International Journal of Electrical, Electronics and Computer Engineering **2**(2): 54-60(2013)

ISSN No. (Online): 2277-2626

Testing And Implementation of Election Administration Approach for Distributed System

S.K. Gandhi and Pawan Kumar Thakur

Department of Computer Science & Engineering,
AISECT University, Bhopal (M.P), India.

(Received 15 July, 2013 Accepted 17 August, 2013)

ABSTRACT: The aim of testing is to prove a concept and verify its validity through the requirements defined and the available resources. Because systems can only be tested during a restricted period of time, testing cannot ensure complete correctness of an implementation or problems associated with the transformation from concept to implementation. The main purpose of this paper is to perform testing and implementation of election administration approach in order to assess the attainment of the system against the objectives that were set initially.

Keywords: Framework, Methodology, Network topology, Fixed Topology Election, Random topologies, Nodes Setup, Connection Files, Node View, Messages

I. INTRODUCTION

Testing is the process of looking for bugs in the implementation of a system through experiments. In order to validate the conceptualization made, it is necessary the definition of the used testing methodology and also the set of tests to be made on proof of concept implemented in accordance with it [1,2]. The solution presented will be tested in several different scenarios, for which the methodology and application of the test should be identical. In this paper section 2 discuss about ISO/IEC 9646- framework and methodology, section 3 we present the test definition of election administration, section 4 implementation of election administration approach. Finally section 5 represents the test execution and 6 conclude the paper.

II. ISO/IEC 9646- FRAMEWORK AND METHODOLOGY

ISO/IEC 9646 is a framework and methodology for conformance testing. This standard was originally developed to provide a platform and define a terminology for the application of tests on open system interconnection (ISO) systems. But due to its low usage, the methodology has been little used for compliance testing of these systems. The testing process described by this methodology is divided into three stages.

(i). Definition of tests. The first phase is the specification of an abstract test suite for the system in question, and is referred to as the definition of tests. The tests are abstract in that they are developed independently of any implementation [3].

(ii). Test implementation. The second phase consists of defining the tests in order to be executed, and is called the test implementation. This stage is to take into account the implementation that will be tested, adapted to the tests defined prior to system implementation.

(iii). Executing test. The last phase, the testing, consists in its execution and observation of results. Which leads to a verdict on the compliance of the system under test with the initial requirements defined [4,5].

III. TEST DEFINITION OF ELECTION ADMINISTRATION

The first test starts with the establishment of the network topology. It then passes through the election mechanism, and finally the super nodes are elected amongst all the nodes present in the created topology. In First test, it be evaluated the election over fixed topologies as shown in Table 1. In this second test, it be evaluated the election over random topologies as shown in , and on a large-size network as shown in Table 2, to verify the scalability of the algorithm [6,7].

A. Election Test – Fixed Topology Election

The intent of this test is to validate the election result on a pre-established network topology. First the topology is defined and validated then the election occurs and the reachable nodes/process is evaluated. The topology of the network created must be the desired before the test takes place. It is now presented a summary of the implementation. This is used to confirm the creation of the topology, to ensure that all nodes are actively involved in the election mechanism and that by the end of the test is possible to verify that the election occurs.

B. Election Test – Random Topology Election

Those simple test cases to evaluate the topology and the algorithm behavior are of very importance, since the tested cases represent several specific topologies present in the network.

Table 1. Details on the Fixed Topology Election Test.

Test	
Test Name:	Fixed Topology Election
Group:	Fixed Topology
Purpose:	Validate the Election
Comments:	This test is conducted to validate the election. It is applied to a pre-established fixed topology and them the validation of the Election.
Behaviour	**Verdict**
! Set the desired topology	
! Verify that the resulting topology was the desired	
? Topology successfully created	
? Election occurred in all nodes/process	
! Check nodes elected	
? All nodes are reachable	
? Not all nodes are reachable	
? Election not occurred	Success
? Topology corrupted	Failure
	Failure
	Failure

Table 2. Details on the Random Topology Election Test.

Test	
Test Name:	Random Topology Election
Group:	Random Topology
Purpose:	Validate the Election
Comments:	This test is conducted to validate the election. It is applied to a pre-established random topology and them the validation of the Election.
Behaviour	**Verdict**
! Set the desired topology	
! Verify that the resulting topology was the desired	
? Topology successfully created	
? Election occurred in all nodes/process	
! Check nodes elected	
? All nodes are reachable	
? Not all nodes are reachable	
? Election not occurred	
? Topology corrupted	Success
	Failure
	Failure
	Failure

Although the algorithm needs to provide also an efficient election regardless, the network structure but also taking into accounts it scalability [7]. Having the results for specific networks the next test evaluates the performance over a large random topology.

For this test its validation is subject to the same requirements as above, but in case of a random topology. Besides being necessary to verify the created topology, it is also necessary to confirm that all nodes participating in the mechanism are also reachable. If all these points are checked the test achieved success.

IV. IMPLEMENTATION OF ELECTION ADMINISTRATION

After defining the tests, these will be applied to the solution created. The creation of the solution was to define structures to record the messages and create tasks associated with the creation of the necessary criteria. To check the result of the test, it must be used a simulator that represents identically the environment where the algorithm should be applied.

A. Nodes Setup and Connection Files

The Nodes Setup File is used to define the nodes and its type may be election admin and candidate nodes and Connections File used to establish the connection between the nodes these two files established the network topology [8,10,11]. It's also possible to define if a new node, that joins the network, will have the role of election admin's or simple node , and modify this parameter later. This option is set by a file that the simulator uses to setup the initial nodes. The following Table 3 display the data structure of Node setup file.

Table 3. Nodes Setup File.

Joining Time	Type of Node	Node ID	Case
At 2	Election Admin's	E_ID1	EA
At 3	Candidate Node	C_ID1	1
At 4	Candidate Node	C_ID2	2
At 5	Candidate Node	C_ID3	3
At 6	Candidate Node	C_ID4	4
At 7	Candidate Node	C_ID5	5
At 8	Candidate Node	C_ID6	6
At 9	Candidate Node	C_ID7	7

Table 4. Connections between the nodes.

Nodes	Connected Nodes (Neighbours and Admin's node)						
EA	1	2	3	4	5	6	7
1	2	3				EA	
2	1	3		6		EA	
3	1	2		4		EA	
4	3	5				EA	
5	4	6		7		EA	
6	2	5		7		EA	
7	7	5		6		EA	

The network nodes are EA, 1, 2, 3, 4, 5, 6 and 7, and after each nodes are all the connections established to him. Therefore, for the first line, the node EA will have established connections from and to the node 1, 2, 3, 4, 5, 6, and 7. The second line, the node 1 will have established connections from and to the node 2, 3 and EA. The same it's visible for the nodes 2, 3, and EA. Node 2 has connections to 2, 3, 6, and EA and the node 3 has connections to 1, 2, 4 and EA. This kind of setup offers good control for the construction and maintenance of the network topology, with needs to be controllable.

B. Node View

The first step in implementing the algorithm is for each node to send information about their neighbours to all its neighbours in order to create the view. Information relating to the local knowledge, that is the neighbours and admin's nodes on the network, is called the View. Each view of a node is unique across the network. Thus while new node are entering the network, information about the neighbours surrounding each node is always updated. This is done when a new node join the network and sends a JOIN message to another node nearby and to admin's node, if the received node and admin's node responds, the connections is established, if not respond, the new node sends a new JOIN message to connect to other node nearby until it receives a response. As a node connects to another, the table that records all the node connections is updated. This table is called: NodeView my_RoutingTable. This information will be sending randomly at each node, to avoid any future possibility of saturation the network. This occurs in each node by the function sendMsgUtilityToAll(){} independent of the node state be candidate or super. The NodeView my_RoutingTable store the Node ID, and the others node connected to it. The NodeView structure is visible is Fig. 1.

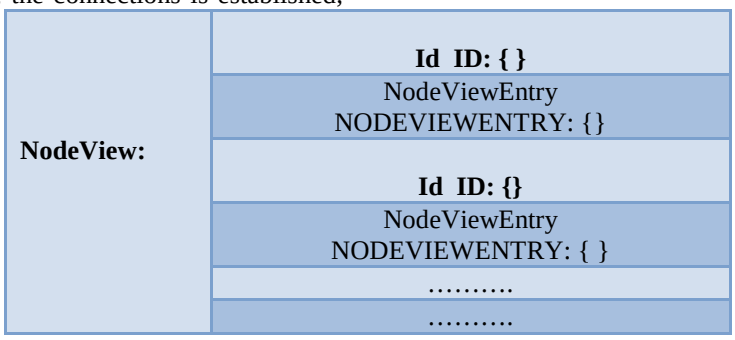

Fig. 1. The Node View content.

The node ID and its neighbours are sent over the network using the message: MY_UTILITY, to the others neighbours and admin's node. This message has a TTL of 2, so it will only be propagated by two nodes in the network. When another nodes receives the MY_UTILITY message it will be propagated to all neighbours, and forwarded to other nodes. With the use of TTL = 2, the information reaches all nodes that are at distance of 2 hops from the node issuing the message. Thus the local information is sent to a predetermined distance, without the risk of flooding the network. With this process the neighbours node receives the MY_UTILITY message, sets the TTL = 1, and resend the message. Once the message is forwarded, each node also created a dedicated structure to store all messages that are forwarded or received with TTL = 1, its name is Vector<My_Data> vec_Data, and the process occurs in the function: storage_msg(msg).This vector is responsible to store the source node ID, the NodeView and the n_hops (Number of hops). With the messages storage, the node now has information on the other nodes that are at distance of two hops, but also its others neighbours. These data will be needed to calculate the future benefits that a combination has, in relation to others. Once this process is concluded, the node now has a range ser of information to be used in the election mechanism.

C. Messages

There are number of message created for support of exchanging data between the nodes:

(i). MY_UTILITY. The MY_UTILITY message exchange information relative to the neighbours of each node and election admin's. The messages exchanged consist of essential fields for their propagation in the network as the Source, Destination and Type as shown in Fig. 2. In case the MY_UTILITY message has additionally the fields: Key, Mode, NodeView and Hops.

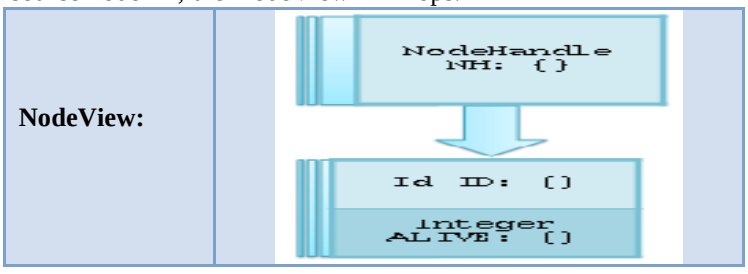

Fig. 2. NodeViewEntry content.

Has the messages are received by the node in real-time, information about the nodes that are on the network will need to be storage. For this the node will have a local vector called vec_Data, used to store all this information about the surrounding nodes. This information it's extremely valuable, since it contains the source for calculating the iterations as shown in Fig. 3.

(ii). MY_ELECTION. An election message is sent to announce an election. The MY_ELECTION message exchange information relative to nodes and election admin's. Sending this message will be conducted for all nodes that are present in the surrounding area so that the outcome of the election arrives to the node/s to be elected. It could have been a selective choice of destination that is admin's node in order to avoid sending the message to the others nodes since this message will not take on too much bandwidth between nodes, or not consume more battery power on the device [13,12].

Message Type: MY_UTILITY	**Id ID: { }**
	NodeViewEntry NODEVIEWENTRY: {}
	String KEY {}
	NodeHandleSOURCE: {}
	NodeHandleDESTINATION: {}
	Int TYPE: {}
	Int MODE: {}
	Int TTL: {}
	NodeView: NODEVIEWMESSAGE: {}

Fig. 3. MY_UTILITY Message.

Message Type: MY_ELECTION	**String Key: { }**
	NodeHandleSOURCE: {}
	NodeHandleDESTINATION: {}
	Int TYPE: { }
	Int MODE: { }
	Int TTL: { }
	ElectionMessage: ELECTIONMESSAGE: { }

Fig. 4. Election Message.

(iii). VERIFY_MESSAGE. A verify message to the current coordinator. Election Administration verifies election message sent by any process or concurrent process. The VERIFY_MESSAGE message exchange information relative to current coordinator and election admin's as shown in Fig. 5.

Message Type: VERIFY_MESSAGE	**String Key: { }**
	NodeHandleSOURCE: {}
	NodeHandleDESTINATION: {}
	Int TYPE: {}
	Int MODE: {}
	Int TTL: {}
	VerifyMessage: VERIFY_MESSAGE: {}

Fig. 5. Verify Message.

(iv). ALIVE_MESSAGE. An alive message to the next highest process number if the current coordinator is fail. Election Administration will simply find out the alive process with the highest process number using helper. The ALIVE_MESSAGE exchange information relative to election admin's and alive node with the highest ID as shown in Fig. 6 .

Message Type: VERIFY_MESSAGE	**String Key: { }**
	NodeHandleSOURCE: {}
	NodeHandleDESTINATION: {}
	Int TYPE: {}
	Int MODE: {}
	Int TTL: {}
	CoordinatorMessage: COORDINATORMESSAGE: {}

Fig. 6. Alive Message.

(v). COORDINATOR_MESSAGE. The Coordinator message is send to all processes as a new coordinator of the system. The COORDINATOR_MESSAGE exchange information relative to election admin's and all the node/process of the system as shown in Fig. 7 .

Message Type: COORDINATOR_MESSAGE	**String Key: { }**
	NodeHandleSOURCE: {}
	NodeHandleDESTINATION: {}
	Int TYPE: {}
	Int MODE: {}
	Int TTL: {}
	CoordinatorMessage: COORDINATORMESSAGE: {}
	String Key: { }
	NodeHandleSOURCE: {}
	NodeHandleDESTINATION: {}
	Int TYPE: {}
	Int MODE: {}
	Int TTL: {}
	CoordinatorMessage: COORDINATORMESSAGE: {}
	…….
	…….

Fig. 7. Coordinator Message.

(vi). QUERY_MESSAGE. A query message is send when a crashed process is up. The up process can query to the election admin about the current coordinator. The QUERY_MESSAGE. exchange information relative to a crashed up node and election admin's node as shown in Fig. 8.

Message Type: QUERY_MESSAGE	**String Key: { }**
	NodeHandleSOURCE: {}
	NodeHandleDESTINATION: {}
	Int TYPE: {}
	Int MODE: {}
	Int TTL: {}
	CoordinatorMessage: QUERY_MESSAGE: {}

Fig. 8. Query Message.

V. TEST EXECUTION

There is a set of known topologies that can be found in different day-to-day contexts that difficulties the setting of a semi-centralized architecture. The use of the topologies present in real life will help bring closer the applicability of the solution with the today infrastructure. For this test is used the "Star" topology. In test execution of election administration approach in normal case (NC) when a process normally detects the failure of the coordinator process it sends election message to the EA and waits for to receive coordinator message. EA sends verify message to the current coordinator to be sure about the election and sends alive message to the next highest process number to check either the current highest process is alive or not and gets a reply message [9,11]. EA selects that process new coordinator of the system and sends coordinator message to all processes.

In test execution of election administration approach in query after crash recovery case (QCRC) when a process ordinary recover from failure it send a query message to the EA.. If the query processes number is higher than EA elect that process as current coordinator and send the coordinator message to all other process of the system. If a the process with lower number it sends coordinator message of current coordinator of the system. In test execution of election administration approach in concurrent election case (CEC) when more than one process may detect that the coordinator process has crashed. They will send election message to EA After verification will consider election request of the process having higher process number [8].

VI. CONCLUSION

In first test of election administration is evaluated the election over fixed topologies, in second test election evaluated over random topologies and on a large-size network verify the scalability of the algorithm. To getting the election administration algorithm results, we offers different types of methods to evaluate the algorithm performance regarding tables, vectors, structures or any other local variable, it's possible to record information on the number of exchanged messages and other statistics options and to record all in a file. The use of star topology is mainly due primarily to confront the solution developed with a topology in which there is a clear solution. The most obvious solution to an election by a network administrator will be the election of a single central node/process, which should serve to indexing mechanism for all others. The desired topology has achieved.

REFERENCES

[1] Sinha P.K, Distributed Operating Systems Concepts and Design, Prentice-Hall of India private Limited, (2008).

[2] H. Garcia-Molina, *"Elections in Distributed Computing System"*, IEEE Transaction Computer, Vol. C-31, pp.48- 59, Jan. (1982).

[3]. Technology, ISO. Information.. *"Open Systems Interconnection, Conformance Testing Methodology and Framework. International Standard IS-9646"*, ISO. 1991. Vols. CCITT X.290–X.294 (1991).

[4]. Tretmans, Gerrit Jan, *"A Formal Approach to Conformance Testing. A formal approach to conformance testing"*, The Netherlands : Hengelo, 1992. 90–9005643–2(1992).

[5]. Tretmans, Jan., *"An Overview of OSI Conformance Testing. University of Twente : Proceeding MOVEP '00 Proceedings of the 4th Summer School on Modeling and Verification of Parallel Processes"*, 25 Janeiro, 2001.

[6]. Dr. Gandhi S.K. and Thakur P. Kumar, *"Designing a Chat room application using peer-to-peer and client-server approach of Distributed Systems"*, Anusandhan: Science Technology & Management Journal of AISECT University, Vol II Issue III, pp. 95-100, March(2013).

[7].Dr. Gandhi S.K. and Thakur P. Kumar, *Designing Issues for Distributed Computing System: An Empirical View*," International Journal of Innovative Research & Development, Vol 1 Issue 11, pp. 326-40, Dec. (2012).

[8]. Dr. Gandhi S.K. and Thakur P. Kumar, *"Election Administration Algorithm for Distributed Computing"* International Journal of Electrical, Electronics and Computer Engineering 1(2): pp. 1-6(2012).

[9] Thakur P. Kumar , Kumar Ram, Ali Ruhi and Malviya Rajendra, *"A New Approach of Bully Election Algorithm for Distributed Computing"*, Int. J. of Electrical, Electronics and Computer Engineering (IJEECE) Vol 1(1): pp. 72-79(2011).

[10]. Dr. Gandhi S.K. and Thakur P. Kumar, *"A comparative study of Bully and Ring Election Algorithm in the Terms of Complexity & Message passing"*, AERA: National Conference on Advances in Engineering Research & Application, March 14-16, 2013, Oriental College of Technology, Bhopal.

[11]. Dr. Gandhi S.K. and Thakur P. Kumar, *"Analysis of Mutual Exclusion Algorithms with the significance and need of Election Algorithm to solve the coordinator problem for Distributed System"* International Journal on Emerging Technologies 4(1): 17-25(2013).

[12].Thakur P. Kumar, *"Reliable failure detector for Distributed System"*, IC-GISM: International Conference on Global Innovation in Science & Management, 2013, Swami Vivekanand University Sagar M.P. India.

[13]. Dr. Gandhi S.K. and Thakur P. Kumar, *"Performance Analysis of Various Election Algorithms in Distributed System" BSS Journal of Computer Application*, 2013.