**RDMSim**: An Exemplar for Evaluation and Comparison of Decision-Making Techniques for Self-Adaptation

Huma Samin, Luis H. Garcia Paucar, Nelly Bencomo *
Cesar M. Carranza Hurtado†, Erik M. Fredericks‡

*SEA, Aston University, Birmingham, UK {h.samin, garcial2, n.bencomo}@aston.ac.uk
†Universidad Pontificia Católica del Perú, Lima, Perú cesarmiguelch@hotmail.com
‡Grand Valley State University, Michigan, USA frederer@gvsu.edu

Abstract—Decision-making for self-adaptation approaches need to address different challenges, including the quantification of the uncertainty of events that cannot be foreseen in advance and their effects, and dealing with conflicting objectives that inherently involve multi-objective decision making (e.g., avoiding costs vs. providing reliable service). To enable researchers to evaluate and compare decision-making techniques for self-adaptation, we present the RDMSim exemplar. RDMSim enables researchers to evaluate and compare techniques for decision-making under environmental uncertainty that support self-adaptation. The focus of the exemplar is on the domain problem related to Remote Data Mirroring, which gives opportunity to face the challenges described above. RDMSim provides probe and effector components for easy integration with external adaptation managers, which are associated with decision-making techniques and based on the MAPE-K loop. Specifically, the paper presents (i) RDMSim, a simulator for real-world experimentation, (ii) a set of realistic simulation scenarios that can be used for experimentation and comparison purposes, (iii) data for the sake of comparison.

Index Terms—Remote Data Mirroring, Self-Adaptive System, Exemplar

I. INTRODUCTION

Remote Data Mirroring (RDM) is a disaster recovery technique used to protect data by storing multiple copies (i.e. replicas) on physically remote servers (i.e. mirrors) [1], [2]. The RDM system tolerates failures by requesting or rebuilding the lost or damaged data samples from another active mirror to facilitate data recovery. Hence, the RDM helps in maintaining data availability and preventing data loss. Furthermore, to ensure that distributed data is not lost or corrupted, the RDM is required to perform the replication and distribution of data in an efficient and reliable way.

Considerable research efforts have targeted the domain of Remote Data Mirroring [3]–[8]. However, the RDM applications are very costly to implement as the equipment used to install such applications is expensive. To the best of our knowledge, there is no exemplar available to support research based on the RDM paradigm.

In this paper, we present RDMSim, an exemplar that simulates a Remote Data Mirroring environment. The goal of the RDMSim is to offer researchers a RDM environment to test and compare their decision-making techniques [9] against other techniques. Other exemplars exist however, they focus on other domains and aspects, such as cloud environments [10], cyber-physical systems [11], traffic management system [12], client-server systems [13] and IoT-based systems [14]. In comparison to [10], that deals mainly with the functionality of cloud environments such as workload management using the addition and removal of virtual machines, RDMSim focuses mainly on the simulation of Remote Mirroring process.

The RDMSim exemplar presented here is implemented in Java, keeping in view the operational model presented in [1], [2]. It simulates the RDM presenting a fully connected network of mirrors. The simulator offers the flexibility of changing the number of mirrors to create a customized RDM network according to the experiment’s requirements. The focus is on the application of self-adaptive realization strategies in the form of the topologies of Minimum Spanning Tree (MST) and Redundant Topology (RT). The application of these topologies have an impact on the different network parameters such as bandwidth consumption and active network links affecting the quality objectives such as the minimization of operational costs and the maximization of the reliability of the network. A trade-off of such impacts has to be taken into account as part of the decision making [15]–[19]. The topological impacts have been defined based on the expert knowledge presented in [6]. Additionally, we provide an implementation of different scenarios that define possible different uncertain environmental contexts for the RDM [20]. A Python version is also publicly available. Researchers can use these scenarios to test their specific decision-making techniques based on Reinforcement Learning [21], Multi-Criteria Decision Analysis [22] and Evolutionary Computation [5] among others. Researchers can also design their own scenarios by modifying the different parameter ranges.

The paper is organized as follows: Section 2 presents the operational model of a RDM. In Section 3, we present the architecture of the RDMSim exemplar. Section 4 provides a description of the different scenarios for the experiments that can be executed by the RDMSim. In Section 5, an example of how to execute experiments using RDMSim is provided, which
II. REMOTE DATA MIRRORING

The RDM application is composed of data servers and network links [1], [2]. It must replicate and distribute data in an efficient manner by minimizing consumed bandwidth and providing assurance that distributed data is neither lost nor corrupted [1]. The RDM application must achieve functional objectives such as construct a connected network and distribute data. These functional objectives can be achieved through alternative realization strategies represented by two different topologies: Minimum Spanning Tree (MST) and Redundant Topology (RT). An MST Topology uses the least possible number of network links to transmit data among different remote servers. Contrarily, an RT topology uses simultaneously, several redundant network links paths to transmit information among remote servers.

The implementation of the RDM considered in this paper should also satisfy the following three quality objectives: Maximization of Reliability (MR), Maximization of Performance (MP) and Minimization of Cost (MC). The levels of satisfaction associated with reliability, performance, and cost of the RDM are determined according to the trade-offs based on:

- A RT Topology offers higher levels of reliability than an MST topology. However, the cost of maintaining an RT topology may be prohibitive in some contexts, given the additional cost of bandwidth consumption required.
- Conversely, a MST Topology offers higher levels of performance with lower levels of cost than an RT topology. However, the reliability of the system can be impacted in a negative way when an MST Topology is used.

Based on the above, we have designed the RDMSim exemplar. Next, we present the architecture for RDMSim.

III. ARCHITECTURE

The RDMSim exemplar has been developed to facilitate the implementation of a two-layered architecture for a self-adaptive RDM, as shown in Fig. 1. The architecture structures a Managing System (based on feedback loop [23], [24]) on top of the Managed System (the RDMSim). We next describe each layer.

A. Managing System

The Managing System, at the upper layer, is responsible for providing the self-adaptive decision-making logic. A feedback loop is implemented to monitor the environment and managed system, adapting the latter when necessary. The feedback loop consists of Monitor-Analyse-Plan-Execute over a Knowledge base K (MAPE-K) [23]. The MAPE-K loop is considered an architectural blueprint for self-adaptive systems and is used to perform adaptation decisions on the Managed System (i.e. RDMSim in our case). When using the RDMSim exemplar, researchers will provide their own decision-making techniques to serve as a Managing System. The Managing System can be based on different techniques such as Multi-Criteria Decision-Making [22], Reinforcement Learning [7], and Evolutionary Computation [3], [8], etc.

B. Managed System

RDMSim represents the Managed System and provides probes and effectors that can be used by the Managing System to interact with the simulator. Probes are used to monitor information (M in MAPE) whereas the effectors are used to execute the adaptation decisions (E in MAPE) on the Managed System.

Next, we present the architecture of the Managed System implemented as Java Packages for the RDMSim software. The components in the architecture for RDMSim, presented in Fig. 1 are as follows:

1) Management Component: which acts as a bridge between the Managing System and other internal components of the RDMSim. It provides an implementation of probes and effectors to be used by the Managing System. The functions provided by the probes and effectors are used to both monitor the status of the RDM (i.e. cost, reliability and performance) and also change the network topology and different network parameters according to the decision made as described in Table I and II respectively.

2) Network Component: which provides an implementation of the main physical elements of the RDM. These elements include the number of mirrors (i.e. servers) and the network links that represent a fully connected network of mirrors. As an example, for 25 mirrors, a network of
300 links will be created. The users of RDMSim can change the number of mirrors to create a custom RDM network for their experiments. The Network Component also provides an implementation of the monitorables and topologies for the network. Specifically, in the RDMSim, we provide an implementation of three monitorables:

**Mon1– Active Network Links**: provides the current active network links to measure the reliability of the RDM. The RDM will provide a higher level of reliability with a larger number of active links.

**Mon2– Bandwidth Consumption**: provides the current bandwidth consumption to measure the operational cost for the RDM in terms of inter-site network traffic. Operational costs will be increased for the RDM with a higher amount of bandwidth consumed. Bandwidth Consumption is measured in GigaBytes per second.

**Mon3– Time to Write Data to mirrors**: measures the performance of the network in terms of writing time to maintain multiple copies of data on each remote site. A big writing time leads to reduction of performance of the RDM. Time to Write Data is measured in milliseconds.

For the communication between the mirrors, we consider synchronous mirroring [2], [5]. During synchronous mirroring, sequential writing is performed to prevent data loss [5]. In sequential writing, the primary mirror (i.e. the sender) waits for an acknowledgement (known as a handshake) regarding the receipt and writing of data from the secondary mirror (i.e. the receiver). This process is performed for each active link on the communication path between the mirrors. Therefore, the time to write data is computed as Total Writing Time = (α * number of active links) * Time to Write Data Unit. Here, α represents a fraction of active links to constitute the communication path between mirrors. α can have a value of greater than zero and less than and equal to one. For our experiments, we have set α = 1.

Similarly, the bandwidth consumption is also dependent on the number of active links. More active links imply more data transmission, which leads to a higher bandwidth consumption [5]. Hence, we compute the Bandwidth Consumption as Total Bandwidth Consumed = (α * number of active links) * Bandwidth per link.

3) **Simulation Component**: which includes the implementation of the uncertainty scenarios [20], [25] that represent the different dynamic environmental conditions that the RDM can face, and which will be simulated. It allows the setting of the simulation properties, such as the number of simulation runs and the chosen uncertainty scenario(s) to be executed by the RDMSim.

A partial class diagram representing the elements of the Management Component, Network Component and Simulation Component is shown in Fig. 2. The NetworkManagement class along with the Probe and Effector interfaces provides an implementation of the Management Component. The classes NetworkProperties, Monitorables, Topology and TopologyList are part of the Network Component and provide an implementation of the corresponding features of the RDM. The SimulationProperties and UncertaintyScenario classes are part of the Simulation Component, and are used to implement the functionalities related to the simulations to be executed.

### IV. Scenarios

Six different scenarios were defined to be used in simulations of the RDM. These scenarios have been designed to simulate different archetypal real situations, which can cause deterioration of satisfaction of the quality objectives of the system in relation to a scenario with stable conditions.

The main goal of the scenarios depicted below, is to evaluate how decision-making techniques and algorithms react under uncertain situations, specially different from the stable conditions. Next, a description for each scenario is presented.

**Default scenario S₀**: For the sake of comparison between techniques, a default scenario is provided that represents an environment envisioned by the requirements experts [4], [6]. For the RDMSim, the following thresholds for the levels of satisfaction associated with reliability, performance and cost are suggested: bandwidth consumption should be on average less than or equal to 40%. Similarly, the time to write data should be on average less than or equal to 45%. On the other hand, the number of active links should be on average greater than or equal to 35% of the total number of links. The initial

---

1. To implement realistic impacts, we vary the time between 10 to 20 milliseconds
2. To implement realistic impacts we vary the Bandwidth per link between 20 to 30 GBps
topology being used is MST topology.

Scenario S1 - Unexpected Packet Loss during MST: The initial topology being used is MST Topology. A period of consecutive and unexpected data packet loss during the execution of the MST Topology generates a reduction on the reliability of the system. Data packet loss represents link failures in the RDM system, which may be caused, for example, by problems with the equipment (e.g. failures in a switch or router or power failures [1]).

Scenario S2 - Unexpected Packet Loss during RT: The initial topology being used is RT Topology. Unexpected data packet loss during the execution of the RT Topology, are generating an unusual rate of data forwarding, which would increase the bandwidth consumption (i.e. cost), and would reduce the system’s performance. As said before, in the RDM, the cost for inter-site links communication is a function of the data sent over them. Therefore, a Redundant Topology (RT), which involves a bigger number of inter-site network links than a Minimum Spanning Tree Topology (MST), is more expensive. Cost increases as the number of active links increases and a reduction on the system’s performance could also be expected.

Scenario S3. Simultaneous occurrence of the scenario S1 and S2. The current topology is randomly generated.

Scenario S4 - MST topology execution failures: The topology being used is MST Topology. Involves the behaviour presented in the scenario S1. Additionally, during the execution of the MST topology, an increment in bandwidth consumption (MC) and the reduction of the system’s performance (MP) is also produced due to synchronous mirroring.

Scenario S5 - RT Topology execution failures. The topology being used is RT Topology. Involves the behaviour presented in the scenario S2. Additionally, the RT topology is also producing a reduction on the reliability of the system (MR) due to failures in the equipment such as routers and switches.

Scenario S6 - Significant site failure. The current topology is randomly generated. This scenario involves the simultaneous occurrence of the scenarios S4 and S5. It is related to a significant site failure [1], [2], where both, repeated and multiple concurrent failures are expected [1] as in the scenarios S4 and S5 but all at the same time. A full-scale site failure may be caused by a power outage affecting all the buildings on different campuses, an earthquake or flood affecting buildings within several metropolitan areas. Under this scenario, the worst-case data loss [2] may occur in different sites (RDM nodes), i.e. a site can be destroyed or inoperative before the full backup of information is shipped to an offsite. Site failure disasters are usually modelled with a failure rate of once per year [2].

V. EXPERIMENTS

In this section, we provide a simple example to describe the steps to develop a custom adaptation logic for performing experiments using the RDMSim. We also demonstrate the execution of different uncertainty scenarios using the custom adaptation logic.

The steps for development of custom adaptation logic are as follows:
**Step: 1 Download the RDMSim Exemplar**

Download the RDMSim package from the RDMSimExemplar repository\(^4\) and install the required libraries.

**Step: 2 Design an Adaptation Solution**

Design an adaptation solution (Managing System) using the Probe and Effector interface functions provided by the RDMSim software as follows:

**A. Loading Configuration Settings and Instantiation of Probe and Effector**

The first step in implementing the custom adaptation logic is to load the configuration settings for the experiment from the configuration file and instantiation of the Probe and Effector components. The Probe and Effector will enable the communication between our custom adaptation logic and RDMSim. This can be done by using the NetworkManagement class in your program as follows:

```java
NetworkManagement network_management;
network_management=new NetworkManagement();
Probe probe;
Effector effector;
probe=network_management.getProbe();
effector=network_management.getEffector();
```

The NetworkManagement class is responsible for loading the configuration parameters and instantiating the Probe and Effector instances. The configuration settings include the parameters like number of simulation time steps, the number of mirrors for the RDM, number of active links and the uncertainty scenario to be considered for the experiments. The details of the configuration parameters is provided in the RDMSim Artefact: User Guide document provided as part of RDMSimExemplar repository.

**B. Monitoring of the RDMSim network using Probe functions**

In order to monitor the RDMSim, we can use the probe functions provided in Table II. For example, to get the values of all the monitorable metrics, at a particular simulation time step, we can use the getMonitorables() function as follows:

```java
Monitorables m=probe.getMonitorables();
```

**C. Performing Adaptations on the RDMSim using Effector functions**

In order to perform adaptations on the RDMSim, we can use the Effector functions provided in Table II. For example, to change the network topology at a particular timestep, we can use the setNetworkTopology() function as follows:

```java
effector.setNetworkTopology(10,"mst");
```

The code above will set the Minimum Spanning Tree (MST) topology for the network at the simulation timestep 10.

A step by step implementation of the MAPE-K loop using steps A to C is provided in the User Guide document.

**Step: 3 Design and Execute Experiments to test the Adaptation Logic**

Once the adaptation solution is designed, an experiment should be designed to test the adaptation logic. For an experiment to be executed, the configuration parameters (provided in the configuration file) should be set to execute a particular simulation scenario. We have assigned some default values for the configuration parameters based on the expert knowledge provided in [6]. You can change the number of simulation runs, the number of mirrors for the network, the uncertainty scenario and the ranges for the different monitorables. The details for the configuration parameters are provided in the User Guide.

**Example: To demonstrate RDMSim working under Default and Detrimental Scenario S\(_1\)**

We demonstrate the execution of experiments under both the default scenario S\(_0\) and uncertainty Scenario S\(_1\). For our experiments, we consider an RDM network of 25 mirrors and 300 network links to create a fully connected network. We have set the default values for the configuration parameters in the configuration file. The satisfaction thresholds for the quality objectives have been set based on the expert knowledge provided in [6]. In order to satisfy the quality objectives of minimization of operational cost and maximization of performance, bandwidth consumption and time to write data should be minimized. Conversely, the quality objective of Maximization of Reliability requires maximization of number of active links. Based on the expert knowledge, the bandwidth consumption should be less than or equal to 40 percent to satisfy minimization of operational cost. Similarly, the time to write data should be less than or equal to 45 percent to satisfy maximization of performance. On the other hand, number of active links should be greater than or equal to 35 percent of total links to satisfy maximization of reliability for the RDM. Once the configuration parameters are setup, we have executed the experiments for 100 simulation runs for the scenarios as shown in Fig. 3 and 4. Under default scenario, the RDMSim will meet the satisfaction thresholds in terms of the value ranges of bandwidth consumption, active links and time to write data. Under uncertainty scenario S\(_1\), the different disturbance levels are introduced to reduce the number of active links affecting the reliability of the system when MST is the selected topology as shown in Fig. 4.

For further validation purposes, we have applied reinforcement learning based decision-making to the RDMSim. We provide our initial evaluation results for the RDMSim using MR-POMDP++ [7], [21] as part of the RDMSimExemplar repository. MR-POMDP++ is based on Multi-Reward Partially Observable Markov Decision Process (MR-POMDP). MR-POMDP is a multi-objective reinforcement learning technique that considers the decision-making agent performing in a partially observable environment. MR-POMDP++ performs adaptations on the basis of the multi-objective utility value computed at each simulation time step.

\(^{4}\)https://doi.org/10.5281/zenodo.4613152
default configuration setup provided in the configuration file. In order to test our decision-making techniques DeSIRE [26] and MR-POMDP++ [27], we have also used the exemplar [14]. Both exemplars [14] and RDMSim, focus on different domains and aspects, the IoT domain and the RDM and effect on quality objectives respectively, and complement each other.

Discussion: An RDM can be seen as a specific example of a more generic type of applications, where the decision making guides self-reconfiguration by identifying a target system configuration to provide the desired system behavior [19], [28]. A set of reconfiguration instructions to reach the desired target configuration is applied (i.e. E in MAPE). These reconfiguration instructions define an adaptation path. Several adaptation paths may be chosen, and most self-reconfiguration approaches select adaptation paths based on trade-offs between several objectives goals, such as performance and reliability [19]. As such the RDMSim can be used to test decision-making techniques applicable to other domains as well.

VI. CONCLUSION

In this paper, we have presented the RDMSim exemplar to provide a simulating environment for the RDM. RDMSim facilitates the researchers to execute experiments in the domain of RDM. To the best of our knowledge, RDMSim is the first simulator to be implemented for this domain. Using RDMSim, researchers can compare their self-adaptive decision-making solutions with other techniques, including ours [21]. We have executed experiments for each scenario presented here, using our own decision-making technique, called MR-POMDP++ [21]. The results are provided in the RDMSimExemplar repository, ready to be used for comparison purposes. Furthermore, RDMSim also provides opportunities for researchers to design their own scenarios for experiments by modification of values in the configuration file using the instructions provided in RDMSim user guide. We hope that the research community will use the RDMSim to evaluate and compare novel solutions in the area of self-adaptive decision-making.

ACKNOWLEDGMENT

This work has been partially supported by The Leverhulme Trust Fellowship “QuantUn: quantification of uncertainty using Bayesian Surprises” (Grant No. RF-2019-548/9) and the EPSRC Research Project Twenty20Insight (Grant No. EP/T017627/1).
