Survivable System by Critical Service Recovery Model: Single Service Analysis

Irving Paputungan and Azween Abdullah
Computer and Information Science Department, University Technology Patroness, Malaysia

Abstract: *This paper reported another recovery model to enhance system survivability. The model focuses on how to preserve the system and resume its critical service while incident occurs by reconfiguring the damaged critical service resources based on available resources without affecting the stability and functioning of the system. There are three critical requisite conditions in this recovery model: the number of pre-empted non-critical service resources, the response time of resource allocation, and the cost of reconfiguration, which are used in some scenarios to find and re-allocate the available resource for the reconfiguration. To validate the viability of the approach, one instance case is provided. The adoption of fault-tolerance and survivability using redundancy re-allocation in this recovery model is discussed from a new perspective.*

Keywords: *Critical service, recovery, resources reconfiguration, survivability.*

Received August 16, 2007; accepted May 12, 2008

1. Introduction

System survivability is the ability of a system to maintain its essential service in a timely manner when it is suffering from attacks, fault, or accidents [14, 2]. The basic idea of survivability is that the systems can achieve their critical services and recover the damaged services as soon as possible when intrusions succeed [5], even after the main components or the resources are damaged. The condition of any system is that there is no absolute security to avoid from failure because of attacks, faults, or accidents [9]. To ensure the system delivers services stably and reliably when fault occurs, we must consider a technique for enhancing system survivability [6].

Recently, the correlated techniques that ensure system survivability are mainly from the views of fault resistance and recognition, which is not satisfying enough to the basic properties of survivability. [9] Developed a model that simulates complete episodes of attacks on network computer systems and the responses of these systems. This approach has involved developing a flexible template that can be used to analyze survivability of network systems. [15] Proposed a novel quantitative analysis method based on grey analysis for network survivability. Both of these techniques assert that the returning of the system to the normal state should be considered in the future.

Another property of survivability is recovery, an ability to maintain or restore the essential or critical service from damage as early as possible to fulfil its mission as conditions permit. Recovery depends on the severity of the damage (i.e., how many resources have been affected), recovery strategies and remaining undamaged resources that are in place. As long as system can reconfigure the destroyed resources under faults, and ultimately keep the critical services running all along, the system will survive.

In the recovery model, we take the concepts of time factor [8, 13], cost factor [9, 10, 13], resource re-allocation/reconfiguration concept [1, 13], and resource redundancy concept [1, 7, 12], as different approach in fault tolerance perspective to return the system back to normal condition. The stability of the entire system is based on active and accurate functioning of each and every service nodes. The system becomes unstable when at least one active service nodes becomes dysfunctional and it resources are pre-empted or denied access.

Firstly, the state transition diagram of the system has been built, which is the simplification of Popstojanova's work [4] to describe the behaviour of a system, then mapping the recovery actions to this transition model. Since our objective is to recover the system when incident occurs, we set the system degrade gracefully when recovery process running [11]. It is a good thought, because most of the recovery techniques placed after incident occurred, the system fails to running. In this paper services and processes are used interchangeably.

2. Problem Definition

The problem can be described as: there is a Real Time System (RTS) which has critical services and non-critical services. Some critical service resources are destroyed by a fault. To maintain the stability and mission of the system, the system has adaptive abilities to recover by re-allocating available resources dynamically to critical service. The recovery process works under RTS circumstance, hence the duration of the process is a great concern. The duration included the response time and the usage time. This work focuses on the response time of the available resources. That is the first requisite condition. The problem is to find out how to reconfigure the critical service resources which can ensure sustainable operation of critical services. It brings along to the next requisite condition, cost. The cost of reconfiguration is calculated by the number of resources and cost of resource. It is assumed that error detection (monitoring) and damage assessment are taken care by some other mechanism such as control system architecture [7]. It is assumed that there will be no further error when a critical service being faulted. Figure 1 provides a typical example of interconnected system structure that contains a master resource controller, some critical services and some non-critical services, where the recovery could be applied. Resource reconfiguration computation will be done by the recovery engine. This paper is limited to single fault on single critical service analysis. It is also assumed that there will be no further error when a critical service being faulted.

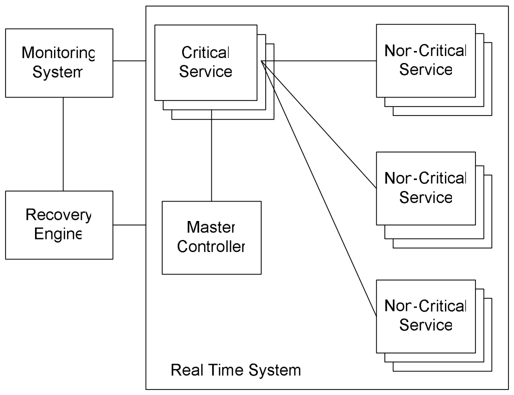

Figure 1. Real time system structure.

The critical service differs in three levels: high, medium and low level depending on its function at the system. The replacement resources to reconfigure the damage can be taken from:

- Redundant resources of critical service.
- Unused/idle redundant resources of the system.
- Released/redundant resources of non-critical service.
- Pre-empted resources of non-critical service that are currently being used.

To avoid instability of the system and more cost while reconfiguring, the response time of available resources should be as quick as possible and the cost of available resources should be as cheap as possible. Furthermore, the number of pre-empted non-critical service resources, if need to be utilized, should be as few as possible, this is considered the third requisite condition.

3. The System Model

Figure 2 depicts the state transition diagram which is used as a framework for describing the behaviour of the system. The system contains 4 states: good state, vulnerable state, fault state, and recovery state. The system moves to vulnerable state if a user violates security policy to access a resource without authorization. Vulnerability is the property of the system, its attendant software and/or hardware, or its administrative procedures, which causes it to enter vulnerable state [4]. The system enters fault state when vulnerability is successfully exploited and the fault unmasked by simple fault tolerance. In the next state, recovery state, the system will be recovered. To limit the damage and protect the system from denial of service while maintaining the critical services, it sets into graceful degradation mode. Critical services are defined as the functions of the system that must be maintained to meet the system requirements even when the failures occurred [2]. In order to survive the critical service, it is critically assumed the recovery process will always be successful; hence there is no fail state.

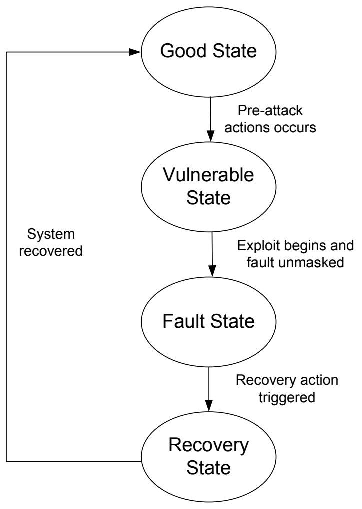
Good State
Pre-attack actions occurs
Vulnerable State
System recovered
Exploit begins and fault unmasked
Fault State
Recovery action triggered
Recovery State

Figure 2. State transition diagram.

4. The Recovery Model

It will often be the case that the effects of a fault will leave the system with greatly reduced resources (processing services, communications capacity, *etc*.,) and substantial changes in the services provided to the users will be necessary. The ability to tolerate certain types of fault is the only practical approach to achieve survivability. In this work, there are several fault types: single fault, multiple sequential faults, and multiple concurrent faults for multiple critical services. In this paper, we only focus on single fault as our basic and the reconfiguration refers to resource redistribution and not structural or topological reconfiguration.

4.1. Reconfiguration Process

Figure 3 described the recovery action that is mapped into the recovery state. The process starts with diagnosing the destroyed resources of the critical services. The diagnosis part will determine where the damage occurred. The analysis part calculates the amount of damaged resources. In order to move back to good state, the available resources to reconfigure critical services resources must be found and re-allocated.

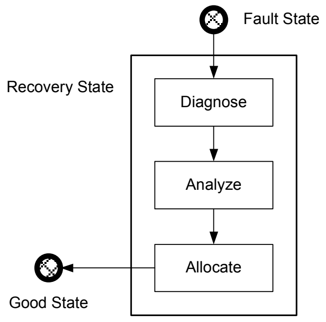

Figure 3. Modules in recovery action.

The process to find the available resource for reconfiguration is based on the tabular method. The four types of resources described in previous sections are defined in three tables, master resources allocation table, critical services table, and non-critical services table. These three tables will be created automatically (dynamic table creation) when the system begins operation, and will be destroyed when the system cease to exist by the OS or by another process within the current system. For the purpose of our discussion we assume the table creation process is spawned by the recovery engine.

The master resource allocation Table 1 shows the total resources currently used and allocated for redundancy inside the running system. Required resources means the resources that the system needs to run all the services i.e., the total of all resources used by the services, critical and non-critical. Used-redundant means the redundant resources that are currently used by the system's service and unused-redundant means the idle redundant resources that are not currently used by the system.

Table 1. Master resource allocation.

Resource	S_1	S_2	S_a
Required (MQR)			
Used Redundant (MUR)			
Unused Redundant (MUNR)			

The critical services resources Table 2 shows the resources used by the service, redundant resources that are available for that service, and the resources that are destroyed by fault.

Table 2. Critical service resource.

Resource	R_1	R_2	R_b
Resource Currently Used (CSCU)			
Redundant (CSR)			
Damage (CSD)			

The non-critical services resources Table 3 shows the resources used by the service and the resources that are released while the services are in progress. Resources are taken-up and released when it is not required.

Table 3. Non critical service resource.

Resource	r_1	r_2	r_c
Resource Currently Used (NCSCU)			
Released (NCSR)			

In the allocation process, there are four scenarios to be analyzed after an error/damage occurs to the critical service resources that find the available resources for reconfiguration. They are:

- Redundant resources available with critical service. The process will check the redundant resources of the critical service. If it is available, then the problem can be fixed without affecting any other services i.e., the required resources can be used for damage recovery.
- Redundant resources available with the system. If there are insufficient redundant resources with the critical service, it will check the unused redundant resources of the system. If it is available, then allocate the available resources.
- Released resources available with non-critical services. If there are insufficient unused resources of the system, it will check the unused or released resources of the non-critical service. If it is available, then allocate the available resource.
- Resources are pre-empted from non-critical services. If there are insufficient released resources of the non-critical services, the process will check the resources that are being used by the non-critical service to pre-empt them.

4.2. Resource Balancing

The most important factor in resource reconfiguration is resource balancing in order to maintain the stability of the system. Take note that the service is functional when there is at least one active resource to support it [3]. The system has three tables of resources. When the system is activated and the resource allocated, the table will note every allocation into those tables. For example, if there is a critical service, that needs four resources to run the service, then those four resources will be allocated and noted in the critical service's resources table (resource currently used row) and in the master resources allocation table (required resource row). The same goes to non-critical service. Thus, if the engine allocated four resources for critical service and four resources for non-critical service, there will be eight required resources in the master table. For redundant resources of critical service, initially it must be at least equal to one resource.

The tables are updated dynamically at run-time and when fault happens. For example, if there are some resources of the non-critical services that have been released, the used resources count must reflect the change. If some resources of the critical service are destroyed, the used resources count must be deducted as well. The required resource of the master table does not change. In our model it is assumed that there will be no further faults to the affected critical service node while it is being reconfigured.

4.3. Recovery Scenario

Now, we explain the recovery steps per each scenario.

Scenario 1: redundant resources available with critical service. If critical service resources destroyed, then firstly it will check its redundant resources to complete the required ones. If available (enough) then allocate.

```
begin
input damaged_resource;
input CS_redundant;
if (CS_redundant-damaged_resource) >= 0 then
    allocate(CS_redundant);
else
    loop until CS_redundant = 0
    allocate(CS_redundant);
    end loop;
    output(damaged_resource);
end if;
end
```

Scenario 2: redundant resources available with the system. If required resource not available with critical service node then look in master table. Check whether there are unused redundant resources available. If available then allocate.

```
begin
input damaged_resource;
input system_unused_redundant;
if (system_unused_redundant-damaged_resource) >= 0 then
    allocate(system_unused_redundant);
else
    loop until system_unused_redundant = 0
    allocate(system_unused_redundant);
    end loop;
    output(damaged_resource);
end if;
end
```

Scenario 3: released resources available with non-critical services. In case scenarios 1 and 2 fail to allocate the resources then look into the non-critical service table for unused or released resources. If available then allocate.

```
begin
input damaged_resource;
input NCS_released_redundant;
if (NCS_released_redundant -damaged_resource) >= 0 then
    allocate(NCS_released_redundant);
else
    loop until NCS_released_redundant = 0
    allocate(NCS_released_redundant);
    end loop;
    output(damaged_resource);
end if;
end
```

Scenario 4: resources are pre-empted from non-critical services. If scenarios 1, 2 and 3 fails then pre-empt the running resources from the non-critical services based on minimum cost and time factors and if possible including minimum interruptions to non-critical services.

```
begin
input damaged_resource;
input NCS_used_resource;
if (NCS_used_resource-damaged_resource) >= 0 then
    allocate(NCS_used_resource);
else
    loop until NCS_used_resource = 0
    allocate(NCS_used_resource);
    end loop;
    output(damaged_resource);
end if;
end
```

Since the 4th scenario is the most important, Figure 4 shows a sample scenario graph for scenario no 4.

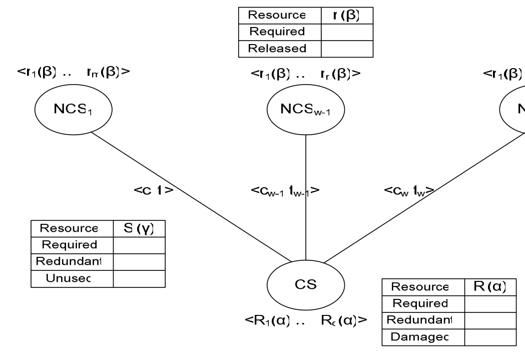

Figure 4. Graph of scenario 4.

CS is the critical service, NCS is the non critical service, R_i is the critical service resource, r_i is the non critical service resource, S_i is the master resource controller, $R_i(\alpha)$ is the amount of each CS resource, $r_i(\beta)$ is the amount of each NCS resource, $S_i(\gamma)$ s the amount of each master resource, ξ is the number of pre-empted NCS node, c is the cost of pre-empting the resource of NCS, t is the response time of NCS, and the edge is represented by a tuple $<c,t>$.

Here we divided the fourth scenario into three cases of solution.

Case 1 $\xi = 1$: in this case, as CS resource has been destroyed, there is only one NCS with the same resource that can be pre-empted. $R_k(\alpha)$ has been destroyed, where $\alpha \geq 1$. We can get the α from pre-empted r_k of NCS_1, such that $r_k(\beta) \geq R_k(\alpha)$, then we pre-empt α from $r_k(\beta)$.

Case 2 $\xi = 1$: in this case, as CS resource has been destroyed, there are more than one NCS with the same resource that can be pre-empted, and the process will choose the best one. $R_k(\alpha)$ has been destroyed, where $\alpha \geq 1$. We can get the α from pre-empted r_k of $\{NCS_1,...,NCS_Z\}$, such that $r_k(\beta) \geq R_k(\alpha)$, with condition $\min(c_i,...,c_z)$ or $\min(t_1,...,t_z)$, then we pre-empt α from $r_k(\beta)$. If there are two possibilities, one is minimum cost but not for response time, another one is minimum time but not for cost, then we should choose the minimum response time, as we want to survive the system.

Case 3 $\xi > 1$: in this case, as CS resource has been destroyed, there is more than one NCS combination with the same resource that can be pre-empted, and the process will choose the best one.

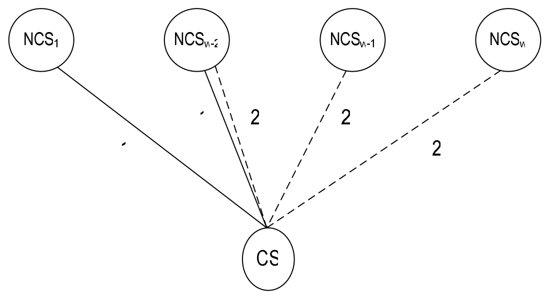

Figure 5. Sample case of scenario 4.

Here assumed there are two solutions, as Figure 5, we have to choose one. $R_k(\alpha)$ has been destroyed, where $\alpha \geq 1$. Let δ be the scheme. We can get the α from pre-empted r_k of: $\delta_1 = \{NCS_1, NCS_{W-2}\}$, such that $\sum_{k=1}^{w} r_k(\beta) \geq R_k(\alpha)$ where its cost $C_1 = c_1 + c_{w-2}$ and time $T_1 = \max(t_1, t_{w-2})$; and $\delta_2 = \{NCS_{W-2}, NCS_{W-1}, NCS_W\}$, such that $\sum_{k=1}^{w} r_k(\beta) \geq R_k(\alpha)$ where its cost $C_2 = c_{w-2} + c_{w-1} + c_w$ and time $T_2 = \max(t_{w-2}, t_{w-1}, t_w)$; with condition $\min(C_1,C_2)$ or $\min(T_1,T_2)$ or $\min(\xi_1,\xi_2)$, then we pre-empt α from $r_k(\beta)$. For this case, we have to consider about the number of ξ as well. If there are two possibilities, one is minimum cost but not for response time, another one is minimum time but not for cost, then we should look at the number of NCS, the minimum one will be chosen.

Hence the conclusion of analysis and allocation process if there is more than one possibility for the fourth scenario, can be applied to scenario 3, will be:

- Check the number of released or pre-empted resource.
- Check the response time of NCS.
- Check the cost of taking NCS resource.

5. Single Fault Case

This section explains the practical application of the model by hypothetical data for single fault. It is supposed a failure detected on resource (R_2) of critical service, CSD_{12}. 20 number of delivery units are being faulted, $CSD_{12} = 20$. The amount, response time, and cost of each service are listed as Tables 4, 5, 6 and 7.

Table 4. Master idle/unused resource.

Resources	Amount	Time	Cost
$MUNR_2$	4	8	7

Table 5. Redundant resources of critical service data.

Resources	Amount	Time	Cost
CSR_{12}	8	5	8
CSR_{22}	9	3	7
CSR_{32}	5	7	3

Table 6. Redundant resources of non critical service data.

Resources	Amount	Time	Cost
$NCSR_{12}$	8	5	7
$NCSR_{22}$	11	5	4
$NCSR_{32}$	18	4	3

Table 7. Currently used resources of non critical service data.

Resources	Amount	Time	Cost
$NCSCU_{12}$	6	2	3
$NCSCU_{22}$	3	6	4
$NCSCU_{32}$	2	4	3

Based on the scenario algorithms in previous section, the possible solutions that satisfy the requisite condition are:

$\delta_1 = (CSR_{12}(8), MUNR_2(4), NCSR_{12}(8))$, $T(\delta_1) = \max(5,8,5)$, $C(\delta_1) = (8*8) + (4*7) + (8*7)$, $N(\delta_1) = 3$.

$\delta_2 = (CSR_{12}(8), MUNR_2(4), NCSR_{22}(8))$, $T(\delta_2) = \max(5,8,5)$, $C(\delta_2) = (8*8) + (4*7) + (8*4)$, $N(\delta_2) = 3$.

$\delta_3 = (CSR_{12}(8), MUNR_2(4), NCSR_{32}(8))$, $T(\delta_3) = \max(5,8,4)$, $C(\delta_3) = (8*8) + (4*7) + (8*3)$, $N(\delta_3) = 3$.

By comparing between the three possible solutions, the δ_3 is selected as the reconfiguration scheme based on cost factor.

6. Summary and Future Work

The research on survivability has become an interesting topic in the field of security, and one of its emphases is how to improve the abilities of emergency response and damage recovery.

In this paper, we presented our preliminary attempts at defining a model to recover a critical service of a system and our plan is to set up a mathematical model for simple basis decision support system for survivability. We proposed an algorithm to analyze and allocate resources for reconfiguring the resources by assigning redundant resources to the critical services and pre-empting resources from non-critical services. The limiting conditions were the response time, minimally pre-empting resources from non-critical services and cost of the implementation. All these limitations have been considered in this paper.

We would like to extend this model for another arbitrary number of faulty critical and non-critical services running concurrently and to consider other limiting conditions as well.

References

[1] Aung A., "Survival of the Internet Application: A Cluster Recovery Model," *in Proceedings of the 6th IEEE International Symposium on Cluster Computing and the Grid Workshop*, UK, pp. 16-19, 2006.

[2] Ellison R., "Survivable Network Systems: An Emerging Discipline," *Technical Report, CMU/SEI-97-153*, 1997.

[3] Elnozahy N. and Plank S., "Checkpointing for Peta-Scale System: A Look into Future of Practical Rollback-Recovery," *Computer Journal of IEEE Transactions on Dependable and Secure Computing*, vol. 1, no. 2, pp. 97-108, 2004.

[4] Goseva K., "Characterizing Intrusion Tolerance Systems Using State Transition Model," *in Proceedings of DARPA Information Survivability Conference and Exhibition*, California, pp. 104-113, 2001.

[5] Ho S. and Cheung W., "Generelized Survivable Network," *Computer Journal of IEEE/ACM Transactions on Networking*, vol. 15, no. 4, pp. 737-749, 2007.

[6] Somesh J. and Wing M., "Survivability Analysis of Networked Systems," *in Proceedings of ICSE 2001*, Toronto, pp. 97-111, 2001.

[7] Knight J., "Error Recovery in Critical Infrastructure Systems," *in Proceedings of Computer Security Dependability and Assurance (CSDA'98) Workshop*, California, pp. 7-40, 1998.

[8] Lin X., "A Framework for Quantifying Information System Survivability," *in Proceedings of International Conference on Information Technology and Applications (ICITA)*, Australia, pp. 234-237, 2005.

[9] Moitra D. and Konda L., "A Simulation Model for Managing Survivability of Networked Information System," *Technical Report, CMU/SEI-2000-TR-020*, 2000.

[10] Park J. and Chandramohan P., "Static vs. Dynamic Recovery Models for Survivable Distributed Systems," *in Proceedings of 37th Hawaii International Conference on System Sciences*, USA, pp. 5-8, 2004.

[11] Park B., *A Self-healing Mechanism for an Intrusion Tolerance System*, Copenhagen, Springer, 2005.

[12] Sullivan K., Knight J., Du X., and Geist S., "Information Survivability Control System," *in Proceedings of 21st Intermational Conference on Software Engineering*, Germany, pp. 107-119, 1999.

[13] Wang J., Wang H., and Zhao G., "ERAS an Emergence Response Algorithm for Survivability of Critical Services," *in Proceedings of IMSCCS 2006*, USA, pp. 312-317, 2006.

[14] Westmark R., "A Definition for Information System Survivability," *in Proceeding of the 37th Hawaii Internal Conference on Systems Sciences (HICSS'04)*, USA, pp. 5-8, 2004.

[15] Zhao G., Wang H., and Wang J., "A Novel Quantitative Analysis Method for Network Survivability," *in Proceedings of IMSCCS,* USA, pp. 22-27, 2006.

Irving Paputungan is a trainee lecturer at the Islamic University of Indonesia, Indonesia. He holds Bachelor degree in informatics engineering from the Islamic University of Indonesia, 2003 and his Master in computer and information science from Universiti Teknologi Petronas Malaysia. His research area is computational and database.

Azween Abdullah obtained his Bachelors degree in computer science in 1985, Master in software engineering in 1999, and his PhD in computer science in 2003.