. These failures can be seen in Fig. 6 with black vertical lines. In each game run, we simulated two link failures with Exponential inter-failure and repair times, with averages of 3 s and 2 s respectively. The effect of each failure depends on the current path selections of the player. If the failure occurs on a link which is used by one or more flows, then τ_{ins} decreases drastically, e.g., the first failure in Fig. 6(a). If the failed link is not used by any flows at the time of failure, then τ_{ins} does not noticeably fluctuate, e.g., the first failure in Fig. 6(d). During the link failure period, the player is expected to manage the flows traversing the failed link. The training scenarios #1–#5 do not train the player for failures, and this will allow us to observe if no training on failures have any measurable impact on the player performance.

We experimented with 8 players and monitored their performances. All the players followed training and testing stages as shown in Fig. 3. We kept track of the throughput the players achieved throughout the game and compared the performances of the players with the two extreme cases: "Optimal" and "No Player".

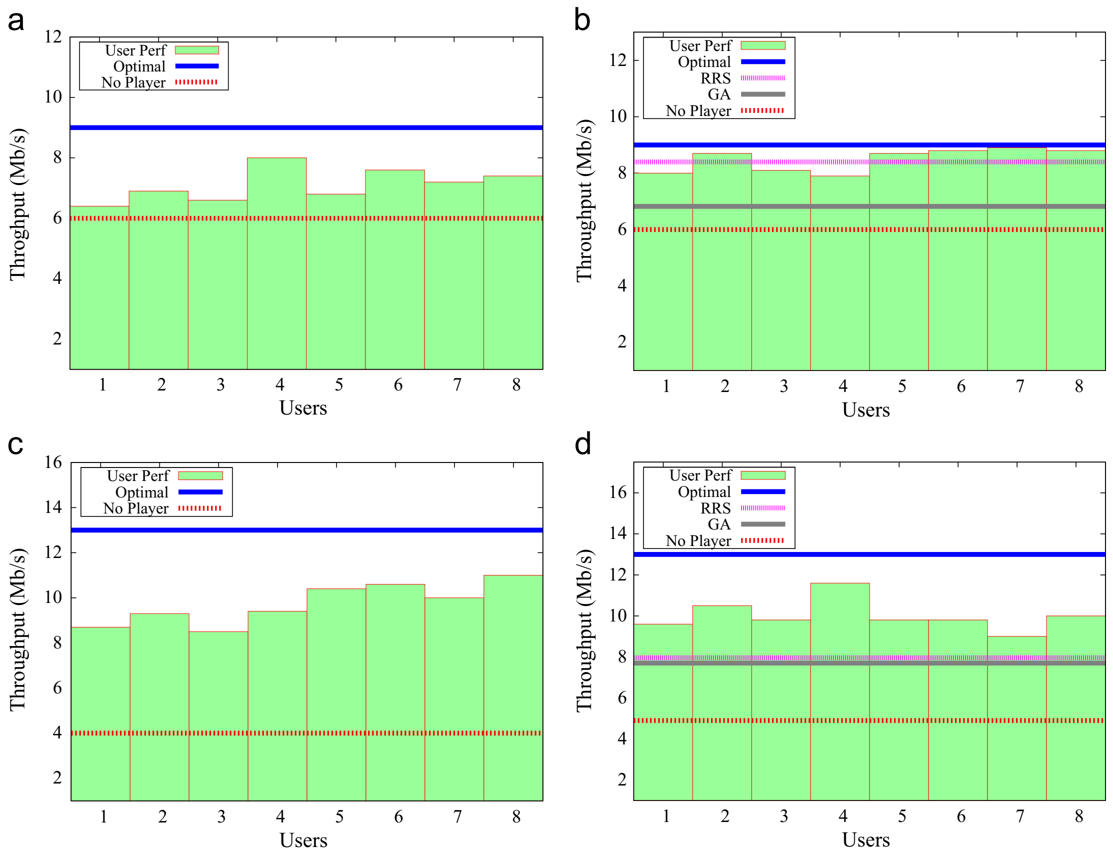

Fig. 6. Player performances compared to the best and the worst possible throughput. (a) Before training (#6) without failures. (b) After trainig (#6′) without failures. (c) Before training (#7) with failures. (d) After trainig (#7′) with failures.

The “Optimal” is the maximum possible cumulative throughput in those particular test cases, while “No Player” refers to the cumulative throughput obtained when the initial configuration of the test case is not changed by the player. To determine the Optimal, we tried out all possible IGP link weights on each of the links and picked the combination giving result to the highest throughput. This exhaustive search for the Optimal worked fine since the networks we experimented with are small. However, finding the Optimal IGP link weights is NP-hard and will require better designed (e.g., linear programming or branch-bound) algorithms for larger networks.

Finding the Optimal is intractable for realistically large networks, and thus heuristics are the reasonable options which can find a good solution quickly. Further, such heuristics better represent a human administrator's evolutionary process of responding to dynamic changes in the network. As the Optimal is typically unreached in a real setting, comparison with heuristics rather than just the Optimal is fairer and helps us to see how well a human player performs against evolutionary optimization algorithms. Thus, we ran two black-box optimization heuristics in test scenarios #6' and #7 to compare the player performance with. We used Genetic Algorithm (GA) and Recursive Random Search (RRS) (Ye and Kalyanaraman, 2004) as the optimization heuristics, which were previously applied to the IGP link weight setting problem (Gonen and Yuksel, 2011). GAs are of the global search heuristic algorithms to find a close-to-optimal solution, while RRS uses random sampling to make use of initial high-efficiency property and restart random sampling to maintain explored high-efficiency.

5.1.1. *Results*

Since the key goal of the game is to maximize the throughput, we focused on the maximum τ_{ins} achieved by the players. Though τ_{cum} is also a good indicator of the players' performance, maximum of instantaneous samples can better characterize the dynamic behavior of the players. Figure 6(a) and (c) show the players' performance in the Abilene topology before the Training stage for the test scenarios #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 scenarios #6 and #7, the optimal throughput was 9 Mb/s and 13 Mb/s while the average player performance was 7.11 Mb/s and 9.73 Mb/s.

Figure 6(b) and (d) shows performance of the players after the Training stage for the scenarios #6' and #7', respectively. Compared to the performance before the training, we can see a significant improvement in the players' performance, particularly in the case without failures, i.e., scenario #6. The average player performance increased to 9.73 Mb/s in the 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 scenario 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 compared to the before training (i.e., #7). This is sizably smaller than the improvement observed for the case without failures (i.e., #6 and #6') since we did not train the players on link failures during training phase (i.e. the scenarios #1–#5 in Fig. 4). The result that players' improvement is small for the failure cases (i.e., #7 and #7') indirectly shows that the players performed worse if they are not trained for failures.

Besides intuitive inferences, we also applied T-test for this experiment to show statistical significance of NMG training on the improvements observed in the performance of players. While applying the statistical test, we used the maximum τ_{ins} achieved by the player throughout a scenario. Thus, T-test compares maximum τ_{ins} values of the 8 players for the test scenarios before and after training. Since we exposed scenarios with and without failure separately (#6–#6' and #7–#7'), we calculated two different p values. p Value for scenarios #6 and #6' is 0.00024. This value is much smaller than threshold 0.05 to be statistically important thus we can conclude that training positively affected the player performance. p value for test scenarios #7 and #7' is 0.27881 which is higher than threshold. As we did not train players on link failure cases during Training stage, performances did not improve significantly in this case.

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 outcome could be attributed to the fact that the human players can observe a larger or even a global view on the system-under-test while the algorithms may only be limited to a local view during their progress.

5.1.2. *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 τ_{ins} and τ_{cum} achieved by these players as the game progresses. To comparatively observe how the players' changes affect the achieved throughput, we also plot τ_{ins} when no updates to the link weights are done, i.e., the “No Player” case. The No Player and τ_{ins} lines fluctuate frequently due to the adaptive behavior of the TCP flows.

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, some links are not used for traffic; and, when those links fail, the throughput is not affected, e.g., the second failure in Fig. 7(a).

Figure 7(c) and (d) plots the best performing player's behavior over time during the test scenarios #7 and #7', respectively. After the training, the player seems to respond to the failures faster and tune the link weights earlier. Further, the player seems to learn how to tune the link weights to discover more bandwidth in the network, even before failures happen (see the jumps in τ_{ins} at around time 2 s and 5.5 s).

Though these results provide a glimpse on potential training of players' reflexes to failures, the players' improvement on the cases with failures was not statistically significant as we indicated in Section 5.1.1. More and longer experiments are needed to verify that the players' response time to failures reduce after training. Further, measuring and characterizing the convergence behavior of players' learning would be an interesting observation. Though our results provide some overall visualization of players' convergence behavior (e.g., Fig. 6(a)), a more detailed and lengthy analysis needs to be made using the timing and occurrences of when players make changes to the link weights, which we have not recorded during our experiments.

5.2. *Training with mastery*

In this training method, we again applied a staged process as shown in Fig. 8. After the tutorial, the player started with a relatively complex network (i.e., test scenario #8) to observe how she performed before the training. Next, we trained the player using scenarios #1–#7 with each case. However, in order to advance from one scenario to another, a player is required to achieve certain throughput increase within a limited amount of time, i.e., Restart_Time. Table 1 shows details of the training scenarios and the number times the players had to retry in order

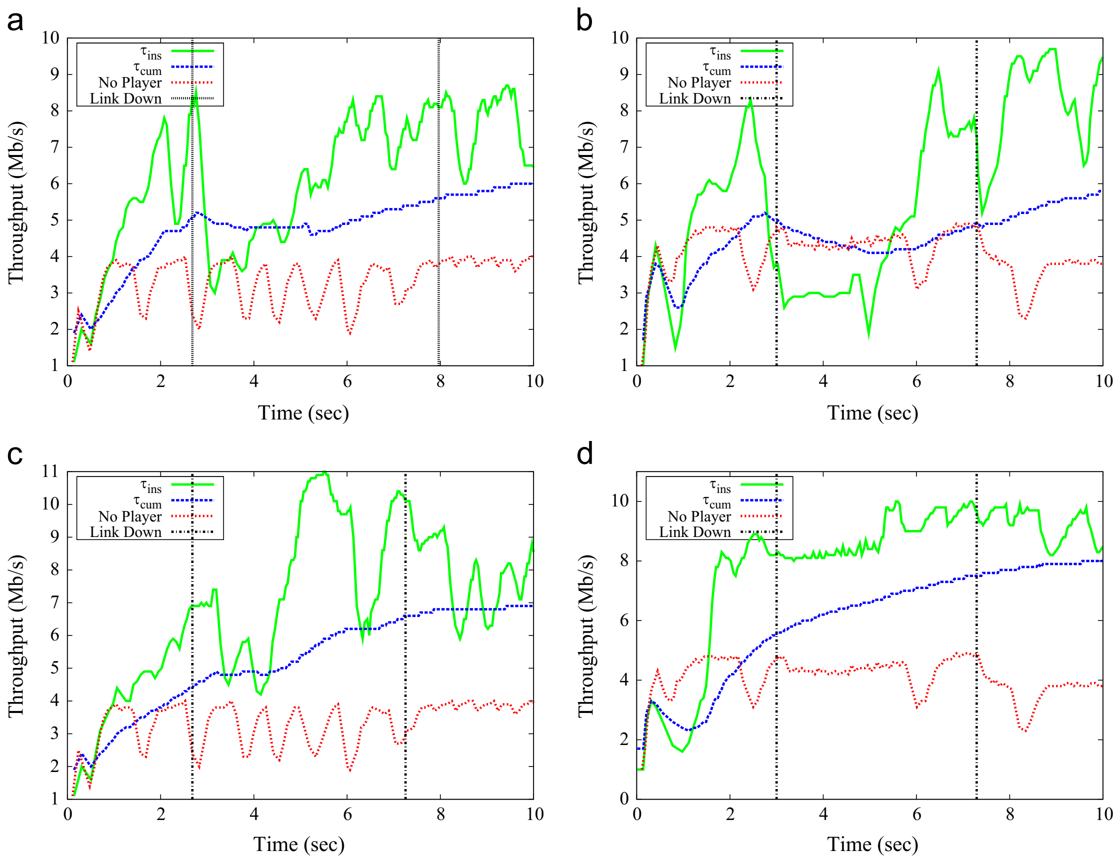

Fig. 7. Performance of the worst and the best players on the scenarios with failures. (a) Worst player (case #7). (b) Worst player (case #7′). (c) Best player (case #7). (d) Best player (case #7′).

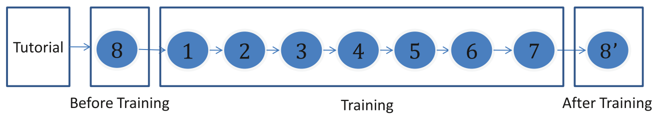

Fig. 8. Training stages: Tutorial, Before Training, Training, After Training.

Table 1
Details of training scenarios of second user experiment.

Scenario	# of Flows	Restart_Time (min)	Average retries
1	1	2	1
2	1	2	1.1
3	2	2	2.1
4	3	2	2.3
5	3	2	2
6	4	2	3.1
7	4	3	4.5

to pass a scenario. The difficulty level of a scenario is mostly dependent on the number of flows among other factors. Other factors that contribute to the difficulty level of a scenario include the number of nodes, the number of links, the variance of propagation delay across the links, and the variance of flow rates. These are mostly topological parameters, which we kept the same across different scenarios in our testing. We also kept Restart_Time relatively constant as the difficulty level of the training scenarios increases. Table 1 shows a strict increase on the average number of retries per player as the difficulty level of the scenarios increases.

After the training stage is over, we exposed scenario #8', which is similar to the scenario #8 in terms of difficulty.

We selected five players among those who attended the first user experiment. Thus, they were very familiar with the NMG framework and the environment. This selection of players allows us to rule out (reduce) favorable results we may observed in the first user experiment due to the players' unfamiliarity to the framework. Further, we targeted the skills that may be helpful in solving the IGP link weight setting problem. Although there are several skills used by network administrators, we focused on two of them:

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. Since we used TCP traffics in our experiments, overall throughput depends on available bandwidth on the path. As in Fig. 2, we expect players to make appropriate link weight changes to flow traffics along the path which maximizes throughput.

Decoupling of flows: Second skill that we targeted is decoupling of flows that are passing over at least one common link. In Fig. 9, there are two TCP flows from node A and node B directed to node D. If both traffic flows pass over link C–D, capacity of this link will be shared which then limits overall throughput to 4 Mbps. 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, network administrators usually face cases where experience of both the skills are required to increase resource utilization. Hence, we first trained players on each skill separately. Then, we trained

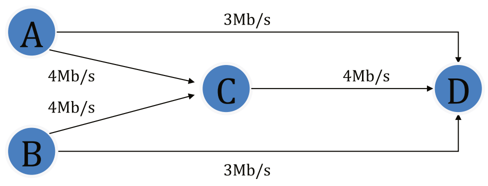

Fig. 9. There are 2 traffic flows from A to D and from B to D.

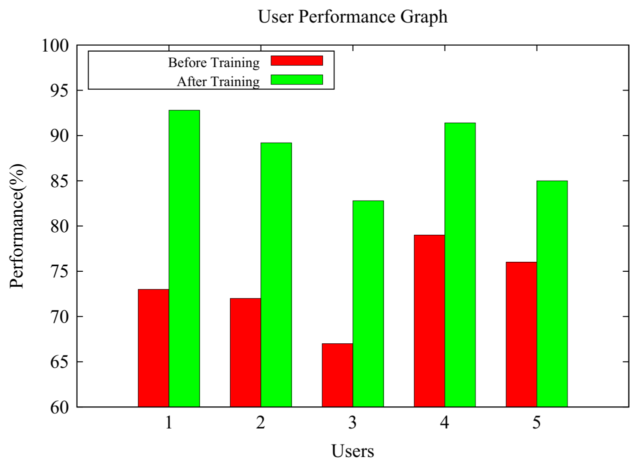

Fig. 10. Comparison of players' performance before and after training.

the cases where experience both skills that are needed to find optimal IGP link weight setting.

5.2.1. *Results*

Players' performance on Abilene network topology (scenario #8 and #8') at the beginning and at the end of the training is given in Fig. 10 as a percentage of optimal solution. Players could go up to 70% of optimal solution before training. However, no one was 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 player. Every player was able to find solution that goes more than 80% of optimal one. Improvement of player performances varies from 13% to 21% in terms of finding close-to-optimal solution. We again applied T-test for the player performances on the test scenarios #8 and #8' based on maximum achieved instant throughput and, p value is computed as 0.00001 which is way below the threshold of 0.05. Thus, we can conclude that "training with mastery" method is effective to train players on IGP link weight setting.

These results are promising since we only focused on two frequently used skills for IGP link weight setting and tested on simple topologies, and traffics. We believe that further improvement in framework and player training methodology would lead to better player performance improvement. The idea that we want to emphasize is trainability of people on IGP link weight setting using interactive simulation engine.

6. Summary and future work

In this work, we introduced a game-like environment, the NMG framework, to train people in terms of network management skills. Our goal was to develop an initial proof-of-concept prototype demonstrating that the NMG framework can train players (even the ones very knowledgable about networking), and improve their skills on an NP-hard network management problem like the IGP link weight setting.

To see efficacy of the framework, we carried out two user experiments with two different training methods. In the first one, five training and two testing scenarios are exposed to 8 players. We observed 21% average performance improvement in players' management skill for the scenario without failures. To see how the players' would perform if they are not trained for a particular network behavior, we exposed the players to scenarios with failures even though they were not trained for failures. The players' performance only increased 2.9% in scenarios with failures, indirectly verifying our intuition that the players improve their skill if they are trained by NMG.

For further improvements, we carried out second user experiment in which we focused on 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. Although the players are selected among those who attended first user experiment, we again observed 13% to 21% skill improvement which supports the success of NMG, particularly when the players are forced to master each stage of the game.

Our work is a first step in exploring the concept of using games to train network administrators. An immediate step into the future work is to extend the quantity and variety of training and testing scenarios in our experiments. Larger network scenarios will require longer time (days or months) to train the players, and incentivizing participants for such long experimentation is beyond our current capabilities that are based on voluntary participants from our lab.

Understanding if NMG can help improve reflexes of the players (i.e., the response time of players to dynamic events like failures) is a critical future work. For example, it will be interesting to see how well players can get trained against demand spikes occurring due to temporary changes to the traffic matrix of the network. Such demand spikes are one of the typical situations where highly experienced administrators (i.e., the target community of NMG) are needed. Further, using metrics other than throughput, such as delay or congestion, for evaluating the performance of players is also a worthy future work. In this paper, we thought that maximizing a value would give a better game-like feeling to the player than minimizing a value. Since both delay and congestion would require the player to minimize a value, we picked the throughput as the metric for the game. However, the other metrics may be of more relevance for certain situations. For example, 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.

7. Open issues and challenges

Several open questions arise when one is to extend NMG to more realistic settings and larger networks. Such extensions involve non-trivial steps such as visualization of larger topologies and more traffic flows. Alternative approaches include (a) *large screens* to visualize larger networks (e.g., NYC subway (Visualization of NYC Subway Rides; Musical Visualization)), (b) *distributed visualization* of a network divided into multiple zones each being handled by a different player on a separate screen with backend communication among zones (e.g., air traffic control (Azuma et al., 2000), power grid (Wu et al., 2005; Overbye and Weber, 2001)), and (c) *multi-level visualization* with zooming capabilities (Shelley and Gunes, 2012).

Another challenge is the visualization and handling of many traffic flows by a single player. Clearly, aggregation of multiple flows into one will be necessary so that the player can plausibly relate to the visualized traffic. The visualization granularity can range from link-level utilizations (i.e. complete ignorance of flows) to separate colored line for each flow. Selection of the right visualization approach will depend on the size of the network at hand and the particular management problem being tackled. For instance, the first approach (a) will probably suffice to address the visualization of the IGP link weight setting on the inter-PoP topology of a large ISP backbone. Regardless, the specifics of the user interface and the visualization granularity will have to be tuned for the particular cases.

The NMG framework promises a lot in terms of reducing the network administrator training costs and having them experiment with scenarios that may arise infrequently in practice. However, it does require setting up of a simulation-based environment that emulates the real control room and knobs of a network administrator in practice. It may not always possible to emulate every detail of a real setting. This study showed that sizable benefits can be reaped by training even with a simple computer game environment. Further research is needed to figure out how much additional benefit is possible if the NMG environment is made more realistic.

Further, a key parameter into the success of NMG is the proper identification of network management problems and skills that the administrators need to work on. If the management problems and the required skills to solve those problems are clearly identified, then it becomes easy to construct the scenarios in the NMG framework. For instance, for the IGP link weight setting problem, we identified two skills (high bandwidth path selection and decoupling of flows). It requires more effort to further identify other skills needed for the IGP link weight setting problem. More interaction with practitioners and highly experienced network administrators is needed to pin down the appropriate and full set of skills for the problem at hand.

Finally, we would like to emphasize that our experiments were not designed to be “fun”, which is a typical expectation from a traditional game. It is certainly possible to extend the NMG concept to design dimensions with more fun. For instance, one may design a multi-player strategy game involving multiple network managers each trying to grow their networks while competing with other neighbor networks – similar to the competition among ISPs on the Internet. Further, simplified versions of NMG can be considered for mobile handhelds where the reflex of the player to various network dynamics is of interest.

Acknowledgements

This work was supported in part by the NSF awards 1321069 and 1321164.

References

Akyildiz IF, Lee A, Wang P, Luo M, Chou W. A roadmap for traffic engineering in sdn-openflow networks. Comput Netw 2014;71(0):1–30.

Anderson EJ, Anderson TE. On the stability of adaptive routing in the presence of congestion control. In: Proceedings of IEEE INFOCOM; 2003. p. 948–58.

Azuma R, Neely I, Daily HM, Geiss R. Visualization tools for free flight air-traffic management. IEEE Comput Graph Appl 2000;20(5):32–6.

Baybutt P. Major hazards analysis an improved process hazard analysis method. Process Saf Prog 2003;22(1):21–6.

Benson T, Akella A, Maltz D. Unraveling the complexity of network management. In: Proceedings of the 6th USENIX symposium on networked systems design and implementation, NSDI'09. Berkeley, CA, USA: USENIX Association; 2009. p. 335–48.

Berragan L. Learning nursing through simulation: a case study approach towards an expansive model of learning. Nurse Educ Today 2014;34(8):1143–8.

Bhargava HK, Krishnan R, Müller R. Electronic commerce in decision technologies: a business cycle analysis. Int J Electron Commer 1997;1(4):109–27.

Carmen M. The power of simulation-based learning. Train Dev 2013;40(4):18–20.

Chatham RE. Games for training. Commun ACM 2007;50:36–43.

Chaudhuri S, Narasayya V. Autoadmin what-if index analysis utility. In: Proceedings of ACM SIGMOD; 1998. p. 367–78.

Chen S, Joshi KR, Hiltunen MA, Sanders WH, Schlichting RD. Link gradients: predicting the impact of network latency on multitier applications. In: Proceedings of IEEE INFOCOM; 2009. p. 2258–66.

Cisco Certifications, ⟨http://www.cisco.com/web/learning/index.html⟩.

Competency-based learning, ⟨http://en.wikipedia.org/wiki/Competency-based_learning⟩.

Enck W, Moyer T, McDaniel P, Sen S, Sebos P, Spoerel S, et al. Configuration management at massive scale: system design and experience. IEEE J Sel Areas Commun 2009;27(3):323–35.

Fitz-Walter Z, Tjondronegoro D, Wyeth P. Orientation passport: using gamification to engage university students. In: Proceedings of the 23rd Australian computer–human interaction conference, OzCHI '11, ACM, New York, NY, USA; 2011. p. 122–5.

Fortz B. Internet traffic engineering by optimizing OSPF weights. In: Proceedings of IEEE INFOCOM; 2000. p. 519–28.

Gonen B, Yuksel M. Network configuration and management via two-phase online optimization. In: Proceedings of IEEE GLOBECOM; 2011.

Gonen B, Yuksel M, Louis S. Probabilistic trans-algorithmic search for automated network management and configuration. In: Proceedings of IEEE GLOBECOM MENS Workshop, Miami, FL; 2010.

Habib A, Khan M, Bhargava B. Edge-to-edge measurement-based distributed network monitoring. Comput Netw 2004;44(2):211–33.

Internet2, University Corporation for Advanced Internet Development, ⟨http://www.internet2.edu/⟩.

Iosup A, Epema D. An experience report on using gamification in technical higher education. In: Proceedings of the 45th ACM technical symposium on computer science education, SIGCSE '14, ACM, New York, NY, USA; 2014. p. 27–32.

Jiang Z, Fernandez E, Cheng L. A pedagogical pattern for teaching computer programming to non-cs majors. In: Proceedings of 18th conference on pattern language of program (PLoP), in conjunction with ACM FLASH 2011; 2011. p. 21–23.

Kerravala Z. As the value of enterprise networks escalates, so does the need for configuration management, Enterprise Computing and Networking, The Yankee Group; January 2004.

Mate, ⟨http://www.cariden.com/products⟩.

Musical Visualization of NYC Daily Schedule, ⟨http://mta.me⟩.

Nicolescu MN, Olenderski A, Leigh R, Louis SJ, Dascalu S, Miles C, et al. A training simulation system with realistic autonomous ship control. Comput Intell 2007;23(4):497–519.

Novak E. Effects of simulation-based learning on students' statistical factual, conceptual and application knowledge. J Comput Assist Learn 2014;30(2):148–58.

NS-2, ⟨http://www.isi.edu/nsnam/ns⟩.

Oppenheimer D, Ganapathi A, Patterson DA. Why do internet services fail, and what can be done about it? In: Proceedings of USENIX USITS; 2003.

Overbye T, Weber J. Visualizing the electric grid. IEEE Spectr 2001;38(2):52–8.

Presuhn R, Case J, McCloghrie K, Rose M, Waldbusser S. Version 2 of the protocol operations for the simple network management protocol (snmp), RFC 3416; December 2002.

Rosen E, Viswananthan A, Callon R. Multiprotocol label switching architecture, IETF RFC 3031; 2001.

Serious Games by BreakAway, ⟨http://www.breakawaygames.com⟩.

Shelley D, Gunes MH. Inner sphere network visualization. Comput Netw 2012;56(3):1016–28.

Srinagesh P. Internet economics. Cambridge, MA: MIT Press; 1997. p. 121–54 [Chapter: Internet cost structures and interconnection agreements].

Srivastava V, Motani M. Cross-layer design: a survey and the road ahead. IEEE Commun Mag 2005:112–9.

Sullivan R, McIntosh N. The competency-based approach to training. In: Proceedings of Learning without walls: a pre-congress seminar; 1995. p. 13–15.

Sun X, Rao SG, Xie GG. Modeling complexity of enterprise routing design. In: Proceedings of the 8th international conference on emerging networking experiments and technologies, CoNEXT '12, ACM, New York, NY, USA; 2012. p. 85–96.

Tariq M, Zeitoun A, Valancius V, Feamster N, Ammar M. Answering what-if deployment and configuration questions with wise. In: Proceedings of the ACM SIGCOMM; 2008. p. 99–110.

Tenebaum J, de Silva V, Langford J. A global geometric framework for nonlinear dimensionality reduction. Science 2000;290(5500):2319–23.

The Stock Market Game, ⟨http://www.stockmarketgame.org⟩.

Törn A, Žilinskas A. Global optimization. of Lecture notes in computer science, vol. 350. Berlin, New York: Springer-Verlag; 1989.

Visualization of NYC Subway Rides, ⟨http://data.fabernovel.com/nyc-subway/⟩.

Wang Z, Crowcroft J. Analysis of shortest-path routing algorithms in a dynamic network environment. ACM CCR 1992;22:63–71.

Wikipedia: Flight Simulator, ⟨http://en.wikipedia.org/wiki/Flight_simulator⟩.

Wu F, Moslehi K, Bose A. Power system control centers: past, present, and future, Proc IEEE 2005;93(11):1890–908.

Ye T, Kalyanaraman S. A recursive random search algorithm for network parameter optimization. In: Proceedings of ACM SIGMETRICS, vol. 32; 2004. p. 44–53.

Ye T, Kaur HT, Kalyanaraman S, Yuksel M. Large-scale network parameter configuration using an on-line simulation framework. IEEE/ACM Trans Netw 2008a;16(4):777–90.

Ye T, Kaur HT, Kalyanaraman S, Yuksel M. Large-scale network parameter configuration using an on-line simulation framework. IEEE/ACM Trans Netw 2008b;16:777–90.