DDDS: A Distributed Disaster Detection System

Omar Hussain Alhazmi

Department of Computer Science, Taibah University, Medina, Saudi Arabia

Summary
Disaster Recovery Plans is a crucial part of the life of an IT center in an organization; it contains policies and procedures to be applied before, during, and after a disaster of an IT system. However, an important part of the disaster recovery process comes when the disaster occurs up until the disaster recovery plan is activated. This is precious time to detect and declare a disaster especially in critical systems. This is important to control the MTD (Mean Tolerable Downtime). Here, we present a distributed disaster detection system that is based on agents. Agents are to be located on different servers in a data center and the will communicate with a central management unit. The DDDS is aimed to complement the work of an existing disaster recovery system, or to run stand-alone if no disaster recovery system exists and act as a warning tool aiding the system administrator.

Key words:
RTO, RPO, MTD, disaster recovery, business continuity.

1. Introduction

Disaster Recovery can be defined as "(DR) Planning and implementation of procedures and facilities for use when essential systems are not available for a period long enough to have a significant impact on the business" [1].
The common perception about disasters is that it can be caused by nature (volcanoes, floods, earthquakes, tsunamis…etc.) or by human action (wars, malicious activities…. etc.). However, the four top causes of disaster in information technology are shown in Table 1 below, [2]:

Table.1 the four causes of IT disasters [2]

Rank	Disaster Cause	Percentage
1	Hardware/Infrastructure failure	55%
2	Human error	22%
3	Software failure	18%
4	Natural disaster	5%

Of course big disasters are rare; hence, smaller disasters occur more frequently; thus, they have significant impact on the systems and would cause serious downtime and outages. Therefore, one can argue that a software detection system that can prove helpful in the 95% (by summing the top 3 causes shown it Table 1) of the total disasters is wise investment and can provide a valuable addition to any data center alongside the traditional disaster recovery systems with all the backups, replications and synchronization. This is especially true given that some disaster recovery systems have no detection and warning mechanism and are applied manually. We don't suggest that Distributed Disaster Detection System (DDDS) is to replace existing DR solution but to complement it.
DDDS system can also run without ad DR solution and thus it is a low cost system to monitor and warn against possible failure of hardware, software or human errors.
In the next section we shall overview some of the related work. Then, in section 3 we will discuss disaster detection. Next, in section 4 we will present Distributed Disaster Detection System (DDDS). Finally, in section 5, we will give some concluding remarks and some future research directions.

2. Related Work

One of the current trends in disaster detection is to have a natural detection system [3][4]; basically, these systems consist of a network of GPS located sensors to monitor weather and other environmental parameters; these systems have access to some satellite/radar capabilities aiming to discover an early signs of natural disasters; an example the system built by NASA [5]. Moreover, these systems are built and managed by governments and civil defense entities.
In the information technology area, there is a considerable effort done on network and mobile network recovery [5]. Ceballous et al. have studied the business continuity, security and interconnection in large enterprises they proposed a complementing technology to overcome some of the challenges facing data centers [6]. The difference is that these networks spans thousands of kilometers. While in our work we discuss having a system on a local data center or multiple datacenters of the same organization.
Substantial advances have been achieved on intrusion detection systems that existed for years [7][8][9]. However, not much work has been done on information disaster recovery detection. Alghamdi and Alaama have proposed an agent based protocol to overcome latency issues existing in current protocols [10].
The scope of this paper is to detected disasters within a data center; this has similarity in concept with here, we experiment with this system to complement intrusion detection systems in order to for cover an important gap of

Manuscript received November 5, 2015
Manuscript revised November 20, 2015

detecting signs of failure within data centers that might be linked to security issues too.

3. Disaster Detection Overview

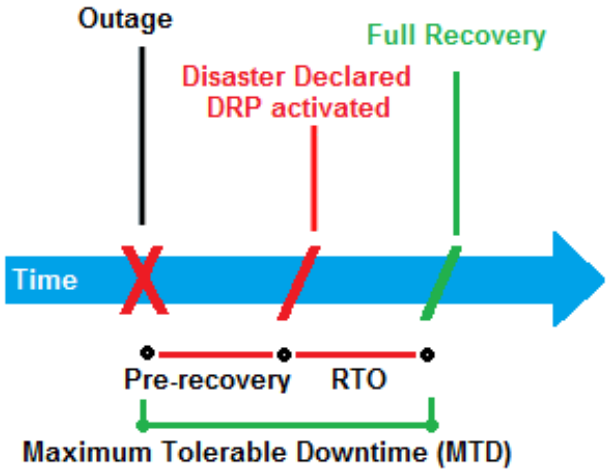

Fig.1 Disaster Timeline, showing MTD, RTO and Pre-recovery times

One critical point as seen in Figure 1 below is the outage of some of the services which will result in a time where some data and services are disrupted; at this point the disaster is declared. The disaster can be declared by:

1. Manually: the system administrator declares it and the recovery process is initiated
2. Automatically: system detects some abnormality and the absence of some services and declares a disaster and automatically starts the recovery process.
3. System-assisted: declaration, here the system has no privilege of declaring the disaster; however, the detection system will alert the system administrator to take action.

Table 2: Disaster and declaration approaches

System	Advantages	Disadvantages
Manually	-Minimizes false negative -cheaper	Need human attention 24/7; therefore, can cause major delay
Automated	Fast response	High rate of false-positive
System-assisted	Combine both advantages	Cost of resource allocation and needs constant monitoring

Both the advantages and disadvantages of each of the systems are previewed in Table 2 above. Besides, as with any detection system; a level of certainty is always factor in wither to declare a disaster or not; the typical four cases are shown in Table 3 below:

Table 3: Cases to decide to declare a disaster or not

Case	Description	Result
False-negative	The system continue working	No alert
True-negative	The system declare a disaster by mistake	Alert
False-positive	The system fails to detect or uncertain to declare a disaster	No alert
True-positive	The system successfully detects a disaster and declare it	Alert

One important factor in detecting a disaster is the detection rate.

4. The Disaster Detection System

As we have seen in the previous section, the automated detection and warning system of disasters plays an important role in the disaster recovery. Indeed, it can improve maximum tolerable downtime (MTD) by minimizing the pre-recovery time by giving earlier signs of disaster. Thus, DDDS will trigger the DR system if the detection returned a positive result.

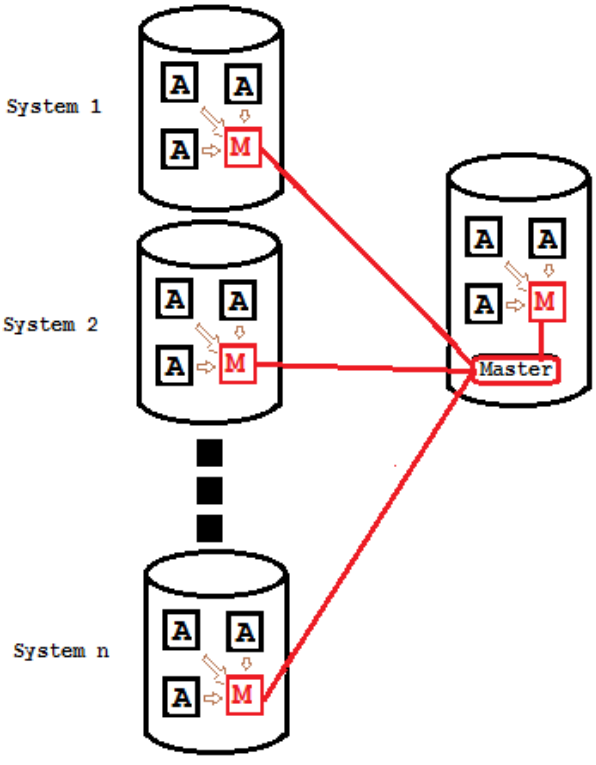

Fig.2-Diaster detection system

The disaster detection system consists of:

- Agents:
- An agent is basically a thread running on a system (an actual server) this thread monitors some files carefully placed in different location on the system. This thread will monitor those files for any change of contents, then, whenever a change occur will send a message to the local management system and alert of an "integrity" issue. Moreover, if the thread could not reach a certain file then it will send a different message as an alert of "availability" issue.
- These threads will run in the background and should consume negligible amount of resources. The administrator should configure and set the threshold values
- The agent will send a 3- tuple message containing:
- I – Agent ID
- L – Location of the issue
- S – issue "Availability" or "Integrity" or both

Fig 3 below, shows the basic algorithm of the agent and how messages are exchanged within the system with the management center

```
Set
S={Availabilty_issue,Integrity_Issue,No_Issues}
//three statuses of the location

Agent(I,L,S)
{
Int i;
For (i=0;i<num_locations;i++)
{
If!Read(location[i])
return(id,loc_id,Availability_Issue);
// if read operation has failed
If (location[i]!= data[i])
// if the data has been modified
return (id,loc_id,Integrity_Issue);
Data[i]=new_data;
Write(location[i];new_data);
// update the data
}
Return(id,void,NO_ISSUES)
// the agent reporting that the system is fine
}
```

Fig 3. DDDS Agent Module

- Management Centers:
  The management center is a thread which will be located at each system (server), they communicate with the virtual Agents and each management center will receive 2-tuple messages from the Agents and will compile a report message to be forwarded as 2-tuple message to the master management center as follows:
  - S – System I.D.
  - I – Type of issue detected

Fig 4 below, shows the basic algorithm for management and how the messages are exchanged with agents and with the master management center

```
Management(I,S)
{Int i;
    For (i=0;i<num_agents;i++)
    {
     Agent(I,L,S);// calling the agent
    If (S!=No_Issues)
    return (I,S) // if there is a problem
send  //the  details  to  the  central
management
// else call the next agent
    }
Return (void,NO_ISSUES;
}
```

Fig 4. DDDS Management Module

- Master Management Center:
  This is the head of the system, the system receives messages from all management center and then determine wither to alert admin or administration system about a possible disaster based on a preset threshold. Hence, the administrator can have a sensitive system or a less sensitive system based on criticality of server. For example, the master management system can give more weight to critical systems and less weight to the non-essential systems.

```
Master_Management_Center(void)
{
Int I;
   For (i=0;i<num_managments;i++)
   {
    Management(I,S);
// call a local management
Print_line("Server:",I,"has",S)
//report the status to the admin
   }
}
```

Fig 5. DDDS Central Management Module

Fig 5above shows the basic algorithm for the master management center and how the messages are exchanged with management centers; and how the DDDS system pass messages to the system administrator.

By using DDDS, a system administrator can have a dashboard to check on system issues and trigger a disaster

recovery. On the other hand the master management center module given in (Fig. 4) can be linked directly to existing DR system to have fully automated disaster detection and recovery system.

5. Conclusions

In this work we suggested a simple software system to give an early alert of disasters caused mostly by hardware failures, human errors and malicious attacks. However, the system is not meant to detect natural disasters.

The system is basically based on passing messages about the status of each system and will help give an assessment about the system's health.

The system is completely a software system; the cost to deploy is negligible. However, the running cost is linked to the required configurations; in other words, if the system sends more status messages it will consume some bandwidth and processing resources, if the configuration is with less messages, these side-effects will also be negligible.

Future work is needed to estimate the amount of resources needed to run the system and will it be feasible to allocate those resources for this purpose. Furthermore, if there is unacceptable resource consumption then may be by optimization we can reduce this overhead costs.

The proposed system could also consider communication health with other remote systems; therefore, the system can be deployed and extended as a monitor to critical systems on multiple sites.

Given the advantage of not having to install special hardware for disaster recovery; on the other hand, it is obvious that the main limitation of the system is that if all servers got down. Thus, the system will also fail; the system is effective against security threats, limited hardware or software failures.

One last limitation that the DDDS currently, is not compatible with virtualization and cloud computing, DDDS could be developed to address these issues.

References

[1] Jennifer Curry, "Top four disaster causes", http://www.latisys.com/blog-post/top-4-causes-of-it-disasters, April, 4th 2014.

[2] http://dictionary.reference.com/browse/disaster-recovery, October 2015.

[3] Colburn, Robert "Sound the Alarm: A History of Disaster Detection and Warning Technologies", The IEEE institute newsletter, September 9 2013.

[4] Flowers, April, " NASA Builds GPS-Based System For Detecting Natural Disasters http://www.redorbit.com/news/science/1113025581/nasa-gps-system-detects-natural-disasters-121113/

[5] Y. Ren, M. Chuah, J. Yang, Y. Chen, "MUTON: detecting malicious nodes in disruption-tolerant networks", IEEE WCNC 2010, April, 2010

[6] Ceballos, Juan DiPasquale, Richard ; Feldman, Robert, "Business continuity and security in datacenter interconnection", Bell Labs Technical Journal., Volume: 17 Issue: 3, 2012.

[7] Denning, Dorothy E., "An Intrusion Detection Model," Proceedings of the Seventh IEEE Symposium on Security and Privacy, May 1986, pages 119–131Fff

[8] Lunt, Teresa F., "Detecting Intruders in Computer Systems," 1993 Conference on Auditing and Computer Technology, SRI International.

[9] Snapp, Steven R, Brentano, James, Dias, Gihan V., Goan, Terrance L., Heberlein, L. Todd, Ho, Che-Lin, Levitt, Karl N., Mukherjee, Biswanath, Smaha, Stephen E., Grance, Tim, Teal, Daniel M. and Mansur, Doug, "DIDS (Distributed Intrusion Detection System) -- Motivation, Architecture, and An Early Prototype," The 14th National Computer Security Conference, October, 1991, pages 167–176

[10] Alghamdi, Hanaan and Alaama, Arwa, DRP-DRP: Data Replication Protocol for Disaster Recovery Planning, International Conference on Innovations in Information Technology, 2008. Pp 228-232.

Omar H Alhazmi received the B.S. degree in Computer Science from King Saud University in 1997, an M.S. degree from Villanova University in 2001, and a Ph.D. degree from Colorado State University in 2007. During 2007-2011, he worked in the National Information Center at Ministry of Interior; later, in 2011 he joined the faculty of computer science at Taibah University in Medina, Saudi Arabia.