CHAPTER THREE: SELF-REGENERATIVE SYSTEM ARCHITECTURE

In this chapter we present our system and elaborate it, the model is self-regenerative based. In order to meet this self-regenerative goal from biologically inspired mechanism, we should find a biological self-regenerative process so that we can map it to the architecture of our model.

3.1 Self-Regenerating System (SRS)

The goal of the SRS program is to develop technology for building computing systems that provide critical functionality at all times, in spite of damage caused by unintentional errors or attacks. All current systems suffer eventual failure due to the accumulated effects of errors or attacks.

Regeneration means reproducing or reconstructing of a lost or injured body part (Carlson, 2007). In computer science, to achieve the SRS program goals, the program will address several key technology areas, such as detection, configuration, updating, ect.

3.2 Bio-inspired

Bio-inspired is the strategies which is acquired or extracted from biology and mapped to computer system in order to have the algorithms typically behaving as a bio-system. The resulting of this mapping will be a system that is robust and regenerative, ect. By adopting properties of biological systems, designed systems operate adequately even in the presence of catastrophic failures and large scale attacks (George, 2003).

We introduce the biologically inspired mechanism, which we have used in the system; the bio-inspired technique we have found is the cell biochemical reaction during the regeneration. In fact, when cell is performing division it has some important process in order to ensure the success of the regenerative activity.

3.3 Cell regeneration and Internal Operations

Cell biochemical reactions contain very complicated operations. When cell initiate division, it starts complex procedure of producing a new cell based on copying the DNA. DNA contains the information of every molecules that acting and performing activities inside the cell. This will lead to enhance networking survivability if applied to Multi-agent system with communication between agents in distributed nodes.

When hormones approach a cell, they first look for a door that to access the cell. Every molecule is stopped by the door, to see whether it is beneficial to the cell or not, the door open only for useful molecule. If a harmful material such as a virus tries to enter the cell door, the situation changes, the cell door analyze the material, discover that it is a harmful and reject it.

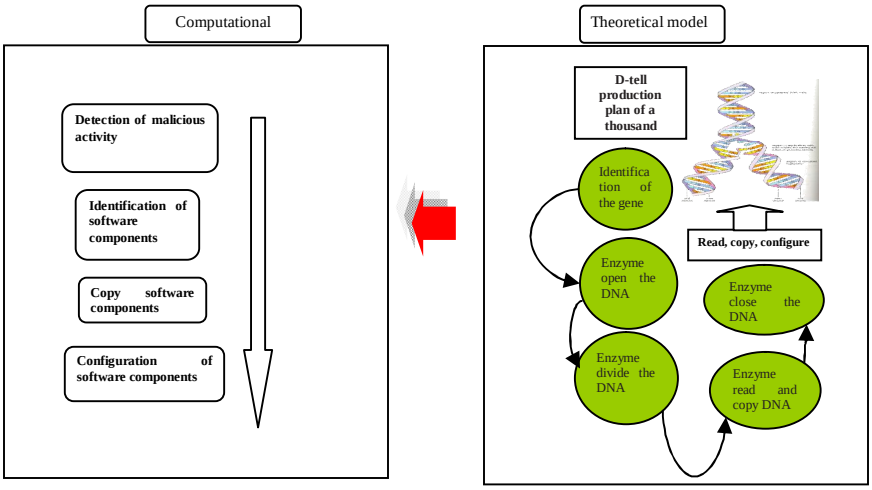


Figure 3-1: Mapping of the Cell regeneration to our Model

When entering the cell, a combination of advance technology and complexity is found. When hormone steps in the door, it immediately taken under control by special protein which carries out the function of the cell, this protein is called enzyme. If required, enzyme immediately put the newly arrived in use, if there is no need for this hormones at that moment, they are placed in storage compound of the cell.

If cell initialize regeneration, first it has procedure and plan for this system which coded in a DNA chain. This coding system includes the production plan of a thousand different enzyme and protein used in the cell. The project of all organic molecules that would be structured in the cells is written in a DNA down to smallest procedure. The production of organic molecule such as protein starts with the identification of the gene that contains required information among the DNA found in the chromosome the enzyme exclusively in charge of this task open the DNA. Another group of enzymes come and divide the DNA into tow, other enzyme goes over one pieces and quirkily read and copy the data. When the replication process is complete, two DNA molecules — identical to each other and identical to the original — have been produced. This mode of replication is described as semi-conservative: one-half of each new molecule of DNA is old; another is new (Camazing et al, 2003).

Now the perfect copy of production plan in a DNA is obtained after completion of the replication process. An enzyme closes the DNA and restores it to the original states. The copy produced from the DNA is called messenger RNA; RNA contains the production plan of protein required for the cell.

The aforementioned natural model can be applied for network software survivability and restoration of resources so as achieve continuity of the services. In our model we have agent that works as enzymes, agent detect components defect in real time. After any detection of software components failure, message is sent by the detection agent to others agents to act. In cell operation we see enzymes generate information concerning procedures of the DNA to be .

3.4 Bio-inspired Self-Regenerative Model

Our Self-regenerative model is a process whereby any system can have the ability to recover from failure. In applying this mechanism to the system, we introduce a multi agent system to model the ability, basically with the purpose of monitoring and detecting the activity of the nodes. The detection system detects the attacks, then by emphasis all nodes to enforce security system and prevent any suspected incoming malady.

The system executes an emergency procedure to first analyze and diagnose the attacks, infected component of the software, afterward performing the localizations of component for the availability.

After detection of attacks the system would deliver signal to all nodes, theses signals or messages are categorized into two, from one node to many node, and from many nodes to many nodes. This messages dedicated to inform nodes them about the incoming attacks and the potential harm expected harm. Furthermore, the nodes would duplicate the component of software respectively to secure continues execution of program.

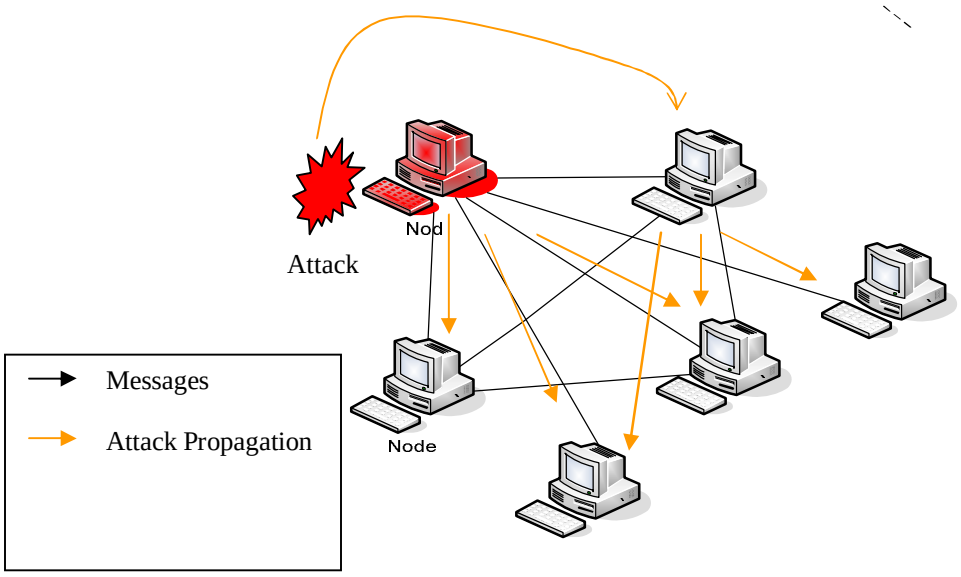


Figure 3-2: System Overview

Here we introduce the notion of cell regeneration Model as an approach to self-regenerate network software. Figure 3-2 shows the network nodes and the communication of software agent throughout the network. Network reliability can be determined by its ability to automatically recover rapidly from attacks and to ensure survivability and availability of software. This ability relies on so many parts such as the ability to determine whether a behavior of a particular program is intrusive or not.

We use agent to describe the procedure, agent can be defined to be autonomous (Hyacinth, 1996) problem solving computational entity, capable of effecting operation in dynamic and open environment, agent can multiple and can have interaction between them, multi-agent system have been proposed for our various process: detection, malicious software part, configuration, and protection part.

3.5 Agent Based System

Agent-based modeling facilitates the implementation of tools for the analysis of environment change. An agent can characterize an individual with capabilities to perceive and react to events in the environment, taking into account its mental state (beliefs, goals), and to interact with other agents in its MAS environment (Pavo´n et al, 2007)

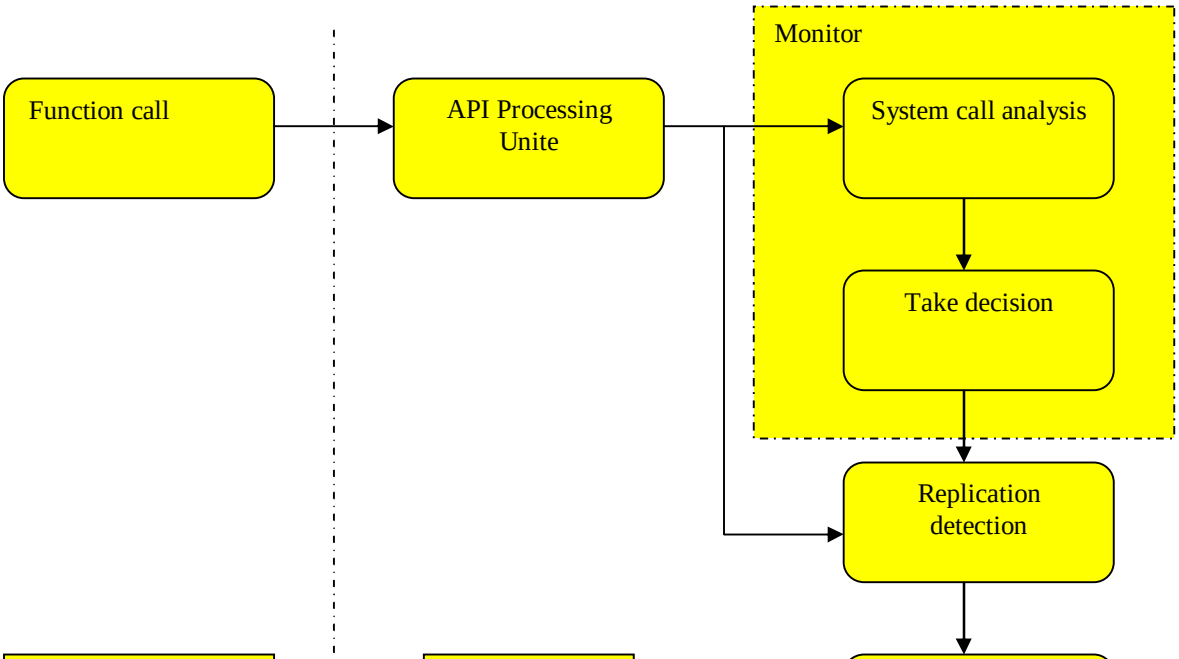
Agent has a characteristic behavior that differentiates itself from other kind of software; there character is used in case of modeling behaviors of agent in system. Some of the attributes are:

- Reactivity: the ability to selectively sense and act.
- Autonomy: goal-directedness, proactive and self-starting behavior
- Collaborative behavior: can work in concert with other agents to achieve a common goal.
- “Knowledge-level” (Newell 1982) communication ability: the ability to communicate with persons and other agents with language more resembling humanlike “speech acts” than typical symbol-level program-to-program protocols

- Inferential capability: can act on abstract task specification using prior knowledge of general goals and preferred methods to achieve flexibility; goes beyond the information given, and may have explicit models of self, user, situation, and/or other agents.
- Temporal continuity: persistence of identity and state over long periods of time.
- Personality: the capability of manifesting the attributes of a "believable" character such as emotion.
- Adaptively: being able to learn and improve with experience.
- Mobility: being able to migrate in a self-directed way from one host platform to another (Pavo´na et al, 2007).

In our model we are going implement and add the GSR monitoring technique into the detection part to trigger the other part of the system. The monitor will detect the function call and identify then analyses the call to make a decision if the function call is a pattern of any malicious.

Replication activities also monitored and set, if any malicious try to replicate within a file or performing a replication to different node, the monitor will detect. After the detection the agent sent messages to perform actions.



As we can see the monitor decide whether the faction has been successfully executed or not, if the call is done the monitor send messages to a particular agent, in case of the call detected prior to the execution also the agent send message to a previously agent

3.8 Run Time Management

RTM is the control system associated with a component that continuously monitors the component operations, analyzes the current state, plans the appropriate corrective actions if needed, and executes these actions to bring the component back to acceptable normal state of operation. The RTM control and management algorithm is shown in below.

```
If message received
Receive component (ID, location..ect)
Check component
While (true)
    If (any changes in CMI management polices)
        Read and update CMI polices
        Execute component
    end if
```

end while
end if

The CMI provides three ports to specify the control and management requirements associated with each software component and/or network resource, as the configuration attributes that are required to automate the process of setting up the execution environment of a software component, And the policies that must be enforced to govern the operations of the component and/or a resource as well as its interactions with other components or resources.

3.9 The Proposed Architecture of Self-regenerative System

This section carries out a detail description of the architecture presented is representing the self-regenerative system, the model is designed based on the behavior monitoring explained previously. Firstly an overall structure of the agent involved in the process is modeled.

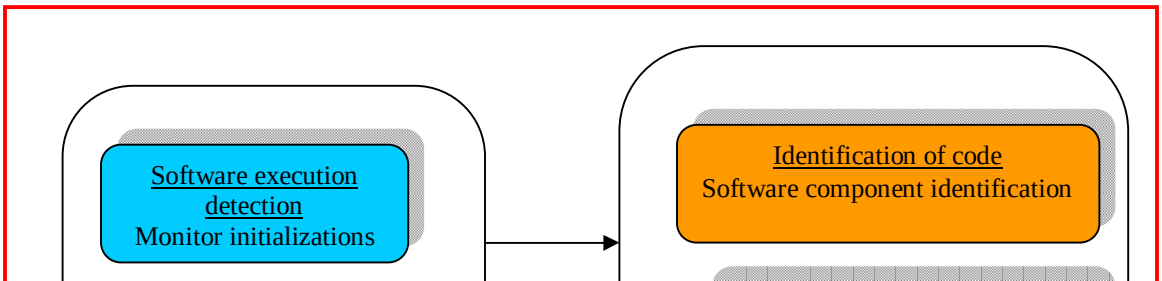


Figure 3-7: The Proposed Self-regenerative Model

We have in this module the component of the procedure that is involving in order to perform the self-regeneration, it perform detection of attacks software which is initializing an execution, this software may be useful, and some of them may be intrusive and dangerous, and could cause damage to the normal software.

In our system (Model) we have a four component, these component are working continuously to monitor all system behavior:

- Detection.

- Replication and configuration,
- Run time management.
- Prevention.

The first component (i.e Detection) perform the detection of the instructions and the purpose of the instructions or function call, according to the instruction analyze the system will decide whether it is harmful or not, if it is harmful, the system would initialize a signaling to all nodes in this network, this signals or massages aim to inform nodes about the initialization and the execution of the malicious software which is found in the first node.

The first node would identify the malicious software. According to the prevention technique of the other node, nodes would tries to prevent the particular malicious software from propagation, and therefore we will limit the propagation of malicious software.

3.10 Functions Description

In this section we elaborate our agents, as mentioned previously, that we have three kind of agent:

- Detection agent.
- Replication and configuration agent
- Runtime management agent.
- Prevention agent.

To describe more clearly we implement finite state theory to have more details of the procedure, and according to our module we have set of states and input, as it assumed above in our model we have four participants.

Each agent has more than one state during the execution, according to the environmental change, the agent sense the change and act accordingly.

Let's say if we have an attack trying to carry out some change or modification in the software component, in this case the detection part monitor and detect an act of software execution, additionally will detect intrusion and function call that involving in the execution of a particular command. We have precise previously the instruction that is totally intrusive and would cause damage to the software will be categorized as harmful or attack, and then this function call will be evaluated whether it is done executed or not. Furthermore, we can see in the first of the detection the malicious software may not tries to cause any harm to this particular Node, it will try to propagate first and then attack . We have also signaling to other nodes for strengthen their protection in case of replication.

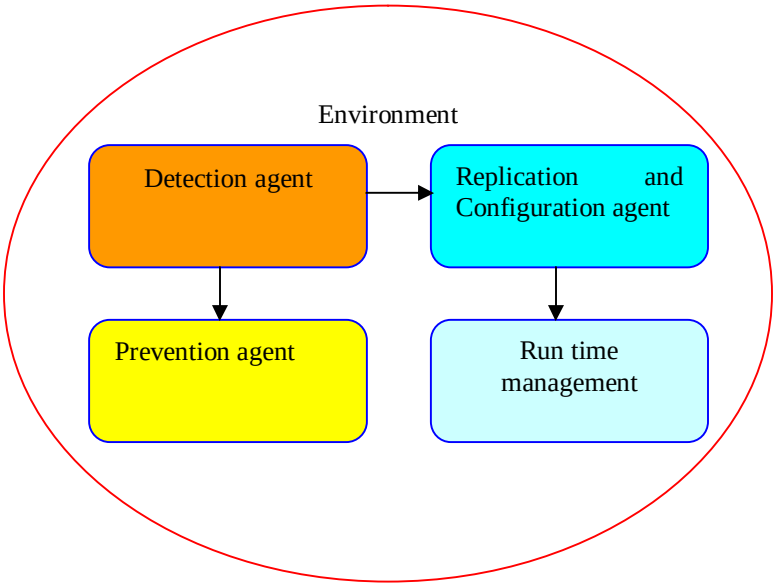


Figure3-8: self-regenerative Component

3.10.1 Detection agent

The detection agent has a set of states, and set of input, one start state and set of accepting states, it also has a transition function, our event in this agent are :

1. The detection detects a new software execution.
2. The detection detects an instruction or function call.
3. The detection analyze the instruction or function call to see whether it is intrusive of not, in order to analyze the instruction, the system would check the content of

the instruction and argument or function call. If the content of the instruction has a command such as addition to any other software of delete the agent of any other software, the state would a potential harm to the software. This event would be considered a harmful.

4. The detection evaluates to see whether the instruction is already done or not.
5. The detection detect whether the malicious software is trying to propagate to other node fro more infection or not.
6. The act of signaling to other nodes.
7. The detection of component fault.

Theses event are totally done with care, such detection of any software or any other event will be checked up precisely, the event must confirm that it has been done, and this would cause an initialization of other event respectively throughout the regeneration cycle.

In diagram below we show the event of this agent. for example, the action or monitoring Software execution would affect only this agent and would cause a transition to another state, let us examine first this automaton, in figure 3.9 the start state *sd1* is the normal state of the software, *sd2* it represent the situation where the software has already executed but has been detected by the software execution detection.

Whenever the environment has changed, it makes the detection agent move from state to another. Furthermore, when the detection agent in stat *sd3*, we can say it has sense the environment change, which mean the attack execute its instruction.

- *Sd4* The state whereby attack replication happen
- *Sd6* The state whereby mean the instruction is intrusive
- *Sd7* The state whereby mean signal has been send
- *Sd8* The state whereby mean the attack already happen or not
- *Sd9* is the last state

In this agent we have a function or class which is involving to demonstrate full regeneration of components. The functions are as follow:

- Software execution detection (SED)
- Instruction detection (ID)
- Instruction analyze (IA) (intrusive or not)
- Instruction Execution Evaluation (IEE)(yes or no)
- Malicious software propagation detection (RD)
- Signaling to other nodes.

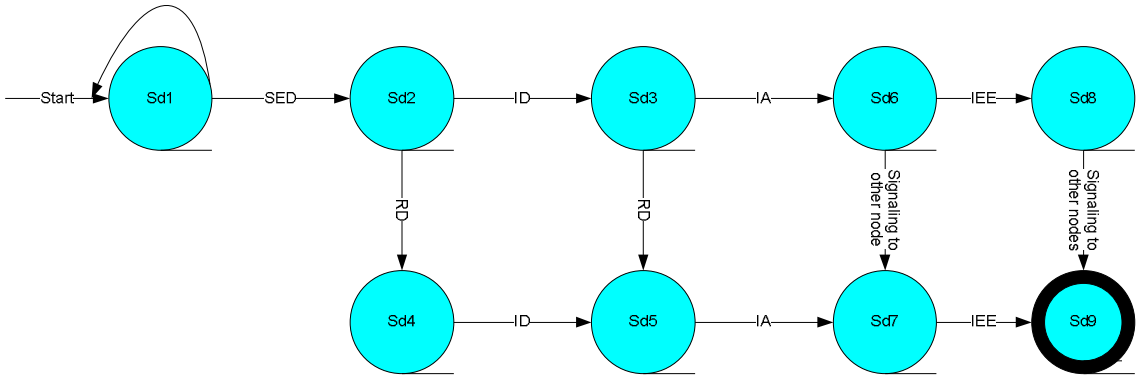


Figure 3-9: The Detection flow

3.10.2 Attacks Situation

In this section, let us consider figure 3-10 which is depict the automaton representing the action of the attack. The attack first start in situation where it is in state *sm1*, when the software initiates or executes it goes to state *sm2*. Furthermore, the software would execute its instruction or tries to replicate itself to other node in order to infect or propagate to as many as possible Nodes, the automaton continue from state *sm2* to either sm4 or *sm3* stats , and the automaton would reach to its final state which is state *sm5* .

The possible event:

1. The initialization of the attack.
2. The attack may tries to replicate.
3. The attack tries to execute its instruction.

In order to save place we introduce an abbreviation for the process of the attack, which is the names of states inputs:

- The software execution (SE).
- The software replication (R).
- The instruction execution (IE)

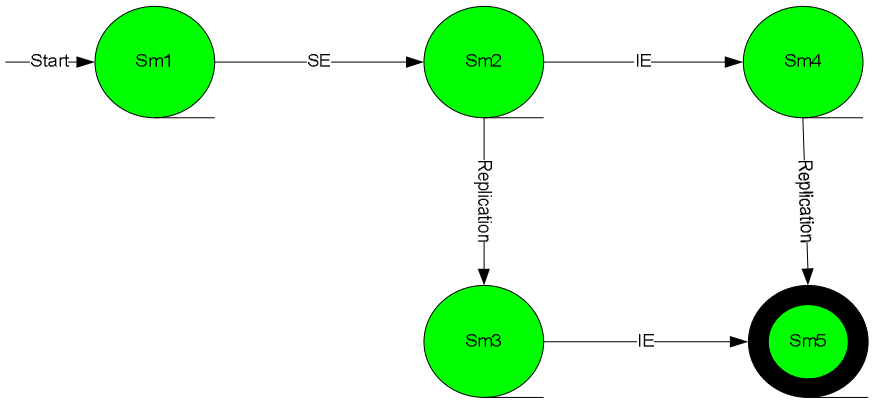


Figure 3-10: Malicious Software Behavior Path

3.10.3 Replication and Configuration and Run Time Management Agent (RC)

Every agent in our self-regenerative model has a specific roll and function, the replication and configuration agent in this part we have two roll in our agent, they are as follow:

Role 1:
Identification of code (IC)
Code copy (CC)

Role2:
Software configuration (SC)

Lets guest the detail in Fig 2.3, we see The replication and configuration agent start first when the harmful software tries to execute its instruction in order to cause a harm to the normal software, the automaton from state *sr1* would identify which code of the software or which part of the software will be destroyed, the automaton would move to stat *sr2*. The next action in the RCA automaton is to copy or replicate the particular component, from this state will continue to state *sr3*. After the code has been copied the automaton event next is the configuration and the management of the run time of the code replicated the procedure now in its final state which is sd4.

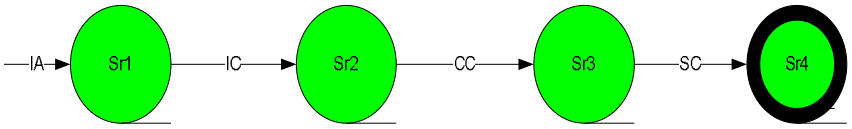


Figure 3-11: The Replication and Configuration Part

3.10.4 Prevention Agent (P)

In this part we explain the prevention of the malicious software that causes the first node from spreading or propagating from the first to other. Let's assume that the harmful software tries to propagate from the node another, if the malicious software follow the path which is the execution of its command or function call that we explain in state *sm3* in attack situation, the prevention agent will send a signal to other node to take a preventive action regarding the propagation. Furthermore, if the malicious software follows the *sm4* state in malicious software behavior, the procedure is taken and it is signaling to other nodes, this action will take the prevention agent from state *sp2* to *sp3*.

Furthermore, in figure 2.4 we have the automata move to *sp4* by performing the prevention of the malicious which is trying to propagate from first node.

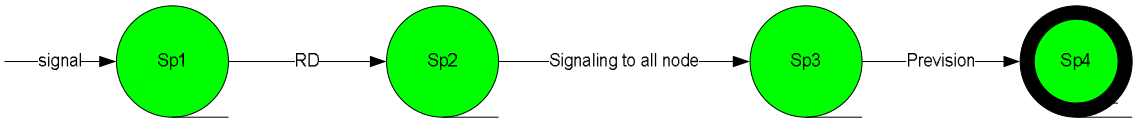


Figure 3-12: The Prevention Process

3.10.5 Run Time Management Agent (RC)

Run time management is associated with the replication and configuration agent that continuously manage the components operation. Replication and configuration agent keep replicate component if it receives signals of threat from detection agent, if replication and configuration agent change a particular component it automatically inform the replication and configuration agent to manage the component run time.

Managing component run time is to execute the component in order to deliver the service required by the other nodes from these components. For example, if a node is in state of utilizing a certain component and suddenly an attack happen to the other node

making that component in risk. The replication and configuration agent take the charge to replicate the component, after replicating the component, it then configured. After configuration, the replication and configuration agent send message to the run time management. The run time manages starts running the component for the requested node.

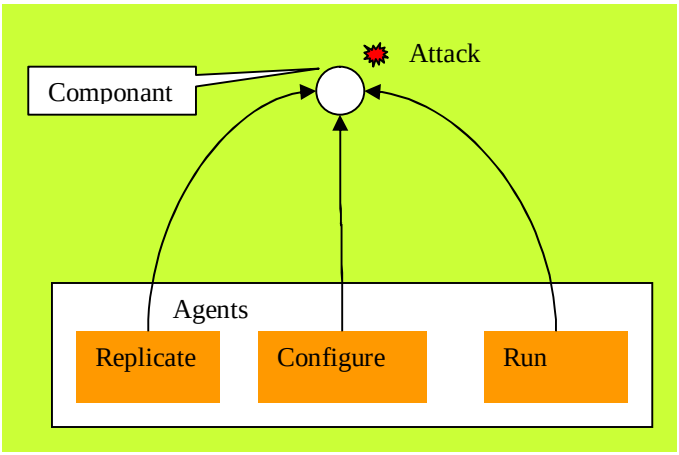


Figure 3-13: The Process of Component

3.11 Finite State Automaton Theory Implementation

A finite state machine (FSM) or finite state automaton or simply a state machine, is a model of behavior composed of a finite number of states, transitions between those states, and actions. A finite state machine is an abstract model of a machine with a primitive internal memory.

Current *state* is determined by past states of the system. As such, it can be said to record information about the past, i.e. it reflects the input changes from the system start to the present moment. A *transition* indicates a state change and is described by a condition that would need to be fulfilled to enable the transition. An *action* is a description of an activity that is to be performed at a given moment. A FSM can be represented using a

state diagram (or state transition diagram). Besides this, several state transition table types are used.

Before we start describing using the finite state machine, we present some explanation of certain abbreviation.

Alphabet	**Description**
SED	Software execution detection
IA	Instruction analyze (intrusive or not)
RD	Replication detection
IEE	Instruction execution evaluation (done or not)
S	Signaling to nodes
SE	Software execution
IE	Instruction execution
R	Replication
IC	Identification of codes (which will be harmed)
CC	Copy of code
SC	Software configuration
P	Prevention

Table 3-1: Agents Alphabet

We introduce the formal notice associated with finite automata. First we suppose our automata are:

$$F= (Sd, \sum, \delta, Sd1, F)$$

Where:

1. Sd is a finite set of states.
2. Σ is a finite set of input symbols.
3. Sd1, a member of Sd , is the start state.
4. F, a subset of Sd, is the set of final (or accepting) state.
5. δ, the transition function is a function that takes a state in Q and an input symbol in Σ as argument and returns a subset of Q.

3.11.1 Part 1: The Detection Agent

In the case of the first agent that is the detection agent, the possible way that the agent might follow in a successful round can one of this ways specified in the figure below. The detection can either detect initialization of malicious software after that detect replication or detect initialization after that execution of malicious function call, and so on.

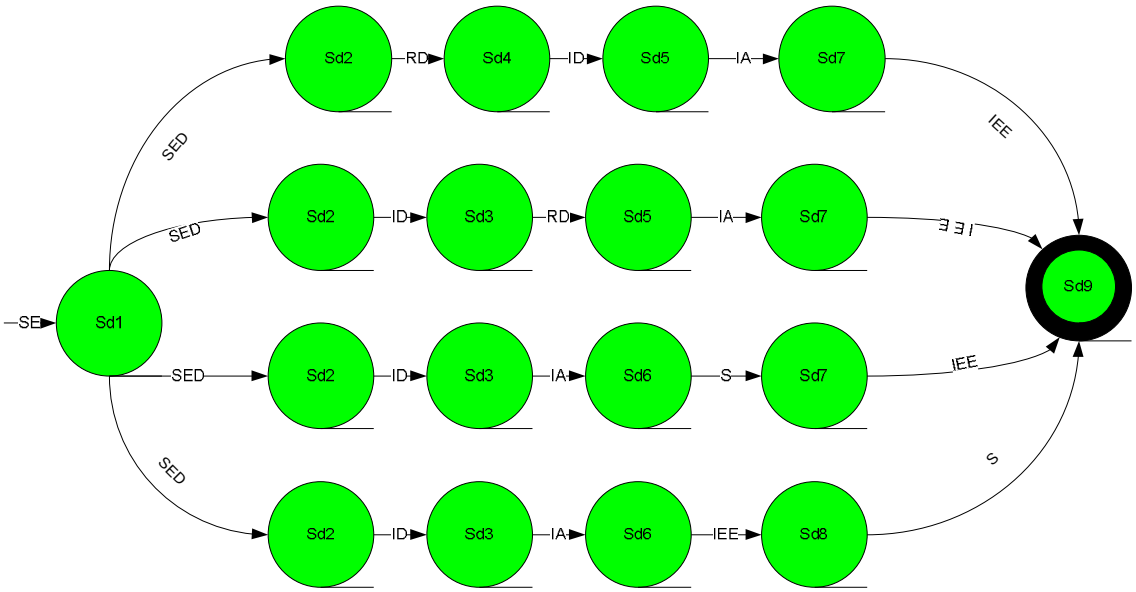


Figure 3-14: The Detection Agent Flow

We specify that a successful detection can be formally as:

$$(\{Sd_1, \ldots Sd_n\}, \Sigma, \delta, Sd_1, \{Sd_9\})$$

The agent has some finite set of state, let it be:

S= {Sd1,Sd2,…….Sdn} *9>n>0*

We now try to describe the automaton formally, in order to get some notation describing the model, now we present some meaning of states appeared in figure above.

State	**Description**
Sd1	Is the state of before the attack happen

	(normal state of the node)
Sd2	Is where the initialization of the malicious software
Sd3	The malicious initialize and tried to replicate in order to infect more nodes
Sd4	The attack is trying to carry out an instruction
Sd5	Is the state where the attack propagate and carry out an instruction
Sd6	State where it the instruction contain (delete , add,) an intrusive action
Sd7	A state which the signal to other node has been sent or the instruction has been analyzed that it is intrusive
Sd8	Where the instruction evaluation has been done , which the result is positive in a particular way of the agent
Sd9	Is final state

Table 3-2: Describes states meanings

The formal description of the detection agent is:

$(\{ Sd_1, Sd_2 \ldots Sd_n \}, \{ SED, ID, RD, IA, IEE, S \}, \delta, Sd_1, \{ Sd_9 \})$

In the automaton of the detection agent, we have four accepting string of inputs. We describe only one accepting states in this thesis. The others path would follow the same as path one description.

Path 1 :{SED,RD,ID,IA,IEE}

Path 2 :{SED,ID,RD,IA,IEE}

Path 3: {SED,ID,IA,S,IEE}

Path 4: {SED,ID,IA,IEE,S}

Path 1:

Path of possible accepting state of the detection agent {SED, RD, ID, IA, IEE,S}

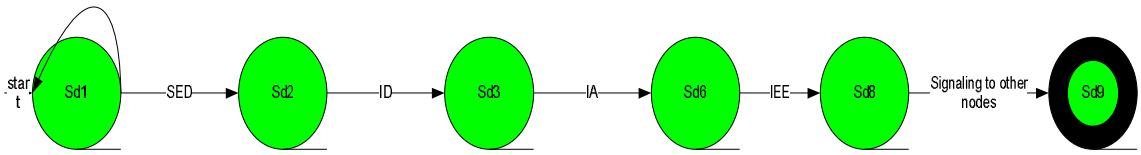


Figure 3-15: detection process

L= {ω | ω has string of (SED,ID,IA,IEE,S)}

$\boldsymbol{\delta}^*(Sd, \omega) = \delta(\boldsymbol{\delta}^*(Sd, x), a)$

$\omega = xa$

x is the string of states

a is the last and final accepting state

First we have

$\boldsymbol{\delta}^*(Sd, x)$

$\boldsymbol{\delta}^*(Sd, x) = p$

Then we formulate by

$\boldsymbol{\delta}^*(Sd, \omega) = \delta(p, a)$

We specify that

$M=(\{Sd1,Sd2,....Sdn\},\{SED,ID,RD,IA,IEE,S\}, \delta, Sd1,\{Sd9\})$ $0<n<$or equal 9}

$\boldsymbol{\delta}^*(Sd1,\varepsilon) = Sd1$

$\boldsymbol{\delta}^*(Sd1,SED) = \delta(\boldsymbol{\delta}^*(Sd1,\varepsilon)SDE) = \delta(Sd1,SED) = Sd2$

$\boldsymbol{\delta}^*(Sd1,\{SED,ID\}) = \delta(\boldsymbol{\delta}^*(Sd1,SED)ID) = \delta(Sd2,ID) = Sd3$

$\boldsymbol{\delta}^*(Sd1,\{SED,ID,IA\}) = \delta(\boldsymbol{\delta}^*(Sd1,\{SED,ID\}IA) = \delta(Sd3,IA) = Sd6$

$\boldsymbol{\delta}^*(Sd1,\{SED,ID,IA,IEE\}) = \delta(\boldsymbol{\delta}^*(Sd1,\{SED,ID,IA\}IEE) = \delta(Sd6,IEE) = Sd8$

$\boldsymbol{\delta}^*(Sd1,\{SED,ID,IA,IEE,S\}) = \delta(\boldsymbol{\delta}^*(Sd1,\{SED,ID,IA,IEE\}S) = \delta(Sd8,S) = Sd9$

The transitional table is:

	SED	ID	IA	IEE	S
*Sd1	{Sd2}	ϕ	ϕ	ϕ	ϕ
Sd2	ϕ	{Sd3}	ϕ	ϕ	ϕ
SD3	ϕ	ϕ	{Sd6}	ϕ	ϕ
Sd4	ϕ	ϕ	ϕ	ϕ	ϕ

Sd5	ϕ	ϕ	ϕ	ϕ	ϕ
Sd5	ϕ	ϕ	ϕ	ϕ	ϕ
Sd6	ϕ	ϕ	ϕ	{Sd8}	ϕ
Sd7	ϕ	ϕ	ϕ	ϕ	ϕ
Sd8	ϕ	ϕ	ϕ	ϕ	**{Sd9}**
#Sd9	ϕ	ϕ	ϕ	ϕ	ϕ

Table 3-3: The Transitional Table of the Process

We can say that a system reached its final state :

$$L(M)=\{x \in \Sigma^{*} | \boldsymbol{\delta}^{*} (Sd, x) \cap F \neq \phi \}$$

The property of the part1, I mean the accepting string is that all states can be reached from start state to final accepting state.

We now see the state transitions probability; Furthermore, the system is only in one state at each time step.

Our detection agent automaton has a certain finite probability of its accepting string of state. When the malicious software initialize, it either tries to harm or tries to replicate for more infection, thus the detection detect the tow action When the automaton follow the way of executing its instruction first , there is one chance of tow that it instruction Analyze and one chance out of tow that it detect replication. For every agent execution in the environment, there exist sequences of actions.

Part 2: Replication and Configuration and Run Time Management Agent (RC)

In this part we have functions of our agent, the function are as follow:

- Identification of code (IC)
- Code copy (CC)
- Runtime manager
- Software configuration (SC)

Lets guest the detail in figure 3-14, we see The replication and configuration agent start first when the harmful software tries to execute its instruction in order to cause a harm to the normal software, the automaton from state a would identify which code of the software or which part of the software will be destroyed, the automaton would move to stat b. The next action in the RCA automaton is to copy of replicate the particular code , from this state will continue to state c. after the code has been copied the automaton event next is the configuration of the code replicated , the will now in the its final state which is d .

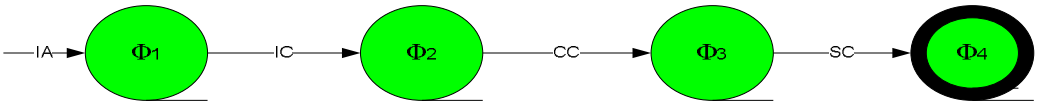


Figure 3-16: The Replication and Configuration Part

The following table explains the state role and sign which is used in the definition.

Sign	Description
Φ1	The state where the agent receive the input of (IA) which is attack instruction evaluation
Φ2	The state of the potential software been identified
Φ3	This state in after identification of code that is going to infected or harmed
Φ4	The state where the code of the software been configure to ensure scalable , and run time management

Table 4: The States descriptions

Let say:

$L = \{\omega \mid \omega \text{ has string of } (IA,IC,CC,SC)\}$

And

$M=(\{ \Phi 1, \Phi 2, \ldots \Phi n\},\{,IAIC,CC,SC\}, \delta, \Phi 1,\{ \Phi 4\})$

We have a transition from state Φ1 to the finale and accepting state Φ4 according to the string of input which is (IA,IC,CC,SC), the calculation is:

$\delta (\Phi 1,IC) = \Phi 2$

And the extended transition function of the accepting string is:

$\boldsymbol{\delta}^{*}(\Phi 1,\{ IA,IC,CC,SC\}) = \delta (\boldsymbol{\delta}^{*}(\Phi 1,\{ IA,IC,CC \}SC) = \delta(\Phi 3,SC) = \Phi 4$

We can now say that the system complete the self-regeneration if it fulfill the state transition to the end. I the detection agent the system should start from the fist state the finish to at the finale state which is instruction execution evaluation. In case of replication and run time management it start from the identification of code and end at

3.12 Agent Interaction Description

Agent in our model are interacting and producing action which will result the self-regenerative action. These agent are taking sensory detection from the environment which they have been deployed in, according to the figure below we have the detection agent detect the change in the component of the system, or detect malicious activities which will harm the component .following that we have replication and configuration and

run time management agent comes timely to configure the system. We have in addition the prevention agent which will prevent malicious from harming the system and from spreading throughout the network.

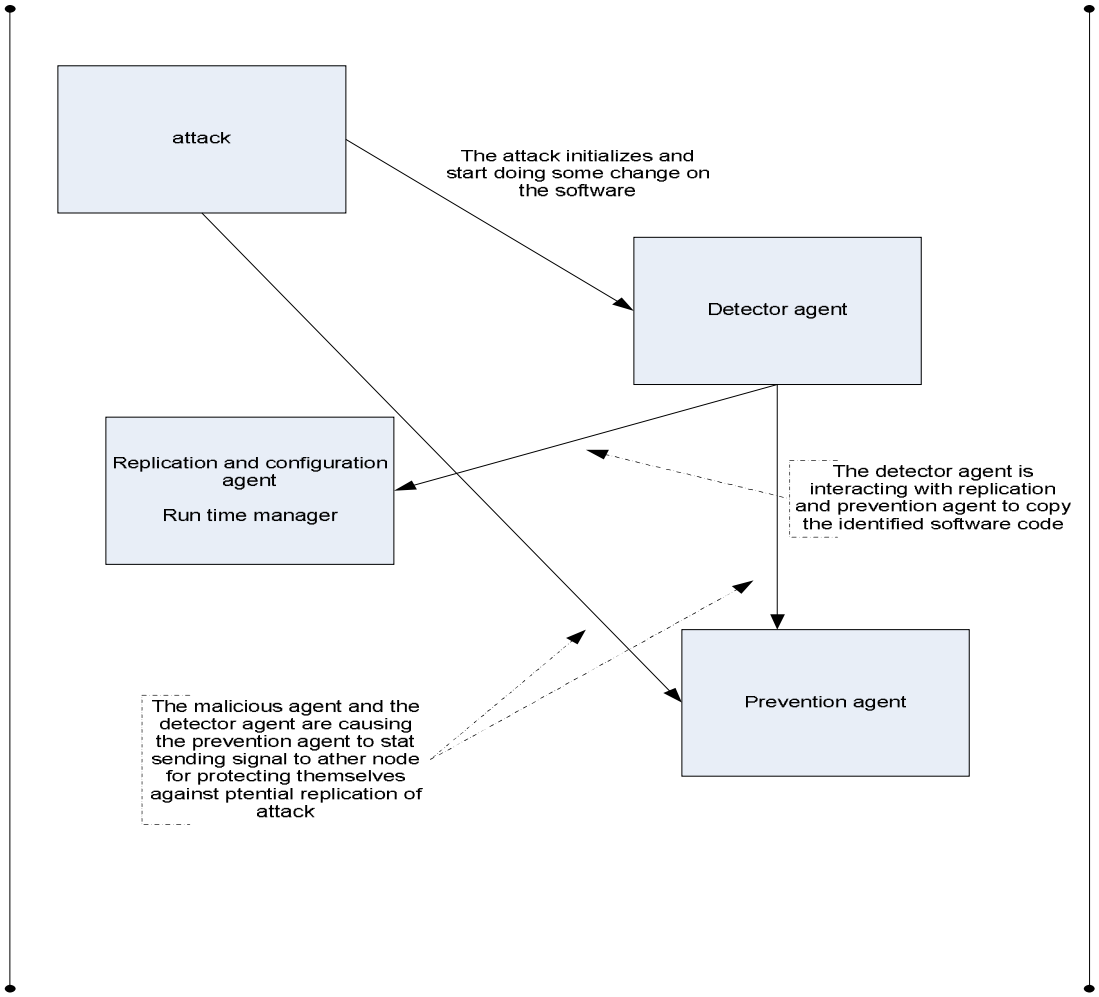


Figure 3-17: Flow of Agents Messaging

3.13 Agent Specification

In general, when specifying a system, we are interested not only in a description of the system but also in ensuring that the system fulfils certain requirement.

When Agent behaves in its environment, it takes sensory input from the environment and produces as output actions that affect it. The interaction is usually an ongoing, non termination on.

Normally an agent will have a repertoire of action available to it; this set of possible action represents the agent capability. The key problem facing agent is that of deciding which of its action it should perform in order to best satisfy its design objectives.

One possible solution is to have the agent explicitly reason about and predicts behaviors of the system.

And the aspect of the interaction between agent and environment is the concept of real time put at its most abstract, a real time interaction is simply one in which time play a part in the evaluation of an agent performance.

The specification of function of the agents formally is described below; we divide into each agent, such as Module 1 is for the detection agent and module 2 for the replication and configuration agent and so on.

- **Module 1: The Detection Agent**

In our model we specify that we would like to build a proactive action in order to detect the malicious activities and also change in the component and reactive in order to create a self-regenerative system. In the proactive part we have the detection agent that would detect the malicious.

We can easily formalize the abstract view of our agent which it has been described so far; let us assume that the detection agent environment may be in any of a finite se H of discrete, instantaneous state.

$$H = \{h, h', \ldots\}.$$

Detection gent (DA) assumed to have a repertoire of possible actions available to them which transform the state of the environment. Let

$$A = \{\alpha, \alpha', \ldots\}.$$

Be the finite set the agent action.

The basic model of this of this agent interaction with their environment can respond with a number of possible states. However, only one state will actually result, through of course, the agent does not know in advance which it will be on the basis of this second state, the agent again choose an action to perform . The environment responds with one of a set of possible states, the agent then chooses another action, and so on.

In detection agent we have the possible environment is finite

Let say:

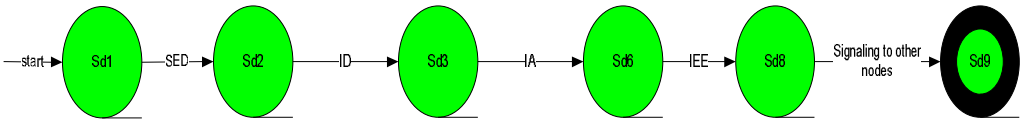


Figure 3-18: Detection Agent Possible Actions

The environment is

$$H = \{h_0, h_1, h_{2,}, h_3, h_4, h_5\}$$

A run, *r,* of the detection agent in an environment is thus a sequence of interleaved environment states and actions, in the first h0 malicious try to execute its command or instruction, in the same time we have the detection agent sense the environment change, thus accordingly the agent will perform an action, and we have:

$$r : h_0 \xrightarrow{\alpha_0} h_1 \xrightarrow{\alpha_1} h_2 .... \xrightarrow{\alpha_{u-1}} h_u$$

The above notation maps the run of the detection agent. Let

- R be the set of all such possible finite sequences (over H and A)
- RA be the subset of these that end with an action
- RH be the subset of these that end with an environment

In order to represent that an effect that the detection agent's action have on an environment, we introduce a state transformer function:

$$\tau : R^A \rightarrow \vartheta(H)$$

Thus a state transformer function maps a run (assumed to end with the action of the detection agent) to a set of possible environment states – those that could result from performing the action.

If $\tau(r) = \phi$ (where r is assumed to end with an action of the detection agent), then there are no possible successor state to the run of the detection agent. In this case , we say that the system has ended its run , we will therefore consider that all runs of the detection agent is terminated .

As for modeling the Detection agent in the system, we say an environment Hv is a triple $Hv =< H, h_0, \tau >$ where

H : is a set of environment states.

$h_0 \in H$: is an initial state.

τ : is a state transformer function.

We now need to introduce a model of the detection agent that inhabit our system,

Let ζ be the symbol of the detection agent, we now say that:

$$\zeta : R^H \rightarrow A$$

Thus an agent make a decision about what action to perform based on the history of the system that it has witnessed to date.

Notice that while environments are implicitly non-deterministic, the agents assumed to be deterministic.

Let AG be the set of all agents.

We say a system is a pair containing an agent and environment , any system will have associated with it a set o of possible *run* ; we denote the set of runs of agents ζ in environment Hv by $R < \zeta, Env >$ contain only terminated *run* , e.i *run r* such that *r* has no possible successor state : $\tau(r) = \phi$

Formally, a sequence

$$(h_{1,} \alpha_1, h_2, h_3, ...)$$

Represent a *run* of an agent ζ in an environment $Hv =< H, h_1, \tau >$ if

- h_1 is the initial state of *Hv*.

- $\alpha_1 = \zeta$ (h_1); and
- U>0.

$$h_u \in \tau((h_1, \alpha_2, ...\alpha_{u-1}))$$

- **Detection agent algorithm:**

Algorithm below

If environment is change $h_1 \in H$ is true
Then Detection agent is detecting Б
Identify function call
If call=malicious function call
Prevent h_1
Else
Allow h_1
Else if environment is change H=h_2
Then detect h_2
Analyze instruction
If h_2='damage'
Prevent h2
Else h_2!='damage'
Ignore
Else if H=h_3
Then signaling to other nodes ŋ
Else ignore

- **Module 2: The Replication and Configuration Agent** ζ_c

The replication and configuration agent is considered as a tow part:

- the action of detection in performing i from the detection agent and

According to the б Instruction Analyze in detection agent, RC agent would indentify which of part of component is going to damaged, if the code identified ς is the identification of the component.

We will then have the replication of the component ϖ .

If the ϖ is done then the replication is succeeded

The RC agent is in the state of configuration of software, if the component has been copied or if the component is available somewhere else. The component is configured Then the agent will put the configured component in runtime to scale the system.

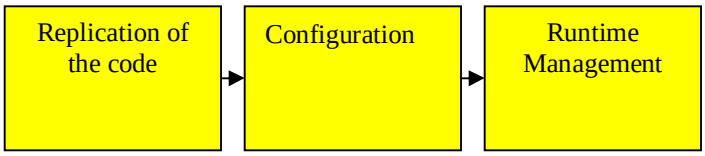


Figure 3-19: The Agents Involved in the Process

If The RC Agent succeed in its mission then we can say that the system is achieving its design objectives, therefore, the system is bio-inspired integrated, according to the model of the internal cell operation that we have mentioned in the beginning of the chapter, we will adjust our survivable model if it is encountered any malfunction or failure .

RC agent algorithm:

```
If  IA = attack  then
    Identify component
      If IEE = not done then
           Copy component
           Else locate the component
      If old component = not new then
          Configure component
         Configure Runtime
      Else ignore
Else ignore
End
```

- **Module 3: Prevention Agent**

This model is the prevention agent, it is also sending signal to other nodes in the network.

In this agent we have a set of environment change, which going to be the interaction between the agent in some part between tow agent in one node , and the other between tow agent on different nodes.

The interactions from:

S : signal from detection agent ζ_p

μ : replication detection by detection agent ζ_p

ŋ :signal from other nodes

The actions available to this agent is :

Ɗ : signal to other nodes

ρ_r : Replication prevention

ρ_e : Execution prevention

Let say if the agent receive signal from the detection agent, it will automatically send a signals to other node and perform the prevention execution ρ_e .

Again if the prevention agent see the detection agent ζ_d detect an attack replication it will then perform the signal to other node ŋ and the replication prevention ρ_r .

If the prevention agent ρ_e receive a signal from other node' prevention agent S it will prevent the attack replication from other nodes.

3.14 CASES CONSIDERATION & SCENARIOS

We introduce the phases of the system which it is going respond to environment change to prevent the damage and improve services:

3.14.1 Scenario No 1: Attack with Propagation

If attack has happened in one node; there is a two hypothesis:

- It may cause a damage to this node software component , and
- Then transfer itself to another node in the network.

At this point we have an attack to one node, in this part the system would have to perform a proactive regeneration of the component and the functions of the software to prevent interruption. Beside, the system again perform a reactive action for other node in the network, because, the attack might propagate itself to those node and cause different damage.

Regeneration of the actual node is performed by the actions of the agent which is RC agent , this agent react if the is any attack detected by the detection agent the two agent are interacting in order to perform the component regeneration

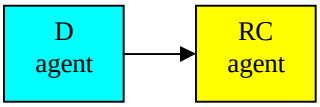


Figure 3-20: Detection Agent Interact with the RC Agent

The part of the attack propagation is also a crucial in this case; the system now has to send some signal to the other node in order to protect themselves from this thread. If the detection agent detects the attack in the first node, it will automatically send the signal through prevention agent (P Agent).

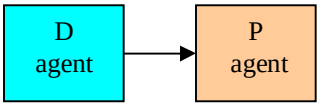


Figure 3-21: Detection Agent and the Prevention Agent

Figure below illustrate how the first Node detects the malisons, and then if the attack try to propagate, the Node will send a signaling to other node in order to take the proactive action.

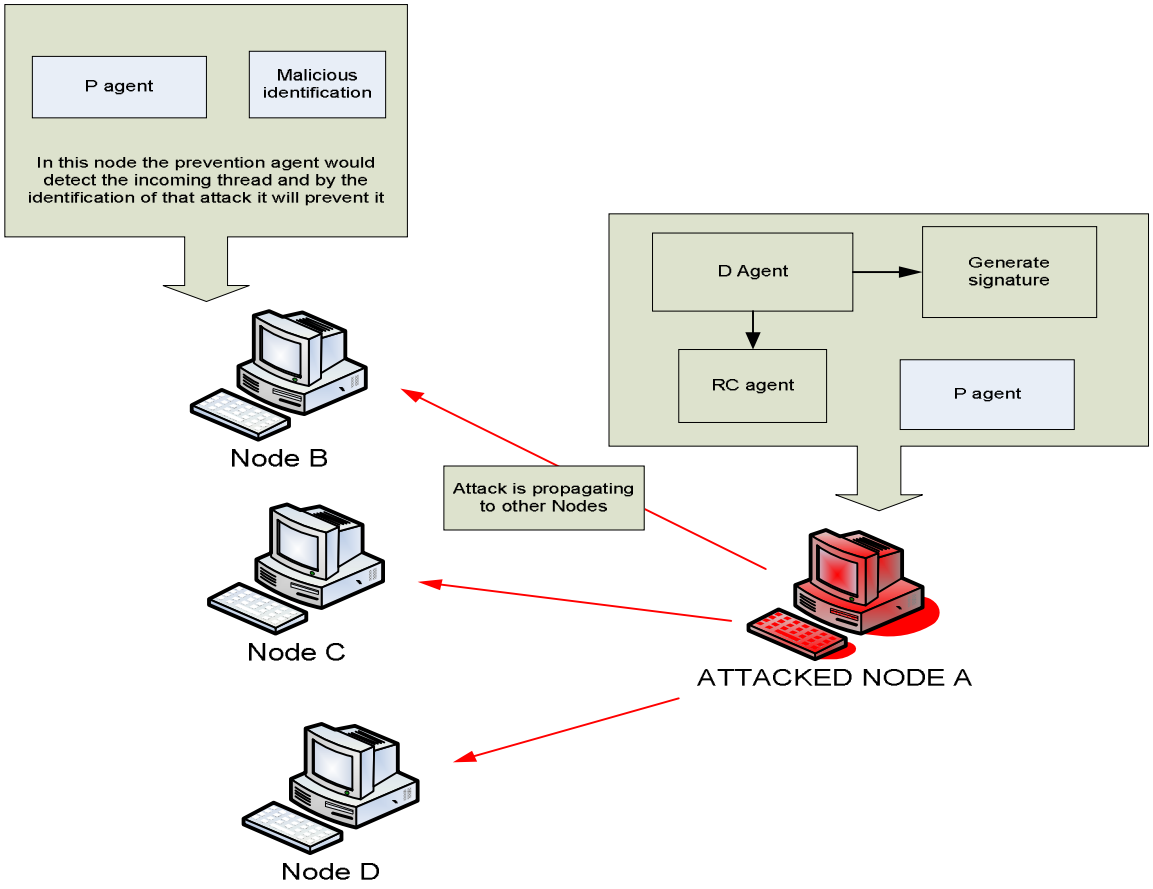


Figure 3-22: Attacks and its Propagation

The following figure depict the effect of the messaging among the the agent over time, in this scenario which the attack is happen sequentially in one node to another in the system, as we can see the detection agent first detect the software execution

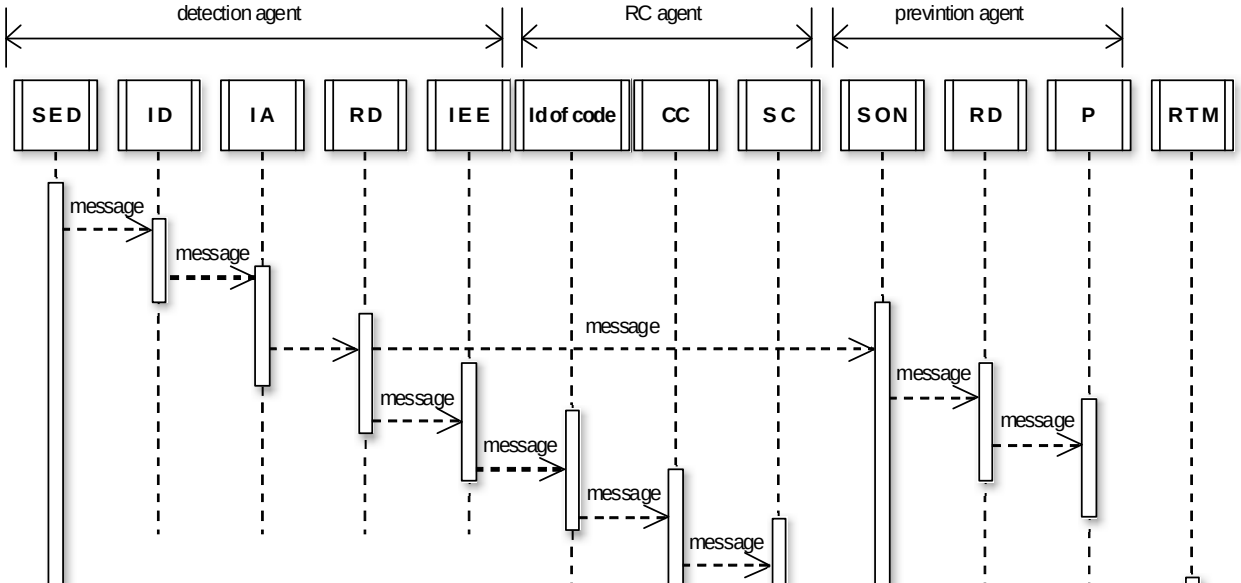


Figure 3-23: Attack and Propagation Sequence Diagram

Scenario above illustrates how the model interacts and responds if attack happen and there is an attack preparation going on.

3.14.2 Scenario N 2: Only one Attack

If attack happen to one only node and then not propagate to other node, in this case we have only one hypothesis:

Damage may happen to this node software component.

This probability is when attack initializes or start executing, which in result will cause damage to the component of the software, or cause the system to loose runtime software. In the detection agent of this node with the software execution detection (SED), and

again through the instruction detection (ID) and instruction analyze (IA). The result of this procedure is the system is under attack. The detection has two parts; it may detect the malicious execution before performing its mission, or after the damage has been done.
If it is before the damage the action will be:

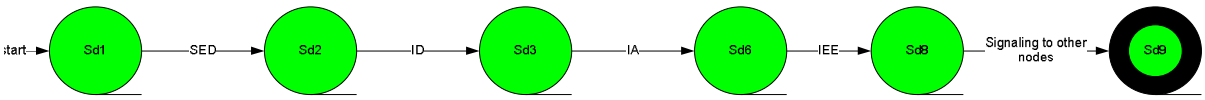


Figure 3-24: The detection agent

And the component regeneration is:

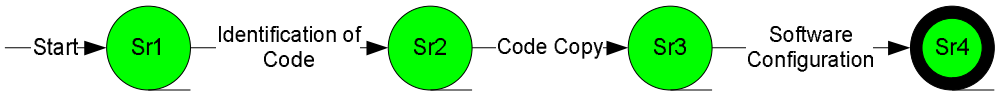


Figure 3-25: The RC Agent

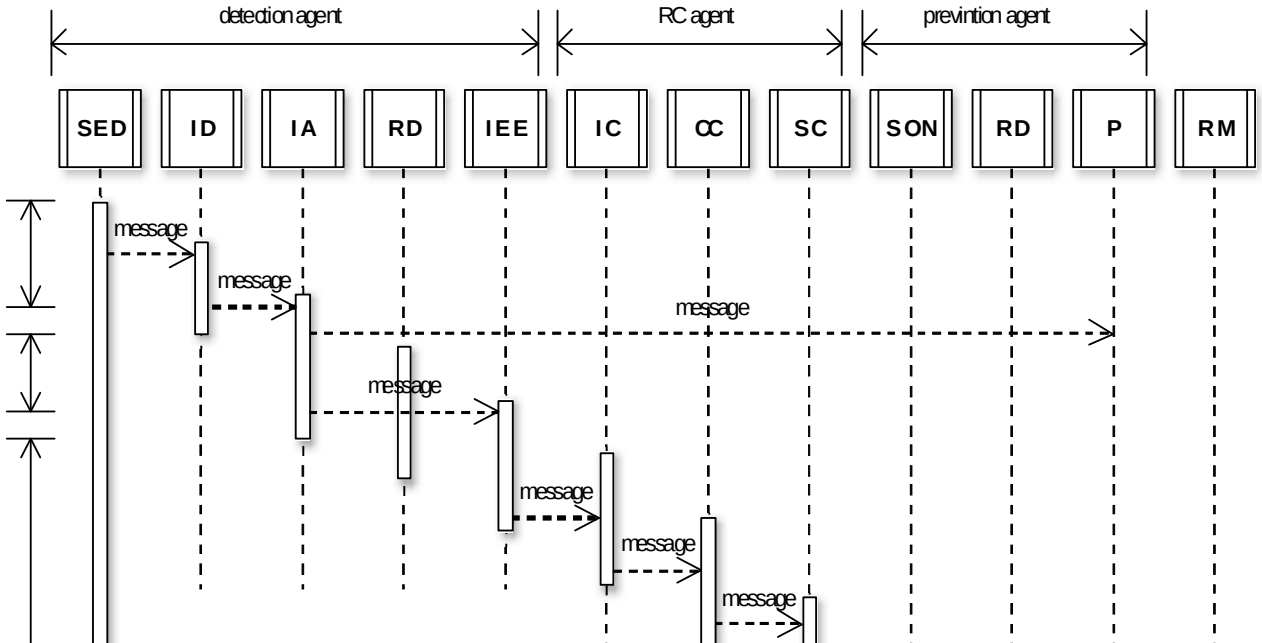


Figure 3-26: Only one attacks Sequence Diagram

The above sequence diagram shows how the interactions between agents during attack, the specific scenario is in case of only one attack, we can see the detection monitor the functions call and identify whether its a pattern of a malicious or not. The monitor interacts with other agent in order to finish the regeneration of the components as depicted in figure 3-22.

3.14.3 Scenario No 3: Concurrent Attack

If we have multiple attacks in same time, we consider the network is under total attack, the hypothesis of this case is:

- Attack happen to all nodes in the network, and potentially the component of this node is damaged.
- Attack happened to most of the node.

In this case the attack initialize in all the node in the network , as we have said in the previews case , if the attack try to execute its instruction it is been detected by the detection agent (D agent). In the detection agent we have the software instruction detection (SED) detect the initialization of the attack by attack instruction.

After the detection of the attack, the replication and regeneration agent (RC agent) will identify the software component, which is after the attack been performed its damage. Then (RC agent) would configure the component of the software

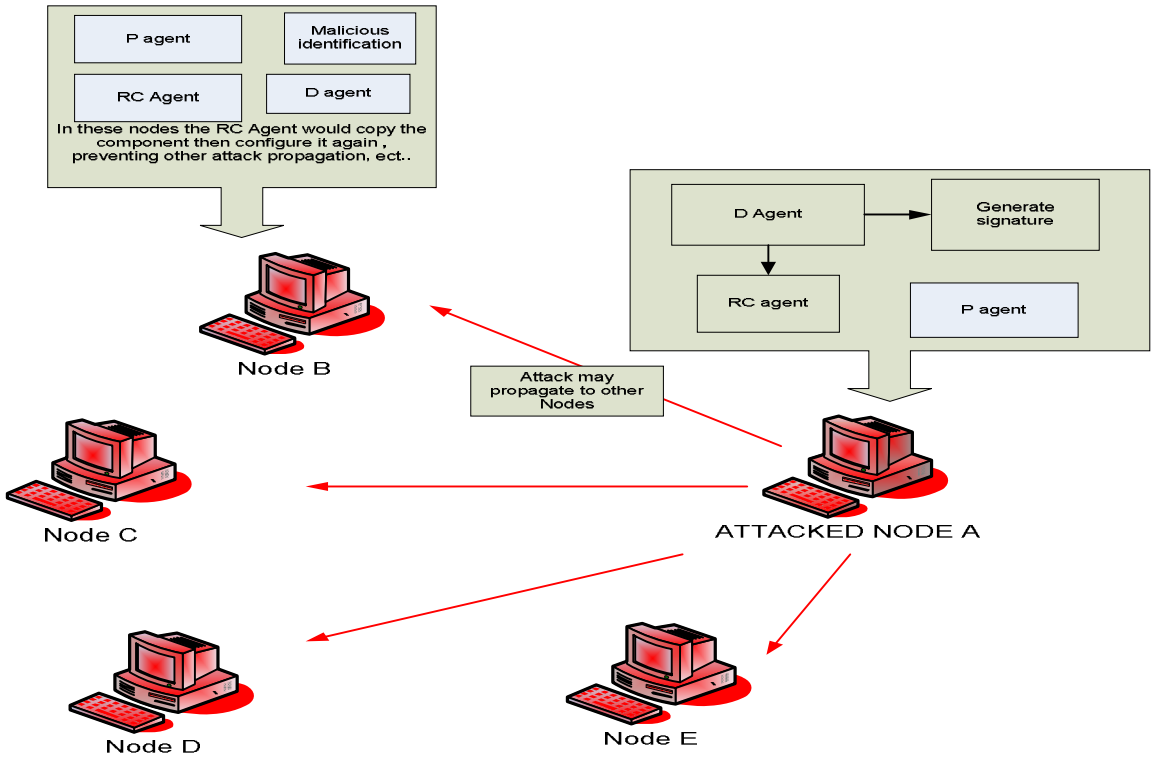


Figure 3-27: The Behavior of the Nodes Agent in Case of Concurrent Attack

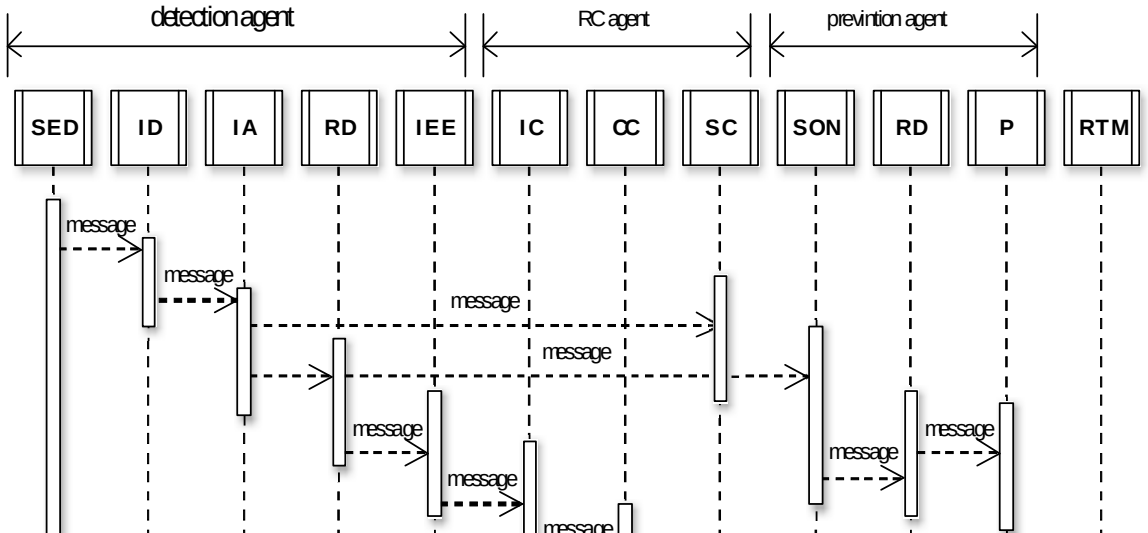


Figure 3-28: Concurrent Attacks Sequence Diagram

Figure 3-24 sequence diagram illustrate how the communications between agents during attack, the particular scenario in this case is concurrent attacks, we can see the detection monitor the functions call and identify whether it's a pattern of a malicious or not, alongside the replication also considered. The monitor interacts with other agent in order to finish the regeneration of the components as explained in figure 3-24.

3.14.4 Scenario No 4: Sequential Attack

If attack happened sequentially to the nodes, the hypothesis in this case is attack happen in one node then another attack to another node and so on.

The attack in this case is happening in a sequential mode, which is in the first the malicious attack one node then to the second.

In the first attack to the node, we have the detection agent which will detect the initialization of the attack by its instruction, then generate the malicious signature and distribute to the nodes in the network, in this moment the same attack would not be able to cause any thread to other node.

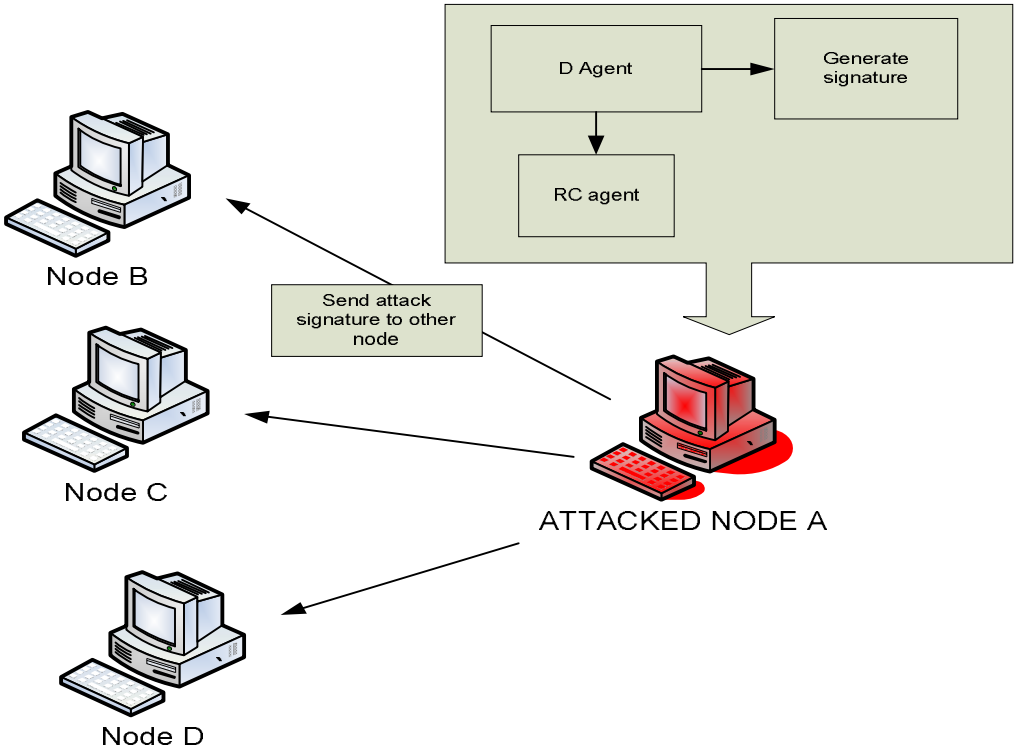


Figure 3-29: The Sequential attack

If the malicious attack sequentially, this may be node B or node C. The node would perform the same proactive action in order to come out of this attack, beside it send the attack signature to other node.

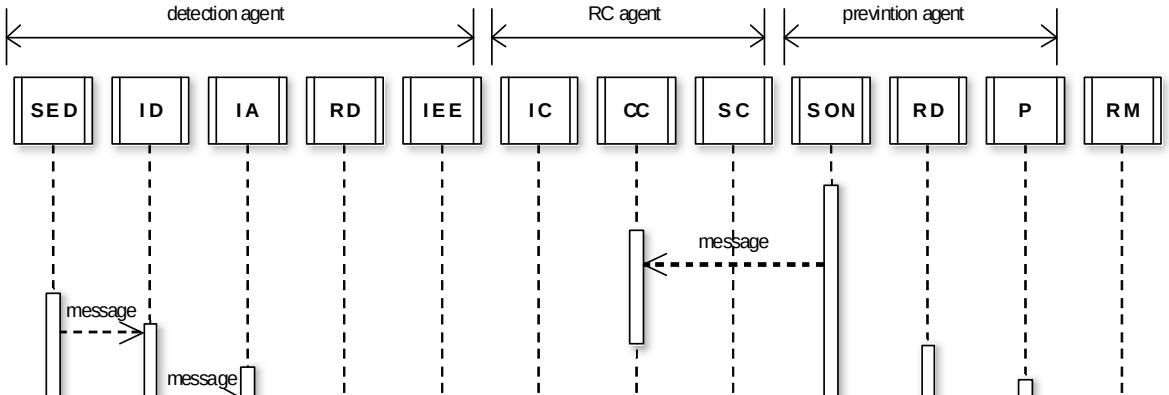


Figure 3-30: Sequential Attacks Sequence Diagram