

# A Multidimensional Robust Greedy Algorithm for Resource Path Finding in Large-Scale Distributed Networks

Aida Vosoughi  
Department of Electrical and  
Computer Engineering  
North Dakota State University  
Fargo, ND, 58108-6050, USA

Kashif Bilal  
Department of  
Computer Science  
COMSATS Institute of IT  
Abbottabad, Pakistan

Samee Ullah Khan  
Department of Electrical and  
Computer Engineering  
North Dakota State University  
Fargo, ND, 58108-6050, USA

Nasro Min-Allah  
Department of  
Computer Science  
COMSATS Institute of IT  
Islamabad, Pakistan

Juan Li  
Department of Electrical and  
Computer Engineering  
North Dakota State University  
Fargo, ND, 58108-6050, USA

Nasir Ghani  
Department of Electrical and  
Computer Engineering  
University of New Mexico  
Albuquerque, NM  
87131-0001, USA

Pascal Bouvry  
Department of Electrical and  
Computer Engineering  
University of Luxembourg  
Luxembourg

Sajjad Madani  
Department of  
Computer Science  
COMSATS Institute of IT  
Abbottabad, Pakistan

## ABSTRACT

This paper proposes a robust greedy algorithm and four of its variants for the resource path finding problem in distributed networks. In contrast to the existing solutions that rely on a single minimum cost path for each request, the proposed algorithm makes use of finding “robust” paths for each request within the network. We give a mathematical definition of robustness for the resource path finding problem in distributed networks. The four proposed variants are then compared with each other and with a traditional “non-robust” path finding algorithm. The simulation results show interesting improvement in solution quality when robustness is incorporated into the path finding algorithm for distributed networks.

## 1. INTRODUCTION

Peer-to-peer systems are distributed systems consisting of interconnected nodes with the purpose of sharing resources such as content, CPU cycles, and storage. These networks must be capable of adapting to failures while maintaining acceptable connectivity and performance [2]. Peer-to-peer architectures have been employed for a variety of different application categories. One of the most important categories of peer-to-peer networks which has gained much popularity as of recent is content distribution networks category. As distributed systems become more sophisticated, features

like performance and robustness become more important.

Any peer-to-peer content distribution system relies on a network of peers within which requests and messages must be routed with efficiency and fault tolerance. Ensuring that authorized users have access to data and associated assets when required is very important. This property entails stability in the presence of failure. The routing or resource path finding mechanism is crucial to the operation of the system, as it affects the system’s adaptability to failures and performance. For a survey on the robustness of the existing P2P networks see [8]. In many P2P networks, robustness and adaptability to resource failures are addressed by adding redundancy to networks in different forms. Such as replicating data in different nodes and replicating links between nodes [5]. However, cost of redundancy is always a concern. On the other hand, the existing algorithms for resource path finding do not account for the probable resource failures. Instead, they find a single best path only on the basis of the current conditions without looking into the future for probable uncertainties such as resource failures. Every time a path fails a new path must be found and assigned to replace the broken path, resulting in a large number of path switches for each request.

Robustness can be defined as the degree to which a system can function correctly in the presence of uncertainties in system parameters [9]. In the context of resource path finding algorithms, the robustness of an algorithm may be defined as the degree to which the algorithm can find paths conforming to the Quality of Service (QoS) requirements for each request such that QoS requirements corresponding to that request remains satisfied for a longer period of time and the number of required reroutings become less in the presence of resource failures. Therefore, the current state of the art resource path finding methodologies for distributed networks cannot be classified as robust. The reason being that the heuristics: (a) solely aim to find a single near optimal path given some constraints and (b) do not account for possible resource failures, such

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

*FIT'10*, December 21-23, 2010, Islamabad, Pakistan.

Copyright 2010 ACM 978-1-4503-0342-2/10/12 ...\$10.00.

as terminals and links failures.

Lack of robustness in current path finding algorithms results in unreliable and inefficient performance. Therefore, robust algorithms for finding paths of resources in distributed networks are timely to consider and important to discover. In this paper, we propose a generalized robust resource path finding algorithm and we analyze and compare the degree of robustness of four of its variants. In the proposed method, a set of multiple alternative paths are found for each request and the path assigning process takes the probability of path failures into account as a key criterion.

## 2. PROBLEM FORMULATION

In distributed networks, when one of the users generates a service request, the path finder algorithm must specify a minimum cost path of resources from the source terminal to the destination terminal in which the requested service resides. Moreover, the discovered path must satisfy the user-defined QoS constraints.

The whole of the distributed network can be viewed as an undirected graph with each node representing a resource and edges representing the links between resources. We denote a resource by  $r_i$  and the set of all of the resources in a distributed network by  $R = \{r_1, r_2, \dots, r_N\}$ . Let the physical distance and bandwidth of the link between  $r_i$  and  $r_j$  be denoted by  $d_{ij}$  and  $b_{ij}$ , respectively. If there does not exist a link between resources  $r_i$  and  $r_j$ , then  $d_{ij} = 0$  and  $b_{ij} = 0$ . We assume that  $b_{ij} = b_{ji}$ . Relaxing the aforementioned constraint will have no bearing on our problem formulation.

At any given time  $t$  there may be  $M$  active requests present in the distributed network. The set of active requests in the network is represented by a set  $Q = \{q_1, q_2, \dots, q_M\}$ . Each request is denoted by a triplet  $q_i = \langle s, d, ds \rangle$ , where  $s \in R$  is the source node that generates the request,  $d \in R$  is the destination node where the service resides, and  $ds$  is the size of the data block that must be transferred to fulfill the request. A resource path  $p_i$  for a request  $q = \langle s, d, ds \rangle$  is an ordered set of resources connecting  $s$  and  $d$ , i.e.,  $p_i = \{s, r_1, r_2, \dots, d\}$ .

For simplicity, in this paper, we consider latency of a path as the only QoS constraint. Addition of other QoS constraints will have no bearing on our generalized methodology to construct a robust algorithm for resource path finding in distributed networks. The latency tolerance ( $TL$ ) is a user-defined positive number less than one. For each request  $q$ , the following inequality must hold:

$$L(q) \leq \min L(q) \times (1 + TL), \quad (1)$$

where  $L$  and  $\min L$  denote the latency and the minimum feasible latency of  $q$ , respectively. The latency of a link between resources  $r_i$  and  $r_j$  is calculated as [6]:

$$l_{ij} = \frac{d_{ij}}{C} + \frac{ds}{b_{ij}}, \quad (2)$$

where  $C$  is a constant which is called the wave propagation speed of the transmission media. Traditionally, the goal of a path finding algorithm in a distributed network is to find a path of resources for each request which satisfies the QoS constraints. However, resources may fail at any instance of time. Therefore, the goal of our proposed algorithm is to find a set of alternative paths for each request that together guarantee a degree of QoS in the presence of resource failures.

The reader must note that in this paper resource path discovery in a distributed network is assumed to be performed using one of the many existing distributed networks path discovery algorithms

[2, 8], and finding paths in a network is not the aim of this research. In fact the focus of our proposed solution is to always pick the most robust paths among the existing paths that can be discovered using various distributed/centralized heuristic algorithms.

## 3. DEFINITION OF ROBUSTNESS

In this section, we make use of four-step FePIA procedure to define the robustness of a distributed system. FePIA is a general procedure for deriving a robustness metric for any desired computing environment [1]. The name for the above procedure stands for identifying the performance features, the perturbation parameters, the impact of perturbation parameters on performance features, and the analysis to determine the robustness. Each step of the FePIA procedure is now described for a distributed system where we assume, without loss of generality, that latency of the requests is the only robustness requirement:

Step 1: *Describe quantitatively the requirements that make the system be considered robust:* A distributed system may be considered robust if the inequality in Eq. (1) holds for each request within the network.

Step 2: *Identify the system and environmental parameters (a) that are uncertain and (b) whose impact on the system performance features selected is to be evaluated in the given robustness analysis:* We select to analyze the uncertainty of resource failures in a distributed network.

Step 3: *Identify the impact of perturbation parameters on the system performance features:* Resource failures result in path failures which in turn lead to violation of latency requirements in the network.

Step 4: *Determine the smallest collective variation in the values of perturbation parameters that will cause any of the performance features to violate its acceptable variation:* The smallest number of failures which result in violation of the robustness constraint stated in step 1 (latency in our case) defines the degree of robustness of the system. A larger minimum for perturbation value indicates the system is more failure-tolerant. That is, if latency requirements for requests are only violated after a large enough number of failures, the system is said to be robust. This will be discussed more in Section V.

As we will discuss in the following sections, our proposed algorithm for path assigning in distributed systems improves the robustness of the system because it assigns multiple paths to each request and the algorithm picks paths which have a smaller probability of failure. Therefore our proposed algorithm can be considered as a robust method for assigning resource paths in network.

## 4. PROPOSED APPROACH

### 4.1 Preliminaries

#### 4.1.1 PDF

From literature we know that the failure of an "entity" can be modeled as a random variable sampled from a known Probability Density Function (PDF). We represent the time at which resource  $r$  fails by random variable  $T_r$ . In this research, it is assumed that the probability distributions describing the random variables (time of resource failures) were created from measurements of the failure times of actual resources in the network. A typical method for creating such a distribution relies on a histogram estimator [10] that produces PDF. Note that, it is reasonable to assume that failure times of different resources are independent from each other since resources (terminals, links, and etc.) are expected to fail and recover independent of the rest of the network.

The probability of  $r$  failing after time  $t$  can be derived from the Cumulative Distribution Function (CDF) as follows:  $Pr(T_r > t) = 1 - CDF(t)$ . Accordingly, the probability of a working path  $p_i$  failing after time  $t$  is defined as:

$$Pr(T_{p_i} > t) = 1 - \prod_{j=1}^{|p_i|} Pr(T_{r_j} < t). \quad (3)$$

It means the probability is equal to the probability of at least one of the resources in path  $p_i$  fails after  $t$ . This is true because a path fails if at least one of its resources fails. Note that we assume each resource recovers from a failure after a recovery time; thus, some of the resources of path  $p_i$  which is working at time  $t$  may have had failed and recovered before  $t$ . But our proposed algorithm needs to know the probability that a path  $p_i$  fails after the current time ( $t$ ) to decide whether to assign  $p_i$  to a request or not. As it can be seen from Eq. (3), a working path  $p_i$  at current time ( $t$ ) may less probably fail in the future time (in any time  $t_0$  greater than  $t$ ) if many of its resources have had their failure times before time  $t$  (and obviously recovered before  $t$ .) Such a path can be a good choice to be assigned to a request because there is a good chance it may not fail at all.

#### 4.1.2 Cost Function

As mentioned previously, the traditional distributed networks path finding algorithms primarily focus on the path QoS constraints. Therefore, the “cost function” only consists of QoS parameters, such as latency. In contrast, our proposed algorithm must consider both QoS requirements and the failure probability of resources to achieve robust performance.

We define the cost of a path between source and destination at time  $t$  as:

$$cost(p_i, t) = w \times L(p_i, t) + (1 - w) \times Pr(T_{p_i} > t), \quad (4)$$

where  $w$  is a weighting parameter. The definition above reveals that the path with the lowest value of cost function  $cost(p_i, t)$  among the set of all possible alternative paths for a request  $q$  is the path which has the least latency and the lowest probability of failure (in the future) with respect to the weighting parameter  $w$ , at current time. When  $w = 1$ , the cost function is the same as the traditional distributed networks path finding cost function. When  $w = 0$ , the latency of the path is not considered in cost function and the cost is the probability of path failure. If  $w = 0.5$ , the algorithm considers both path failure probability and latency with the same weight. Therefore, parameter  $w$  is determined based on the goal of the path finding algorithm.

## 4.2 Robust Greedy path finding algorithm for distributed networks

Algorithm 1 represents our proposed greedy solution for robust path finding in distributed networks. Before we explain our proposed technique, we define the necessary variables that may make the pseudo-code of Algorithm 1 more readable. Let  $AP_q$  denote the set of  $Max$  number of possible alternative paths of request  $q$  where  $Max$  is a parameter denoting a maximum on the number of alternative paths for  $q$ , and  $Net$  represents a given distributed network graph.  $L(p_i, t)$  is the latency of path  $p_i$  at current time  $t$  and  $l_{ij}$  is the latency of the link between  $i$ th and  $j$ th resources. The counter  $Path\_Fails_q$  counts the number of path failures occurring for request  $q$ . The variable  $vt_q$  stands for the violation time of  $q$ , i.e., the time at which  $q$  no longer conforms to the latency requirements. The flag  $fl(r, t)$  is equal to 1 if and only if resource  $r$  is inactive at time  $t$ . The flag  $flp(p_i, t)$  is equal to 0 if and only if

path  $p_i$  is a working path at  $t$ . Let  $Current\_Path_q$  be the path that is assigned to  $q$  at the current time. Algorithm 1 describes a general greedy methodology based on Eq. (4).

As it can be seen in the pseudo-code, at first all best  $Max$  number of alternative paths from source to destination are found using a function called  $Find\_Alt\_Paths()$  which can be implemented using a k-shortest-paths algorithm [3]. Then the latency of the path with lowest latency is stored in  $minL_q$ . After that, in a while loop and in any instant of time, failed resources are marked as inactive (by setting flag  $fl$ ) and all the paths that use any one of failed resources are also marked as inactive (by setting flag  $flp$ ). Moreover, if a request is currently assigned with a failed path, the flag  $Recompute_q$  is set to show that a new path must be assigned to that request. The process of assigning a new path to a request consists of picking the working path with minimum cost among the set of all alternative paths that conform to latency requirement (satisfy Eq. (1)) at current time. The cost of a path is calculated based on Eq. (4). Since the algorithm always assigns the path with the lowest cost to a request, it is a greedy algorithm. Note that if there is no path that can be assigned to the request that conforms to the latency requirement, the request is considered as violated. Below, we describe the four possible variants of our proposed algorithm.

#### 4.2.1 Latency only fault-tolerant greedy algorithm

If  $w = 1$  in Eq. (4), then  $cost(p_i, t) = L(p_i, t)$ . This will affect line 36 of Algorithm 1 where the cost of each path is calculated. This makes the algorithm assign paths to requests solely on the basis of latencies. Because, multiple paths are found for each request, the algorithm tolerates path failures. However, this is a naive extension to make a given algorithm robust.

#### 4.2.2 Robust greedy algorithm

If  $w = 0$  in Eq. (4), then  $cost(p_i, t) = Pr(T_{p_i} > t)$ . This makes the algorithm assign paths with lowest probability of failure to requests. Latency is not considered in the cost of a path, but it is not sacrificed either, because all of the alternative paths for each request conform to the latency requirement of that request.

#### 4.2.3 Modified latency only fault-tolerant greedy algorithm

Fault-tolerance of a system is directly proportional to the number of redundant resources [7]. Therefore, the fault-tolerance of a distributed network can be increased if two requests with same source and destination ( $q_1 = \langle s, d, ds_1 \rangle$ ,  $q_2 = \langle s, d, ds_2 \rangle$ ) are assigned to two different paths. This variant is different from 1) in which for the same condition, the requests may both be assigned to a same minimum latency path while conforming to the bandwidth constraints.

#### 4.2.4 Modified robust greedy algorithm

This variant is an extension of variant 2) with modifications as proposed in variant 3).

The aforementioned four variants are benchmarked against a non-robust algorithm where only a single minimum latency path is assigned to each request. Thus, a request violates the QoS requirement as soon as its single possible path fails.

## 5. EXPERIMENTS AND DISCUSSION

The four above variants of the proposed algorithm plus the non-robust algorithm have been implemented and run on a network with over 3000 nodes generated using Inet 2.1 ([4]). The experiments were repeated for a Normal and Exponential probability distribution of resource failure times. Fig. 1 and Fig. 2 compare the re-

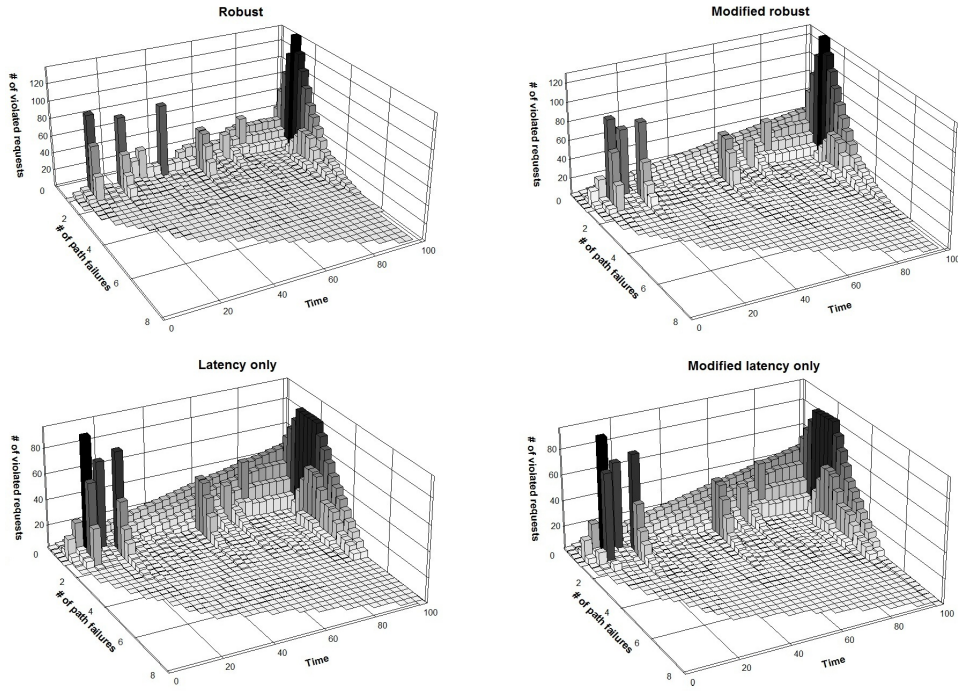


Figure 1: Comparing results of different algorithms for Normal distribution

sults of four different variants of our proposed algorithm where the probability distribution of resource failure times is Normal and Exponential respectively.

In both figures, 3-D bar charts represent number of violated requests (those which no longer conform to the latency requirement) at any instant of time and also the number of path failures that occurred for those requests before violation. Note that the network, PDF, and the set of requests used for the simulations of all four variants were the same in both cases (Normal and Exponential distributions). This makes the resultant charts for the different versions comparable. We run all the simulations for 100 time units as the figures reveal. The peaks in the bar charts happen when a key resource in the network fails causing many requests to be violated. But those requests with violation time = 100 (bars on the 100th unit of time) are actually those requests that have never been violated.

As we mentioned in Section III an algorithm that (1) postpones the requests violations and (2) make them have a smaller number of path failures (rerouting) is of interest. Our proposed algorithm is multidimensional in this sense. We can see in both Fig. 1 and Fig. 2 that robust variants of the algorithm gives better results than latency only variants. This was completely expected because in the robust variants those paths with the least probability to fail are assigned to requests while in latency only versions this probability is not taken into account at all. Therefore, when robust algorithm is used more requests remain satisfied for longer times with less number of path failures. This in turn leads to a more robust system according to our definition of robustness in Section III.

It can be seen in Fig. 1 and 2 that in all the variants the majority of requests are not violated till the end of the simulation. This is in complete contrast with the non-robust version in both probability distribution cases where the majority of the requests are violated by the first half of the simulation time and there is no conforming requests left by the end of the simulation time. See Fig. 3 and 4 for the results of simulation of non-robust path finding algorithm

in the same network for Normal and Exponential distributions respectively. Note that the PDF used for resource failure times of the simulations of Fig. 3 (Fig. 4) is the same as the PDF used for simulations of Fig. 1 (Fig. 2).

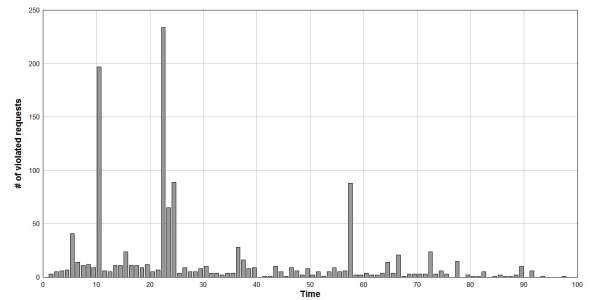


Figure 3: Results of non-robust algorithm for Normal distribution

Thus, the important result is that having a set of multiple alternative paths for each request instead of one significantly increases the robustness of the system. Moreover, the robust variant of our proposed algorithm leads to a more robust system since it postpones request violations and decreases number of path failures (path switches) for each request.

In our simulation results, the modified robust and modified latency only variants give non-significant better results than robust and latency only variants, respectively. The difference may be more significant for networks with larger number of requests that share source and destination.

## 6. CONCLUSION

In this paper we proposed a robust greedy algorithm for path assigning in distributed networks. This algorithm always stores a

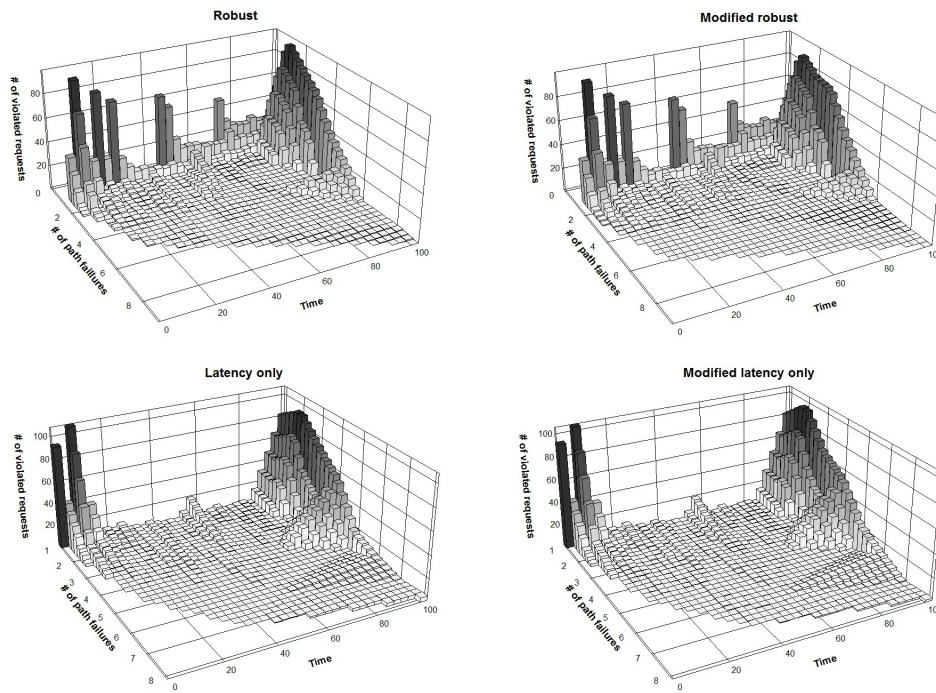


Figure 2: Comparing results of different algorithms for Exponential distribution

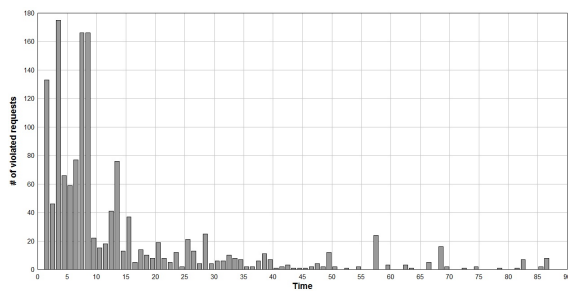


Figure 4: Results of non-robust algorithm for Exponential distribution

set of multiple alternative paths for each request in the network. Therefore, it can always replace the path if the assigned path fails due to resource failures in the system. In the proposed method, the replacing path is always the best path according to the defined cost function. We proposed a cost function that includes both QoS requirement and the probability of path failures. Thus, the paths that will most likely fail in the future time are not assigned to requests. This makes the system more robust since the system will be more tolerant against resource failures in the system. Our method makes use of the probability density functions of the resource failure times. The proposed algorithm can be implemented in both distributed or centralized manner based on applications and needs. The experimental results are very promising since by using this algorithm far more requests remain satisfied for a longer time with a smaller number of path switches.

## 7. REFERENCES

- [1] S. Ali, A. A. Maciejewski, H. J. Siegel, and J.-K. Kim. Measuring the robustness of a resource allocation. *IEEE*

*Transactions on Parallel and Distributed Systems*, 15:630–641, 2004.

- [2] S. Androutsellis-Theotokis and D. Spinellis. A survey of peer-to-peer content distribution technologies. *ACM Comput. Surv.*, 36(4):335–371, 2004.
- [3] D. Eppstein. Finding the k shortest paths. *SIAM J. Comput.*, 28(2):652–673, 1999.
- [4] C. Jin, Q. Chen, and S. Jamin. Inet: Internet topology generator, 2000.
- [5] S. U. Khan, A. A. Maciejewski, and H. J. Siegel. Robust cdn replica placement techniques. In *IPDPS '09: Proceedings of the 2009 IEEE International Symposium on Parallel & Distributed Processing*, pages 1–8, Washington, DC, USA, 2009. IEEE Computer Society.
- [6] L. L. Peterson and B. S. Davie. *Computer Networks: A Systems Approach, 3rd Edition*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2003.
- [7] D. K. Pradhan, editor. *Fault-tolerant computer system design*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1996.
- [8] J. Risson and T. Moors. Survey of research towards robust peer-to-peer networks: search methods. *Comput. Netw.*, 50(17):3485–3521, 2006.
- [9] J. Smith, L. D. Briceño, A. A. Maciejewski, H. J. Siegel, T. Renner, V. Shestak, J. Ladd, A. Sutton, D. Janovy, S. Govindasamy, A. Alqudah, R. Dewri, and P. Prakash. Measuring the robustness of resource allocations in a stochastic dynamic environment. In *In Proceedings of the 21st International Parallel and Distributed Processing Symposium (IPDPS 2007)*, 2007.
- [10] L. Wasserman. *All of Statistics: A Concise Course in Statistical Inference*. Springer Science+Business Media, New York, NY, USA, 2005.

```

for  $q \in Q$  do
   $AP_q = \text{Find\_Alt\_Paths}(s, d, \text{Net}, \text{Max})$ ;
   $\text{Path\_Fails}_q \leftarrow 0$ ;
  if  $AP_q = \emptyset$  then
     $vt_q \leftarrow 0$ ;
  else
     $vt_q \leftarrow \infty$ ;
    for  $p_i \in AP_q$  do
       $L(p_i, t) \leftarrow \sum_{j=1}^{|p_i|-1} l_{j,j+1}$ ;
    end
     $\text{min}L_q \leftarrow \min L(p_i, t)$ ;
  end
end
 $t \leftarrow 1$ ;
while  $t < \text{Simulation\_Time}$  do
  if  $t > 1$  then
    for  $q \in Q, vt_q = \infty$  do
      for  $p_i \in AP_q$  do
        for  $r_j \in p_i$  do
          if  $fl(r_j, t-1) = 0$  and
             $fl(r_j, t) = 1$  then
             $flp(p_i, t) \leftarrow 1$ ;
            if  $\text{Current\_Path}_q = p_i$  then
               $\text{Recompute}_q \leftarrow 1$ ;
               $\text{Path\_Fails}_q ++$ ;
              break;
            end
          end
        end
      end
    end
  end
  if  $t = 1$  or  $\text{Recompute}_q = 1$  then
    for  $q \in Q, vt_q = \infty$  do
      for  $p_i \in AP_q$  do
        if  $flp(p_i, t) = 0$  then
           $L(p_i, t) \leftarrow \sum_{j=1}^{|p_i|-1} l_{j,j+1}$ ;
           $\text{Pr}(T_{p_i} > t) \leftarrow$ 
             $1 - \prod_{j=1}^{|p_i|} \text{Pr}(T_{r_j} < t)$ ;
           $\text{cost}(p_i, t) \leftarrow w \times L(p_i, t) +$ 
             $(1-w) \times \text{Pr}(T_{p_i} > t)$ ;
        end
      end
       $\text{Min}P \leftarrow \text{argmin}_{p_i}(AP_q)$ ;
      if  $L(\text{Min}P, t) < \text{min}L_q \times (1 + TL)$  then
         $\text{Current\_Path}_q \leftarrow \text{Min}P$ ;
      else
         $vt_q \leftarrow t$ ;
      end
    end
  end
   $t \leftarrow t + 1$ ;
end

```

**Algorithm 1:** Robust Greedy Path Finding Algorithm