A Monitoring Solution for Basic Behaviors of Objects in Distributed Systems

Phuc Tran Nguyen Hong and Son Le Van
Danang University of Education, Danang, Vietnam
E-mail: phuc.nguyenhong@mobifone.vn, anhtm.dng@vnpt.vn
Correspondence: Phuc Tran Nguyen Hong
Communication: received 26 June 2016, revised 9 January 2017, accepted 27 March 2017

Abstract: Information about communication behaviors of objects in a distributed system is critical because it will provide comprehensive data on the operations of the objects in the system. In addition, this information will support system administrators in quickly detecting special states or events, potential risks, as well as locations of errors that occur in the system. In this paper, we propose a method to model basic operations for monitored objects in distributed systems and a basic monitoring solution for these operations to monitor communication operations between objects in these systems. These proposals focus on a hierarchical architecture of objects in distributed systems, consisting of multiple levels such as monitored objects, networks, domains, and global systems. Based on these models, we can build a suitable monitoring solution to support system administrators in operating and diagnosing communication behaviors of objects in distributed systems.

Keywords: *Distributed systems, object monitoring, behavior model.*

I. INTRODUCTION

Distributed systems (DSs) are complex systems that consist of many heterogeneous devices, topologies, services and technologies, and also include a large number of communication events interacting with each other on a geographically large scale. Therefore, DSs have always challenged system administrators [1–3]. A hardware malfunction, a faulty process, or an abnormal event may affect other events taking place at different locations in the running environment of a system. These problems can have bad effects on the performance and stability of the system, they can also cause errors to related processes and incorrect results of distributed applications. In order to ensure the effectiveness of DS operations, monitoring information in general and behaviors of each object in particular are the key issues in network management. Many technical solutions have been researched and developed to support administrators in monitoring systems, and have achieved certain results.

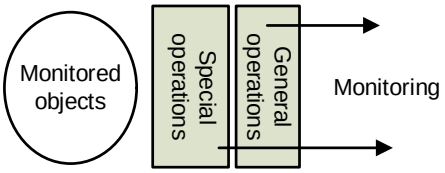

Figure 1. Monitoring groups in distributed systems.

Through the survey of some typical monitoring works [3–5], we are aware that there are many implementation solutions to deploy monitoring, including hardware, software, and hybrid solutions. However, with such advantages as flexibility, mobility, and ease of maintenance, software solutions have been widely deployed in monitoring products.

In addition, we find that the monitoring processes for DSs can be divided into two groups, as shown in Figure 1. *Specific monitoring* (SM) consists of monitoring systems that monitor specific issues of objects in a DS. Notable SM systems are MonALISA [6] and MOTEL [7]. It can be seen as a special layer to monitor details, such as traffic, performance and computing. *General monitoring* (GM) consists of monitoring systems that monitor general operations of objects in a DS, such as built-in tools of devices or utilities in the operating system (OS), and it can be seen as a common monitoring layer which provides abilities to monitor basic architectures and operations of monitored objects for system administrators.

Thus, monitoring solutions for GM are considered as high level monitoring facilities before using other monitoring solutions for SM to analyze the DS more deeply. Although general operations of monitored objects in DSs are critical issues in behavior monitoring, they are now primarily based on tools that are developed by device vendors or operating systems such as management commands provided by OS and device management tools. These built-in tools have some disadvantages, including discrete monitoring information and device independence. As a result,

the global problems cannot be solved and it is difficult for administrators to monitor group objects, such as networks and domains [3, 8]. In order to effectively deploy a behavior monitoring system for DSs, both modeling approaches and monitoring solutions for operations of monitored objects are important, so further research should be continued to develop them more effectively.

Motivated from the earlier research results in DSs, set theory and finite state machine theory [9–11], the objective of this paper is to model basic communication operations of objects in DSs by using the communicating finite state machine, and to present a monitoring solution that is suitable with the DS management architecture. Our proposed model consists of four monitoring levels: local objects, networks, domains and global systems. Therefore, administrators can monitor both local operations and communication operations of monitored objects in the systems, as well as special states or events, errors that occur in the systems. Doing so will actively support administration tasks. In order to demonstrate the feasibility of our proposed model, we design an Import Billing Data (IBD) system that can monitor file processing operations on Vietnam Mobile Telecom Services (VMS) network.

The rest of the paper is organized as follows. Section II presents related works. Section III introduces a behavior model based on the communicating finite state machine (CFSM) and a composition operation for many CFSMs. Section IV presents a behavior model for monitored objects in DSs, such as nodes, networks, domains and global distributed systems. Section V presents a monitoring solution for the behavior monitoring problem of DSs, including monitoring entities. Section VI gives implemented tools and experimental results. Finally, Section VII gives conclusions and future works.

II. Related Works

According to surveys of some typical monitoring and management systems [3, 6, 7, 12–14], most of monitoring systems are deployed to solve specific monitoring groups, such as parallel, distributed computing monitoring and performance monitoring. An advantage of these groups is that there are various monitoring requirements for each specific problem. However, a disadvantage is that most of these products operate independently, so they cannot integrate or inherit from each other. It can cause difficulty to administrators in operating and managing these products. In addition, the system performance will also be greatly affected when running concurrently with these products.

Simple Network Management Protocol (SNMP) is a standard protocol that is used to collect useful system information about network devices. SNMP has been widely deployed in most of TCP/IP network managements, system resources and traffic monitoring. SNMP uses a manager–agent model which contains three basic components: manager, agents and a management information base [13]. The communication between the manager and the agents can be implemented by two methods: polling (request–response) and trapping. The only weakness of the management system associated with this model is the manager.

Hofman *et al.* [14] proposed a DS monitoring approach with the ZM4/SIMPLE system. ZM4/SIMPLE is a hybrid monitoring solution for analyzing functional behaviors and evaluating performance of programs, by using a hardware monitor system (ZM4) and a software package (SIMPLE). The solution is already used for performance evaluation, optimization and debugging of parallel programs in parallel and distributed systems. It provides high monitoring performance because ZM4 is a dedicated hardware monitor. However, ZM4/SIMPLE monitors only events in multiprocessor systems and Local Area Networks (LANs).

Logean [7] proposed an approach for monitoring and testing communication services that are built on top of middleware with the MOTEL system. MOTEL consists of two modules: monitoring management and testing management. Monitored information is used to test the runtime behavior of the services. It is used in industry and the communication market. Since MOTEL is deployed in centralized models, its weakness is the monitoring server.

Joyce *et al.* [12] proposed an interactive monitoring approach with the Jade monitoring system that consists of a monitoring environment and tools. The monitoring tools support the observation and control of messages passing in a distributed application composed of a set of concurrently executing processes. Jade was designed to be extensible and separate tasks of detecting and collecting information from tasks of analyzing and displaying information. It consists of three main components: channel, controller and console. The monitoring system is conjuncted with the components and controlled by a single workstation, so monitoring is not flexible and it is difficult to deploy in large systems.

Newman *et al.* [6] proposed a performance monitoring approach based on the MonALISA framework. Communication interactions between monitoring agents of MonALISA are implemented by message-passing methods. MonALISA provides online monitoring services to support performance management and optimize grid computing systems and applications. The solution is used to monitor Wide Area Networks (WANs).

As mentioned above, the information about status, events and behaviors of the components in monitored objects (MOs) plays an important role in supporting administration tasks. For example, it allows administrators to obtain gen-

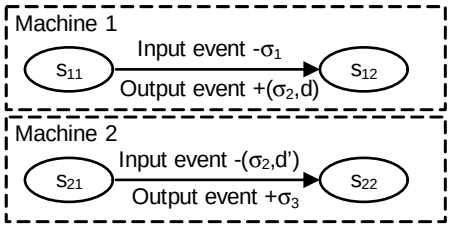

Figure 2. CFSM communication model.

eral information about operations of the entire system. This information is necessary for administrators, before they look into other detailed and specific information. However, the GM information is primarily based on specific integrated tools developed by device vendors or operating systems. These built-in tools not only provide discrete information on each component but also operate independently. Hence, they can neither connect the components in the system nor solve global problems, such as gathering monitored network information, monitored domain information and global system information. It takes a large amount of time to analyze objects in the inter-networks. Therefore, administrators may fail to effectively monitor the general operations of DSs with these tools. In order to overcome the above limitations, we propose a DS monitoring solution that is based on hierarchical monitoring entities, including: local object monitoring entities, network, domain, and global system monitoring entities. The solution will support administrators actively in monitoring the general operations of DSs.

III. Behavior Model

A behavior model that is used to present states and reactions of objects before/after received events and communicating finite state machines (CFSM) is considered suitable for modeling communication operations [15, 16]. In this model, state transitions of the state machines are triggered by input events. The output events are then associated with each transition, as shown in Figure 2.

The communication process of the CFSM occurs as follows. Machine 1 first receives an event σ_1 at time t. It then moves from state s_{11} to state s_{12} and emits toward Machine 2 another event σ_2 at time $t + d$, where d is delay of σ_2. Next, machine 2 receives the event σ_2 at time $t' = t + d + d'$, where d' is the link delay. Based on these communication operations, the CFSM can be expressed as

$$\mathrm{CFSM} = (\Sigma_{\mathrm{in}}, \Sigma_{\mathrm{out}}, S, \delta, s_0), \tag{1}$$

where Σ_{in} is a finite set of input events, Σ_{out} is a finite set of output events, S is a finite set of states, s_0 is the first state ($s_0 \in S$), and $\delta : S \times \Sigma_{\mathrm{in}} \rightarrow S \times \left(\Sigma_{\mathrm{out}} \times d\right)^*$ is a transition function; the superscript * denotes the set of output events, including NULL. We imply that the transition is associated with time delay, so we will ignore the variable d in expressions of the transition.

The set of all events of the state machine is given by $\Sigma_{\mathrm{es}} = \Sigma_{\mathrm{in}} \cup \Sigma_{\mathrm{out}}$. In order to determine a state and an event of δ, we use two projections PS and PE such that, for an input event, $\mathrm{PS}_{\mathrm{in}}(\delta) : S \times \Sigma_{\mathrm{in}} \rightarrow S$ and $\mathrm{PE}_{\mathrm{in}}(\delta) : S \times \Sigma_{\mathrm{in}} \rightarrow \Sigma_{\mathrm{in}}$, and for an output event, $\mathrm{PS}_{\mathrm{out}}(\delta) : S \times (\Sigma_{\mathrm{out}})^* \rightarrow S$ and $\mathrm{PE}_{\mathrm{out}}(\delta) : S \times \left(\Sigma_{\mathrm{out}}\right)^* \rightarrow \left(\Sigma_{\mathrm{out}}\right)^*$.

We can combine many CFSMs into a general composition CFSM by using an operation of parallel composition in [9]. Let CFSM_1 and CFSM_2 respectively be two state machines following the model in (1). Accordingly,

$$\mathrm{CFSM}_1 = (\Sigma_{\mathrm{in\_1}}, \Sigma_{\mathrm{out\_1}}, S, \delta_1, s_{0\_1}),$$
$$\mathrm{CFSM}_2 = (\Sigma_{\mathrm{in\_2}}, \Sigma_{\mathrm{out\_2}}, S, \delta_2, s_{0\_2}).$$

Then, the composition is expressed by

$$\mathrm{CFSM}_1 \,||\, \mathrm{CFSM}_2 = (\Sigma_{\mathrm{in}}, \Sigma_{\mathrm{out}}, S, \delta, s_0), \tag{2}$$

where $\Sigma_{\mathrm{in}} = \Sigma_{\mathrm{in\_1}} \cup \Sigma_{\mathrm{in\_2}}$, $\Sigma_{\mathrm{out}} = \Sigma_{\mathrm{out\_1}} \cup \Sigma_{\mathrm{out\_2}}$, $S = S_1 \times S_2$, $s_0 = (s_{0\_1}, s_{0\_2})$, and $\delta = \delta_1 \times \delta_2$. With $s_1 \in S_1, s_2 \in S_2$ and $\sigma \in \Sigma_{\mathrm{in}}$, we have $\delta : S_1 \times S_2 \times \Sigma_{\mathrm{in}} \rightarrow S_1 \times S_2 \times (\Sigma_{\mathrm{out}})^*$.

Let $\mathrm{TG}(s)$ be the set of all trigger events of the CFSM at state s. The transition function δ of the composition can be expressed as follows:

$$\delta\big((s_1, s_2), \sigma\big) = \begin{cases} \big(\delta_1(s_1, \sigma), \delta_2(s_2, \sigma)\big), & \text{if } \sigma \in \mathrm{TG}(s_1) \\ & \wedge \sigma \in \mathrm{TG}(s_2), \\ \big(\delta_1(s_1, \sigma), s_2\big), & \text{if } \sigma \in TG(s_1) \\ & \wedge \sigma \notin \mathrm{TG}(s_2), \\ \big(s_1, \delta_2(s_2, \sigma)\big), & \text{if } \sigma \notin \mathrm{TG}(s_1) \\ & \wedge \sigma \in \mathrm{TG}(s_2). \end{cases} \tag{3}$$

The state transition process $\delta = \delta\left((s_{11}, s_{21}), \sigma_{11}\right)$ was already described in Section III of [17] that uses the projections $\mathrm{PS}_{\mathrm{out}}$ and $\mathrm{PE}_{\mathrm{out}}$ with output events and output states to explain the interactive communication with two CFSMs.

In order to be clear of the composition operation in (2), we consider a model of interactive communication between two communicating state machines F_1 and F_2, initiated by an external trigger event σ_{11}, as shown in Figure 3, in which $\{s_{11}, s_{12}, \ldots\}$ is the state space of F_1, $\{s_{21}, s_{22}, \ldots\}$ is the state space of F_2, $\{\sigma_{21}, \ldots\}$ is the set of input events of F_2 receive from F_1, σ is the output event of F_2 to external side. Let (m, p) present a communication event, where m is a message, p is a communication port, and d be the time delay. Figure 3 presents the communication events with time delay $d_i > 0$ (delay of event and delay of link) and the composition result shows all states and events that are sent and received between F_1 and F_2.

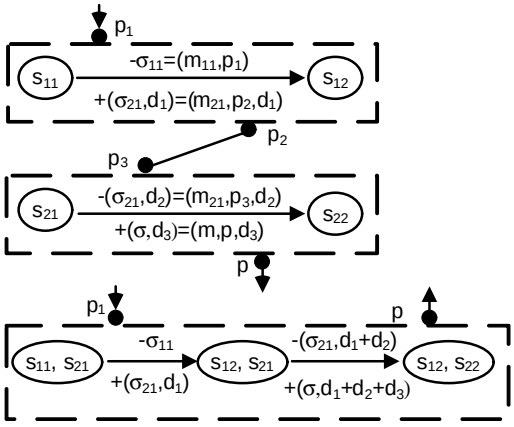

Figure 3. The composition with time delay.

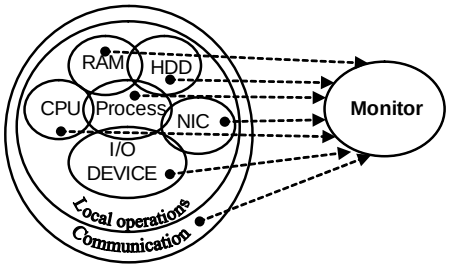

Figure 4. General operations of monitored objects.

IV. The Behavior Model for Monitored Objects in Distributed Systems

A DS consists of many heterogeneous devices: stations, servers, routers, etc. These devices communicate with each other in the DS and they are considered as MOs. Each MO consists of many hardware and software resources associated with information about their states and behaviors. The information can be divided into two parts: internal part including local operations which are internal, external part including communication operations, as shown in Figure 4.

Both local operations and communication operations are based on system resources of the MOs (CPU, RAM, IO device, etc.). These operations provide the corresponding system states and events such as hardware resources, software resources, and errors or anomalies which are critical for the system administrator's work. This section focuses on describing the CSFM-based behavior model for MOs.

1. Behavior Model for Monitored Objects

Behaviors for MOs in DSs are expressed by local and communication operations. Therefore, a behavior model of an MO contains a set of behavior models of the components of the MO (processes, CPU, etc.). In order to describe behaviors of components, we use the CFSM as shown in Figure 5.

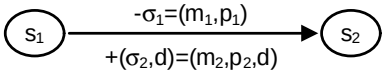

Figure 5. Behavior model for component.

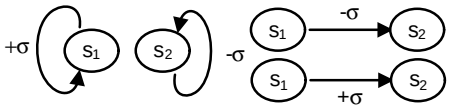

Figure 6. Some special cases of behavior model.

The behavior model presents the way events are received and emitted, and transition states belonging to the components. In some special cases, the components stay in their state as null transition or transit from state s_1 to state s_2 without emitting another event σ. We ignore these cases in our behavior model.

Since an MO consists of a set of basic components (Process, CPU, RAM, IO device, etc.) and related operations are controlled by the OS, its behaviors contain operations such as resource allocation and I/O operations. Therefore, the model for system operations of these components can be presented by using the CFSM as follows.

The behavior model for operations of processes is expressed as

$$F_{\text{Proc}} = (\Sigma_{\text{in\_Proc}}, \Sigma_{\text{out\_Proc}}, S_{\text{Proc}}, \delta_{\text{Proc}}, s_{0\_\text{Proc}}), \quad (4)$$

where $\Sigma_{\text{in\_Proc}}$, $\Sigma_{\text{out\_Proc}}$, S_{Proc}, δ_{Proc}, and s_{0_Proc} are similar to those in (1). This model is able to describe basic states and operations of the processes (communication events, running state, error state, etc.). We are interested in using F_{Proc} to describe behaviors of communication and monitoring processes between clients and servers in the IBD system, which will be presented in Section V.

Similarly, in the following, we have the behavior models for operations of the CPU, RAMs, IO devices, the HDD and the NIC, respectively:

$$F_{\text{CPU}} = (\Sigma_{\text{in\_CPU}}, \Sigma_{\text{out\_CPU}}, S_{\text{CPU}}, \delta_{\text{CPU}}, s_{0\_\text{CPU}}), \quad (5)$$

$$F_{\text{Mem}} = (\Sigma_{\text{in\_Mem}}, \Sigma_{\text{out\_Mem}}, S_{\text{Mem}}, \delta_{\text{Mem}}, s_{0\_\text{Mem}}), \quad (6)$$

$$F_{\text{IO}} = (\Sigma_{\text{in\_IO}}, \Sigma_{\text{out\_IO}}, S_{\text{IO}}, \delta_{\text{IO}}, s_{0\_\text{IO}}), \quad (7)$$

$$F_{\text{HDD}} = (\Sigma_{\text{in\_HDD}}, \Sigma_{\text{out\_HDD}}, S_{\text{HDD}}, \delta_{\text{HDD}}, s_{0\_\text{HDD}}), \quad (8)$$

$$F_{\text{NIC}} = (\Sigma_{\text{in\_NIC}}, \Sigma_{\text{out\_NIC}}, S_{\text{NIC}}, \delta_{\text{NIC}}, s_{0\_\text{NIC}}). \quad (9)$$

Hence, the behavior model of the MO, denoted by F_{MO}, is related to the set of state machines $\{F_{\text{Proc}}, F_{\text{CPU}}, F_{\text{Mem}}, F_{\text{IO}}, F_{\text{HDD}}, F_{\text{NIC}}\}$, and is thus obtained by a composition operation as follows:

$$\begin{aligned} F_{\text{MO}} &= F_{\text{Proc}} || F_{\text{CPU}} || F_{\text{Mem}} || F_{\text{IO}} || F_{\text{HDD}} || F_{\text{NIC}} \\ &= (\Sigma_{\text{in\_MO}}, \Sigma_{\text{out\_MO}}, S_{\text{MO}}, \delta_{\text{MO}}, s_{0\_\text{MO}}). \end{aligned} \quad (10)$$

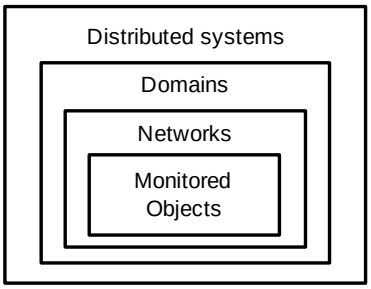

Figure 7. Group of monitored objects in distributed systems.

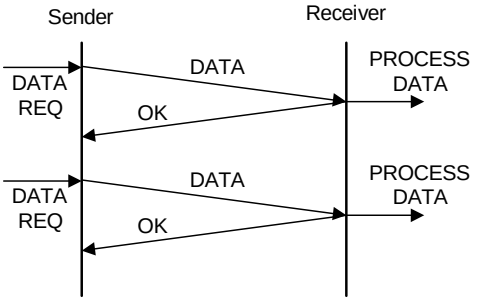

Figure 8. Simple data transmission protocol.

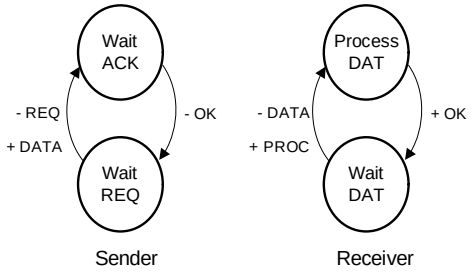

Figure 9. Behavior model for Sender-Receiver.

2. Behavior Model for Group Objects

According to results from earlier research on DSs and monitoring systems [3, 18], we can see that DSs consist of many heterogeneous objects and their topologies. In general, the topology of a DS is a hierarchical structure, which includes domains, networks and physical devices. The domains can communicate with each other by communication networks. Each domain is a hierarchical structure of many heterogeneous networks and devices. In each network, the domains can collaborate, exchange and share information with each other. In fact, this topology can be varied during operation of the system because of scalability and reconfiguration. The objects like domains, networks or the global DS are seen as group objects and can be presented as in Figure 7.

The group structure has been widely used in DS management and monitoring. The multi-level domain has been used to manage and monitor DSs, such as the Domain Name System and the distributed network management with multi-level domain [19]. Consequently, to deploy the behavior model for DSs, it is important to investigate the model for group objects, in addition to the model for MOs.

In the following, we will describe the behavior models for different group objects. First, consider a network MS which consists of k monitored objects $\{\mathrm{MO}_1, \mathrm{MO}_2, \ldots, \mathrm{MO}_k\}$. These objects are connected with each other and have communication operations over the network. Based on the previous behavior model for MOs, the behavior model for the network MS, denoted by F_{MS}, is a set of $\{F_{\mathrm{MO\_1}}, F_{\mathrm{MO\_2}}, \ldots, F_{\mathrm{MO}\_k}\}$ and is given by

$$\begin{aligned} F_{\mathrm{MS}} &= F_{\mathrm{MO\_1}} || F_{\mathrm{MO\_2}} || \ldots || F_{\mathrm{MO}\_k} \\ &= (\Sigma_{\mathrm{in\_MS}}, \Sigma_{\mathrm{out\_MS}}, S_{\mathrm{MS}}, \delta_{\mathrm{MS}}, s_{0\_\mathrm{MS}}) . \end{aligned} \tag{11}$$

Next, consider a domain that consists of m networks $\{\mathrm{MS}_1, \ldots, \mathrm{MS}_m\}$ and each network is a communicating state machine F_{MS}. The behavior model for the monitored domain (MD) F_{MD} is then a set of $\{F_{\mathrm{MS\_1}}, F_{\mathrm{MS\_2}}, \ldots, F_{\mathrm{MS}\_k}\}$ and is given by

$$\begin{aligned} F_{\mathrm{MD}} &= F_{\mathrm{MS\_1}} || F_{\mathrm{MS\_2}} || \cdots || F_{\mathrm{MS}\_m} \\ &= (\Sigma_{\mathrm{in\_MD}}, \Sigma_{\mathrm{out\_MD}}, S_{\mathrm{MD}}, \delta_{\mathrm{MD}}, s_{0\_\mathrm{MD}}) . \end{aligned} \tag{12}$$

Finally, consider a global DS which consists of a set of n domains $\{\mathrm{MD}_1, \mathrm{MD}_2 \ldots, \mathrm{MD}_n\}$ and each domain is a communicating state machine F_{MD}. The behavior model for this global DS F_{DS} is a set of $\{F_{\mathrm{MD\_1}}, F_{\mathrm{MD\_2}}, \ldots, F_{\mathrm{MD}\_n}\}$ and is given by

$$\begin{aligned} F_{\mathrm{DS}} &= F_{\mathrm{MD\_1}} || F_{\mathrm{MD\_2}} || \cdots || F_{\mathrm{MD}\_n} \\ &= (\Sigma_{\mathrm{in\_DS}}, \Sigma_{\mathrm{out\_DS}}, S_{\mathrm{DS}}, \delta_{\mathrm{DS}}, s_{0\_\mathrm{DS}}) . \end{aligned} \tag{13}$$

The composition result shows all the communicating events and states of machines in the interactive communication process. Consequently, the particular information about states and events of objects in the model can be collected based on components of this model to solve specific requirements for monitoring issues.

3. Sample for Behavior Model and Monitoring Entity

We consider a simple reliable data transmission protocol in which the receiver confirms to the sender that the data transmission process is ok, and the sender then continues to send data when it receives requests. The data exchange process between the two hosts can be illustrated in the Figure 8. The sender and the receiver are modeled by the two CFSMs, as shown in Figure 9.

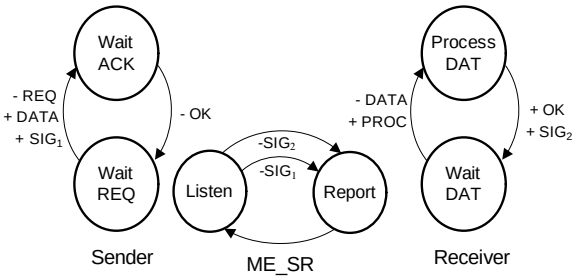

Figure 10. Monitoring entity for Sender-Receiver.

First, the sender and the receiver are initialized at the state *Wait REQ* (Wait for Request) and the state *Wait DAT* (Wait for Data), respectively. When the sender receives a data request (*-REQ*), data (*+DATA*) will be sent to the receiver. The sender will move to the next state, *Wait ACK*, to wait for feedback from the receiver. After receiving the data from the sender, the receiver will move to state *Process DAT*. If the data transmission process is ok, the receiver will send the event *OK* (*+OK*) to the sender and move to the first state (*Wait DAT*). When the event *OK* is received, the sender moves to the state *Wait REQ*. The above CFSM shows that the sender has to wait for an acknowledgment from the receiver before transmitting next data. The interactive communication process between the sender and the receiver will continue until the end of data processing.

Suppose that we want to consider some events in the communication process between two hosts, such as *REQ* and *OK*. A monitoring entity ME_SR is designed to detect *REQ* at the server side and event *OK* at the client side. ME_SR will make a progress report for these events. The state diagram of sender-receiver is updated, as shown in Figure 10.

In an extension of the model, we use two additional events for monitoring purposes; SIG_1 at the sender side and SIG_2 at the receiver side. ME_SR will have a state *Listen* that waits for new events to come in. With the input events SIG_1 and SIG_2, ME_SR moves to state *Report* that aims to make a monitoring report.

V. Monitoring Solution for Behavior of Distributed Systems

The objectives of monitoring systems are to observe, collect and inspect information about operations of hardware/software components and communication events of objects in DSs. This information actively supports system management activities.

The general architecture of monitoring systems can be divided into three parts [8]: Monitoring Entity (ME), Monitoring Application (MA) and MO, as shown in Figure 11.

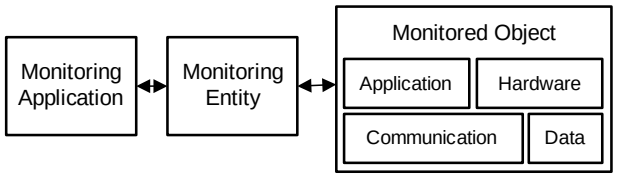

Figure 11. General monitoring architecture.

Algorithm 1: ME_INFO (generates behavior reports to MA)

Inputs: Object X with behavior model F_X, management application MA
Output: Monitoring reports based on the status and events of monitored object X
procedure ME_INFO(X, MA)
  if X does not exist in the system **then**
    Send "X does not exist" to MA
  else
    Extract states and events based on projections PS_{in}, PE_{in}, PS_{out} and PE_{out} in Section II.
    Generate monitoring reports.
    Send monitoring reports to MA.
  end if
end procedure

The ME is designed to instrument the MO. The information on instrumentation will be analyzed to generate the corresponding monitoring reports and be sent to the MA. The MA is designed to support management objects (i.e., administrators and other management agents). The MA interacts with the ME to generate monitoring requirements and present the monitoring results obtained from the ME.

In order to describe monitoring results of the ME, we use a procedure called ME_INFO(X, MA), where X is the monitored object and MA is the management application mentioned above. The procedure ME_INFO is set up as follows and summarized in Algorithm 1.

We can see clearly that a DS monitoring system consists of many MEs and MAs. They do not fix and operate independently on each domain of the DS. Monitoring information is exchanged between MEs and MAs by message passing. The design of MEs should follow the hierarchical architecture of the DS (see Figure 7). As a result, the monitoring system will be a set of behavior MEs {ME_MO, ME_MS, ME_MD, ME_DS} that can cooperate with each other in the monitoring system.

ME_MO should be installed on all MOs in the DS due to the fact that they not only observe and collect monitoring information of the MOs, but also provide monitoring reports to a network monitoring entity ME_MS. The ME_MS runs a composition operation in order to synthesize monitored

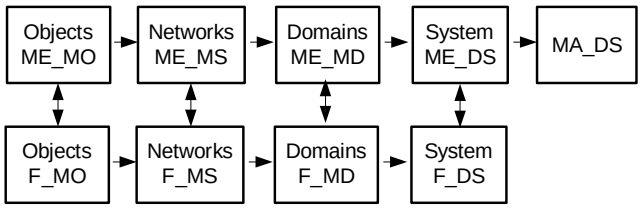

Figure 12. Architecture of monitoring entities.

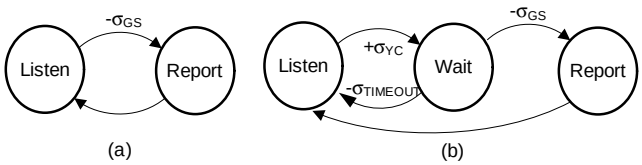

Figure 13. State machine of monitoring entities.

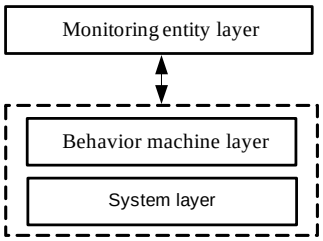

Figure 14. Solution for monitoring entity ME_MO.

information from ME_MO and provide monitoring reports to a domain monitoring entity ME_MD.

The operation of ME_MD and ME_MS have also run into similar processes. Behavior MEs are the state machines presented in Figure 13. Behavior information of MOs is received automatically as shown in Figure 13(a) and MEs send monitoring requests as shown in Figure 13(b).

In order to observe and collect states as well as events of an MO in DSs, we use a solution presented in Figure 14.

The system layer consists of the OS, drivers, system utilities, protocols, tools and interfaces for other monitoring systems, etc. This layer provides both local and communication operations, including states and events of the system such as hardware/software resources, errors or anomalies on MOs.

The behavior machine layer describes behaviors of MOs, including a set of basic monitored components $\{F_{\text{Proc}}, F_{\text{CPU}}, F_{\text{MEM}}, F_{\text{IO}}, F_{\text{HDD}}, F_{\text{NIC}}\}$. This layer provides technical basis to describe states, events of monitored components and behavior information about the MOs.

The monitoring entity layer consists of a set of monitoring state machines presented to basic monitored components. These machines collect directly operation information about the components (*e.g.*, Process, CPU, MEM, IO device, HDD, NIC) from the behavior machine layer. Besides, these machines can collect operation information

TABLE I
BASIC CHARACTERISTICS OF COMPONENTS

Num	Component	Monitored characteristics
1	Process	Basic status such as New, Running, Waiting, Terminated. Communication operations and resource requirements.
2	CPU	Status and CPU operations.
3	MEM	Status and MEM operations.
4	HDD	Status and HDD operations.
5	IO device	Status and IO operations.
6	NIC	Status and NIC operations.

of components indirectly from the system layer. We use protocols, Application Program Interfaces (APIs) and built-in tools of the OS, such as OS commands, the Window API, the Linux API and libraries. Popular protocols used in network management to monitor the status and traffic of MOs include the Internet Control Message Protocol (ICMP) and the SNMP. These tools are used to observe and collect system information as well as communication operations.

Since hardware and software resources of an MO are managed by the OS, the behavior information of basic components of the MO can be collected from the system layer. Therefore, the solution is suitable for behavior monitoring for MOs in DSs.

VI. IMPLEMENTED TOOL AND EXPERIMENT

1. Monitoring Operations of Process in DS

Operations of MOs in DSs are based on a set of basic components (*e.g.*, Process, CPU, RAM, IO device, NIC, HDD). Operation information of these components provides for system administrators essential information about the behaviors of the MOs. Basic characteristics of the monitored components are presented in Table I. The monitoring model for operations of the components share the same characteristics with the components, so we focus on a presentation for monitoring operations of processes in DSs only, monitoring operations of other components will be done similarly.

In this section, we use the previous behavior monitoring model to deploy the IBD system in which administrators are able to monitor both operations of login and import processes for billing data file. All states and events of these processes are displayed in the monitoring form of the IBD system. Administrators then quickly detect remote users, detailed operations of import processes as well as errors and error positions that occur during the billing data file import.

The IBD system is built on a *client–server* architecture. Clients send login requirements and data importing require-

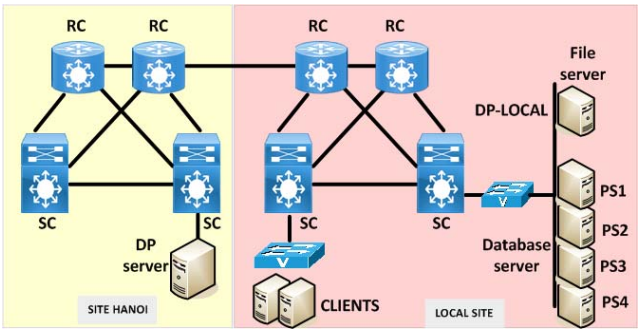

Figure 15. Network architecture of the IBD system.

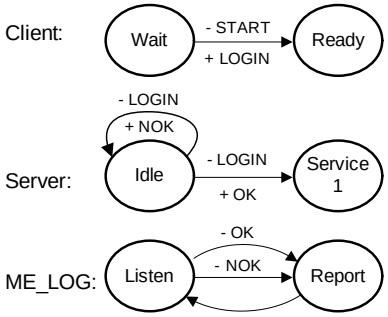

Figure 16. CFSM for login process.

ments, meanwhile the server provides many processes communicated with clients to run login and import service. The IBD system runs on a complex topology of VMS network.

All billing data files contain Data Printing server (DP) at Hanoi site – the headquarter of VMS network. Local networks of VMS (Danang, Nhatrang, Hochiminh, etc.) will copy the billing data files from the DP server to file server (DP–LOCAL). Local sites use core switches (SCs) and router switches (RCs) to communicate with Hanoi site. Users start the client service and the database server runs processes to import billing data into PS_1, PS_2, PS_3 and PS_4. In order to monitor login events, states and events of importing files, as well as errors, we use some basic CFSMs that are designed for IBD systems as follows.

In the login processes between clients and the server, clients will move from state *Wait* to state *Ready* when receiving a login requirement (*-START*) and then send the login event (*+LOGIN*) to the server. The server will move to state *Service* if the login process is successful or, otherwise, stay on its stage. The monitoring entity ME_{LOG} reports login events. Basic operations for the login process are presented in Figure 16.

In the billing data importing process, the clients start first at the state *Wait*. Then, they move to the state *Ready*, when receiving event *REQ_FI*, to import data file *FI* and emits event *IMP_FI* to the server. The server moves to the

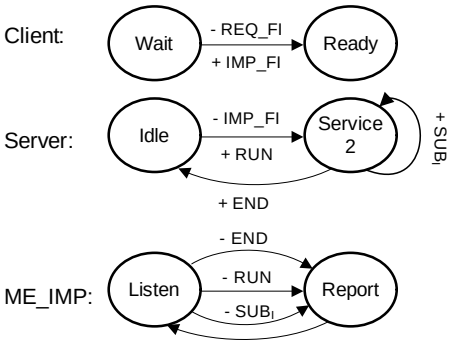

Figure 17. CFSM for data importing process Billing data importing process.

state *Service* when receiving *IMP_FI* from the clients and runs the import service. It emits event *RUN* to advertise the running state for the monitoring entity ME_IMP. The state *Service* of the server consists of many operations, such as file checking, start/stop, database connection. Each operation will emit event SUB_i. ME_IMP will receive these events from the server to make monitoring reports. The server emits event *END* when the service is done. The basic management for the import process is presented in Figure 17. Besides, some management operations are also deployed to support administrators such as to stop error processes, pause or restart processes.

Based on these basic models, the IBD system is deployed to monitor basic operations of billing data processing as well as login process on VMS network. Figure 18 illustrate the interface of the monitoring panel for the IBD system as an illustration. The bottom left part shows the Session list which includes login information (users, IP and time). The bottom right part displays on-going user behaviors (start events, stop events, etc.). The top part display on-going connection status, file import processing, errors, etc. In general, all details of the import process will be displayed in the monitoring panel. Administrators can then view all system operations and be able to quickly detect abnormal events and error states, which may occur while the system is operating.

In order to evaluate the performance of the IBD system, we used several Sun servers on VMS network with the same configuration. The dataset consists of 10 files. The experimental parameters are presented in Table II.

Figure 19 presents the processing time (in seconds) needed to import whole data files into the printing database server. It indicates that the processing time for whole data files varies significantly from using one printing server to using more servers (2, 3 and 4 servers). When we use many printing servers to import data files, the processing time for whole data files will significantly reduce as we can see the

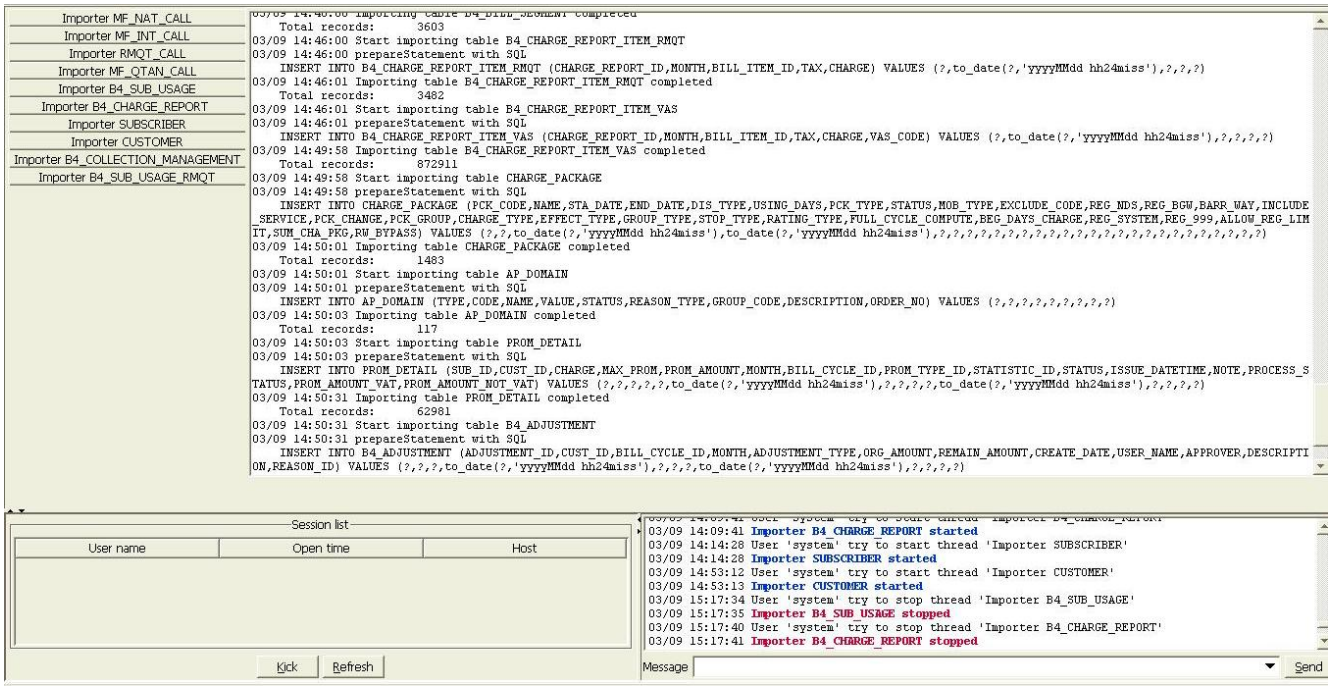

Figure 18. Interface of the monitoring panel for the IBD system.

TABLE II
BASIC CHARACTERISTICS OF COMPONENTS

Parameter	Value	Note
Printing servers	4	PS_1, PS_2, PS_3, PS_4
Billing data file	10	four data types
Clients	2	10.151.50.43, 10.151.50.45
Database server	4	Oracle DB

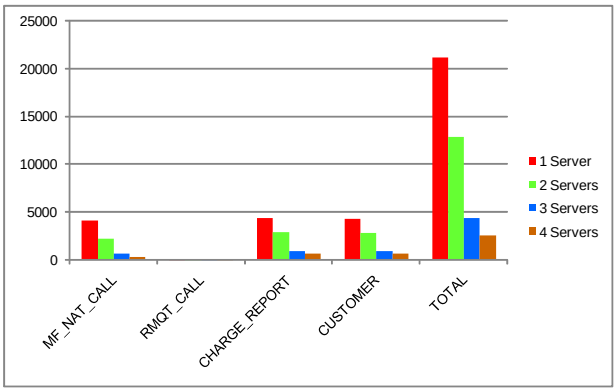

Figure 19. Processing time (in seconds) of IBD system.

TOTAL values in Figure 19, for the total running time. However, the processing time takes only a few seconds when the size of the data files is very small, and thus using either one or several printing servers does not make significant differences in the processing time, as we cannot see the RMQT_CALL values in the figure.

Experimental results show that the proposed behavior model can support administrators in actively monitoring DSs; administrators can quickly observe many important events or states of MOs. A 100 Mb Ethernet network will give 8127 Ethernet frames with the maximum frame size of 1538 bytes (including TCP and IP headers). Using the proposed behavior model can automatically detect events and states and quickly send management information (in milliseconds). On the contrary, with built-in tools such as OS commands, it would take administrators much time to implement activities such as remote access, authentication and command parameters. As a consequence, these tools will take much time for monitoring and cannot monitor some specific events such as detailed status, importing progress, error positions, or events of users. In a nutshell, we can see that the proposed behavior model for MOs in DSs is feasible. We can collect local operations as well as communication operations of MOs based on the suitable monitoring system.

2. On-line Monitoring for Interactive Operations between Objects

On-line monitoring for interactive operations between nodes and DSs are a big challenge for system administrators. It takes them much time to monitor these interactive communication operations, because they need special dis-

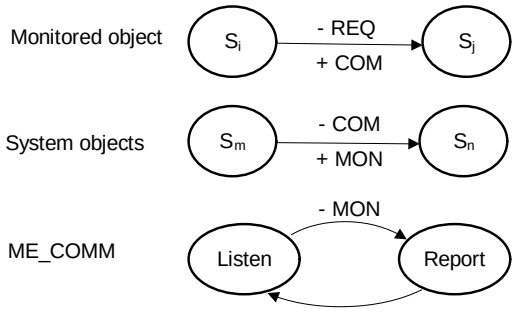

Figure 20. CFSM for interactive operations.

crete tools or offline data of security devices (firewalls, intrusion detection systems, etc.). However, they can automatically monitor interactive operations with the proposed behavior model as shown in Figure 20.

When system objects receive interactive requirements (*-COM*) from monitored objects, they will send the monitored event (*+MON*) to monitoring entity ME_COMM. This entity is deployed for all system objects to collect interactive information and to make monitoring report for this interactive operation. Each network, each domain and the global system will be monitored by their corresponding MEs. Based on this behavior model, we built an experiment to monitor interactive operations of nodes in VMS network. The interface of the monitoring panel for this experiment is illustrated in Figure 21.

The top part of the monitoring panel is the logical network diagram of VMS network. System objects in VMS network include physical devices and device groups, which are presented in domains and networks of the logical diagram. They can communicate with each other in the system. All interactive operations of MOs are then displayed in the bottom part of the monitoring panel. The information effectively supports system administrators in deciding what interaction operations are happening in the system, such as the interacted node or interacted system areas (domain and network) with the MO. The hierarchical monitoring solution is suitable for the DS architecture. Monitoring data are sent through the local network. On the contrary, the central monitoring solutions have a weak point at monitoring server. Monitoring data are sent through all domains of the DS.

VII. Conclusions and Future Works

The behavior model for MOs plays an important role in the development of monitoring solutions that provide system administrators with essential information about objects in DSs, such as local operations, communication operations, events, states, and errors. Based on this information, administrators will quickly detect special states or events, interactive operations between objects as well as errors and their positions that occur during operations of the system. In this paper, we have proposed a behavior model based on the CFSM that can describe basic operations of the MOs of DSs. In addition, we have proposed a hierarchical monitoring solution, consisting of four monitoring levels (object, network, domain and global system), that can support important information about operations of objects. This solution will support administrators to overcome the disadvantages of specific built-in tools in monitoring DSs.

In order to effectively deploy the proposed behavior monitoring solution for DSs, some studies follow. In general, application to large systems should be considered. In addition, a deeper analysis of states or events occurring in the DS is of interest. Moreover, using a dynamic management model and an effective communication model for MEs should be considered in order to optimize the behavior monitoring algorithms. Finally, for large-scale systems, it is of interest to use analysis techniques that help reduce computational complexity with respect to a large volume of monitoring information.

References

[1] A. D. Kshemkalyani and M. Singhal, *Distributed computing: principles, algorithms, and systems*. Cambridge University Press, 2008.

[2] G. Coulouris, J. Dollimore, T. Kindberg, and G. Blair, *Distributed Systems: Concepts and Design*, 5th ed. USA: Addison-Wesley Press, 2011.

[3] P. T. N. Hong and S. Le Van, "An Online Monitoring Solution for Complex Distributed Systems Based on Hierarchical Monitoring Agents," in *Fifth International Conference on Knowledge and Systems Engineering*, 2014, pp. 187–198.

[4] C. Guo, J. Zhu, and X.-L. Li, "A Generic Software Monitoring Model and Features Analysis," in *Second International Conference on Networks Security, Wireless Communications and Trusted Computing*, 2010, pp. 61–64.

[5] S.-Y. Yang and Y.-Y. Chang, "An active and intelligent network management system with ontology-based and multi-agent techniques," *Expert Systems with Applications*, vol. 38, no. 8, pp. 10 320–10 342, 2011.

[6] H. B. Newman, I. C. Legrand, P. Galvez, R. Voicu, and C. Cirstoiu, "MonALISA: A distributed monitoring service architecture," in *Proceedings of the Computing in High Energy and Nuclear Physics (CHEP)*, 2003, pp. 680–687.

[7] X. Logean, "Run-time monitoring and on-line testing of middleware based communication services," Ph.D. dissertation, Ecole Polytechnique Federale De Lausanne, 2000.

[8] P. T. N. Hong and S. Le Van, "A Monitoring Model for Hierarchical Architecture of Distributed Systems," *International Journal of Advanced Computer Science and Applications (IJACSA)*, vol. 6, no. 1, pp. 54–62, 2015.

[9] C. G. Cassandras and S. Lafortune, *Introduction to discrete event systems*, 2nd ed. Springer US, 2008.

[10] G. A. Wainer and P. J. Mosterman, *Discrete-event modeling and simulation: theory and applications*. CRC Press, 2016.

[11] W. Hu and H. S. Sarjoughian, "A co-design modeling approach for computer network systems," in *Winter Simulation Conference*, Dec 2007, pp. 685–693.

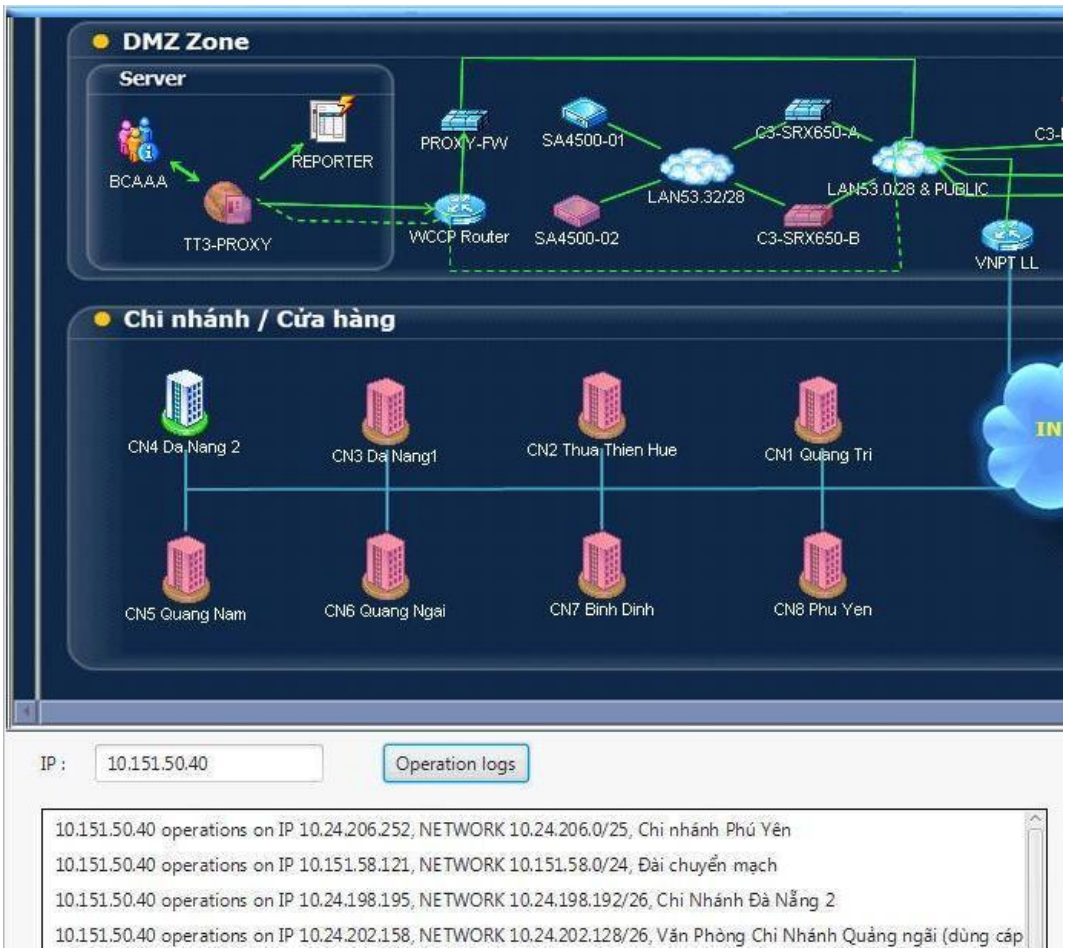

Figure 21. Interface of the monitoring panel for interactive operations in VMS network.

[12] J. Joyce, G. Lomow, K. Slind, and B. Unger, "Monitoring Distributed Systems," *ACM Transactions on Computer Systems*, vol. 5, no. 2, pp. 121–150, Mar. 1987.

[13] Nguyen Thuc Hai, *Mạng máy tính và các hệ thống mở*. Hanoi, Vietnam: Vietnam Education Publishing House, 1997.

[14] R. Hofmann, R. Klar, B. Mohr, A. Quick, and M. Siegle, "Distributed performance monitoring: methods, tools, and applications," *IEEE Transactions on Parallel and Distributed Systems*, vol. 5, no. 6, pp. 585–598, Jun 1994.

[15] G. Holzmann, *Design and Validation of Computer Protocols*. New Jersey, USA: Prentice Hall, 1991.

[16] Y. Pencolé, M.-O. Cordier, and L. Rozé, "A decentralized model-based diagnostic tool for complex systems," *International Journal on Artificial Intelligence Tools*, vol. 11, no. 03, pp. 327–346, 2002.

[17] P. T. N. Hong and S. Le Van, "The Basic Behavior Modeling for Monitored Objects in Distributed Systems by Using Communicating Finite State Machine," *Int. J. Comp. Net. & Wireless Com.*, vol. 4, no. 6, pp. 380–386, 2014.

[18] S. Le Van and P. T. N. Hong, "Developing the supporting system for monitoring the TCP/IP network based on ICMP and SNMP," *Journal of Information and Communication Technology*, pp. 41–47, 2004.

[19] K.-H. Lee, "A distributed network management system with multi-level domain approach," in *Singapore ICCS '94. Conference Proceedings.*, vol. 2, Nov 1994, pp. 789–793.

Phuc Tran Nguyen Hong received his B.Eng. in Information Technology and M.Sc. in Computer Science from Danang University of technology in 1997 and 2003, respectively. Currently, he is a Ph.D. candidate at the Faculty of Information Technology, Danang University of Technology. His research interests include distributed systems, system monitoring and communication networks.

Son Le Van received his Ph.D. from Donetsk Technical University in 1997. He is an associate professor in the Department of Computer Science, Danang University of Education. His main research interests include operating systems, distributed systems, monitoring solutions, communication networks, cloud computing.