Online Virtual Machine Evacuation for Disaster Resilience in Inter-Data Center Networks

Omran Ayoub, Amaro de Sousa, Silvia Mendieta, Francesco Musumeci and Massimo Tornatore

***Abstract*—With the risk of natural disaster occurrence rising globally, the interest in innovative disaster resilience techniques is greatly increasing. In particular, Data Center (DC) operators are investigating techniques to avoid data-loss and service downtime in case of disaster occurrence. In cloud DC networks, DCs host Virtual Machines (VM) that support cloud services. A VM can be migrated, i.e., transferred, across DCs without service disruption, using a technique known as "online VM migration". In this paper, we investigate how to schedule online VMs migrations in an alerted disaster scenario (i.e., for those disasters, such as tsunami and hurricanes, that grant an alert time to DC operators) where VMs are migrated from a risky DC, i.e., a DC at risk to be affected by a disaster, to a DC in safe locations, within a deadline set by the alert time of the incoming disaster. We propose a multi-objective Integer Linear Programming (ILP) model and heuristic algorithms for efficient online VMs migration to maximize number of VMs migrated, minimize service downtime and minimize network resource occupation. The proposed approaches perform scheduling, destination DC selection and assign route and bandwidth to VM migrations. Compared to baseline approaches, our proposed algorithms eliminate service downtime in exchange of an acceptable additional network resource occupation. Results also give insights on how to calculate the minimum amount of time required to evacuate all VMs with no service downtime. Moreover, since the proposed approaches exhibit different execution times, we design an 'alert-aware VM evacuation' tool to intelligently select the most suitable approach based on the number and size of VMs, alert time and available network capacity.**

***Index Terms*—Data center networks, Disaster resilience, Virtual machine migration.**

I. Introduction

Several recent weather-based disasters around the world had very negative impacts on cloud networks, causing Data Center (DC) shutdown, consequent data-loss and intolerable downtime of cloud services. For example, in 2012, hurricane Sandy flooded New York taking down a significant number of DCs for days, causing permanent damage in users' data and temporary service disruption. In 2011, the Great East Japan Earthquake and Tsunami caused progressive failures to Sendai data center leaving 30% of its customers unreachable [1]. The impact of such natural disasters led the International Telecommunication Union - Telecommunication Standardization Sector (ITU-T) to establish investigation groups focusing on disaster relief systems and new techniques to achieve resilient information and communication technology during disasters. With the rising risk of natural disasters [2], cloud DC operators, as well as researchers, have put the proactive disaster-resilient management of cloud networks on top of their agenda [3]–[5].

Omran Ayoub, Silvia Mendieta, Francesco Musumeci and Massimo Tornatore are with Politecnico di Milano, Italy. Amaro de Sousa is with Institute of Telecommunications, University of Aveiro, Aveiro, Portugal.

Corresponding author email: omran.ayoub@polimi.it.

Cloud networks are composed of a number of geographically distributed DCs interconnected by a communication network. They play an indispensable role in delivering latency-sensitive and bandwidth-hungry services to end users. These services run on Virtual Machines (VMs) hosted by DCs, where a single VM could support a cloud service [6]. As service disruption is a major concern, DC operators are investigating new approaches to avoid cloud-service downtime and virtualization offers an effective platform to avoid downtime. Thanks to virtualization, it is possible to perform "online" VM migration, i.e., the service is alive during VM migration. So, in case of a predictable disaster occurrence, one can live-migrate VMs from a DC in a risky zone, i.e., a DC located in zone at high risk to be affected by a disaster, without incurring any service downtime [7], [8].

In this paper, we investigate how to use inter-DC online VM migration for disaster resilience. In particular, we focus on the case of weather-based disasters such as hurricanes, floods, or tornadoes, where an alert can be issued before the occurrence of the event. An example of such case is the Great East Japan Earth- quake and Tsunami, the 9.0 Magnitude earthquake which hit 130 km into the western Pacific Ocean east of Sendai City, issuing a 20-minute alert before the resulting tsunami reached the coastline. During the interval of time granted by this alert (i.e., before a deadline set by the alert), several inter-DC online VM migration can be executed from the risky DCs towards safe DCs, with the objective of maximizing the number of VMs evacuated and avoiding service downtime. To maximize the number of VMs migrated within the deadline, a problem of routing and bandwidth assignment must be solved jointly with the problem of scheduling the VM migration (i.e., deciding the starting and ending time of a VM's migration). We refer to this problem as "Alert-based Online VM Migration for Disaster-Resilience" (Alert-VMmig). In our previous work [9], we proposed an Integer Linear Programming (ILP) model to solve this problem. In this paper, we enhance the ILP model and propose various heuristic approaches to investigate large instances of the problem. We also develop an 'alert-aware VM evacuation' tool which selects the most suitable heuristic approach based on the amount of time granted by the alert. The main contributions of the paper are as follows:

- We formally introduce the Alert-VMmig problem in inter-DC networks and propose a multi-objective ILP model to solve it. The ILP jointly decides whether to migrate a VM or not, the migration strategy (online or offline),

the routing and bandwidth assignment, destination DC selection and scheduling of VMs migrations such that the number of VMs migrated is maximized, the average service downtime is minimized and the overall network resource occupation due to VMs migration is minimized.
- We propose various heuristic approaches, namely 'Online RO-min', 'Online B-min' and a baseline approach, namely, 'Offline', to investigate large instances of the problem. Each of the heuristics exhibits different execution times with respect to the others. Since the execution time is also consuming part of the alert time, we develop an 'alert-aware VM evacuation' tool which intelligently selects the most suitable approach, among the various heuristics proposed, based on the deadline, number and characteristics of VMs and capacity of the links.
- We show that, for a given case characterized in number and characteristics of VMs (e.g., size and rate of which users access its data), available time until evacuation deadline and available capacity, there exists thresholds on minimum amount of time required to 1) evacuate all VMs, i.e., no data is lost and 2) evacuate all VMs online, i.e., no data is lost and service downtime is eliminated.

The rest of the paper is organized as follows. Sec. II provides background knowledge on VM migration. Sec. III formally states the problem and presents the proposed ILP formulation. Sec. IV presents heuristic approaches. Sec. V presents the alert-aware VM evacuation tool. Sec. VI discusses numerical results. Sec. VII draws the main conclusions.

II. Background on VM Migration

VM migration requires transferring all VM data, i.e., disk, memory and processors states, from a source server, i.e., the server in which the VM is currently hosted and running, to a destination server, i.e., the server which will host the VM after the migration is completed [10]. To perform VM migration, an amount of bandwidth must also be allocated along a path from the source to the destination server. The bandwidth assigned to perform the migration is referred to by *migration bandwidth* (B_{mig}) while the amount of time it takes to complete the migration is referred to as *migration duration* (T_{mig}).

Two classes of VM migration exist: 1) *Offline Migration* and 2) *Online Migration*. The *offline VM migration* [10] consists in halting the VM at its source server, transferring all its data (referred as VM size), and then re-activating it at destination server once migration is completed. Fig. 1(a) shows a scheme of an offline VM migration. The VM cannot be accessed while the offline VM migration process is taking place, and therefore the service it runs becomes unavailable to users, resulting in a service interruption or, in other words, *service downtime*, which roughly coincides with the *migration duration* of an offline migration [10]. For instance, given a VM with size V = 10 Gbit to be transferred with a migration bandwidth (B_{mig}) = 1000 Mbit/s, the migration duration is (T_{mig}) = $V/(B_{mig})$ = 10 seconds and, hence, the service downtime is T_{down} = 10 seconds. For some services, this 10-seconds service downtime is intolerable, which is a main shortcoming of offline migration.

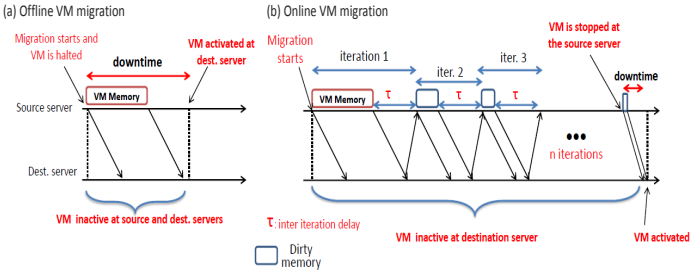


Fig. 1: A schematic representation of (a) offline and (b) online VM migration.

To shorten service downtime, *Online VM migration*, or live VM migration, has been proposed [7]. Online VM migration consists in transferring the VM while it is still running and being accessed by users [11]. During migration, the VM memory is "dirtied" (i.e., modified) due to users activity, and therefore additional information (the dirtied data) needs to be transferred in an iterative process, making use of a mechanism known as *pre-copy* [10], to make sure the VM at the destination server is synchronized with the VM at the source server. Hence, the migration duration not only depends on VM size and migration bandwidth such as in the offline VM migration, but also on the rate at which the VM memory gets dirtied. The parameter capturing this rate is called memory dirtying rate. The dirtying rate (Dr) might vary based on the type of VM, its hosted application, as well as the number of served users and their activity[1]. Figure 1(b) shows how online migration works. The first iteration is used to transfer the original VM memory to the destination, while the following iterations are used to transfer the "dirtied" memory, i.e., the memory blocks that were modified by user. The duration of each iteration depends on the amount of memory dirtied during the previous iteration and the migration bandwidth assigned. An inter-iteration delay τ is also shown in Fig. 1(b), which is due to end-to-end network delay, processing delay at either end, or a combination of both. The iterative copy phase stops when a specific stop condition is met. The stop condition could be a predefined number of iterations or a predefined amount of dirtied memory left to be transferred. In the final iteration, the VM is stopped at its source, remaining dirtied memory is copied, and then the VM is reactivated at the destination server. Consequently, downtime coincides only with the amount of time required to transfer the remaining dirtied data in the last iteration and to activate the VM at destination server and is therefore significantly reduced to an order of hundred milliseconds.

Due to the dirtying rate, a non-linear relationship between B_{mig}, the bandwidth assigned to migrate a VM, and T_{mig}, the migration duration, arises. This relationship is represented in the function-points curves shown in Fig. 2. The three curves in Fig. 2 correspond to the migration of a VM with size 10 Gbit and for different dirtying rate values, i.e., $Dr = 0$ (offline migration), $Dr = 50$ Mbit/s and $Dr = 100$ Mbit/s[2]. We used the model in [12] to obtain these curves, which

[1] Dr is usually expressed in number of dirtied pages per second (pages/sec) but here we refer to Dr in bit/s.

[2] Note that the VM migration process cannot converge in case the bandwidth assigned is less than the dirtying rate, which represents a lower bound on migration bandwidth.

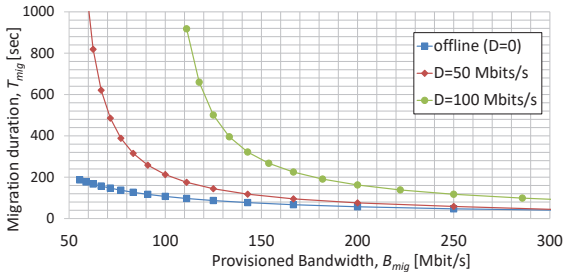


Fig. 2: Function points curves for a VM of size 10 Gbit and different dirtying rates: Dr = 0 (offline), Dr = 50 Mbit/s and Dr = 100 Mbit/s.

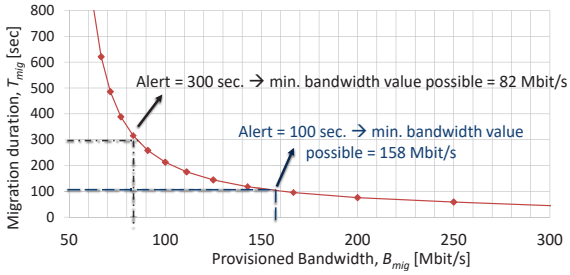


Fig. 3: Example of the impact of evacuation deadline on the set of possible bandwidth values to perform online migration of a VM of size 10 Gbit.

will be indicated as *function-points curves* (see Ref. [13] for a detailed explanation). On one hand, we note that, as bandwidth increases, the advantage of online migration in terms of total migration duration is reduced. On the other hand, reducing the migration bandwidth produces a drastic increase in the total migration duration, due to hardly meeting the stopping condition, e.g., longer time to reach an amount of dirtied memory left to be transferred below predefined threshold. Moreover, it is worth noting that the overall amount of data transferred in online VM migration, i.e., the product of migration bandwidth assigned and migration duration (in other words, the amount of network resources reserved for VM migration) significantly increases when assigning low bandwidth values for migration. For all these reasons, the assignment of bandwidth for online VM migration is not trivial. Moreover, note that, in a scenario where migration must end before a given deadline (in our case, the alert time), the assignment of migration bandwidth becomes more decisive.

Let us focus on the impact of the evacuation deadline on the function-points curve of a VM. Consider the example in Fig. 3, where a VM of size 10 Gbit is migrated subject to a dirtying rate of 50 Mbit/s. Here, the deadline imposes a lower bound on the bandwidth value chosen to perform the migration process. For example, for a deadline of 300 seconds, all values of migration bandwidth B_{mig} larger than or equal to 82 Mbit/s are possible, but, if deadline becomes even stricter, e.g., 100 seconds, then a smaller set of bandwidth values (compared to the case with a deadline of 300 seconds) can be assigned for migration, namely only bandwidth values larger than or equal to 158 Mbit/s. The deadline imposes strict constraints on the VMs migration process, especially when multiple VMs are involved, as the VMs share the available bandwidth, and therefore route and bandwidth assignments as well as the scheduling of different VMs become highly interdependent.

Due to 1) the lower bound imposed by the deadline on bandwidth values for VM migration and migration duration, 2) the limited amount of network capacity and of time available to migrate all VMs, 3) the dependency between bandwidth, routing and scheduling assignments of multiple VM migrations and 4) the non-linear relationship between T_{mig} and B_{mig}, efficient routing, bandwidth and scheduling assignments to VMs' migrations are needed, and are indeed decisive, to i) maximize number of VMs migrated (evacuated), ii) reduce downtime and iii) minimize network resource occupation[3] in presence of an alert of a disaster.

Several reactive data evacuation techniques for disaster resilience have been investigated. Refs. [14], [15] proposed to maximize data evacuated from DC located in a disaster zone to DCs in safe locations considering an evacuation deadline (i.e., time of occurrence of the disaster). Ref. [16] formulated an optimization problem to maximize the profit from emergency data backup in inter-DC networks considering a progressive disasters scenario. Moreover, Ref. [17] proposed disaster-aware algorithms to perform rapid data evacuation while minimizing the backup window, i.e., the amount of time required to evacuate all data. Note that, in these works, the amount of data to evacuate is known a priori (it is a given to the problem), while in our work, the total amount of data to evacuate is not known a priori, as it actually depends on the function-point (migration bandwidth and the respective migration duration) assigned to migrate each VM, which is a decision variable of the problem. From a methodological point of view, the fact that amount of data to evacuate is not known requires a drastic variation with respect to previously-proposed deadline-driven data evacuation techniques [15] and to existing maximum data flow algorithms [18]. Note that selecting the function point which minimizes the product of migration bandwidth and migration duration for one VM migration does not necessarily imply maximizing the number of VMs migrated, as the scheduling might result infeasible.

Other studies have proposed efficient routing and bandwidth assignment solutions for online VM migration with the aim of improving the network utilization [12], minimizing VM migration duration [19], energy efficiency [20] or minimizing the overall network resources [13]. Refs. [21], [22] proposed a method for live migration of VMs assuming quality-of-service constraints and Ref. [23] presented a theoretical analysis to estimate the bandwidth required to satisfy given constraints on the total migration time and the downtime of a single VM migration. Ref. [24] proposed efficient migration strategies for multiple VMs to minimize migration duration. Moreover, Refs. [25] and [26] proposed VM scheduling algorithms however not taking the routing and bandwidth assignment aspect into consideration. With respect to all these works, our problem has significant differences as we consider an evacuation deadline, which constrains the bandwidth assignment and the scheduling of VMs migration. Moreover, in our work, the migration process (involving the routing and bandwidth assignment and scheduling) needs to be optimized to maximize number of

[3]The amount of network resources occupied by a VM migration is the product of T_{mig} with B_{mig} and with the number of links of the migration routing path.

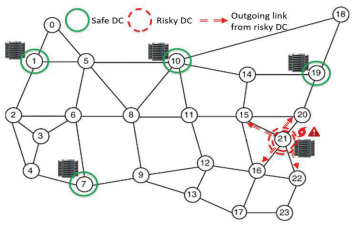


Fig. 4: USA24 network topology. DC nodes are highlighted.

VMs migrated within a given amount of time, an aspect that was not investigated previously.

III. Mathematical Modeling

A. Problem Statement

The Alert-VMmig problem is stated as follows. **Given** a physical wide area network topology (as in Fig. 4), a DC affected by a disaster, a set of candidate safe destination DCs (i.e., DCs not affected by the disaster), a set of VMs running at the affected DC (with their sizes and function-points curves) and an alert time (i.e., the evacuation deadline), we **decide** i) the VMs to migrate[4], ii) the routing path and migration bandwidth for each VM migration, iii) the scheduling (i.e., the migration starting time) and migration ending time[5] of each VM migration with the **objective** of, in decreasing order of priority, 1) maximizing number of migrated VMs[6], 2) minimizing sum of service downtime of all VMs (i.e., maximizing online VM migration) and 3) minimizing overall network resource occupation (RO). The RO of a VM migration is given by the product of migration duration, migration bandwidth and number of links of its routing path from source DC to destination DC. We consider that the same entity manages the network resources and the DC (i.e., storage and computing) resources, and therefore we assume the existence of an orchestrator capable of solving the Alert VMmig problem[7]. Note that the proposed Alert-VMmig model is flexible and can be to be extended to account for the case in which transport and DC resources are managed separately. In this case, the DC operator can still rely on a pre-allocated amount of transport resources for VM migration, even without a full knowledge of the physical transport network.

B. ILP Formulation

This section presents the proposed Integer Linear Programming (ILP) formulation to address the Alert-VMmig problem.

Sets:

- The network is represented by a set of nodes N and a set of links E.
- Set V represents the set of VMs hosted at the DC in risk.
- Set $R \subset N$ represents the set of safe DC nodes.
- Set P_r is set of paths from the DC in risk to each safe DC node $r \in R$. Each routing path $p \in P_r$ is characterized by its set of links E_p with a number of links $|E_p|$.
- $P = \bigcup_{r \in R} P_r$ is the set of all paths and P_e is the set of paths that include link $e \in E$ (we exclude from set E the links that are not included in any path $p \in P$).
- I_{vp} is the set of function-points for each path $p \in P$.
- Time is defined as a set T of discrete time slots, each one with a fixed duration of 1 second.

Parameters:

- c_e represents the capacity (in bits/second) of link $e \in E$.
- S_r and Q_r represent the storage capacity and the computing capacity of each safe DC node $r \in R$, respectively.
- S_v and Q_v represent the size and the computational requirements of of VM $v \in V$ in the DC in risk, respectively.
- Function-point $i \in I_{vp}$ of VM $v \in V$ is characterized by 3 parameters associated with the online mode and 3 parameters associated with the offline mode: the online bandwidth $b_v^{ip,on}$, the online migration time $d_v^{ip,on}$, the online downtime $\psi_v^{ip,on}$ (in online, $\psi_v^{ip,on} = 0$), the offline bandwidth $b_v^{ip,off}$, the offline migration time $d_v^{ip,off}$ (in offline, $d_v^{ip,off} = \frac{S_v}{b_v^{ip,off}}$) and the offline downtime $\psi_v^{ip,off}$ (in offline, $\psi_v^{ip,off} = d_v^{ip,off}$).
- $t \in T$ represents the ending instant of the t^{th} time slot and the ending time of the last slot is the alert time A.

Decision Variables:

x_v^{ipt} binary variable which is equal to 1 if VM $v \in V$ is migrated online through path p with function point $i \in I_{vp}$ (i.e., it is migrated with bandwidth $b_v^{ip,on}$ resulting in a migration time $d_v^{ip,on}$ and downtime $\psi_v^{ip,on}$) and ending its migration in time instant t (it means that it starts being migrated at the beginning of time slot $t - d_v^{ip,on} + 1$). Note that the ending time t associated with variable x_v^{ipt} cannot be shorter than the migration time $d_v^{ip,on}$. For mathematical clarity, all variables are used in the formulation proposed next but the variables x_v^{ipt} such that $t < d_v^{ip,on}$ are set to zero in the implementation.

y_v^{ipt} binary variable which is equal to 1 if VM $v \in V$ is migrated offline through path p with function point $i \in I_{vp}$ (i.e., it is migrated with bandwidth $b_v^{ip,off}$ resulting in a migration time $d_v^{ip,off}$ and downtime $\psi_v^{ip,off}$) and ending its migration in time instant t (it means that it starts being migrated at the beginning of time slot $t - d_v^{ip,off} + 1$). Again, the ending time t associated with variable y_v^{ipt} cannot be shorter than the migration time $d_v^{ip,off}$ and, for clarity, all variables are used in the formulation but the variables y_v^{ipt} such that $t < d_v^{ip,off}$ are set to zero in the implementation.

Objectives:

$$maximize \sum_{v \in V} \sum_{p \in P} \sum_{i \in I_{vp}} \sum_{t \in T} \left(x_v^{ipt} + y_v^{ipt} \right) \quad (1)$$

[4]A VM might not be migrated from the affected DC.

[5]Note that the ending time of a VM migration is determined when deciding the migration starting time and assigning the migration bandwidth, which determines the migration duration.

[6]We consider that all VMs have the same priority and therefore we focus on the objective of maximizing the total number of VMs migrated. We note that our proposed solutions can easily adapt to the case in which VMs are characterized by different priorities as discussed later.

[7]Technologies for orchestration in inter-DC wide area network jointly managing transport and computing resources are available [27], [28]).

$$minimize \sum_{v\in V}\sum_{p\in P}\sum_{i\in I_{vp}}\sum_{t\in T}\left(\psi_v^{ip,on}x_v^{ipt}+\psi_v^{ip,off}y_v^{ipt}\right) \quad (2)$$

$$minimize \sum_{v\in V}\sum_{p\in P}\sum_{i\in I_{vp}}\sum_{t\in T}\left(b_v^{ip,on}d_v^{ip,on}|E_p|x_v^{ipt} + b_v^{ip,off}d_v^{ip,off}|E_p|y_v^{ipt}\right) \quad (3)$$

Objective (1) is the maximization of the number of migrated VMs[8], Objective (2) is the minimization of the total service downtime and Objective (3) is the minimization of the network resource occupation (RO).

Subject to:

$$\sum_{p\in P}\sum_{i\in I_{vp}}\sum_{t\in T}\left(x_v^{ipt}+y_v^{ipt}\right)\leq 1, v\in V \quad (4)$$

$$\sum_{v\in V}\sum_{p\in P_e}\sum_{i\in I_{vp}}\left(\sum_{\tau=t}^{min(A,t+d_v^{ip,on}-1)}(b_v^{ip,on}x_v^{ip\tau}) + \sum_{\tau=t}^{min(A,t+d_v^{ip,off}-1)}(b_v^{ip,off}y_v^{ip\tau})\right)\leq c_e, \quad e\in E, t\in T \quad (5)$$

$$\sum_{v\in V}\sum_{p\in P_r}\sum_{i\in I_{vp}}\sum_{t\in T}\left(S_v(x_v^{ipt}+y_v^{ipt})\right)\leq S_r, r\in R \quad (6)$$

$$\sum_{v\in V}\sum_{p\in P_r}\sum_{i\in I_{vp}}\sum_{t\in T}\left(Q_v(x_v^{ipt}+y_v^{ipt})\right)\leq Q_r, r\in R \quad (7)$$

Constraints (4) guarantee that each VM $v \in V$ can by migrated at most once on one routing path $p \in P$, using a function-point $i \in I_{vp}$ and ending on one single time instant $t \in T$. Constraints (5) guarantee that at each link $e \in E$ and each time slot $t \in T$, the total bandwidth of the VMs being migrated on a routing path using link e (i.e., paths $p \in P_e$) is not higher than the link capacity. Constraints (6) guarantee that the total storage size of the VMs migrated to each safe DC $r \in R$ is not higher than the available DC storage capacity. Similarly, constraints (7) guarantee that the total computational needs of the VMs migrated to each safe DC $r \in R$ is not higher than the available DC computational capacity.

To solve the Alert-VMmig problem, we start by optimizing Obj. (1). Then, Obj. (2) is optimized only when the optimal solution value of Obj. (1) is $|V|$, i.e., all VMs can be migrated. To optimize Obj. (2), we replace constraints (4) by

$$\sum_{p\in P}\sum_{i\in I_{vp}}\sum_{t\in T}\left(x_v^{ipt}+y_v^{ipt}\right)=1, v\in V \quad (8)$$

[8]In case VMs are of different priorities, a parameter for each VM indicating its priority can be added. Consequently, the objective function is modified such as to maximize sum of priorities of all VMs evacuated instead of their number.

Tab. I: Overview of the three heuristic approaches.

	Obj. 1: Maximize #VMs migrated	Obj. 2: Minimize Downtime	Obj. 3: Minimize RO	Scheduling
Online RO-min	✓	✓	✓	✓
Online B-min	✓	✓	X	X
Offline	✓	✓	✓	✓

to guarantee that all VMs are migrated while minimizing the total service downtime. Then, Obj. (3) is optimized only when the optimal solution value of Obj. (2) is 0, i.e., all VMs can be migrated online (in the computational results, we consider that the downtime of online migrations is negligible and, so, we have set $\psi_v^{ip,on} = 0$ in all cases). To optimize Obj. (3), besides using constraints (8) instead of constraints (4), we eliminate variables y_v^{ipt} from the model as they are not needed (i.e., all VMs can be migrated online) reducing in this way the total number of variables to half.

IV. Heuristic Approaches

To address the scalability limitation, in terms of run-time, of the ILP approach described Sec. III, we propose two heuristic approaches, namely, *Online Minimum Resource Occupation (Online RO-min)* and *Online Minimum Bandwidth (Online B-min)*. In addition, we develop an *Offline VM Migration* approach referred to as *Offline*[9]. Before describing the approaches in detail, we give a general overview of them in Tab. I.

Online RO-min (described in detail in Sec. IV-A) performs online migration and focuses on the three objectives of the problem, i.e., on maximizing the number of VMs migrated, minimizing VMs' downtime and minimizing network RO. To efficiently minimize network RO, *Online RO-min* searches for the function-point that leads to minimum resource occupation and performs scheduling and, therefore, as we will discuss more in Sec. VI, it requires higher computational time with respect to *Online B-min* and *Offline*. The *Online B-min* strategy (described in detail in Sec. IV-B) serves as a benchmark strategy and it performs online migration focusing on the first two objectives only, i.e., on maximizing the number of VMs migrated and minimizing downtime. *Online B-min* assigns the minimum possible bandwidth value, considering evacuation deadline, for the online migration of a VM. Therefore, as the migration process can be executed until evacuation deadline, *Online B-min* does not perform any scheduling, which translates into an advantage in terms of computational time. In the *Offline* migration benchmark strategy, all VMs are migrated in an offline manner. *Offline* assigns high bandwidth values for the migration process such as to minimize migration duration of each VM and therefore minimize the service downtime. Note that for *Offline*, the amount of data transferred per VM equals the size of the VM and, given that the bandwidth assignment is already set such as to minimize downtime, minimizing overall RO translates into migrating VMs along shortest paths between DC in danger zone and DCs in safe locations.

[9]Note that *Offline* is, like other approaches, a proactive approach and 'offline' refers to the migration of VMs and not the methodology.

A. Online RO-min Algorithm

Figure 5 shows the flow chart of *Online RO-min* algorithm. First, K shortest paths between DC in danger zone and DCs in safe locations are inserted in list I and sorted in increasing order of number of hops. Then, VMs are sorted in decreasing order of i) size and ii) dirtying rate[10]. Then, the points of the function-points curve for each VM having a T_{mig} less than evacuation deadline are sorted in list P_v in increasing order of the product of B_{mig} and T_{mig}, i.e., in increasing order of the amount of network resource occupation required per link to perform online migration of VM v. Then, for each VM v, we iterate over the list of function points, P_v, and list of shortest path, I, to check if a path capable of accommodating the online migration of VM v is available and if destination DC can host VM v. If possible, resources are reserved for VM v and next VM is considered. Note that scheduling is performed such that the migration of a VM is set to start at earliest time possible. If no path can accommodate an online migration of VM v, either to lack of bandwidth or time, the VM is migrated offline assigning the path and bandwidth that provide minimum downtime[11]. If no path can accommodate the offline migration of VM v, the last VM that was set to be migrated online is set to be migrated offline, releasing network resources and making room (bandwidth and time) to migrate VM v offline. In case all previously considered VMs are already set for offline migration and no network resources are available to accommodate the VM under consideration, *Online RO-min* initiates an offline migration procedure to maximize number of VMs migrated offline. Finally, the approach terminates when iterating over all VMs.

Time complexity: *Online RO-min* first uses Dijsktra algorithm to find the K shortest paths between DC in risk and the safe DCs, resulting in a complexity of $O(NlogN)$ in a network of N nodes. *Online RO-min* algorithm further iterates over all VMs V, number of paths available between DC in risk and safe DCs P, starting instant of time to migrate VM T, and over all function-points F to perform VM migration, resulting in a complexity of $O(V \cdot P \cdot T \cdot F)$. The time complexity of Online RO-min is therefore in the order of $O(V \cdot P \cdot T \cdot F)$.

B. Online B-min

Online B-min performs similarly to *Online RO-min*, however, instead of searching for the function-point which minimizes RO, it simply assigns the minimum bandwidth possible for the online migration process of a VM. Doing so, *Online B-min* makes use of all available time (from the moment the migration process starts until the evacuation deadline) to migrate a VM and therefore scheduling is not taken into consideration, which reduces complexity and allows to significantly reduce execution time. As shown in Tab. I, *Online B-min* does not aim to reduce network RO but only focuses on the first two objectives. As we will see later, this allows to save on execution time and therefore provide more time for the evacuation process.

[10]For the case in which VMs are characterized by priorities (i.e., a priority parameter), the sorting can be changed imposing a decreasing order of the ratio of priority and amount of network resources required for migration. Following this order, the heuristic approaches migrate VMs while maximizing the sum of priorities of VMs migrated.

[11]To save on execution time, we set a flag that checks whether online migration is feasible for subsequent VMs and, when not possible, VMs are directly considered for offline migration instead of being checked for online migration first.

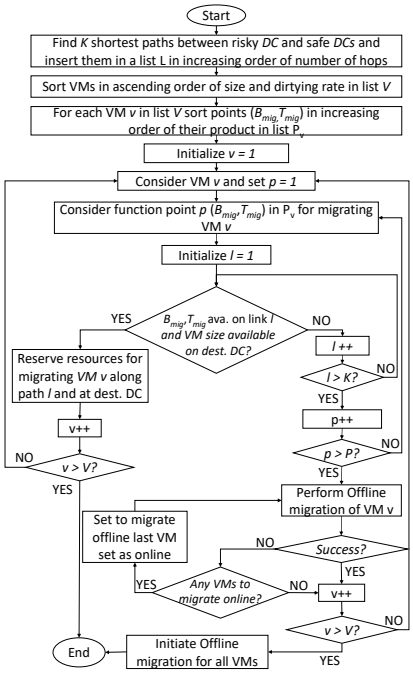


Fig. 5: Flowchart of the *Online RO-min* heuristic approach.

V. Alert-Aware VM evacuation Tool

The execution time of each heuristic approach needs to be considered as a part within the time interval since receiving the disaster alert and the evacuation deadline, as depicted in Fig. 6. Hence, it is decisive, in an alert-based disaster scenario and for a given case study, to consider the approach which mostly optimizes the objectives discussed in Sec. III. Note that a case study is defined by the number of VMs, their sizes and dirtying rate, the location of DC in risk in the network, the number of outgoing links from DC and their capacity, location of safe DCs and a predefined alert time.

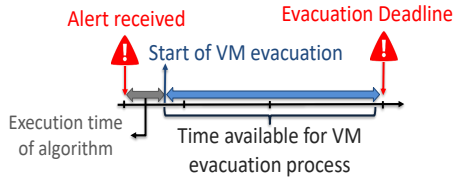


Fig. 6: Schematic representation highlighting the effect of the execution time on the start of the migration process.

For this aim, we develop an *Alert-Aware VM evacuation* tool which makes use of a pre-collected database of execution times of the heuristic approaches in various problem instances simulated before the time of the disaster[12] and calculates the ideal amount of data required to perform the migration of all VMs for each approach (referred to by G and is explained

[12]The larger is the database, i.e., the more problem instances are simulated before occurrence of disaster, the more accurate is the estimation of execution times and therefore the selection of most convenient heuristic approach.

Tab. II: An example of the comparison performed by *Alert-aware VM evacuation* tool considering 1000 VMs each of S_v = 10 Gbit and dirtying rate of 100 Mbit/s to evacuate from a DC with l = 3 outgoing links each of capacity C = 100 Gbit/s and a deadline of 50 seconds.

Approach	Exec' (sec)	D' (sec)	G (Gbit)	MaxData (Gbit) [Eqn. 11]
Offline	6	44	[Eqn. 8] 10000	13200
Online B-min	10	40	[Eqn. 9] 20000	12000
Online RO-min	15	35	[Eqn. 10] 12000	10500

mathematically later) and then compares it to the maximum possible amount of data that can be evacuated within the deadline ($MaxData$). Based on comparing the G value of each of the approaches with $MaxData$, the alert-aware VM evacuation tool decides which approach to use. The ideal amount of data required to perform the migration of all VMs is defined differently for each the approaches. For *Offline*, G_{Off} is defined as the sum of the sizes of all VMs

$$G_{Off} = \sum_{v} S_v, \tag{9}$$

where S_v is the size of VM v. For *Online B-min*, G_{Bmin} depends on the evacuation deadline as *Online B-min* assigns the minimum bandwidth value possible for a VM ($B_{min,v}$) and therefore it can be calculated as follows:

$$G_{Bmin} = \sum_{v} B_{min,v} \cdot T^{min}_{mig,v}, \tag{10}$$

where $T^{min}_{mig,v}$ is the migration duration for $B_{min,v}$. For *Online RO-min*, G_{ROmin} is calculated as follows:

$$G_{ROmin} = \sum_{v} B_{RO,v} \cdot T^{RO}_{mig,v}, \tag{11}$$

where $B_{RO,v}$ and $T^{RO}_{mig,v}$ correspond to the function point which provides the lowest provide of $B_{mig,v}$ and $T_{mig,v}$ such that $T_{mig,v}$ is lower than evacuation deadline. Finally, we calculate *MaxData* as follows:

$$MaxData = l \cdot C \cdot D, \tag{12}$$

where l is the number of outgoing links from DC in danger zone, C is link's capacity and D is evacuation deadline.

An example of the comparison that can be established is shown in Tab. II. First, we see that each of the approaches require a different execution time and, therefore, have a unique new deadline (denoted by D'). Consequently, each of the approaches have a different value of $MaxData$. Comparing G and $MaxData$ for each of the approaches provides intuition on which approach to use. For instance, the example shows that online VM migration approaches require more capacity to evacuate all VMs online (G is greater than $MaxData$) while *Offline* seems more feasible as $G_{offline}$ is lower than $MaxData$.

Fig. 7 shows how the *Alert-Aware VM evacuation* tool works. For each heuristic approach, *Alert-Aware* estimates the minimum execution time $Exec$ considering the number of VMs referring to the data base of execution times. Then, it calculates the new deadline D' and estimates the minimum execution time corresponding to deadline D', $Exec'$. If $Exec'$ is less than D - D', meaning that the amount of time reserved for execution is less than that deducted from the actual deadline, *Alert-Aware* calculates the required amount of data

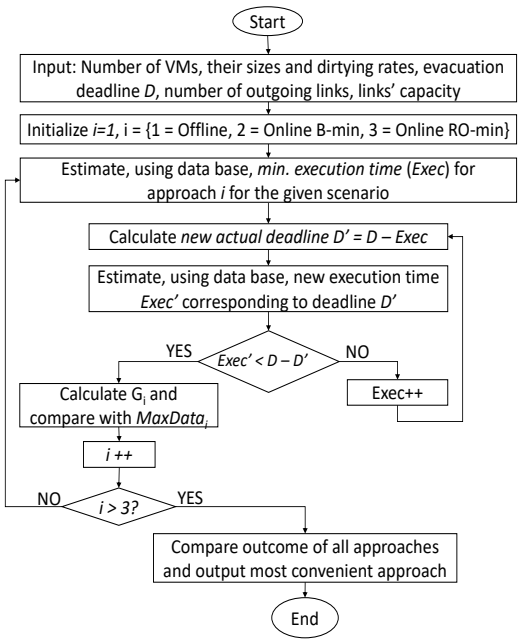


Fig. 7: Flowchart of the alert-aware VM evacuation tool.

for approach i to perform all VM migration, G_i, and compares it with $MaxData_i$. After iterating over all approaches, *Alert-Aware* gives as an output the approach of which G is less than $MaxData_i$. If more than one approach satisfies this inequality, *Alert-Aware* gives priority to 1) *Online RO-min*, 2) *Online B-min* and 3) *Offline*. If $Exec'$ is equal or higher than D - D', $Exec$ is incremented and then D' and $Exec'$ are re-calculated until $Exec'$ lower than D - D' is found.

VI. Numerical Results and Discussion

We evaluate the performance of the ILP and heuristic approaches in terms of the following metrics:

- Total number of VMs migrated (both online and offline).
- Overall downtime (seconds). The overall downtime of all VMs migrated. A downtime of zero seconds implies that all VMs are migrated online.
- Average network RO per VM migration (RO_{avg}), defined as the average amount of network resources used to migrate a VM and calculated as the product of T_{mig}, B_{mig} and number of hops traversed to reach dest. DC.
- Algorithm execution time. Amount of time taken to provide a solution.

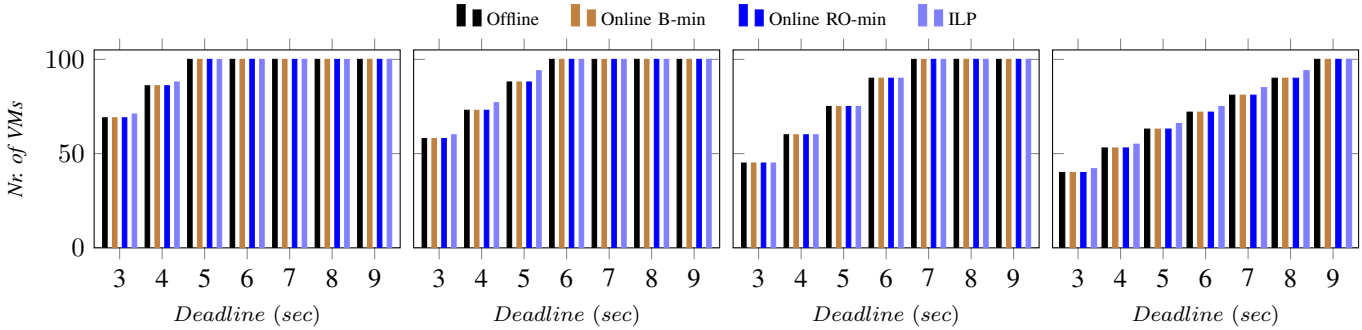
The topology considered in this study is the USA-24 network shown in Fig. 4, constituted by |N| = 24 nodes and |E| = 43 bidirectional links, each with 100 Gbit/s capacity[13]. We consider 5 DC locations, of which one is affected by a weather-based disaster and 4 serve as candidate DC locations[14] for VMs (highlighted in Fig. 4).

A. Validation of Heuristic Approaches

We validate the heuristic approaches comparing their performance to that of the ILP model in four case studies. Due to the

[13]We assume this capacity is reserved for the migration process.

[14]Note that we consider that each candidate DC location can host as many VMs as can be migrated to it.



(a) Case study # 1 (b) Case study # 2 (c) Case study # 3 (d) Case study # 4

Fig. 8: Number of VMs evacuated for increasing evacuation deadline (alert) for the ILP model and heuristic approaches.

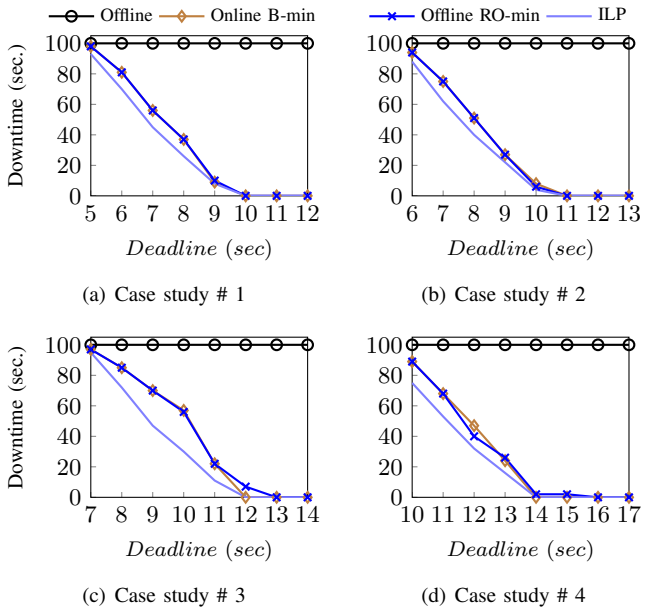
complexity of the problem and the computational limitations of the ILP model, we consider in each case study only 100 VMs and an evacuation deadline ranging between 3 and 20 seconds. Note that, in each case study, we consider function points that permit VM evacuation within the evacuation deadline. 10 combinations of VM size and dirtying rate are considered in each of the case studies, as follows:

- case 1: VMs size = $\{10, 12, 14, 16, 18\}$ Gbit. Dirtying rate = $\{200, 500\}$ Mbit/s.
- case 2: VMs size = $\{14, 15, 16, 17, 18\}$ Gbit. Dirtying rate = $\{200, 500\}$ Mbit/s.
- case 3: VMs size = $\{20, 22, 24, 26, 28\}$ Gbit. Dirtying rate = $\{200, 500\}$ Mbit/s.
- case 4: VMs size = $\{32, 34, 36, 38, 40\}$ Gbit. Dirtying rate = $\{200, 500\}$ Mbit/s.

These small case studies (realistic numbers of VMs in a DC are in the order of thousands) are for heuristic-validation purposes. In these cases, we neglect the impact of the execution time on the deadline, as the aim is to compare solutions obtained by the heuristic approaches to optimal solution obtained by ILP. Following sections discuss larger problem instances.

Objective 1: Number of migrated VMs Figure 1 shows the number of VMs migrated with respect to the alert time for the 4 different case studies described above. First, as expected, in all case studies, the number of VMs migrated increases with larger alert time until a specific value of alert time (depending on the case study) where all VMs are migrated. For example, for case study 1, all VMs are migrated assuming an alert time of 5 seconds while, for case study 4, all VMs can be only migrated at an alert time of 9 seconds or higher. This shows that, given a case study in terms of number of VMs, their size and network capacity, there exist a threshold on the minimum amount of time required to perform migration for all VMs from a DC affected by a disaster. Comparing the performance of the heuristic approaches to that of the ILP model, results show that in most cases heuristics' and ILP's performances coincide, while in very few cases one or two VMs less are migrated by the heuristic approach, yielding a maximum optimality gap of 2%.

Objective 2: Average Downtime Figure 9 plots the average downtime as a function of the alert time for all the approaches. Note that, in each case study, we show results with minimum



(a) Case study # 1 (b) Case study # 2

(c) Case study # 3 (d) Case study # 4

Fig. 9: The overall downtime (seconds) with respect to value of evacuation deadline for the ILP model and heuristic approaches.

value of the alert time which enables the migration of all the 100 VMs. In other words, the values of the second objective (minimizing downtime) are shown for alert time values when the first objective (migrating all VMs) is fully optimized. In all case studies, and for all online strategies, the average downtime starts off at a relatively high value, between 70 and 95 seconds, meaning that most of the VMs are migrated offline, and then decreases until it reaches 0 seconds, which means that all VMs are migrated online, at a specific value of alert time, showing that there exist a threshold on the minimum amount of time required to perform online migration for all VMs in a given case study. The threshold differs from one case study to another depending on the number of VMs, their size, dirtying rate and network capacity. Comparing the performance of *Online RO-min* and *Online B-min* to that of the ILP, we see that *Online RO-min* and *Online B-min*, in some cases, e.g., for an alert time of 6 and 8 seconds for case study 1 and an alert time of 7 and 10 for case study 4, shows higher average downtime up to 1 second more with respect to that of the ILP (less than 10% of optimality gap). This difference decreases as the alert time increases until it reaches 0% for, e.g., alert time higher than 10 seconds in case study 1 and

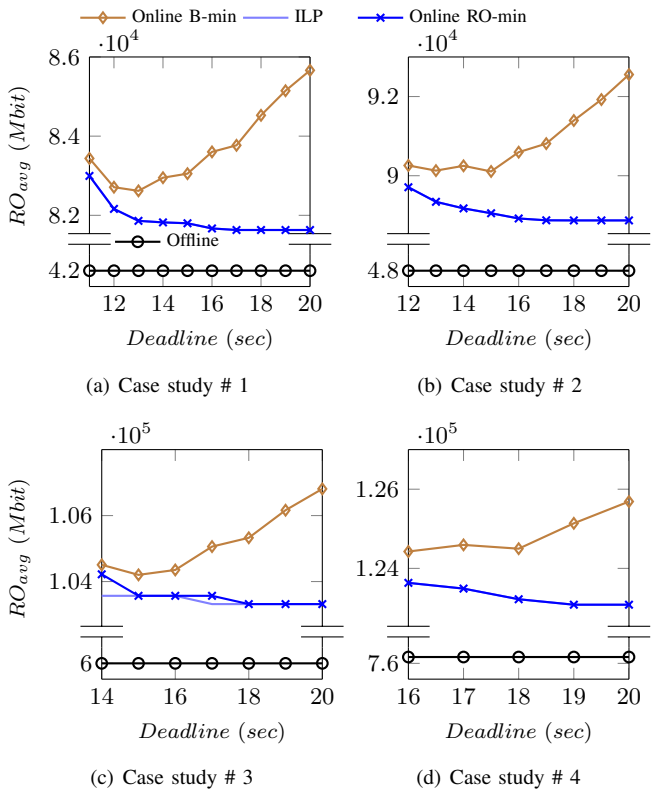


Fig. 10: Average network resources occupation per VM migration in function of alert time for the ILP model and heuristic approaches.

higher than 16 seconds in case study 4. For *Offline*, overall downtime does not decrease since all VMs are migrated offline *Offline* assigns always the highest bandwidth values to reduce migration duration.

Objective 3: Network Resource Occupation Figure 10 plots the average network RO per VM migration with respect to alert time for the four approaches[15]. Results show that, in all cases, *Online RO-min* approaches the optimal (ILP-based) solution, as RO_{avg} decreases for higher values of alert time. This is because for higher values of deadline, more function-points become available, i.e., their migration duration is lower than evacuation deadline, and therefore both approaches utilize function-points which result in lower network RO. On the contrary, *Online B-min* utilizes more network RO for higher values of deadline. This is because *Online B-min* considers function-points with minimum bandwidth values possible, i.e., function-points with migration duration slightly lower than the evacuation deadline, which does not necessarily minimize network RO. In fact, *Online B-min* is not designed to minimize network RO (see Tab. I) but to maximize the overall number of VMs migrated and minimize downtime.

We now compare RO_{avg} of the ILP model and of *Offline* to quantify the trade-off between downtime and network RO (y-axis in each of the graphs in Fig. 10 is truncated for better graphical representation). Results show that the extra amount of resources required to perform online migration ranges, for the cases considered, from 57% (case 4) to 100% (case 1). The percentage of additional network resources varies depending on VMs sizes and their dirtying rates. Specifically, when the dirtying rate is high as in case 1 (dirtying rate is 25% of VM size on average), the additional percentage of network resources required is high. In the cases in which dirtying rate makes less percent of VM size as in cases 3 and 4 (around 13% and 9%, respectively), percent of additional network resources to perform online migration decreases to 73% and 57%, respectively. This gives an intuitive idea, in a disaster-resilient scenario, of the additional amount of resources required to perform online VM migration and eliminate service downtime.

Execution Time: In terms of execution time, the ILP model takes up to 900 seconds[16], in some cases, to provide a solution, whereas the heuristic approaches, for the above considered case studies, have an execution time of around 1 second. Note that for larger instances of the problem, the heuristic approaches require much larger execution time (e.g., in the order of tens of seconds). Specifically, *Online RO-min* approach consumes more alert time to provide a solution than *Online B-min* and *Offline* where *Offline* has the lowest execution time. A more detailed analysis of the execution time will be discussed in the following sections.

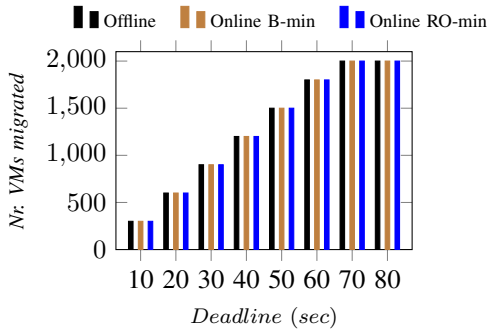
B. Larger Problem Instances

We now compare the performance of the three heuristic approaches over a larger problem instance, i.e., considering 2000 VMs each of size 10 Gbits and dirtying rate of 200 Mbps for an alert time ranging between 10 and 150 seconds with a step of 10 seconds. We consider the network topology reported in Fig. 4 and perform, for each value of evacuation deadline, 5 simulations considering in each a different DC in danger zone and report averaged results. We assume that each candidate destination DC can host up to 500 VMs.

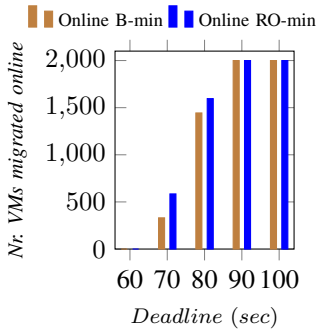
Figure 11(a) plots the total number of VMs migrated with respect to deadline for all the three heuristic approaches. First, we note that all approaches provide the same number of migrated VMs regardless of the alert time. Specifically, for a relatively short alert, e.g., 10 seconds, only 300 VMs are migrated. The number of VMs migrated then increases as the evacuation deadline is longer until all 2000 VMs are migrated at a deadline of 70 seconds. This confirms that for a given number of VMs, their sizes and dirtying rates as well as the number of outgoing links and their capacity, there exist a minimum threshold on the amount of time needed to migrate all VMs. In Fig. 11(b) we focus on the total number of VMs migrated online with respect to deadline for the online approaches, *Online B-min* and *Online RO-min*. First, we note that for evacuation deadlines up to 60 seconds, none of the VMs were specifically migrated online. Indeed, for such values of evacuation deadline, not all the VMs were guaranteed migration (as shown in Fig. 11(a)) and therefore both approaches perform offline migration such

[15]Note that also in this case we show results for values for which all the VMs are migrated online.

[16]The fact that for small problem instances the execution time of the ILP is in the order of minutes suggests that in a practical scenario, with tens of thousands of VMs to be migrated, the ILP cannot be applied, as the complexity of the problem does not allow it



(a) Nr. VMs migrated vs. Deadline

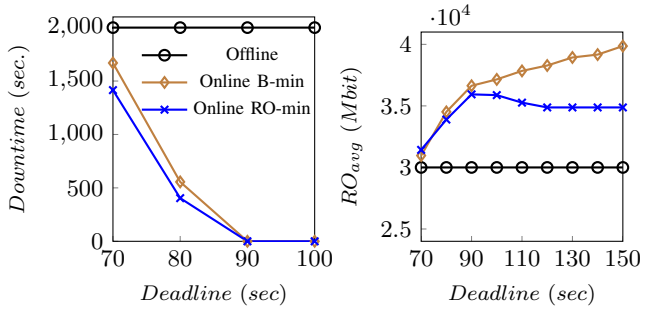


(b) Nr. VMs migrated online vs. Deadline

Fig. 11: The number of VMs evacuated (a) and (b) number of VMs evacuated online with respect to evacuation deadline for the heuristic approaches.

as to completely fulfill the first objective, i.e., maximizing the number of VMs migrated. For an evacuation deadline of 70 seconds, after guaranteeing migration (evacuation) of all VMs, both approaches migrate some of the VMs online. In particular, *Online RO-min* migrates more VMs online (a total of 585) than *Online B-min* which migrates 331. Similarly, for an evacuation deadline of 80 seconds, *Online RO-min* manages to migrate 1596 VMs online while *Online B-min* migrates 1444 VMs online. This is because *Online RO-min*, when performing online migration, assigns bandwidth values which minimize the overall resource occupation thus leaving capacity to migrate more VMs online than *Online B-min*. Finally, for an evacuation deadline of 90 seconds, both strategies migrate all 2000 VMs online thus achieving minimum downtime. This shows that there exist a threshold, given number, size and dirtying rate of VMs and number of outgoing links and their capacity, on the minimum amount of time needed to completely perform online migration and achieve minimum service downtime.

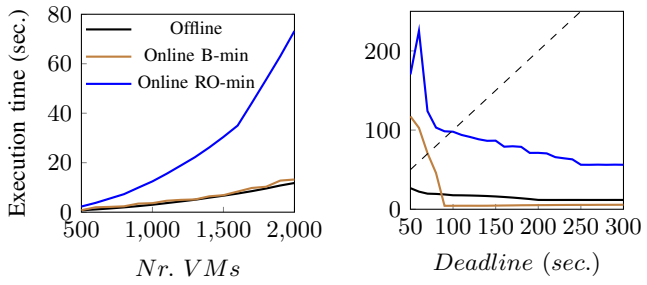
The impact of migrating VMs online on service downtime is shown in Fig. 12(a) that plots the overall downtime with respect to evacuation deadline. Results show that, for values of evacuation deadline of 70 and 80 seconds, *Online RO-min* achieves lower service downtime than *Online B-min* and then, for 90 seconds evacuation deadline, both approaches can achieve 0 second service downtime. Finally, we plot in Fig. 12(b) the average resource occupation per VM with respect to the evacuation deadline for the three approaches. Note that for *Offline* the average RO remains constant as the actual amount of data transferred in offline migration remains



(a) (b)

Fig. 12: Overall downtime (a) and (b) average network RO per VM migration with respect to the value of evacuation deadline.

the same. We show the plot starting from an evacuation deadline of 70 seconds, in which all VMs are migrated (as seen from Fig. 11(a)). Results show that between evacuation deadlines of 70 and 90 seconds, the average RO increases for *Online B-min* and *Online RO-min*. This is because both approaches are migrating more VMs online and therefore utilize more network resources for the migration. Starting from an evacuation deadline of 90 seconds, i.e., the moment after which all VMs are migrated online, the average RO for *Online RO-min* slightly decreases for higher values of evacuation deadline while it increases for *Online B-min*. This is due to the fact that for longer evacuation deadlines, *Online RO-min* is capable of assigning function points (B_{mig},T_{mig}) that permit minimizing average network RO. This advantage is achieved at the cost of a higher execution time due to scheduling performed by *Online RO-min*. For *Online B-min*, however, the average RO continues on increasing for higher values of evacuation deadline. This is because *Online B-min* assigns minimum bandwidth value possible for the online migration even if it occupies more network resources due to longer migration duration. As previously mentioned, *Online B-min* does not perform scheduling and therefore, as we will see in next section, saves on execution time. Finally, comparing the resource occupation of online strategies to that of *Offline*, results show that *Online B-min* and *Online RO-min* utilize up to 40% and 15% more network resources, respectively. This shows that there exist trade-off between minimizing the downtime and network RO that, for *Online RO-min*, can be considered acceptable. It is important to highlight that this trade-off highly depends on the function-point curve of the VMs, i.e., on the size and the dirtying rate of the VM. It is worth noting, although we do not report fully detailed numerical analysis, that the location of the DC in risk in the network and the number of outgoing links of the DC has a direct impact on the evacuation process. For instance, in the case in which the location of DC in risk is central (e.g., if at node 10 in Fig. 4), less time is eventually required to evacuate all VMs and eliminate service downtime due to the more links (and therefore paths) reaching the risky DC than in the case in which the location of DC in risk is peripheral (e.g., if at nodes 1 or 19). This behaviour could not be directly observed in Figs. 8 and 9, where, for sake of conciseness, we have reported only averaged results of multiple evaluations while changing location of DC in risk in the network.



(a) (b)

Fig. 13: Execution time of the three heuristic approaches while varying (a) number of VMs and (b) evacuation deadline.

C. Analysis on Execution Time

We investigate the impact of 1) number of VMs and 2) deadline on the execution time of each of the heuristic approaches (Fig. 13). Fig. 13(a) shows the execution time (in seconds) for the various approaches varying the number of VMs from 500 to 2000 for an evacuation deadline of 200 seconds. Execution time increases progressively for greater number of VMs for all approaches. In particular, *Offline* and *Online B-min* show comparable and relatively low execution time. Conversely, *Online RO-min* has a significantly higher execution time in all cases. This is because *Online RO-min* extensively performs scheduling to further minimize overall network RO and therefore consumes more time whereas *Offline*, which does not perform online migration, and *Online B-min*, which does not perform scheduling, save significant amount of time. Moreover, Fig. 13(b) shows the execution time for the various approaches while varying the evacuation deadline from 50 to 300 seconds for the case with 2000 VMs. In Fig. 13(b) we also plot a straight dashed line representing the *Deadline*. Results show that, for *Online B-min* and *Online RO-min*, and for relatively low values of evacuation deadline, the execution time exceeds the deadline, meaning that the two online migration strategies cannot be considered. We also notice that the difference of execution time of the three approaches, in most of the cases, is significant and can directly affect the decision on which strategy to use for VMs migration. Moreover, the execution times of all approaches decrease as the value of evacuation deadline increases. This is because a larger evacuation deadline means that more data can be evacuated, making it easier for the heuristic approaches to provide a solution in less time. Note that it is possible to estimate an upper bounds, or safe estimations, of the execution time of a given instance by considering similar instances with larger number of VMs that are known to require larger execution time. For instance, consider the case with 1500 VMs with *Online RO-min*, which shows an execution time of 40 seconds; In case a different problem instance with the same characteristics but lower number of VMs, a 40-seconds execution time can be considered an upper bound on the execution time, and therefore a safe estimation *Online RO-min*.

To clarify further this concept, we show an example of the effect of execution time on the choice of the approach to use in Tab. III considering 2000 VMs (each of size 10 Gbit and dirtying rate 200 Mbps) and an alert of 200 seconds. First, we see that each of the approaches has a different execution time than the other approaches, 6 seconds for *Offline*, 3 seconds for *Online B-min* and 26 seconds for *Online RO-min*. Consequently, each approach has a different amount of time remaining to perform the migration (which we refer to in the table as deadline representing the amount of time from a solution is provided to perform VM migration until the time disaster hits), 194 seconds for *Offline*, 197 seconds for *Online B-min* and 173 seconds for *Online RO-min*. Comparing the results obtained for the three approaches, we see that, on one hand, *Online RO-min* migrated only 1702 out of 2000 (*Objective 1*), which rules it out from being the best approach to consider in this case. On the other hand, both *Offline* and *Online B-min* succeeded in migrating all 2000 VMs but *Online B-min* results in a lower total downtime (1840 seconds) than *Offline* (2000 seconds), making *Online B-min* the best choice of approach in this case. Note that *Objective 3* (minimize overall network RO) is omitted in Tab. III as none of the approaches was able to optimize *Objective 2* (i.e., minimizing downtime to zero).

D. Alert-Aware VM evacuation Tool: Results

This section discusses further results comparing the performance of the heuristic approaches while taking their execution time into consideration.

We test the alert-aware VM evacuation tool for a case study of 2000 VMs with dirtying rates of 200 and 500 Mbps. Tab. IV reports the estimated execution time of each of the heuristic approaches, results obtained while considering the execution time of each heuristic approach, and decision on the most convenient approach to consider taken by the alert-aware VM evacuation tool for several values of evacuation deadline. For example, for an alert of 50 seconds, *Offline* provided the best performance as it migrates all 2000 VMs (while online approaches did not manage to migrate all VMs due to longer execution time). For an alert of 100 seconds, *Online B-min* provides the best performance as it migrates all VMs from the DC in danger zone, whereas *Online RO-min* migrates only 1945 VMs due to its longer execution time (and hence shorter deadline). This particular case shows that a lower execution time permits migrating all VMs. Conversely, for alert of 200 and 400 seconds, *Online RO-min* provided the best performance migrating all VMs, minimizing downtime and minimizing average network RO.

VII. Conclusion

We formally introduced the problem of online VMs evacuation for alert-based disaster resiliency in an inter data center network. The problem can be modeled as a routing,

Tab. III: Example of the effect of execution time on the performance of the heuristic approaches for a case of 2000 VMs and alert time of 200 seconds.

Approach	Execution Time (sec)	Deadline (sec)	Outcome	
			Nr. VMs Evacuated	Downtime
Offline	6	194	2000	2000
Online B-min	3	197	2000	1840
Online RO-min	26	173	1702	1702

Tab. IV: Performances of the heuristic approaches and the decision of the alert-aware VM evacuation tool for different values of evacuation deadline.

Alert (sec)	Offline					Online B-min					Online RO-min					Most Convenient Approach	Alert-aware VM evacuation Decision
	Estimated Execution Time (sec)	Time Remaining (sec)	VMs Migrated	Downtime (sec)	Average RO (Mbit)	Estimated Execution Time (sec)	Actual Deadline (sec)	VMs Migrated	Downtime (sec)	Average RO (Mbit)	Estimated Execution Time (sec)	Actual Deadline (sec)	VMs Migrated	Downtime (sec)	Average RO (Mbit)		
50	10	40	**1500**	1500	-	18	32	**1200**	1200	-	26	24	**732**	732	-	**Offline**	**Offline**
100	7	93	**2000**	**2000**	-	15	85	**2000**	**1878**	-	25	75	**1945**	**1945**	-	**Online B-min**	**Online B-min**
200	5	195	**2000**	**2000**	-	5	195	**2000**	**622**	-	15	185	**2000**	**451**	-	**Online RO-min**	**Online RO-min**
400	5	295	**2000**	**2000**	-	5	295	**2000**	**0**	**112031**	14	286	**2000**	**0**	**60383**	**Online RO-min**	**Online RO-min**

bandwidth and scheduling assignment problem for the VMs migration from a DC located in a risky zone, i.e., in a zone likely to be affected by a disaster, to other DCs in safe locations. This migration must happen within an evacuation deadline, with the objective of maximizing the number of VMs migrated, minimizing overall downtime of VMs and minimizing overall amount of network resource occupation due to migration of VMs. For this problem, we proposed a multi-objective ILP model and heuristic approaches, namely, *Online RO-min* and *Online B-min* to solve it. We also compare the proposed approaches with a baseline evacuation approach, namely, *Offline*, where all VMs are migrated in an offline manner, i.e., when maximum service downtime is observed. First, we validate *Online RO-min* and *Online B-min* comparing their performance to that of the ILP model and then perform evaluations on larger instances of the problem. Results show that, given the size of the VMs, the dirtying rate, the time available and the network capacity available, there exist thresholds on minimum amount of time required to 1) evacuate all VMs and 2) evacuate all VMs online, thus eliminating service downtime. Results also show that there exists a trade-off between eliminating service downtime and network resource occupation, which highly depends on number of VMs and their dirtying rate. In addition, due to the importance of execution time for the problem at hand, we investigated the execution time of each of the proposed approaches in different network scenarios. Evaluations show that i) the proposed approaches have significantly different execution times of which for Online RO-min in some cases is higher than the evacuation deadline and ii) the execution time can greatly affect the performance of an approach. To tackle this issue, we developed a tool, referred to as Alert-Aware VM evacuation, that selects the most convenient approach, among the heuristic approaches, to consider in order to optimize the objectives of the problem in a given case study.

ACKNOWLEDGMENT

Research leading to these results was supported by the National Science Foundation grant 1818972 and by the Fundação para a Ciência e a Tecnologia (Portugal) grant CENTRO-01-0145-FEDER-029312 (ResNeD project).

REFERENCES

[1] Cho, Kenjiro, et al. "The Japan earthquake: the impact on traffic and routing observed by a local ISP." Proceedings of the Special Workshop on Internet and Disasters. ACM, 2011.

[2] Dominey-Howes, Dale. "Hazards and disasters in the Anthropocene: some critical reflections for the future." Geoscience Letters 5.1, 2018.

[3] T. Gomes et al., "A survey of strategies for communication networks to protect against large-scale natural disasters," International Workshop on Resilient Networks Design and Modeling (RNDM), Halmstad, 2016.

[4] J. Rak et al., "RECODIS: Resilient Communication Services Protecting End-user Applications from Disaster-based Failures," 2016 18th International Conference on Transparent Optical Networks (ICTON), Trento, 2016

[5] A. Pašić et al., "FRADIR-II: An Improved Framework for Disaster Resilience," 11th International Workshop on Resilient Networks Design and Modeling (RNDM), Nicosia, Cyprus, 2019.

[6] Manzalini, Antonio, et al. "Clouds of virtual machines in edge networks." IEEE Communications Magazine 51.7, pp. 63-70, 2013.

[7] Clark, Christopher, et al. "Live migration of virtual machines." Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation. USENIX Association, 2005.

[8] Travostino, Franco, et al. "Seamless live migration of virtual machines over the MAN/WAN." Future Generation Computer Systems 22.8, pp. 901-907, 2006.

[9] Ayoub, Omran, et al. "Efficient Online Virtual Machines Migration for Alert-Based Disaster Resilience." 15th International Conference on the Design of Reliable Communication Networks, 2019.

[10] Nelson, Michael, Beng-Hong Lim, and Greg Hutchins. "Fast Transparent Migration for Virtual Machines." USENIX Annual technical conf., 2005.

[11] Ye, Kejiang, et al. "Live migration of multiple virtual machines with resource reservation in cloud computing environments." IEEE 4th International Conference on Cloud Computing, 2011.

[12] Mandal, Uttam, et al. "Bandwidth provisioning for virtual machine migration in cloud: Strategy and application." IEEE Transactions on Cloud Computing 6.4, pp. 967-976, 2016.

[13] Ayoub, Omran, et al. "Efficient routing and bandwidth assignment for inter-data-center live virtual-machine migrations." IEEE/OSA Journal of Optical Communications and Networking 9.3, B12-B21, 2017.

[14] Ferdousi, Sifat, et al. "Disaster-aware data-center and content placement in cloud networks." IEEE International Conference on Advanced Networks and Telecommunications Systems, 2013.

[15] Ferdousi, Sifat, et al. "Rapid data evacuation for large-scale disasters in optical cloud networks." Journal of Optical Communications and Networking 7.12, B163-B172, 2015.

[16] Xie, Xiaokang, et al. "Evacuate before too late: distributed backup in inter-DC networks with progressive disasters." IEEE Transactions on Parallel and Distributed Systems 29.5: 1058-1074, 2017.

[17] Yao, Jingjing, et al. "Minimizing disaster backup window for geo-distributed multi-datacenter cloud systems." 2014 IEEE International Conference on Communications (ICC). IEEE, 2014.

[18] Goldberg, Andrew V., et al. "A new approach to the maximum-flow problem." Journal of the ACM 35.4, pp. 921-940, 1988.

[19] Cerroni, Walter, et al. "Optimizing live migration of multiple virtual machines." IEEE Trans. on Cloud Computing 6.4, pp. 1096-1109, 2016.

[20] L. Zhang, et al. "Energy-Aware Virtual Machine Management in Inter-Datacenter Networks Over Elastic Optical Infrastructure," in IEEE Transactions on Green Communications and Networking, vol. 2, no. 1, pp. 305-315, 2018.

[21] Yang, Lei, et al. "QoS Guaranteed Resource Allocation for Live Virtual Machine Migration in Edge Clouds." IEEE Access 8, 78441-78451, 2020.

[22] Abali, Bulent, et al. "Live virtual machine migration quality of service." U.S. Patent No. 9,619,258, 2017.

[23] Zhang, Jiao, et al. "Delay guaranteed live migration of virtual machines." IEEE Conference on Computer Communications (INFOCOM), 2014.

[24] Sun, Gang, et al. "A new technique for efficient live migration of multiple virtual machines." Future Generation Computer Systems 55, pp. 74-86, 2016.

[25] Nashaat, Heba, et al. "Smart elastic scheduling algorithm for virtual machine migration in cloud computing." The Journal of Supercomputing 75.7, pp. 3842-3865, 2019.

[26] Ying, Chen, et al. "Raven: Scheduling Virtual Machine Migration During Datacenter Upgrades with Reinforcement Learning." Mobile Networks and Applications, 2020.

[27] Francescon, Antonio, et al. "X–MANO: Cross–domain management and orchestration of network services." IEEE Conference on Network Softwarization (NetSoft), 2017.

[28] Duliński, Zbigniew, et al. "Dynamic Traffic Management for SD-WAN Inter-Cloud Communication." IEEE Journal on Selected Areas in Communications, 2020.