REMRR: A Reliability Evaluation Method for Dynamic Edge Computing Network under Time Constraints

Liang Chen, Jianpeng Qi, Xiao Su, Rui Wang

Abstract—While the concept of Artificial Intelligent Internet of Things (AIoT) is booming, computation and/or communication-intensive tasks accompanied by several sub-tasks are slowly moving from centralized deployment to edge-side deployment. The idea of edge computing also makes intelligent services sink locally. But in actual scenarios like dynamic edge computing networks (DECN), due to fluctuations in available computing resources of intermediate servers and changes in bandwidth during data transmission, service reliability becomes difficult to guarantee. Coupled with changes in the amount of data in a service, the above three problems all make the existing reliability evaluation methods no longer accurate. To study the effect of distributed service deployment strategies under such a background, this paper proposes a reliability evaluation method (REMRR) based on lower boundary rule under time constraint to study the degree of the rationality of a service deployment plan combined with DECN. In this scenario, time delay is the main concern which would be affected by three quantitative factors: data packet storing and sending time, data transmission time and the calculation time of executing sub-tasks on the node devices. While the last two are in dynamic scenarios. In actual calculation, based on the idea of the minimal paths, the solution set would be found that can meet requirements in the current deployment. Then the reliability of the service supported by the solution sets would be found out based on the principle of inclusions-exclusion combined with the distribution of available data transmission bandwidth and the distribution of node available computing resources. Besides an illustrative example was provided, to verify the calculated reliability of the designed service deployment plan, the NS3 is utilized along with Google cluster data set for simulation. The results gained from NS3 proved the accuracy and applicability of the proposed reliability evaluation method.

Index Terms—Reliability, Edge Computing, Hierarchical Task, Distributed Services, Dynamic Networks.

I. INTRODUCTION

WHEN a large number of devices compositing the Internet of Things entered people’s lives in various scenarios including smart cities, smart industries and smart homes, the era of the Internet of Everything comes. Gartner said that 75% of enterprise data would be generated on the edge by 2022 [1], and the IDC believes that 41.6 billion edge devices would be interconnected by 2025, with the data volume close to 80ZB [2]. While the new concepts of Industrial Internet of Things, Urban Internet of Things, Home Internet of Things and so on, bring opportunities [3], there are also countless challenges. When computation and/or communication-intensive tasks get gradually closer to the edge [4], one of the challenges is that the data processing capability of the Internet of Things as a perception network has never been comparable to the data center. Then cloud computing just makes up for this shortcoming. This computing paradigm combines the Internet of Things and cloud computing with the Internet. The devices on the edge would collect data and send them to the cloud for computation and storage. But the cloud computing is not that perfect.

Although centralized cloud services like Amazon Web Service (AWS) helped companies effectively reduce IT investment costs and maintenance costs [5], their shortcomings are also obvious. Limited network transmission performance, high energy consumption and maintenance costs, difficulty for privacy guarantee, and poor real-time quality, have yet to be resolved. Edge computing emerged in this situation. Edge topics have appear in related research fields since 2015 [6], such as Mobile Edge Computing (MEC) [7], micro-cloud and fog computing [8]. Compared with cloud computing owning excellent computing capabilities, edge computing uses nearby services to provide faster service response, simultaneously realizes effective use of local resources showed in Fig[1]. Edge devices are generally considered to own insufficient computing capabilities [9], which is in line with reality that edge devices would not execute difficult computation task. The general way works that setting edge servers between cloud computing and edge devices to complete the computing tasks of the data transmitted from the devices instead of entirely relying on the cloud computing center [10]. Through this mode, only part of the intermediate data or the final calculation results need to be transmitted to the cloud, which greatly reduces the cost of communicating with the cloud. It is unreasonable for the camera at the gate of a community to recognize someone passing by without auxiliary equipment. This task requires an additional VGG-Net [11] or any model designed for face recognition. Obviously, the computing capability does not support it to complete this task. But it is possible to perform preliminary processing on the captured pictures and videos. A complete VGG16 model with 138M parameters will take up close to 500Mb of disk space, but the pre-process operation will not require so many parameters. A service completed in the form of multiple sub-services, could improve
the application’s scalability, portability, and availability. [12]

Edge computing, this computing paradigm has present outstanding performance and practicality, while many researchers have published relevant work [13] [14], there exists still two issues. Without cloud computing, the network transmission latency brought by high-volume data transmission is reduced, but the realization of most computing work outside the cloud center would cause the computational latency. The edge servers and devices cannot rival with cloud on computing resources, so the computational delay becomes an issue that must be considered when designing and deploying distributed services. The next is that the stability of cloud computing mode disappeared. Due to redundant configuration, cloud computing guarantees that the requests from the devices can always be completed attributing to sufficient computing resources. But when combining distributed services with edge computing scenarios, it is necessary to consider the issue of computing resources. Specially each edge server is generally responsible for more than one service, which means that its available resources are changeable with time [15]. When a new task arrives, the amount of currently available computing resources of the server is unknown, making it difficult to estimate the execution time. And the available network bandwidth is always changing, which also make the entire time cost fluctuating. The execution time of the task becomes difficult to estimate due to these uncontrollable factors, and the originally designed service is likely to become an unreliable service, making it difficult to realize a major advantage of edge computing. For example, short-term network congestion or insufficient machine performance will lead to a decrease in quality of service (QoS).

In addition, the change in the amount of data in transmission process of distributed tasks, which means the amount of data that needs to be transmitted at one service is different, also makes the existing reliability evaluation methods inapplicable [16]. To solve the dilemma, this paper models the distributed tasks and deployment plan in the above scenario based on the concept of reliability, built a DECN scenario and proposed a method to calculate the probability that a distributed task can be completed, also named reliability. Aiming at the unreliability of bandwidth, bandwidth distribution sampling was used to represent the change in available bandwidth. In view of the changes in computing resources of computing nodes, the probability distribution of available computing resource is summarized based on historical data modeling. Combining the characteristics of distributed tasks, find the solution set that meets the requirements to calculate the reliability of completion. The proposed method can not only tell the credibility of the current deployment plan to complete the assigned tasks, but also give a theoretical support in the early stage to help service providers deploy a more perfect service.

The detailed contributions of this paper are as follows:

- This study provides a brand new reliability evaluation method aiming at popular edge computing scenarios with complicated dynamic edge computing networks. To our best known, it is the first try that network bandwidth and computing resources distribution are jointly considered to evaluate the reliability of edge computing distributed services under the complex dynamic scenarios with changeable data size.
- Different from traditional network reliability evaluation, dynamic changes in the amount of data on the deployment plan and the computing performance of nodes are considered and implemented.
- This study achieved static separation of composite node and transmission node considering the actual scenario, which means that the proposed DECN can fit nodes with different functions.

In the rest of this paper, section 2 introduced related works about reliability in network transmission and edge computing. Theoretical support and basic experimental procedures were presented in section 3, along with a reliability algorithm. An illustrative example and simulation were given in Section 4 to bring more convincing details. Section 5 concluded the whole study and gave some directions in future research, focusing on the content that has not been explored in depth in this paper.

II. RELATED WORKS

The traditional reliability often appeared in related manufacturing and supply chain researches and applications [17]. While the focus on reliability is shifted to computer-related research, it is found that many researchers in network transmission use it [18], called “networks reliability analysis”. When combining edge computing with deep learning services like Fig. 2 the reliability index of the system also needs to be considered. Then the reliability discussed in this article is different also, which is used to express success probability of distributed services in edge computing.

A. Networks Reliability

Researchers have done a lot of work on the reliability of network transmission [16] [19]. Paper [20] uses the concept of elasticity, which combines network capacity and reliability that affect service quality. The proposed calculation method
more comprehensively considers the influencing factors of service reliability, uses rerouting to solve failures and splitting data to solve network congestion. When Lin is modeling the cloud and fog system, the proposed stochastic-flow cloud–fog computing network model effectively evaluates the probability of successful data transmission in the IoT scenario [21]. Since their model ignored the computing process of nodes and the dynamic computing resources of nodes, their method cannot be directly applied to edge computing scenarios with distributed tasks. Ruen et al. used an approximate Lagrangian relaxation algorithm to solve the problem of generating the maximum reliability of a computer network under a delay constraint [22]. In paper [23], Lin et al. creatively applied deep neural network to the calculation of network reliability and gained promising performance.

However in addition to thinking of the minimum boundary required for data transmission, it is also necessary to consider the minimum boundary required for task when computing resources are dynamically changing. Combined with minimal cut vectors, Ramirez-Marquez proposed the Monte Carlo simulation method to estimate the reliability of multi-state networks, and achieved better numerical results than enumeration method, inclusion and exclusion principles [24]. It provides an effective way to solve the problem from other directions. However, in certain aspects this simulation method still lacks sufficient applicability, taking into account the calculations that may exist in the network. It is similar from the perspective of providing services and completing tasks. But in the scenario of edge computing, in addition to data transmission, computing is also an important point which need to be considered. So measuring the reliability of a service requires comprehensive consideration, especially when computing resources and network bandwidth fluctuate.

B. Edge Computing Reliability

When Dong et al. designed the optimal offloading strategy and allocation algorithm in the resource-constrained mobile edge computing system [25], they considered the reliability of the task which was based on the inert shadow scheme. The way that relying on redundancy deployment to improve reliability is commonly used, like the disaster tolerance performance of the server services. Li et al. are studying task allocation of the edge computing system composed of ES and EU [26]. Their model selectively offload part of the tasks to the local devices to complete the calculation, and the remaining part will be calculated by ES with regular model retraining and parameter return. However, it is assumed that the local model will execute at a fixed CPU frequency, and the dynamic change of the available resources of the devices and edge server are not taken into account. The delay and energy consumption results obtained in this way are lacking of some important consideration. Kang and others have questioned the current pure cloud computing paradigm in paper [27]. By comparing the pure cloud and cloud-edge collaborative computing paradigm, they explained the importance of sinking part of the computing to the edge. And proposed a framework, which can adaptively cut the model at a reasonable division point and send them to different computing units with the goal of lowest delay and energy consumption. There is no doubt that the work of their thesis is very representative. But the work does not take the dynamic factors of resources into consideration, which means that it achieves a limited lowest delay only in the reasonable division of the analysis task in a static way. Compared with the above work, in a typical edge computing scenario, REMR not only combines the network transmission delay and the calculation delay of different nodes, but also realizes dynamic modeling according to the fluctuation of network bandwidth and the availability of nodes resources. On this basis, the proposed reliability evaluation method can effectively offer the feasibility of the current deployment plan, and it has been verified on the simulation platform NS3 that the calculation results are almost the same as the simulation results.

III. METHODS AND MATERIALS

In DECN, a distributed service or application is deployed on the device side and edge servers that can provide computing resources according to the corresponding strategy like Fig. 3. To improve the reliability of global task completion, there usually exists several deployment plans in one DECN situation. After the service deployment finished according to the hierarchical structure, the model assigned to the devices will process the input data, and the intermediate data obtained will be transmitted to the edge server through the line for the next step of the whole service. For calculations, depending on the distribution of models and the amount of calculations, different deployments may have different numbers of intermediate servers. In the end, the results of all deployments will be transmitted to the cloud, which mainly plays a role of monitoring and recording. There may also exist some servers for transmission only.

Similar to traditional network transmission, each transmission line in one deployment has a possible capacity and corresponding probability distribution. The difference is that the traditional network transmission model is no longer applicable here because the middle servers need to complete a large number of computing tasks. Therefore, for intermediate service providers, additional considerations have been taken. Combining the characteristics of edge servers and considering the services that may have been running, the computational performance is modeled in a dynamic mode, and expressed
notations could be explained as:

of its sub-tasks. All nodes designed for the deployment plan according to the geographical order and hierarchical structure to different servers and devices on one deployment plan, or computing task would be reasonably divided and assigned. DECN can be established. A hierarchical distributed service Once the above structure parameters are determined, a typical workflow of the entire model can be summarized and displayed in the Fig. 4.

A. DECN Scenario

A DECN is composed of several deployment plans, which could be represented by $L = \{S, I, X, Y, P\}$, the detailed notations could be explained as:

- $S$: the set of nodes in a deployment plan, where $s_i$ means the $i$-th node in the deployment plan;
- $I$: the set of branches in a deployment plan, where $I_i$ means the $i$-th branch in the deployment plan;
- $X$: the capacity of branches, where $X_i$ is the maximal capacity of $b_i$;
- $Y$: the performance of nodes in a deployment plan, where $Y_i$ is the best performance that the $s_i$ can provide, and $y_i$ is the actual performance that the $s_i$ would provide;
- $P$: the changes in data size after computing nodes, where $p_i$ means the ratio of data output to input through the node

Once the above structure parameters are determined, a typical DECN can be established. A hierarchical distributed service or computing task would be reasonably divided and assigned to different servers and devices on one deployment plan, according to the geographical order and hierarchical structure of its sub-tasks. All nodes designed for the deployment plan will jointly complete the task. Each node that exists both input and output need extra time to finish receiving and sending work of data packets, named lead time, which would be attached to branches. For the convenience of calculation, no more details are given here. In fact, the delay of transmission is composed of many factors, but this study has simplified it here [28].

Besides the lead time, the transmission for intermediate data of a model is more essential for the whole mission considering time constraints. For one deployment plan, the time required for transmission could be represented by the sum of the time on each branch. Equation (1) provides the answer of lead-time and transmission time. Where $x_i$ represents the available bandwidth on the branch $i$ and $C$ is the initial input data. $L_i$ is the leadtime on the branch $i$. Rounding up is for calculation convenience.

$$t = \sum_{i=1}^{n} L_i + \sum_{i=1}^{n} \left[ \frac{C}{x_i} \right]$$

(1)

The calculation of $t$ so far is similar to traditional network transmission reliability research [19], because only the time it takes for data to be transmitted on the deployment plan has been considered. In DECN, once the data arrives at a node as input, the node would be activated to execute local task. At present, it is considered that the running time of the execution calculation of the copy running on each node is proportional to the size of the input data, and inversely proportional to the available computing resource. In addition, due to the unknowability of the tasks amount already running on the node, the performance of the node is considered in a dynamic situation, which is difficult to perceive available computing resource of the node when the computing task is coming. So changeable $y_i$ like third term in equation [2] is used to represent the required computing resource.

$$t = \sum_{i=1}^{n} L_i + \sum_{i=1}^{n} \left[ \frac{C}{x_i} \right] + \sum_{i=1}^{n} \frac{C}{y_i}$$

(2)

Each $x_i$ and $y_i$ is independent, which means they can take any value among their range. However, the size of the output data after the computing node would not be exactly the same as the input data, such as dimension reduction and convolution operations. The ratio of output to input of the calculation executed on the node would be saved as a parameter and used as data change factors. Combining the data size change of the node and an initial demand $C_0$, the time finally completing the whole task for the deployment plan can be expressed as:

$$t = \sum_{i=1}^{n} L_i + \sum_{i=1}^{n} \left[ \frac{C_0 \prod_{k=0}^{i-1} S_k}{x_i} \right] + \sum_{i=0}^{n} \frac{C_0 \prod_{k=0}^{i-1} S_k}{y_i}$$

(3)

Under the constraint of time $T$, if the entire computing task which takes $C_0$ as input can execute successfully, the inequality $t \leq T$ need be satisfied. The above formula is OK under normal situation, but when the rate appears 0, attention need to be paid to it during the experiment. Because the node could have no computing task, the data must be transmitted.
B. MSV and RSDP Algorithms

For any specific deployment plan in DECN, the load time is fixed first. Assuming that the solution \( (x_i, y_i) = \{X_i(x_0, x_1, x_2, ..., x_n), Y_i(y_0, y_1, y_2, ..., y_n)\} \) satisfies \( t \leq T \). Time variable \( t \) is mostly determined by the transmission time and calculation time, affected by the available bandwidth of all deployment plans and computing resource of all nodes. This solution \( (x_i, y_i) \) is one of the feasible solutions for the reliability of the deployment plan, which could be answered by follow equation (4):

\[
R_i = \Pr\{X, Y | t \leq T\} = \Pr\left(\prod_{i=1}^{n} x_i \prod_{i=1}^{n} y_i\right)
\]

Due to the diversity of deployment plan bandwidth distribution and node computing resources distribution, there usually exists many solutions similar to the solutions \( \{z\} \). Assuming that there are totally \( z \) solutions in the set of \( \mathcal{O}_z \) where stores every solution for the inequality \( t \leq T \), the global reliability of a single deployment plan can be defined as:

\[
R_{a} = \Pr\left(\bigcup_{i=1}^{z} R_{i}\right)
\]

However, in this case the calculation method for reliability of \( \mathcal{O}_z \) exists a lot of repeated calculations, bring a low efficiency. So, the Minimal Status Vectors (MSV) algorithm is used to eliminate all redundant calculation in this paper. Vector operations are depicted as the following principles:

- **Rule 1.** \( Y \leq X_i(y_1, y_2, ..., y_n) \leq (x_1, x_2, ..., x_n) \); \( y_i \leq x_i \) for each \( i = 1, 2, ..., n \).
- **Rule 2.** \( Y < X_i(y_1, y_2, ..., y_n) < (x_1, x_2, ..., x_n) \); \( Y \leq y_i < x_i \) for at least one \( i \).

It would be removed for such kind of \( X \) in \( \mathcal{O}_z \), that the \( X \geq Y \) (except for \( X \)) and \( \forall X, Y \in \mathcal{O}_z \). Then, the remaining solutions are more representative, also named MSVs. Here is a proof that the reliability of distributed tasks on a certain deployment plan can be represented by the probability calculated by all MSVs on the plan.

**Proof.** \( \forall P_i, P_j \in \mathcal{O}_z \) and \( P_i \geq P_j \), obeying above principles rule.

\[
P_j = X_i(x_0, x_1, x_2, ..., x_n)Y_j(y_0, y_1, y_2, ..., y_n);
P_j = X_j(x_0, x_1, x_2, ..., x_n)Y_j(y_0, y_1, y_2, ..., y_n);
Pr(P_i) = \prod_{k=0}^{n} p(x \geq x_k)p(y \geq y_k)
\]

because: \( P_i \geq P_j \), assuming that \( X_i(x_0) \geq X_j(x_0) \) and \( X_j(x_1, x_2, ..., x_n) = X_j(x_1, x_2, ..., x_n) \), \( Y_i = Y_j \) therefore:

\[
Pr(P_j) = Pr(P_i) + Pr\{X(x_0) = X_j(x_0)\}
\]

therefore:

\[
Pr(P_j) = Pr(P_i) + p(x = X_i(x_0)) \times \prod_{k=1}^{n} p(x \geq x_k)p(y \geq y_k)
\]

so \( Pr\{P_i, P_j\} \) exists duplicate calculations without properly process on the solution set. So MSVs method is used to remove the solution set that will be recalculated.

If there exists \( z \) MSVs in total in new answer set called \( \mathcal{O}_{MSV} \), the reliability of deployment plan change into (6):

\[
R = \Pr\left(\bigcup_{j=1}^{z} R_j\right)
\]

The calculation of equation (6) is a joint probability problem of multiple probabilities. To avoid repeated calculations, the principle of inclusion and exclusion is a feasible idea for solving this kind of problem. The RSDP algorithm mentioned in paper (29) provides an efficient way using recursive thinking. In the case where all solutions \( R_i = Pr\{\} \) can be found, the reliability of \( \mathcal{O}_z \) which contains a total of \( z \) solutions could be answered by \( R = RSDP(\mathcal{O}_z) \).

C. The REMR Algorithm

The REMR means a reliability evaluation method with MSV and RSDP algorithm. Once the DECN is established, the input data and time constraint identified, the service reliability of a single and global deployment can be obtained by the Algorithm (1). The line 25 gave the result of a single deployment plan, while the global reliability could be calculated by line 27.

The whole algorithm is mainly composed of several parts. The lines 4 to 10 is to find a solution set that meets the requirements, and lines 11 to 24 is to find all MSVs in this solution set. Then the RSDP method combined with the MSVs is used to solve the reliability of the corresponding deployment plan in line 25. Finally, the reliability of the entire scheme is calculated by joint probability formulas.

Assuming that there are a total of \( m \) initial vector in the solution set, the complexity of the algorithm for finding MSVs is about \( O(m^2) \). If there are a total of \( n \) MSVs for one deployment plan, the complexity of the RSDP method would be \( O(n^3) \). The last step of whole algorithm, the joint probability only have \( O(k) \) complexity if there exists \( k \) plans in a DECN.

IV. EXPERIMENTS IN DECN

A DECN designed like Fig[5] is used to explain the entire solution procedure where the deployment plan of the service have been shown. In this example, \( S_0 \) node means a edge device where the task data from and the cloud is the terminal for the deployment plan. The compound nodes will play a role in transmission task and computing task simultaneously, while the transport nodes will only account for transmission task.

The TABLE I depicts the probability distribution of available bandwidth on each branch and the TABLE II shows the probability distribution of available compute resources on each server and each device, while each capacity of

![Fig. 5: An illustrative DECN example](image-url)
Algorithm 1 Multi-Deployment Plans Reliability

Input:
- Initial \( DECN : S, I, X, Y, P; \)
- Input \( \text{input\_data}, T; \)
- Solve: \( \text{R}_{\text{MSVs}}, \text{RSDP}(\text{O}_{\text{MSVs}}); \)

Output:
- \( R: \) reliability of whole \( DECN; \)
- \( t = \sum \text{leadtime} + \text{transitime} + \text{computingtime}; \)
- Suppose there are \( K \) deployment plans;
- for \( k = 1 : K \) do
- Create a traversal table \( O_{\text{init}} \) with \( N \) vectors for \( X, Y; \)
- Create two empty collections \( O_z, O_{\text{MSVs}} \)
- for \( n = 1 : N \) in \( O_{\text{init}} \) do
- if \( t_n \leq T \) then
- \( O_z = O_z \cup \text{Vector} : n \)
- end if
- end for
- Suppose there are \( Z \) solutions in \( O_z, L \) MSVs in \( O_{\text{MSVs}}; \)
- for \( z = 1 : Z \) in \( O_z \) do
- for \( l = 1 : L \) in \( O_{\text{MSVs}} \) do
- if \( MSV : l \geq Solution : z \) then
- Remove \( MSV : l \) from \( O_{\text{MSVs}}; \)
- end if
- end for
- end for
- for \( l = 1 : L \) in \( O_{\text{MSVs}} \) do
- if \( Solution : z \geq MSV : l \) then
- Break and next \( z; \)
- end if
- end for
- \( O_{\text{MSVs}} = O_{\text{MSVs}} \cup \text{solution} : z \)
- end for

\( R_k = \text{RSDP}(O_{\text{MSVs}}); \)

\( R = \bigcup_{k=1}^{K} R_k; \)

bandwidth and compute resources is set to be a multiple of integer for simplicity. The bandwidth distribution based on historical data is derived from paper [30], used to describe the probability of each branch’s available bandwidth. Then none of papers in network reliability study the computing resources. The distribution of computing resources TABLE II in this paper comes from the analysis of the Alibaba servers dataset. Alibaba provides a huge dataset, which includes the log records of CPU usage about server clusters [3]. This study used the data to simulate the CPU usage frequency of the edge server and converted it into available CPU resources, as shown in the Fig. 6. The \( \text{output/input} \) in the TABLE II represented the ratio of the output data calculated by the server to the size of the input data received by the server, which is used to calculate the change degree in the deployment plan.

A. An Illustrative Example

Assuming that \( C_0 = 15 \) units of data would be the input of the three deployment plans with time constraint \( T = 25s, \) the reliability of the task could be present by

\[
R = \{R_a, R_b, R_c\}, \text{where} \ a = \{s_0, s_{12}, s_{13}, s_{11}, \text{cloud}\}, \ b = \{s_0, s_{21}, s_{22}, \text{cloud}\}, \ c = \{s_0, s_{31}, s_{32}, \text{cloud}\}. \text{To explain this indicator} \ \text{output/input} \ \text{with more detail, TABLE III is draw where the DP represented different deployment lines. They have one initial edge device in common. Therefore, the initial input} \ C_0 \ \text{is all 15 units. Taking DP1 as an example, The indicator of the device is 0.8, so the amount of data send to the first node} \ C_1 \ \text{of DP1 becomes 12. Here the indicator is 1.2, so the amount of data} \ C_2 \ \text{becomes 14.4. Since the} \ s_{12} \ \text{in DP1 is a transmission node without computing, the output} \ C_3 \ \text{is totally the same as input, also 14.4, the node} \ s_{13} \ \text{would finish the final computing task and send a result to the cloud. Usually the output here would be very small, so the indicator is set to be 0.01, to present a tiny computing task. This section would firstly take} \ R_a \ \text{as the example to describe}
\]

| branch | lead time | capacity |
|--------|-----------|----------|
|        |           | 0  | 1  | 2  | 3  | 4  | 5  |
| s_1    | 2         | 0.04| 0.01| 0  | 0.02| 0  | 0.93|
| s_2    | 1         | 0.03| 0.01| 0.02| 0.94| 0  | 0  |
| s_3    | 1         | 0.02| 0.01| 0  | 0.04| 0.93| 0  |
| s_4    | 1         | 0.03| 0.01| 0  | 0.06| 0  | 0.9 |
| s_5    | 2         | 0.02| 0.01| 0.01| 0  | 0.96| 0  |
| s_6    | 2         | 0.04| 0.01| 0.95| 0  | 0  | 0  |
| s_7    | 1         | 0.03| 0.01| 0  | 0.02| 0.94| 0  |
| s_8    | 1         | 0.03| 0.01| 0.95| 0  | 0  | 0  |
| s_9    | 2         | 0.03| 0.01| 0.01| 0.93| 0  | 0  |
| s_10   | 1         | 0.02| 0.01| 0.02| 0.95| 0  | 0  |

Fig. 6: CPU usage distribution with time in Alibaba servers cluster.

TABLE II: Available Nodes Capacity Probability

| branches | output/input | capacity |
|----------|--------------|----------|
|          | 0            | 1        | 2        | 3        | 4        | 5        | 6        |
| s_0      | 0.8          | 0.0      | 0.01     | 0.09     | 0.26     | 0.37     | 0.20     | 0.07     |
| s_11     | 1.2          | 0.0      | 0.01     | 0.09     | 0.26     | 0.37     | 0.20     | 0.07     |
| s_12     | 1.0          | 0.0      | 0.01     | 0.09     | 0.26     | 0.37     | 0.20     | 0.07     |
| s_13     | nan          | 0.0      | 0.01     | 0.09     | 0.26     | 0.37     | 0.20     | 0.07     |
| s_21     | 1.2          | 0.0      | 0.01     | 0.09     | 0.26     | 0.37     | 0.20     | 0.07     |
| s_22     | nan          | 0.0      | 0.01     | 0.09     | 0.26     | 0.37     | 0.20     | 0.07     |
| s_31     | 1.2          | 0.0      | 0.01     | 0.09     | 0.26     | 0.37     | 0.20     | 0.07     |
| s_32     | nan          | 0.0      | 0.01     | 0.09     | 0.26     | 0.37     | 0.20     | 0.07     |
specific calculation process. According to the equation (6), all MSVs need to be found. Combining the equation (3), the given initial input data and time threshold $T$, implicit enumeration method is utilized in the above-mentioned DECN to gain all solutions for the inequality $t = 4 + \sum_{i=1}^{4} \left\lceil \frac{C_0 \prod_{k=0}^{i-1} S_k}{x_i} \right\rceil + \sum_{i=0}^{4} \frac{C_0 \prod_{k=0}^{i} S_k}{y_i} \leq 25s$ without thinking of repetition. The changes in the amount of input data after node processing are shown in the table III. $C_0$ represents the initial input, and the amount of data would always change after going through different nodes of the deployment plan, similarly $C_i$ represents the amount of data output after the $i$-th node, and the rest is the same. Implicit enumeration refers to the removal of attempts to impossible answers on the basis of enumeration. First 2460 solutions were found in solving the inequality. After filtering the solution set, 14 MSVs remained, while part of them were listed in the table IV.

### TABLE III: Changes of Data Size in Deployment Plans

|   | $C_0$ | $C_1$ | $C_2$ | $C_3$ | $C_4$ |
|---|---|---|---|---|---|
| DP1 | 15 | 12 | 14.4 | 14.4 | 0.01 |
| DP2 | 15 | 12 | 14.4 | 0.01 | 0.01 |
| DP3 | 15 | 12 | 14.4 | 0.01 | 0.01 |

### TABLE IV: The MSVs for A Deployment Plan

| No | $X(x_1, x_2, x_3, x_4)$ | $Y(y_1, y_2, y_3)$ |
|---|---|---|
| 1 | (2, 3, 4, 1) | (5, 1, 6) |
| 2 | (2, 3, 4, 1) | (6, 1, 5) |
| 3 | (3, 3, 3, 1) | (4, 1, 5) |
| 4 | (3, 3, 3, 1) | (5, 1, 4) |
| 5 | (3, 3, 4, 1) | (4, 1, 4) |
| ... | ... | ... |
| 14 | (4, 3, 4, 1) | (4, 1, 3) |

Then the reliability of deployment plan $a$ can be obtained by the improved RSDP algorithm. The reliability $R_a = 0.82344$. The reliability of $R_b = 0.87679$ and $R_c = 0.90955$ can also be get by same way above, while they respectively own 7 and 17 MSVs. Finally the global reliability $R = R_a \cup R_b \cup R_c$. Considering that the three reliability are independent, so

$$R = R_a + R_b \ast (1 - R_a) + R_c \ast (1 - R_a) \ast (1 - R_b) \quad (7)$$

The obtained result is 0.99653, which presents the probability of the DECN finishing the task perfectly. More similar experiments based on the same DECN were carried out and the results were record at the TABLE V below.

### TABLE V: Numerical Experiments

|   | C=14 | C=15 | C=16 |
|---|---|---|---|
| T=20 | 0.94930 | 0.87567 | 0.75312 |
| T=21 | 0.97163 | 0.94656 | 0.82634 |
| T=22 | 0.98670 | 0.97449 | 0.91367 |
| T=24 | 0.99606 | 0.99438 | 0.98362 |
| T=25 | 0.99729 | 0.99653 | 0.99207 |

B. Experiments on Google Cluster Trace Dataset

Fig. 7: the Distribution of subtasks number in part Google Cluster Trace Dataset

The google cluster data trace represents 29 day’s worth of Borg cell information from May 2011, on a cluster of about 12.5k machines. According to the analysis of the dataset in this study, it was found that most of the services were decomposed into several sub-services. The distribution of the sub-services number can be statistically obtained as shown in Fig.7 by counting the frequency of event type 0 in each job. Obviously most of the services are decomposed into no more than 10 sub-services. This experiment selected a service with 3 sub tasks in the data-set for evaluation. These three sub-services run on different machines based on the time sequence, so this study monitored the available resources on these three machines based on three days of data. Fig.8 showed the situation of one machine of the three, and it was translated into a probability distribution shown in Fig.9. Since the cluster data-set have no bandwidth distribution data, the bandwidth data-set used in last experiment was adopted again. The reliability is calculated to be 0.98657, while the input data size is 14 and time constraint is 20. Although possibly affected
by the bandwidth, the success rate of its task completion is still very high which could be explained by the high spare rate of machines.

V. CONCLUSIONS

Changeable factors including computing resources, transmission bandwidth and intermediate data volume make the previous reliability evaluation methods less accurate. This paper provides a new reliability evaluation method by establishing a DECN scenario under time constraint. Aiming to present the dynamic scenario of edge computing with distributed services, the built model have edge devices and edge servers, owning different probability distribution to describe their respective capacity. Through the MSVs and RSDP algorithm, the reliability based on solution set could be obtained. Then an example was introduced to explain how the model and method work. At last, NS3 was used to experiment as the simulation, which is in line with the proposed method almost. With the method, it would be convenient to tell how reliable a distributed service is and how to make the service more reliable.

Some directions worthy of further research in the future are listed as follows. The first is to optimize the time calculation way when the nodes executing tasks, while the current evaluation method is not detailed enough. Then the RSDP Algorithm could be improved on time as showed in paper [32]. More importantly, hoping to provide a framework, so that the proposed reliability evaluation method could be used to segment a distributed service or a DNN model rather than manual assignments.

Fig. 9: available CPU resource distribution in part of Google Cluster Trace Data-set.
lurks and how to tame it,” Proceedings of the IEEE, vol. 107, no. 2, pp. 280–306, 2018.

[29] M. J. Zuo, Z. Tian, and H.-Z. Huang, “An efficient method for reliability evaluation of multistate networks given all minimal path vectors,” IIE transactions, vol. 39, no. 8, pp. 811–817, 2007.

[30] C.-F. Huang, Y.-K. Lin, and L. C.-L. Yeng, “Routing scheme of a multistate computer network employing a retransmission mechanism within a time threshold,” Information sciences, vol. 340, pp. 321–336, 2016.

[31] J. Guo, Z. Chang, S. Wang, H. Ding, Y. Feng, L. Mao, and Y. Bao, “Who limits the resource efficiency of my datacenter: An analysis of alibaba datacenter traces,” in 2019 IEEE/ACM 27th International Symposium on Quality of Service (IWQoS), pp. 1–10, 2019.

[32] G. Bai, M. J. Zuo, and Z. Tian, “Ordering heuristics for reliability evaluation of multistate networks,” IEEE Transactions on Reliability, vol. 64, no. 3, pp. 1015–1023, 2015.