Evaluation of the Placement of Network Services

Todd Sproull and Roger D. Chamberlain
Department of Computer Science and Engineering, Washington University in St. Louis, St. Louis, Missouri, USA

Abstract—*Network services are used in the Internet today for a variety of functionality, from Voice Over IP (VoIP) relays to network game servers. Determining the placement of these network services and measuring the quality of the placement in realistic scenarios is a challenging problem. This paper explores different methodologies for evaluating the placement of network services. The strategies include: simulations of with thousands of nodes, emulation of different topologies with hundreds of nodes, and Internet deployment for a diverse selection of nodes and network communications. In addition, comparisons to a centralized communication model are studied. This range of exploration enables a better understanding of the impact the selection of supernodes provides on a variety of applications and platforms.*

1. Introduction

Network researchers are faced with a range of challenges when developing a new service or application. One of these challenges is determining the proper set of evaluation techniques for the new service. In order to fully evaluate any new service, the process requires evaluation from different perspectives and implementations. This process varies from exploring the essential algorithms at the heart of the service to deploying a full implementation running on the Internet. Evaluating a new service across this field of implementation options provides a deeper understanding along with validation of ideals and assumptions at different levels.

We break the assessment process down into three categories. First we have simulation. Here we consider discrete event simulation models where the functional and performance properties of a candidate Internet service are being investigated. Second we have emulation. In emulation physical (or virtual) nodes along with a physical (or virtual) network exist in some constrained topology (such as Emulab [1]) to provide a virtual sandbox for our service. Lastly we consider experimentation as physical (or virtual) nodes and networks on an Internet topology (such as PlanetLab [2]).

Typical computer science research first investigates a simulation model, then builds a prototype to evaluate in emulation and finally a full deployment through experimentation. We are turning this model around. Here, we present an application that can utilize our proposed service and demonstrate improvement. Next, we evaluate the service itself through experimentation with Planetlab. We then evaluate the service on a constrained topology through emulation on Emulab. Finally a variety of network topologies that we could not deploy on are created and evaluated through simulation.

The network service we are evaluating is the Supernode Placement in Overlay Topologies (SPOT) [3]. This research leverages our initial work that proposed several placement algorithms. In this paper we explore the service in more detail and investigate it's behavior using different techniques. First, we will briefly discuss SPOT's behavior. Next we demonstrate the feasibility of SPOT as a service to determine Internet game server placement. Then we investigate SPOT with emulation and finally evaluate a simulator developed for SPOT.

2. Background

Here we provide detail about the general behavior of SPOT along with an example. A more complete explanation can be found in [3]. SPOT selects a subset of network nodes to be supernodes (SNs). A supernode is one that provides additional services to the network. Consider a SN as the game server node in a first person shooter game played on the Internet or a relay node to assist in Voice over IP (VoIP) communication. We now define the problem formally

2.1 Problem Statement

Consider a graph G = (V,E) where V are the nodes and E represent links between the nodes. We define some subset of V as V_A, which are active sending message nodes. Of these V_A nodes we define a subset V_W that represent those nodes that are willing to become SNs. We then define V_k as the active nodes that are currently assigned as SNs, where k is the number of SNs we are interested in assigning. Therefore, $V_k \subseteq V_W \subseteq V_A \subseteq V$.

We also define a demand t for some node u as $t(u)$. The shortest distance between nodes u and v is $d(u, v)$. We are interested in finding a set of k nodes to be assigned SNs. Therefore (from [3]) we want to determine V_k such that

$$\forall S \in 2^{V_W}, |S| = k \rightarrow \left(\sum_{u \in V_A} t(u)d(u, V_k) \leq \sum_{u \in V_A} t(u)d(u, S) \right) \quad (1)$$

We also define a $TotalCost$ [3] for any S where $|S| = k$ as the following:

$$TotalCost = \sum_{u \in V_A} t(u)d(u, S) \quad (2)$$

This gives us a measurement of how well one selection of SNs compares to another.

2.2 General Behavior

As the size of the network increases, calculating the distances and demands to and from all nodes becomes expensive. In order to reduce the amount of communication required, nodes are divided into groups called *neighborhoods*. The size of the neighborhood is based on a predetermined distance metric r from the current SN. Inside each of these neighborhoods complete distance and demand information is determined. As the SN assignment changes inside a neighborhood, the center of the neighborhood refocusses on the new SN. Therefore the members inside the neighborhood may change after each assignment.

Nodes initially joining the network connect to a well known bootstrap node for authentication and initialization. Once authenticated, a node is either promoted to SN status or provided the address of an SN to connect to. Each is then notified by that SN if the node is close enough to join the neighborhood. Those nodes not joining the neighborhood locate the closest node in the neighborhood to act as a *neighborhood representative*. A neighborhood representative provides mechanisms for nodes outside the neighborhood to influence the future assignments of SNs. This is accomplished by the neighborhood representatives aggregating demand from nodes outside the neighborhood and representing it as their own. Where demand represents the desire for a node to use the service and we are assuming a demand of unity for each node. An example is now scenario is now provided.

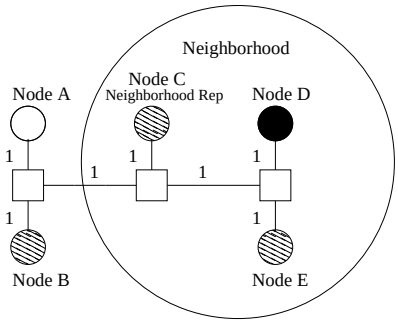

Fig. 1: Example neighborhood with SN at node D and neighborhood representative at node C with an r value of 3.

Figure 1 depicts a network with three nodes inside and two nodes outside of a neighborhood. The nodes are connected with routers (denoted as squares) and hop distances (denoted as wires between nodes and routers). A distance of two exists between Nodes A and B in Figure 1. Nodes willing to become a SN are denoted with partially filled in circles (nodes B, C, and E). The solid circle represents the current SN (node D). Nodes that are active but choose not to become the SN are represented with a plain circle (node A). In this figure, the radius metric r=3, therefore nodes nodes C and E are inside the neighborhood (with distances of 3 and 2 to the SN, node D, respectively). Nodes A and B are too far from the SN therefore placed outside the neighborhood, node A is also not willing to become an SN (it is in the active state, but not willing to become an SN) and would not join the neighborhood even if it was close enough. This is because any node not in the willing to become an SN state can not join the neighborhood because all nodes in the neighborhood must be eligible to become an SN. Nodes A and B must find their closest neighborhood representative and will select node C. With this topology, nodes A and B use node C as their neighborhood representative, and node C reports a demand of 3 to the SN, with node E reporting a demand of 1. With the node demands and topology information for nodes C, D, and E, the SN, node D, is ready to determine if it will reassign the SN to a new location. To do this, the SN solves the local k-median problem for three nodes with the specified demand and topology information using integer linear programming (ILP). In this example, the output of the k-median problem assigns node C as the SN and the neighborhood in Figure 2 is created.

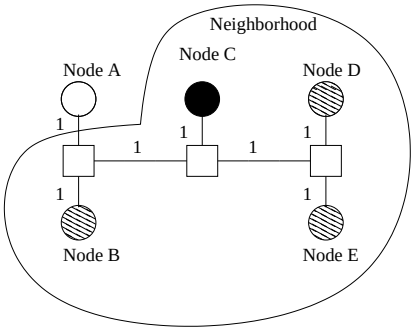

Fig. 2: Example neighborhood with SN at node C.

This SN placement strategy strikes a balance between complete global knowledge and a very limited local view. With local knowledge of the distance to each node and associated demand, the SN is able to solve the k-median or problem while not completely ignoring nodes outside of the neighborhood by considering aggregate demand information.

2.3 Software Implementation

SPOT is written in 12000 lines of multithreaded Java code. Each node initializes by listening on a well-known TCP socket and forks a thread for each incoming command. In

order to solve the local k-medians problem the GNU Linear Programming Kit (GLPK) [4] is utilized.

3. Experimentation

3.1 Introduction

The first type of evaluation is via *experimentation*. Here, we are deploying the SPOT system on the Planetlab testbed. This testbed consists of over 1000 nodes distributed around the world. Each researcher is allowed access to a *slice* of every single node in the network. This is very useful with regards to the diversity of systems and networking environments. This type of environment increases the realism and quality of experimentation greatly.

The test setup involved deploying SPOT on 50 nodes in the Planetlab environment. The size of the experiment may appear small, however due to the unpredictable nature of Planetlab, hundreds of nodes are unavailable at any given time.This number is also common with other researchers working with distributed systems [5].

3.2 Planetlab Evaluation

The initial experiments on Planetlab involved collecting statistics about the nodes. In Figure 3 the cumulative distribution function is provided for the number of hops necessary to reach all of the nodes.

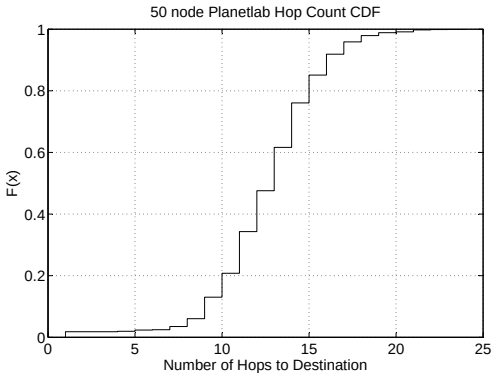

Fig. 3: CDF of the number of hops necessary for nodes to reach each other in the 50 node Planetlab experiments.

Next we deployed the SPOT system on all 50 nodes and selected an arbitrary node (not one of the 50) to operate as the bootstrap server. Once the software was deployed, 50 experiments were run with k values of 1, 2, and 3 SNs, where k represents the number of SNs deployed. The results of the experiments are shown in Figure 4. These results use the hop count metric. From the results, the total cost decreases as the number of SNs increase. This is to be expected as adding more SNs should decrease the total network cost. These results are reported using the whisker-box plot that presents the lower (0.25), median, and upper quartiles (0.75) of the data. Also included are the minimum and maximum values. The interquartile range (IQR) is defined as the upper quartile minus the lower quartile. Also, any value that is 1.5 times the upper quartile or 1.5 times the lower quartile is consider an outlier and denoted with a circle.

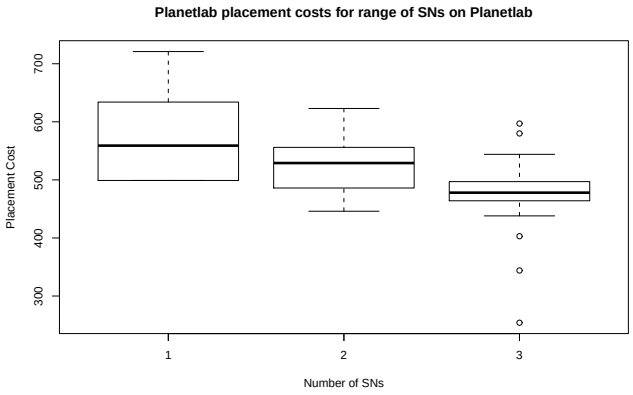

Fig. 4: 50 node Planetlab r-SPOT experiment using the number of hops distance metric illustrates the placement costs for k=1,2, and 3 SNs.

In these experiments we measured the total time to locate SNs, the number of iterations of the algorithm to assign an SN and the total amount of network traffic generated from all of the nodes in the experiment. Due to space constraints only the time to assign an SN is illustrated, Figure 5. The remaining data can be seen in [6]. From the results, as the number of SNs increase, so too does the total time necessary to assign the SNs.

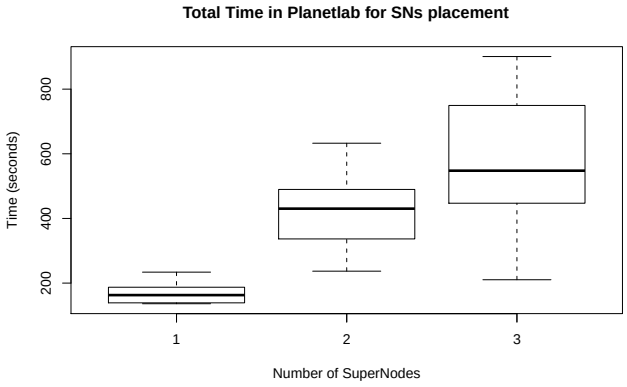

Fig. 5: 50 node Planetlab r-SPOT experiment illustrating the total time to place k=1, 2, and 3 SNs.

4. Game Servers

4.1 Introduction

The previous results have dealt with SPOT, its ability to place SNs throughout a network and measurements as-

sociated with it. Ultimately, the improvements that SPOT provides for applications utilizing its service are important. In order to evaluate that, we turn our focus to online video games, namely multi-player first-person shooters. Multi-player first person shooters (such as Quake III Arena and Half-Life [7]) are very sensitive to the latency from the client to the game server. Typically anywhere from 16 - 64 people connect to a single server or host. This host sends game updates to all players connected to the server. If the client has a RTT to the server greater than 180 - 200 ms, it can greatly reduce the quality of the experience as well as fairness in the game itself [8]. Therefore when creating a multiplayer game, it is important to choose the game server carefully.

4.2 Planetlab Evaluation

In order to evaluate the effects of server selection, the 50 node setup in Planetlab was studied. Using these 50 nodes, the ping data collected earlier was used to evaluate the RTTs letting each node become the *server* in a online game. Therefore, we are interested in the RTT from each client to that server. From this data 7 of the 50 servers or 14% of the nodes would be unable to satisfy the requirement that every node maintain a RTT under 180 ms. This demonstrates the importance of carefully selecting a SN. The RTTs are shown in Figure 6.

Next SPOT was run across all 50 nodes with k=1 and it selected node 8 as the SN, the average RTT is 60 ms to the SN and the maximum RTT is 139 ms as shown in Figure 7, from [3]. The optimal value is selecting node 6 as the SN with an average RTT of 40 ms and a maximum RTT of 118 ms. The worst case selection is node 42, with an average RTT of 127 ms, a worst case RTT of 1545 ms and three nodes over the 180 ms threshold.

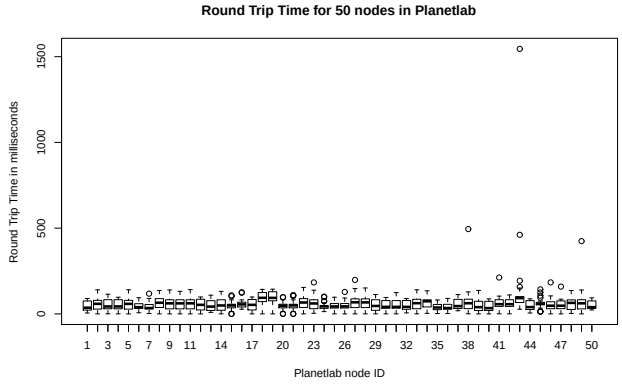

Fig. 6: Round Trip Times from each node to all 50 nodes in Planetlab.

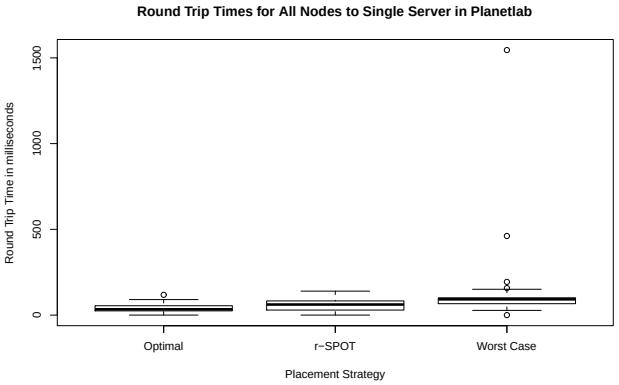

Fig. 7: Round Trip Times from each node to a single server selected based on the particular placement algorithm times.

4.3 Updating the SN as the topology changes

We are now interested in the effects of a dynamic game state where players join the game after some period of time. Here, we are interested in whether it is necessary to recalculate the location of the SN after a number of new players join. Consider a 40 player node topology taken from the original set of 50 nodes in the previous experiment. We use r-SPOT (one of the SPOT algorithms defined in [3]) to determine a location of the SN (node 7) and measure the RTT from all the players to that SN. Now suppose 10 more players join the game and the SN is not re-evaluated with all 50 players. With node 7 still serving as the SN, one of the new nodes joining is unable to play the game due to a large RTT (1545 ms to node 7). However, if the SN is re-evaluated and moved to node 8, all players are able to participate. The results of this experiment are shown in Figure 8 with a whisker-box plot of all RTTs. From the figure, when node 7 is the SN in the 40 node experiment the average RTT is 44 ms. Once the 10 additional nodes join, the average RTT jumps to 81 ms with the outlier node experiencing a large delay to SN 7. When the topology is re-evaluated the SN moves to node 8 and the average RTT drops to 60 ms. This illustrates the importance of re-evaluating the SN assignment in order to maximize the number of players in the game.

5. Emulation

Emulation experiments with SPOT are now presented. Two different placement algorithms were developed for SPOT, r-mod and r-SPOT. The r-mod algorithm takes a related approach [3] and adopts it to our problem. The r-SPOT algorithm improves upon r-mod with various optimizations [3]. The network topologies consist of hierarchical networks of size 100, 200, 300, and 400. The Emulab physical nodes were of the type pc3000. The pc3000 are 3 GHz Pentium 4 CPUs with 2 GBytes of RAM. There are 160 nodes on Emulab of this type. In order to create larger sized

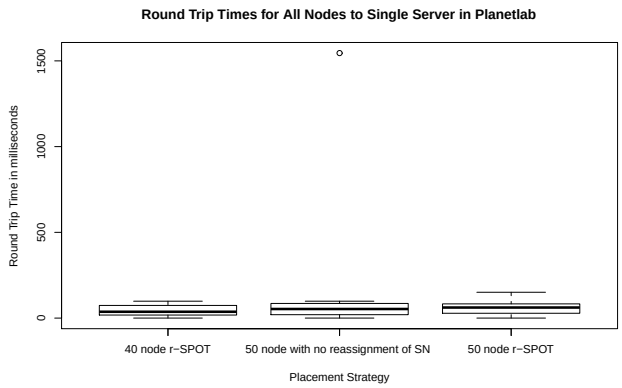

Fig. 8: Round Trip Times from each node to a single server for a 40 node topology with r-SPOT SN placement, a 50 node topology using the 40 node SN placement, and a 50 node topology with a new r-SPOT SN placement.

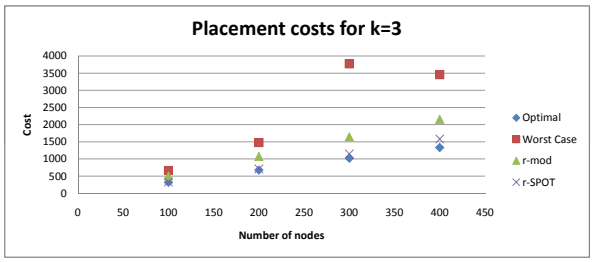

Fig. 9: Results of placement for three SNs with various sized network topologies [3].

networks, virtualization is used with Emulab. The virtual machine consisted of FreeBSD jails with an assignment of 10 virtual machines per physical node. This allowed for larger experiments (greater than 160 nodes) while not over-utilizing a single physical machine. The radius value (r) was set to 2 in all of the experiments, consistent with related work [9]. The experiments were evaluated with a k value (number of SNs) of 1 and 3. The SPOT software was loaded on each node and a special bootstrap node was also created running the bootstrap software. Each node ran a script which would execute the SPOT Java application and connect to the bootstrap node.

In Figure 9 (from [3]) the results for k=3 show improved placement for r-SPOT compared to r-mod. For example, in the 300 node experiment r-SPOT was only 13% above the optimal compared to r-mod with an average placement cost that is 61% higher than optimal.

In addition to the cost of the resulting network topology upon algorithm completion, other metrics of interest are also evaluated for r-SPOT. The total number of iterations to reach a finishing state and total system time necessary before the experiments finished are discussed next.

The first experiment uses a whisker-box plot to display the system time necessary to locate SNs in a network topology (Figure 10). From this graph we can see total system time increase as the number of SNs increase. This is to be expected as increasing the number of SNs to place increases the total amount of work and the time to complete it.

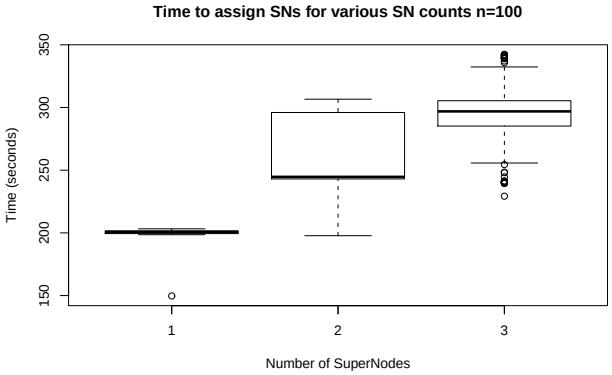

Fig. 10: Total time to place SNs in r-SPOT algorithm.

Additional experiments are run to understand some of the tradeoffs between r-SPOT and an optimal placement strategy. In Figure 11 the total time to locate three SNs is computed for various topology sizes in comparison to a global solution. The global solution requires full topological information, which is equivalent to increasing the neighborhood size to include all nodes in the network. At the 250 node size it starts to become increasingly more expensive to place nodes with the centralized optimal solution.

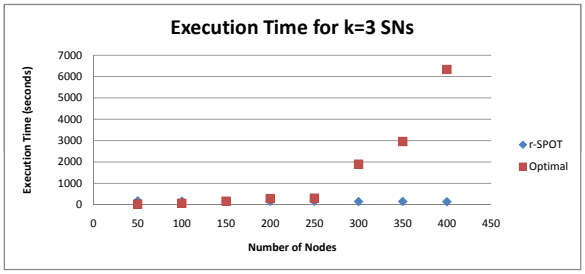

Fig. 11: Time to place 3 SNs in r-SPOT.

This section has presented the SPOT system deployed on Emulab using the r-mod and r-SPOT algorithms. The evaluation demonstrates the improved placement performance of the r-SPOT algorithm and how both approaches compare to the optimal results.

6. Simulation

6.1 Introduction

Simulation provides an excellent opportunity to extend the previous results to large, and more realistic network

topologies. With research testbeds, a node size limit is reached somewhere in the hundreds of nodes. In order to evaluate systems larger than that, simulation is very useful. Also with simulation, the experiments can easily run on different topologies.

In order to provide simulation with SPOT, a discrete event simulator called SPOTSim was developed. SPOTSim was written in Java and models all of the communication between nodes running SPOT. It also interfaces with the same ILP solver (GLPK) as SPOT.

The simulator was developed after creating SPOT, therefore the true functionality of the working system was captured in the simulation environment. Typically a simulator is developed first and the final implementation ends up behaving somewhat differently due to real world constraints. This is much less the case with SPOTSim, which provides a fairly realistic model of SPOT's behavior.

6.2 SPOTSim Evaluation

In order to test the validity of the model, experiments were run on Emulab with SPOT and on SPOTSim with the topology deployed on Emulab. The first experiment illustrates the placement scores of both SPOT and SPOTSim for k=3 in Figure 12.

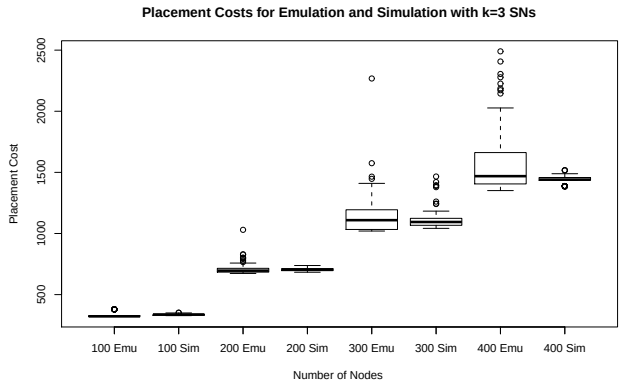

Fig. 12: Comparison of Emulation and Simulation placement results for various topologies locating three SNs.

Although the simulation and emulation results are closely related, some differences are found in the more costly emulation placement results. These results are not captured by the simulation model. The higher scores in emulation are due to the missed opportunities for SNs to join with other SNs and create larger neighborhoods. In the simulator, the merge operations occur with perfect knowledge of the other available SNs.

The strength of the simulation model is its ability to experiment on a larger number of nodes and more interesting topologies. To accomplish this a topology generator was used to aid in the design and creation of larger more realistic topologies. The topology generator used is BRITE [10]. Based on previous related work [9] the BA-2 router level topology was selected for our simulations. A range of sizes were created (500, 1000, and 1500 nodes) using the BRITE's default growth rate parameters.

Using the three topologies created with BRITE, simulations were performed and the average placement cost is presented. In Figure 13 SPOTSim and the optimal results are presented. From the figure, SPOTSim is able to place SNs with a cost of less than twice that of the optimal.

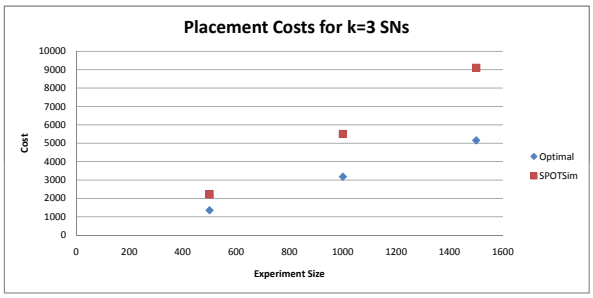

Fig. 13: Placement costs for SPOTSim simulations compared against optimal placement costs.

Simulations were also run with various neighborhood node sizes. Thus far, all simulation and emulation results were executed with a r or neighborhood size of two units, where units are some metric such as network hops. In Figure 14 three different values for the default neighborhood size are experimented with placing k=3 SNs in the 500 node router topology. From the figure, increasing the default neighborhood size r reduces the cost of placing SNs. The mean placement costs are 1045, 954, and 965 for r values of 1, 2, and 3. A default neighborhood size (r) of two provides a 7% reduction in median cost and setting r to three reduces the costs by an additional 1%.

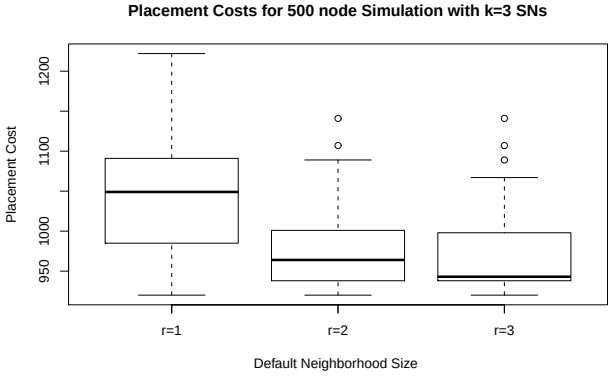

Fig. 14: Whisker-box plot of placement costs for SPOTSim for 500 nodes with varying initial neighborhood sizes.

6.3 Revisiting the Game Server with Simulation

We now return to the game server placement problem initially presented using Planetlab. With the simulator we are able to insert a Planetlab topology into the simulator to evaluate the performance of SPOT. Due to the unreliable nature the Internet and Planetlab, evaluating the same set of more than 50 nodes can be rather challenging. With this switch to simulation we are able to investigate a larger topology of 263 Planetlab nodes around the world. The RTTs were collected to and from all nodes in the evaluation. Next we evaluated the individual RTT from each node to the candidate SN. Figure 15 depicts the maximum number of players that can join the candidate SN server. A player can join the server if the RTT to that server is less than 180 ms. From the figure, 107 potential SNs can support 200 or more players in a single game (the largest is 236), also 116 potential SNs support 100 or more players. The least number of players came from the pair of nodes located in Uruguay, supporting 4 and 5 players each. Finally, a comparison is provided showing the relation between solving the k-medians problem and finding an SN that supports the most number of players. In Figure 16, a bar graph represents the total number of players that could connect to a single SN in the best and worst case. Also shown are the results of the k-medians optimal solution and the SPOTSim solution with respect to the number of players each SNs supports. From the results, the maximum number of players in a single game with the best SN placement is 236 players, while the ILP solver and SPOTSim selected nodes supporting 227 and 226 players, respectively. This helps to demonstrate the ability of SPOT to determine SN locations for a first person shooter.

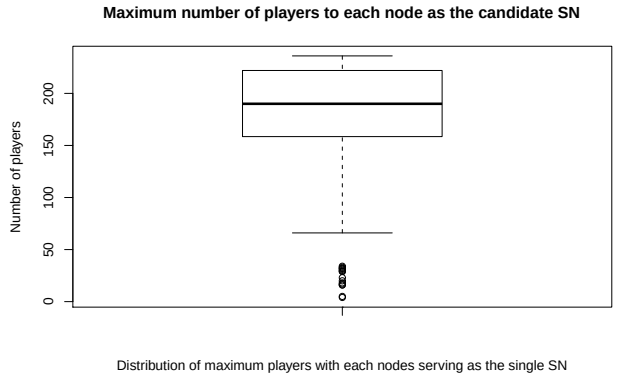

Fig. 15: Max players for each nodes as SN.

7. Conclusion

This paper has presented the SPOT system evaluated in three different environments, experimentation, emulation and simulation. This research demonstrates a fully functioning P2P system that provides SN placement for a range of applications. An example application with SPOT locating a game server showcases the benefits of this network application.

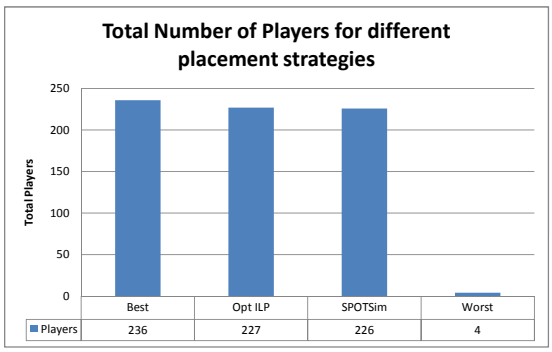

Fig. 16: Number of players supported for various SN selection strategies.

Through this variety of experiments a better understanding of the performance of SPOT is gained. This work motivates the use of different domains to evaluate a large distributed system. Finally, the creation of the SPOTSim simulator allows researches to discover the impact of different placement algorithms on a much wider set of network topologies.

References

[1] B. White, J. Lepreau, L. Stoller, R. Ricci, S. Guruprasad, M. Newbold, M. Hibler, C. Barb, and A. Joglekar, "An integrated experimental environment for distributed systems and networks," in *Proc. of 5th Symp. on Operating Systems Design and Implementation*, Dec. 2002, pp. 255–270.

[2] L. Peterson, T. Anderson, D. Culler, and T. Roscoe, "A blueprint for introducing disruptive technology into the internet," *SIGCOMM Comput. Commun. Rev.*, vol. 33, no. 1, pp. 59–64, 2003.

[3] T. Sproull and R. Chamberlain, "Distributed algorithms for the placement of network services," in *International Conference on Internet Computing*, Las Vegas, NV, Jul. 2010.

[4] "GLPK - GNU Linear Programming Kit." [Online]. Available: http://www.gnu.org/software/glpk/

[5] G. Smaragdakis, V. Lekakis, N. Laoutaris, A. Bestavros, J. W. Byers, and M. Roussopoulos, "The EGOIST Overlay Routing System," in *Proceedings of ACM CoNEXT 2008*, Madrid, Spain, December 2008.

[6] T. Sproull, "Design and evaluation of distributed algorithms for placement of network services," PhD Dissertation, Washington University in Saint Louis, Department of Computer Science and Engineering, Aug. 2009.

[7] G. Armitage, M. Claypool, and P. Branch, *Networking and Online Games: Understanding and Engineering Multiplayer Internet Games*. John Wiley & Sons, June 2006.

[8] G. Armitage and P. Branch, "Distribution of first person shooter online multiplayer games," *International Journal of Advanced Media and Communication*, vol. 1, no. 1, pp. 59–75, October 2005.

[9] N. Laoutaris, G. Smaragdakis, K. Oikonomou, I. Stavrakakis, and A. Bestavros, "Distributed placement of service facilities in large-scale networks," in *IEEE Infocom 2007*, Anchorage, AK, May 2007.

[10] A. Medina and J. Beyers, "Brite: an approach to universal topology generation," *9th Int'l Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, August 2001.