A Monitoring Model for Hierarchical Architecture of Distributed Systems

Phuc Tran Nguyen Hong
Danang University of Education
The University of Danang
Danang, Vietnam

Son Le Van
Danang University of Education
The University of Danang
Danang, Vietnam

***Abstract*—Distributed systems are complex systems and there are a lot of the potential risks in the system, so system administrators need to have some effective support tools for network management. The architecture information of distributed systems is an essential part of distributed system monitoring solutions, because it provides general information about monitored objects in the system for administrators, as well as supports administrator in quickly detecting change of topology, error status or potential risks that arise during operation of distributed systems. The modeling approaches have an important role in developing monitoring solutions, in which they are background to develop algorithms for monitoring problems in distributed systems. This paper proposes an approach in order to model for hierarchical architecture of objects in distributed systems, in which consists of architecture of monitored objects, networks, domains and global distributed systems. Based on this model, a basic monitoring solution for hierarchical architecture of distributed systems is developed and this solution is able to provide administrators more important architecture information such as the topology of hardware components, processes, status of monitored objects, etc.**

Keywords—architecture; distributed systems; model; monitored objects; monitoring

I. INTRODUCTION

Distributed systems (DS) are complex systems, which have always challenged for system administrator a lot [5,9]. A hardware malfunction, a faulty process or an abnormal event occurs on the system may affect other events taking place at different locations in the running environment of system. These symptoms can cause a bad effect on performance and stability of the system, they can also cause of errors of related processes and incorrect results of distributed applications. In order to ensure the effective operation of DS, global system information in general and information of each object in particular is critical issues. Many technical solutions have been researched and developed to support administrators in monitoring the system. Through the survey and review some typical monitoring works such as [10,11,13,14,15,16,17] in paper [4,8], in which we presented in detail the technical details, some advantages as well as disadvantages of these solutions. The survey result on implementation solutions and function of monitoring systems is presented in Table I and II.

TABLE I. THE IMPLEMENTATION SOLUTION

Monitoring System	Implementation Solution		
	Software	*Hardware*	*Hybrid*
JADE [13]	•		
MonALISA [11]	•		
MOTEL [17]	•		
ZM4/SIMPLE [14]			•
NON-INVASIVE MONITOR [8]		•	

We are aware that there are many implementation solutions to deploy monitoring system. However, with the advantages such as flexibility and mobility, the ease of maintenance, etc the software solution has been widely deployed in many TCP/IP monitoring products.

TABLE II. THE FUNCTION OF MONITORING SYSTEMS

Monitoring System	Monitoring Function			
	Computing	*Performance*	*Object*	*Operation*
ZM4/SIMPLE [14]	•			
JADE [13]	•			
MonALISA [11]		•		
SNMP [8,16]		•		
MOTEL [17]			•	
CorbaTrace [10]			•	
Tools (OS,...) [8]				•

From Table II, we see that the monitoring systems for DS can be divided into two groups: specific monitoring (SM) and general operations (GM) for monitored object in DS.

- SM consists of monitoring systems that monitor specific issues of monitored objects in DS such as MonALISA, MOTEL, SNMP, etc. SM can be seen as a special monitoring layer to monitor details such as traffic, performance, computing, etc. Most of these solutions in SM are only focused on solving the requirements for specific monitoring issues between

objects and have not yet been really interested in the global architecture of monitored objects in DS. For example, ZM4/SIMPLE is deployed to do performance evaluation for and parallel and distributed programs; MonALISA is deployed to monitor and help manage and optimize the operational performance of Grids; etc.

- GM consists of monitoring systems that monitor general operations of the monitored objects in DS such as built-in tools of devices or utilities in OS (Operating System). GM can be seen as a common monitoring layer in which provide abilities to monitor architectures and operations of monitored objects (MO) such as configuration, status, communication, connections, etc. For example, taskmgr and netstat commands are in Windows OS; prstat command is Solaris OS, etc.

Therefore, we can divide monitoring for DS into two basic stages:

- The first stage is general monitoring with monitoring solutions in GM, the global architecture information of monitored DS in general and the information about general operations of monitored objects in particular are essential in this stage, because they can support administrator for quickly detecting common errors or error domains that arise during operation of the system [4].
- The second stage is extended survey with monitoring solutions in SM in order to go into more detail in special monitoring information.

Thus, the monitoring solutions in GM are considered as a high level monitoring facilities to monitored DS before using other monitoring solutions in SM to deeper analysis. However, GM are now mainly based on tools (OS, utilities) that developed by device vendors side or operating systems side. These built-in tools have some disadvantages such as discrete monitoring information, independent of each device, etc [4,8], hence the global of DS cannot be solved with these built-in tools. The global architecture should be continued to research and develop more effective, the goal of the paper focus on solving this problem base on modeling for architecture of MO and building hierarchical monitoring entities respectively.

When monitored systems have basic changes about architectures, behaviors and operation environments, the technical solutions must be modified and updated appropriately for new changes and management requirements. As system specification methodology is generally and flexibly, the modeling approach is considered more appropriate for systems that have a lot of changes and the approach is widely used in discrete event systems, computer protocols [1,3,7]. In the DS, the modeling approach also achieved some certain results [2,6]. The modeling approaches play an important role, in which it is used as a basis layer for algorithm and solution development in monitoring, diagnosing and controlling issues independently. Therefore, the modeling for MO in DS is really necessary, the objective of the paper is based on the research results on DS and set theory [1,4,6], we focus on building a formal model for the hierarchical architecture of MO in DS, in which consists of architecture of monitored objects, networks, domains and global distributed systems. We also present a basic monitoring model for the hierarchical architecture of DS, in which can show DS topology visually as well as the local operations and the communication operations of MOs in the DS.

The paper is organized as follows: In section II, we present architectural model for a MO in DS and the composition operation that allows us to combine many MOs into a composition model, we describe hierarchical architecture of monitored objects in DS. Section III focuses on the modeling solution that is able to monitor the architecture of DS. Finally, section IV concludes with the current work and future perspectives.

II. THE ARCHITECTURE MODEL FOR DISTRIBUTED SYSTEMS

DS consists of many heterogeneous devices such as stations, servers, routers, etc. These devices are considered physical objects in DS and communicate to each other in the system; each device consists of many hardware components such as CPU, HDD, etc. and software components such as processes. These components are associated with information about the corresponding states and behaviors, general operations of MO is described by Fig. 1, they can be divided into two basic parts such as internal part – local operations and external part – communication operations [4].

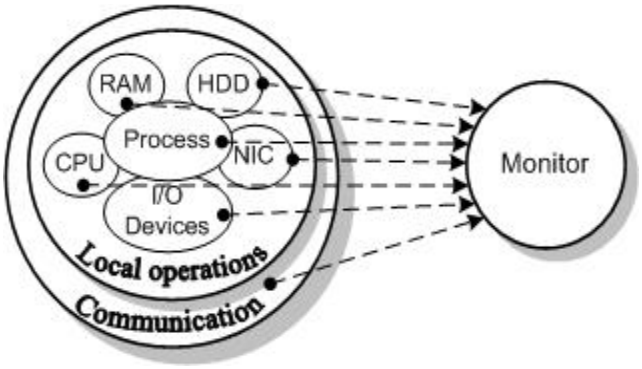

Fig. 1. General operations of the monitored object

- The local operations: these operations include processing, computing, resource requirements for process computations. The operations are locally performed within that object and use system resource such as CPU, RAM, etc. in running time.
- The communication operations: these operations are functions that interact with other objects in the system such as inter-process communication, controlling to interact with management system, etc. These operations are used to communicate with other objects on the system.

All of local and communication operations are based on system resource of MO such as CPU, RAM, I/O, etc. and information of these operations is dynamic in their running process, while system resource of MO is static information. Therefore, architecture of MO will consist of static information of MO and dynamic information of local and communication operations.

MOs are considered as nodes that are connected according to specific architecture and can perform interactive communication to each other. Hence, Architecture model describes the structure of nodes along with the related information of each node, the link between nodes, message propagation via its port, etc. Based on this information, we can determine the physical structure and the state of the nodes in the system.

From result of research on DS and monitoring systems, we can see that DS consists of many heterogeneous objects and topologies that communicate to each other. With point of view the domain-based management for large scale systems, the multi-level domain has been used to manage for DS [18], in which consists of local object level, network and domain level. The hierarchical architecture of monitored objects in DS can be presented as Fig. 2:

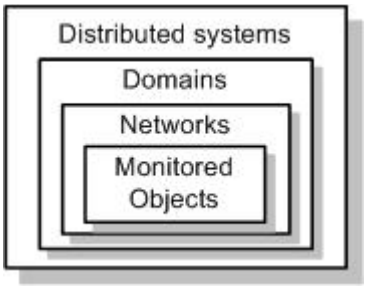

Fig. 2. The hierarchical architecture of objects in DS

Therefore, the architecture model of DS and monitoring model are presented in hierarchical architecture as Fig. 2 in order to deploy a suitable monitoring solution.

A. Architecture Model of MO

Let AM be an architecture model of monitored node, the AM is a 9-tuple and expressed as follows:

$$AM = (NODES, DOMAINS, NETS, LINKS, PORTS, port, communication, status, event) \quad (1)$$

NODES = {set of static and dynamic information of nodes}= {NODES_S} ∪ {NODES_O} ∪ {NODES_A}

where: NODES_S consists of system resource information of MO and this is static information such as Cpu, RAM, etc; NODES_O consists of information about local and communication operations such as processes; NODES_A consists of error or abnormal information of hardware and software components such as I/O errors, overload; NODES_O and NODES_A are dynamic information.

DOMAINS = {set of domain information such as name,...}

NETS ={set of network information such as IP, network,...}

LINKS = {set of link information between nodes}

PORTS= {set of port: internal and external port}

port is a function that identify communication ports in NODES: local ports (internal) and external ports (send/receive to nodes not in *NODES*), port(*NODES*) ∈ *PORTS*

communication is a function that identify communication connections between nodes, {(*NODES*,*PORTS*) → (*NODES*, *PORTS* x d)}, delay d =[t_{min}; t_{max}]

status is a function that identify node states in which consist of normal or abnormal status, status(*NODES*) ∈{*S_NOR*} or {*S_ABNOR*}, where: *S_NOR* is set of normal status such as up, communicating,...; S_AB*NOR* is set of abnormal status such as down, overload,...

event is a function that identify node events such as request, messages,... These events consist of internal (internal_events) and external events (external_events)

In order to visually present architecture model, we denote *AM* for architecture model, n∈*NODES*, d∈*DOMAIN*, net∈*NETS*, L∈*LINKS*, {p_1, p_2}∈*PORTS*. So architecture model AM can be visually described as Fig. 3

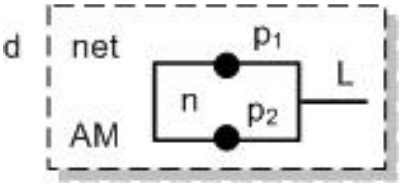

Fig. 3. The architecture model of a node

With this architecture model, we can determine the related information of node such as resource information, operations, status, etc base on elements of AM.

For example, give an architecture model AM of running node as Fig. 3 without communication operations, and then AM can be expressed as follows:

AM=({*n*}, {*d*}, {*net*}, {*L*},{p_1, p_2}, *port*, *communication*, *status*, *event*)

Where {*n*}={system information such as device name, CPU,...} ∪ {running processes, I/O operations,...} ∪ {error status,...};

port={internal ports: p_1, p_2};

communication={no communication with others}; *status*={up};

event={local operation events}

Therefore, architecture model of monitored object will give us more important information about that object such as local operations (internal operations) as well as communication operations (external operations). Based on this architecture information, we can determine operations, errors or abnormal states that occur in running time of the node.

B. Compositon Model

DS is complex system in which consists of many heterogeneous devices (nodes) and is organized according to hierarchical architecture as Fig. 2. So architecture model of DS will be set of architecture model AM of nodes in system. In order to ensure more efficient to build architecture model of DS, we use composition operation as described here.

Let AM_1, AM_2 be architecture model of node 1 and node 2 in system, let || be composition operator (concurrent) for AM_1 and AM_2. Composition operation is shown in Fig. 4.

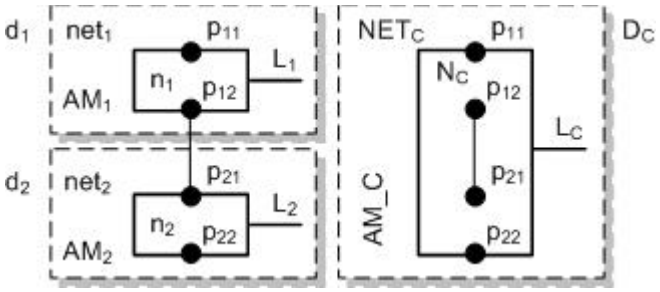

Fig. 4. Composition architecture of two nodes

The architectural model after composition AM_1 and AM_2 is AM_C, and AM_C is also a 9-tuple as expression (2).

$$AM\_C = AM_1 \,||\, AM_2 = (NODES_C, DOMAIN_C, NETS_C, LINKS_C, PORTS_C, port, communication, status, event) \quad (2)$$

Where:

$NODES_C = NODES_1 \cup NODES_2 = \{N_C\} = \{n_1, n_2\}$;

$DOMAIN_C = DOMAIN_1 \cup DOMAIN_2 = \{D_C\} = \{d_1, d_2\}$;

$NETS_C = NETS_1 \cup NETS_2 = \{NET_C\} = \{net_1, net_2\}$;

$LINKS_C = LINKS_1 \cup LINKS_2 = \{L_C\} = \{L_1, L_2\}$;

$PORTS_C = PORTS_1 \cup PORTS_2 = \{p_{11}, p_{12}, p_{21}, p_{22}\}$;

$port = port(NODES_C) = PORTS_C.internal \cup PORTS_C.external$

with $PORTS_C.internal = \{p_{12}, p_{21}\}$;

$PORTS_C.external = \{p_{11}, p_{22}\}$

$communication = communication(NODES_C, PORTS_C) = \{(n_1,p_{12}) \leftrightarrow (n_2,p_{21}), (n_1,p_{11}) \leftrightarrow (n_i,p_i), (n_2,p_{22}) \leftrightarrow (n_i,p_i)\}$, $i \notin \{1,2\}$

$status = status(NODES_C) \rightarrow \{S\_NOR\}$ or $\{S\_ABNOR\}$

where:

$status(NODES_C) \in \{S\_NOR\}$ when $status(n_1) \in \{S\_NOR\}$ and $status(n_2) \in \{S\_NOR\}$;

$status(NODES_C) \in \{S\_ABNOR\}$ when $status(n_1) \in \{S\_ABNOR\}$ or $status(n_2) \in \{S\_ABNOR\}$

$event = event(NODES_C) = internal\_events(NODES_C) \cup external\_events(NODES_C)$

$internal\_events(NODES_C) = internal\_events(n_1) \cup internal\_events(n_2) \cup \{\sigma_{12}\}$;

$external\_events(NODES_C) = external\_events(n_1) \cup external\_events(n_2) - \{\sigma_{12}\}$

with $internal\_events(n_1)$: local events in node 1;

$internal\_events(n_2)$: local events in node 2;

σ_{12}: communication events between node 1 and 2

Therefore, composition model AM_C describes operation information of two nodes in which consist operations of each node and communication between node 1 and node 2.

Similar to architecture information of MO, we can easily determine operations, errors or abnormal states of node 1 and node 2 that occur in running time based on elements in the model AM_C.

C. Modelling for Architecture of DS

As we presented in section II, topology of DS can be seen as hierarchical structure consists of many levels such as local object, network and domain level, in which global DS consists of n (n>0) domains and can communicate with each other via telecommunication networks, each domain consists of m (m>0) heterogeneous networks interconnect to each other, and each the network consists of k (k>0) physical devices. All off them can collaborate, exchange and share information to each other. Therefore, the modeling for architecture of DS will be done with four levels: MO model, network model, domain and global DS model. The architecture model for DS can be expressed as follows:

- *The architecture model of MO (AM_MO):* AM_MO describe architecture information of MO and is expressed as follows:

$$AM\_MO = (NODES_{MO}, DOMAIN_{MO}, NETS_{MO}, LINKS_{MO}, PORTS_{MO}, port, communication, status, event) \quad (3)$$

- *The architecture model of a network (AM_MS)*: Give a network consists of k monitored objects $\{MO_1, MO_2,...,MO_k\}$ and set of $\{AM\_MO_1, AM\_MO_2,..., AM\_MO_k\}$ is architecture model of these objects. Hence, AM_MS is a composition model of architecture model AM_MOs respectively:

$$AM\_MS = AM\_MO_1 \,||...||\, AM\_MO_k \quad (4)$$

From composition result of expression (2), AM_MS is expressed as follows:

$$AM\_MS = (NODES_{MS}, DOMAIN_{MS}, NETS_{MS}, LINKS_{MS}, PORTS_{MS}, port, communication, status, event) \quad (5)$$

- *The architecture model of domain (AM_MD)*: Similar to the AM_MS, give a domain consists of m networks corresponding to $\{AM\_MS_1,\ldots, AM\_MS_m\}$, AM_MD is a composition model of AM_MSs respectively:

$$AM\_MD = AM\_MS_1 \,||...||\, AM\_MS_m \quad (6)$$

AM_MD is expressed as follows:

$$AM\_MD = (NODES_{MD}, DOMAIN_{MD}, NETS_{MD}, LINKS_{MD}, PORTS_{MD}, port, communication, status, event) \quad (7)$$

- *The architecture model of DS (AM_DS)*: As DS is a set of n domains $\{AM\_MD_1,\ldots, AM\_MD_n\}$, so AM_DS is a composition model of AM_MDs respectively:

$$AM\_DS = AM\_MD_1 \,||...||\, AM\_MD_n \quad (8)$$

AM_DS is expressed as follows:

$$AM\_DS = (NODES_{DS}, DOMAIN_{DS}, NETS_{DS}, LINKS_{DS}, PORTS_{DS}, port, communication, status, event) \quad (9)$$

From expression (3)÷(9), we see that AM_MO, AM_MS, AM_MD and AM_DS are built from composing architecture model of basic objects. Thus, information of model AM_MO, AM_MS, AM_MD and AM_DS will describe all of system information, operations, links and state information (normal, abnormal, error) of elements in them. For example, related information of any network will describe in expression (5), so $NODES_{MS}$ will describe information of all MO in a network because $NODES_{MS} = NODES_1 \cup NODES_2 \cup \ldots$ in which consists of system information, operations and error or abnormal information of all MO. Communication ports $PORTS_{MS}$ will display all of ports of objects in the network, because $PORTS_{MS} = PORTS_1 \cup PORTS_2 \cup \ldots$. Therefore, in order to determine error or abnormal states of network according to AM_MS, we only observe $NODES\_A_{MS}$, because $NODES\_A_{MS} = NODES\_A_1 \cup NODES\_A_2 \cup \ldots$

III. The Monitoring Solution for Hierarchical Architecture of Distributed Systems

A. The Teachnical Base and Basic Monitoring Solution

The objective of the monitoring system is observation, collection and inspection information about the operations of the hardware and software components, communication events of MO. This information supports actively in system management.

The general monitoring architecture can be divided into 3 parts as Fig. 5.

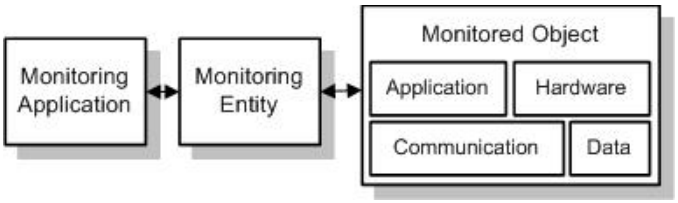

Fig. 5. General monitoring architecture

Monitored Object (MO) consists of independent objects such as switches, routers, workstations, servers, ... these objects have their own hardware and software resource. In order to describe architecture information of MO at time t, we use function *monitoring_info(MO,t).*

Monitoring Application (MA) is designed to support for the management Objects (administrators or other management agents). MA entity interacts with monitoring entity to support the generation of monitoring requirements and present the results of monitoring are measured from monitoring entity.

ME (Monitoring Entity) is designed to instrument the monitored objects, the instrumentation information of the system will be processed to generate the corresponding monitoring reports and send to MA. In order to describe result of monitoring entity ME at time t, we use function result_ME(ME,t).

Thus, monitoring result of ME at time t for MO can be expressed as follows:

$$result\_ME(ME, t) = monitoring\_info(MO, t) \quad (10)$$

The monitoring system for DS consists of more MEs and MAs, they are not fixed and independently operate on each domain of DS, and monitoring information is exchanged between the MEs and MAs by message passing.

With the hierarchical architecture model of DS is presented as the previous session, hierarchical architecture of DS consists of four levels such as MO, network, domain and global DS. In order to collect the architecture information of DS, monitoring entities are designed in accordance with the hierarchical architecture of DS and we use four monitoring entities to monitor hierarchical architecture of DS:

- *The monitoring entity ME_MO* for object: *ME_MO* observes and collects the architecture information of MO. Because architecture model of MO is expressed as *AM_MO* in (3), the monitoring result of *ME_MO* at time t can be expressed as follows:

$$result\_ME(ME\_MO, t) = monitoring\_info(AM\_MO, t) \quad (11)$$

- *The monitoring entity ME_MS* for network: *ME_MS* observes and collects the architecture information of a network. Because architecture model of a network is expressed as *AM_MS* in (4), the monitoring result of *ME_MS* at time t can be expressed as follows:

$$result\_ME(ME\_MS, t) = monitoring\_info(AM\_MS, t) \quad (12)$$

- *The monitoring entity ME_MD* for domain: *ME_MD* observes and collects the architecture information of a domain. Because architecture model of a domain is expressed as *AM_MD* in (6), the monitoring result of *ME_MD* at time t can be expressed as follows:

$$result\_ME(ME\_MD, t) = monitoring\_info(AM\_MD, t) \quad (13)$$

- *The monitoring entity ME_DS* for distributed systems: *ME_DS* observes and collects the architecture information of DS. Because architecture model of DS is expressed as *AM_DS* in (8), the monitoring result of *ME_DS* at time t can be expressed as follows:

$$result\_ME(ME\_DS, t) = monitoring\_info(AM\_DS, t) \quad (14)$$

From expression (11÷14), the monitoring system for hierarchical architecture of DS will be set of monitoring entities {*ME_MO, ME_MS, ME_MD, ME_DS*} that are designed as Fig. 6.

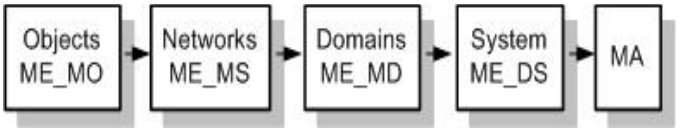

Fig. 6. Architecture of monitoring entities

The monitoring entities *ME_MO* will be installed on all of MO in DS, they observe and collect the architecture of MOs, and supply monitoring reports to network monitoring entity *ME_MS*. *ME_MS* runs composition operation in order to synthesize monitored information from *ME_MOs* in the same network and supply network monitoring reports to domain monitoring entity *ME_MD*. The operation of *ME_MD* and *ME_MS* has also run into similar processes as above. The monitoring implementation of *ME_MO* is designed as Fig. 7.

Fig. 7. The monitoring implementation of ME_MO

In order to observe and collect the architecture of MO in DS, we use protocols, APIs and built-in tools of operating system. The popular protocols are used in management network to monitor status or traffic of MO such as ICMP (Internet Control Message Protocol) [12,19], SNMP (Simple Network Management Protocol) [4,19]. The APIs and tools are used to observe and collect system information, operations as well as communication ports of components in MO such as the Window API, Linux API, libraries,

The modeling for monitoring solution bases on four levels such as MO, network, domain and global DS which are suitable with point of view the domain-based management, this hierarchical monitoring architecture have advantages to develop some distributed algorithms in levels of DS management in which the level MO focus on observing and collecting the architecture information of MO, level ME_MS, ME_MD and ME_DS are responsible for synthesizing and processing the monitoring information.

Therefore, the collection and composition process for building the architecture of DS is implemented as following sequence:

MO → network → domain → global DS

The collection and composition process of hierarchical monitoring architecture are described detail in Fig. 8.

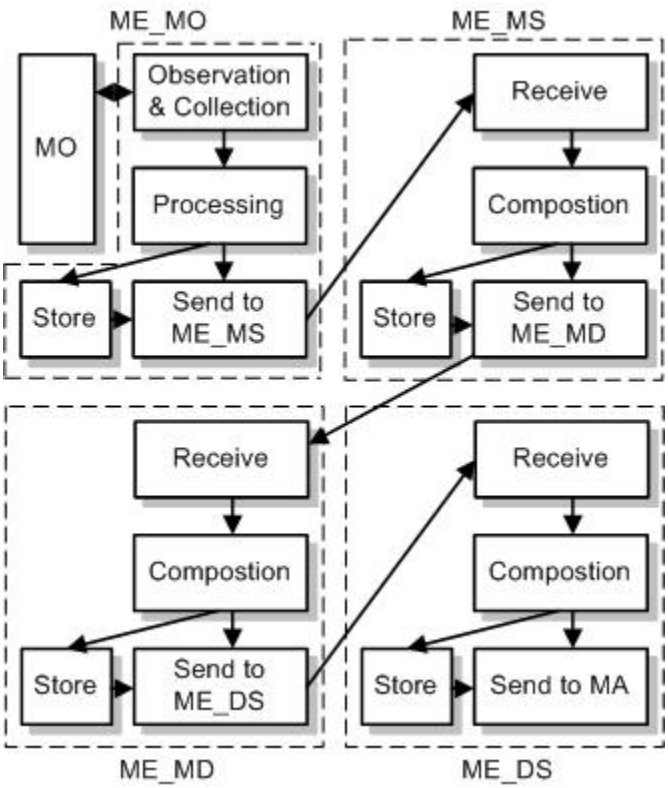

Fig. 8. Basic monitoring process for architecture of DS

At each level of monitoring entities (MO, network, domain, global DS) will collect full monitoring information of their monitored objects. First level, ME_MO collects and processes all of monitoring information of components such as Processes, CPUs, etc. Second level, ME_MS composes all of monitoring information of MOs in the same network and creates the monitoring report for architecture of this network. Third level, ME_MD composes all of monitoring information of networks in the same domain and creates the monitoring report for architecture of this domain. Fourth level, ME_DS composes all of monitoring information of domains in DS and builds the monitoring report for architecture of DS.

In order to analyze the architecture information of DS, the sequence of steps is implemented as follows:

global DS → domain → network → MO.

For example, suppose that distributed system CDS consists of two domains $\{d_1, d_2\}$, each of domains contains one network: net_1 in domain d_1, net_2 in domain d_2, network net_1 consists two nodes $\{n_1, n_2\} \in NODES$, and network net_2 consists three nodes $\{n_3, n_4, n_5\} \in NODES$.

After the step ME_DS composes all of monitoring information for architecture of DS, we have all of architecture information of CDS that is expressed by the architectural model AM_DS in (9). Therefore, the architecture of CDS is analyzed as follows:

$DOMAIN_{CDS}$={d_1, d_2};

domain(d_1)={net_1}; *domain*(d_2)={net_2};

$NETS_{CDS}$ ={net_1, net_2};

net(net_1) = {n_1, n_2};

net(net_2) = {n_3, n_4, n_5};

From above architecture information, the hierarchical architecture of CDS is presented as Fig. 9.

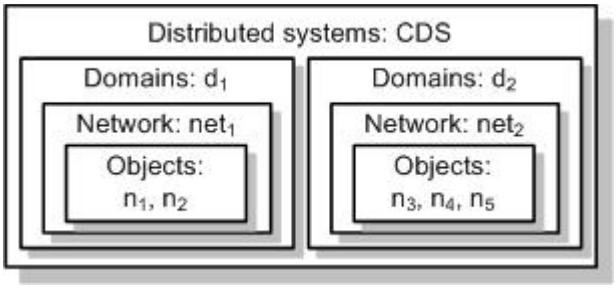

Fig. 9. The architecture of CDS

In normal case, all of monitored objects {n_1, n_2, n_3, n_4, n_5} are running smoothly, set of information of objects in CDS contains in $NODES_{CDS}$ which consits of system resource information $NODES\_S_{CDS}$, information about operations $NODES\_O_{CDS}$ and error information $NODES\_A_{CDS}$. Because the CDS has not any error, $NODES\_A_{CDS}$ has not any description. Suppose that objects n_5 is down or overload, then $NODES\_A_{CDS}$ contains down state or overload state of n_5. Base on $NODES\_A_{CDS}$, we will monitor all of errors or abnormal of CDS.

The connection and communication information of objects in CDS such as $LINKS_{CDS}$, $PORTS_{CDS}$, *port*, *communication*, etc will support us in building algorithms to display network visualization which consists of communication operations and link diagrams of nodes, networks and domains.

B. The Initial Experimental Results

Based on the model is presented in the previous sections, we designed a MCDS (Monitoring for Complex DS) system that consists of a set of monitoring entities (*ME_MO*, *ME_MS*, *ME_MD*, *ME_DS* as Fig. 6) for monitored objects, group of monitored networks, monitored domains and global system. The goal of MCDS is that monitor the architecture and operations information of devices on the VMSC3 system (a network system of VMS company at Vietnam), in which the architecture of monitored system can be displayed in hierarchical architecture as Fig. 2; operations information consists of local and communication operations (as Fig. 1) of monitored objects in VMSC3 system such as process, communication ports, etc.

The initial experimental results are shown in Fig. 10, in which presents some monitoring forms of MCDS such as group of forms about basic architecture of objects and group of objects in VMSC3, as well as general description information about objects in system such as devices name, IP,...; group of forms about the communication and local operations information of monitored objects such as system information (descriptions, locations, OS…), hardware information (Cpu, Ram, I/O, …) and information on the operations of the processes, status, communication, etc. This information is collected by *ME_MO* and will be used to send to other monitoring entities (ME_MS, ME_MD and ME_DS) by the message passing mechanism.

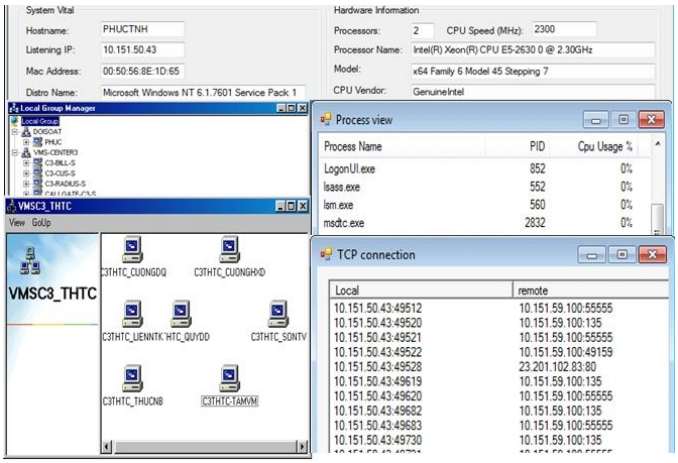

Fig. 10. MCDS for the monitored object in VMSC3

In order to evaluate this monitoring model for hierarchical architecture of distributed systems, we use some notation such as M_{our} for our model; M_{GM} for monitoring models is mainly based on tools (OS, utilities). Some evaluations as follows:

- *Monitoring presentation*: Because built-in tools only run object itselft or by remote, so discrete monitoring information, independent of each device. Therefore, M_{GM} focuses on presenting monitoring information directly of objects MO in DS, it is only local part of DS. M_{our} presents monitoring results as the hierarchical architecture such as objects, networks, domains and global DS, so the presentation results consist of local part and global system, it provides an overview on monitored DS for administrators and is more appropriate for architecture of complex DS in the practical environment.
- *Monitoring function*: Solutions in SM group are only pay attention to solve special monitoring issues between MOs (computing, traffic, etc). M_{GM} focuses on general operations of MOs (devices, components) in DS, however this solution provides discrete monitoring information and have some disadvantages that we presented in [4,8]. M_{our} monitors general operations of MOs in DS with multi-levels, so it provides multi-level monitoring reports (local objects, networks, domains, etc). Therefore, M_{our} supports administrators in managing monitored objects in system more advantage and quickly detecting errors, potential risks arising during operation of DS based on elements of model at each level.

- *Implementation time*: Because most of the built-in tools in M_{GM} monitor DS by using discrete tools (OS or device vendors utilities), we have to type (or select) one or more commands respectively. M_{our} monitor based on MEs and MEs communicate to each other by the message passing. So, monitoring time with M_{our} will take less than M_{GM}. In order to evaluate for monitoring time, basic monitoring time for object O_i between M_{GM} and M_{our} are expressed as follows:

$$t_{GM}(O_i) = t_{Remote}(O_i) + \sum_{i=1}^{n} t_T(C_i) + \sum_{i=1}^{n} t_{R\_P}(C_i) \quad (15)$$

$$t_{our}(O_i) = t_{mes}(O_i) + \sum_{i=1}^{n} t'_{R\_P}(C_i) \quad (16)$$

Where: $t_{GM}(O_i)$ and $t_{our}(O_i)$ are monitoring time of M_{GM} and M_{our} for object O_i; $t_{mes}(O_i)$, $t_{Remote}(O_i)$, $t_T(C_i)$, $t_{R\_P}(C_i)$ and $t'_{R\_P}(C_i)$: time for clicking function and monitoring message to object O_i, remote to object O_i, typing (or selecting) command C_i, running-presentation results of command C_i respectively.

Suppose that $t_{R\_P}(C_i)=t'_{R\_P}(C_i)$ for same monitoring function in M_{GM} and M_{our} with the same O_i.

The experimental results are implemented in VMSC3, in which nodes work on MS windows environment. Result consists of some cases as follows:

Monitoring implementation in object itself: $t_{Remote}(O_i)=0$, $t_{mes}(O_i)\approx 2\div 3$s (clicking), $t_T(C_i)\approx 10\div 20$s; with components as Fig. 2, we use about 7 commands respectively (n=7), so $\sum_{i=1}^{7} t_T(C_i) \approx$ 70÷140s, hence $t_{GM}(O_i)>t_{our}(O_i)$ (approximately 68÷137s)

Monitoring implementation for a remote object on LAN: $t_{Remote}(O_i)\approx 20\div 30$s, $t_{mes}(O_i)\approx 3\div 4$s, $t_T(C_i)\approx 11\div 22$s, so $\sum_{i=1}^{7} t_T(C_i)$ ≈77÷154s, hence $t_{GM}(O_i)>t_{our}(O_i)$ (approximately 94÷180s)

Therefore, we are easily aware that $t_{GA}(O_i) > t_{our}(O_i)$

When monitoring implements for a group of m objects on a network S_i: $t_{GA} = \sum_{i=1}^{m} t_{GA}(O_i)$ for M_{GM} and time for M_{our} is $t_{our} = \sum_{i=1}^{m} t_{our}(O_i) = t_{our}(ME\_MS_{S_i})$, where: $ME\_MS_{S_i}$ is a ME_MS for S_i and can be seen as a network object. Hence, the bigger monitoring time t_{GM} compares with t_{our} ($t_{GM}>>t_{our}$) when the bigger m is.

The hierarchical architecture model of monitored objects in DS and experimental result show that our proposed model is feasible and will overcome the disadvantages of specific built-in tools in monitoring hierarchical architecture of DS, as well as actively support administrators in managing DS in according to multi-level such as object level, network, domain and global DS level. Some actively results of above proposed model are presented in Table III.

TABLE III. SOME RESULTS BETWEEN BUILT-IN TOOLS (GM) AND MCDS

Issue	Specific built-in tool	MCDS
Monitoring function	Monitoring for general operations of MOs in DS, based on tools that developed by device vendors side or operating systems side	Monitoring for general operations of MOs in DS with multi-levels, based on set of monitoring entities: objects, networks, etc.
Implementation of monitoring requirements	Administrators must have good skill to use all support tools (ultilities) integrated with monitored objects and OS of MOs.	Administrators only run monitoring requirements in MCDS by click on menu.
Implementation method	Manual method, based on remote connection and tool is manually executed.	Automatic method, based on implementing of monitoring agents.
Monitoring scope	Discrete, objects, local	Local, global, large scale DS
Monitoring time	Depending on skill of the administrators and network infrastructure.	Depending on monitored network infrastructure.
Error detection	Manually	Automatic warning
Diagnosing, and evaluation	Manually, depending on the skill of the administrator, local.	Automatic, multi-level: objects, netwoks, domains,…

IV. CONCLUSION

The modeling has an important role in the development of efficient algorithms for the monitoring problems in DS. This paper proposes a modeling method for the basic architecture of objects in DS, the monitoring solution for hierarchical architecture of DS. With the proposed models, we develop the MCDS solution that supports administrators for monitoring information visually such as the DS topology, the operations and status information of objects in the system, etc. Based on the monitoring entities, we easily develop extensions for these monitoring entities to provide complete online architecture information that effectively support for administrators, as well as allow storing monitoring data into database for the synthesis, evaluation and analysis of historical monitoring data later. This information is actively useful for the appropriate management decisions and controlling actions the monitored system.

In order to effectively deploy the monitoring solution for the distributed systems, we continue investments to complete the solution and optimize for monitoring algorithms, the dynamic management model and effective communication model for monitoring entities, as well as the analyzing techniques that optimize the computations for the large number of monitoring information in the large-scale systems.

REFERENCES

[1] Christos G. Cassandras, Stéphane Lafortune, "Introduction to Discrete Event Systems", 2nd edition, Springer, 2008.

[2] Gabriel A. Wainer, Pieter J. Mosterman, "Modeling and simulation theory and applications", CRC Press, 2011.

[3] Gerard J. Holzmann, "Design and validation of computer protocols", Prentice Hall, 1991.

[4] Phuc Tran Nguyen Hong, Son Le Van, "An online monitoring solution for complex distributed systems based on hierarchical monitoring agents", The Fifth International Conference on Knowledge and Systems Engineering, pp 191-202, 2013.

[5] Son Le Van, Phuc Tran Nguyen Hong, "Researching on an online monitoring model for large-scale distributed systems", Proceedings of the 13th National Conference in Information and Communication Technology, Hungyen, Vietnam, 2010.

[6] Weilong Hu, Hessam S. Sarjoughian,"A co-design modeling approach for computer network systems", Proceedings of the 2007 Winter Simulation Conference, 2007.

[7] Yannick Pencolé, marie-odile cordier, Laurence Rozé, "A decentralized model-based diagnostic tool for complex systems", International Journal on Artificial Intelligence Tools (IJAIT), 2002.

[8] Phuc Tran Nguyen Hong, Son Le Van, Huy Nguyen Xuan, "The technical overview report on some monitoring solutions for distributed systems", The technical servey report, Danang University of Technology, The University of Danang, Vietnam, 2014.

[9] George Coulouris, Jean Dollimore, Tim Kindberg and Gordon Blair, "Distributed systems concepts and design", 5th Edition, Addison Wesley Press, 2011.

[10] http://corbatrace.sourceforge.net

[11] http://monalisa.caltech.edu/monalisa.htm

[12] https://www.ietf.org/rfc/rfc792.txt

[13] Jeffrey Joyce , Greg Lomow, Konrad Slind, Brian Unger, "Monitoring Distributed Systems", ACM Transactions on Computer Systems, 5(2), pp. 121-150, 1987.

[14] R.Hofmann, "The Distributed Hardware Monitor ZM4 and its Interface to MEMSY", Universit¨at Erlangen, IMMD VII, 1993.

[15] Sheng-Yuan Yang, Yi-Yen Chang, "An active and intelligent network management system with ontology-based and multi-agent techniques", Expert Systems with Applications,38(8), 2011.

[16] Phuc Tran Nguyen Hong, Son Le Van, "Monitoring of large-scale distributed systems based on SNMP development", The Journal of Science and Technology, Danang University, I(8),79-84, 2012.

[17] Xavier Logean, "Run-time Monitoring and On-line Testing of Middleware Based Communication Services", PhD dissertation, Swiss Federal, 2000.

[18] Kwang-Hui Lee, "A Distributed Network Management System", Global Telecommunications Conference, IEEE,1994.

[19] Aman Mahajan, Haresh Joshi , Sahil Khajuria , Anil k Verma, "ICMP, SNMP: Collaborative Approach to Network Discovery and Monitoring", International Journal of Smart Sensors and Ad Hoc Networks (IJSSAN) ISSN No. 2248-9738 Volume-1, Issue-3, 2012.