# Critical Service Recovery Model for System Survivability

IRVING VITRA PAPUTUNGAN[1] , AZWEEN ABDULLAH[2] , LOW TAN JUNG[3]
Computer and Information Sciences Department
Universiti Technologi Petronas
Bandar Sri Iskandar 31750 Tronoh
MALAYSIA

*Abstract*: - In this paper, we propose a recovery model to enhance system survivability. The model focuses on how we preserve the system and resume its critical service while incident occurs by reconfiguring the system based on the remaining available resources without affecting the stability of the system. There are three motivating factors in our recovery model, the response time of reconfiguration, the cost of reconfiguration, and the number of pre-empted non-critical service resources. The adoption of fault-tolerance using redundancy in our model is discussed from a new perspective.

*Key-Words :* - critical service, fault-tolerance, reconfiguration, resources, survivability, redundancy

## 1. Introduction

System survivability breaks limits of traditional security concepts, and emphasizes the abilities of a system to achieve its critical service in a timely manner when it is suffering from disruptive faults or accidents [1] [7]. That the systems can achieve their critical services and recover the damaged services as soon as possible when damages are detected, is the basic idea of survivability, even after the main components are damaged or destroyed. The condition of any system is that there is no absolute security to avoid from failure because of disruption, faults, or accidents [2]. In order to safeguard the system to provide services stably and reliably when fault occurs, we must consider a technique for enhancing system survivability [3].

Recently, the related 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. [2] 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. [6] 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 the system can reconfigure by de-allocating destroyed resources under faults, and ultimately keep the critical services running all along, the system will survive. In this paper, we address the recovery problem by resources reconfiguration and reallocation. There are some good approaches and effective techniques of how to recover the system, such as ERAS [9], Hybrid Model [4], Cluster Recovery [12], and Survivability Framework [5].

In our model, we adopt the concepts of [5], [9], [12], and [4] combined with redundancy to recover the system back to the good state. In our model we assign resource redundancy at the outset and keep a table for each critical and non-critical service nodes. 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.

We start with summarizing briefly the characteristics of the system we addressed. Next is the State Transition Model [8] 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. It is a good thought, because most of the recovery techniques placed after incident occurred, the system fails to run. In this paper services and processes are used interchangeably. It is worth

mentioned here that the model proposed in this paper is a generic model that can be applied in any systems i.e. readers should not restrict him/herself to just computer systems or computer network systems. Therefore terms like attack, nodes, real-time, etc are not restricted to only the computer systems or computer network systems.

## 2. The System Characteristics

There are a number of characteristics that the systems posses in whole or in part which are important in constraining the ways which these system approach fault recovery. The charac-teristics are as follows [10]:

- Heterogeneous and Very Large number of nodes.
- Different structure of critical infrastructure.
- Required good performance.
- Extensive database and low level redundancy.

## 3. The System Model

Error recovery in distributed applications can take several forms: system-level approach to fault tolerance, fault tolerance in wide area networking applications, and reconfigurable distributed systems. Our focus is reconfiguring the system through de-allocation of destroyed resources and reallocation of remaining available resources based on redundant or pre-empted resources at the fastest possible time with minimum service node disruption.

Figure 1 depicts the state transition model which we use 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 it enables a user to read or modify information without authorization, grant or deny an entity access to a resource without authorization. "Without authorization" means in violation of the system's security policy. Vulnerability is the property of the system, its attendant software and/or hardware, or its administrative procedures, which causes it to enter vulnerable state. The system enters fault state when vulnerability is successfully exploited and the fault unmasked by simple fault tolerance. In the next state, recovery state, since the system needs to recover, limit the damage and protect the system from DoS, it goes into graceful degradation mode, while maintaining the critical services. Critical services are defined as the functions of the system that must be maintained to meet the system requirements even when the failures occur that threaten the system. In order to survive the critical service, we assume the recovery process will always successful; hence there is no fail state.

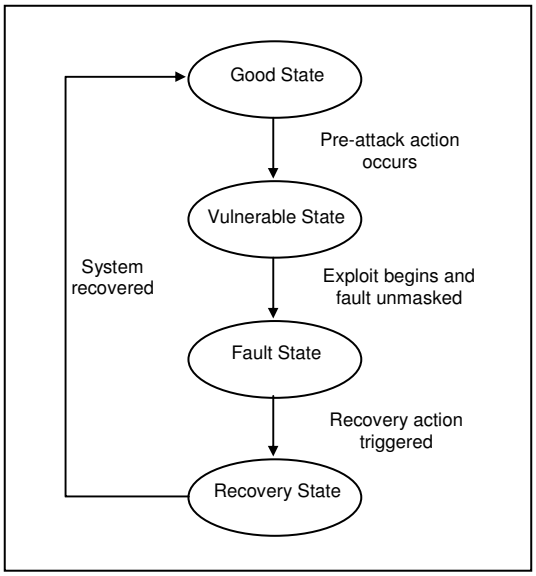


Fig.1 System State Transition Model

## 4. The System Recovery Model

It will often be the case that the effects of a fault leave the system with greatly reduced resources (processing services, communications capacity, etc) and substantial changes in the service provided to the user will be necessary. The ability to tolerate certain types of fault is the only practical approach to achieving survivability. Beside considering single fault for certain critical service, we will have to consider multiple single sequential faults and multiple concurrent faults. In the first instance, the system has to reconfigure the reconfigured system and in the latter, reconfiguration must be simultaneous and consistent in order to maintain stability. Our approach focuses on the first instance as well.

The recovery engine defines requirements for the availability of the system. It guarantees that tasks can be performed exactly at the required moment, which an access is possible at the required moment and that resources are not demanded unnecessarily or are withheld.

This includes all functions ensuring that the resources are available and usable when demanded by an authorized subject (e. g. user or a process performing a user task) and all functions that prevent or restrict an influencing of time-critical operations.

The recovery engine shall comprises all functions for:

- The error detection and bridging,

- The limitation of consequences of errors to the operation (of the system),
- The minimization of interrupts and performance loss,
- The reaction to external demands and outputs within fixed time (the focus point of this paper).

### 4.1 Problem Definition

The problem can be described as: There is a Real Time System in which has critical service and its resources are destroyed by fault. The critical service can be high level critical service or medium level critical service, and low level critical service. In order to sustain its running, it needs some resources to reconfigure the damaged one. The resources can be taken from 1) Redundant Resources of critical service, 2) Unused Redundant of the system, 3) Released Resources of non-critical service, and 4) Pre-empt Resources of non-critical service that are being currently used. In Real-Time-System, each resource will be changed frequently. It means the amount of those resources is different from time to time. The stability of the system must be maintained, even though the unstable condition occurred. It means that the number of pre-empted non-critical service resources should be as few as possible. Since this process placed under graceful degradation mode, we need to consider the duration of the process. The duration, included the response time and the usage time, must be less then the graceful degradation time. Here we focus on the response time of pre-empting non-critical service resources. The problem is to find out how to reconfigure the critical service which can ensure sustainable operation of critical services. The problem is depicted in figure 2. We assume that error detection (monitoring) and damage assessment are taken care by some other mechanism such as control system architecture [11] and we also assume that there will be no further fault can be caused on to the critical service when it is already damaged partially or completely.

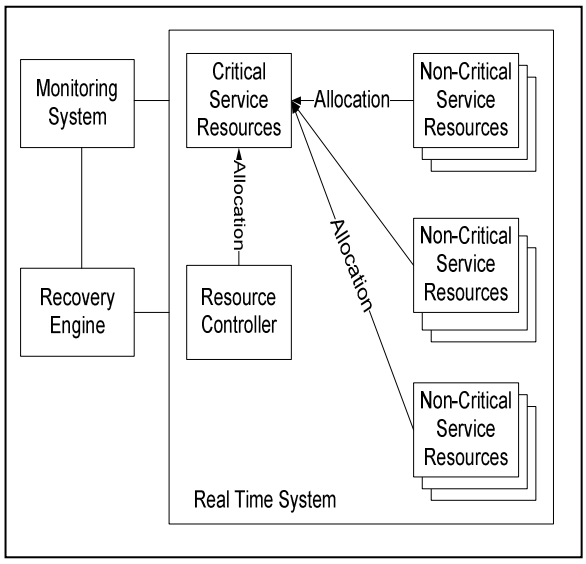


Fig.2 Abstract Problem Definitions

### 4.2 Reconfiguration Process

In the recovery state, we mapped our recovery actions. The process starts with diagnosing the destroyed resources that the critical services used. Then it will analyze the available resources that can be used for reconfiguration. And to move back to good state, we allocate the available resources to reconfigure critical services resources. See figure 3.

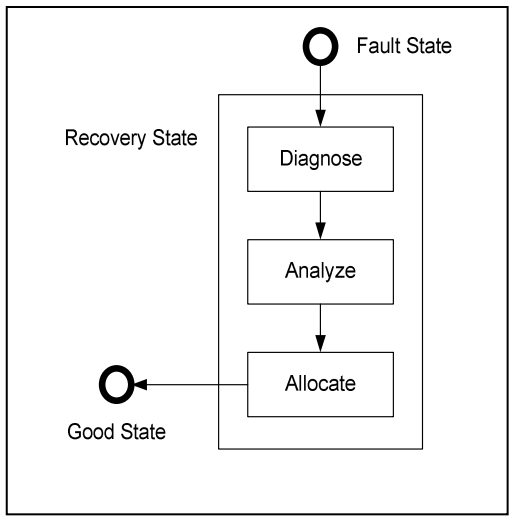


Fig.3 Recovery Process

In the diagnosing process, we assess the damaged resources. The number of critical service resources that are destroyed.

We define all the four resources, described in previous sections, 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. See table 1, 2, and 3.

| Resource | $R_1$ | $R_2$ | $R_n$ |
|---|---|---|---|
| Required Resource | | | |
| Used Redundant | | | |
| Unused Redundant | | | |

Table 1 Master Resources Allocation Table

| Resource | $R_1$ | $R_2$ | $R_n$ |
|---|---|---|---|
| Resource Currently Used | | | |
| Redundant Resource | | | |
| Damage Resource | | | |

Table 2 Critical Service Resources Table

| Resource | $R_1$ | $R_2$ | $R_n$ |
|---|---|---|---|
| Resource Currently Used | | | |
| Released Resource | | | |

Table 3 Non-Critical Service Resources Table

*The master resource allocation table* (Table 1) shows the total resource currently used and allocated for redundancy inside the running system. Required resource means the resource 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 redundant resources that are not currently used by the system. $R_1$, $R_2$, and $R_n$ are the resources of the system.

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

*The non-critical services resources table* (Table 3) shows the resource 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.

In the analysis process, there are some scenarios to analyze after an attack happens to the critical service resources.

1. Checking the redundant resources of the critical service
2. If there is not enough redundant resources of the critical service, it will check the unused redundant of the system
3. If there is not enough unused resources of the system, it will check the released resources of non-critical service
4. If there is not enough released resources of non-critical service, it will check the resources that currently used by non-critical service.

In the last process, allocation, we find the feasible scheme of reconfiguration from available resources that is optimum for critical service.

### 4.3 Resources Balancing

The most important thing in resource reconfiguration is resource balancing, in order to keep the stability of the system. We have three tables of resources. When the system is on and the resource allocated, the table will note every allocation into those tables. For example, if there is a critical service, it needs four resources to run the service. It means those four resources will be allocated and noted in the critical service resources table (resource currently used row) and in the master resources allocation table (required resource row). The same way goes to non-critical service. Thus, if we 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, its number must be at least equal to one resource.

The tables are updated dynamically when the system runs and faulted. For example, if there are some resources of the non-critical service that has been released, the used resources must be deducted. If there are some resources of critical service destroyed, the used resources must be deducted as well. But the required resource of master table does not change. In our model we assume that there will be no further faults to the affected critical service node while it is being reconfigured.

## 5. Scenario Analysis

We will explain the practical application of Critical Service Recovery model by a certain system in a given environment for each scenario. We start up assuming that the system detects the resources that critical service uses are destroyed in a certain run. In order to sustain its running, the system should provide resources for the critical service. In the entire scenario below, the master resource allocation table values must equal to the sum of values in the critical and non-critical services tables. The values are updated or balanced on a real-time basis. We define the three tables below with arbitrary number of resources to explain our concept.

| Resource | $R_1$ | $R_2$ | $R_3$ |
|---|---|---|---|
| Required Resource | 15 | 15 | 15 |
| Used Redundant | 6 | 6 | 6 |
| Unused Redundant | 5 | 5 | 5 |

Table 4 Master Resources Allocation Table

| Resource | $R_1$ | $R_2$ | $R_3$ |
|---|---|---|---|
| Resource Currently Used | 5 | 5 | 5 |
| Redundant Resource | 2 | 2 | 2 |
| Damage Resource | 0 | 0 | 0 |

Table 5 Critical Service Resources Table

| Resource | $R_1$ | $R_2$ | $R_3$ |
|---|---|---|---|
| Resource Currently Used | 10 | 10 | 10 |
| Released Resource | 0 | 0 | 0 |

Table 6 Non-Critical Service Resources Table

There is a critical service that needs 3 kinds of resources ($R_1$, $R_2$, and $R_3$) to runs its service. The critical service needs 5 resources for each resource to accomplish its mission. The non-critical service needs 10. So the required resource equals to 15.

*Scenario 1*: Redundant Resources available with critical service

| Resource | $R_1$ | $R_2$ | $R_3$ |
|---|---|---|---|
| Required Resource | 15 | 15 | 15 |
| Used Redundant | 6 | 6 | 6 |
| Unused Redundant | 5 | 5 | 5 |

Table 7 Master Resources Allocation Table

| Resource | $R_1$ | $R_2$ | $R_3$ |
|---|---|---|---|
| Resource Currently Used | 3 | 5 | 5 |
| Redundant Resource | 2 | 2 | 2 |
| Damage Resource | **2** | 0 | 0 |

Table 8 Critical Service Resources Table

| Resource | $R_1$ | $R_2$ | $R_3$ |
|---|---|---|---|
| Resource Currently Used | 10 | 10 | 10 |
| Released Resource | 0 | 0 | 0 |

Table 9 Non-Critical Service Resources Table

One of the resources $R_1$, of a critical service is destroyed. It requires 2 resources to fix it. Since there are enough redundant resources from the critical service (5), the problem can be easily fixed without affecting any other services.

*Scenario 2*: Redundant resources available with the system

| Resource | $R_1$ | $R_2$ | $R_3$ |
|---|---|---|---|
| Required Resource | 15 | 15 | 15 |
| Used Redundant | 6 | 6 | 6 |
| Unused Redundant | 5 | 5 | 5 |

Table 10 Master Resources Allocation Table

| Resource | $R_1$ | $R_2$ | $R_3$ |
|---|---|---|---|
| Resource Currently Used | 1 | 5 | 5 |
| Redundant Resource | 2 | 2 | 2 |
| Damage Resource | **4** | 0 | 0 |

Table 11 Critical Service Resources Table

| Resource | $R_1$ | $R_2$ | $R_3$ |
|---|---|---|---|
| Resource Currently Used | 10 | 10 | 10 |
| Released Resource | 0 | 0 | 0 |

Table 12 Non-Critical Service Resources Table

One of the critical resources $R_1$ is destroyed. It requires 4 resources to fix it. Unfortunately there are only 2 redundant resources. In this case, the recovery process will be fixed by adding 2 resources from the unused redundant resources of the system; from the master resource allocation controller.

*Scenario 3*: Released resources available with non-critical services

| Resource | $R_1$ | $R_2$ | $R_3$ |
|---|---|---|---|
| Required Resource | 15 | 15 | 15 |
| Used Redundant | 6 | 6 | 6 |
| Unused Redundant | 1 | 5 | 5 |

Table 13 Master Resources Allocation Table

| Resource | $R_1$ | $R_2$ | $R_3$ |
|---|---|---|---|
| Resource Currently Used | 0 | 5 | 5 |
| Redundant Resource | 2 | 2 | 2 |
| Damage Resource | **5** | 0 | 0 |

Table 14 Critical Service Resources Table

| Resource | $R_1$ | $R_2$ | $R_3$ |
|---|---|---|---|
| Resource Currently Used | 7 | 10 | 10 |
| Released Resource | 3 | 0 | 0 |

Table 15 Non-Critical Service Resources Table

One of critical resources $R_1$ is destroyed. It requires 5 resources to fix it. There are only 2 redundant resources of critical service and 1 unused redundant of the system. The problem will be fixed if

the recovery process takes another 2 from released resource of non-critical service resources.

*Scenario 4*: Resources are pre-empted from non-critical services.

| Resource | $R_1$ | $R_2$ | $R_3$ |
|---|---|---|---|
| Required Resource | 15 | 15 | 15 |
| Used Redundant | 6 | 6 | 6 |
| Unused Redundant | 1 | 5 | 5 |

Table 16 Master Resources Allocation Table

| Resource | $R_1$ | $R_2$ | $R_3$ |
|---|---|---|---|
| Resource Currently Used | 0 | 5 | 5 |
| Redundant Resource | 2 | 2 | 2 |
| Damage Resource | **5** | 0 | 0 |

Table 17 Critical Service Resources Table

| Resource | $R_1$ | $R_2$ | $R_3$ |
|---|---|---|---|
| Resource Currently Used | 9 | 10 | 10 |
| Released Resource | 1 | 0 | 0 |

Table 18 Non-Critical Service Resources Table

One of critical resources $R_1$ is destroyed. It requires 5 resources to fix it. There are only 2 redundant resources of critical service, 1 unused redundant of the system and 1 released resource of non-critical service. The problem will be fixed if the recovery process pre-empts 2 resources from non-critical service resources that currently used. However, since we consider about the stability, the usage of it must be as small as possible.

## 5.1 Recovery Steps

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

If scenario 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 scenario 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
```

Beside of those scenarios, we still have to consider about multiple sequential faults. It means there will be other faults after the reconfiguration finished, either to the recent reconfigured critical service or other critical services. All we have to deal with this kind of fault is finishing the reconfiguration one by one using the same way as described above.

Since the fourth scenario is the most important, Fig 4 below shows a sample scenario graph for that scenario.

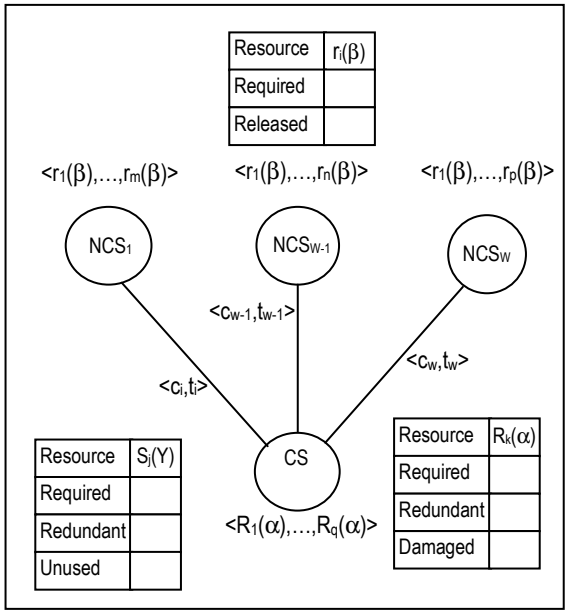


Fig.4 Scenario Graph For Scenario 4

CS = Critical Service
NCS = Non Critical Service
$R_i$ = Critical Service Resource
$r_i$ = Non Critical Service Resource
$S_i$ = Master Resource Controller
$R_i(\alpha)$ = The Amount of each CS Resource
$r_i(\beta)$ = The Amount of each NCS Resource
$S_i(\gamma)$ = The Amount of each Master Resource
$\xi$ = The number of pre-empted NCS node
$c$ = Cost of pre-empting the resource of NCS
$t$ = 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 $\alpha$ from pre-empted $r_k$ of $NCS_1$, such that $r_k(\beta) \geq R_k(\alpha)$, then we pre-empt $\alpha$ 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 $\alpha$ from pre-empted $r_k$ of $\{NCS_1,\ldots,NCS_Z\}$, such that $r_k(\beta) \geq R_k(\alpha)$, with condition $\min(c_i,\ldots,c_z)$ or $\min(t_1,\ldots,t_z)$, then we pre-empt $\alpha$ 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.

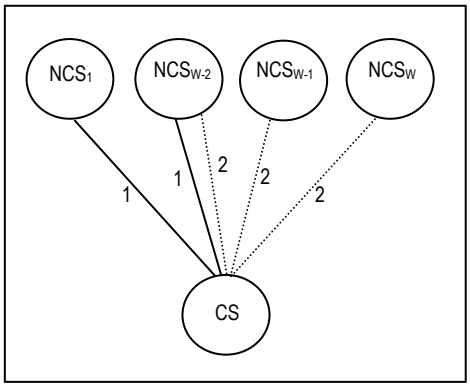


Fig.5 Sample Case 3 Graph

Here assumed there are two solutions, we have to choose one. $R_k(\alpha)$ has been destroyed, where $\alpha \geq 1$. Let $\delta$ be the scheme. We can get the $\alpha$ 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 $C_1 = c_1 + c_{w\text{-}2}$ and time $T_1 = \max(t_1, t_{w\text{-}2})$; and $\delta_2 = \{NCS_{W\text{-}2}, NCS_{W\text{-}1}, NCS_W\}$, such that $\sum_{k=1}^{w} r_k(\beta) \geq R_k(\alpha)$ where its cost $C_2 = c_{w\text{-}2} + c_{w\text{-}1} + c_w$ and time $T_2 = \max(t_{w\text{-}2}, t_{w\text{-}1}, t_w)$; with condition $\min(C_1,C_2)$ or $\min(T_1,T_2)$ or $\min(\xi_1,\xi_2)$, then we pre-empt $\alpha$ from $r_k(\beta)$. For this case, we have to consider about the number of $\xi$ 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.

## 6. Conclusion 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 an arbitrary number of faulty critical and non-critical services running concurrently and to consider other limiting conditions as well.

*References*


[1] R. Westmark, A Definition for Information System Survivability, *Proceeding of the 37th Hawaii Internal Conference on Systems Sciences (HICSS'04), IEEE*, 2004.

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

[3] Somesh. J, J.M. Wing, Survivability Analysis of Networked Systems, *ICSE 2001, Toronto*, 2001.

[4] J. Park, P. Chandramohan, Static vs. Dynamic Recovery Models for Survivable Distributed Systems, *Proceedings of 37th Hawaii International Conference on System Sciences*, 2004.

[5] X. Lin, M. Zhu, R. Xu, A Framework for Quantifying Information System Survivability, *ICITA 2005, IEEE*, 2005.

[6] G. Zhao, H. Wang, J.Wang, A Novel Quantitative Analysis method for Network Survivability, *IMSCCS 2006, IEEE*, 2006.

[7] R. Ellison, D. Fisher, and et al, Survivable network systems: An Emerging Discipline. Technical Report, *CMU/SEI-97-153*, 1997. Revised 1999.

[8] K. Goseva-Popstojanova, et al., Characterizing Intrusion Tolerance Systems using State Transition Model, *DARPA Information Survivability Conference and Exhibition*, June 2001.

[9] J. Wang, H. Wang, G. Zhao, ERAS – an Emergence Response Algorithm for Survivability of Critical Services, *IMSCCS 2006, IEEE*, 2006.

[10] J. Knight, M. Elder, X. Du, Error Recovery in Critical Infrastructure Systems, *Proceedings of the 1998 Computer Security, Dependability, and Assurance (CSDA'98) Workshop*, Williamsburg, VA, November 1998.

[11] K. Sullivan, J. Knight, X. Du, S. Geist, Information Survivability Control System, *Proceeding of 21st International Conference on Software Engineering, IEEE*, 1999.

[12] M.M. Aung, K. Park, J.S. Park, Survival of the Internet Application: A Cluster Recovery Model, *Proceeding of the 6th IEEE International Symposium on Cluster Computing and the grid Workshop*, 2006.