Disaster-Aware Datacenter Placement and Dynamic Content Management in Cloud Networks

Sifat Ferdousi, Ferhat Dikbiyik, M. Farhan Habib, Massimo Tornatore, and Biswanath Mukherjee

I. Introduction

With the rise in demand for cloud services, huge amounts of digital content are being created and shared all the time over the network. It is crucial that the network supporting such services is resilient to data loss or service disruptions, hence making cloud network design an important problem. Datacenters are mega-centers of computing and storage resources, and a network of datacenters is called a cloud network [1]. Cloud services include, among other applications, popular Web applications, distributed grid computing, and some mission-critical applications with high bandwidth requirements [1,2]. The contents and services are replicated over multiple datacenters, so that a user request can be served by any datacenter that hosts the required content. Such services require network infrastructures with high capacity, high availability, low latency, robustness, etc., to serve the rising volume of traffic, and optical networks are well suited to meet these requirements [3].

Considering the role of cloud services today, any disruption of content/service is a major concern. For example, in 2011 and 2012, a major cloud provider's cloud crash took down a number of sites for days, causing permanent damage in many customers' data; in 2012, a major airline's operation was disrupted from a carrier's two fiber-optic cuts; and a survey shows that, in 2011, 19% of the data loss experienced by businesses was from the cloud [4–6]. Besides natural disasters such as earthquakes and tornadoes, human-made disasters pose a major threat to cloud networks. National security agencies report rising alarms about the increasing risk of terrorist activities, such as weapons of mass destruction (WMD) attacks (e.g., nuclear, chemical, EMP, biological) [7].

Network failures due to disasters are correlated; i.e., secondary or cascading failures resulting from the initial failures (such as aftershocks and power outages [8]) cause further damage to network resources. For instance, due to the Japanese earthquake and tsunami of 2011, power outages affected 1500 telecom buildings during the main shock in March and another 700 telecom buildings during the aftershock in April [7]. Hence, the vulnerability of the network and content to multiple correlated, cascading, and colocated failures is an important problem to study. In this work, we investigate disaster-aware datacenter placement and dynamic content management through proactive and reactive approaches that can help providers to design a disaster-resilient cloud network.

Understanding the nature of disaster failures in telecom networks is important for designing resilient networks. Risk/hazard maps of disasters can be obtained and matched with the physical topology of a network to determine its possible risky zones [9]; e.g., the US West Coast is

Manuscript received September 18, 2014; revised May 13, 2015; accepted May 26, 2015; published June 30, 2015 (Doc. ID 223299).

S. Ferdousi (e-mail: sferdousi@ucdavis.edu), F. Dikbiyik, M. Farhan Habib, M. Tornatore, and B. Mukherjee are with the University of California, Davis 95616, USA.

F. Dikbiyik is also with Sakarya University, Sakarya, Turkey.

M. Tornatore is also with Politecnico di Milano, Milan, Italy.

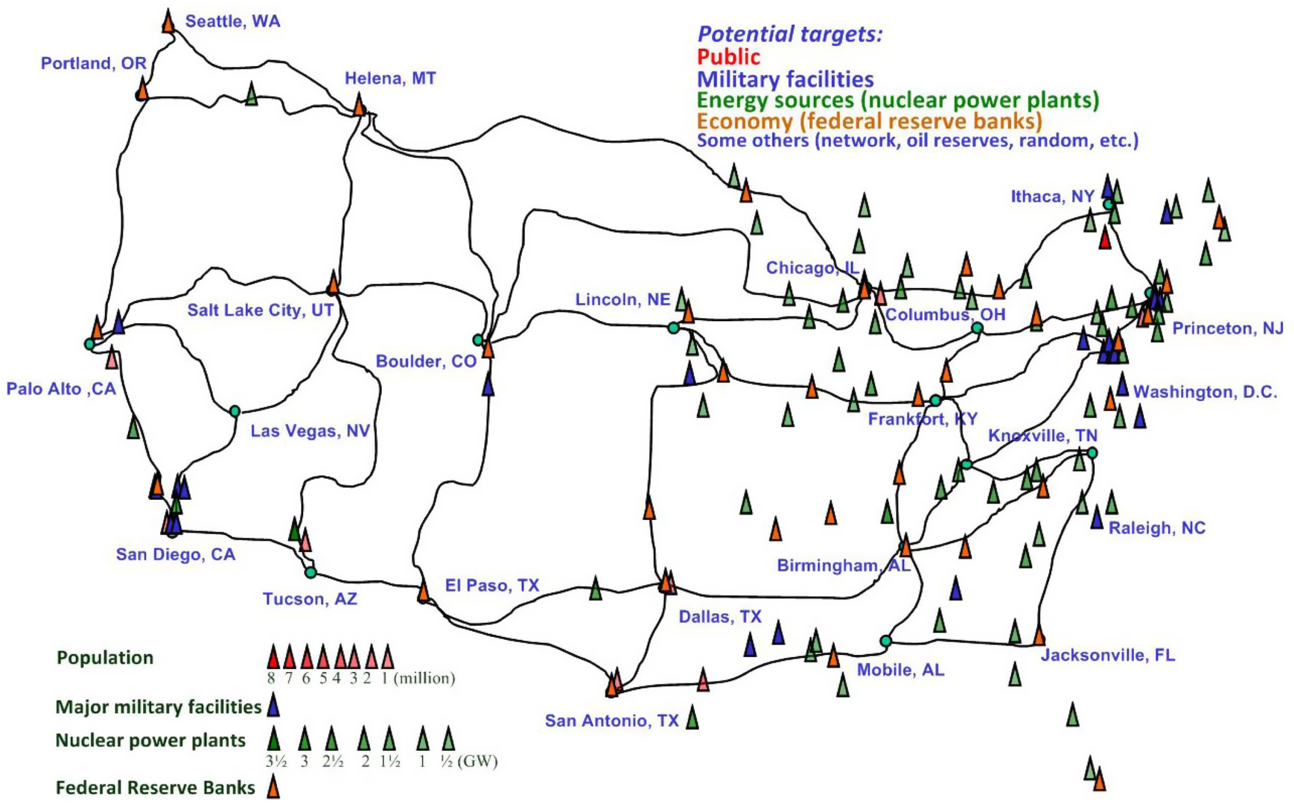

Fig. 1. Risk map for WMD attacks on US-wide topology.

more vulnerable to earthquakes, whereas the East Coast is more vulnerable to hurricanes. Terrorist attacks generally target populated cities and important government, military, or resource facilities such as power grids and datacenters (DCs) [7]. As an example, we acquired information on possible locations of different major facilities from various public sources, and generated a WMD risk map on a US-wide topology by considering possible attacks and correlations between them (see Fig. 1). Such maps can help to develop a disaster-aware network design.

Today, major cloud service providers select DC locations based on energy availability, developable land, available workforce, user demands, etc. [10–12]. In addition, it is imperative to consider the disaster map of the physical locations. Studies on DC placement problems consider minimizing network cost, energy, latency, etc. [13,14], and there have been studies on content or replica placement in a DC network with the objective of reducing network cost and latency [15,16]. But addressing DC *and* content placement for a disaster-resilient network has not been studied in detail before. For instance, even though major cloud providers place their DCs in distinct physical locations to isolate failures from each other, they still experience data loss at times of disaster events [4].

We propose a risk analysis in a given network with a set of candidate DC locations to estimate how much, in terms of cost or penalty, a network operator may lose probabilistically in case of a possible disaster and define it as *expected content loss* or *risk* (a similar risk metric is defined in [9]). Generally speaking, having more replicas of content means greater availability and reliability, and less vulnerability. But we aim to emphasize the placement of the replicas, i.e., the locations of the hosting DCs, to ensure disaster resiliency. In the case of a disaster or an attack, we want to ensure that a sufficient number of replicas of content survive during/after a disaster; i.e., all (or most) replicas are placed in disaster-free zones; otherwise, the content may become vulnerable to loss. For example, having two surviving replicas of content after a disaster is more desirable than having one surviving replica because the last replica may be lost due to a post-disaster correlated or uncorrelated failure. To ensure such resiliency, it may not suffice to have a large number of replicas, if all (or most) of them are hosted in DCs located in disaster-vulnerable regions. Rather, a reasonable number of replicas strategically placed in geographically distributed and disaster-resilient DC locations can enhance the possibility of surviving replicas and hence reduce risk in the case of a disaster. Our risk analysis can be used to identify suitable locations in the network for DC placement and, hence, the initial placement of the contents. Such analysis can also be applied to any existing set of DCs to help operators choose disaster-resilient DC(s) from various DC sites to host more important or critical contents.

Since network settings and disaster scenarios are dynamic in nature, one-time placement of the contents might not ensure disaster resiliency. Dynamic content management is required to make the content placement adaptive to any network and disaster alert updates to reduce *risk* at all times. Using our method, any existing content placement can be analyzed, at any given time, to understand the possibility of contents being vulnerable to loss due to disasters and adapt accordingly. This study will help cloud operators to design and prepare their infrastructure against disasters.

Our contribution to the DC and content placement problem is twofold: first, we formulate a disaster-aware DC and content placement approach that statically assigns the DCs and contents to locations that are least vulnerable to disasters, and hence *risk* (expected content loss) is minimized. As a part of this initial placement design, we also adapt a budget constraint (based on storage cost in DCs) to ensure that our disaster-aware approach is not too costly. We then propose a disaster-aware dynamic content-management (DADCM) algorithm that dynamically adjusts the placement of content replicas over a fixed set of DCs depending on the changing network and disaster scenarios such that *risk* of the network is reduced. We compare our disaster-aware approaches with disaster-unaware approaches for the DC and content placement problem. Our results show that, for the typical US-wide network scenario studied in this work, disaster-aware placement provides about 45% risk reduction in terms of expected content loss due to a given set of disasters compared with a disaster-unaware approach, which minimizes the average path cost. We also present a content-management cost analysis of our approaches based on real-world cloud pricing.

The rest of the study is organized as follows. Section II presents the static disaster-aware datacenter and content placement problem, and its mathematical model. In Section III, we provide the algorithm for DADCM. In Section IV, we discuss a content-management cost model. Section V provides illustrative results showing the effectiveness of our approaches. Finally, Section VI concludes the study.

II. Disaster-Aware Datacenter and Content Placement

We propose a disaster-aware DC and content placement design based on risk analysis in terms of *expected content loss*. We consider two (nonexclusive) contributing factors to the expected content loss:

- **Expected loss due to unavailability**: Content is vulnerable to loss if a DC hosting the content is unavailable due to a disaster; i.e., if a DC is damaged and is unavailable due to a disaster, then its content also becomes unavailable.
- **Expected loss due to unreachability**: Content is vulnerable to loss if a DC hosting the content is unreachable from a user node due to a disaster. Thus, if the routes connecting the users to the DC are damaged due to a disaster, then the content becomes unreachable from the users. Here, the reachability of content is similar to the concept of content connectivity defined in [7] for the virtual network mapping problem.

Considering the access latency requirements of users, we analyze the expected content loss due to unreachability with a reasonable finite number of link-disjoint routes between the user node and a DC. Thus, DCs are placed not too far away from users while finding the optimal locations for placing a DC, so that a user can obtain content from a DC via one of the link-disjoint k-shortest paths between the user and the DC.

We introduce, for each content c, an importance factor, α_c, which represents the importance or value of the content. Usually, content is associated with a popularity metric, but for mission-critical (e.g., military) applications, the importance is a more appropriate choice. Popular content may not be the most important content and vice versa. By using this factor, we ensure the survivability of the most important content first. The higher the importance of content, the higher the penalty due to loss of that content.

Since placing a DC requires careful planning, huge investment, and resources, the total number of DCs in the network is limited. For replica management, we constrain the maximum number of replicas per content. To maintain consistency among different replicas hosted at different DCs, the replicas need to be synchronized or updated to the same state periodically. We do not consider full replication where all content is replicated at every DC because this may create more background traffic between the DCs due to synchronization. (We consider an uncapacitated network and hence do not consider any bandwidth constraint for the links.) On the other hand, to ensure content availability against a DC failure, content should be replicated in at least two or a minimum number, r, DCs. It follows that the decision of the actual number of replicas of content c, R_c, might depend on many factors. Therefore, we make R_c depend on both α_c and demands (in terms of number of user requests), η_c, while also guaranteeing that the value of R_c stays between r and a maximum value R_{max} (assumed to be provided by the service provider, and bounded by the total number of DCs in the network). In formulas, R_c is calculated as follows:

$$\rho_c = \lceil \gamma_1 \cdot \eta_c + \gamma_2 \cdot \alpha_c \rceil, \qquad R_c = max\{r, min\{\rho_c, R_{max}\}\}.$$

Here ρ_c denotes a weighted sum of η_c and α_c, and the parameters γ_1 and γ_2 will be chosen independently based on the service provider's priorities. For example, to prioritize more popular services, service providers may put more weight on η_c, whereas for more critical applications, higher weight on α_c would be more appropriate.

We analyze the network with a probabilistic disaster model. Network equipment in a disaster zone [represented as a shared-risk group (SRG), i.e., a set of links and nodes that fail simultaneously] fails with some probability, which depends on the dimensions of equipment (e.g., link length), its distance from the disaster's epicenter, the type of disaster, etc. [17,18]. For our case study, we consider that the closer a network element is to the epicenter of a disaster, the higher the failure/damage probability. We consider that one disaster occurs at a time in the network.

The probability of a DC or a path connecting a user to a DC being damaged due to a disaster can be obtained as input (from given disaster and network data) based on which the probability of *unavailability* and *unreachability* can be determined. Let P_d^m be the probability that DC d is damaged by disaster m (hence, the content at d is *unavailable*), and let Q_{kds}^m be the probability that path k connecting DC d and user s is damaged by disaster m. Then, the probability that all the k paths from user s to DC d are damaged simultaneously by disaster m (hence, the content at d is *unreachable*) is $\prod_{k\in L_{ds}} Q_{kds}^m$ (here, we consider the path failures to be independent). When a DC becomes unavailable, all of its adjacent links also become unavailable, and any path via these links will be unreachable. Given the damage/failure of DC d, the probability that the links adjacent to DC d are damaged is $\prod_{k\in L_{ds}^m} Q_{kds}^m$, where $L_{ds}^m \supseteq L_{ds}$ is the set of paths whose links adjacent to DC d are within the zone of disaster m. For a set of paths, L_{ds}, between DC d and user s, the probability P_{ds}^m that DC d is damaged (unavailable *or* unreachable from user s) due to disaster m can be computed as

$$P_{ds}^m = \left\{P_d^m + \prod_{k\in L_{ds}} Q_{kds}^m\right\} - P_d^m \cdot \prod_{k\in L_{ds}^m} Q_{kds}^m, L_{ds}^m \supseteq L_{ds}.$$

Risk is defined in terms of *expected content loss*, which depends on the damage probability, P_{ds}^m. Given that content c is lost (either unavailable or unreachable) and letting J_{dcs}^m indicate that c at DC d is lost from user s due to disaster m, the *expected loss* of content c can be derived based on α_c, the importance factor of c, as follows:

$$expected\_loss_c = (P_{ds}^m \cdot J_{dcs}^m) \cdot \alpha_c.$$

Thus, we can say that *risk*, defined as expected content loss, is in the units of α_c. If, for example, α_c of content c is some monetary value to lose a replica, then the *risk* can be expressed in the expected amount of money to lose replica(s) of c due to disaster m. We formulate the *risk* of the network as the total *expected loss* incurred for all contents, all DCs, all users, and all disaster events:

$$Risk = \sum_{m\in M}\sum_{c\in C}\sum_{s\in S}\sum_{d\in D}(expected\_loss_c).$$

Please note that the more content is replicated in *disaster zones*, the larger the expected loss is going to be, because J_{dcs}^m will be 1 for the replicas that are affected by disaster m. Irrespective of the number of replicas of content, our model focuses on the *actual placement* of the replicas. We argue that content c_1 with two replicas, *all* placed in *disaster-free* locations, incurs less risk than content c_2 with four replicas, *all* placed in disaster zones. Similarly, content c_3 with four replicas, *all* placed in disaster zones, incurs more risk than content c_4 with two replicas, *all* placed in disaster zones. Even though c_3 has more replicas, there is a greater penalty for losing more replicas compared to c_4. This is done to enforce that all (or most) of the replicas of content are placed in low-risk or no-risk DCs. The objective function does not distinguish between two content replicas and four content replicas, *all* placed in *disaster-free* locations, since both have zero risk value (J_{dcs}^m will be 0 for the replicas unaffected by disaster m).

Given a network topology, a set of user nodes, a set of candidate DC locations, a set of contents (with corresponding demands, importance factors, and maximum number of replicas), a set of precomputed link-disjoint k-shortest paths between the nodes, and a set of possible disaster events with given failure/damage probabilities, we need to assign DCs and contents to candidate DC locations such that the *risk* (i.e., expected content loss) of the network is minimized. We formulate the problem as an integer linear program (ILP) with the objective of risk minimization as follows:

A. Input Parameters

- $G(V, E)$: Physical topology of the network; V is the set of nodes, and E is the set of links.
- $D \in V$: Set of candidate DC locations.
- $S \in V$: Set of user (requesting) nodes.
- M: Set of disaster events.
- L_{ds}: Set of paths between nodes, $d \in D$ and $s \in S$.
- C: Set of contents.
- β: Number of DCs to be placed, $\beta > 0$.
- η_c: Demand in number of user requests for content $c \in C$.
- α_c: Importance metric of content $c \in C$.
- R_c: Number of replicas per content $c \in C$ calculated based on α_c and η_c, $r \leq R_c \leq R_{max}$, $R_{max} < \beta$.
- $X_d^m \in \{0, 1\}$: 1 if node $d \in D$ is damaged by disaster $m \in M$.
- $Y_{kds}^m \in \{0, 1\}$: 1 if path $k \in L_{ds}$ is damaged by disaster $m \in M$.
- P_d^m: Probability of damage of node $d \in D$ by disaster $m \in M$.
- Q_{kds}^m: Probability that path $k \in L_{sd}$ to node $d \in D$ from node $s \in S$ is unreachable due to disaster $m \in M$.
- P_{ds}^m: Probability of unavailability or unreachability of node $d \in D$ from node $s \in S$ due to disaster $m \in M$.
- $G_{cs} \in \{0, 1\}$: 1 if node $s \in S$ requests content $c \in C$.

B. Variables

- $T_{dc} \in \{0, 1\}$: 1 if content $c \in C$ is available at node $d \in D$.
- $B_d \in \{0, 1\}$: 1 if there is a DC at node $d \in D$.
- $A_{dc}^m \in \{0, 1\}$: 1 if content $c \in C$ is unavailable at node $d \in D$ due to disaster $m \in M$.
- $H_{dcs}^m \in \{0, 1\}$: 1 if node $d \in D$ containing content $c \in C$ is unreachable from node $s \in S$ due to disaster $m \in M$.
- $J_{dcs}^m \in \{0, 1\}$: 1 if disaster $m \in M$ occurs and content $c \in C$ in node $d \in D$ is lost from node $s \in S$ due to unavailability or unreachability caused by disaster $m \in M$.

C. Problem Formulation

$$Minimize: \sum_{m\in M}\sum_{c\in C}\sum_{s\in S}\sum_{d\in D} \alpha_c \cdot J_{dcs}^m \cdot P_{ds}^m \qquad (1)$$

subject to

$$J_{dcs}^{m} \geq \frac{A_{dc}^{m} + H_{dcs}^{m}}{2} \quad \forall s \in S, \ \forall d \in D, \ \forall c \in C, \ \forall m \in M, \quad (2)$$

$$J_{dcs}^{m} \leq A_{dc}^{m} + H_{dcs}^{m} \quad \forall s \in S, \ \forall d \in D, \ \forall c \in C, \ \forall m \in M, \quad (3)$$

$$A_{dc}^{m} = X_{d}^{m} \cdot T_{dc} \quad \forall s \in S, \quad \forall d \in D, \quad \forall m \in M, \quad (4)$$

$$H_{dcs}^{m} = \prod_{k \in L_{sd}} Y_{kds}^{m} \cdot T_{dc} \quad \forall c \in C, \quad \forall m \in M, \quad (5)$$

$$B_d \geq \frac{\sum_{c \in C} T_{dc}}{Z} \quad \forall d \in D, \quad (6)$$

$$B_d \leq \sum_{c \in C} T_{dc} \quad \forall d \in D, \quad (7)$$

$$\sum_{d \in D} B_d \leq \beta, \quad (8)$$

$$\sum_{d \in D} T_{dc} \leq R_c \quad \forall c \in C, \quad (9)$$

$$\sum_{d \in D} T_{dc} \geq r \quad \forall c \in C, \quad (10)$$

$$J_{dcs}^{m} \leq G_{cs} \quad \forall c \in C, \quad \forall d \in D, \quad \forall s \in S, \quad \forall m \in M. \quad (11)$$

The objective function in Eq. (1) minimizes the overall *risk* of the network in terms of expected content loss due to unavailability and unreachability. The expected loss of content is adjusted according to its importance, α_c. Equations (2) and (3) indicate that content can be expected to be lost due to unavailability or unreachability or both. These equations are the linearization of the logical *OR* of variables on the right-hand side of the equations. Equations (4) and (5) determine the expected content loss due to unavailability and unreachability, respectively. The main decision variable in this ILP is T_{dc}, and other variables are derived based on this. Equations (6) and (7), which are linearizations of the logical *OR* similar to Eqs. (2) and (3), indicate whether node d hosts a DC (here, $Z \geq |C|$). Equation (8) bounds the total number of DCs in the network. The solution selects β best DCs (in terms of risk) from a set of candidate DCs. Equations (9) and (10) bound the number of replicas of content c. To ensure content availability, there must be at least r replicas of content c, whose upper bound is R_c. Equation (11) indicates that a user node s obtains content c if s has demand for c; i.e., we consider unavailability and unreachability of c from s only when s requests for c.

To evaluate the effectiveness of our disaster-aware approach, we study a disaster-unaware minPathCost DC and content placement design with the objective to minimize the average path cost (i.e., average distance from a user node to the nearest replica of content) while placing the DCs and the contents. The path distance is proportional to access latency, which is important to minimize in DC placement. Given a network topology, a set of user nodes, a set of candidate DC locations, and a set of contents (with corresponding demands and maximum number of replicas), DCs and contents are assigned to the candidate locations such that the average path cost in the network is minimized. This approach is formulated based on a simpler model of the disaster-aware approach without any risk analysis. Since there is no notion of disaster/risk in this approach, only Eqs. (6)–(10) are adapted from the disaster-aware ILP. The objective function is

$$Minimize{:}Maximum\left[\sum_{s \in S}\sum_{d \in D}(U_{sd} \cdot V_{sd})\right],$$

where U_{sd} is the path cost between user s and DC d, and V_{sd} indicates whether user s is served from candidate DC d (a user is served from one DC). The content replication and DC placement constraints are similar to the disaster-aware approach; additional constraints are formulated to express path existence and content placement. Our disaster-aware approach only minimizes the risk in the network and does not aim to minimize the network cost in terms of resources, for which path cost can be a trade-off. But since only a limited set of paths is used for DC reachability, it places DCs within a set of shortest paths from the user nodes.

Content incurs a storage cost; i.e., the monetary cost of storing it in a DC and the charge for storage capacity is based on the average amount of data stored (in GB) (details are presented in the cost analysis in Section IV). In practice, different DCs have different pricings for storage and bandwidth usage (see Section IV), and, hence, during placing contents in DCs, it is important to consider cost. It should be noted that, since our design solves only for the initial content placement, we consider only the *storage cost* in DCs. We propose an adaptation of our disaster-aware DC and content placement design by incorporating a storage cost constraint—a *budget* in the ILP. The objective is still to minimize the overall risk for all contents in the network, but now the placement will be bounded by a budget. As a benchmark for our budget constraint, we study a disaster-unaware minStorageCost DC and content placement approach with the objective of minimizing the storage cost in DCs.

The disaster-unaware minStorageCost DC and content placement problem is formulated based on a simpler version of our proposed disaster-aware approach (without any risk analysis) with the objective to minimize the storage cost in DCs. Given a network topology, a set of user nodes, a set of candidate DC locations with corresponding per GB storage cost, and a set of contents (with corresponding demands, maximum number of replicas, and size in GB), DCs and contents are assigned to candidate locations such that the overall storage cost in DCs is minimized. Since there is no notion of disaster/risk in this approach, only Eqs. (6)–(10) are adapted from the disaster-aware ILP. The objective function is

$$Minimize \sum_{c\in C}\sum_{d\in D} T_{dc}(B_c \cdot St_d),$$

where T_{dc} (as in our disaster-aware approach) indicates whether content c is hosted at DC d, B_c is the size (in GB) of content c, and St_d is the storage cost (per GB) for DC d. The content replication and DC placement constraints are similar to the disaster-aware approach; additional constraints express content placement. This approach ensures that the content placement incurs the minimum storage cost. Our original disaster-aware approach minimizes only the risk without any cost constraint, for which storage cost can be a trade-off. Hence, we first minimize cost in the minStorageCost approach, and then we employ a budget constraint (based on the minimum cost) in the disaster-aware DC and content placement design to achieve risk minimization within a cost bound. The budget constraint is as follows:

$$\sum_{c\in C}\sum_{d\in D} T_{dc}(B_c \cdot St_d) \le Budget,$$

where $Budget = minCost$, the minimum storage cost obtained from the disaster-unaware minStorageCost placement approach, since this is the minimum possible storage cost for placing a given number contents in DCs. Besides the content unavailability and unreachability constraints, the DC and content placement decision in the disaster-aware approach will now also be bounded by DC storage cost. The solution will still minimize the overall risk in the network but within a given budget. Hence, the solution will have less flexibility in minimizing risk compared with the original approach (without any budget constraint), but the placement will be more cost-efficient. Depending on the provider's priority, the budget can be adjusted to achieve better/lower risk minimization. The higher the budget (for safeguarding the contents and the resources in the network from disasters), the higher the scope for risk minimization.

In our disaster-aware ILP, the number of variables is $|V| + |V||C| + |M||C||V| + 2|M||C||V||V|$, $O(|M||C||V|^2)$, and the number of constraints is $2|V| + 2|C| + |M|(3|C||V||V| + |V||V| + |C|) + 1$, $O(|M||C||V|^2)$. For the disaster-unaware approaches that minimize resources without any risk consideration, the number of variables is at most $|V| + |V||V| + |C||V| + |C||V||V|$, $O(|C||V|^2)$, and the number of constraints is $2|V||V| + 3|C||V| + 2|V| + |C| + 1$, $O(|V|^2 + |C||V|)$. Thus, the product of the number of contents and the number of disasters yields additional complexity for the disaster-aware approach versus disaster-unaware approaches. With network size, disaster type, and number of contents (in the range of hundreds and thousands), the problem size can grow significantly towards limited scalability.

III. Disaster-Aware Dynamic Content Management

The disaster-aware DC and content placement approach ensures that DCs and contents are placed at locations with minimum risk based on disaster probabilities. But the probability of disasters and content properties (demands and importance) can be time-varying. Suppose the contents in the initial DC locations are less vulnerable to loss from a given set of disasters, but a change in disaster probabilities can make a currently safe DC location become risky at a later time. Once established, it is not viable to relocate a DC based on changing the risk profile. Rather, with a given set of DCs, and updated information on disaster probabilities, we can rearrange the placement of the contents such that the expected content loss is reduced at any given time. In response to an upcoming disaster alert, such a reconfiguration might need to be performed in a very limited time scale, ranging from minutes to hours depending on the type of alert. Also, initially, we consider a fixed number of replicas for content, but with time, the required number of replicas can change. The placement can be reconfigured periodically, e.g., based on hourly/daily traffic variation, or whenever content properties change.

But rearranging contents among a set of DCs is resource intensive since it involves adding replicas in some DCs and deleting them in others. So we need to reduce the number of rearrangements and hence replication cost, e.g., network bandwidth consumption (cost is linearly proportional to the number of replicas) [15], and the background traffic for synchronization among different replicas of content.

We propose a DADCM algorithm that adjusts the content placement based on how disaster risks and content popularity/importance evolve in time. After the proposed disaster-aware static approach (obtained from ILP) decides on the DC placement and the initial content placement, we propose a dynamic scheme for reconfiguring the initial placement of the contents among the chosen set of DCs depending on network updates. Hence, the solution of the static approach serves as the initial placement input for the algorithm. The problem is to dynamically decide where content is to be replicated so that risk, in terms of *expected content loss*, is reduced under dynamic settings of disaster events and demands. While performing DADCM, we consider 1) reducing the number of content rearrangements, 2) satisfying the quality of service (QoS) (minimum access latency) of users, and 3) reducing the number of replicas within risk requirements.

Satisfying the latency constraint with risk reduction is a challenge because these two objectives are fairly contradictory. Some parts of the network may pose the least threat and hence minimum risk, but such parts may be distant from users, hence violating the minimum latency requirement. Also, placing contents closer to popular regions can reduce latency, but such regions can be vulnerable to disasters. We further try to reduce the number of replicas (as long as demands are met) after satisfying risk and latency requirements to save resources.

We analyze risk based on existing DC placement with new *updated* settings: a new set of user nodes and hence new demands for content c (S'_c), a new required number of replicas (R'_c) determined based on new demands (η'_c) and new importance factors (α'_c), and a new set of disaster events (M'). Given a network with existing DC locations and dynamic settings, we need to update the placement of contents to reduce risk, i.e., *expected content loss* in the network.

Our proposed heuristic, DADCM, updates the placement of *one content at a time* with the goal of global risk reduction for all contents. We assume the service provider provides the following:

1. *Threshold risk*: A reasonable risk value, $threshold\_risk_c$, which satisfies the expected level of availability of content c against a disaster. It can be roughly associated with the recovery point objective (RPO), a metric used in business continuance and disaster recovery of a system. It indicates the amount of data loss (or at risk of being lost), measured in time, that a system can tolerate or the maximum acceptable period in which data might be lost in the case of a critical event [19–21]. Our goal is to achieve this threshold value, or, in case $threshold\_risk_c$ cannot be met, at least to guarantee the minimum achievable risk with given disaster alerts.

2. *Latency distance*: An average latency-aware distance, F'_{sd}, between the user nodes and the serving DCs, which satisfies the minimum latency constraint of the users as determined by QoS. For latency in an optical network, it is important to consider the propagation delay of light in fibers, which is about 5 µs/km. Such delay can be minimized by choosing the shortest-distance path. During content placement, we consider latency by choosing a limited set of paths for reachability. But to conform to QoS, we check whether the shortest-path distance, F_{sd}, between a serving DC d and a requesting user node s is within F'_{sd}. If the latency constraint cannot be satisfied due to risk requirements, node s will at least have the content available via a longer-latency path.

For DADCM, we compute *for each content*, at run time, some risk parameters based on which we decide whether a content placement should be updated. The parameters are calculated based on the optimization objective function in Eq. (1) as follows:

1. Partial risk per DC:

$$partial\_risk_c(d) = \sum_{m \in M'} \sum_{s \in S'_c} \alpha'_c \cdot J^m_{dcs} \cdot P^m_{ds}.$$

For each content c, we retrieve the *old placement*, D_{c_old}, i.e., set of DCs where c is initially replicated, and compute $partial\_risk_c(d)$ that *each DC d* may incur with *new updated settings*. It is computed *per DC* because we want to analyze how each DC location independently contributes to the risk (hence it is *partial* risk) of a content placement. We sort the DCs based on $partial\_risk_c(d)$ values to determine which DC locations are safe (low risk) and which are risky for hosting c in the new settings. We repeat this risk-based sorting on the remaining set of DCs, D'_c (by assuming that c is hosted in these DCs). These DCs are candidate locations for future hosting of c in case the current placement of c needs to be updated.

2. Minimum risk:

$$min\_risk_c = \sum_{i=1}^{R'_c} partial\_risk_c(d_i).$$

We determine the new required number of replicas R'_c for content c based on η'_c and α'_c. We choose R'_c safe DC locations (DCs with the lowest $partial\_risk_c(d)$ values) from the list of all the DCs, D. We denote the combined risk, i.e., the sum of all individual $partial\_risk_c(d)$ values of these DCs, as min_risk_c. This is the lowest achievable risk for c because placing c in these DCs will incur the minimum possible risk with new settings.

3. Current risk:

$$curr\_risk_c = \sum_{m \in M'} \sum_{s \in S'_c} \sum_{d \in D_{c\_new}} \alpha'_c \cdot J^m_{dcs} \cdot P^m_{ds}.$$

For each content c, we acquire the *new placement*, D_{c_new}, i.e., the set of DCs where c is replicated after adjusting for new settings. We compute $curr\_risk_c$ as the total risk of this updated placement and then analyze whether this is acceptable.

Algorithm: Heuristic DADCM is applied *per content* and is divided into five phases, each addressing a separate goal as shown in Fig. 2. The phases are described in Algorithm 1, and the notations used in the algorithm are defined below:

- C: Set of contents.
- D: Set of all DCs.
- S'_c: Set of new user nodes requesting content $c \in C$.
- M': Set of new disaster events.
- α'_c: New importance factor of content $c \in C$.
- η'_c: New demand for content $c \in C$.
- D_{c_old}: Set of old DCs hosting content $c \in C$, $D_{c\_old} \in D$.
- D_{c_new}: Set of new DCs hosting content $c \in C$, $D_{c\_new} \in D$.
- D'_c: Set of candidate DCs to host content $c \in C$, $D'_c \in D$.
- R_c: Current number of replicas for content $c \in C$.
- R'_c: New required number of replicas for content $c \in C$.
- $threshold\_risk_c$: Threshold risk of placing content $c \in C$.
- $partial\_risk_c(d)$: Partial risk incurred by DC $d \in D$.
- min_risk_c: Minimum risk incurred by R'_c DCs $d \in D$ for content $c \in C$.
- $curr\_risk_c$: Current/updated risk incurred by DCs $d \in D_{c\_new}$ currently hosting content $c \in C$.
- F_{sd}: Shortest-path distance between nodes $s \in S'_c$ and $d \in D$.
- F'_{sd}: Latency-aware distance between nodes $s \in S'_c$ and $d \in D$.
- S_{c_Lat}: Set of user nodes covered by latency constraint, $S_{c\_Lat} \in S'_c$.
- S_{c_noLat}: Set of user nodes not covered by latency constraint, $S_{c\_noLat} \in S'_c$.
- $L_{users}(d)$: Set of user nodes served by DC $d \in D$ within latency constraint.
- $DC\_count(s)$: Number of DCs that serve user node $s \in S'_c$ within latency constraint.
- D_{c_lat}: Set of candidate DCs that satisfy users' latency constraint for content $c \in C$, $D_{c\_lat} \in D'_c$.

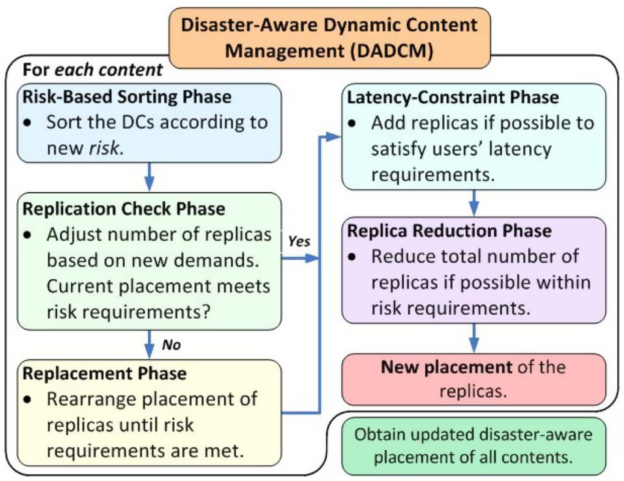
Disaster-Aware Dynamic Content Management (DADCM)
For each content
Risk-Based Sorting Phase
• Sort the DCs according to new risk.
Replication Check Phase
• Adjust number of replicas based on new demands. Current placement meets risk requirements?
Yes
No
Replacement Phase
• Rearrange placement of replicas until risk requirements are met.
Latency-Constraint Phase
• Add replicas if possible to satisfy users' latency requirements.
Replica Reduction Phase
• Reduce total number of replicas if possible within risk requirements.
New placement of the replicas.
Obtain updated disaster-aware placement of all contents.

Fig. 2. Algorithm DADCM.

Algorithm 1 Disaster-Aware Dynamic Content Management

Risk-Based Sorting Phase

```
Input: $C$, $D$, $S'_c$, $M'$, $\alpha'_c$, $D_{c\_old}$, $R'_c$.
Output: Sorted $D_{c\_old}$, $D'_c$.
1: Sort set of all contents, $C$, in descending order based on
   number of user requests $(s,c)$.
2: for each $c \in C$ do
3:    for each $d \in D$ do
4:       Compute $partial\_risk_c(d)$
5:    end for
6:    Sort the set of all DCs, $D$, in ascending order based
      on $partial\_risk_c(d)$
7:    Compute $min\_risk_c$
8:    $D'_c \leftarrow \{D \backslash D_{c\_old}\}$ //Get the set of remaining candidate
      DCs, $D'_c$ from $D$ by excluding $D_{c\_old}$.
9:    Sort $D_{c\_old}$ and $D'_c$ in ascending order based
      on $partial\_risk_c(d)$
10:   Replication_Check_Phase ($D_{c\_old}$, $D'_c$)
11: end for
```

Replication Check Phase

```
Input: $R_c$, $R'_c$, $D_{c\_old}$, $D'_c$, $threshold\_risk_c$.
Output: $D_{c\_new}$, $D'_c$, $curr\_risk_c$.
1: $\Delta R_c \leftarrow R'_c - R_c$
2: if $\Delta R_c > 0$ then
3:    Add replicas to $\Delta R_c$ DCs with lowest $partial\_risk_c(d)$
      in $D'_c$
4:    $D_{c\_new} \leftarrow \{D_{c\_old} + D'_c[1{:}\Delta R_c]\}$;  $D'_c \leftarrow \{D \backslash D_{c\_new}\}$  //
      Update placement with new replicas to $D_{c\_new}$ and
      update $D'_c$
5:    $curr\_risk_c \leftarrow$ a very large number
6: else if $\Delta R_c < 0$ then
7:    Delete redundant replicas from $\Delta R_c$ DCs with highest
      $partial\_risk_c(d)$ in $D_{c\_old}$
8:    $D_{c\_new} \leftarrow \{D_{c\_old} \backslash D_{c\_old}[end{:}-1{:}\Delta R_c]\}$; $D'_c \leftarrow \{D \backslash D_{c\_new}\}$
      //Update placement to $D_{c\_new}$ and update $D'_c$
9:    Compute $curr\_risk_c$ with $D_{c\_new}$
10: else if $\Delta R_c = 0$ then
11:   $D_{c\_new} \leftarrow D_{c\_old}$ //Keep old placement and update it
      as $D_{c\_new}$
12:   Compute $curr\_risk_c$ with $D_{c\_new}$
13: end if
14: if $curr\_risk_c > threshold\_risk_c$ then
15:   Replacement_Phase ($D_{c\_new}, D'_c, curr\_risk_c$)
16: else
17:   Latency-Constraint_ Phase ($D_{c\_new}, D'_c$)
18: end if
```

Replacement Phase

```
Input: $D_{c\_new}$, $D'_c$, $min\_risk_c$, $threshold\_risk_c$, $curr\_risk_c$.
Output: $D_{c\_new}$, $D'_c$, $min\_risk_c$, $curr\_risk_c$.
1: $min\_risk_c \leftarrow max\{min\_risk_c, threshold\_risk_c\}$
2: while $curr\_risk_c > min\_risk_c$ do
3:    Add replica in DC with lowest $partial\_risk_c(d)$ in $D'_c$.
4:    Delete replica from DC with highest $partial\_risk_c(d)$
      in $D_{c\_new}$.
5:       $D_{c\_new} \leftarrow \{D_{c\_new} + D'_c[1]\}$,   $D_{c\_new} \leftarrow \{D_{c\_new} \backslash D_{c\_new}$
         $[end]\}$;  $D'_c \leftarrow \{D \backslash D_{c\_new}\}$  //Update placement to
         $D_{c\_new}$ and update $D'_c$
6:    Compute $curr\_risk_c$ with $D_{c\_new}$
7: end while
8: Latency-Constraint_Phase ($D_{c\_new}$, $D'_c$)
```

Latency-Constraint Phase

```
Input: $D_{c\_new}$, $D'_c$, $S'_c$, $F_{sd}$, $F'_{sd}$.
Output: $D_{c\_new}$, $D'_c$, $DC\_count(s)$, $L_{users}(d)$.
1: for each $d \in D_{c\_new}$ do
2:    for each $s \in S'_c$ do
3:       Compute $F_{sd}$
4:       if $F_{sd} \le F'_{sd}$ then
5:          $S_{c\_Lat} \leftarrow \{S_{c\_Lat} \cup \{s\}\}$ //Put node $s$ in $S_{c\_Lat}$
6:          $L_{users}(d) \leftarrow \{L_{users}(d) \cup \{s\}\}$ //Put node $s$ in $L_{users}(d)$
            for DC $d$
7:          $DC\_count(s) \leftarrow DC\_count(s) + 1$ //Increment count
            of DCs, $DC\_count(s)$, for node $s$
8:       end if
9:    end for
10: end for
11: $S_{c\_noLat} \leftarrow \{S'_c \backslash S_{c\_Lat}\}$ //Find set of user nodes not
    covered within $F'_{sd}$, $S_{c\_noLat}$, from $S'_c$
12: if $S_{c\_noLat} = \varnothing$ then
13:   $D_{c\_new} \leftarrow D_{c\_new}$ //Keep placement $D_{c\_new}$
14: else
15:   Sort $D'_c$ in ascending order based on $partial\_risk_c(d)$
16:   for each $s \in S_{c\_noLat}$ do
17:      for each $d \in D'_c$ do
18:         Compute $F_{sd}$
19:         if $F_{sd} \le F'_{sd}$ then
20:            $D_{c\_lat} \leftarrow \{D_{c\_lat} \cup \{d\}\}$ //Put DC $d$ in $D_{c\_lat}$
21:            $L_{users}(d) \leftarrow \{L_{users}(d) \cup \{s\}\}$ //Put node $s$ in
               $L_{users}(d)$ for DC $d$
22:            $DC\_count(s) \leftarrow DC\_count(s) + 1$ //Increment
               $DC\_count(s)$ for node $s$
23:            break
24:         end if
25:      end for
26:   end for
27:   if $D_{c\_lat} = \varnothing$ then
28:      $D_{c\_new} \leftarrow D_{c\_new}$ //Keep placement $D_{c\_new}$
29:   else
30:      for each $d \in D_{c\_lat}$ do
31:         Add replica in DC $d$
```

```
32:         end for
33:         $D_{c\_new} \leftarrow \{D_{c\_new} + D_{c\_lat}\}$ //Update placement $D_{c\_new}$
34:     end if
35:     $D'_c \leftarrow \{D \backslash D_{c\_new}\}$ //Update placement $D'_c$
36: end if
37: Replica_Reduction_Phase ($D_{c\_new}$, $D'_c$, $DC\_count(s)$, $L_{users}(d)$)
```

Replica Reduction Phase

```
Input: $D_{c\_new}$, $D'_c$, $min\_risk_c$, $DC\_count(s)$, $L_{users}(d)$.
Output: $D_{c\_new}$, $D'_c$, $curr\_risk_c$,.
1: Sort $D_{c\_new}$ in descending order based on $partial\_risk_c(d)$
2: for each $d \in D_{c\_new}$ do
3:     $essential\_replica \leftarrow FALSE$
4:     for each $s \in L_{users}(d)$ do
5:         if $DC\_count(s) \leq 1$ then
6:             $essential\_replica \leftarrow TRUE$
7:             break
8:         end if
9:     end for
10:    if $essential\_replica = FALSE$ then
11:        Compute $curr\_risk_c$ with $D_{c\_new}$ without DC $d$ ($\{D_{c\_new} \backslash \{d\}\}$)
12:        if $curr\_risk_c < min\_risk_c$ then
13:            Delete replica in DC $d$
14:            $D_{c\_new} \leftarrow \{D_{c\_new} \backslash \{d\}\}$ //Update the placement $D_{c\_new}$
15:            for each $s \in L_{users}(d)$ do
16:                $DC\_count(s) \leftarrow DC\_count(s) - 1$ //Decrement $DC\_count(s)$ for node $s$
17:            end for
18:        else
19:            $D_{c\_new} \leftarrow D_{c\_new}$ //Keep placement $D_{c\_new}$
20:        end if
21:    end if
22: end for
```

In the **risk-based sorting phase**, we sort all the DCs that host the replicas of content c and the remaining candidate DCs based on $partial\_risk_c(d)$ values to differentiate risky DC locations from safe DC locations. Based on this sorting and the new required number of replicas, R'_c, we determine the minimum risk, min_risk_c.

In the **replication-check phase**, after comparing R_c and R'_c, if content c requires more replicas, we add replicas in new safe (low-risk) candidate DCs and update the current placement. If c has more replicas than required by new settings, we delete the redundant replicas from high-risk DCs and update the current placement. We then check whether the $curr\_risk_c$ value of the current placement of c meets the required $threshold\_risk_c$ value. If the risk requirements are not satisfied, we move to the replacement phase in order to reduce the risk of the placement, or else we move to the latency-constraint phase. It should be noted that whenever we add more replicas, even if we place additional replicas in new safe DCs, the high-risk DCs from the original placement remain. Hence, we aim to reduce risk by always moving to the replacement phase in the case of adding replicas.

In the **replacement phase**, to reduce or limit the number of rearrangements in the network, we aim to keep the existing placement of a content c as long as it meets the $threshold\_risk_c$ value even if it is not the best solution with the new settings. Hence, we set min_risk_c as $max\{min\_risk_c, threshold\_risk_c\}$. If risk requirements for c are not met, we rearrange the placement of *one replica at a time* until min_risk_c is achieved. We start with adding a replica in a new candidate DC with the lowest $partial\_risk_c(d)$ value, and deleting a replica from a currently hosting DC with the highest $partial\_risk_c(d)$ value. In the worst case, all replicas may need to be rearranged or re-replicated at new safer DC locations, but this can achieve the best possible placement or the minimum achievable risk at the expense of resource usage.

After risk reduction, we verify whether the current placement of replicas conforms to the users' latency requirements. In the **latency-constraint phase**, we compute the shortest-path distance F_{sd}, between every DC d hosting content c and every user node s requesting c. We check whether F_{sd} is within the latency distance, F'_{sd}, and obtain S_{c_Lat}, a set of user nodes covered by the latency constraint. Thus, we get S_{c_noLat}, the set of user nodes for which no DC from the current placement satisfies the latency constraint. For such node s in S_{c_noLat}, we find a new safe candidate DC d from D'_c that meets the latency constraint of F'_{sd} for s and add a replica in d. If no such DC is found, content placement remains unchanged.

In the **replica reduction phase**, we reduce the number of replicas by deleting redundant replicas given that overall risk reduction is achieved. For each DC d, we obtain $L_{users}(d)$, a set of user nodes that are being served by d *within the latency constraint* of F'_{sd}. If any such node s in $L_{users}(d)$ exists that is served only by DC d (i.e., node s is served by only one DC within the latency constraint or $DC\_count(s) = 1$), then the replica at d is essential. If all user nodes served by d are also served by other DCs within the latency constraint or $DC\_count(s) \geq 1$ for all such nodes, then the replica at d is not essential and can be deleted. We analyze $curr\_risk_c$ of the placement without d; if the risk requirements are satisfied, we delete the replica from DC d and update the placement to *new placement*. If replica reduction is not possible either due to latency constraint or risk requirements, content placement remains unchanged.

The resulting placement is the final placement for content c. We thus obtain updated placements of all contents and determine the total risk in the network. Accounting for all five phases in the algorithm DADCM, and considering the average-case sorting time as $O(|V| \log |V|)$, the complexity of the heuristic is $O(|C||M||V|^5)$.

IV. Content-Management Cost Analysis

Ensuring disaster-aware dynamic content management (DADCM) in the cloud can be resource intensive since it involves content replication and rearrangement among DCs. We want to analyze the cost of resource utilization associated with our dynamic content-management scheme.

The monetary cost for cloud resource utilization, such as storage and bandwidth usage for outbound data, is usually charged based on data usage per billing period, e.g., on a monthly basis [22]. Content incurs a storage cost, i.e., the monetary cost of storing it in a DC for a billing period, and a bandwidth cost, i.e., the monetary cost for replicating or updating it over the cloud during a billing period.

Content replication means copying content from a hosting DC to a DC that does not host the content, so it involves inter-DC data transfer or bandwidth cost [22]. A new replica can be created or copied from a master replica or another replica in the nearest location (depending on the cloud provider's policies). To maintain consistency among different replicas of content, all replicas need to be synchronized or updated to the same state, and this incurs data transfer or bandwidth cost. Replicas can be updated, periodically from a master replica or based on the most recent update. Frequent content updates lead to more background traffic for synchronization. Since such traffic includes only updates, for simplicity, we assume the synchronization data as a fraction of the entire data. Thus, each content incurs a storage cost of all its replicas and a bandwidth cost for replication and synchronization in the cloud.

To develop a *content-management cost model* for our case study, we explore some current cloud-pricing schemes. According to information from some major cloud providers [23–26], data storage and data transfer (bandwidth) costs are determined based on the volume of data usage per month. Most cloud providers today charge for outbound (egress) data transfers based on the volume of data going out of a provider's DC via the Internet in a given billing cycle (typically a flat rate per GB), and inbound transfers are usually free. The cost of outbound data transfer can vary depending on the destination region or zone. For example, the cost of outbound traffic within the US is uniform, whereas it varies for traffic going outside of the US [23,24]. The charge for storage capacity is based on the average daily amount of data stored (in GB) over a billing period. Some providers also charge for metadata functions associated with copying or deleting files. These operational costs are generally negligible (compared with storage and bandwidth cost) based on aggregated requests per month. Providers also offer volume discount or tiered pricing schemes where costs per GB differ based on the total consumption in a monthly period [23–27]. A pricing example can be found in [23].

For our content-management cost model, we consider for content c storage cost ($Cost_c^{Str}$), replication cost ($Cost_c^{Rep}$), and synchronization cost ($Cost_c^{Syn}$). Let st_d be the cost *per GB* of storage capacity in DC d and bw_d be the cost *per GB* of outbound bandwidth transfer from DC d. Let B_c be the size (in GB) of content c. Three different costs are devised as follows:

1. Storage cost: Replicas of content are hosted at different DCs that can have different storage pricing. The storage cost of content is the combined storage cost of all the replicas. Let D_c be the set of DCs where the replicas of content c are hosted. Hence,

$$Cost_c^{Str} = \sum_{d \in D_c} st_d \cdot B_c.$$

2. Replication cost: Creating or adding new replicas of content involves copying the content from one DC to other DCs. Hence, adding n new replicas incurs n data transfers and n times additional storage capacity. Let D_{c_add} be the set of DCs where new replicas are to be added, and let the content be replicated from DC d_a to the DCs in D_{c_add}. Then,

$$Cost_c^{Rep} = n \cdot bw_{d_a} \cdot B_c + \sum_{d \in D_{c\_add}} st_d \cdot B_c, \qquad d_a \notin D_{c\_add}.$$

3. Synchronization cost: Synchronization traffic includes fractional data updates. Let ΔB_c be the fractional size (in GB) of content c that requires updating. For m replicas, $m-1$ replicas need to be synchronized and, hence, incur $m-1$ data transfers (update cost of the mth replica in the local server can be ignored). Let $m-1$ replicas be synchronized based on the most updated replica at DC d_a. Hence,

$$Cost_c^{Syn} = (m-1) \cdot bw_{d_a} \cdot \Delta B_c.$$

We assume that the size of the content will remain the same after an update and that the update will occur only on ΔB_c. It can happen that the replicas will need additional (or reduced) storage capacity after being updated, in which case the synchronization cost will need to be adjusted to include the corresponding storage cost.

While adding or updating a replica can incur additional cost, deleting a replica can reduce $Cost_c^{Str}$ and $Cost_c^{Syn}$ for content; we assume the cost of deletion and other similar requests to be negligible. Both $Cost_c^{Str}$ and $Cost_c^{Syn}$ are proportional to the number of existing replicas. Whether a content replica is added, deleted, or updated, the corresponding costs can be analyzed based on our model.

For cost analysis, we apply our content-management cost model to our DADCM algorithm. We compute, *for each content*, the cost incurred from the initial disaster-unaware placement and from the updated disaster-aware placement, and then determine the total cost of placement of all the contents in both cases. For disaster-unaware placement, we analyze the cost associated with placement D_{c_old}. Since this is the initial placement and no replicas are added or deleted, only $Cost_c^{Str}$ and $Cost_c^{Syn}$ are incurred. Through subsequent phases of the algorithm, replicas can be added and/or deleted and the costs change accordingly. For disaster-aware placement, we analyze the cost of the updated placement D_{c_new}. Besides $Cost_c^{Str}$ and $Cost_c^{Syn}$, $Cost_c^{Rep}$ can also be incurred if additional replicas are added through content replication or rearrangement. Since our algorithm aims to reduce or limit the number of content rearrangements in the network, replication costs will be limited. Furthermore, since the replica reduction phase in our algorithm may reduce the total number of replicas in the network, it may be possible that the associated storage cost and synchronization cost of the updated placement are lower than the associated costs of the initial placement. In case no replica reduction takes place and additional replicas are added, the total cost of the disaster-aware updated placement will be higher than the cost of the disaster-unaware initial placement. Thus, using this cost model, we can determine the costs that may be incurred while designing a disaster-aware content-management approach.

V. Illustrative Numerical Examples

To evaluate the benefits of our disaster-aware DC and content placement design, we simulated a 24-node USnet topology, with a focus on WMD attacks. Based on Fig. 1, we consider possible locations of major military facilities as probable targets of WMD attacks and modeled 10 WMD attack zones as shown in Fig. 3.

We consider not only primary attacks but also correlated/cascading effects such as secondary attacks and power outages. Based on information in [28] and considering large-scale disaster and multiple correlated/cascading effects, we assume a failure span of 1000 km around the targeted areas. The probabilities of damage/failure on nearby nodes and links are estimated with reasonable assumptions (between 0 and 1) based on their distances from the target's epicenter. For example, in Fig. 3, in the case of attack event m_1 (shown with corresponding disaster zone), two links (3–4 and 3–7) are estimated to be damaged with probability 1, and nodes 3, 4, and 7 have estimated damage probabilities of 0.7, 0.4, and 0.05, respectively (decreasing with distance). Similarly, from attack event m_2, node 5 is estimated to be damaged with probability 1; consequently, all associated links are failed, and neighboring nodes 3 and 4 have estimated damage probabilities of 0.05 and 0.1, respectively. Here, attack event m_1 can cause expected loss due to unreachability if DCs are placed at nodes 3 and 4, and m_2 can cause expected loss due to unavailability if a DC is placed at node 5. The worst case in terms of link failure is attack event m_{10} where node 19 gets disconnected from the network.

We consider all network nodes as candidates for DC, and we consider eight DCs for placement. The user nodes are distributed according to population density and their vicinity to military facilities because these regions will generate the most requests, and we consider all nodes as user nodes. Typically, in a cloud network, the number of contents can be very large, but, since the ILP has limited scalability, we first consider a small number of contents, e.g., 20, with α_c assigned on a scale of 1–10. We assume that all user nodes can request all contents. We calculate R_c using $\gamma_1 = 0.2$, $\gamma_2 =$ a random number generated uniformly between 0.2 and 0.5, $r = 2$, and $R_{max} = 6$. These values are based on the total number of DCs and demands in the network. We chose to put more weight on α_c since it is a contributing factor in risk minimization, and we want to ensure survivability of the more important contents. We assume an average content size of 2 GB, and hence the value of B_c ranges from 1 to 3 GB. DC storage pricing is discussed in the simulation parameters for the dynamic content-management heuristic.

We compare our disaster-aware and the disaster-unaware (minPathCost) DC and content placement approaches. As shown in Fig. 4, disaster-aware placement (shown in green and bold lines) assigns the DCs in locations avoiding the nodes vulnerable to WMD attacks. For instance, based on the attack events, the worst candidate nodes for a DC are nodes 5, 20, 21, and 19, and the next worst candidates are nodes 3 and 4 since they are associated with high link failures from both attack events m_1 and m_2. The contents are distributed according to their importance and their demands from user nodes. Disaster-unaware (minPathCost) placement (shown in red and dashed lines) assigns DCs to locations that may be vulnerable to attacks since it only minimizes the path cost from user nodes to DC nodes. In the case of a WMD attack, DCs at nodes 5, 20, and 21 will incur large expected content loss due to unavailability, whereas DCs at nodes 3 and 4 will incur high expected content loss due to unreachability. Such placement may not be desirable.

We compare the expected content loss (i.e., risk) between disaster-aware and disaster-unaware (minPathCost) approaches for different numbers of contents (10–40) as shown in Fig. 5. We calculated the risk of the two approaches based on the objective function in Eq. (1). We average the risk values over numbers of contents; the average value for 10 contents in the disaster-aware approach is normalized to 1, and other values are adjusted accordingly. The results report an average over 20 randomized instances and show the normalized per-content risk

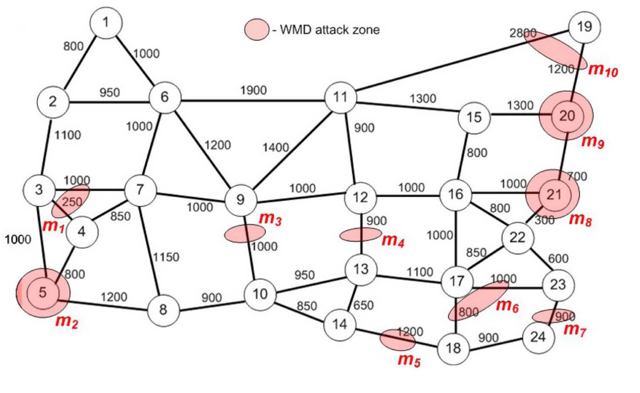

Fig. 3. 24-node USnet topology (distances in kilometers) with possible WMD attack zones.

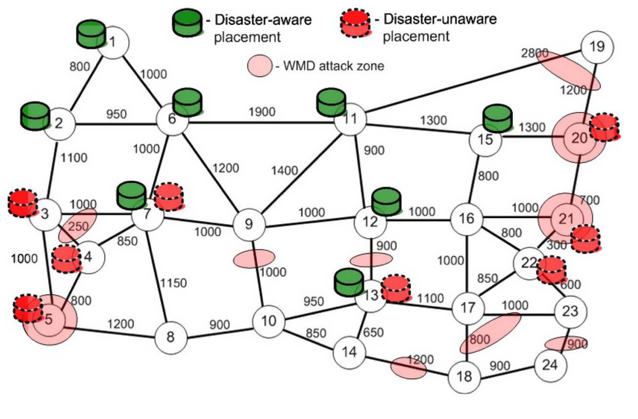

Fig. 4. Disaster-aware and disaster-unaware (minPathCost) DC and content placement.

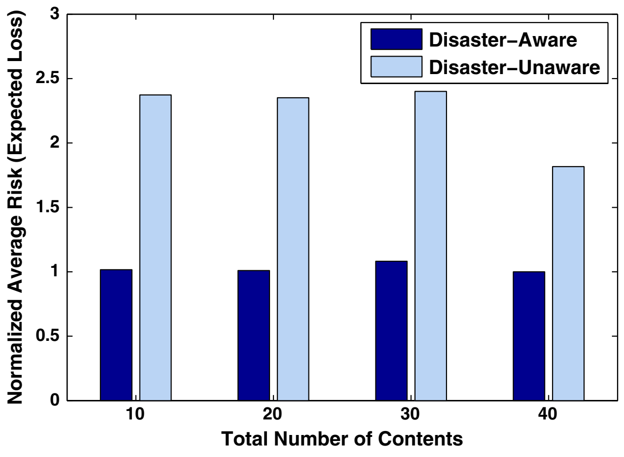

Fig. 5. Risk comparison between disaster-aware and disaster-unaware (minPathCost) approaches.

(for the disaster-aware approach) with 95% confidence intervals [0.84366, 1.1563], [0.89288, 1.1071], [0.8782, 1.1218], and [0.95398, 1.046] for 10, 20, 30, and 40 contents, respectively. It can be shown that the disaster-aware approach incurs significant improvement of about 45%–57% risk reduction over the disaster-unaware minPathCost approach for this typical US-wide network scenario and given WMD zones.

In Fig. 6, we compare the resource usage, i.e., the average path cost of the two approaches, for different numbers of contents. The disaster-aware approach incurs about 16% higher path cost over the disaster-unaware minPathCost approach for 20 contents. For other numbers of contents, about 6%–10% higher path costs are incurred. Hence, the general trend of the results for this typical network scenario and given WMD zones shows that our disaster-aware approach is efficient and not very resource intensive.

In Fig. 7, we compare expected content loss (risk) of the disaster-aware approach, *with* and *without budget*, for different numbers of contents (10–40). We set $Budget = minCost$ obtained from the minStorageCost ILP. Risk values are averaged over numbers of contents; the average value for 10 contents from the disaster-aware approach *without budget* is normalized to 1, and other values are adjusted accordingly. The budget-constrained approach ($DA\ /w\ budget$) shows about an 8%–10% increase in risk, for different numbers of contents, compared with the unconstrained approach ($DA\ w/o\ budget$). Even with the minimum budget, we observe only an 8%–10% increase in risk; hence, by keeping the cost bounded, the trade-off in risk is quite reasonable.

To study the impact of the budget on risk minimization, we increase the budget in small percentages and compare the risk. We use $Budget = x * (minCost)$ with $x = [0.05, 0.1, 0.15, 0.2]$. We show that risk decreases with increasing budget and converges to the minimum risk of the unconstrained approach only after a 15% increase in budget. Thus, we infer that, by keeping the budget minimum, we compromise risk by only about 10%, while with only a 15% increase in budget, we can achieve the optimal risk. This study can help network operators to budget their disaster-awareness goals accordingly.

For our disaster-aware dynamic content management (DADCM) design, we simulated the 24-node USnet topology under the same settings as before. We alter the values of η_c, α_c, and disaster probabilities within a reasonably small range (±10–20%) to simulate dynamic settings. We experimented with 10 different inputs for disaster probabilities, η_c, α_c, and hence R_c, for different numbers of contents, and observed $curr\_risk_c$ values. Based on the average of these values, we used $threshold\_risk_c = 100$ for our examples. We used $F'_{sd} = 3000$, based on distances (in kilometers) in the given topology. For content-management cost analysis, we used $st_d = [0.03 \quad 0.07 \quad 0.045]$ and $bw_d = [0.1 \quad 0.08 \quad 0.12]$; we chose a set of values for each parameter, based on real-world cloud pricing (per GB per month) [23–26], to simulate different pricings at different DCs. Each DC is randomly assigned a storage cost and an outbound bandwidth cost from the set of values of st_d and bw_d, respectively. For instance, DC 2 has st_2 of \$0.07/GB and bw_2 of \$0.1/GB. For our examples, we used

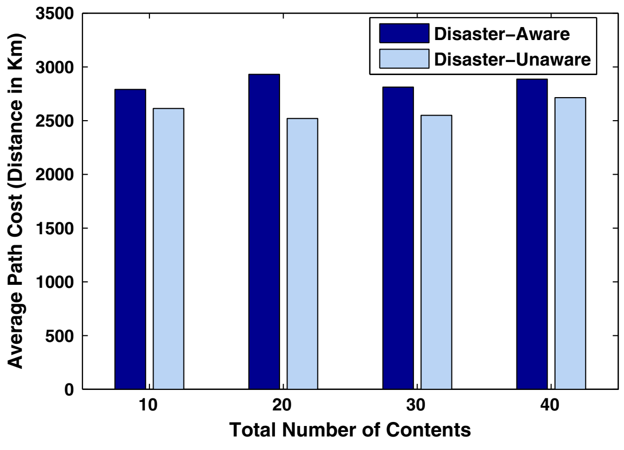

Fig. 6. Average path-cost comparison between disaster-aware and disaster-unaware (minPathCost) approaches.

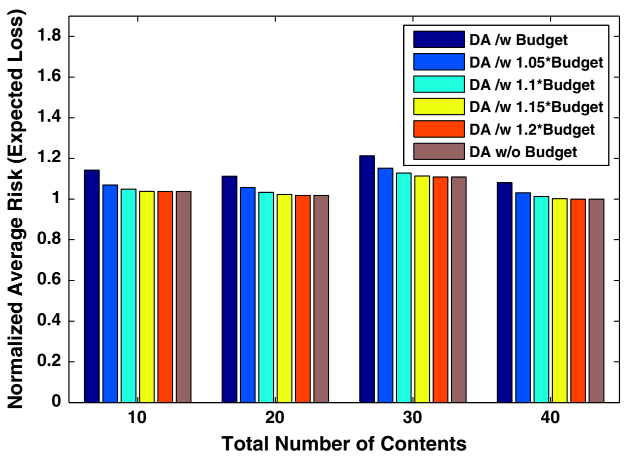

Fig. 7. Risk comparison of disaster-aware approach without budget and with increasing budget.

$B_c = 2$ GB as a typical content size and $\Delta B_c = 0.6$ GB by assuming that, typically, 30% of content gets updated during synchronization and that the size of the content remains the same after an update.

In Fig. 8, we show expected content loss or *risk* for large numbers of contents (100–5000) with dynamic settings. For comparing the disaster-aware approach with a disaster-unaware approach, we employ content placement based on demands in the network without any risk minimization. We then apply our heuristic on the disaster-unaware placement and compare the risk. Risk values are averaged over numbers of contents, and the average value for 100 contents for the disaster-aware approach is normalized to 1, and other values are adjusted accordingly. Our results show that the disaster-aware approach provides significant improvement in risk reduction (about 45%) over the disaster-unaware approach, for this typical US-wide network scenario.

In Fig. 9, we compare the risk of the dynamic disaster-aware approach for different values of the latency (QoS) parameter. The higher the latency constraint, the smaller the latency-aware distance. The placement becomes more restrictive because it limits the number of candidate DCs, and hence our algorithm is forced to place contents in relatively risky DCs, if required, as opposed to safer but far-away DCs. Risk reduction improves as the *latency distance* (and hence the set of candidate DCs) is increased; it improves significantly by about 30% at 3000 km from 2000 km. Hence, there is a trade-off between satisfying the latency constraint and minimizing risk. Note that risk reduction is not that significant after 3000 km and becomes almost constant after 4000 km. For the US-wide topology, the coast-to-coast distance is 5000 km; a latency constraint of 5000 km is equivalent to no latency constraint and hence it gives the maximum possible risk reduction. For other types of network topologies with higher node distribution and connectivity, such a phenomenon may be observed sooner.

To demonstrate the performance of our heuristic, we also compare the risk of content placement obtained from our heuristic and the static disaster-aware approach in Table I. For comparison, we adjust the ILP to content placement only, instead of joint DC and content placement, so that both approaches solve for the same fixed set of DCs. Even though the heuristic is a dynamic scheme and the ILP solves for the static placement scheme, we obtain, from the ILP, a benchmark for content placement for a given instance. In other words, the optimum solution from the ILP provides a lower bound for possible risks at a given instance, and we evaluate how the solution from our heuristic follows the lower bound at that instance. We compare our results for 10–40 contents (low numbers due to limited scalability of ILP). Risk values are averaged over numbers of contents, and the average value for 10 contents in the static approach is normalized to 1, and other values are adjusted accordingly. We observe that the risk values provided by our heuristic follow closely the lower bound obtained from ILP with average deviation of about 4.5%.

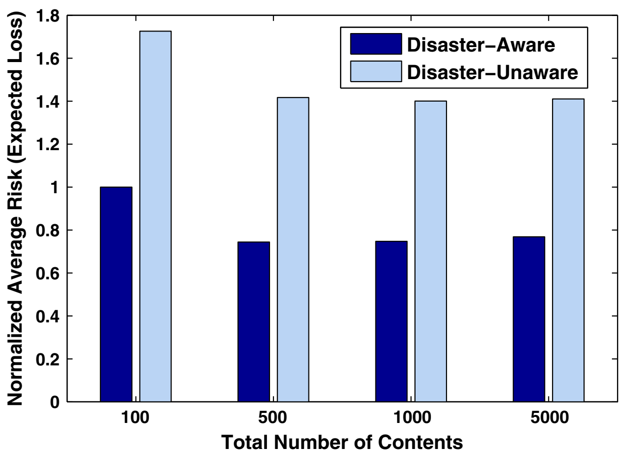

Fig. 8. Risk comparison between disaster-aware and disaster-unaware approaches with dynamic settings.

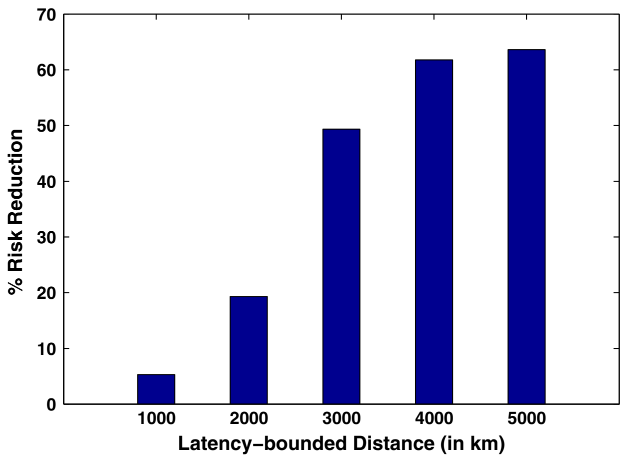

Fig. 9. Risk comparison of dynamic disaster-aware approach with different QoS values.

In Table II, we compare the content-management costs of the disaster-aware and disaster-unaware approaches for a large number of contents (100–5000) with dynamic settings. Cost values are averaged over numbers of contents, and the average value for 100 contents for the disaster-aware approach is normalized to 1, and other values are adjusted accordingly. Our heuristic limits the number of content rearrangements in the network, which, in turn, helps to limit replication cost. Also, the replica reduction phase in our heuristic helps to reduce the number of replicas and, hence, the overall cost in terms of storage and synchronization. Our results for this typical network scenario and given WMD zones show that the disaster-aware approach incurs about 16.5% lower cost over the

TABLE I
RISK COMPARISON BETWEEN HEURISTIC AND LOWER BOUND

Number of Contents	Lower Bound	Heuristic
10	1.00	1.09
20	1.24	1.29
30	1.37	1.40
40	1.39	1.43

TABLE II
CONTENT-MANAGEMENT COST COMPARISON BETWEEN DYNAMIC DISASTER-AWARE AND DISASTER-UNAWARE APPROACHES

Number of Contents	Disaster-Aware	Disaster-Unaware
100	1.00	1.17
500	1.02	1.21
1000	0.91	1.08
5000	1.01	1.21

disaster-unaware approach for 5000 contents, which can be considerable savings in terms of network resource usage for service providers.

Our heuristic is an efficient solution since, while reducing the overall risk in the network, it also reduces the network resource usage and satisfies the latency constraint.

VI. CONCLUSION

Recent disasters have shown that cloud networks are vulnerable to large-scale disaster failures, and it is becoming challenging for providers to ensure content availability at all times. We have provided a solution for the disaster-aware datacenter and content placement problem that aims to minimize the overall risk of a network in terms of expected loss of content. Compared with a disaster-unaware approach, it provides significant improvement in risk reduction. We have also analyzed the impact of an operational budget constraint on such placement. We then developed a dynamic content-management solution as an enhancement to the initial placement to make the network adaptable to changing conditions and disaster alerts. Besides reducing risk in our dynamic approach, we took into consideration QoS (latency) constraints and network resource usage. We have also presented a cost analysis of employing our dynamic content-management scheme in the network based on real-world cloud pricing. Our methods can highly benefit service providers in designing disaster-resilient cloud networks.

ACKNOWLEDGMENTS

This work has been supported in part by Defense Threat Reduction Agency (DTRA) Grant No. HDTRA1-10-1-0011. Preliminary versions of this work were presented in parts at IEEE ANTS 2013 and OFC 2014.

REFERENCES

[1] M. F. Habib, M. Tornatore, M. De Leenheer, F. Dikbiyik, and B. Mukherjee, "Design of disaster-resilient optical datacenter networks," *J. Lightwave Technol.*, vol. 30, no. 16, pp. 2563–2573, Aug. 2012.

[2] C. Develder, M. De Leenheer, B. Dhoedt, M. Pickavet, D. Colle, F. D. Turck, and P. Demeester, "Optical networks for grid and cloud computing applications," *Proc. IEEE*, vol. 100, no. 5, pp. 1149–1167, May 2012.

[3] C. Lam, H. Liu, B. Koley, X. Zhao, V. Kamalov, and V. Gill, "Fiber optic communication technologies: What's needed for datacenter network operations," *IEEE Commun. Mag.*, vol. 48, no. 7, pp. 32–39, July 2010.

[4] http://www.datacenterknowledge.com/archives/2013/01/15/amazon-to-add-capacity-to-us-east-region/.

[5] http://www.seattletimes.com/seattle-news/alaska-airlines-our- ticketing-system-up-and-running-again/.

[6] http://www.techradar.com/us/news/world-of-tech/roundup/one-in-two-businesses-will-experience-data-loss-this-year-1093376.

[7] B. Mukherjee, M. F. Habib, and F. Dikbiyik, "Network adaptability from disaster disruptions and cascading failures," *IEEE Commun. Mag.*, vol. 52, no. 5, pp. 230–238, May 2014.

[8] A. Bernstein, D. Bienstock, D. Hay, M. Uzunoglu, and G. Zussman, "Power grid vulnerability to geographically correlated failures—analysis and control implications," Columbia University, Tech. Rep., June 2012.

[9] F. Dikbiyik, M. Tornatore, and B. Mukherjee, "Minimizing the risk from disaster failures in optical backbone networks," *J. Lightwave Technol.*, vol. 32, no. 18, pp. 3175–3183, 2014.

[10] http://www.google.com/about/datacenters/inside/locations/.

[11] http://finance.yahoo.com/news/yahoos-data-center-ny-194329808 .html.

[12] A. Greenberg, J. Hamilton, D. Maltz, and P. Patel, "The cost of a cloud: Research problems in data center networks," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 39, no. 1, pp. 68–73, Jan. 2009.

[13] J. Xiao, B. Wu, X. Jiang, P. Ho, and S. Fu, "Data center network placement and service protection in all-optical mesh networks," in *Proc. DRCN*, Hungary, Mar. 2013.

[14] X. Dong, T. El-Gorashi, and J. Elmirghani, "Green IP over WDM networks with data centers," *J. Lightwave Technol.*, vol. 29, no. 12, pp. 1861–1880, June 2011.

[15] Y. Chen, R. H. Katz, and J. D. Kubiatowicz, "Dynamic replica placement for scalable content delivery," in *Proc. Int. Workshop on Peer-To-Peer (IPTPS)*, Cambridge, MA, Mar. 2002.

[16] N. Bartolini, F. Presti, and C. Petrioli, "Optimal dynamic replica placement in content delivery networks," in *Proc. Int. Conf. on Networks (ICON)*, Sydney, Australia, Sept. 2003, pp. 125–130.

[17] P. Agarwal, A. Efrat, S. Ganjugunte, D. Hay, S. Sankararaman, and G. Zussman, "The resilience of WDM networks to probabilistic geographical failures," in *Proc. IEEE Int. Conf. on Computer Communications (INFOCOM)*, Shanghai, China, Apr. 2011, pp. 1521–1529.

[18] X. Wang, X. Jiang, and A. Pattavina, "Assessing network vulnerability under probabilistic region failure model," in *Proc. IEEE High Performance Switching and Routing Conf. (HPSR)*, Cartagena, Spain, July 2011.

[19] http://googleenterprise.blogspot.com/2010/03/disaster-recovery-by-google.html.

[20] http://en.wikipedia.org/wiki/Recovery_point_objective.

[21] http://wikibon.org/wiki/v/Defining_RPO_and_RTO.

[22] L. Jiao, J. Li, T. Xu, and X. Fu, "Cost optimization for online social networks on geo-distributed clouds," in *Proc. Int. Conf. on Network Protocols (ICNP)*, 2012, pp. 1–10.

[23] https://developers.google.com/storage/pricing#_PricingExample.

[24] http://azure.microsoft.com/en-us/pricing/overview/.

[25] https://www.synaptic.att.com/clouduser/.

[26] http://aws.amazon.com/s3/pricing.

[27] http://searchstorage.techtarget.co.uk/tip/Cloud-storage-pricing-revealed-Hidden-costs-include-data-migration-and-access-fees.

[28] T. L. Weems, "How far is far enough," *Disaster Recov. J.*, vol. 16, no. 2, 2003.