PAPER

Progressive Forwarding Disaster Backup among Cloud Datacenters

Xiaole LI[†], **Hua WANG**[††a)], **Shanwen YI**[†††], ***Nonmembers***, ***and*** **Linbo ZHAI**[††††], ***Member***

SUMMARY The periodic disaster backup activity among geographically distributed multiple datacenters consumes huge network resources and therefore imposes a heavy burden on datacenters and transmission links. Previous work aims at least completion time, maximum utility or minimal cost, without consideration of load balance for limited network resources, likely to result in unfair distribution of backup load or significant impact on daily network services. In this paper, we propose a new progressive forwarding disaster backup strategy in the Software Defined Network scenarios to mitigate forwarding burdens on source datacenters and balance backup loads on backup datacenters and transmission links. We construct a new redundancy-aware time-expanded network model to divide time slots according to redundancy requirement, and propose role-switching method over time to utilize forwarding capability of backup datacenters. In every time slot, we leverage two-step optimization algorithm to realize capacity-constrained backup datacenter selection and fair backup load distribution. Simulations results prove that our strategy achieves good performance in load balance under the condition of guaranteeing transmission completion and backup redundancy.

key words: *progressive forwarding disaster backup, load balance, progressive forwarding, redundancy-aware time-expanded network, role-switching*

1. Introduction

Cloud datacenters are geographically distributed (geo-distributed) datacenters deployed around the world. Consisting of huge amount of data, they are widely leveraged by large enterprises such as Amazon, Google, Microsoft, to provide various services for millions of users all over the global [1], [2]. Unfortunately, cloud datacenters are facing various natural or man-made disasters [3]. Take American companies for example, the failure of cloud datacenters results in huge economic loss, costing as much as $700 billion every year by unplanned datacenter outages [4]. Therefore, to improve disaster tolerance, it is necessary to leverage periodic disaster backup among cloud datacenters. Disaster backup activity aims to distribute multiple replicas of the data newly generated over a past time period to geo-distributed datacenters. It is worth noting that disaster backup activity is not real-time but always happens in a continuous time period (e.g., from 3 a.m. to 6 a.m. every day) [5]. Compared to disaster evacuation [3], disaster backup activity has a relatively loose deadline constraint. On the other hand, due to the existence of thousands of servers in each datacenter with backup requirement (source datacenters) in large enterprises [6], disaster backup requires bulk transfers consisting of terabytes or even petabytes of data to three or more remote sites (backup datacenters) [7]. Previous work aims at least completion time, maximum utility or minimal cost, without consideration of load balance for limited network resources (e.g., forwarding capability of source datacenters, receiving capacity of backup datacenters, transmission capability of network links, etc.). The unfair distribution of backup load may result in heavy burden or severe congestion on source datacenters and some critical transmission links. Especially because the source datacenters always play dual roles as backup data sender and daily application server, this unbalanced load distribution not only impacts the normal operation of daily network applications, but also impedes the satisfaction of temporary traffic requirements [7]. Therefore, in disaster backup activity, it is of great significance to mitigate forwarding pressure of source datacenters and maintain balanced distribution of backup load.

In existing research works, the disaster backup problem among cloud datacenters mainly involves transfer scheduling, backup datacenter selection, routing searching, etc. Y. Wang et al. construct elastic time-expanded network (TEN) model to represent the time-varying network status, and apply store-and-forward mode to reduce the peak traffic load on transmission links [7]. They aim to lexicographically reduce maximum link utilization caused by multiple bulk data transfers. However, their strategy only considers flow distribution on links without optimization of backup datacenter selection and routing searching, and therefore is not suitable for disaster backup activity in practice. Some researchers leverage multicast to reduce flow completion time and bandwidth usage for inter-datacenter data replication [6], [8], [9]. Y. Wang et al. try to maximize overall utility gain for dynamically-arriving bulk transfers with different deadlines across geo-distributed datacenters [10]. They leverage store-and-forward intermediate datacenters to support fast sending from source datacenter, and propose algorithms aiming to fully utilize the available link bandwidth during every time slot. However, the works [6], [8]–[10]

Manuscript received January 26, 2019.
Manuscript revised June 22, 2019.
Manuscript publicized August 19, 2019.

[†]The author is with School of Information Science and Engineering, Linyi University, Linyi, Shandong Province, China.
[††]The author is with School of Software, Shandong University, Jinan, Shandong Province, China.
[†††]The author is with School of Computer Science and Technology, Shandong University, Jinan, Shandong Province, China.
[††††]The author is with School of Information Science and Engineering, Shandong Normal University, Jinan, Shandong Province, China.
a) E-mail: wanghua@sdu.edu.cn (Corresponding author)
DOI: 10.1587/transinf.2019EDP7030

have not considered load balance problem on backup datacenters and transmission links.

There are some recent researches specially for disaster backup scenarios. Some works focus on the goal of minimizing backup completion time in mutual disaster backup model [11], [12]. J. Yao et al. select backup datacenter with least hops and compute maximum flow routing for every application datacenter to realize rapid backup transmission among geo-distributed datacenters [11]. P. Lu et al. select backup datacenter with largest available capacity and leverage shortest routing to construct transmission path [12]. However, the one-to-one backup model cannot provide sufficient redundancy while up to three or more replicas are required in practice [7], [8]. In our earlier work, we jointly consider receiving capacity and redundancy requirement, specify bandwidth allocation proportion according to backup requirement, and improve network transmission capability with adjustment rules [13], [14]. However, none of the researches above has considered mitigating burden on source datacenters and load balance on transmission links during flow scheduling process.

Unlike the works aforementioned, some works focus on emergency backup [15]–[17]. L. Ma et al. propose an Integer Linear Program-based theoretical framework to select backup datacenters and determine transmission paths for minimal cost of backup transmission and data storage [15]. P. Lu et al. prioritize the data that will be destroyed by disasters and leverage time-constrained data migration to obtain maximum utility [16]. X. Xie et al. evaluate endangered data value and network resource cost, and employ the improved alternation direction method of multipliers to get emergency backup solution with maximum profit [17]. Especially, they leverage TEN model in favor of dynamic flow scheduling over time [16], [17]. However, emergency backup is actually pre-disaster evacuation, which aims to evacuate endangered data as much as possible in a limited period of time. Therefore, these strategies above are not suitable for regular disaster backup activity considering load balance.

As shown above, in the time-varying network, previous studies have not jointly considered mitigating data forwarding burden on source datacenters and fair load distribution in backup activity. The huge network resource consumption is likely to cause congestion on some datacenters or critical transmission links, whereas the forwarding capability of backup datacenters is not fully utilized.

In this paper, we propose a new progressive forwarding disaster backup strategy. We obtain bandwidth requirement for every backup request, construct a new redundancy-aware time-expanded network (RA-TEN) model to divide time slots according to redundancy requirement. Based on RA-TEN, backup datacenters act as backup-and-forward center one by one until redundancy requirement is satisfied, to utilize their forwarding capability and therefore mitigate forwarding burdens on source backup datacenters. Furthermore, we formulate the disaster backup activity as a new two-step problem consisting of facility location and fraction multi-commodity flows over time. To obtain ideal load balance distribution on backup datacenters and transmission links during every time slot, we design a two-step Progressive Forwarding Disaster Backup (PFDB) algorithm including backup datacenter selection and link load distribution for multiple backup transfers. To flexibly and centrally manage data transmission and routing according to transfer properties and network status, we leverage the Software Defined Network (SDN) [18] as running network environment. In the SDN scenarios, we use central controller to collect network information, and distribute flow scheduling rules in every time slot.

The following chapters are organized as follows. In Sect. 2, we describe network model and construct RA-TEN to illustrate the main idea of progressive forwarding disaster backup. In Sect. 3, we propose and analyze PFDB algorithm in detail. In Sect. 4, we compare our algorithm with the state-of-the-art algorithms. At last, we summarize the paper and look forward to further research.

2. Network Formulation and RA-TEN Model

2.1 Network Formulation

We leverage a connected and directed graph $G = (V, E)$ to represent the cloud datacenter network, where V is the set of nodes, and E is the set of physical links between nodes. We use $DC = \{dc_1, dc_2, \dots\}$ to denote the set of original source datacenters. We use $BD = \{bd_1, bd_2, \dots\}$ to represent the set of candidate backup datacenters. Backup datacenters receive and store replicas to guarantee redundancy of r for the data in DC. We use e_{uv} to denote the directed link originating from node $u \in V$ and ending with node $v \in V$. To obtain high data availability and disaster tolerance, we leverage many-to-many relationship between source datacenters and their backup datacenters as in our earlier work [14].

We leverage $BK = \{BK_1, BK_2, \dots\}$ to denote the total backup transfers from all source datacenters, and use the $BK_i = \{bk_{i1}, bk_{i2}, \dots\}$ to denote the tens of or even hundreds of backup transfers originating from dc_i. Each backup transfer bk_{ij} which means the transmission of a replica is specified as a 5-tuple $(num, dt_{ij}, am_{ij}, bb_{ij}, t_{ij})$. Here num represents the unique number for the data block and its replicas, dt_{ij} represents its backup destination, am_{ij} represents its data amount, bb_{ij} represents the least bandwidth required to finish bk_{ij} on time (bottom bandwidth), and t_{ij} represents its time slot for backup data transmission. Because disaster backup activity always happens in a continuous time period, we can leverage dl to denote the unified deadline for all backup transfers. Then we can use $bb_{ij} = (r \cdot am_{ij})/dl$ to compute the bottom bandwidth for bk_{ij}.

For the division problem of time slots, we mainly focus backup transfers which always consume huge bandwidth and therefore impact network status significantly. We define a time slot $t \in \{t_1, t_2, \dots, t_r\}$ as the time period of transferring a replica. We obtain bottom bandwidth for every backup transfer via multiple paths. We define the transmission path set for all backup transfers by $path(t) =$

$\{path_1(t), path_2(t), \dots\}$ in the time slot t. We use $path_i(t) = \{p_{i1}(t), p_{i2}(t), \dots\}$ to denote the path set for BK_i and $p_{ij}(t)$ to denote the set of multiple paths for bk_{ij} in the time slot t. We use $c(e_{uv})(t)$ to denote the available bandwidth on e_{uv} in the time slot t. We use $bw(p_{ij}(t))$ to denote the bandwidth allocated to $p_{ij}(t)$ via multiple paths. We use $bw(path_i(t))$ to denote the total bandwidth allocated to BK_i in the time slot t as follows:

$$bw(path_i(t)) = \sum_{p_{ij}(t) \in path_i(t)} bw(p_{ij}(t)) \quad (1)$$

In the network G with time-varying network link capacity and backup datacenter storage space, the problem aims to mitigate source datacenter burden and balance backup loads. In our earlier work [13], [14], we have not considered mitigating data forwarding burden on source datacenters and fair load distribution on transmission links in disaster backup activity. To overcome these shortcomings, we analyze time-varying network for different stages of disaster backup activity and solve the progressive forwarding disaster backup.

2.2 RA-TEN Model

The TEN [19] can be used to convert dynamic flow over time problem to static flow problem in multiple time slots. Some researchers have improved TEN to divide time slots according to transmission deadline, describe network status over time, and design dynamic scheduling algorithms for multiple bulk data transfers among geo-distributed datacenters [7], [17]. However, their strategies have not considered backup datacenter selection and routing searching [7], or mitigating heavy burden on datacenters and transmission links [17]. In this paper, we propose RA-TEN model suitable for load balance in regular disaster backup activity.

To mitigate data forwarding burden concentrated in source datacenters, we newly propose role-switching method over time for backup datacenters. Given the unified deadline as total backup time period, we divide it equally into r time slots according to the redundancy requirement. Due to the huge network resource consumption in disaster backup activity, we mainly focus on the impact by backup transfers, rather than other daily network applications. In the first time slot, we transfer backup data from the original source datacenters, select backup datacenters and schedule flows. Then we leverage progressive forwarding disaster backup to guarantee redundancy. At the beginning of every following time slot, we check the backup datacenters in the previous time slot. For those ones with backup data whose replica number is less than the required redundancy, we define them as new source datacenters in the new time slot, and leverage forwarding disaster backup to the nearest available backup datacenters without the same backup data replicas by multipath routing. In this way, multiple different backup datacenters undertake the forwarding task in turn, thereby greatly reducing the burden on the original source datacenters.

In Fig. 1 and Fig. 2, we give a simple example to compare the traditional method with the role-switching method in RA-TEN model. The node dc_i covered by gray shade represents source datacenter with backup requirement. The nodes bd_1, bd_2, and bd_3 covered by blue shade represent candidate backup datacenters to store replicas. v_1 and v_2 are intermediate nodes.

In RA-TEN model, we divide the total time period into three parts according to the redundancy requirement of 3 replicas for dc_i. Then we search backup routing by two different strategies. In Fig. 1 (a)–(c), we construct backup routing for every replica with the same source datacenter by least hop number. In the three searching rounds, we choose backup routing as $dc_i \to bd_1$, $dc_i \to v_1 \to bd_2$ and $dc_i \to v_1 \to v_2 \to bd_3$. In Fig. 1 (d), we obtain the whole backup routing with total hop number as 6. dc_i acts as backup data sender throughout the whole process.

In Fig. 2 (a)–(c), we construct backup routing for every replica by our newly proposed role-switching method. The searching process of backup routing with role-switching method is as follows:

- In the first time slot, dc_i acts as source datacenter, with bd_1, bd_2, and bd_3 as candidate backup datacenters. We select bd_1 with the least hop number to be the backup datacenter and obtain the backup routing $dc_i \to bd_1$ which is the same as Fig. 1 (a). In the residual time slots, instead of acting as source datacenter, dc_i will act as an ordinary intermediate node.
- In the second time slot, we check bd_1 (the backup data-

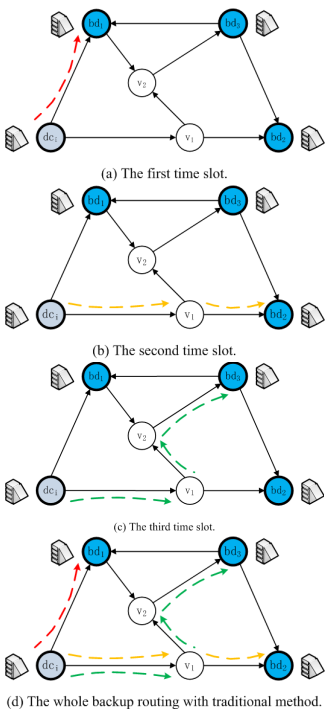

Fig. 1 Backup routing with the same source datacenter dc_i in RA-TEN model (r = 3).

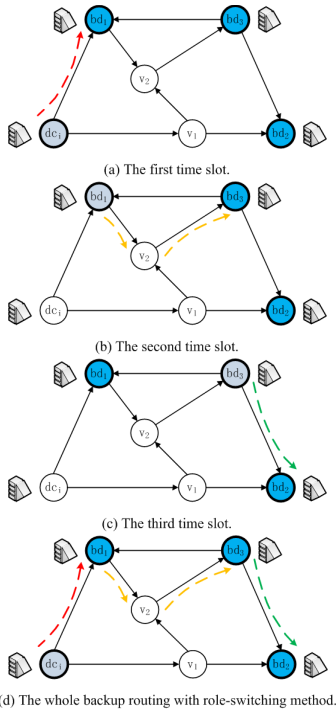

Fig. 2 Backup routing with role-switching method in RA-TEN model (r = 3).

center in the previous time slot) and find that the number of $dc_i's$ replica is 1 which is less than the required redundancy. So we make bd_1 as source datacenter, with bd_2, and bd_3 as candidate backup datacenters in this time slot. We select bd_3 with the least hop number to be the backup datacenter, and obtain the backup routing $bd_1 \rightarrow v_2 \rightarrow bd_3$ with hop number as 2. In the residual time slots, bd_1 will act as an ordinary intermediate node or candidate backup datacenter.

- In the third time slot, similarly, we make bd_3 as source datacenter. Although bd_1 and bd_2 are reachable candidate backup datacenters with the same hop number, we finally choose bd_2 as backup datacenter because the same replicas already exists in bd_1. And therefore we choose backup routing as $bd_3 \rightarrow bd_2$ with the hop number as 1.
- As a summary in Fig. 2 (d), the backup datacenter in the current time slot will act as new source datacenter to send the replica in the next time slot until we obtain sufficient redundancy. The role-switching method not only mitigates heavy forwarding burden concentrated in the same source datacenter dc_i as in Fig. 1 (d), but also shares forwarding burden among multiple backup datacenters in different time slots. Furthermore, we finally obtains smaller hop number as 4 which means less network resource consumption in Fig. 2 (d).

By the role-switching method, after receiving a replica in the previous time slot, the backup datacenter will act as source datacenter for this replica in the current time slot. The role-switching process continues until we obtain sufficient replicas in geo-distributed backup datacenters. This method shifts the forwarding task for the same backup data from original source datacenters to multiple backup datacenters over time with higher forwarding capability and more storage space than ordinary datacenters. Therefore we can effectively alleviate the forwarding pressure on the same source datacenter.

We should add that we mainly focus on mitigating heavy burden on the source datacenter during disaster backup process. For easy display, we assume that bottom bandwidth can be satisfied by single path and therefore have not considered multipath routing. However, this is an ideal assumption and is not suitable for the practical applications. In the next section for specific algorithm design, we will further consider load balance on transmission links, and satisfaction of bottom bandwidth requirement in practice. We distribute replicas on backup datacenters with receiving capacity, and allow backup transmission through multipath routing to obtain sufficient bandwidth for backup transfers under deadline constraint. We will give detailed introduction in Sect. 3.

3. Algorithm Design

Ant Colony Optimization (ACO) [20] is a highly innovative meta-heuristic algorithm inspired by the ant foraging behavior in the real world. In the foraging process, the single behavior of an ant is relatively simple, but the whole behavior of ant colony reflects intelligence. Ant colony can find the shortest path to food source in different environments, because they can transmit information through certain information mechanism, which is called pheromone. Ants release pheromones on the path that they have passed. They will follow the path of higher pheromone concentration in probabilistic, and each passing ant will leave its new pheromone on the path. As time progresses, the concentration of pheromone accumulated on the shorter path gradually increases, and therefore more ants will select the path. After a period of time, the entire ant colony will follow the shortest path to food source by this positive feedback mechanism. ACO is a heuristic global optimization algorithm in evolutionary computing fields. It is widely used to find paths for bulk data transmission in large-scale network [14], [21].

In this paper, we use $P_v(q+1)$ to describe the possibility of a certain node v being chosen in the $(q+1)th$ iteration. We define it with pheromone intensity $\tau_v(q+1)$ and heuristic information $\eta_v(q+1)$ occupying different importance factors $\phi = 0.8$ and $\varphi = 0.4$ respectively. We denote $CN(v)$ as the candidate set of v. The definition is as follows:

$$P_v(q+1) = \frac{(\tau_v(q+1))^{\phi} \cdot (\eta_v(q+1))^{\varphi}}{\sum_{w \in CN(v)} (\tau_w(q+1))^{\phi} \cdot (\eta_w(q+1))^{\varphi}} \quad (2)$$

3.1 PFDB Algorithm

As shown in Sect. 2, we leverage role-switching method over time to mitigate data forwarding burden concentrated

in original source datacenters. Furthermore, we aim to distribute backup load fairly on backup datacenters. Based on the newly proposed RA-TEN model and existing ACO metaheuristic, we design a two-step PFDB algorithm including capacity-constrained backup datacenter selection (CC-BDS) and fair link load distribution (FLLD), to select backup datacenter and determine transmission paths to realize progressive forwarding disaster backup among cloud datacenters.

We define $DC(t)$ and $BD(t)$ as the set of source datacenters and the set of candidate backup datacenters in the time slot t. The pseudo code of PFDB algorithm is as follows:

Algorithm 1: PFDB algorithm

Data: $G = (V, E)$, DC, BD, BK
Result: progressive forwarding disaster backup solution
Divide dl into $dl = \{t_1, t_2, \cdots, t_r\}$ according to redundancy requirement, and initialize RA-TEN;
for *every time slot* $t \in \{t_1, t_2, \cdots, t_r\}$ **do**
  if *t is the first time slot* **then**
    Set DC as the original set of source datacenters $DC(t)$ and sort the elements in ascending order of backup data amount, set $BK = \{BK_1, BK_2, \cdots\}$ according to $DC(t)$, and set BD as the original set of candidate backup datacenters $BD(t)$;
  else
    Update $DC(t)$ and $BD(t)$ with role-switching method according to the previous time slot;
  Select backup datacenter for every $BK_i \in BK$ with CC-BDS algorithm;
  Optimize link load distribution for every $bk_{ij} \in BK$ with FLLD algorithm;
Return progressive forwarding disaster backup solution;

3.2 CC-BDS Algorithm

In CC-BDS algorithm, we aim to implement backup datacenter selection for every BK_i in the time slot $t \in \{t_1, t_2, \cdots, t_r\}$. We consider fair load distribution on capacity-constrained backup datacenters and minimize the total hop number from every BK_i to the selected backup datacenters. Obviously, this is the facility location problem which is NP-hard [22]. We use $s(bd_k(t))$ to represent the storage capacity of bd_k in the time slot t. We define $\beta_k(t)$ as the ratio of the storage capacity in the backup datacenter bd_k to the total storage capacity of $BD(t)$ as follows:

$$\beta_k(t) = \frac{s(bd_k(t))}{\sum_{bd_k \in BD(t)} s(bd_k(t))}, \forall bd_k \in BD(t) \tag{3}$$

We define $hop_{BK_i}(t)$ as the hop number for BK_i in the time slot t, from the source datacenter holding BK_i to the selected backup datacenter. And we formulate the objective function as follows:

$$minimize\Big(\sum_{BK_i \in BK} hop_{BK_i}(t)\Big) \tag{4}$$

To obtain feasible solution, the following constraints should be satisfied:

$$y_{ik}(t) \cdot \left| \frac{\sum_{bk_{ij} \in BK_i} am_{ij}}{\sum_{BK_i \in BK} \sum_{bk_{ij} \in BK_i} am_{ij}} - \beta_k(t) \right| \le \omega_1, \tag{5}$$
$$\forall bd_k \in BD(t), \forall BK_i \in BK$$

$$\sum_{dc_i \in DC(t)} \Big(y_{ik}(t) \cdot (am_{ij})\Big) \le s(bd_k(t)), \forall bd_k \in BD(t) \tag{6}$$

$$\sum_{bd_k \in BD(t)} z_{ik}^{num}(t) \ge 1, \forall BK_i \in BK \tag{7}$$

$$\sum_{t \in \{t_1, t_2, \cdots, t_r\}} z_{ik}^{num}(t) \le 1, \forall bd_k \in BD(t), \forall BK_i \in BK \tag{8}$$

$$y_{ik}(t) = \begin{cases} 1 & if \ \ bd_k \ \ is \ \ BK_i's \ \ backup \\ & datacenter \ \ in \ \ time \ \ slot \ \ t \\ 0 & otherwise \end{cases} \tag{9}$$

$$z_{ik}^{num}(t) = \begin{cases} 1 & if \ \ the \ \ data \ \ num \ \ has \ \ replica \\ & in \ \ bd_k \ \ in \ \ time \ \ slot \ \ t \\ 0 & otherwise \end{cases} \tag{10}$$

The ω_1 in (5) is a very small nonnegative decimal. The backup load distribution constraint in (5) ensures close approximation of backup load ratio to storage capacity ratio for every bd_k, to achieve fair load distribution on backup datacenters. The storage capacity constraint in (6) ensures that the overall backup load stored in every backup datacenter cannot exceed its maximum storage capacity. The backup redundancy constraint in (7) ensures that at least one replica for the backup data should be placed in some backup datacenter in the time slot t. The remote distribution constraint in (8) ensures that in a backup datacenter, there should be not more than one replica for the same backup data. The variable value constraints in (9) and (10) define the values of $y_{ik}(t)$ and $z_{ik}^{num}(t)$ in different situations.

In the following subsections, we define the pheromone, heuristic information, fitness evaluation, and then propose the pseudo code of CC-BDS algorithm.

3.2.1 Pheromone and Heuristic Information

We give the related definitions in every time slot t. For every BK_i, we set unique pheromone and heuristic information to avoid mutual interference in the selection process. We define hop_{cur} as the total hop number for the current solution cur in the qth iteration, and hop_{gb} as the total hop number for the global best solution gb. After the qth iteration, we choose the used backup datacenters in cur and gb for BK_i, and enhance pheromone intensity by pheromone updating as follows:

$$\tau_k^i(q+1) = (1 - \rho_1(\tau_k^i(q))) + \Delta\tau_k^i(q) \tag{11}$$

$$\Delta\tau_k^i(q) = \lambda_1\chi_i(q) + \lambda_2\sigma_i(q) \tag{12}$$

$$\chi_i(q) = \frac{y_k^{cur,i}}{hop_{cur} \cdot (1 + \epsilon_i^{cur}(q))} \tag{13}$$

$$\sigma_i(q) = \frac{y_k^{gb,i}}{hop_{gb} \cdot (1 + \epsilon_i^{gb}(q))} \tag{14}$$

$$y_k^{cur,i} = \begin{cases} 1 & if \quad bd_k \quad is \quad BK_i's \quad backup \\ & \quad datacenter \quad for \quad cur \\ 0 & otherwise \end{cases} \tag{15}$$

$$y_k^{gb,i} = \begin{cases} 1 & if \quad bd_k \quad is \quad BK_i's \quad backup \\ & \quad datacenter \quad for \quad gb \\ 0 & otherwise \end{cases} \tag{16}$$

$$\epsilon_i^{cur}(q) = y_k^{cur,i} \cdot \frac{1}{|BD(t)|} \cdot \sum_{bd_k \in BD(t)} \left| \frac{\sum_{bk_{ij} \in BK_i} am_{ij}}{\sum_{BK_i \in BK} \sum_{bk_{ij} \in BK_i} am_{ij}} - \beta_k(t) \right| \tag{17}$$

$$\epsilon_i^{gb}(q) = y_k^{gb,i} \cdot \frac{1}{|BD(t)|} \cdot \sum_{bd_k \in BD(t)} \left| \frac{\sum_{bk_{ij} \in BK_i} am_{ij}}{\sum_{BK_i \in BK} \sum_{bk_{ij} \in BK_i} am_{ij}} - \beta_k(t) \right| \tag{18}$$

We use ρ_1 as evaporating parameter controlling the evaporating speed of pheromone after every iteration. We use λ_1 and λ_2 to express the influences of *cur* and *gb* on the increment of pheromone intensity in the $(q+1)th$ iteration. We use $|BD(t)|$ to denote the number of backup datacenters in $BD(t)$. It is worth noting that the value of $|BD(t)|$ might be different in *cur* and *gb*. We use $\frac{\sum_{bk_{ij} \in BK_i} am_{ij}}{\sum_{BK_i \in BK} \sum_{bk_{ij} \in BK_i} am_{ij}}$ to express the ratio of backup load in the backup datacenter bd_k to the total backup load of BK. We define $\epsilon_i^{cur}(q)$ and $\epsilon_i^{gb}(q)$ as the average bias of $\frac{\sum_{bk_{ij} \in BK_i} am_{ij}}{\sum_{BK_i \in BK} \sum_{bk_{ij} \in BK_i} am_{ij}}$ to $\beta_k(t)$ in *cur* and *gb*, respectively. Obviously, the smaller value of average bias means more uniform backup load distribution on backup datacenters.

Heuristic information $\eta_k^i(q+1)$ reflects the prior and deterministic factors on bd_k in the $(q+1)th$ routing search process for BK_i. We define it with the length of shortest path from dc_i to bd_k, and the residual storage capacity in bd_k. The heuristic information in the $(q+1)th$ iteration is as follows:

$$\eta_k^i(q+1) = \varpi_1 \cdot \frac{s(bd_k)}{1 + dis(dc_i, bd_k)} \tag{19}$$

We use ϖ_1 to adjust the value of $\eta_k^i(q+1)$, use $dis(dc_i, bd_k)$ to denote the length of the shortest path from dc_i to bd_k, and use $s(bd_k)$ to denote the residual storage capacity in bd_k.

3.2.2 Fitness Evaluation

In every iteration, we need to evaluate the fitness of the solution. To realize backup datacenter selection, we focus on the following metrics: minimum total hop number and fair load distribution on backup datacenters. So the fitness evaluation for a backup datacenter selection solution bds is a compound function of the two factors above:

$$fitness_1(bds) = \begin{cases} 0 & if \sum_{bd_k \in BD(t)} z_{ik}^{num}(t) < 1, \forall BK_i \in BK \\ \alpha_1 \cdot e^{-hop_{bds}} + \alpha_2 \cdot e^{-(1+\epsilon_{bds})} & otherwise \end{cases} \tag{20}$$

$$\epsilon_{bds} = \sqrt{\frac{1}{|BD(t)-1|} \sum_{bd_k \in BD(t)} \left(load_k - avgload(bds) \right)^2} \tag{21}$$

$$load_k = \sum_{dc_i \in DC(t)} \left(\sum_{bk_{ij} \in BK_i} \left(am_{ij} \cdot y_{ik}(t) \right) \right) \tag{22}$$

$$avgload(bds) = \frac{1}{|BD(t)|} \sum_{bd_k \in BD(t)} \left(load_k \right) \tag{23}$$

To ensure redundancy r, CC-BDS should find at least one available backup datacenter in every time slot to store one replica for BK_i. We use hop_{bds} to denote the total hop number in bds. We use ϵ_{bds} to denote the value of backup load distribution variation in bds. We use $load_k$ to denote the total backup load on bd_k, and use $avgload(bds)$ to denote the average backup load on backup datacenters. α_1 and α_2 are the weighted functions to represent the importance of corresponding factors. We define $\alpha_1, \alpha_2 \geq 0$, and $\alpha_1 + \alpha_2 = 1$. In the real algorithms, different values can be set according to the requirements of the user. In the simulation, we set α_1, α_2 to be 0.6, and 0.4 respectively by experience.

3.2.3 Pseudo Code of CC-BDS

The CC-BDS algorithm is described as follows:

Step 1: Initialize the current solution and the global best solution. In the current time slot, we choose the backup datacenter to place replicas for every backup transfer in BK_i. The selection process leverages greedy method to give priority to the nearest backup datacenter. If the residual storage capacity of the nearest backup datacenter is insufficient to store the backup load generated by the backup transfers in BK_i, we will continue to consider the next nearest backup datacenter until find available backup datacenter. The above selection result is taken as the initial current solution and the initial global optimal solution.

Step 2: Initialize the pheromones, etc.

Step 3: Set the number of iterations.

Step 4: In every iteration, we set a group of ants to search for CC-BDS solutions. Every ant constructs backup datacenter selection solution for every BK_i: we calculate the selection probability according to (2), use the roulette wheel selection procedure to select the backup datacenter for every BK_i, and update the network status. After every iteration, we evaluate the solutions of each ant according to the fitness function, choose the best one as the current solution, and update the global optimal solution according to fitness if necessary. Then we update the value of pheromone.

Step 5: Inspect the terminal condition. If the number of iterations reaches the given maximum value or the best solution has not changed for some time, then go to Step 6. Otherwise, go to Step 4.

Step 6: Return the optimal solution.

The pseudo code of CC-BDS algorithm is as follows:

Algorithm 2: CC-BDS algorithm

Data: $G = (V, E)$, $DC(t)$, $BD(t)$, BK
Result: backup datacenter selection for every BK_i
for *every* BK_i **do**
  Compute the shortest path to every backup datacenter;
  Assign it to the nearest backup datacenter bd_k;
  Compute backup load of bd_k;
  if bd_k *exceeds its backup load capacity* **then**
    Assign BK_i to the next nearest backup datacenter;
Set the above selection solution as *cur* and *gb*;
Initialize pheromone values, etc.;
while *termination condition not met* **do**
  for *every ant* **do**
    for *every* BK_i **do**
      Calculate heuristic information according to (19);
      Select the backup datacenter according to (2);
      Update network $G = (V, E)$;
    Update the solution set;
  Evaluate the solutions with fitness function and choose *cur*;
  if *find a better solution than gb according to fitness function* **then**
    Update *gb*;
  Apply updating rule (11);
Return backup datacenter selection solution;

CC-BDS is terminated when it converges or reaches maximum iteration number. In every iteration, multiple ants participate in the candidate set construction process for BK_i. We set m as the number of ants. At most m candidate backup datacenters are found for BK_i, so the time complexity is approximately $O(m \cdot |DC(t)| \cdot |V|)$.

3.3 FLLD Algorithm

After backup datacenter selection, we allocation bandwidth to every backup transfer bk_{ij} by multipath routing in the time slot $t \in \{t_1, t_2, \cdots, t_r\}$, to complete data transmission before the backup deadline.

We define $u(e_{uv}(t))$ as the link utilization on e_{uv} in the time slot t as follows:

$$u(e_{uv}(t)) = \frac{\sum_{path_i(t) \in path(t)} \sum_{p_{ij} \in path_i(t)} \left(bw(p_{ij}(t)) \cdot x_{uv}^{ij}(t) \right)}{c(e_{uv}(t))} \tag{24}$$

In order to reduce the impact on other daily network services, we aim to minimize the maximum utilization on all transmission links. Therefore, we formulate the objective function as:

$$minimize\Big(maximum\big(u(e_{uv}(t)) \big) \Big) \tag{25}$$

We minimize the maximum link utilization for disaster backup activity by fair load distribution on links. We use en to represent the total number of backup transmission links in the time slot t, and compute the average utilization value $avgu$ of these links as follows:

$$avgu = \frac{1}{en} \sum_{e_{uv} \in path(t)} u(e_{uv}(t)) \tag{26}$$

To obtain feasible solution, the following constraints should be satisfied for all bulk backup transfers:

$$\left(\sum_{path_i(t) \in path(t)} \sum_{p_{ij} \in path_i(t)} \left(bw(p_{ij}(t)) \cdot x_{uv}^{ij}(t) \right) \right) \leq c(e_{uv}(t)), \forall bk_{ij} \in BK \tag{27}$$

$$bw(p_{ij}(t)) \geq bb_{ij}(t), \forall bk_{ij} \in BK \tag{28}$$

$$\sum_{v \in V} f_{uv}^{ij} - \sum_{v \in V} f_{vu}^{ij} = \begin{cases} -bw(p_{ij}(t)) & u = bd_j \\ 0 & otherwise \\ bw(p_{ij}(t)) & u = dc_i \end{cases}, \quad \forall bd_j \in BD(t), \forall dc_i \in DC(t) \tag{29}$$

$$x_{uv}^{ij}(t) = \begin{cases} 1 & if \quad e_{uv} \quad is \quad in \quad the \quad path \quad p_{ij}(t) \\ 0 & otherwise \end{cases}, \quad \forall p_{ij}(t) \in path(t) \tag{30}$$

$$\sqrt{\frac{1}{en-1} \sum_{e_{uv} \in path(t)} \left(u(e_{uv}(t)) - avgu \right)^2} \leq \omega_2 \tag{31}$$

The link capacity constraint in (27) ensures that the overall traffic through every link cannot exceed its maximum link capacity. The bottom bandwidth constraint in (28) ensures that the total bandwidth allocated for every backup transfer should be greater than or equal to its bottom bandwidth. We define f_{uv}^{ij} in (29) as the flow from dc_i to bd_j through e_{uv}. The flow conservation constraint in (29) ensures that for every backup transfer, the input traffic equals to the output traffic at any intermediate node in the transmission paths. The variable value constraint in (30) defines the value of $x_{uv}^{ij}(t)$ in different situations. The ω_2 in (31) is a very small nonnegative decimal. The utilization bias constraint in (31) limits the gap between the utilization of every backup transmission link and the average link utilization.

In the following subsections, we define the pheromone, heuristic information, fitness evaluation, and then propose the pseudo code of FLLD algorithm.

3.3.1 Pheromone and Heuristic Information

In FLLD algorithm, we realize sufficient bandwidth allocation for every bk_{ij} by multipath routing. We consider fair load distribution on network links and minimize the maximum link utilization.

We give the related definitions in every time slot t. For every bk_{ij}, we set unique pheromone and heuristic information to avoid mutual interference in the path searching process. We define mlu_{cur} as the maximum link utilization for the current solution *cur* in the *qth* iteration, and mlu_{gb} as the maximum link utilization for the global best solution *gb*. After the *qth* iteration, we choose the used links in *cur* and *gb* for bk_{ij}, and enhance the pheromone intensity by pheromone updating as follows:

$$\tau_{uv}^{ij}(q+1) = (1-\rho_2)(\tau_{uv}^{ij}(q)) + \Delta\tau_{uv}^{ij}(q) \tag{32}$$

$$\Delta\tau_{uv}^{ij}(q) = (\mu_1\chi_{ij}(q) + \mu_2\sigma_{ij}(q)) \tag{33}$$

$$\chi_{ij}(q) = \frac{y_{uv}^{cur,ij}}{(1+mlu_{cur})(1+alu_{cur})(1+\epsilon_{cur}^{ij}(q))} \tag{34}$$

$$\delta_{ij}(q) = \frac{y_{uv}^{gb,ij}}{(1+mlu_{gb})(1+alu_{gb})(1+\epsilon_{gb}^{ij}(q))} \tag{35}$$

$$y_{uv}^{cur,ij} = \begin{cases} 1 & if\ \ e_{uv}\ \ is\ \ in\ \ p_{ij}(t)\ \ for\ \ cur \\ 0 & otherwise \end{cases} \tag{36}$$

$$y_{uv}^{gb,ij} = \begin{cases} 1 & if\ \ e_{uv}\ \ is\ \ in\ \ p_{ij}(t)\ \ for\ \ gb \\ 0 & otherwise \end{cases} \tag{37}$$

$$alu_{cur} = \frac{1}{en_{cur}} \sum_{e_{uv}\in path(t)} u(e_{uv}(t)) \tag{38}$$

$$alu_{gb} = \frac{1}{en_{gb}} \sum_{e_{uv}\in path(t)} u(e_{uv}(t)) \tag{39}$$

$$\epsilon_{cur}^{ij}(q) = \sqrt{\frac{1}{en_{cur}-1} \sum_{e_{uv}\in path(t)} \left(u(e_{uv}(t)) - alu_{cur}\right)^2} \tag{40}$$

$$\epsilon_{gb}^{ij}(q) = \sqrt{\frac{1}{en_{gb}-1} \sum_{e_{uv}\in path(t)} \left(u(e_{uv}(t)) - alu_{gb}\right)^2} \tag{41}$$

We use ρ_2 as evaporating parameter controlling the evaporating speed of pheromone after every iteration. We use μ_1 and μ_2 to express the influences of *cur* and *gb* on the increment of pheromone intensity in the $(q+1)th$ iteration. We define alu_{cur} and alu_{gb} as the average link utilization for the transmission links in *cur* and *gb*, respectively. We use en_{cur} and en_{gb} to denote the total number of transmission links in *cur* and *gb* respectively. We leverage $\epsilon_{cur}^{ij}(q)$ and $\epsilon_{gb}^{ij}(q)$ as the utilization bias between the utilization of every backup transmission link and the average link utilization in the qth iteration in *cur* and *gb* respectively. It is worth noting that the value of $u(e_{uv}(t))$ might be different in *cur* and *gb*.

Heuristic information $\eta_{uv}^{ij}(q+1)$ reflects the prior and deterministic factors on e_{uv} in the $(q+1)th$ routing search process for bk_{ij}. We define it with the residual bandwidth capacity $c(e_{uv})$ and the link utilization on e_{uv}. We use ϖ_2 to adjust the value of $\eta_{uv}^{ij}(q+1)$. The heuristic information in the $(q+1)th$ iteration is as follows:

$$\eta_{uv}^{ij}(q+1) = \varpi_2 \cdot \frac{c(e_{uv})}{1+u(e_{uv}(t))} \tag{42}$$

3.3.2 Fitness Evaluation

In every iteration, we need to evaluate the fitness of the solution. To realize load balance on disaster backup transmission links, we focus on the following metrics: maximum link utilization and link utilization variation. So the fitness evaluation for a link load distribution solution *lld* is a compound function of the two above:

Algorithm 3: FLLD algorithm

```
Data: G = (V, E), DC(t), BD(t), BK
Result: backup routing for every bk_ij
Set parameters values, initialize pheromone values, etc.;
while termination condition not met do
    for every ant do
        for every bk_ij do
            if its total bandwidth satisfies bottom bandwidth
            constraint in (28) then
                Continue;
            while not reach bk'_ij s backup datacenter do
                Calculate heuristic information according to
                (42);
                Select the next node according to (2);
                Compute bandwidth allocaton according to
                (27) and (29);
                if reach bk'_ij s backup datacenter then
                    Add the datacenter to path_i(t);
                    Update network G = (V, E);
                if the next node not exist then
                    Break;
        Update the solution set;
    if not satisfy bottom bandwidth constraint in (28) then
        Goto 3;
    Obtain cur and evaluate it with fitness function;
    if find a better solution than gb according to fitness function
    then
        Update gb;
    Apply updating rule (32);
Return link load distribution solution;
```

$$fitness_2(lld) = \begin{cases} 0 & if\ \ bw(p_{ij}(t)) \le bb_{ij}(t), \forall p_{ij}(t) \in path(t) \\ \alpha_3 \cdot e^{-(1+mlu_{lld})} + \alpha_4 \cdot e^{-(1+var_{lld})} & otherwise \end{cases} \tag{43}$$

$$var_{lld} = \sqrt{\frac{1}{en_{lld}-1} \sum_{e_{uv}\in path(t)} \left(u(e_{uv}(t)) - avgu\right)^2} \tag{44}$$

We use mlu_{lld} to denote the maximum link utilization in *lld*. We use var_{lld} to denote link utilization variation in *lld*. We use en_{lld} to denote the total number of transmission links in *lld*. Similarly, we set α_3 and α_4 to be 0.7 and 0.3 respectively by experience.

3.3.3 Pseudo Code of FLLD

The FLLD algorithm is described as follows:

Step 1: Initialize the pheromones, etc.

Step 2: Set the number of iterations.

Step 3: In every iteration, we set a group of ants to search for a solution in the current time slot. Every ant constructs a transmission path for every bk_{ij}. First, we check whether the total bandwidth obtained has met $bk'_{ij}s$ bottom bandwidth constraint. If so, we stop transmission path searching for bk_{ij}; otherwise, we calculate the selection probability according to (2), use the roulette wheel selection procedure to select the next node for every bk_{ij}. If the ant reaches the backup data center designated by CC-BDS algorithm, we record the transmission path, updates the network status, and successfully ends the search run for transmission

path; if there is no next available node, the transmission path search fails and is terminated. For every bk_{ij}, if the total allocated bandwidth by multipath routing is less than its bottom bandwidth bb_{ij}, the search process will continue until the convergence condition is satisfied, for example that the total bandwidth satisfies rate requirement or the iteration number reaches a specified value. We obtain the current solution with the path set, and update the global optimal solution according to fitness if necessary. Then we update the value of pheromone.

Step 4: Inspect the terminal condition. If the number of iterations reaches the given maximum value or the best solution has not changed for some time, then go to Step 5. Otherwise, go to Step 3.

Step 5: Return the optimal solution.

The pseudo code of FLLD algorithm is as shown in the previous page.

Similarly, we set m as the number of ants, and set $|bk|$ as the number of backup transfers. For every bk_{ij}, at most m paths are generated, so the time complexity is approximately $O(m \cdot |bk| \cdot |V|)$.

4. Performance Evaluation

4.1 Environment and Configuration

We compare PFDB algorithm with three representative algorithms. First, we implement the basic one-to-many disaster backup transmission algorithm which minimizes the maximum link utilization (Basic-MinMax). Second, we choose the Fair-Rotating and Ratio-Aware Ant Colony Optimization (FRRA-ACO) algorithm in our earlier work [14]. FRRA-ACO specifies flow allocation ratio among backup datacenters with limited receiving capacity, and leverages rotary routing search and ratio adjustment rules to realize fair bandwidth allocation for multiple concurrent transfers. Third, we choose the bulk transfer scheduling algorithm among multiple datacenters (SnF-LexMin-eTEG) [7]. SnF-LexMin-eTEG lexicographically minimizes the network congestion vector with store-and-forward assisted transfer mode, to reduce link congestion and balance the entire network traffic.

We implement above algorithms on a DELL OPTIPLEX 9020 server equipped with eight Intel(R) Core(TM) i7-4790 3.60 GHz CPUs and 8 GB RAM. We perform experiments on the Waxman model [23], which is very close to the real network and is used to generate random network topologies. In our proposed algorithms (i.e., FRRA-ACO [14], and PFDB), we set the maximum iteration number as 50 and the ant number as 50 at the beginning. If there is no evolution in four consecutive circulations, the circulation would stop.

In evaluation experiments, we set available bandwidth uniformly distributed on each link within [1000, 2000] (Gbps). To guarantee sufficient data redundancy, we set the required replica number $r = 3$, and divide 3 time slots accordingly. We set backup transmission deadline as 120 minutes according to previous research [5]. We consider two situations. In the first case, we generate a 30-node network topology as shown in Fig. 3. We use the numbers to denote available bandwidth on the links. We select node 1, 11, 14, 18, 22 and 26 as backup datacenters, observe performance comparison with the increase of total backup data amount. Large enterprises such as Google can process about 100 PB data daily in geo-distributed application DCs [11]. Normally, not exceeding five percent of the daily data requires backup [24]. Therefore, we set the total backup data amount ranging from 0.5 PB to 5 PB. In the second case, we fix the total backup data amount as 3 PB, and observe performance comparison with the increase of node number ranging from 10 to 60.

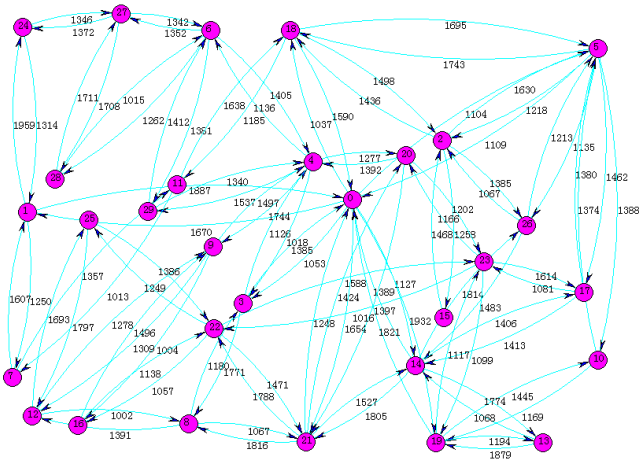

Fig. 3 30-node network topology.

4.2 Simulation Results

We compare PFDB with other algorithms from the aspects of maximum utilization and average utilization on transmission links, and backup load distribution on backup datacenters. At last, we observe the performance changes of PFDB with different values of ϕ and φ in $P_v(q+1)$.

4.2.1 Maximum Link Utilization

In Fig. 4 and Fig. 5, we illustrate the comparison of maximum link utilization with the increase of data amount and number of nodes (including intermediate nodes, source datacenters and backup datacenters), respectively. Especially for performance evaluation of PFDB based on RA-TEN model, we observe its maximum link utilization in different time slots.

FRRA-ACO has not considered load balance on transmission links. It leverages rotary routing search to utilize network transmission capability on different links in iterative runs and prune flows by ratio adjustment rules. However, in order to realize proportional bandwidth allocation, congestion on some links is still unavoidable. Obviously, FRRA-ACO always fully utilizes some links in every time

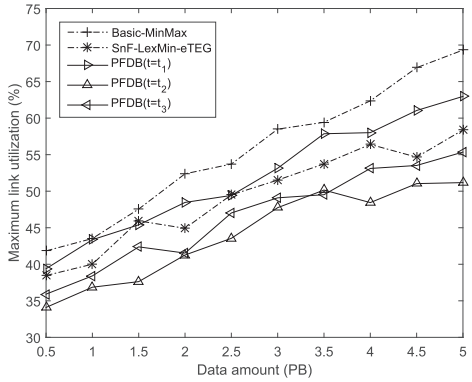

Fig. 4 Comparison of maximum link utilization with increase of data amount.

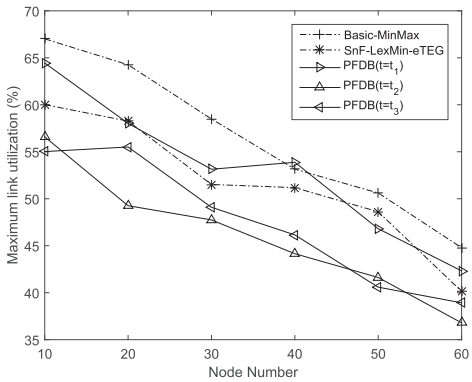

Fig. 5 Comparison of maximum link utilization with increase of node number.

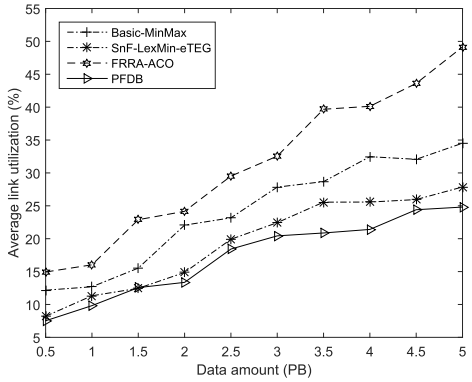

Fig. 6 Comparison of average link utilization with increase of data amount.

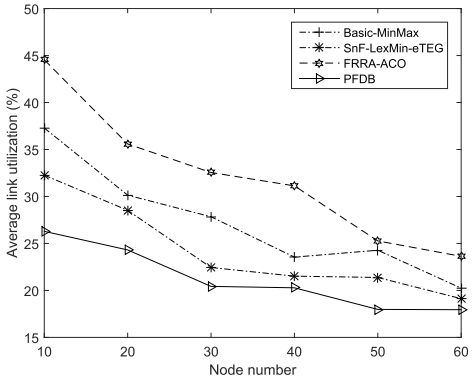

Fig. 7 Comparison of average link utilization with increase of node number.

slot to achieve its optimization objective. Therefore, we have not added it into the comparison of maximum link utilization. Basic-MinMax and SnF-LexMin-eTEG have not considered choosing different backup routing to mitigate congestion for multiple replicas of the same backup data in each time slot. In other words, their transmission paths will remain unchanged in different time slot. In that case, we choose the maximum link utilization value among all time slots for comparison. SnF-LexMin-eTEG outperforms Basic-MinMax because it provides better balance on the traffic by lexicographically minimizing the network congestion with store-and-forward transmission mode.

PFDB guides ants to search for backup transmission paths jointly considering maximum link utilization of existing solutions, and residual bandwidth capacity and utilization on the candidate links. Although its performance is inferior to that of SnF-LexMin-eTEG in the first time slot ($t = t_1$), PFDB improves its performance significantly in the subsequent time slots ($t = t_2$ and $t = t_3$). The possible reasons are: (1) PFDB releases some link capacity occupied by backup transfers in the first time slot. This measure helps to reduce utilization on these busy links. (2) PFDB changes the role of original source datacenter to normal node and chooses new source datacenter in every new time slot. This measure not only mitigates forwarding burden on source node, but also utilizes idle capacity on some links among datacenters. (3) PFDB leverages role-switching method in every time slot. We can construct new path set for backup transfer to adjust backup load distribution on critical links. Most of the time, PFDB outperforms Basic-MinMax and SnF-LexMin-eTEG in reducing maximum link utilization.

4.2.2 Average Link Utilization

In Fig. 6 and Fig. 7, we compare the average link utilization among Basic-MinMax, SnF-LexMin-eTEG, FRRA-ACO, and PFDB with increase of data amount and node number, respectively. Although FRRA-ACO specifies flow allocation ratio for backup datacenters, there is no further load adjustment on transmission links after proportional bandwidth allocation, especially for those shared links. SnF-LexMin-eTEG performs better than Basic-MinMax and FRRA-ACO, because it leverages store-and-forward to multiplex idle links in time dimension and iteratively optimizes the maximum link congestion. With lowest average link utilization, PFDB outperforms others not only because of positive feedback and heuristic search for load balancing, but also benefits from the progressive backup procedure with role-switching method in every time slot to utilize idle links.

4.2.3 Backup Load Distribution

As in our earlier work [14], [25], to obtain greater clarity

about the comparison of backup load distribution fairness on backup datacenters among different algorithms, we introduce the metric called fairness ratio. We define it as the ratio value of fairness between these algorithms. We first define the fair load distribution factor with γ as follows:

$$\gamma = 1 - \sqrt{\frac{1}{bn-1}\sum_{bd_k \in BD}(\gamma_{bd_k} - \overline{\gamma})^2} \tag{45}$$

$$\gamma_{bd_k} = \frac{\sum_{dc_i \in DC}\left(y_{ik}(t_r)\cdot(am_{ij})\right)}{s(bd_k(t_r))}, \forall bd_k \in BD \tag{46}$$

$$\overline{\gamma} = \frac{1}{bn}\sum_{bd_k \in BD}\left(\frac{\sum_{dc_i \in DC}\left(y_{ik}(t_r)\cdot(am_{ij})\right)}{s(bd_k(t_r))}\right) \tag{47}$$

Here we use bn to denote the number of backup datacenters in BD. We use γ_{bd_k} to denote the ratio of total backup data amount in a backup datacenter bd_k to its own storage capacity. We use $\overline{\gamma}$ to denote the average value of γ_{bd_k} for all backup datacenters. We compute γ in the time slot t_r to observe the final load distribution on backup datacenters.

And then, we run algorithms to get their fair load distribution factor as γ_{Basic}, γ_{SnF}, γ_{FRRA} and γ_{PFDB}. We choose the fair load distribution factor of one algorithm as standard value (usually the factor with largest value), and evaluate backup load distribution by the ratio of standard value to the fair load distribution factor of other algorithms. In this paper, we calculate the fairness ratio of FRRA-ACO to other three algorithms as follows:

$$fr_{FRRA-Basic} = \frac{\gamma_{FRRA}}{\gamma_{Basic}} \tag{48}$$

$$fr_{FRRA-SnF} = \frac{\gamma_{FRRA}}{\gamma_{SnF}} \tag{49}$$

$$fr_{FRRA-PFDB} = \frac{\gamma_{FRRA}}{\gamma_{PFDB}} \tag{50}$$

Obviously, smaller fairness ratio implies better balance of backup load distribution among all destination nodes. In Fig. 8 and Fig. 9, we illustrate the comparison of fairness ratio with increasing data amount and node number, respectively. FRRA-ACO outperforms others because it constructs network transmission model which specifies flow ratio to backup datacenters according to their receiving capacity and therefore controls backup load distribution to achieve load balance on backup datacenters. Neither of Basic-MinMax and SnF-LexMin-eTEG has considered load balance problem on backup datacenters. But the latter performs better because it lexicographically and iteratively minimizes the maximum link congestion, favorable for more balanced load distribution. PFDB guides backup datacenter selection process to ensure the approximation of backup load ratio to storage capacity ratio for every backup datacenter in CC-BDS, to achieve uniform backup load distribution. For convenience of flow scheduling to balance link load, we do not limit the flow ratio to backup datacenters as in our earlier work [14]. But PFDB still achieves uniform distribution of backup load on backup datacenters and outperforms Basic-MinMax and SnF-LexMin-eTEG.

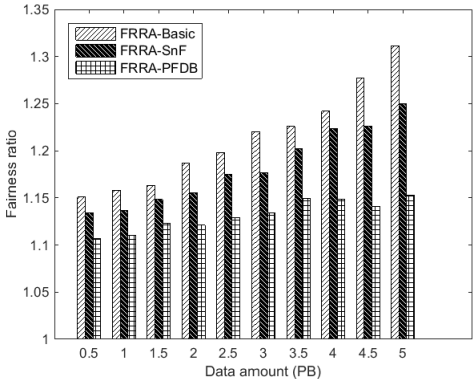

Fig. 8 Comparison of fairness ratio with increase of data amount.

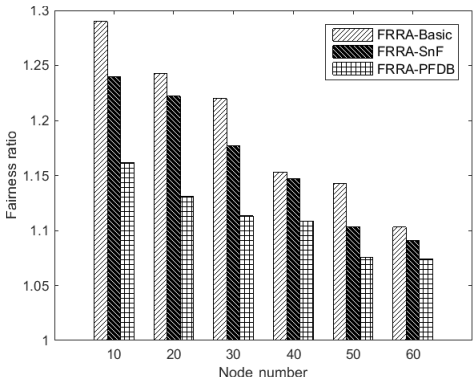

Fig. 9 Comparison of fairness ratio with increase of node number.

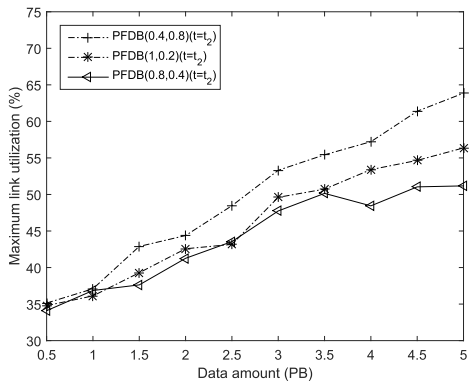

Fig. 10 Comparison of maximum link utilization with increase of data amount.

4.2.4 Performance Changes of PFDB

In the previous experiments, we set $\phi = 0.8$ and $\varphi = 0.4$ in $P_v(q+1)$. Here we evaluate link utilization and load distribution while changing values of ϕ and φ. We set $\phi = 0.4$ and $\varphi = 0.8$, $\phi = 0.8$ and $\varphi = 0.4$, $\phi = 1$ and $\varphi = 0.2$, respectively, and carry out experiments in the first case of Sect. 4.1. In Fig. 10, Fig. 11 and Fig. 12, we illustrate the comparison with different values of ϕ and φ. Special to note is that, we choose the time slot $t = t_2$ to observe maximum link utiliza-

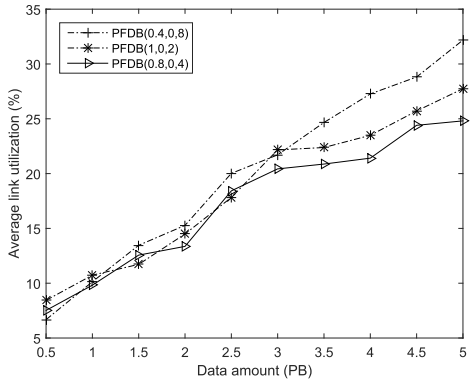

Fig. 11 Comparison of average link utilization with increase of data amount.

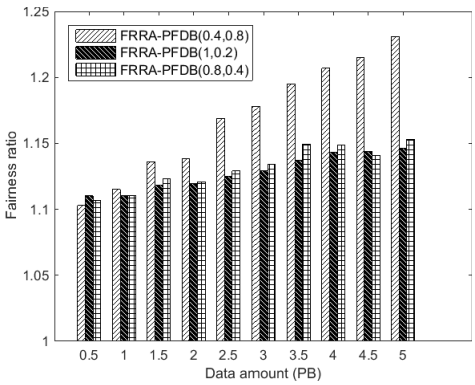

Fig. 12 Comparison of fairness ratio with increase of data amount.

tion, because the optimization effect is most obvious in this time slot.

As in Fig. 12, PFDB performs best in load distribution fairness when $\phi = 1$ and $\varphi = 0.2$, because the larger pheromone weight can lead to more emphasis on fair load distribution in the backup datacenter selection stage. But at the same time, the influence of heuristic information is weakened, reduceing the attention to the distance between source datacenter and backup datacenter. Then in the link load distribution stage, more transmission links may be occupied, and the difficulty of link utilization optimization increases. As a result, the optimization effect of link utilization with $\phi = 1$ and $\varphi = 0.2$ is not the best among these cases.

In the case of $\phi = 0.4$ and $\varphi = 0.8$, the influence of pheromone is weakened in the link load distribution stage, reducing the attention to maximum link utilization and average link utilization. Therefore, the performance in reducing link utilization is not ideal. The same situation also occurs in the backup datacenter selection phase: weakened influence of pheromone reduces the attention to the fairness of backup load distribution and therefore the performance in backup load distribution is not ideal.

With the setting of $\phi = 0.8$ and $\varphi = 0.4$, PFDB obtains relatively good performance in reducing link utilization and improving load distribution fairness simultaneously. Therefore, we use this setting in the implementation process of PFDB.

5. Conclusion

We focus on the load balance problem in disaster backup transmission among cloud datacenters. We formulate a new two-step problem consisting of facility location and fraction multi-commodity flows over time, propose progressive forwarding strategy to distribute backup loads, and mitigate burdens on backup datacenters and transmission links in different time slots with a two-step algorithm in which every backup datacenter acts as backup-and-forward center by the role-switching method over time. Under the condition of guaranteeing transmission completion and backup redundancy, our strategy achieves good performance with lower maximum link utilization, lower average link utilization, and acceptable backup load distribution fairness on backup datacenters.

For further research, we will aim at reducing disaster backup cost with multicast jointly considering backup datacenter selection and transmission paths.

Acknowledgments

The study was supported by the National Natural Science Foundation of China (NSFC No. 61672323), the Natural Science Foundation of Shandong Province (Grant No. ZR2019MF072, ZR2018LF007, ZR2017MF050, ZR2016YL011), the Fundamental Research Funds of Shandong University (Grant No. 2017JC043), the Key Research and Development Program of Shandong Province (2017GGX10122, 2017GGX10142, 2019GNC106027, 2018GGX101005, 2017CXGC0701, 2016GGX109001), and the Project of Shandong Province Higher Educational Science and Technology Program (No. J17KA049). We especially thank the editors and the reviewers for their valuable comments and suggestions.

References

[1] Q. Xia, Z. Xu, W. Liang, and A.Y. Zomaya, "Collaboration- and fairness-aware big data management in distributed clouds," IEEE Trans. Parallel Distrib. Syst., vol.27, no.7, pp.1941–1953, July 2016.
[2] Y. Ogawa, G. Hasegawa, and M. Murata, "Virtual network allocation for fault tolerance balanced with physical resources consumption in a multi-tenant data center," IEICE Trans. Commun., vol.E98-B, no.11, pp.2121–2131, 2015.
[3] S. Ferdousi, M. Tornatore, M.F. Habib, and B. Mukherjee, "Rapid data evacuation for large-scale disasters in optical cloud networks," J. Opt. Commun. Netw., vol.7, no.12, pp.B163–B172, Dec. 2015.
[4] How effective is your data centers disaster recovery plan? [Online]. Available: http://www.lifelinedatacenters.com/datacenter/effective-disaster-recovery-plan/
[5] N. Laoutaris, M. Sirivianos, X. Yang, and P. Rodriguez, "Inter-datacenter bulk transfers with netstitcher," Proc. ACM SIGCOMM 2011, Toronto, Ontario, Canada, pp.74–85, 2011.
[6] Y. Zhang, J. Jiang, K. Xu, et al., "BDS: A centralized near-optimal overlay network for inter-datacenter data replication," Proc. 13th Eurosys Conference, Porto, Portugal, 2018.

[7] Y. Wang, S. Su, A.X. Liu, and Z. Zhang, "Multiple bulk data transfers scheduling among datacenters," Comput. Netw., vol.68, pp.123–137, 2014.
[8] M. Noormohammadpour, C.S. Raghavendra, S. Rao, and S. Kandula, "Dccast: Efficient point to multipoint transfers across datacenters," Proc. USENIX Workshop on Hot Topics in Cloud Computing (HotCloud 17), Santa Clara, CA, 2017.
[9] M. Noormohammadpour, C.S. Raghavendra, S. Kandula, and S. Rao, "QuickCast: Fast and efficient inter-datacenter transfers using forwarding tree cohorts," Proc. IEEE INFOCOM 2018, Honolulu, HI, USA, pp.225–233, 2018.
[10] Y. Wu, Z. Zhang, C. Wu, C. Guo, Z. Li, and F.C.M. Lau, "Orchestrating Bulk Data Transfers across Geo-Distributed Datacenters," IEEE Trans. Cloud Comput., vol.5, no.1, pp.112–125, Jan. 2017.
[11] J. Yao, P. Lu, L. Gong, and Z. Zhu, "On fast and coordinated data backup in geo-distributed optical inter-datacenter networks," J. Lightw. Technol., vol.33, pp.3005–3015, July 2015.
[12] P. Lu, L. Zhang, X. Liu, J. Yao, and Z. Zhu, "Highly efficient data migration and backup for big data applications in elastic optical inter-data-center networks," IEEE Netw., vol.29, no.5, pp.36–42, Sept./Oct. 2015.
[13] X. Li, H. Wang, S. Yi, and X. Yao, "Receiving-capacity-constrained rapid and fair disaster backup for multiple datacenters in SDN," Proc. ICC 2017, Paris, France, pp.1–6, 2017.
[14] X. Li, H. Wang, S. Yi, X. Yao, F. Zhu, and L. Zhai, "Redundancy-guaranteed and receiving-constrained disaster backup in cloud data center network," IEEE Access, vol.6, pp.47666–47681, 2018.
[15] L. Ma, X. Jiang, B. Wu, T. Talebi, A. Pattavina, and N. Shiratori, "Cost-efficient data backup for data center networks against ϵ-time early warning disaster," Proc. HPSR 2016, Yokohama, Japan, pp.22–26, 2016.
[16] P. Lu, Q. Ling, and Z. Zhu, "Maximizing utility of time-constrained emergency backup in inter-datacenter networks," IEEE Commun. Lett., vol.20, no.5, pp.890–893, May 2016.
[17] X. Xie, Q. Ling, P. Lu, W. Xu, and Z. Zhu, "Evacuate before too late: distributed backup in inter-DC networks with progressive disasters," IEEE Trans. Parallel Distrib. Syst., vol.29, no.5, pp.1058–1074, May 2018.
[18] B.A.A. Nunes, M. Mendonca, X.-N. Nguyen, K. Obraczka, and T. Turletti, "A survey of software-defined networking: Past, present, and future of programmable networks," IEEE Commun. Surveys Tuts., vol.16, no.3, pp.1617–1634, 3rd Quart., 2014.
[19] L.R. Ford and D.R. Fulkerson, "Constructing Maximal Dynamic Flows from Static Flows," Operations Research, vol.6, no.3, pp.419–433, 1958.
[20] M. Dorigo and G. Di Caro, "Ant colony optimization: a new meta-heuristic," Proc. CEC99, Washington, DC, USA, pp.1470–1477, 1999.
[21] X. Li, H. Wang, S. Yi, X. Yao, F. Zhu, and L. Zhai, "Optimizing concurrent evacuation transfers for geo-distributed datacenters in SDN," Proc. ICA3PP, Helsinki, Finland, pp.99–114, 2017.
[22] V. Arya, N. Garg, R. Khandekar, A. Meyerson, K. Munagala, and V. Pandit, "Local search heuristic for k-median and facility location problems," SIAM J. Comput., vol.33, no.3, pp.544–562, 2004.
[23] M. Naldi, "Connectivity of Waxman topology models," Comput. Commun., vol.29, no.1, pp.24–31, 2005.
[24] J. Dean and S. Ghemawat, "Mapreduce: simplified data processing on large clusters," Commun. ACM, vol.51, no.1, pp.107–113, Jan. 2008.
[25] X. Li, H. Wang, S. Yi, and L. Zhai, "Cost-efficient disaster backup for multiple data centers using capacity-constrained multicast," Concurr. Comput.-Pract. Exp., DOI: 10.1002/cpe.5266.

Xiaole Li received the Ph.D. degree from Shandong University, China, in 2019. He is currently a teacher in Linyi University, China. His main research interests include network optimization, network algorithms, network intelligence, and network architecture and protocol.

Hua Wang received the Ph.D. degree from Nanjing University of Science and Technology, China, in 2003. From 2005 to 2008, he was a Post-Doctoral Fellow with the School of Computer Science and Technology, Shandong University, China. He is currently a Professor of the School of Software, Shandong University, China, where he leads the Network Optimization Research Group. His research interests include network optimization, network algorithms, network intelligence, network architecture and protocol, and network simulation. His research has been supported by China Next Generation Internet Project, and NSF of China.

Shanwen Yi received the master's degree from Shandong University, China, in 2010. He is currently pursuing the Ph.D. degree with Shandong University. His main research interests include network optimization, network algorithms, network intelligence, and network architecture and protocol.

Linbo Zhai received the Ph.D. degree from Beijing University of Posts and Telecommunications, China, in 2010. From 2014 to 2017, he was a Post-Doctoral Fellow with the School of Computer Science and Technology, Shandong University, China. He is currently a teacher in Shandong Normal University, China. His main research interests include cognitive radio, crowdsourcing and distributed network optimization.