The Basic Behavior Modeling for Monitored Objects in Distributed Systems by Using Communicating Finite State Machine

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 while the system is operating, so administrators must have effective monitoring tools for network management to ensure stability and performance of distributed systems. The monitoring for behaviors of objects in the system is essential to support administrators in detecting abnormal states or events quickly as well as errors' positions that occur in the system. In order to deploy the behavioral monitoring system effectively, the modeling for behaviors of monitored objects in the system is an important issue which uses to develop algorithms for the solution. This modeling needs to be more researched and developed appropriately in behavioral monitoring issue for monitored objects in distributed systems. In this paper, we propose a methodology to model the basic behaviors for monitored objects in distributed systems by using the communicating finite state machine. Based on this model, we can design a monitoring solution that monitors activities of objects in distributed systems and will effectively support administrators in system operating, diagnosing and controlling communication behaviors of complex distributed systems.**

Keywords-behavior; communicating finite state machine; distributed systems; model; monitoring

I. INTRODUCTION

Distributed systems (DS) are complex systems, which have always challenged for system administrator a lot [1,4,7]. Monitoring and controlling information of the network in general and activity status of each object (device) in particular is the issues of primary attention in network management. 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 [10,11,12,13,14,15], we are aware that there are many implementation solutions to deploy monitoring such as hardware, software and hybrid solution. However, with the advantages such as flexibility and mobility, the ease of maintenance, etc, so the software solution has been widely deployed in many TCP/IP monitoring products [7].

We also 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 traffic, performance, computing,... SM can be seen as a special monitoring layer and most of these solutions in SM have not yet been really interested in the system operations of monitored objects in DS.
- GM consists of monitoring systems that monitor general operations of the monitored objects in DS such as built-in tools of devices or ultilities in OS. GM can be seen as a common monitoring layer in which provide abilities to monitor architectures, operations and behaviors of monitored objects (MO) in DS such as configuration, status, behavior communication, etc

The GM is considered as a high level monitoring facilities to monitored DS before using orther monitoring solutions in SM to deeper analysis for DS. The behaviors of objects in the system are critical issues in solution of GM to support administrators in detecting abnormal states or events quickly as well as errors' positions that occur in the system. Some solutions have been supporting in behavior diagnosing and monitoring such as MOTEL [15], a decentralized model-based diagnostic [9],... Although system operations of monitored objects in DS are critical issues in behavior monitoring, they have not yet been really interested in most of the monitoring system. In order to effectively deploy the behavioral monitoring system for DS, the modeling approach for behaviors of monitored objects in the system is an important issue and should be continued to research and develop more effective. The goal of the paper propose a methodology to model the basic behaviors for the communication operations of objects in distributed systems by using the communicating finite state machine.

When monitored systems have basic changes about architectures, behaviors, activity environments, the technical solutions must be modified and updated appropriately for new changes and management requirements. With system specification methodology is general and flexible, the modeling approach is considered more appropriate for systems that have a lot of changes, this one is widely used in discrete event systems, computer protocols [2,5,9]. The modeling approach has also achieved some certain results in queue management [2], distributed applications [3], simulation for computer operations [8], etc. Some mathematical theories are used in the system modeling such as Petri Net [3,5], FSM (Finite State Machine) [5], etc. However, FSM is more commonly used in presenting events, states and state transitions for large-scale systems [2,5]. Therefore, we continue to research on using the communicating finite state machines to model for the communication operations of objects in DS.

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 components of hardware and software resources, and these ones are associated with information about the corresponding states and behaviors. This information can be divided into two basic parts: internal part – local operations and an external part – communication operations as Figure 1 [7].

- Local operations include processing, computing, resource requirements for process computations. These operations are locally performed within that object.
- Communication operations are used to communicate with other objects on the system such as interaction with management system, inter-process communication.

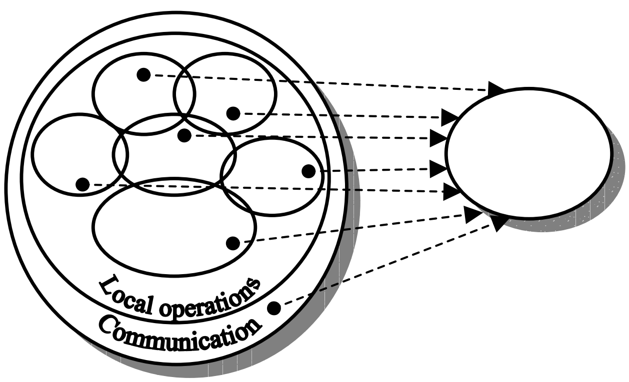


Figure 1. General operations of the monitored object

In order to deploy the monitoring system effectively, the modeling for monitored objects (*MO*) in the system is really necessary, we need to continue research and develop a behavior model of these objects appropriately. The objective of the paper is based on the research results on set theory and finite state machine theory [2,3], we focus on developing a formal model for the communication operations of objects in DS. With this model, we can describe the local operations and communication operations that are called activities between objects in DS, as well as this model can support us in developing a behavioral monitoring solution that is suitable for architecture of DS.

II. BEHAVIOR MODEL

Behavior model presents the states and the reactions of objects before/after the received events, communicating finite state machines (CFSM) model is considered suitable for modeling the communication activities (send/receive) [5,9]. In this model, state transitions of the state machines are triggered by the input event and associate the output event with each transition as shown in Figure 2.

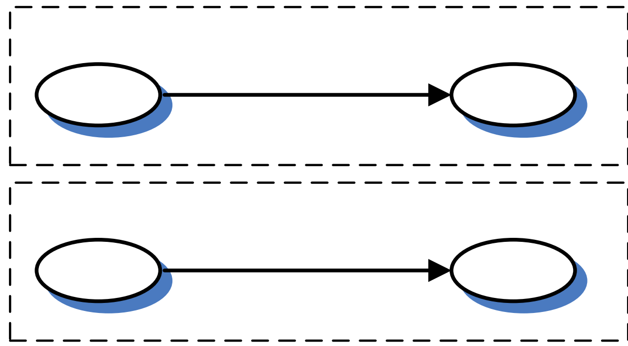


Figure 2. Communication model of CFSM

When Machine 1 receives event σ_1 at time t, it moves from state s_{11} to s_{12} and emits to Machine 2 event σ_2 at time t+d (d is delay of σ_2), Machine 2 receives event σ_2 at time t'=t+d+d' (d' is delay of link). Based on these communication activities, the communicating finite state machine can be expressed as follows:

$$CFSM = \left(\Sigma_{in}, \Sigma_{out}, S, \delta, s_0\right) \qquad (1)$$

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 ($s_0 \in S$); δ is transition function, $\delta : S \times \Sigma_{in} \to S \times \left(\Sigma_{out} \times D\right)^*$ (D is delay time and * denotes set of output events, including null output).

Set of all events of state machine $\Sigma_{es} = \Sigma_{in} \cup \Sigma_{out}$, in order to determine the state and event of δ, we use two projections PS and PE:

- Input event:

  $PS_{in}: S \times \Sigma_{in} \to S$ and $PE_{in}: S \times \Sigma_{in} \to \Sigma_{in}$
- Output event:

  $PS_{out}: S \times (\Sigma_{out})^* \to S$ and $PE_{out}: S \times (\Sigma_{out})^* \to (\Sigma_{out})^*$

We can combine many CFSM into a composition CFSM by using the parallel composition operation [2]. Let $CFSM_1$, $CFSM_2$ be state machines as expression in (1), the result of composition is expressed as follows:

$$CFSM = CFSM_1 \parallel CFSM_2 = (\Sigma_{in}, \Sigma_{out}, S, \delta, s_0) \quad (2)$$

Where: $\Sigma_{in} = \Sigma_{in\_1} \cup \Sigma_{in\_2}$ (set of input events of machine 1 and 2); $\Sigma_{out} = \Sigma_{out\_1} \cup \Sigma_{out\_2}$ (set of output events of machine 1 and 2); $S = S_1 \times S_2$ (set of states of machine 1 and 2); $s_0 = (s_{0\_1}, s_{0\_2})$ (first states of machine 1 and 2).

With $s_1 \in S_1$, $s_2 \in S_2$ and $\sigma \in \Sigma_{in}$

$\delta = \delta_1 \times \delta_2 = S_1 \times S_2 \times \Sigma_{in} \rightarrow S_1 \times S_2 \times (\Sigma_{out})^*$

Let k_h(s) be the set of all trigger events of CFSM at state s, the transition function δ can be expressed as follows:

$$\delta((s_1,s_2),\sigma) = \begin{cases} (\delta_1(s_1,\sigma),\delta_2(s_2,\sigma)) & if \quad \sigma \in k\_h(s_1) \wedge \sigma \in k\_h(s_2) \\ (\delta_1(s_1,\sigma),s_2) & if \quad \sigma \in k\_h(s_1) \wedge \sigma \notin k\_h(s_2) \\ (s_1,\delta_2(s_2,\sigma)) & if \quad \sigma \notin k\_h(s_1) \wedge \sigma \in k\_h(s_2) \end{cases} \quad (3)$$

Event space Σ_{es} consists of internal events $\Sigma_{internal}$ and external communication events $\Sigma_{external}$ of objects, so these ones can be divided into parts such as set of internal emit events Σ_{e_int}, set of external emit events Σ_{e_ext}, set of internal receive events Σ_{r_int}, set of external receive events Σ_{r_ext}. Σes is expressed as follows:

$$\Sigma_{es} = \Sigma_{out} \cup \Sigma_{in} = (\Sigma_{e\_int} \cup \Sigma_{e\_ext}) \cup (\Sigma_{r\_int} \cup \Sigma_{r\_ext}) \quad (4)$$

In order to understand the transition function δ as an expression in (3) clearly, we consider the model of interactive communication between two communicating state machines F_1 and F_2 with two cases (delay communication or no delay communication) in next section.

III. THE INTERACTIVE COMMUNICATION BETWEEN TWO CFSMS

Communication events between two communicating state machines F_1 and F_2 are initiated by an external trigger event σ_{11} and shown in Figure 3, in which $\{s_{11}, ..., s_{1n}\} \in S_1$ is state space of F_1; $\{s_{21}, ..., s_{2m}\} \in S_2$ is state space of F_2; $\{\sigma_{12}, ..., \sigma_{1i}, ...\} \in \Sigma_{r\_int\_1}$ is set of input events of F_1 receive from F_2; $\{\sigma_{21}, ..., \sigma_{2j}, ...\} \in \Sigma_{r\_int\_2}$ is set of input events of F_2 receive from F_1.

According to projections PS_{out}, PE_{out} are described in section II, the state transition process with function $\delta = \delta((s_{11}, s_{21}), \sigma_{11})$ can be expressed in next section.

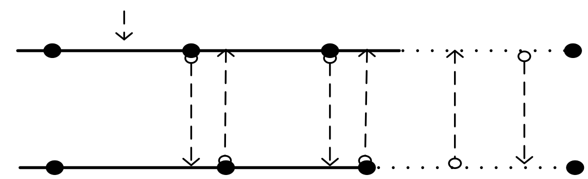


Figure 3. The interactive communication between F1 and F2

A. *Communication Consists of Internal Events and External Events with no Delay*

There is no delay in the communication (d=0) and the communication processes can be divided into three parts:

1) F_1 receives σ_{11} ($\sigma_{11} \in \Sigma_{r\_ext\_1}$), F_1 runs state transition δ and emits output event to F_2

$\delta = \delta((s_{11}, s_{21}), \sigma_{11}) = (PS_{out}(\delta_1(s_{11}, \sigma_{11})), PE_{out}(\delta_1(s_{11}, \sigma_{11})), s_{21})$

Where: $PE_{out}(\delta_1(s_{11}, \sigma_{11})) \subseteq \Sigma_{e\_int\_1}$ and $PE_{out}(\delta_1(s_{11}, \sigma_{11})) \subseteq \Sigma_{r\_int\_2}$

F_2 receives event $PE_{out}(\delta_1(s_{11}, \sigma_{11}))$ instantly, so $\delta = \delta((PS_{out}(\delta_1(s_{11}, \sigma_{11})), s_{21}), PE_{out}(\delta_1(s_{11}, \sigma_{11}))) = (PS_{out}(\delta_1(s_{11}, \sigma_{11}), PS_{out}(\delta_2(s_{21}, PE_{out}(\delta_1(s_{11}, \sigma_{11})))), PE_{out}(\delta_2(s_{21}, PE_{out}(\delta_1(s_{11}, \sigma_{11})))))$

Where: $PE_{out}(\delta_2(s_{21}, PE_{out}(\delta_1(s_{11}, \sigma_{11})))) = \{\sigma_{12}\} \subseteq \Sigma_{e\_int\_2}$ and $PE_{out}(\delta_2(s_{21}, PE_{out}(\delta_1(s_{11}, \sigma_{11})))) = \{\sigma_{12}\} \subseteq \Sigma_{r\_int\_1}$

2) F_1 receives σ_{12} ($\sigma_{12} \in \Sigma_{r\_ext\_1}$) from F_2 and this communication continues between F_1 and F_2

$\delta = \delta((s_{1i}, s_{2j}), \sigma_{1i})$ with i=2..n, the result of transition δ:

$PS_{out}(\delta_1(s_{1i}, \sigma_{1i})) = s_{1(i+1)}$, because $PE_{out}(\delta_1(s_{1i}, \sigma_{1i})) \cap \Sigma_{e\_int\_1} \subseteq \Sigma_{r\_int\_2}$ so F_2 run δ_2.

Transition δ with any event σ_{2j} ($\sigma_{2j} \in \Sigma_{r\_int\_2}$) is similar to the previous case σ_{1i}, $\delta = \delta((PS_{out}(\delta_1(s_{1i}, \sigma_{1i})), s_{2j}), PE_{out}(\delta_1(s_{1i}, \sigma_{1i})))$ with j=2..m, the result of transition δ:

$PS_{out}(\delta_2(s_{2j}, \sigma_{2j})) = s_{2(j+1)}$, $PE_{out}(\delta_2(s_{2j}, \sigma_{2j})) \cap \Sigma_{e\_int\_2} = \{\sigma_{1(j+1)}\}$, $\{\sigma_{1(j+1)}\} \subseteq \Sigma_{r\_int\_1}$

3) End of communication between F_1 and F_2 with two cases

a) m=n: the end of communication is at F_2 (F_2 emits σ with $\sigma \notin \Sigma_{in\_1}$ or $\sigma = \emptyset$)

Composition state machine get finish state (s_{1n}, s_{2n}) and event $PE_{out}(\delta_2(s_{2n-1}, \sigma_{2n-1}))$, with $PE_{out}(\delta_2(s_{2n-1}, \sigma_{2n-1})) \cap \Sigma_{out\_2} = \emptyset$ (no event) or $P_{Eout}(\delta_2(s_{2n-1}, \sigma_{2n-1})) \cap \Sigma_{out\_2} = \{\sigma\} \notin \Sigma_{in\_1}$

b) m<n: the end of communication is at F_1 (F_1 emits σ with $\sigma \notin \Sigma_{in\_2}$ or $\sigma = \emptyset$)

Composition state machine get finish state (s_{1n}, s_{2m}) and event $PE_{out}(\delta_1(s_{1m}, \sigma_{1m}))$, with $PE_{out}(\delta_1(s_{1m}, \sigma_{1m})) \cap \Sigma_{out\_1}=\varnothing$ (no event) or $PE_{out}(\delta_1(s_{1m}, \sigma_{1m})) \cap \Sigma_{out\_1}=\{\sigma\} \notin \Sigma_{in\_2}$

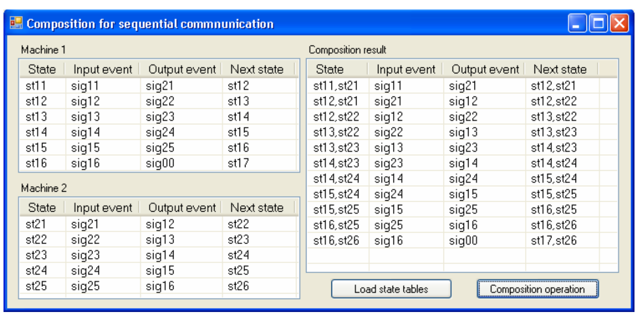
B. Communication Consists of Internal Events and External Events with Delay

Similar to the previous section, result of composition state machine get all of the events and states that are described by transition function δ with time delay d>0 between events of F_1 and F_2.

IV. Experimental Results for CFSM and the Behavior Model for Monitored Objects in DS

A. Experimental Results for CFSM

In order to present the behavior of objects, behavior model and behavior composition in DS, we use a state table as in Fig. 4 to describe all of the transition rules that fully describe the relationship between the states and the corresponding events (eg: state st11, input event sig11, output event sig21 and transition state st12 on the left side of form). Based on these state tables, we develop experimental forms that are able to compose of the communication behaviors of state machines. The composition results are shown in Figure 4 and Figure 5 in which consist of the interactive communication model and general behavior model (sync and asynchronous transitions) with delay case.


Composition for sequential commnunication

Machine 1

State	Input event	Output event	Next state
st11	sig11	sig21	st12
st12	sig12	sig22	st13
st13	sig13	sig23	st14
st14	sig14	sig24	st15
st15	sig15	sig25	st16
st16	sig16	sig00	st17

Machine 2

State	Input event	Output event	Next state
st21	sig21	sig12	st22
st22	sig22	sig13	st23
st23	sig23	sig14	st24
st24	sig24	sig15	st25
st25	sig25	sig16	st26

Composition result

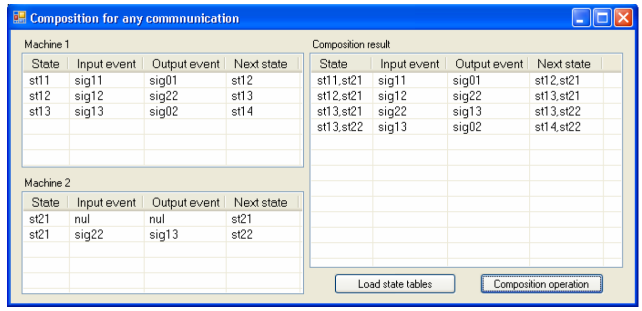
State	Input event	Output event	Next state
st11,st21	sig11	sig21	st12,st21
st12,st21	sig21	sig12	st12,st22
st12,st22	sig12	sig22	st13,st22
st13,st22	sig22	sig13	st13,st23
st13,st23	sig13	sig23	st14,st23
st14,st23	sig23	sig14	st14,st24
st14,st24	sig14	sig24	st15,st24
st15,st24	sig24	sig15	st15,st25
st15,st25	sig15	sig25	st16,st25
st16,st25	sig25	sig16	st16,st26
st16,st26	sig16	sig00	st17,st26

Load state tables | Composition operation

Figure 4. Composition result of two interactive communication machines

Figure 4 descibes a interactive communication process in which communication events between machine 1 and machine 2 are done consecutively. Machine 1 and 2 are on left of the form and composition result is on right of the form. The end of communication process is at machine 1 with output event sig00. The composition result shows all of the communicating events, states of two machines in the interactive communication process.

Figure 5 descibes a general communication process in which communication events consit of interactive communication events between machine 1 and machine 2 such as sig22, sig13 and distinctive events of each machine such as sig01, sig02,... we use nul event to describe that machine 2 is not transitioned it's state while machine 1 run state transition with input event . The end of communication process is at machine 1 with output event sig02. The composition result shows that common events between two behavior models are synchronized respectively and state of each model is not affected by distinctive events of each one.


Composition for any commnunication

Machine 1

State	Input event	Output event	Next state
st11	sig11	sig01	st12
st12	sig12	sig22	st13
st13	sig13	sig02	st14

Machine 2

State	Input event	Output event	Next state
st21	nul	nul	st21
st21	sig22	sig13	st22

Composition result

State	Input event	Output event	Next state
st11,st21	sig11	sig01	st12,st21
st12,st21	sig12	sig22	st13,st21
st13,st21	sig22	sig13	st13,st22
st13,st22	sig13	sig02	st14,st22

Load state tables | Composition operation

Figure 5. Composition result of two general behavior models

From the expression 1÷4 and the results in figure 4÷5 show that the application of CFSM to model for the behaviors of one or many monitored objects in DS is feasible and this model is called the behavior model of monitored objects. Furthermore, we can composite many behavior models of monitored objects into a composition model of monitored system.

The communication behavior of the monitored objects (MO) in general can be presented as figure 6, the communication events are expressed by (m, p), where m: the message sends or receives; port p sends or receives respectively.

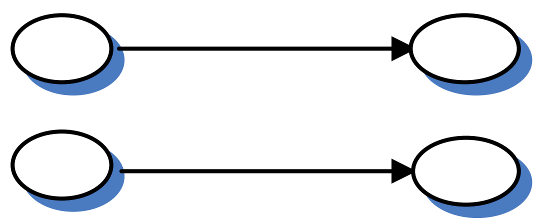


Figure 6. The communication behavior between MOs

C_1, C_2: monitored objects; σ_{11}: input events of C_1; σ_{21}: the communication event between C_1 and C_2; σ_{22}: output events of C_2; m_{11}, $m_{21,}$ $m_{22:}$ the messages contain the relative information of events; $p_{1\div}p_4$: communication ports. With this design it is clearly that we will be able to use CFSM in (1) to model for behaviors of C_1 and C_2.

B. Behavior Model for Monitored Objects in DS

From result of research on DS and monitoring systems in papers [6,7], we can see that DS consists of many heterogeneous objects and topologies. However, topology of DS in general can be showed as a hierarchical structure consists of domains and physical devices which can collaborate, exchange and share information to each other. In fact, this topology is variable during operation of the system due to scalability and reconfiguration. With point of view the domain-based management for large scale systems, the multi-level domain has been used to manage for DS [17], in which consists of local object level and domain level. The hierarchical architecture of monitored objects in DS can be presented as Figure 7.

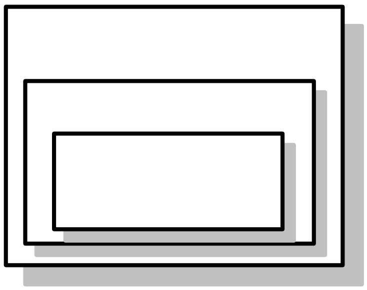


Figure 7. The hierarchical architecture of monitored objects in DS

Therefore, in order to deploy the behavior modeling for DS in general, the MO of DS, domain and global DS are objects that are focused on investigating to model.

As in Figure 1, *MO* is a set of components {Proc, CPU, Mem, IOdev, Comm, …} and the related operations are done by operating system in which consists of operations such as resource location, I/O operations, etc. We model for each of the component related operations corresponds to a communicating state machines as follows:

- The behavior model for Process related operation (F_Proc)

$$F\_Proc=(\Sigma_{in\_Proc}, \Sigma_{out\_Proc}, S_{Proc}, \delta_{Proc}, s_{0\_Proc}) \quad (5)$$

- The behavior model for CPU related operation (F_CPU)

$$F\_CPU=(\Sigma_{in\_CPU}, \Sigma_{out\_CPU}, S_{CPU}, \delta_{CPU}, s_{0\_CPU}) \quad (6)$$

- The behavior model of Mem related operation (F_Mem)

$$F\_Mem=(\Sigma_{in\_Mem}, \Sigma_{out\_Mem}, S_{Mem}, \delta_{Mem}, s_{0\_Mem}) \quad (7)$$

- The behavior model of IOdev related operation (F_IOdev)

$$F\_IOdev=(\Sigma_{in\_IOdev}, \Sigma_{out\_IOdev}, S_{IOdev}, \delta_{IOdev}, s_{0\_IOdev}) \quad (8)$$

- The behavior model of Comm related operation (F_Comm)

$$F\_Comm=(\Sigma_{in\_Comm}, \Sigma_{out\_Comm}, S_{Comm}, \delta_{Comm}, s_{0\_Comm}) \quad (9)$$

Hence the behavior model of MO in DS (*F_MO*) corresponds to set of communicating state machines *{F_Proc, F_Cpu, F_Mem, F_IOdev, F_Comm,…}*:

$$F\_MO = F\_Proc \,||F\_CPU \,||\, F\_Mem \,||\ldots = (\Sigma_{in\_MO}, \Sigma_{out\_MO}, S_{MO}, \delta_{MO}, s_{0\_MO}) \quad (10)$$

Network domain consists of a set of $\{MO_1, MO_2,\ldots,MO_n\}$ that corresponds to set of communicating state machines $\{F\_MO_1, F\_MO_2,\ldots, F\_MO_n\}$ in this domain. Similar to *F_MO*, The behavior model of network domain (*F_MD*) is expressed as follows:

$$F\_MD = F\_MO_1 \,||F\_MO_2 \,||\ldots ||F\_MO_n = (\Sigma_{in\_MD}, \Sigma_{out\_MD}, S_{MD}, \delta_{MD}, s_{0\_MD}) \quad (11)$$

DS consists of a set of $\{MD_1, MD_2,\ldots,MD_m\}$ that corresponds to set of communicating state machines $\{F\_MD_1, F\_MD_2,\ldots, F\_MD_m\}$ in this system. *The behavior model of global DS (F_DS)* is expressed as follows:

$$F\_DS = F\_MD_1 \,||F\_MD_2 \,||\ldots ||F\_MD_m = (\Sigma_{in\_DS}, \Sigma_{out\_DS}, S_{DS}, \delta_{DS}, s_{0\_DS}) \quad (12)$$

Models in expression (10)÷(12) present the behaviors of the MO, domain and global DS which based on the parallel composition of the basic object models respectively. Base on input events (Σ_{in}), output events (Σ_{out}) and transition function (δ) of the model, the particular information about states or events of objects in the model can be collected to solve for special requirements of issues such as monitoring, controlling and diagnosing. For example, give an application client-server which has state diagram as shown in Figure 8

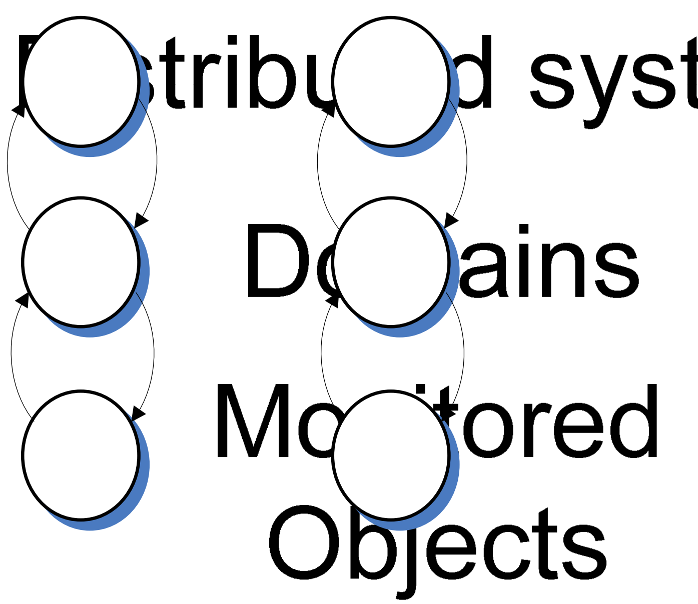


Figure 8. Basic operation model of application client-server

First time, server and client process are started, and server state is Idle, client state is Ready. When client has requirements, client sends REQ to server and move to Wait state. Once REQ is received, server will transition to the Service state to run services, server send FIN to client after server finish services and return to the Idle state. After receiving the FIN, client will return to Ready state.

Suppose that server process detect any error, it transition to Error state and send ERR to client. After recovring error, server will return to Idle state and send OKE to client. On the other hand, once ERR comes in, client will transition to Warn state. Once OKE comes in, client will transition to Ready state.

We suppose that a monitoring entity ME_CL need to detect event REQ on server side, event ERR and OKE on client side, and ME_CL will create monitoring report for these events. In order to do that, the state diagram of application client-server is updated as shown in Figure 9.

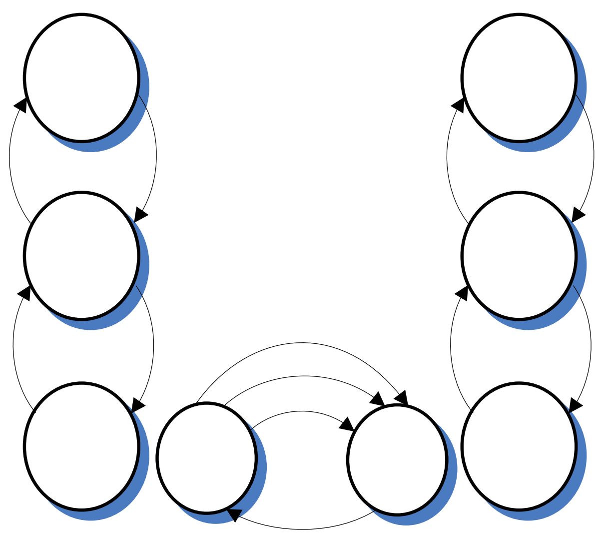


Figure 9. The extension model for monitoring operations

In this extension model, we use additional events SIG_1 on server side, SIG_2 and SIG_3 on client side to serve for monitoring purpose. ME_CL will have Listen state continuously that waits for new event to come in. With input event is SIG_1, SIG_2 or SIG_3, ME_CL move to Report state which will create monitoring report.

Therefore, the modeling for objects in DS by using previous proposal model will help us build monitoring entities which have to be able to quickly collect status, as well as events of special interest of monitored objects. This model will actively supports for behavior monitoring issues. In order to monitor for the behaviors of the monitored objects in DS, the monitoring entities must be associated with the specific architecture of DS and we will present the behavioral monitoring solution in another research.

V. CONCLUSION

The behavior modeling for monitored objects of DS has an important role in the development of efficient algorithms for the behavior monitoring and diagnosis problems in DS. By using the communicating finite state machines, we propose a method to develop the basic behavioral model and composition model of state machines for monitored objects in DS, we can present behaviors of monitored objects through input events, output events, states and transition function of these ones. Based on this model, we can develop a behavioral monitoring solution that is able to monitor activities and communication behaviors of objects in distributed systems; will actively support administrators in system operating.

REFERENCES

[1] Ajay D. Kshemkalyani, Mukesh Singhal, “Distributed Computing Principles, Algorithms, and Systems”, Cambridge University Press, UK, 2008.

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

[3] Gabriel A. Wainer, Pieter J. Mosterman, “Modeling and simulation theory and applications”, CRC Press, USA, 2011.

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

[5] Gerard J. Holzmann, “Design and validation of computer protocols”, Prentice Hall, USA, 1991.

[6] Son Le Van, Phuc Tran Nguyen Hong, Constructing the assisting system for monitoring the TCP/IP network based on ICMP and SNMP, Telecommunications and Information technology journal, Special issue: Research and development on Telecommunications and Information technology, 41-47, 2004.

[7] Phuc Tran Nguyen Hong, Son Le Van, An online monitoring solution for complex distributed systems based on hierarchical monitoring agents, KSE 2013 conference, Springer, 191-202, 2013.

[8] Weilong Hu, Hessam S. Sarjoughian, A co-design modeling approach for computer network systems, Proceedings of Winter Simulation Conference, 685-693, 2007.

[9] Yannick Pencolé , marie-odile cordier, Laurence Rozé, A decentralized model-based diagnostic tool for complex systems, International Journal on Artificial Intelligence Tools, 11 (3) , 327-346, 2002.

[10] Chang-Guo Guo, Jun Zhu, Xiao-Ling Li, "A Generic Software Monitoring Model and Features Analysis", Proceedings of the 2010 Second International Conference on Networks Security, Wireless Communications and Trusted Computing, 01, pp 61-64, 2010.

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

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

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

[14] 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.

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

AUTHORS PROFILE

Phuc Tran Nguyen Hong received the B.E.Sc. degree and the M.S. degree in IT from University of Danang, Danang City, Vietnam, in 1997 and 2003. His research interests include distributed systems, system monitoring and communication networks.

Son Le Van is a professor in Department of Computer Science, Danang University of Education, The University of Danang, Vietnam. His research interests include distributed systems, monitoring solutions and communication networks.