Router Partitioning using a Genetic Algorithm

Zack Fout
Computer Science and Engineering Department
Southern Methodist University
Dallas, Texas USA
zfout@smu.edu

Andrew Fulsom
Computer Science and Engineering Department
Soutern Methodist University
Dallas, Texas USA
afulsom@smu.edu

Harley Swick
Computer Science and Engineering Department
Soutern Methodist University
Dallas, Texas USA
hswick@smu.edu

JD Francis
Computer Science and Engineering Department
Soutern Methodist University
Dallas, Texas USA
jdfrancis@smu.edu

ABSTRACT

In this paper we determine partitions of a network using a genetic algorithm (GA) in order to minimize the mean slowdown of packet flows throughout the network. Increasingly, network performance has become an integral portion of the quality of services delivered over the Internet. As a result, information about services furnished by the application and transport layers must be provided to the network layer, in order to achieve application-specific network performance. This new paradigm is a significant departure from the architectural implications of the end-to-end principle of networking, and therefore new methods of defining computer networks with software must be sought. Our proposed solution to achieving application-specific performance constraints is to partition networks into end-to-end paths according to the lengths of packet flows serviced. We achieve this partitioning of networks with a genetic algorithm. We will compare the global mean slowdown of network performance for both unpartitioned networks and networks that have been partitioned using our genetic algorithm and find that networks partitioned with our genetic algorithm reduce the global mean slowdown of network performance with a variety of packet distributions. Our simulation results show that partitioned networks have the potential to reduce the slowdown of network performance for well defined packet distributions.

1. INTRODUCTION

Our research approach is motivated by the increasing need for end-to-end services to provide network performance as a fundamental part of the provided application[3]. Under the current end-to-end principle that is employed by most networks on the Internet today, application-specific information is solely handled by the end nodes communicating with one another while the intermediary nodes of the network who transit packets between the end nodes do not use application-specific information to inform their routing decisions. This paradigm has a number of advantages, the foremost being the simplification of the network layer by requiring applications to implement the functionality that is specific to their needs. However, as network performance becomes an increasingly integral portion of services that are provided over the network, applications must have some means by which they influence routing decisions made at the network layer in order to maintain quality of service. This new paradigm in end-to-end routing, where the performance of the network is directly proportional to the quality of the service being provided, is at odds with the well established division of network communication into a layered hierarchy because the network layer must be aware of application layer performance constraints in order to provide the level of network performance the application requires. Thus, the network layer decisions are increasingly defined by the software that uses them to provide services to users. Therefore, the end-to-end principle of computer network design must be amended in order to accommodate the requirements of these new service models. Our area of interest, then, is to devise a method by which application layer information may influence routing decisions made at the network layer, in order to provide for more reliable levels of network performance. We begin by defining our methods of characterizing network performance and our generalized methodology for approaching this problem.

Consider a routing network where packet flows arrive on one side of the network topology and must flow through the network and exit on the other side of the network topology. Further, it is possible that a packet flow may exist between every possible pairing of input and output nodes at the edges of the network topology. A packet flow is defined here as a sequence of packets that are exchanged between a source and destination host at opposite edges of the network topology. If a shortest path algorithm routing approach is employed by the network, the intermediary network nodes in the center of the network are likely to receive a significantly larger portion of the total network traffic than intermediary nodes that exist at the edge of the network topology. This scenario arises because the shortest path between two nodes at opposite edges of the network will likely be composed of a majority of intermediary nodes that exist in the center of the network topology rather than being composed of a majority of nodes at the edge of the network topology. The end result is that intermediary nodes at the center of the network topology will transit a disproportionate amount of overall network traffic when compared to intermediary nodes at the edge of the network. This causes network congestion

at these nodes which may affect the performance of the network as a whole. In order to characterize this scenario, we use the mean delay of all packet flows in the network to determine the overall performance provided by the network. The induced delay of a packet flow may be characterized by the following formula:

$$flow\ slowdown = \frac{(finish\ time\ -\ available\ time)}{flow\ time}$$

where the finish time is the point in time at which the packet exchange is terminated, the available time is the total amount of time that has been allocated to the packet flow, and the flow time is the total number of packets sent during the exchange. By calculating this metric for all packet flows throughout the network and taking an average, a general characterization of the performance of the network may be made.

Now that we have determined a method for characterizing network performance, we may formally define our area of interest: we seek a method by which a given network may be partitioned into subsets of routers that are biased toward packet flows which require a long length route through the network, packet flows which require a medium length route through the network, or packet flows which require a short length route through the network. Our intuition is that by dividing the network into partitions of routers based upon the length of the packet flows serviced by those routers the traffic load at the interior of the network topology will be alleviated because both long and medium length packet flows will tend to follow a path through the edge of the network rather than a shorter path through the middle. We will utilize a genetic algorithm to determine this partitioning of the network topology. By using the global mean delay within the network as our fitness function for our genetic algorithm, we will simulate partitions of the network based on flow length characteristics and determine if the optimization provided by our genetic algorithm balances the disadvantages of the increased network complexity and computational cost imparted by the GA. We find that GAs are well suited toward reaching optimal network partitions with certain packet size distributions, while other packet size distributions do not impart a significant reduction to the global mean delay of packet flows within the network. We theorize as to the nature of this difference in optimality versus computational cost with different packet size distributions, and conclude that under certain circumstances the benefits of implementing a GA defined network routing decision structure exceed the computation complexity therein.

In the following sections, we will introduce the related research that has been performed in the field of software defined network partitioning for end-to-end routing of communications, and the applications of genetic algorithms to this field. Next, we will introduce our approach to the problem, and then our detailed research methods. Finally, we will explore the results of our network simulation, analyze these results in the context of our problem statement, and discuss the conclusions that we have reached.

2. RELATED WORK

A great amount of research has been focused upon the end-to-end principle of communication over a network. The end-to-end principle argues that application-specific functionality should only be implemented by the end hosts communicating with each other and that therefore the intermediary nodes which constitute the network topology should be concerned with the routing of packets and nothing more. This paradigm imparts a number of advantages to the logic needed to successfully route packets between end hosts. Namely, the method conserves internal network resources by placing the implementation of desired functionality upon the end hosts who want to communicate, rather than placing this functionality upon the network itself. This increases the reliability of the network by simplifying the underlying routing logic, because the end hosts must implement the functionality they desire themselves in order to achieve completeness within the network, and therefore the network need not be made subject to this constraint as well[3].

However, the end-to-end principle of network communication has its disadvantages as well. One of the major disadvantages presented by this communication model is that the network does not implement any dynamic routing logic; rather, the network simply ensures that packets are sent to their desired destination, and nothing more. This "dumb" network poses a number of optimization issues, one of the most glaring issues being that increased traffic in portions of the network may impact traffic in other portions of the network that are ostensibly unrelated[6]. Additionally, there are a number of end-to-end routing problems that have been found to be prevalent in the Internet that we aim to avoid[5]. By optimizing our partitioning method for minimizing the global mean slowdown of the entire network, we attempt to avoid many of the potential issues caused by end-to-end routing. These new issues in computer network routing, where application-specific information has an increasingly larger impact on the efficacy of the underlying routing infrastructure, have important implications for the future of the end-to-end principle of computer network routing. The most obvious implication is that routing networks must have some facilities by which they may become aware of the type of application traffic that the routing network is transiting if they are to scale well with applications where network performance is an integral part of the delivered service. While this paradigm is at odds with the layered hierarchy of computer networks, where each layer acts as a black box providing services to each other, it has become clear that the software using a computer network has increasingly come to define how that network performs its most basic tasks and therefore new methods of intelligent, application aware routing must be devised. We argue for the usage of a genetic algorithm to implement this application aware routing, as it provides the advantage of leveraging some application information within the network layer while attempting to keep both the network and the application layer as distinct from one another as possible.

Furthermore, an extensive amount of research has been undertaken in the field of genetic algorithms with respect to their applicability to solving optimization problems that present themselves in computer networks. In particular, a number of researchers have already studied the application of genetic algorithms to the problem of determining shortest path routes for network flows, often building upon the logic employed by Djikstra's Algorithm[7][4]. While these studies have shown that utilizing genetic algorithms can improve the speed of routing packets through a network, this is not directly related to our area of interest. Additionally, we elected to not utilize genetic algorithms for the underly-

ing routing through our networks in order to focus on our area of interest and compare directly to the Open Shortest Path First method that is prevalent in network routing. Rather, we seek to modify the Open Shortest Path First method in such a way that intermediary network nodes servicing a particular packet flow length will preferentially choose neighboring nodes that preferentially service similar packet flow lengths. Our goal then is not to alter the Open Shortest Path First method of computer network routing per se; rather, we seek to inform the method with packet flow length information so that shortest paths are calculated in terms of lengths of packet flows services and not merely the actual shortest path to a destination node. A great deal of scientific effort has been expended to define the structure of genetic algorithms, their operators, and the limitations they lend themselves to [1].

Finally, partitioning networks into sub-networks has been found to better utilize network resources[2]. By separating one network into multiple networks that are specialized for different flows of traffic, congestion can be more effectively be avoided. The work done by Caria, et al. demonstrates that effectively partitioning a network can improve communications. Our research aims to further improve network performance by using genetic algorithms to quickly find the optimal network partitioning. This is a suitable problem for genetic algorithms to optimize, due to the large nature of the search space we will be exploring; that is, the large number of possible partitions that a given network topology may be characterized by. Normal brute force algorithms fail to achieve practical results given the size of the partitioning search space, and it is our goal to illustrate that genetic algorithms are well suited to approaching this problem by using information about previous configurations of the network to reach an optimal partitioning of the network topology. However, a key distinction between our research and that done by Caria, et al. is how the word partitioning is defined. For Caria, et al.'s work, a partition is a sub-network that performs routing within itself and knows only vaguely how to route to other sub-networks. Our approach defines partitioning as a way of tagging the routers in a network such that the routing of the network is changed to reduce the mean global slowdown of the network by reducing the congestion of routers that are located near the center of the network topology. There are no published works that describe an approach to reducing the global mean slowdown of a network utilizing a partitioning method such as ours. By using a genetic algorithm, we further distinguish this work from any performed prior. This novel method of preventing congestion in a network should provide a new avenue for improving the efficacy of networks, while attempting to preserve the end-to-end principle of networking routing as much as possible.

3. GENETIC ALGORITHMS

Genetic algorithms are a type of search algorithm that model the process of natural selection. Some of the key terms involved with genetic algorithms are: individual, a potential solution; population, a group of individuals; parents, the initial population; children, the offspring of the parents or of earlier children; and genes, the descriptive characteristics of individuals. Genetic algorithms model the evolution of a population over a given number of generations or until a terminating condition for the algorithm has been reached. The most fit members of the population, that is, the memebers of the population which have been evaluated as candidates for breeding by the fitness function, reproduce to create a new generation of fit individuals by sharing genes and through mutation processes[7]. The general algorithm for a GA is as follows:

1. Create an initial population of parents.
2. Rank the population using a specified fitness function.
3. Choose the most fit individuals to create children.
4. Create a new generation of children using crossover (combining parts of both parents) and mutation (randomly changing some parts of a parent).
5. Replace the least fit individuals with the newly created children.
6. Repeat steps 2 - 5 until the generation number is reached or the termination condition is satisfied.

When formulating a genetic algorithm, it is important to first define both the populations under scrutiny and the individuals which compose the population. We have defined a single population to be the collection of possible partitions for a single network topology. That is given a network topology, an individual is defined to be a set of partitions of the nodes within the topology and the population of interest is the set of all possible sets of partitions within the network. Further, it is equally important to define how individuals within the population will be evaluated according to their fitness. We have defined our fitness function to be the reduction of the global mean delay of packet flows within a given partitioning of a network. To understand how this fitness function operates, consider two partitionings of a network wherein the first partitioning tags all nodes in the network as short, while the second partitioning contains network nodes with long tags along paths over the edge of the network and nodes with short tags through the middle of the network. When comparing these two individuals, it is likely that the second individual will have a reduction in global mean delay of packet flows throughout the network and therefore be evaluated as more fit than the first individual.

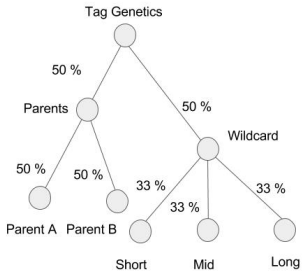

Figure 1: Crossover and Mutation Algorithm

Now that the basic goal of our fitness function has been presented, it is important to understand how the fitness function actually operates. After a partitioning of a network has been established, that is, after an individual has been selected from the population of possible partitions, the individual is evaluated by simulating packet flows across all

nodes within the network and parameters are used to constrain the number of packets sent. Then, using the formula described in our introduction, the flow slowdown is calculated for each individual path in the partitioning and these calculations are averaged to obtain the global mean slowdown of the partitioning. By performing these operations for each partitioning of the network, partitionings may be characterized and compared according to the fitness criteria in order to select individuals which maximize the global mean slowdown of packet flows.

With our fitness criteria established, we must now consider how the genetic operators of crossover and mutation are implemented. As illustrated in Figure 1 above, our crossover function is relatively simple. Given two parent individuals, who have distinct partitionings of the underlying network topology, a 25 percent probability exists that partitions of the network that compose Parent A will be passed on to the new individual, and a 25 percent probability exists that partitions of Parent B will be passed on to the new individual. This process is performed by computing this probability for each partition in the parent and appending the partition to the new individual if it is chosen. The remaining 50 percent of partitions that compose the new individual are determined using our mutation criteria. This process simply computes a random path partition with the given tags of short, medium, or long and appends it to the new individual. While this is a relatively large mutation rate compared to other implementations of genetic algorithms, we were motivated to incorporate this degree of variability into our genetic algorithm given the large size of our search space.

4. ROUTER PARTITIONING APPROACH

For our approach we implemented a genetic algorithm in Python to create partitionings of a given network topology. While this approach is designed to improve upon the Open Shortest Path First (OSPF) method that is typically used in network routing, it makes use of Dijkstra's Algorithm for the actual end-to-end routing through the network. Dijkstra's Algorithm operates by comparing the edge weights, or costs, of a graph to rapidly find the least costly path from one node to another. Our method uses Dijkstra's Algorithm in a similar manner. We modify Dijkstra's Algorithm by tagging the nodes of the network in such a way that they are interpreted by Dijkstra's Algorithm differently depending on the length of the packet flow that the routing path is being computed for.

The algorithm 'tags' nodes within the network with a number 0, 1, or 2 corresponding to the three different classifications for the types of flows; short, medium, or long. When Dijkstra's algorithm evaluates an edge between a node, if the tag for the next node in the path does not match the type of flow that is currently being routed, it also adds the tagged value, thus modifying the result of the calculated edge weight. This establishes the end-to-end routing of the network in such a way that routers will primarily transmit packet flows of the length that they are tagged for, while maintaining a fully connected network. If routers only transmitted flow lengths that they have been tagged to prefer, then it is possible for some sections of the network to become disconnected from the rest of the network for some flow lengths. The optimal tagging for the network, then, is determined using our genetic algorithm, which finds the best solution relatively quickly compared to other heuristic methods.

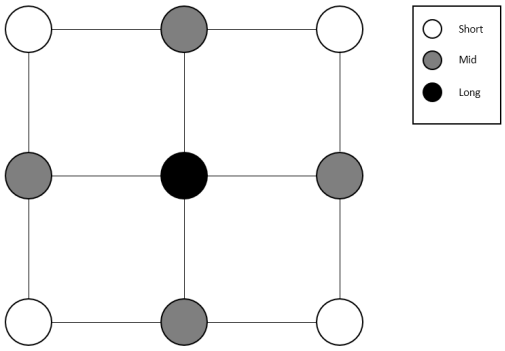

Figure 2: Example Network Partitioning

When selecting the most fit members of a generation, the top two individuals are selected to breed the next generation of children. The "top" individual is selected by comparing each individual of the population. This is repeated for the remaining individuals in the population to get the second most fit individual. Our fitness function, used to quantify how fit an individual is to reproduce, is defined in three different ways. The first way is to compare the total time that it takes to run a fixed group of packet flows through individual router partitions. The second is to compare the number of iterations of packet flow simulation it takes to fully send all of the packet flows through the individual router partitions. The third method is to compare the mean slowdown of flows through the individual partitions. For our final tests, we used the fitness function for comparing mean slowdown.

When breeding children, we use the top two individuals from the previous generation to create thirty new individuals for a total population size of thirty-two. Each child is created by generating a collection of tags by crossing over tags from each parent and by randomly mutating tags to one of the three possible taggings. For each tag within an individual, there is a 50 percent chance that the tag will be derived from the parents or from a mutation. For those that are derived from the parents, there is an equal chance that the tag will be selected from either parent. For those that are derived from mutations, there is an equal chance that it will be any of the three possible tags. Figure 1 illustrates this probability breakdown in the form of a tree. This selection process is done for each newly created individual in the population. The two parents that are used to create the new generation are kept in the population to ensure that there are no steps back in fitness from generation to generation. This process is performed a specified number of times before the most fit individual from the last generation is selected to be the final partitioning for the network. For our tests, fifty generations were used to create suitably fit solutions.

5. RESULTS

To test our algorithm, we created an untagged network and evaluated the mean global slowdown across the network while routing packets through the network using Dijkstra's algorithm for routing. We then partition that same network using the genetic algorithm and test routing the

same packets through the network. This effectively compares traditional Open Shortest Path First to our method of tagging the network for end-to-end routing. For the comparison between the two methods, two hundred simulations are performed and the average mean global slowdown of the network is calculated for both methods. The percent change from the untagged network to the tagged network is then calculated. The table below shows data from tests using a genetic algorithm that is trained on uniform data and subsequently compared to an OSPF network using uniformly distributed data.

Improvement from OSPF Network to Partitioned Network Using Uniformly Distributed Packet Flows(%)
0.647875665314
-2.08484661479
-1.52990187485
3.90123132723
-4.36481727998
6.14048750503
1.95285966301
4.12204640655
4.47935496644
1.496636389
-6.70771144794
-5.15678143639
-4.82753062729
-2.51457277843
-7.09883802956
-2.81351859922
6.51104368616
2.09626410268
0.164327330202
-0.606857463674
-0.518576263903
1.05411064041
-0.989041991993
Average
-0.2891198577

Tests were performed to compare genetic algorithms trained on a variety of packet distributions and compared to Open Shortest Path First networks using different types of packet distributions. These are represented graphically in Figures 3-5. Each is a histogram showing the results of two hundred simulations for both the OSPF network and partitioned network. Figure 3 shows the results using a partitioning trained on uniform data and tested using data with a Pareto distribution and high Pareto index.

For this comparison, the partitioned network was 1.5% more performant than the OSPF network, meaning that the average global mean slowdown for the partitioned network was 1.5% lower than that of the OSPF network.

Figure 4 shows the results of a comparison of a partitioning trained on data that follows a Gaussian distribution compared to an OSPF network using Gaussian distributed data for the simulations. This comparison resulted in the partitioned network performing 2.5% better that the OSPF network.

Finally, Figure 5 visualizes the results of the comparison between an OSPF network and a network partitioning trained on Pareto distributed data with a high Pareto index and simulated using data that also followed a Pareto distribution using a high Pareto index. This comparison resulted

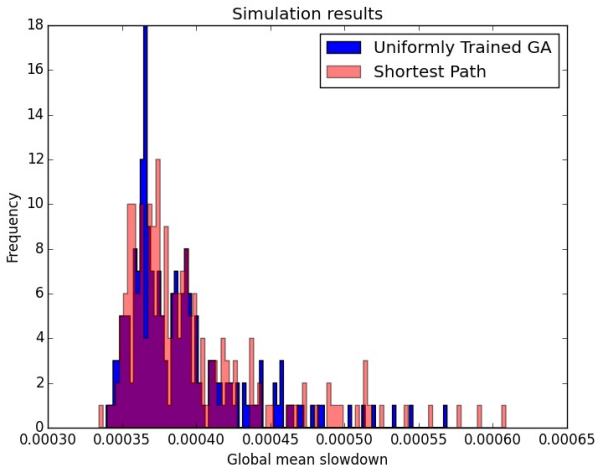

Figure 3: Uniformly trained partitioning tested with Pareto data using a high Pareto index

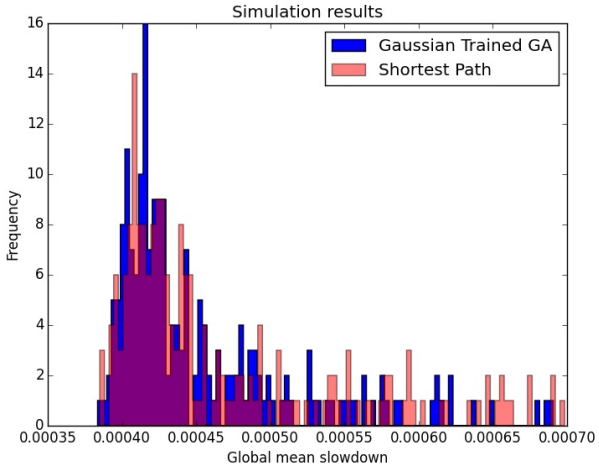

Figure 4: Gaussian trained partitioning tested with Gaussian data

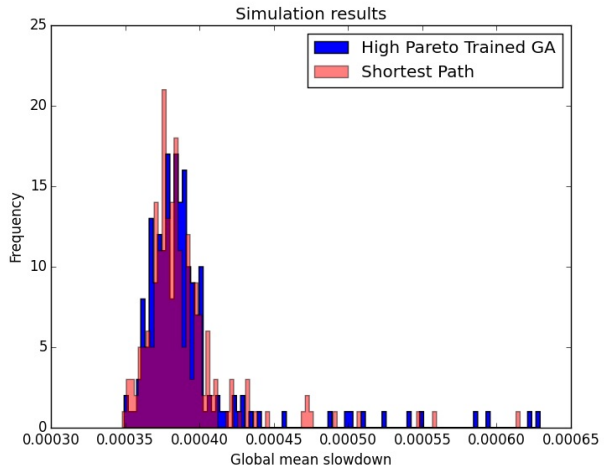

Figure 5: Pareto trained partitioning tested with Pareto data using a high Pareto index

in a net loss in performance, with the partitioned network having a higher mean global slowdown than the OSPF network.

6. RESULTS ANALYSIS

Overall the results of the genetic algorithm determined paths versus the shortest paths are that the genetic algorithm can get a small percentage of performance increase. The level of performance is dependent on several factors. The first factor is that the distribution of packet sizes makes a difference. Figure 3 shows how an interesting case where the training data was different than the testing data. In this case there was an actual performance increase over the shortest path algorithm. Figure 4 shows a more widespread distribution of results as a result of the Gaussian distribution. In these scenarios the GA seems to do better. Where as in Figure 5 on the high Pareto index the genetic algorithm did not do well. More runs of the simulations can cause other varying results. The average performance of the genetic algorithm is never more than 2%. Which seems difficult to justify the computational cost of training the algorithm. Higher results are given when the the training/testing data is extremely patterned. In such cases the algorithm is able to overfit the data. However, the likelihood of this happening is low.

7. CONCLUSIONS

We implemented an end-to-end routing method using router partitioning with a genetic algorithm designed to minimize the mean global slowdown in the network. Routers are tagged to indicate what type of packet flows they will primarily transmit, either short, medium or long. The optimal collection of router taggings for a particular network is generated using a genetic algorithm. This breeds the most fit solution for a particular network using training data that can be configured to any type of data distribution. The network then routes the packet flows using Dijkstra's Algorithm with varying costs based on the length of the packet flow and the tags of the nodes that it passes through. Overall, while this method can yield less congested networks, the computational cost of generating the partitioning for a network does not seem to be worth the marginal decrease in mean global slowdown that it yields. However, if a network can guarantee that the packet flows moving through it will match a particular distribution, then the genetic algorithm can overfit the data and produce higher results. In this case, then the cost of partitioning the network may be worth the reduced mean global slowdown.

8. REFERENCES

[1] Chang Wook Ahn and Rudrapatna S Ramakrishna. A genetic algorithm for shortest path routing problem and the sizing of populations. *Evolutionary Computation, IEEE Transactions on*, 6(6):566–579, 2002.

[2] Marcel Caria, Tamal Das, Admela Jukan, and Marco Hoffmann. Divide and conquer: Partitioning OSPF networks with SDN. *CoRR*, abs/1410.5626, 2014.

[3] James Kempf, Rob Austein, et al. The rise of the middle and the future of end-to-end: Reflections on the evolution of the internet architecture. Technical report, 2004.

[4] Gihan Nagib and Wahied G Ali. Network routing protocol using genetic algorithms. *International Journal of Electrical & Computer Sciences IJECS-IJENS*, 10(02):40–44, 2010.

[5] Vern Paxson. End-to-end routing behavior in the internet. In *ACM SIGCOMM Computer Communication Review*, volume 26, pages 25–38. ACM, 1996.

[6] Jerome H Saltzer, David P Reed, and David D Clark. End-to-end arguments in system design. *ACM Transactions on Computer Systems (TOCS)*, 2(4):277–288, 1984.

[7] Yagvalkya Sharma, Subhash Chandra Saini, and Manisha Bhandhari. Comparison of dijkstra's shortest path algorithm with genetic algorithm for static and dynamic routing network. *International Journal of Electronics and Computer Science Engineering*, 1(2):416–425, 2012.