

Contents lists available at [SciVerse ScienceDirect](http://SciVerse.Sciencedirect.com)

Journal of Computer and System Sciences

www.elsevier.com/locate/jcss

Distributed data possession checking for securing multiple replicas in geographically-dispersed clouds

Jing He^a, Yanchun Zhang^{a,*}, Guangyan Huang^a, Yong Shi^{b,c}, Jie Cao^d

^a Centre for Applied Informatics, School of Engineering & Science, Victoria University, PO Box 14428, Melbourne, VIC 8001, Australia

^b Research Center on Fictitious Economy and Data Science, Chinese Academy of Sciences, China

^c College of Information Science and Technology, University of Nebraska at Omaha, USA

^d Jiangsu Provincial Key Laboratory of E-Business, Nanjing University of Finance and Economics, Nanjing, 210003, China

ARTICLE INFO

Article history:

Received 25 August 2010

Received in revised form 20 May 2011

Accepted 8 December 2011

Available online 29 December 2011

Keywords:

Cloud computing

Multiple replicas data possession checking

Data security

Optimal spanning tree

Complete bidirectional directed graph

ABSTRACT

Distributing multiple replicas in geographically-dispersed clouds is a popular approach to reduce latency to users. It is important to ensure that each replica should have availability and data integrity features; that is, the same as the original data without any corruption and tampering. Remote data possession checking is a valid method to verify the replicas's availability and integrity. Since remotely checking the entire data is time-consuming due to both the large data volume and the limited bandwidth, efficient data-possession-verifying methods generally sample and check a small hash (or random blocks) of the data to greatly reduce the I/O cost. Most recent research on data possession checking considers only single replica. However, multiple replicas data possession checking is much more challenging, since it is difficult to optimize the remote communication cost among multiple geographically-dispersed clouds. In this paper, we provide a novel efficient Distributed Multiple Replicas Data Possession Checking (DMRDPC) scheme to tackle new challenges. Our goal is to improve efficiency by finding an optimal spanning tree to define the partial order of scheduling multiple replicas data possession checking. But since the bandwidths have geographical diversity on the different replica links and the bandwidths between two replicas are asymmetric, we must resolve the problem of Finding an Optimal Spanning Tree in a Complete Bidirectional Directed Graph, which we call the FOSTCBDG problem. Particularly, we provide theories for resolving the FOSTCBDG problem through counting all the available paths that viruses attack in clouds network environment. Also, we help the cloud users to achieve efficient multiple replicas data possession checking by an approximate algorithm for tackling the FOSTCBDG problem, and the effectiveness is demonstrated by an experimental study.

© 2011 Elsevier Inc. All rights reserved.

1. Introduction

In recent years, cloud computing has become a hot research topic that aims to flexibly deliver infrastructure, platforms and software as services in a pay-as-you-go model on the Internet so that users are able to access and deploy applications from anywhere in the world [1,2]. Cloud computing has the potential to share hardware and software resources at the datacenters (or clouds) with the general public to reduce the price of Software as a Service (SaaS) [1]. Particularly, a SaaS provider can explore multiple geographically-dispersed datacenters according to the users' location distribution [3] or the

* Corresponding author. Fax: +61 399194908.

E-mail addresses: jing.he@vu.edu.au (J. He), Yanchun.Zhang@vu.edu.au (Y. Zhang), abyssshuang@gmail.com (G. Huang), yshi@gucas.ac.cn (Y. Shi), caojie690929@163.com (J. Cao).

web pages' local popularity [4] to build a SaaS and provide data (or computation) replicas that are physically located in distant datacenters to reduce the latency to the end users [5–8]. A typical example is that a SaaS uses a distributed web crawler [9] to collect web pages at a single site hosted on a datacenter may store data in a local storage facility and then distribute data replicas to other remote datacenters. Amazon CloudFront [10] offers a solution to this problem by using the edge locations for faster data delivery to the end users.

However, data possession evidence is important to convince the cloud users that the replicas distributed to multiple geographically-dispersed clouds are available and consistent with the original data. Data availability and integrity are two vital features of the clouds. Actually, distributing replicas to multiple geographically-separated clouds may bring risks either in storage or during transmission. Transmitting the replicas from one cloud to another far away cloud may cause a partial or whole data loss. Besides, the replicas may be tampered or deleted by attackers or malicious clouds. Since the volume of the replica data is large, it will cost the user a lot of time to access and check the entire replica data. Also, the I/O costs for remotely transmitting the replica data to the user are expensive. Therefore, efficient data-possession-verifying methods generally sample and check a small hash (or random blocks) of the data to reduce the I/O cost greatly. Most of the recent research on data possession checking considers only single replica. However, multiple replicas data possession checking is much more challenging, since it is difficult to optimize the remote communication cost among multiple geographically-dispersed clouds.

The simplest method, which we call centre-oriented multiple replicas data possession checking (e.g. the MR-PDP scheme in [20]) is to check each replica one-by-one directly from a verifier. However, it is not efficient to check all of the replicas, since the direct access to the replicas from the verifier may not be the most efficient path due to the limited communication bandwidth on the Internet [11,12]. The advantage of centre-oriented checking is that it avoids introducing new risks of untrusted factors, but its disadvantages are: (1) the bandwidth of the verifier is the bottleneck since it connects to all replicas; and (2) the connection between the verifier and each target replica may be not efficient.

In this paper, we provide a novel efficient Distributed Multiple Replicas Data Possession Checking (DMRDPC) scheme to overcome the two disadvantages of centre-oriented checking. In the DMRDPC scheme, we first find an optimal spanning tree to define the partial order of scheduling multiple replicas data possession checking. This is a very complex task, since bandwidths have geographical diversity on different links of different replicas and the bandwidths between two replicas are asymmetric, and thus we have to find an optimal spanning tree with the verifier as the root in a Complete Bidirectional Directed Graph (CBDG), which connects the verifier and all the replicas. Then, according to the scheduling partial order, we start the data possession checking from the verifier, who checks all of its children. Those replicas that have passed the verification can go on checking the data possession of their children. If some replicas fail in the checking, they can obtain one copy from its parent before they continue checking the data possession of their own children.

The advantages of our DMRDPC scheme are two fold: we may achieve better communication speed to access each replica; and several checking tasks can be executed in parallel since they are initialed from the location of the different replicas.

We notice that reducing the communication time for multiple replicas data possession checking is more critical, since the computation time can be simply determined by the sampling-based data possession checking algorithms. Also, our DMRDPC scheme reduces the communication time by choosing a better route to check replicas in multiple separated clouds instead of reducing the transmitted data size. The research in this paper will help the cloud users to achieve efficient multiple replicas data possession checking. The effectiveness is demonstrated by an experimental study.

The rest of this paper is organized as follows. In Section 2, we provide the DMRDPC scheme and discuss its most significant problem: the FOSTCBDG problem. In Section 3, we introduce related work. In Section 4, we develop the theories to exactly tackle the FOSTCBDG problem. In Section 5, we provide a novel approximate algorithm for resolving the FOSTCBDG problem. In Section 6, we evaluate our proposed algorithm by an experimental study. Finally, Section 7 concludes this paper.

2. Problem definition

In this section, we first model the communication topology in the process of the multiple replicas data possession checking in the cloud computing environment, then we provide the DMRDPC scheme and discuss that the Finding Optimal Spanning Trees in the Complete Bidirectional Directed Graph (FOSTCBDG) problem is the key problem in the DMRDPC scheme. Finally, we present the details of the FOSTCBDG problem.

2.1. Modeling communication topology using complete bidirectional directed graph

The communication in the process of distributed multiple replicas data possession checking is different from broadcasting in P2P wireless networks, where every node receives data directly and simultaneously from the source node through an omnidirectional wireless channel, and the completion time of broadcasting in the worst-case has the complexity of $O(n^2)$ [13]. But there are new factors to determine the communication speed in the process of distributed multiple replicas data possession checking:

- The available bandwidths between remote clouds vary greatly, which are not predictable or not always proportional to geographical distances.
- The available network bandwidths for bidirectional connections between two clouds are asymmetric.

Table 1
Bandwidth test for Google websites.

| Checked From | Google Sites (Kbps). | | | | | | |
|----------------------------|----------------------|------------|-------|-------|-----------|------------|------------|
| | de | au | hk | br | cl | uk | us |
| Santiago, Chile (760 Kbps) | 43 | 46 | 54 | 54 | 91 | 53 | 65 |
| Beijing, China (5 Mbps) | 107 | 305 | 386 | 338 | 253 | 312 | 253 |
| Sydney, Australia (5 Mbps) | 81 | 175 | 148 | 90 | 77 | 145 | 97 |
| Dortmund, Germany (5 Mbps) | 576 | 1,146 | 1,186 | 1,184 | 780 | 1,169 | 775 |
| Gloucester, UK (5 Mbps) | 768 | 572 | 676 | 606 | 570 | 860 | 212 |
| Chicago, IL (45 Mbps) | 165 | 992 | 948 | 1,183 | 760 | 1,170 | 651 |
| Washington, DC (3 Mbps) | 132 | 352 | 364 | 394 | 325 | 390 | 452 |
| Los Angeles, CA (1.5 Mbps) | 135 | 456 | 1 | 338 | 351 | 4 | 449 |
| Detroit, MI (5 Mbps) | 154 | 327 | 365 | 365 | 268 | 360 | 417 |

Note: de: google.de, au: google.com.au, hk: google.com.hk, br: google.com.br, cl: google.cl, uk: google.co.uk, us: google.com.

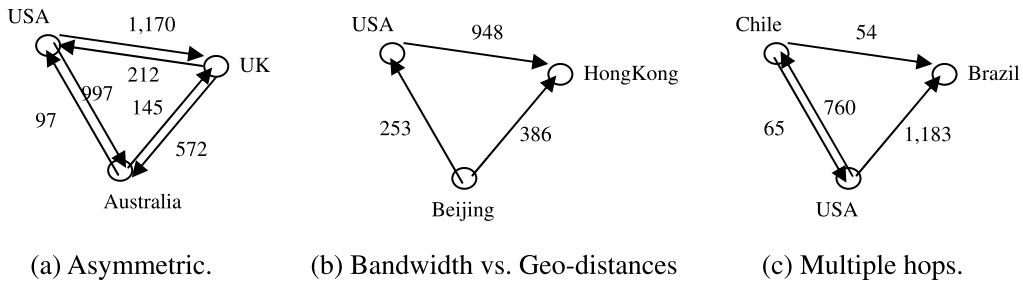
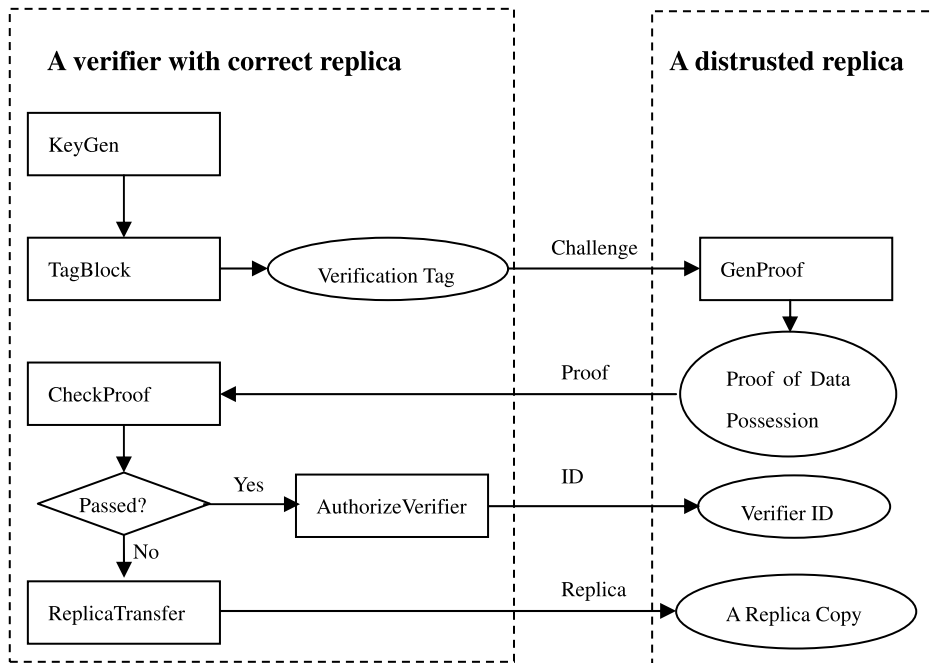


Fig. 1. Bandwidths examples.

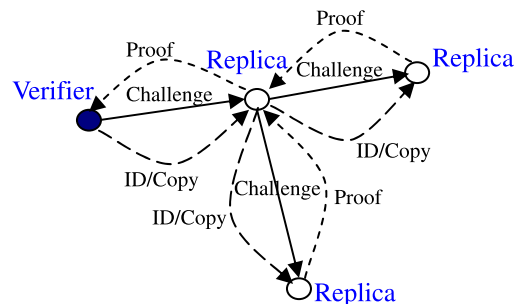
We first study the geographical diversity [14] of the network bandwidths for data transmitting in a clouds environment. We check the download speeds of several Google datacenters (or clouds) [15] located in different countries from a set of other non-Google clouds in different locations around the world by using the tool in [16]; these download speeds are used as bandwidths on the links of inter-clouds. We achieved the bandwidths in Table 1 on 30th April, 2010; note that the results vary with time. We always choose download instead of upload to transmit data between two clouds for the transmitting efficiency. Observing the bandwidths in Table 1, we can summarize the following points:

- Bandwidths do not decrease with increasing geographic distances. Table 1 shows that local visiting does not achieve the fastest transfer of data. Also, Fig. 1(b) shows that the bandwidth from Beijing to Hong Kong is 386 Kbps, which is lower than the bandwidth 948 Kbps from the USA to Hong Kong, although Beijing is geographically closer to Hong Kong than the USA.
- Bandwidth between a pair of sites is asymmetric. Fig. 1(a) shows some examples: the bandwidth of visiting a website in the UK from the USA is 1170 Kbps, while the bandwidth of visiting a website in the USA from the UK is 212 Kbps; we denote this as $(UK, USA) = (1170 \text{ Kbps}, 212 \text{ Kbps})$. Other connections are $(UK, AU) = (145 \text{ Kbps}, 572 \text{ Kbps})$ and $(AU, USA) = (997 \text{ Kbps}, 97 \text{ Kbps})$.
- The transmitting speeds do not increase with the number of hops. Fig. 1(c) shows an example where the transmitting speed from Chile to Brazil is lower than the transmitting speed through a two-hop path: from Chile to the USA and then to Brazil.

Based on the above factors, we represent the topology of multiple replicas data possession checking in the clouds network by a Complete Bidirectional Directed Graph (CBDG) $G = (V, E)$, where V is a set of clouds (called vertices), $E = V \times V$ is a set of bidirectional communication links (called arcs) and bandwidths are set as weights on arcs. Note that there are two arcs: \vec{vu} and \vec{uv} with different weights between a pair of vertices: v and u , where $v \in V$ and $u \in V - \{v\}$.



(a) A data possession checking process between a verifier and a replica.



(b) Distributed multiple replica data possession checking.

Fig. 2. The DMRDPC Scheme.

The advantages of a CBDG are: (1) the CBDG flexibly models the bandwidths with geographical diversity; and (2) the CBDG expresses asymmetric bandwidths between clouds.

2.2. Distributed Multiple Replicas Data Possession Checking (DMRDPC) scheme

In this subsection, we present the Distributed Multiple Replicas Data Possession Checking (DMRDPC) scheme. Our DMRDPC scheme consists of six basic algorithms:

- KeyGen: is a key generation algorithm run by verifiers.
- TagBlock: is run by verifiers to generate the verification tag for a file block, which will be used for challenging the distrusted replicas.
- GenProof: is run by replicas to generate a proof of data possession.
- CheckProof: is run by verifiers to validate the proof of data possession generated by replicas.
- ReplicaTransfer: is run by verifiers to transmit a correct copy of the replica to the verifying-failed replicas.
- AuthorizeVerifier: is run by verifiers to authorize a verifier ID to the verifying-passed replicas.

Generally, single replica data possession checking like in [19] only comprises the first four algorithms; but our DMRDPC scheme for checking multiple replicas includes two more special algorithms: ReplicaTransfer and AuthorizeVerifier. We explain the main idea of the DMRDPC scheme in Fig. 2. In Fig. 2(a), we plot a data possession checking process between

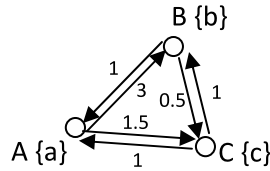


Fig. 3. Example of FOSTCBDG problem.

a verifier and a replica. First KeyGen generates keys that will be used later and TagBlock generates the verification tag to challenge the target distrusted replica. The distrusted replicas use GenProof to generate proof to respond to the challenge and then the proof is transmitted back to the verifier. The verifier user CheckProof to determine if the challenge has passed. If yes, the verifier uses AuthorizeVerifier to generate a verifier ID and sends it to the replica side; if no, the verifier uses ReplicaTransfer to transmit a correct replica copy to the replica side to replace the incorrect replica.

In Fig. 2(b), we show the data flow for checking multiple replicas. Note that the scheduling partial order is determined by an optimal spanning tree.

We have modeled the communication topology of the cloud computing environment into a CBDG. Suppose the CBDG network is known. The DMRDPC scheme is given as follows:

- Step 1: Find an optimal spanning tree, *OST*, in a CBDG, where *OST* connects all the replicas to the verifier and the verifier is the root of the tree.
- Step 2: In *OST*, only the parent nodes that have been proved to possess correct replica data can verify their children nodes. If a node, *i*, fails the data possession checking, *i* will get a copy from its parent.
- Step 3: If all the replicas are checked, this algorithm stops.

In the DMRDPC scheme, which is different from the MR-PDP scheme in [20], communication time is the principal cost compared to computation time, since the bandwidth for remote large data transmission is very limited. Thus, we schedule the data possession checking task for replicas in an optimal partial order to reduce the total transmission time. The verifier will firstly check data possession of some replicas and then those replicas that have been proved will go on checking data possession of the rest replicas. Thus, we may achieve better bandwidth to access each replica; besides, several checking tasks can be executed in parallel since they are initialed from the location of the different replicas. According to [17–19], the computation time of data possession checking for one replica is fixed, since all the replicas are the same. But it spends different communication time on data possession checking for two replicas on different clouds, since the data transmission speeds are different on the Internet. Thus, this paper focuses on optimizing the communication time of multiple replicas data possession checking. Note that only the replicas that have been proved correct can execute the data possession checking for other unchecked replicas.

It is critical to find an optimal spanning tree, *OST*, in a CBDG, since *OST* can help generate the optimal partial order of verifying scheduling and can reduce the communication time, particularly when a replica fails in the verifying and thus a large replica needs to transmit from the verifier to the replica. Thus, generating an *OST* is the most important part of the DMRDPC scheme and it needs to tackle the FOSTCBDG problem introduced in the next subsection.

2.3. Finding Optimal Spanning Trees in the Complete Bidirectional Directed Graph (FOSTCBDG) problem

We suppose a Complete Bidirectional Directed Graph (CBDG) with *m* vertices, where dispersed datacenters are vertices and weights (e.g. available network bandwidth) are set on directed arcs that connect pairs of datacenters. The FOSTCBDG problem is to minimize the communication time by finding an optimal spanning tree to define the partial order for scheduling multiple replicas data possession checking, starting from the verifier to the other *m* – 1 replicas that are located in remote different datacenters. The FOSTCBDG problem can be tackled by finding an optimal spanning tree in the CBDG. Without loss of generality, we define the FOSTCBDG problem in Definition 1.

Definition 1. The FOSTCBDG problem is defined as follows:

- (1) **INPUT:** Initially, *m* distributed clouds each of which has a replica, are connected by a complete bidirectional directed graph with $(m - 1)m$ weighted arcs and only one vertex, *v*, as the verifier vertex.
- (2) **OUTPUT:** A spanning tree, which defines the partial order to schedule the multiple replicas data possession checking process to check the other *m* – 1 vertices that have replicas.
- (3) **OBJECTIVE:** The objective is to minimize the total communication time in the process of *m* – 1 replicas data possession checking.

An example of the FOSTCBDG problem is shown in Fig. 3. Given a cloud set $\Sigma = \{A, B, C\}$ and time for moving a unit data on each arc, there are data sets: {*a*} on cloud A, {*b*} on cloud B and {*c*} on cloud C shown in Fig. 3. The aim is to

check whether $a = b = c$ as shown in Fig. 3. Suppose A is the verifier, there are only three spanning trees: $\{A \rightarrow B, A \rightarrow C\}$, $\{A \rightarrow B \rightarrow C\}$ and $\{A \rightarrow C \rightarrow B\}$ for completing 2 replicas (on B and C) data possession checking; the optimal spanning tree is the last one.

We will provide the exact solution in Section 4 and approximate solutions in Section 5 for resolving the FOSTCBDG problem.

3. Related work

In this section, we present the literature survey in relation to three aspects: data possession checking, clouds network model and finding optimal spanning trees.

3.1. Data possession checking schemes

Remote data possession checking protocols have been developed to check the integrity and availability of users' data; generally, they satisfy two requirements: (1) the verifier does not need access to the remote entire replica data in the process of data possession checking; (2) the verifier has unlimited verification times [20]. In [17], probabilistic proofs of possession are generated by sampling random sets of blocks from the server to reduce I/O costs. In [18], Krohn–Freedman–Mazires' function is adopted to generate a small hash of the data for checking instead of checking the entire data. Moreover, checking multiple replicas is more complex than checking one replica. In [20], a Multiple-Replica Provable Data Possession (MR-PDP) protocol is proposed to check whether multiple replicas are really stored at the cloud storage servers and the incorrect replicas can be recovered from correct replicas. The MR-PDP protocol in [20] discusses the computation time but does not discuss communication time. An efficient Remote Data Possession Checking (RDPC) scheme in [19] reduces both computation time and communication time; the communication time is reduced since this scheme reduces the transmitted data size. Our DMRDPC scheme focuses on improving communication time by finding an optimal route in a CBDG.

3.2. Clouds network models

In [21], the clouds network where the nodes have homogeneous receiving rates are considered. The research that is similar to this paper deals with more diverse bandwidths; but this work cannot satisfy our goal, either. For instance, in [22], different bandwidths on different links are considered; but the bidirectional bandwidths between two nodes is symmetric. The most related clouds network model is in [23], where bandwidths inter nodes are different and bandwidths between two nodes are asymmetric; but it doesn't define the model formally. Our proposed clouds network model is suitable for a more general situation.

3.3. Finding optimal spanning trees

Our goal is to find an optimal spanning tree in a CBDG; this is a totally new problem. In [24], a distributed algorithm is designed to compute a spanning tree in an extended LAN. This algorithm does not satisfy our goal, since it only aims to find a loop-free spanning tree but not an optimal spanning tree. In [25], a Generalized Minimum Spanning Tree Problem (GMSTP) is considered. Given an undirected graph whose nodes are partitioned into mutually exclusive and exhaustive node sets, the GMSTP is to find a minimum-cost tree which includes exactly one node from each node set. The GMSTP for an undirected graph is proved to be NP-hard [25]. A Minimum Spanning Tree (MST) can be taken as a special case of the GMSTP where each cluster consists of exactly one node [26]. However, the FOSTCBDG problem is different from the GMSTP, since it aims to find an optimal tree from a large number of spanning trees in a CBDG; this is far more complex than finding an MST in an undirected graph.

4. Theories for exactly finding optimal spanning trees in complete bidirectional directed graph

In this section, to resolve a general FOSTCBDG problem, we first define three sub problems: All Replicas Verified by All Verifiers (ARVAV problem), All Verifiers Work (AVW problem) and At Least One Verifier Work (ALOVW problem). Then, we develop several theorems to compute the number of possible spanning trees involved in an FOSTCBDG problem.

4.1. Basic concepts

We use A to denote a set in which each vertex (or node) is proved to have a correct replica, and use B to denote a set in which each vertex (or node) has not been verified. If a node in B is verified by any node in A , this node is removed from B and put into A . If a node is verified, that means the node has a correct replica. We call nodes in A verifiers and nodes in B replicas.

Definition 2 (All Replicas Verified by All Verifiers (ARVAV problem)). Given two sets of nodes: n verifiers in A and k replicas in B , $h_n(k)$ denotes the number of solutions for the ARVAV problem that satisfies the following conditions:

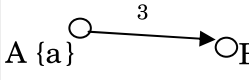
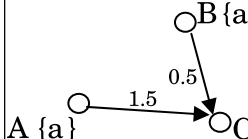
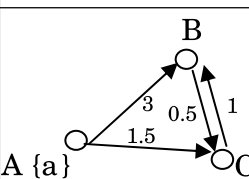
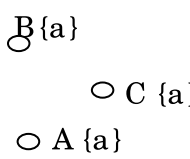
| Number of solutions | Graph | Number of solutions | Graph |
|---|---|---|---|
| $F_1(2) = 1$ solution: A, $A \rightarrow B$ (Cost: 3). $f_1(2) = 1$ solution. |  <p>(a) One node to one.</p> | $F_2(3) = 2$ solutions: (1) A, B, $A \rightarrow C$ (Cost: 1.5); (2) A, B, $B \rightarrow C$ (Cost: 0.5). $f_2(3) = 0$ solution. |  <p>(c) Two nodes to one.</p> |
| $F_1(3) = 3$ solutions: (1) A, $A \rightarrow B$, $A \rightarrow C$ (Cost: 4.5); (2) A, $A \rightarrow B$, $B \rightarrow C$ (Cost: 3.5); (3) A, $A \rightarrow C$, $C \rightarrow B$ (Cost: 2.5). $f_1(3) = 3$ solutions. |  <p>(b) One node to two.</p> | $F_3(3) = 1$ solution. $f_3(3) = 0$ solution. |  <p>(d) Three nodes to zero.</p> |

Fig. 4. Simple examples.

- (1) Every replica in B must be verified by a verifier in A;
- (2) Every verifier in A must verify at least one replica in B.

Definition 3 (All Verifiers Work (AVW problem)). Given a CBDG graph with m nodes, $f_n(m)$ ($m > n$) denotes the number of solutions of an AVW problem that is defined as: each of n verifiers in A that initially has a correct replica must verify at least one of the $m - n$ replicas in B. This means all of the n verifiers in A must participate in the activity of verifying the remaining $m - n$ replicas in B. Every replica in B must be verified.

Definition 4 (At Least One Verifier Work (ALOVW problem)). Given a CBDG graph with m nodes, $F_n(m)$ ($m > n$) denotes the number of solutions for an ALOVW problem that is defined as: At least one of the n verifiers in A that initially have the correct replicas must verify all of the remaining $m - n$ replicas in B. Note that there may be one verifier, two verifiers, ..., and n verifiers in A that participate in the activity of verifying the remaining $m - n$ replicas in B. Every replica in B must be verified.

4.2. Theories for resolving the FOSTCBDG problem

The FOSTCBDG problem is more complex than the GMSTP [25], since we cannot determine whether a spanning tree is optimal or not unless we enumerate all the possible spanning trees. This subsection presents theories to compute the number of possible spanning trees involved in an FOSTCBDG problem.

4.2.1. Simple examples

First, we show simple examples to compute the number of spanning trees in an ALOVW problem and an AVW problem based on Definition 4 and Definition 3. Fig. 4 shows that $F_1(2) = 1$, $F_1(3) = 3$, $F_2(3) = 2$ and $F_3(3) = 1$ are simple examples of the number of solutions for the ALOVW problem; and $f_1(2) = 1$, $f_1(3) = 3$, $f_2(3) = 0$ and $f_3(3) = 0$ are simple examples of the number of solutions for the AVW problem. Note that in the process of identifying a spanning tree, all the arcs whose head endpoints are adjacent to the nodes with data are removed from the CBDG graph to simplify the problem.

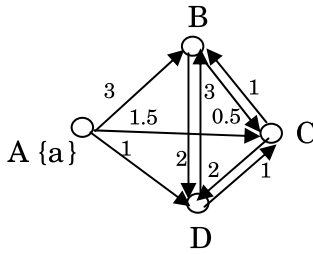
Fig. 5 shows the process of computing the number of spanning trees for an FOSTCBDG problem with 4 vertices. The smaller scale (less than 4 vertices) FOSTCBDG problems are already given in Fig. 4, which are explored to compute $F_1(4)$. We can see from Fig. 5 that $f_1(4) = C_3^1 F_1(3) + C_3^2 F_2(3) + C_3^3 F_3(3) = 16$, then $F_1(4) = f_1(4) = 16$. More examples are shown in Fig. 6 and Fig. 7, where $F_2(4) = 8$, $f_2(4) = 2$ (deduced by using Definition 3) and $F_3(4) = 3$.

4.2.2. General theories

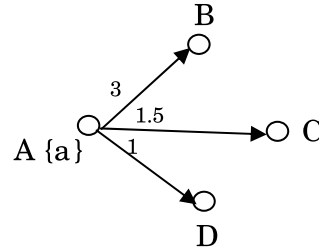
Theorem 1. The number of solutions for the ARVAV problem is given by

$$h_n(k) = \begin{cases} n^k - \sum_{i=1}^{n-1} C_n^i h_i(k) & (k \geq n), \\ 0 & (k < n), \end{cases} \tag{1}$$

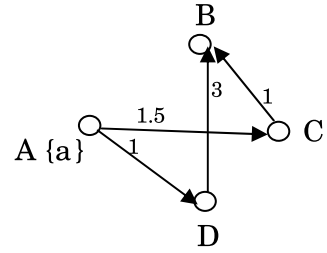
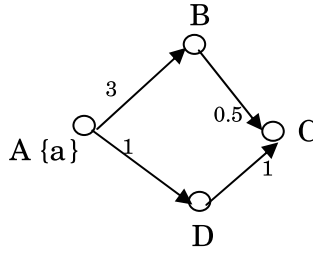
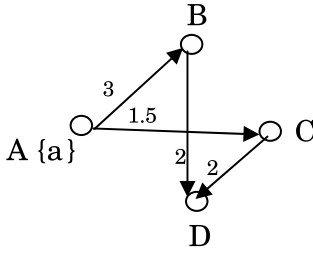
where $C_n^i = n! / (i!(n - i)!)$.



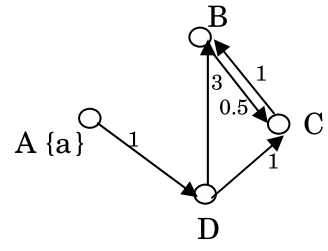
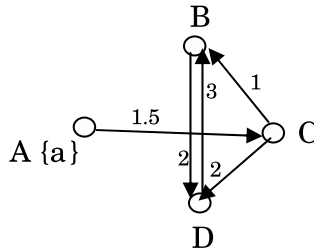
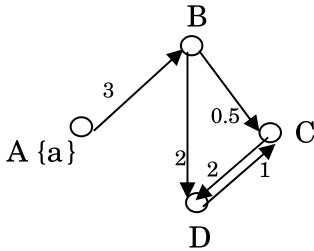
(a) One node to three. $F_1(4) = f_1(4) = C_3^1 F_1(3) + C_3^2 F_2(3) + C_3^3 F_3(3) = 9 + 6 + 1 = 16$ solutions.



(b) Three direct transfers. $C_3^3 F_3(3) = 1$ solution.



(c) Two direct transfers. $C_3^2 F_2(3) = 6$ solutions, where $F_2(3) = 2$ for each graph.



(d) One direct transfer. $C_3^1 F_1(3) = 9$ solutions.

Fig. 5. $F_1(4) = f_1(4)$.

Proof. Suppose $h_i(k)$ ($i = 1, 2, \dots, n - 1$) is already known; we prove that Eq. (1) is correct, given $i = n$. n verifiers are put in A set and k replicas unverified are put in B set.

(1) A total of n^k solutions satisfy that each verifier in B verifies at least one replica in A ; these solutions can be clustered into n groups:

- Group 1: n verifiers in A verify k replicas in B ;
- Group 2: $n - 1$ verifiers in A verify k replicas in B ;
- ...
- Group n : 1 verifier in A verifies k replicas.

That is $n^k = C_n^n h_n(k) + C_n^{n-1} h_{n-1}(k) + \dots + C_n^1 h_1(k)$, where $C_n^i h_i(k)$ is the number of solutions for Group i in which only i verifiers take part in the activity of verifying k replicas. Thus, $C_n^n h_n(k) = n^k - (C_n^{n-1} h_{n-1}(k) + \dots + C_n^1 h_1(k)) = n^k - \sum_{i=1}^{n-1} C_n^i h_i(k)$ ($n \leq k$).

(2) Based on Definition 2, we can easily deduce that

$$h_1(k) = 1, \tag{2}$$

since there is only one solution for the problem that redundant verifying happens to some k replicas. According to Eq. (1), $h_2(k) = 2^k - C_2^1 h_1(k) = 2^k - 2$; we can validate that this is correct, since only two solutions, “AA...A” and “BB...B”, do not satisfy the criteria of the ARVAV problem among 2^k solutions.

(3) Also, there are 0 solutions for the ARVAV problem when $n > k$; since it cannot satisfy criteria (2) in Definition 2, that is at least $n - k$ verifiers do not verify any of the k replicas.

This proves Theorem 1. \square

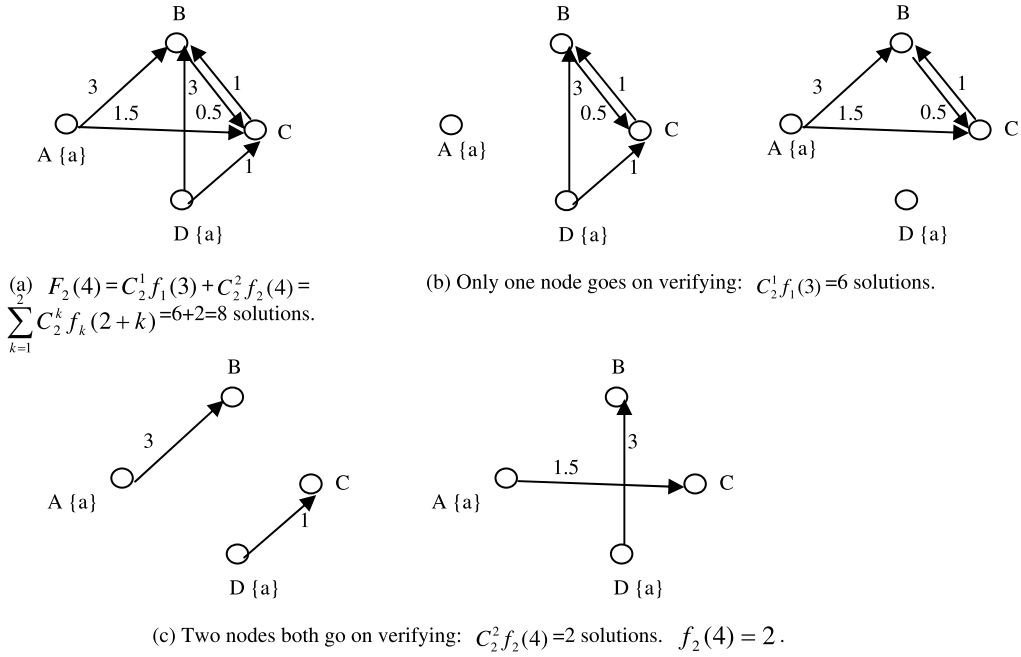


Fig. 6. $F_2(4)$ and $f_2(4)$.

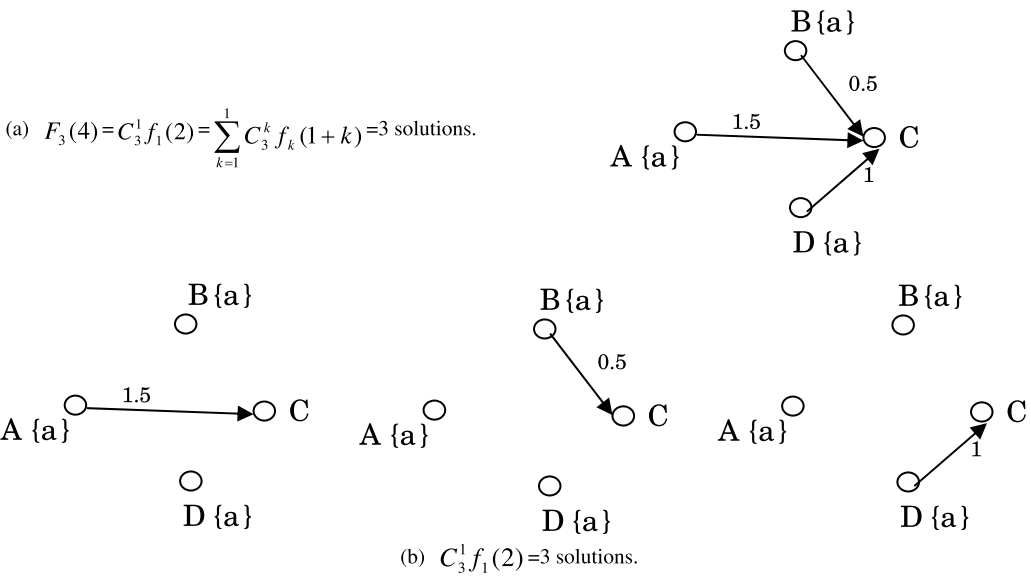


Fig. 7. $F_3(4)$.

Corollary 1.

$$h_n(k) = n^k - \sum_{j=1}^{n-1} (g(j) C_n^{n-j} (n-j)^k), \tag{3}$$

where

$$g(j) = \begin{cases} \sum_{t=1}^{j-1} (g(j-t) C_j^t) - 1 & (j > 1), \\ 1 & (j = 1). \end{cases} \tag{4}$$

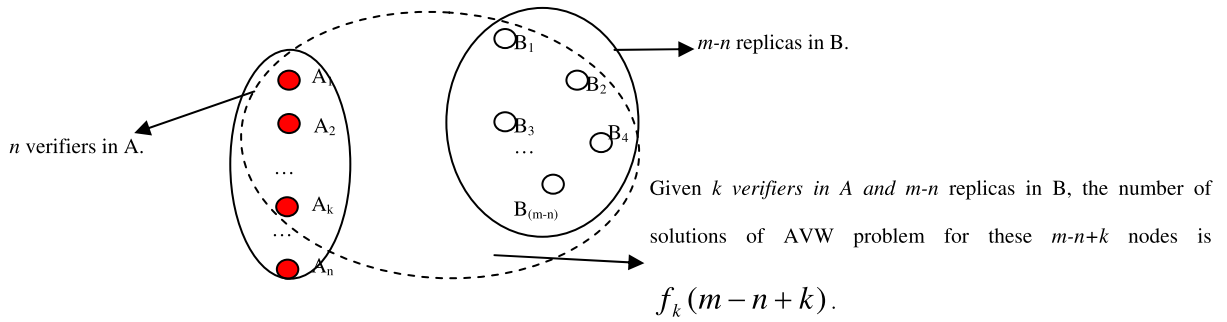


Fig. 8. ALOVW problem.

Proof. According to Theorem 1,

$$h_n(k) = n^k - \sum_{i=1}^{n-2} (C_n^i h_i(k)) - C_n^{n-1} h_{n-1}(k), \tag{5}$$

$$h_{n-1}(k) = (n-1)^k - \sum_{i=1}^{n-2} (C_{n-1}^i h_i(k)). \tag{6}$$

Then, let α_1 denote the coefficient of $h_{n-1}(k)$ of Eq. (5), that is $\alpha_1 = C_n^{n-1}$. Thus, Eq. (5) $-\alpha_1 \times$ Eq. (6), that is given by

$$h_n(k) = n^k - \alpha_1(n-1)^k + \sum_{i=1}^{n-3} ((C_{n-1}^i \alpha_1 - C_n^i) h_i(k)) + (C_{n-1}^{n-2} \alpha_1 - C_n^{n-2}) h_{n-2}(k). \tag{7}$$

Then, let α_2 denote the coefficient of $h_{n-2}(k)$ of Eq. (7), that is $\alpha_2 = C_{n-1}^{n-2} \alpha_1 - C_n^{n-2}$.

$$h_{n-2}(k) = (n-2)^k - \sum_{i=1}^{n-3} (C_{n-2}^i h_i(k)). \tag{8}$$

Eq. (7) $+\alpha_2 \times$ Eq. (8), that is given by

$$h_n(k) = n^k - \alpha_1(n-1)^k + \alpha_2(n-2)^k + \dots. \tag{9}$$

Without loss of generality, we denote the coefficient of $(n-j)^k$ by $\alpha_j = g(j)C_n^{n-j}$. And $g(1) = 1, g(2) = 1, g(3) = 5, g(4) = 29, g(5) = 209, \dots$, and $g(j) = \sum_{t=1}^{j-1} (g(j-t)C_j^t) - 1$.

This proves Corollary 1. \square

Theorem 2. The number of solutions for the ALOVW problem as shown in Fig. 8 is given by

$$F_n(m) = \begin{cases} \sum_{k=1}^{\min(m-n,n)} C_n^k f_k(m-n+k) & (m \neq n), \\ 1 & (m = n), \end{cases} \tag{10}$$

where $C_n^k = n! / (k!(n-k)!)$.

Proof. Simple examples in Section 3.2.1 have proved that Theorem 2 is correct when $m < 5$ and $n < 3$. For example, $f_1(4) = \sum_{k=1}^3 C_3^k F_k(3) h_1(k)$ and $h_1(k) = 1$. Now we prove that Theorem 2 is correct when $m \geq 5$ or $n \geq 3$.

- (1) According to Definition 4, the solutions of an ALOVW problem are clustered into $p = \min(m-n, n)$ groups: one verifier in A, two verifiers in A, ..., and p verifiers in A that participate in the activity of verifying the remaining $m-n$ replicas. For each Group k ($p \geq k \geq 1$), the number of solutions for choosing k A nodes from n A nodes is C_n^k . Suppose k verifiers in A have been selected to participate in the verifying work as shown in Fig. 5, there are $f_k(m-n+k)$ solutions for the problem that each of these k verifiers in A must verify one of the left $m-n$ B nodes, since this is an AVW problem (see Theorem 3). Thus, $F_n(m) = \sum_{k=1}^{\min(m-n,n)} C_n^k f_k(m-n+k)$ ($m \neq n$).
- (2) If $m = n$, no verifying is needed. In this case, we define $F_n(m) = 1$; Fig. 4(d) shows an example.

This proves Theorem 2. \square

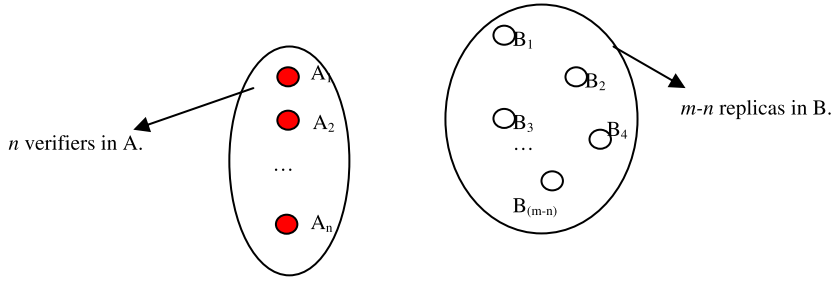


Fig. 9. AVW problem.

Theorem 3. The number of solutions for the AVW problem as shown in Fig. 9 is given by

$$f_n(m) = \begin{cases} \sum_{k=n}^{m-n} C_{m-n}^k F_k(m-n) h_n(k) & (n \leq m/2), \\ 0 & (\text{otherwise}), \end{cases} \quad (11)$$

where $C_{m-n}^k = (m-n)! / (k!(m-n-k)!)$.

Proof. Simple examples in Section 3.2.1 have proved that Theorem 3 is correct when $m < 5$ and $n < 2$; now we prove that Theorem 3 is correct when $m \geq 5$ or $n \geq 2$.

- (1) According to Definition 3, each of $\{A_1, A_2, \dots, A_n\}$ verifies at least one replica in B as shown in Fig. 9; suppose there are k ($m-n \geq k \geq n$) replicas in B which are verified by verifiers in A , then there are a total of C_{m-n}^k solutions. The number of solutions for the problem that n verifiers in A participate in the activity of verifying k replicas in B is $h_n(k)$. Given k replicas in B that have been verified, there are $F_k(m-n)$ solutions for the problem that these k verified replicas in B go on verifying the rest unverified replicas in B , since this is an ALOVW problem. Thus, $f_n(m) = \sum_{k=n}^{m-n} C_{m-n}^k F_k(m-n) h_n(k)$ ($n \leq m/2$).
- (2) If $n > m/2$, since n verifiers verify $m-n$ replicas, there are some verifiers in A that never have a chance to participate in the verifying work, and thus the AVW problem has no solution; therefore, the number of solutions is zero.

This proves Theorem 3. \square

Corollary 2. $f_1(m) = F_1(m)$.

Proof. According to Theorem 2, $F_1(m) = \sum_{k=1}^1 C_1^k f_k(m-1+k) = f_1(m)$. Examples are shown in Fig. 4(a) and (b). \square

To resolve the FOSTCBDG problem on a CBDG with exactly m vertices, we need to choose an optimal one from $F_1(m)$ possible spanning trees; this is not efficient and may be just for the worst case. Algorithms that aim to resolve the FOSTCBDG problem efficiently must provide good schemes to prune the possible spanning trees.

5. Efficient approximate algorithm for tackling the FOSTCBDG problem

In this section, we provide a novel Replace the Smallest Bandwidth Edge (RSBE) algorithm to approximately resolve the FOSTCBDG problem.

- **Step 1.** Build an initial flat tree, which is composed of edges from the verifier vertex A to every replica vertex B_i .
- **Step 2.** For the edge $\langle A, B_s \rangle$ with smallest bandwidth, w_{AB_s} in the tree, the B_s is verified at time $t_s = 1/w_{AB_s}$. Try to replace it by an edge $\langle B_k, B_s \rangle$, where $k \neq s$ and $t_s > t_{ks}$, where t_{ks} is the earliest time that the B_s is verified from A through any B_k to B_s .
- **Step 3.** Repeat replace the smallest bandwidth on unused edges until no replace happens.

An example of the RSBE algorithm is shown in Fig. 10.

6. Performance evaluation

In this section, we evaluated the performance of our proposed scheme by using the bandwidths in a real world clouds network.

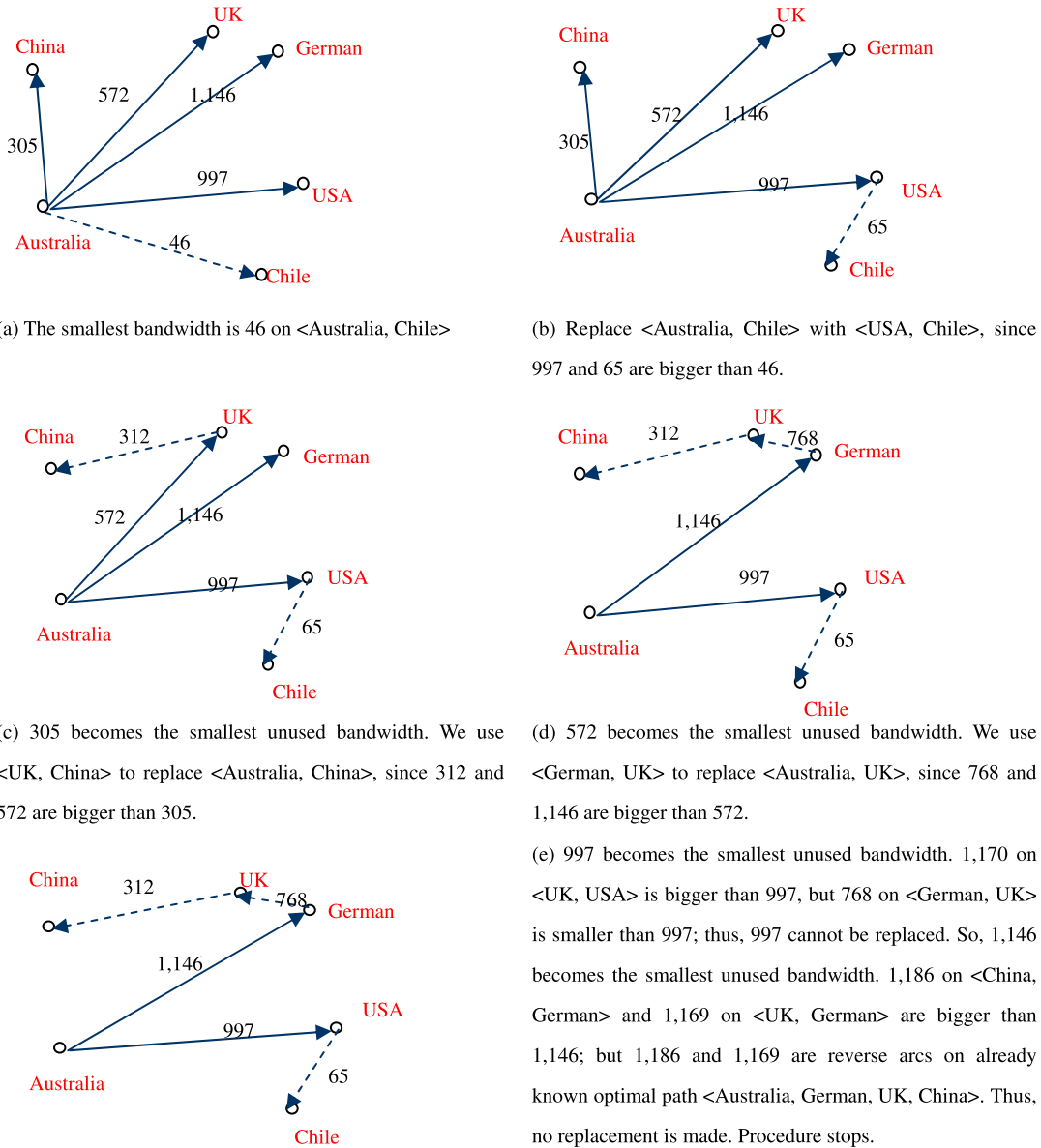


Fig. 10. Process of RSBE.

6.1. Experimental setup

We briefly present the experimental setup. We adopt a CBDG as shown in Fig. 11, where bandwidths are marked on links according to Table 1 (in Section 2). We use the RSBE algorithm to generate approximate optimal spanning trees. There are three types of transmitting data: *Challenge*, *Proof* and *ID/Copy* as shown in Fig. 2(b). Compared to communication time, computation time in the DMRDPC is small. Also, the data sizes of *Challenge*, *Proof* and *ID* are far smaller than the data size of *Copy* (e.g. replica data). We assume the volume of a replica is 1 GB in this experiment. Since transmitting large replica data consumes a lot of time but replica data transmitting happens randomly only when the verifying fails, we evaluate the performance of the DMRDPC scheme by the communication time in the worst case, in which all the replicas fail the challenge.

6.2. Efficiency

In this subsection, we discuss the efficiency of the exact algorithm and the approximate algorithm (e.g. RSBE) for the FOSTCBDG problem. In an exact algorithm for the FOSTCBDG problem, given a verifier on an Australian (Au) cloud, which will verify the replicas on the other 5 clouds in China (Cn), Chile (Cl), UK, German (Ge) and USA. According to theories in

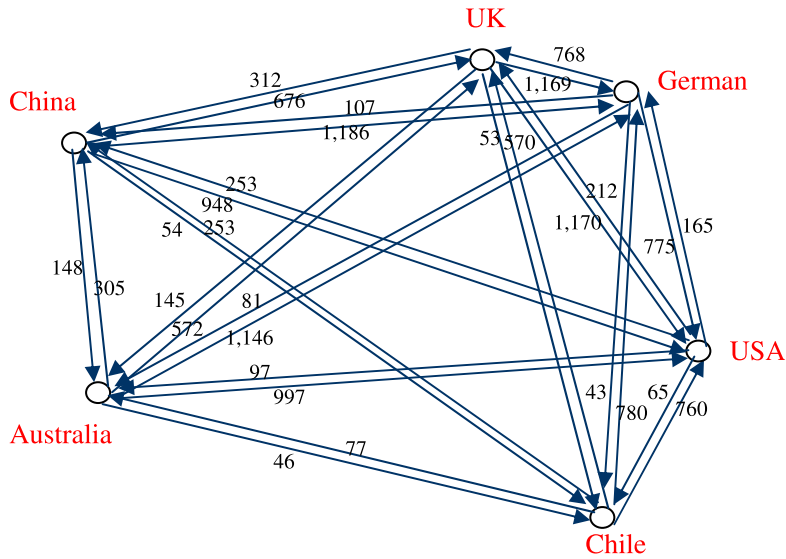


Fig. 11. An CBDG with 6 vertex, $F_1(6) = 1226$.

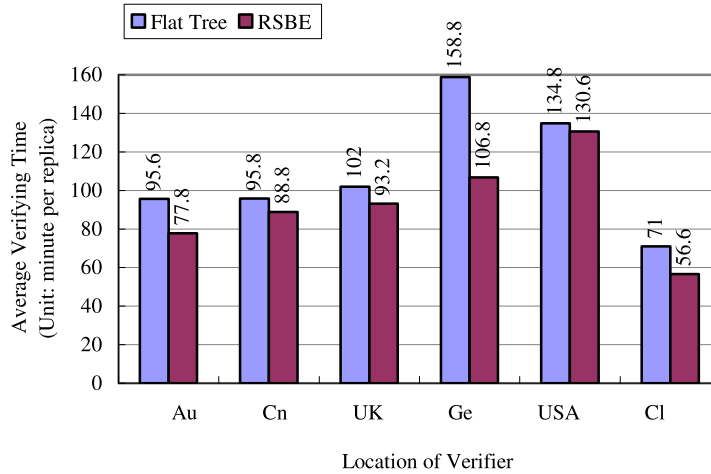


Fig. 12. Effectiveness.

Section 3, $F_1(6) = \sum_{k=1}^5 C_5^k F_k(5) = C_5^1 F_1(5) + C_5^2 F_2(5) + C_5^3 F_3(5) + C_5^4 F_4(5) + C_5^5 F_5(5)$, where $F_1(5) = C_4^1 F_1(4) + C_4^2 F_4(2) + C_4^3 F_3(4) + C_4^4 F_4(4) = 125$, since $F_1(4) = 16$, $F_2(4) = 8$, $F_3(4) = 3$ are given by Fig. 5, Fig. 6 and Fig. 7, and according to Theorem 2, $F_4(4) = 1$. According to Theorems 1, 2 and 3, $F_2(5) = C_2^1 f_1(4) + C_2^2 f_2(5) = C_2^1 F_1(4) + C_2^2 (C_3^2 F_2(3) h_2(2) + C_3^3 F_3(3) h_2(3)) = 50$. In the same way, we achieve $F_3(5) = 8$, $F_4(5) = 4$ and $F_5(5) = 1$. Thus, we need to check $F_1(6) = 1226$ spanning trees in Fig. 11 to find the optimal one. So, we can see the RSBE algorithm shown in Fig. 11 is more efficient.

6.3. Effectiveness

We have conducted six experiments to evaluate the effectiveness of our proposed method; in each experiment, one cloud is chosen as the verifier to start data possession checking of the other 5 clouds. The results are shown in Fig. 12. Fig. 12 shows the time efficiency achieved by the RSBE-based method compared to that achieved by the flat-tree-based method. Fig. 12 shows that the RSBE-based method reduces the average verifying time per replica in all of the six experiments.

7. Conclusions

In this paper, we provide a novel efficient Distributed Multiple Replicas Data Possession Checking (DMRDPC) scheme to validate the availability and data integrity in the cloud computing environment. In the DMRDPC scheme, the problem of Finding an Optimal Spanning Tree in a Complete Bidirectional Directed Graph (the FOSTCBDG problem) is vital, since the op-

timal spanning tree can be used to improve communication efficiency by optimizing the partial order of scheduling multiple replicas data possession checking. Particularly, on the Internet, data routing often aims to minimize the number of hops; but in a clouds environment, multi-hop among clouds may be more efficient than a single hop. With the increasing number of clouds, the FOSTCBDG problem is becoming more and more critical to improve the performance of cloud computing. Thus, we provide basic theories for the FOSTCBDG problem. We also propose an efficient Replace the Smallest Bandwidth Edge (RSBE) algorithm to approximately resolve the FOSTCBDG problem. The effectiveness of our proposed DMRDPC is validated by an experimental study, where bandwidths of a CBDG are simulated by checking the download speeds of several Google websites from the non-Google clouds at multiple locations around the world.

Acknowledgments

This work was partially supported by Australian Research Council (ARC) Linkage Project: LP0882957.

References

- [1] M. Armbrust, A. Fox, R. Griffith, A.D. Joseph, R. Katz, A. Konwinski, et al., Above the Clouds: A Berkeley View of Cloud Computing, EECS Department, University of California, Berkeley, 2009.
- [2] I. Foster, Y. Zhao, I. Raicu, S. Lu, Cloud computing and grid computing 360-degree compared, in: Grid Computing Environments Workshop (GCE'08), 2008, pp. 1–10.
- [3] J. Delfos, T. Tan, B. Veenendaal, Design of a web-based LBS framework addressing usability, cost, and implementation constraints, World Wide Web 13 (4) (2010).
- [4] S. Asadi, X. Zhou, G. Yang, Using local popularity of web resources for geo-ranking of search engine results, World Wide Web 12 (2009) 149–170.
- [5] S. Agarwal, J. Lorch, Matchmaking for online games and other latency-sensitive P2P systems, in: SIGCOMM, 2009.
- [6] A. Adaya, J. Dunagan, A. Wolman, Centrifuge: integrating lease management and partitioning for cloud services, in: NSDI, 2010.
- [7] J. Leskovec, E. Horvitz, Planetary-scale views on an instant-messaging network, in: Proceedings of the 16th International Conference on World Wide Web (WWW'08), 2008.
- [8] J. Broberg, R. Buyya, Z. Tari, MetaCDN: Harnessing 'storage clouds' for high performance content delivery, J. Network Comput. Appl. 32 (5) (2009) 1012–1022.
- [9] P. Boldi, B. Codenotti, M. Santini, S. Vigna, Ubcrawler: a scalable fully distributed web crawler, in: Software: Practice & Experience, 2004.
- [10] CloudFront, <http://aws.amazon.com/cloudfront/>.
- [11] J. Gray, Distributed computing economics, Queue 6 (3) (2008) 63–68.
- [12] J. Gray, D. Patterson, A conversation with Jim Gray, ACM Queue 1 (4) (2003) 8–17.
- [13] A.E.F. Clementi, A. Monti, F. Pasquale, R. Silvestri, Broadcasting in dynamic radio networks, J. Comput. System Sci. 75 (2009) 213–230.
- [14] S. Agarwal, J. Dunagan, N. Jain, S. Saroiu, A. Wolman, Volley: automated data placement for geo-distributed cloud services, in: Proc. of 7th USENIX Symposium on Networked Systems Design and Implementation (NSDI), San Jose, CA, 2010.
- [15] Google Data Center FAQ, <http://www.datacenterknowledge.com/archives/2008/03/27/google-data-center-faq/>.
- [16] Checks website performance from multiple locations around the world, <http://internetsupervision.com/scripts/urlcheck/check.aspx>.
- [17] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, D. Song, Provable data possession at untrusted stores, in: CCS'07, Alexandria, Virginia, USA, 2007.
- [18] D.L.G. Filho, P.S.L.M. Barreto, Demonstrating data possession and uncheatable data transfer, IACR ePrint archive, Report 2006/150, 2006, <http://eprint.iacr.org/2006/150>.
- [19] L. Chen, G. Guo, An efficient remote data possession checking in cloud storage, Int. J. Digital Content Technol. Appl. 5 (4) (2011) 43–50.
- [20] Z. Hao, N. Yu, A multiple-replica remote data possession checking protocol with public verifiability, in: Proc. of 2010 Second International Symposium on Data, Privacy, and E-Commerce, 2010.
- [21] T. Chiba, T. Endo, S. Matsuoka, High-performance MPI broadcast algorithm for grid environments utilizing multi-lane NICs, in: Proc. of 7th IEEE International Symposium on Cluster Computing and the Grid (CCGrid'07), 2007, pp. 487–494.
- [22] K. Takahashi, H. Saito, T. Shibata, K. Taura, A stable broadcast algorithm, in: Proc. of 8th IEEE International Symposium on Cluster Computing and the Grid (CCGrid'08), 2008, pp. 392–400.
- [23] T. Chiba, M. den Burger, T. Kielmann, S. Matsuoka, Dynamic load-balanced multicast for data-intensive applications on clouds, in: Proc. of 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing (CCGrid'10), 2010, pp. 5–14.
- [24] R. Perlman, An algorithm for distributed computation of a spanning tree in an extended LAN, ACM SIGCOMM Comput. Commun. Rev. 15 (4) (1985) 44–53.
- [25] Y.S. Myung, C.H. Lee, D. Tcha, On the generalized minimum spanning tree problem, Networks 26 (1995) 231–241.
- [26] C.P. Petrica, New models of the generalized minimum spanning tree problem, J. Math. Model. Algorithms 3 (2004) 153–166.