Journal of the Nigerian Association of Mathematical Physics
Volume26, (*March, 2014*), pp 467 – 475

Formal Characterization of a Mobile Agent Operational Environment

Imianvan Anthony Agboizebeta[1] *and Akinyokun Oluwole Charles*[2]

[1]Department of Computer Science,
University of Benin, Benin City. Nigeria.
[2]Department of Computer Science
Federal University of Technology, Akure. Nigeria

Abstract

The resources of computer networks are vast. For the purpose of developing a mobile agent for computer networks, proper specification of the computer network resources cannot be undermined. The aim of this paper is to produce a framework for the assessment and evaluation of computer network resources using mobile agent technology with formal specification of the network resources. The detail specification of the resources of the network is provided using Formal Method (Zed notations). The proposed system provides takeoff point for mobile agent developers.

Keywords: Mobile Agent, Formal Specification, Zed notations, Computer Network, Schema

1.0 Introduction

A computer network is a group of computers connected together and separated by physical distance. Over the ages, searching for resources in a network often involves physical movement of the network administrator from one machine to another. The approach was not only stressful but introduces some delays in monitoring events on the network. Besides, events on the network were not monitored as they arise and network administrators were bored with the issue of which computer to monitor next. There was therefore a need to have intelligent software that would autonomously search for and interact with network resources on the network administrators' behalf. A mobile agent is such intelligent software **[1]**.

Mobile agents are autonomous and intelligent programs that are capable of moving through a network, searching for and interacting with the resources on behalf of the network administrator. Mobile agent is an executive program that can migrate at times of its own choosing from one machine to another in a network **[1, 2]**. Mobile agent technology has been applied to electronic commerce transactions **[3]**, distributed information retrieval **[4]**, and network management **[5]**.

Network management is a means to effectively deploy and coordinate network resources. That is, it helps to plan, administer, analyze, evaluate, design and expand communication networks to meet demands at all times, at reasonable cost and optimum capacity. Effective network management will require monitoring and controlling the resources of the network. The assessment and evaluation of network resources are thus crucial part of network management **[1]**.

Mobile agent could be activated and launched from one computer to another for the purpose of autonomously searching for and interacting with network resources on the network administrator's behalf. Conscious efforts at developing a mobile agent for the assessment and evaluation of computer networks include **[1,2]**. This research attempts to identify the resources of computer networks and then specifies those resources using formal methods (Zed notations) in order to enhance the development of expert system using mobile agent technology for the assessment of those resources.

2.0 Formal Specification

A specification written and approved in accordance with established (mathematical) notations is a formal specification. Z ('zed'), for instance is a formal notation based on set algebra and predicate calculus for the specification of computing systems. Z specification of systems employs the power of discrete mathematics. Formal methods are becoming more accepted in both academia and industry as one possible way in which to help improve the quality of both software and systems. Note however that formal method is not a panacea, but one more weapon in the armoury against making design mistakes. The Z notation is useful to organize and communicate thoughts within a design team **[6]**.

Corresponding author: ***Imianvan Anthony Agboizebeta,*** E-mail: tonyvanni@yahoo.com, Tel.: +2347069742552

Since a formal specification is precise, if such a specification is wrong, it is easier to tell where it is wrong and correct it. Using a formal notation increases the understanding of the operation of a system especially early in a design. It helps to organize the thoughts of a designer, making clearer, simpler designs possible. Formal specification provides a check that the system will behave as expected by the designer. The use of formal methods can help to explore design choices. Such methods aid the design team in reasoning about the operations of the system in clear terms before and during its implementation **[7]**.

Z uses various conventions to identify particular types of schema and state variables used in operation specification:

a. If any variable name, N, is followed by ′ , for example N′, this means that it represents the value of the state variable N after the operation. In Z terminology, N is *decorated with* a dash.
b. If a schema name is decorated with ′, this introduces the dashed values of all names defined in the specification together with the invariant applying to these values.
c. If a variable name is decorated with ! , this means that it is an output, for example, 'message!'.
d. If a variable name is decorated with ? , this means that it is an input, for example 'amount?'.
e. If a schema name is prefixed with Xi (Ξ), this means that dashed versions of the variable defined in the named schema are introduced. For all variable names introduced in the schema, the values of corresponding dashed names are the same. That is, the values of state variables are not changed by the operation.
f. If a schema name is prefixed with the Greek character Delta (Δ), this implies that the values of one or more state variables will be changed by the operation where that schema is introduced. For all variable names introduced in such named schema, corresponding dashed names are also introduced and may be referenced in operations.

3.0 Design of a Mobile Agent

The mobile agent environment is presented in Figure 1. The environment is characterized by server machine, which connects to a number of workstations. Therefore, there are two categories of environment. First, is the server machine environment, which is composed of some hardware devices such as main and secondary memory, high duty printer, specialized printers, scanners, switches, modems, and network ports. There are also some categories of software such as operating system, front-end software, back-end software and utility software. Second, is the workstation environment which, is comprised of some hardware devices and software systems of, perhaps, lesser capacity than that of the server machine environment.

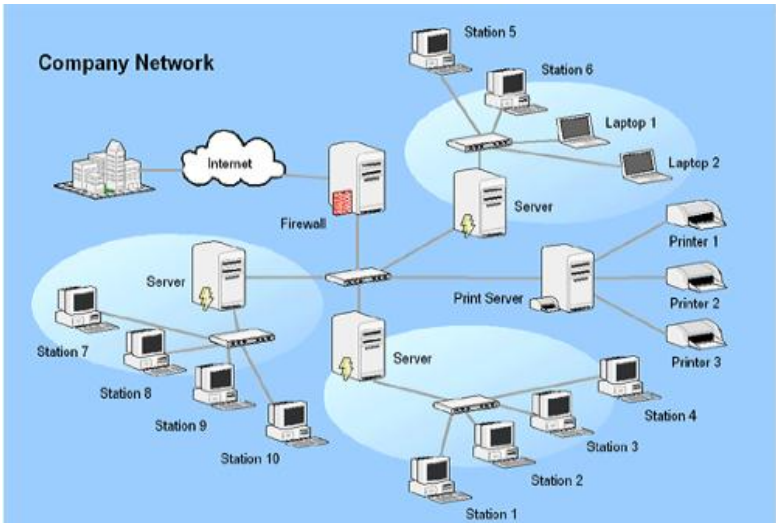

Figure 1: Mobile Agent Environment.

A mobile agent is multi dimensional and can be launched from any workstation or server machine to survey the state of the resources in any other workstation or server in the computer network environment. Thus, there is a source and one or more targets of a mobile agent.

The architecture of the mobile agent is conceptualized in Figure 2. The architecture is composed of back-end and front-end engine. The back-end engine is made up of the server machine and workstations which are considered to be static. The front-end is the software-based interface which constitutes the mobile agent and is dynamic and mobile in nature. In this research, the following mobile sub-agents are considered.

a. Bandwidth evaluator.
b. Memory evaluator.
c. Response time evaluator.
d. Files evaluator.
e. Input-Output device evaluator.

Each of the resources of a computer network environment has unique features; hence the evaluation of each resource can be developed by a unique model.

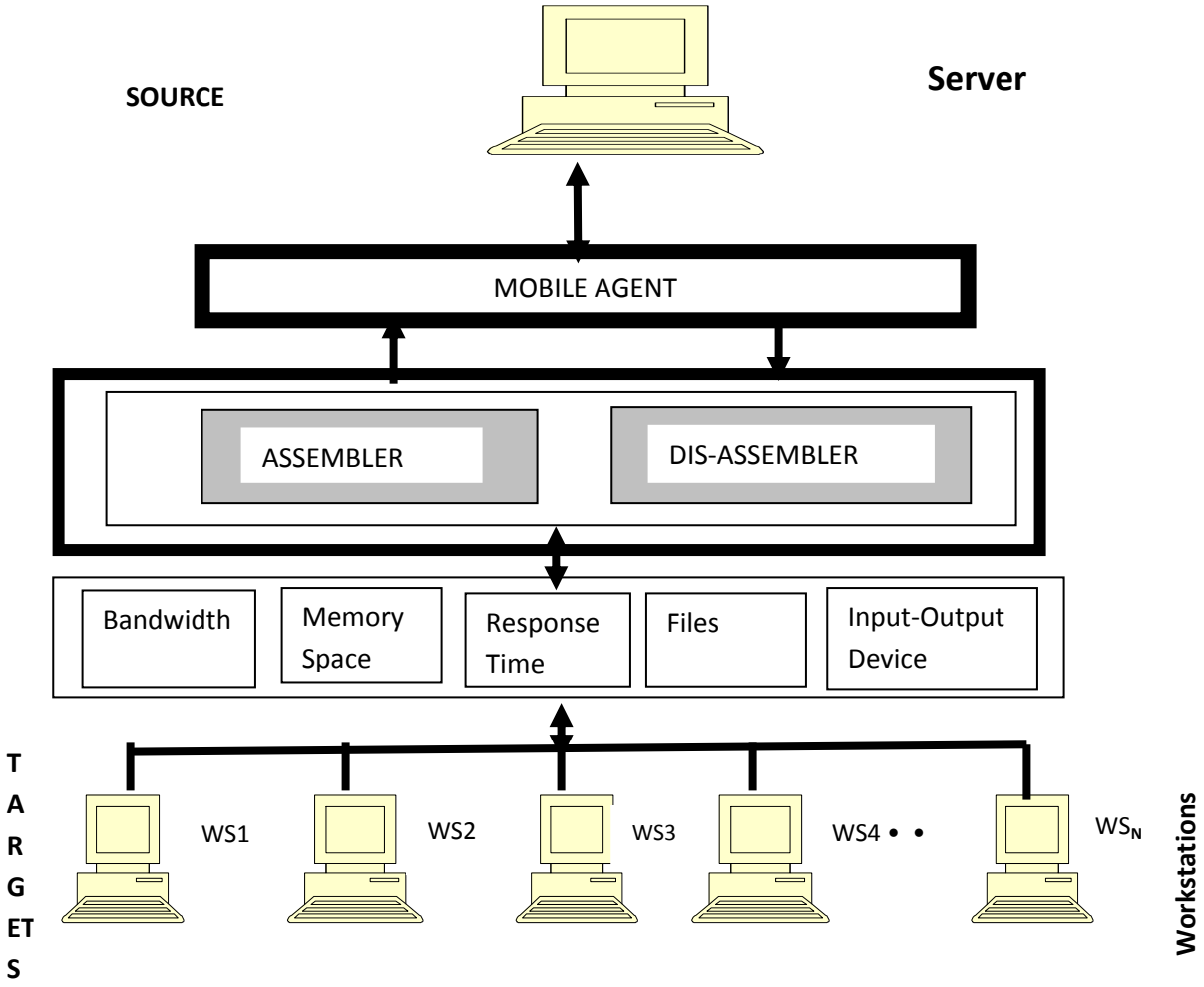

Figure 2: Architecture of the Mobile Agent.

The detail design of the mobile agent with respect to each computer network resource is presented hereafter. It is noted that there is a source of take off of a mobile agent which is considered to be the server machine and two or more target workstation. The platform for the takeoff of the mobile agent at the source and the platform for its landing at the target computer are the operating system. For clarity, the architecture in simplified form is presented in Figure 3. At the source, the mobile agent is decomposed into its constituent parts. Each subagent will interacts with the host operating system of the target and its appendages or utility programs for network monitor and cyber clock for the purpose of assessing and evaluating the resources that are available. The results obtained by all the subagents after a successful visit to a set of target workstations are assembled for the purpose of reporting them for external analysis, interpretation, policy formulation and decision making.

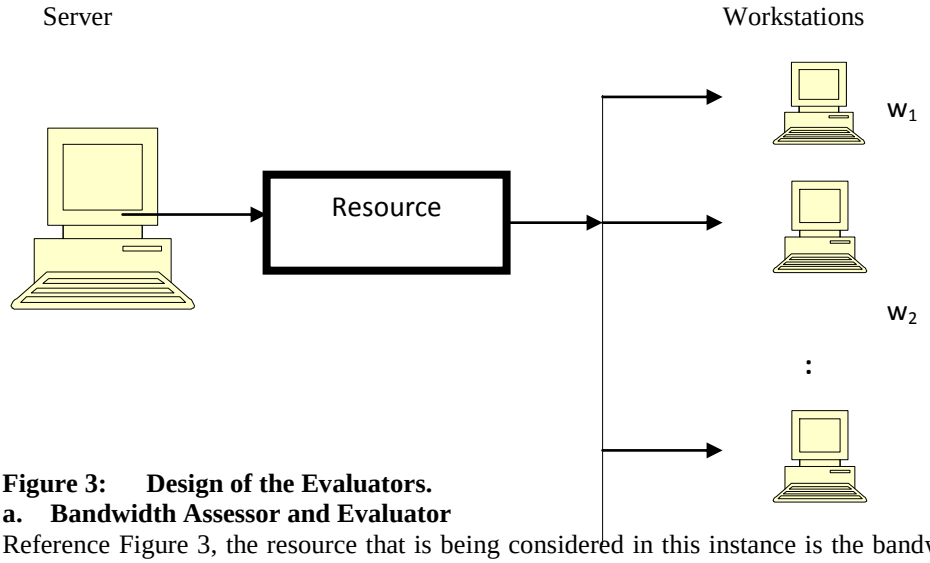

Figure 3: Design of the Evaluators.

a. Bandwidth Assessor and Evaluator

Reference Figure 3, the resource that is being considered in this instance is the bandwidth. The design has three major components, namely: input, bandwidth evaluator and output report. The input to the bandwidth evaluator is the target machine identity and the number of workstations. The platform for its take off is the host operating system of the server machine, while the platform for assessment and evaluation at the target is the operating system of the target workstation. The medium of transmission between the source and target machine is a cable. The evaluator gets into the files of the target operating system to collect relevant information desirable for estimating the bandwidth used in the target workstation for a given transmission time. The bandwidth evaluator generates report on the packets transmitted, size of each packet, transmission time, used bandwidth within the transmission time and percentage of the used bandwidth. The used bandwidth in the computer network environment denoted by 'B' is evaluated by:

$$B = \sum_{i=1}^{n} \sum_{j=1}^{m} b_{i,j} / t_j \quad - \quad (1)$$

where $b_{i,j}$ represents the bandwidth used in transmitting jth packet in ith workstation and t_j is the transmission time of jth packet. The percentage of the used bandwidth (B_u) to the bandwidth subscribed (B_s) to by the computer network environment denoted by 'P' is evaluated as:

$$P = 100B_u/B_s \quad - \quad (2)$$

The formal specification of the logic of bandwidth evaluation using Z Schema notation is presented in Figure 4 as follows:

BandwidthEvaluator

NumberofPackets?, NumberofWorkstations? : $\mathbb{N}$
PacketSize, TargetPacketSize? : $\mathbb{Z}$
TransmissionTime, TargetTime? : $\mathbb{Z}$
BandwidthUsed, BandwidthUsed′ : $\mathbb{Z}$
i, j, m, n : $\mathbb{N}$
BandwidthSubcribed, BandwidthPercentage : $\mathbb{Z}$

BandwidthUsed ← 0
n ← NumberofWorkstations?
m ← NumberofPackets?
$\forall i,\ 1 \le i \le n$ /* loop over workstations or targets */ {
$\forall j,\ 1 \le j \le m$ /* loop over packets transmitted in ith workstation*/ {
PacketSize ←TargetPacketSize? ; TransmissionTime ← TargetTime?
BandwidthUsed′ ← BandwidthUsed + (PacketSize / TransmissionTime)
DisplayData (PacketSize, TransmissionTime, BandwidthUsed) } }
BandwidthPercentage ← (BandwidthUsed′ / BandwidthSubcribed) * 100
DisplayData (BandwidthUsed′, BandwidthSubcribed, BandwidthPercentage)

Figure 4: Formal Specification of Bandwidth Evaluator

b. Memory Assessor and Evaluator

The memory evaluator gets into the files of the target operating system to collect relevant information desirable for estimating the memory used by packets in the target workstation in a given transmission time. The memory evaluator generates report on the packet size, memory used, period of usage, subscribed memory, and percentage of memory used. The primary or secondary memory of a target computer used over a period of time 't' denoted by 'R' is evaluated by:

$$R = \sum_{i=1}^{n} \sum_{j=1}^{m} r_{i,j} \quad (3)$$

where $r_{i,j}$ represents the primary or secondary memory space used by the jth packet in the ith workstation. The percentage of the used memory denoted by 'P_r' is evaluated as: $P_r = 100R_u/R_w$

where R_w represents the memory size of the target computer and R_u represents the memory size used by the packets. The formal specification of the logic of memory evaluation using Z Schema notation is presented in Figure 5 as follows:

MemoryEvaluator

NumberofPackets?, NumberofWorkstations? : $\mathbb{N}$
PacketSize, TargetPacketSize? : $\mathbb{Z}$
TimeFrame, TargetPeriodofMemoryusage? : $\mathbb{Z}$
Memoryused, Memoryused′ : $\mathbb{Z}$
i, j, m, n : $\mathbb{N}$
SubcribedMemory, PercentageMemoryused : $\mathbb{Z}$

```
TotalMemoryused ← 0
n ←NumberofWorkstations?
m ←NumberofPackets?
∀i, 1 ≤ i ≤ n      /* loop over workstations */ {
    ∀j, 1 ≤ j ≤ m  /* loop over the packets transmitted in ith workstation */ {
        PacketSize ←TargetPacketSize?
        TimeFrame  ← TargetPeriodofMemoryusage?
    Memoryused′ ←  Memoryused + (PacketSize / TimeFrame )
        DisplayData (PacketSize, MemoryUsed, TimeFrame) } }
    PercentageMemoryused ← (Memoryused′ / SubcribedMemory) * 100
DisplayData (Memoryused′, SubcribedMemory, PercentageMemoryused)
```

Figure 5: Formal Specification of Memory Evaluator

c. Response Time Assessor and Evaluator

The response time evaluator gets into the files of the target operating system to collect relevant information desirable for estimating the response time used in the target workstation for a given packet transmitted. The response time evaluator generates report on the packets transmitted, queuing delay (the time packet need to wait before transmission), packet propagation time (time required to transmit a packet through the network), the processing delay, and the destination delay. The response time of all the transmitted packets in the computer network environment denoted by 'T' is evaluated by:

$$T = \sum_{i=1}^{n} \sum_{j=1}^{m} t_{i,j} / r \quad (4)$$

where $t_{i,j} = q_{i,j} + g_{i,j} + p_{i,j} + d_{i,j}$ represents the packet delays used for the jth packet in the ith workstation and 'r' is the number of packets.

$j = 1,2,\ldots,m$ (number of packets recorded in the target at period t).

$i = 1, 2,...n$ (number of targets).

$q_{i,j}$ = Queuing delay time.

$g_{i,j}$ = Packet propagation time.

$p_{i,j}$ = Processing delay time.

$d_{i,j}$ = Destination delay time.

The formal specification of the logic of response time evaluation using Z Schema notation is presented in Figure 6 as follows:

ResponseTimeEvaluator

NumberofPackets?, NumberofWorkstations? : $\mathbb{N}$

QueuingDelay?, Propagationdelay? , Processingdelay?, DestinationDelay? : $\mathbb{Z}$

ResponseTime, SumofWaitingandDelay, SumofWaitingandDelay′ : $\mathbb{Z}$

p, g, d, q : array[1..m, 1..n] of $\mathbb{Z}$

i, j, m, n : $\mathbb{N}$

SumofWaitingandDelay ← 0
n ← NumberofWorkstations?
m ← NumberofPackets?

$\forall i,\ 1 \le i \le n$ /* loop over workstations */
{
  $\forall j,\ 1 \le j \le m$ /* loop over packets transmitted in the ith workstation */
  {
    $p_{i,j}$ ← Processingdelay?
    $q_{i,j}$ ← QueuingDelay?
    $g_{i,j}$ ← Propagationdelay?
    $d_{i,j}$ ← DestinationDelay?
  SumofWaitingandDelay′ ← SumofWaitingandDelay + ($p_{i,j}$ + $q_{i,j}$ + $g_{i,j}$ + $d_{i,j}$)
  DisplayData ($p_{i,j}$, $q_{i,j}$, $g_{i,j}$, $d_{i,j}$)
  }
  ResponseTime ← SumofWaitingandDelay′ / m
  DisplayData (SumofWaitingandDelay′, ResponseTime)
}
DisplayData (ResponseTime)

Figure 6: Formal Specification of Response Time Evaluator

d. Files Assessor and Evaluator

FilesEvaluator

```
Target_id?, NumberofWorkstations?, NumberofFiles? : ℕ
DriveLabel : CHAR
DirectoryName, FileName, FileCategory : seq CHAR
ReportDriveLabel!, ReportDirectoryName!: WINDOW
, ReportFileName!, ReportFileCount! : WINDOW
i, j, m, n, count, TotalFileCount : ℕ
```

```
n ← NumberofWorkstations?
m ← NumberofFiles?
count ← 0
∀i, 1 ≤ i ≤ n      /* loop over workstations */
  {
   ∀j, 1 ≤ j ≤ m   /* loop over number of files */
    {
     ReportDriveLabel!← DISPLAY (DriveLabel);
     ReportDirectoryName! ← DISPLAY (DirectoryName)
     ReportFileName! ← DISPLAY (FileName)
     count′ ← count + 1
     ∀a,b : FilesEvaluator • a ≠ b ⇒ a. DriveLabel ≠ b. DriveLabel
     ∀c,d : FilesEvaluator • c ≠ d ⇒ c. DirectoryName ≠ d. DirectoryName
     ∀e, f : FilesEvaluator • e ≠ f ⇒ e. FileName ≠ f.FileName
     ∀g, h : FilesEvaluator • g ≠ h ⇒ g. FileCategory ≠ h.FileCategory }
 TotalFileCount′ ← TotalFileCount + count′ };
 ReportFileCount! ← DISPLAY (TotalFileCount′)
```

Figure 7: Formal Specification of Files Evaluator

The formal specification of the logic of assessing files using Z Schema notation is presented in Figure 7 above. The file evaluator gets into the files of the target operating system to collect relevant information desirable for estimating files in use in the target workstation. Files evaluator generates report on the filename(s), directory / folder name, file size, count of the number and category of files available.

e. Input-Output (I-O) Device Assessor and Evaluator

The evaluator gets into the files of the target operating system to collect relevant information desirable for estimating I-O device in use in the target workstation. I-O device evaluator generates report on device name, device makes, and device identity.

The formal specification of the logic of assessing I-O devices using Z Schema notation is presented in Figure 8 as follows:

I-ODeviceEvaluator

i, n : ℕ; Target_id?, NumberofWorkstations? : ℕ
I-ODeviceName, I-ODeviceMake, I-ODeviceIdentity : seq CHAR
ReportDeviceName!, ReportDeviceMake!, ReportDeviceIdentity! : WINDOW

---

n ← NumberofWorkstations? ;
$\forall i,$ 1 ≤ i ≤ n /* loop over workstations */ {
ReportDeviceName! ← DISPLAY (I-ODeviceName)
ReportDeviceMake! ← DISPLAY (I-ODeviceMake)
ReportDeviceIdentity! ← DISPLAY (I-ODeviceIdentity)
$\forall a, b$: I-ODeviceEvaluator • a ≠ b ⇒ a. I-ODeviceName ≠ b.I-ODeviceName
$\forall c, d$: I-ODeviceEvaluator • c ≠ d ⇒ c. I-ODeviceMake ≠ d.I-ODeviceMake
$\forall e, f$: I-ODeviceEvaluator • e ≠ f ⇒ e. I-ODeviceIdentity ≠ f.I-ODeviceIdentity }

Figure 8: Formal Specification of Input-Output Device Evaluator

4. Creation of the Mobile Agent

This involves developing the mobile agent functionality and then adding it to the universal set of agent. At this point, the agent implementation code is loaded, made executable and set for launching. The detail design of the CREATE procedure of the mobile agent system is presented in Figure 9 as follows:

Agent

AgentName : seq CHAR
∀ a, b : Agent • a ≠ b ⇒ a.AgentName ≠ b.AgentName

Create_Agent

AgentSet, AgentSet′ : ℙ Agent
NewAgent? : Agent

NewAgent? ∉ AgentSet
∀ k : AgentSet • k.AgentName ≠ NewAgent?.AgentName
AgentSet′ = AgentSet ∪ (NewAgent?)

Figure 9: CREATE Agent Procedure

The disassembling of the mobile agent into subagents is achieved using the interaction presented in Figure 10 whereas the assembling of the agent after execution is accomplished as presented in Figure 11.

Disassemble (Agent, SubAgent, n)

AgentSet : ℙ Agent
S : array[1..n] of AgentSet
i, j : 1..n
disjoint_ : ℙ (i ⟶ ℙ Agent)
Partition : (i ⟶ ℙ Agent) ⟷ Ag ℙ

∀ S : i ⟶ ℙ Agent; T : A ℙ t •
(disjoint S ⇔ ∀ i, j : dom S | i ≠ j • S(i) ∩ S(j) = ϕ) ∧
(S Partition T ⇔ disjoint S ∧ U { i : dom S • S (i)} = T)

Figure 10: Disassembling of the Agent

ASSEMBLE(SubAgent, n, Agent)

Agent ::= BandwidthEvaluator | MemoryEvaluator | ResponseTimeEvaluator | FilesEvaluator | I-ODeviceEvaluator

AgentSet, AgentSet′ : ℙ Agent

SubAgent : array[1..n] of AgentSet

i : 1..n

∀ SubAgent : i → ℙ Agent

AgentSet′ ← AgentSet ∪ {SubAgent(i)}

Figure 11: Assembling of the Agent

Finally, the process of the mobile agent disposal is presented in Figure 12.

DisposeAgent

AgentSet, AgentSet′ : ℙ Agent

Remove? : Agent

Remove? ∈ Agentset

$\exists_1$ k : Agentset • k.AgentName = Remove?.AgentName

AgentSet′ = AgentSet \ {Remove?}

Figure 12: Disposal of the Agent

5.0 Conclusion

The architecture of the mobile agent is presented. A modular architecture is proposed whereby each network resource is administered by a subagent called evaluator. The logic of each of the subagents is depicted. The input requirements and output reports for the evaluators are identified and specified. A formal specification method that relies on set notations and discrete mathematics called Z (Zed) notations is used to present the logic of the subagents. Finally, creation, disassembling and assembling as well as disposal schema of the mobile agent system were also provided. Mobile agent developers could use the formal specification using Zed notations provided in this paper as basis for takeoff.

References

[1]. Imianvan Anthony Agboizebeta (2009), "Development of a Mobile Agent System for Evaluating the Use of Bandwidth in a Computer Network", PhD Thesis, Federal University of Technology, Akure, Ondo State. Nigeria.

[2]. Aderounmu G. A. (2001), "Development of an intelligent Mobile Agent for Computer Network Performance Management", *PhD Thesis, Obafemi awolowo University, Ile-Ife, Nigeria.*

[3]. Weina He and Gaoyuan Liu (2011), The application of mobile agent in e-commerce, 3rd International Conference on Advanced Computer Control (ICACC), 2011, HARBIN.

[4]. Djamel Eddine Menacer, Habiba Drias, Christophe Sibertin-Blanc (2012), MP-IR: Market-Oriented Mobile Agents System for Distributed Information Retrieval, Advances in Intelligent and Soft Computing, Volume 122, pages 379-390, 2012.

[5] Huy Hoang To, Shonali Krishnaswamy, and Bala Srinivasan (2005), Mobile Agents for Network Management: When and When Not! , ACM Syposium on Applied Computing.

[6]. Diller A., (1994), Z : An Introduction to Formal Methods, (2nd edition), John Wiley and Sons

[7]. Spivey J. M. (1998), "The Z notation: A Reference Manual", Prentice Hall International, United Kingdom.