Fault-tolerant Strategy to Solving Masquerading Faults in Safety-critical Machines

Adeosun Olajide Olusegun, Olajide Blessing Olajide, Adeosun Titilayo Helen

Abstract—This research presents a fault-tolerant strategy to solving masquerading faults among nodes in safety-critical machines. The possibility of having a failure-free critical machine operation over a specified time in a given environment for a given purpose is hinge on the reliable operation of its constituent nodes. Particular measures must be taken to ensure that software-defect masquerading faults among nodes do not cause the machine to hang or crash during its operation schedule while taking turns to use the shared network. Hence, this work designed and implements a fault-tolerant model which is based on authentication, network management intelligent system, and triple modular redundancy (TMR) concept in a Time-Triggered Architecture (TTA) of star topology to solve masquerading faults.

Index Terms—**critical-machine, fault-tolerant, masquerading fault, node, safety-critical, time-triggered architecture, triple modular redundancy.**

1. INTRODUCTION

Critical machines are systems in which defects could have a dramatic impact on human life, the environment or significant assets. The presence of some faults in these systems, accidental (design, interaction, etc) as well as intentional (human, virus, etc) are inevitable. Due to catastrophic consequences (human, ecological, and financial disasters) that could involve a fault, these systems must be fault-tolerant [1]. The system that implements several embedded system are referred to as machine in this research context and the individual embedded systems that makes up the machine are referred to as nodes. For such machine, dependable message delivery among nodes is crucial to overall system dependability. Most, if not all collaborative safety-critical embedded systems often have strict dependability requirements that are influenced by factors such as cost and size constraints or the system architecture [7]. These factors cause nodes to use a single shared network for all local message communication. The architecture of embedded systems forms the safety-critical core of the services provided and machine dependability [2]. One threat to this dependable message delivery is a software-defect masquerade fault, where a software-defect causes one node or process to send a message as having come from another node or process. Unfortunately, many embedded system designs do not address this particular failure mode. Traditionally, masquerading has been viewed as a malicious attack, rather than a fault tolerance problem. This is a misconception because embedded system's network are typically closed (i.e., their networks are not accessible to outside intruders) and usually do not include security methods for preventing malicious attacks. Also, most fault tolerance techniques used in embedded systems not only fail to prevent masquerading, but also assume fault models in which masquerade faults do not occur. For example, the Byzantine fault model assumes that the identity of each general is correct at all time of operation of its model [6]. If software-defects within the system itself cause masquerading then this invalidates the assumption. The existing fault-tolerant strategy uses a Time-triggered Star architecture and protocol that has clique avoidance algorithm and group membership service deployed in the software of the nodes and star coupler to detect and prevent masquerading fault in embedded system network. In the advent of masquerading faulty node the particular node is driven to a freeze state until the nodes are repaired and the machine is restarted. In most cases the machine are designed to fail in a safe mode and little or nothing as being done to make the machine recover its faulty nodes automatically this result in service down time for the machine while waiting for the engineer to rectify the faulty.

In the case of a safety-critical machine whose nodes are critical and must be up and running all through the machine operation schedule, driving such node to the freeze state could be disastrous and result to loss of life or huge resources. Also if the node's transmission control is dependent on the previously transmitted node(s) then the masquerading fault generate a ripple fault effect on the other nodes yet to transmit and corrupt the machine's output. In other words the machine's dependability is compromised and may even cause the machine to halt its service. Hence, the need for an enhance fault tolerance strategy to improve on this deficiency of the existing strategies.

The strategy of this work ensured the detection of masquerading nodes, prevent them from transmitting at the wrong time stamp and ensure they are not driven to freeze state rather they are driven back to the initial stage. At the initial stage they still have the opportunity to transmit in their normal authorized time stamp frame. Therefore, this strategy ensures that the machine does not halt its service before the completion of its operation schedule.

2. RELATED RESEARCH

Sarika and Verghese [8] proposed a new algorithm for detecting a particular phishing attack called tabnabbing using distributed software agents. Tabnabbing is an attack which prompts the user to enter login details to well-known websites by impersonating those websites when the webpage is idle for some time. The distributed agent's method is based on the incorporation of autonomous distributed agents with strong level of intelligence. A distributed multi-agent system presents a great capacity for high level of learning, distribution of tasks and responsibilities, fault recovery, and adaptation to new changes. The work consists of agents in 3 levels and they communicate with each other as needed. The level1 agents checks the URL of the webpage and confirms whether it is not blacklisted and checks the layout of webpage and confirms whether it is not changing when the webpage is

idle for some time. The level 2 agent detects the webpage as a phishing page or a legitimate page. In this mechanism level1 agent acts as an agent for detecting URL obfuscation and detects tabnabbing attack. Level 2 agent is acting as a webwatcher when the threshold values of actual webpages differ from the values received from its phished webpage.

The multi-faceted approach to fraud detection in personal communication systems proposed, [9] consists of Personal Communication Agents (PCAs) in conjunction with rudimentary user profiles (UPs) of the subscriber's communication patterns, Mobility Network Agents (MNAs) that interact with PCAs and Fraud Breaking Agents (FBAs). The work however advocated that this approach is required because several points in the network can be attacked. First, a subscriber phone can be cloned. Second, a cloned phone cannot be detected until a reliable profile is constructed normally at a call processing center with access to switch information and billing records. Third, companies need to be able to detect a fraudulent call quickly and intercept it before significant charges are accumulated, i.e. calls in progress need to be monitored in real time for fraudulent activity.

Ashraf and Awarta [3] proposes a new software pattern called Recovery Block with Backup Voting (RBBV) to improve the reliability of the classical Recovery Block (RB) method in those situations where it is difficult to construct an effective acceptance test. Further to this, fourteen proven design techniques for hardware and software were reviewed, analyzed and represented using the new template.

Viacheslav [10] proposes several design optimization strategies and scheduling techniques that take fault tolerance into account in the context of distributed real-time systems. The design optimization tasks addressed include, among others, process mapping, fault tolerance policy assignment, and checkpoint distribution.

Luis [5] observed that Controller Area Network CAN is been used in some safety-critical applications and was able to point out that CAN inherent event-triggered transmission mode does not favor dependability. In other words it is easier to detect errors and build fault-tolerant mechanisms for time-triggered communication protocols. His work also advocated that CAN nodes may fail uncontrollably e.g., babbling idiot failure mode and that using bus guardians grants fail-silence in the time domain, favoring the design of fault-tolerant mechanisms.

D¨obel, H¨artig and Engel [4] developed a software based approach to task replication on top of the L4 Fiasco microkernel. Their approach involved a master-controller task, which monitored the execution of redundant tasks. The controller handled CPU exceptions, page faults and system calls made by the redundant tasks and ensured they had identical state at these points.

3. METHODOLOGY

A fault-tolerant model based on authentication, network management intelligent system, and triple modular redundancy concept was implemented in a Time-Triggered Architecture (TTA) of star topology. The TTA of star topology is used because it includes Time Distribution Media Access (TDMA) and a redundant star coupler that will not allow the bus guardian to become a single point of failure. The strategy employed the use of the following basic fault-tolerant design concepts:

(i) Modularity: the proposed strategy views each node in the machine as an independent module that has a mechanism to ensure fault tolerance.

(ii) Fault containment: If a node is faulty, the design prevents it from contaminating the others.

(iii) Redundancy: The time-triggered star architecture to model a generic embedded system network has two redundant shared communication channels.

(iv) Repair: in the case of a node that develop masquerading fault, this strategy denies such node of access to the shared channel and then reset it back to its initial state as it was before the transmission session started.

3.1 Model Architecture

The fault-tolerant model identifies masquerading nodes and stops their effect from propagating into the embedded critical-system shared network. It does this by replacing the faulty nodes with their replicas that have the initial correct credential to transmit at the correct time slot. The model utilizes authentication service and LWA service to detect masqueraders. And to ensure the node's availability, reliability and dependability in service delivering to the shared network, a triple modular redundancy (fault-tolerant mechanism) was deployed at each node within the critical machine to perform node repair. Therefore, each node is abstracted as a module of three healthy identical nodes.

The strategy's model (fig 1) is made up of three major modules:

(a) Node's Authentication Module (NAM)
(b) Node Health Monitor Module (NHMM)
(c) Nodal Fault-tolerant System Module (NFTSM)

These modules were deployed on the bus guardian that is resident on the star coupler in TTA of star topology architecture to accomplish the set objectives.

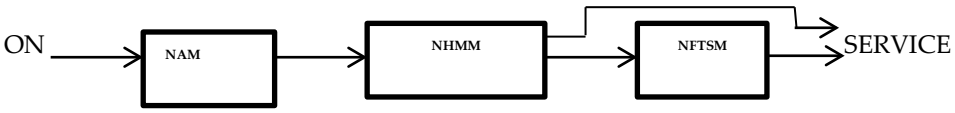

Fig 1: Model of the Fault Tolerant Strategy's Modules

3.1.1 Node Authentication Module (NAM)

During node configuration the programmer statically assign identity (Id) to all nodes within the critical machine. He also assigned a secret key to each individual node which is only known to the node and the bus guardian in the star couplers. The secret keys are not broadcasted periodically in the Message Description List (MEDL) as the Id and Number of Slot (noS) to other nodes on the network.

The NAM comprises of two servers:

(i) Authentication Server (AS)
(ii) Ticket Granting Server (TGS)

The AS keeps the database of the statically assigned secret key (SK) and their corresponding Id. A symmetric cryptosystem for using the secret key to encrypt the Id of the nodes requesting authentication is also available on this server. The research work uses mono-alphabetic cipher substitution as opined by Caeser Cipher for encryption and decryption in the AS.

The TGS is the server that receives the authenticated credentials from the nodes after the node as decrypted the encrypted Id with its own copy of the SK. It then adds a session key to the credential before returning it back to the node. The credential that is handed back to the node that comprises of the SK, Id and the SEK. The SEK is the addition of Node Time Slot (TS) and noS. The SEK is used to gain access to the shared network at the timeslot indicated in it. This is made possible by a variable called timeout counter inside the node that counts down the SEK until it becomes zero then the node is granted access to the shared channel. i.e., Timeout counter = SEK-1 until it eventually becomes zero. This happens simultaneously on all the nodes before transmission. The first node to count to zero transmit first and the next to get to zero do so after the first has completed the use of the channel in its time frame and so on in that manner till the last node in a normal fault free scenario. The TGS also keeps a database of node Id versus the SEKg. This database is used to keep a copy of the assigned SEK to the node as a guide. When the timeout counter of a node equals zero and it wants to transmit, it first tender its SEK to the TGS, then the TGS uses the SEKg to compare the tendered SEK parameters. If there is any discrepancy in the SEK to the stored SEKg then the node attempting transmission is reported to be masquerading.

3.1.2 Node Health Monitor Module (NHMM)

The NHMM houses the lightweight agents LWAs that utilizes their intelligent attributes to detect and identify any discrepancy in the SEK parameters of any node attempting to transmit on the network to the correct parameters of the authorized node to transmit at that particular time slot that is stored in the TGS database. The LWA also migrate to any faulty node to inform the faulty node module of the need for transfer of service to a redundant replica of the faulty node. It does this function by reporting the faulty node to the fault management mechanism at the node. And hence the fault is rectified without any impairment done to the critical embedded system network. The LWA used in this module are condition sensor and problem identification agent type, both of which are classified as diagnostic agents.

3.1.3 Nodal Fault-tolerant System Module (NFTSM)

After The faulty node has being identified and prevented from using the shared network at the wrong TS by the NAM and NHMM, the NFTSM is used to replace the faulty node with its replica that is viable and healthy. This module implements a Triple Modular Redundancy (TMR) with a Standby Replacement (SR) fault tolerance mechanism to achieve its task. The LWA from the NHMM reports to the faulty node that one of its constituent nodes is faulty. The TMR module then uses the disagreement detector to identify the faulty node and service is transfer to one of the remaining two spare. Here, the nodes do not fail or freeze and so the critical machine does not experience any down time. The faulty node can be repaired or reconfigured in an offline mode or online mode through the Field Programmable Array FPGA interface of the faulty node.

3.2 The Node

The safety-critical embedded system (nodes) is made up of three identical nodes in parallel. The individual node has the capability to perform independently the assigned task within a critical machine. A single node is abstracted as a finite state machine with six different states and can transit from one state to another base on the fault

tolerant logical function. The states are COLD state, INITIAL state, ACTIVE state, FAULT state, TIMEOUT state and the FINAL state. The node state diagram is shown in fig 2. At any instance of a node-shared network communication only a node in the TMR module is active while the others are on SR. To help the disagreement detector to select a replica to replace a masquerading node a Basic Fault-tolerant System BFS switching concept is employed.

3.2.1 TMR with SR

The architecture of TMR with SR used at each node is such that each node is made a module that has three identical nodes that has the capacity to perform the same function independently. At any instance one node in the module is operational and the remaining two serves as standby or spares. The fault detector is used to confirm the faulty node within the module which the LWA reports to it. After confirming truly that the operational node is masquerading then it put a stop to its operation and uses its inbuilt vote-logic to vote the faulty node out and select one out of the viable spares to make it operational. Hence breakdown at the nodal level is handled and each node is guaranteed of availability and reliable service delivery in

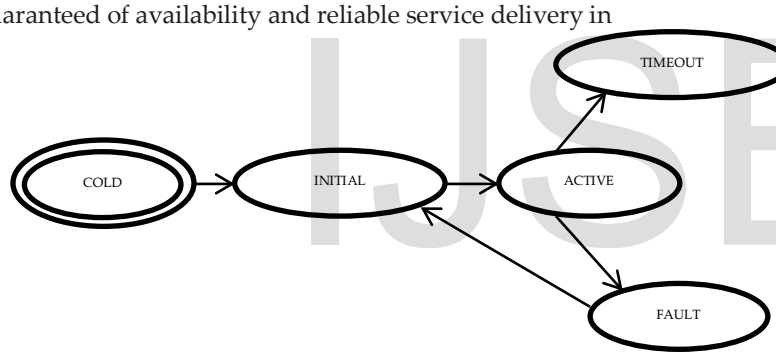

Fig 2: Node State Diagram

the machine operation schedule. The faulty node can then be reconfigure via the FPGA and made to become a viable spare for the module when the programmer comes for service routine. Figure 3.3 shows the TMR with the standby sparing structure of the nodes in the module. Note that the standby system used in this research is the Cold Standby System at each critical node module. According to McCarthy [11], the reliability of each individual critical node within the critical machine is expressed as:

$$R_{node} = 1-(1-r)^{s+1} = 1-(1-r)^{2+1} \quad (1)$$

Where s is the number of node spares and

r is the reliability of the individual replicas of the node

4. RESULT AND DISCUSSIONS

This work used simulation to support the use of authentication, Light Weight Agent (LWA) and Triple Modular Redundancy (TMR) to solve software-defect masquerading faults in safety-critical embedded system network. Simulink models of matlab version a2012b were developed and simulated to investigate the performance of the critical machines with the fault-tolerant scheme. A critical-machine without the scheme was also modeled and observed for comparison purpose. To observe the operation of the fault-tolerant system and the non-fault-tolerant system, four nodes were experimented in ten successive trials. The simulation demonstrated how each node transit from cold state to final state. It also randomly introduces fault(s) into the nodes at every instance of the machine operation. The non-faulty nodes are denoted by T and the faulty nodes by F. The number of failure (NumF) and the Number of Survival nodes (NumS) were computed automatically. So also reliability for each instance of the critical machine operation was calculated.

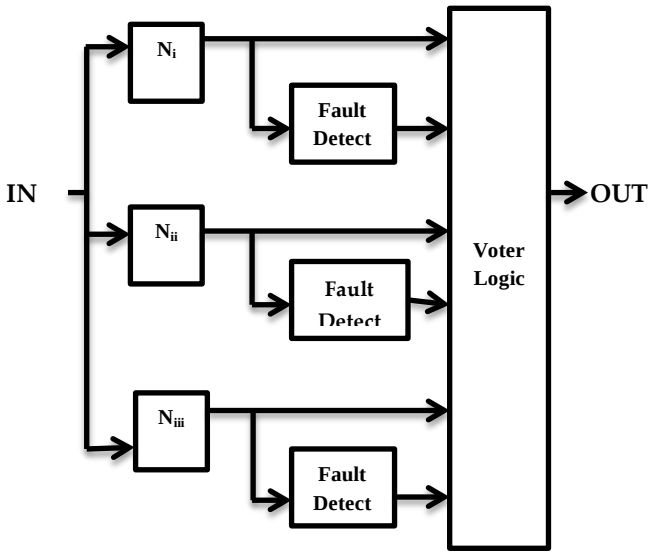

Fig 3: TMR with Standby Sparing Structure at the Node

4.1 Non-fault-tolerant Scenario

Table 1 showed the node Id, the NumF and NumS and the corresponding reliability for each machine trial. When NumF is 2 and NumS is 2 the reliability of the machine is 0.5 (50.00%). likewise when NumF is 3 and NumS is 1 the reliability is 0.25 (25.00%). From these results it is observed that the reliability of the critical machine is less than 1.

A	B	C	D	NumF	NumS	R	Time(s)
F	T	F	T	2	2	0.5	11
T	T	F	F	2	2	0.5	25
F	F	F	F	4	0	0	45
F	F	F	F	4	0	0	63
F	T	T	F	2	2	0.5	82
F	T	F	F	3	1	0.25	104
T	F	T	F	2	2	0.5	117
F	T	T	T	1	3	0.75	125
F	T	F	F	3	1	0.25	138
F	T	F	T	2	2	0.5	146

Hence, the machine does not have the degree of reliability expected of safety-critical machines because it can fail at any time and the degree of trust for the machine is low. The possibility of all the nodes surviving till the end of the machine operation is not guaranteed.

Table 1: Non Fault-tolerant Critical Machine

4.2 Fault-tolerant Scenario

For a system to be fully reliable, all the starting nodes must survive till the end of the system's operation.

$$\text{Reliability (R)} = \frac{\text{NumS}}{\text{NumO}} \quad (2)$$

Where:

NumS is the number of survived nodes; and

NumO is the original number of working nodes at the start of the machine.

This implies that NumO must be equal to NumS. The results of the performance of fault-tolerant machine (that has the strategy deployed in its network) is shown in table 2. When NumF is 2 and NumS is 2 the reliability is 1. So also, when NumF is 3 and NumS is 1 reliability is still 1. The reliability of the fault-tolerant machine is found to be 1 all through the trials. This is as a result of all the starting nodes surviving till the end of the critical machine operation. With these results, the fault-tolerant strategy has achieved the aim of increasing the reliability of critical machines by solving the problem of masquerading nodes in embedded nodes of the machines.

4.3 Fault-tolerant Machine versus Non-fault-tolerant Machine

The results of fault-tolerant critical machines (fig 4) showed that at 0.5 (50.00%) reliability; fault-tolerant strategy increased its reliability by 50.00% to give 1 (100.00%). So also when the machine has a reliability of 25.00% its reliability is increased by the strategy to give an increase of 75.00% so as to make it 1 (100.00%). Having recorded these results, the fault-tolerant strategy has successfully optimized the reliability of the critical machines.

A	B	C	D	NumF	NumS	R	Time(s)
T	T	F	T	1	3	1	10
F	F	T	F	3	1	1	25
F	T	T	T	1	3	1	29
F	F	F	T	3	1	1	37
T	T	F	F	2	2	1	54
T	T	F	T	1	3	1	66
T	T	F	T	1	3	1	74
T	T	T	T	0	4	1	79
T	T	T	T	0	4	1	85
F	T	F	T	2	2	1	97

Table 2: A Masquerading Fault-tolerant Critical Machine

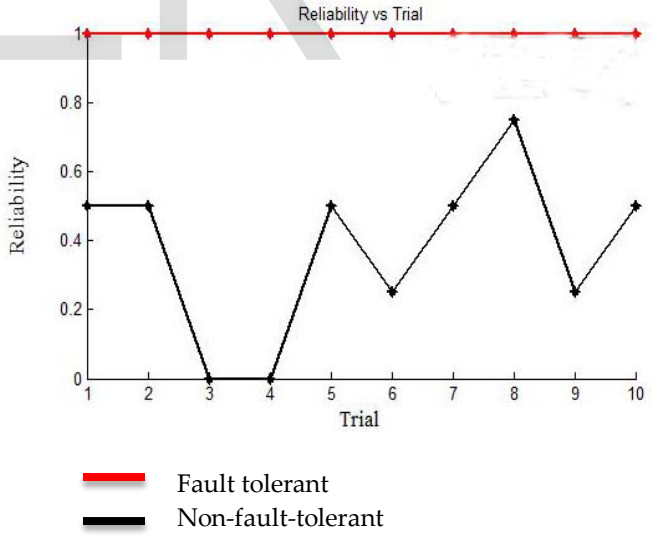

Fig 4: Reliability Distribution for the critical machine

5. CONCLUSION AND RECOMMENDATIONS

This work has presented a theoretical design and a prototype implementation of a proactive fault-tolerant strategy for masquerading fault management among safety-critical embedded systems. Correspondingly the results from the simulated generic safety-critical system model shows that the approach of this strategy to solving masquerading is viable if properly deployed in an

embedded system network. It is capable of providing a reliable, dependable and always-available machine to organizations, industries, banks and every other social sector that hinges on computer automation in delivering their services. The resulting conclusion from the comparison shows that critical machine with the fault-tolerant scheme is more reliable than non-fault-tolerant scheme; 74.10% reliability increased was recorded. In a nutshell, this paper has presented a proactive approach to masquerading fault management in safety-critical embedded systems.

The approach of this work has been modest with some limitations. Hence, other areas of importance of embedded system network that can be explored include remotely connected wireless embedded node networks, security and production cost minimization. More research in these three aspects combine with this work will provide users of critical machines with the needed critical system that will be reliable and dependable.

REFERENCES

[1] A. Avizienis, "Dependable Systems of the Future: What is still needed?," IFIP International Federation for information processing Volume 156, pg. 79-90, 2004.

[2] A. Ademaj, "Slightly-Off-Specification Failures in the Time-Triggered Architecture," Proceedings of the Seventh IEEE International Workshop on High Level Design Validation and Test (HLDVT'02), Cannes, France, pp7-12, 2002.

[3] A. Ashraf and B. Awarta, "Design Patterns for Safety-Critical Embedded Systems," Diese Dissertation ist auf den Internetseiten der Hochschulbibliothek online verfügbar, pp19, 2010.

[4] B. D¨obel, H. H¨artig, and M. Engel, "Operating system support for redundant multithreading," Proceedings of the tenth ACM international conference on Embedded software. ACM, pp. 83–92, 2012.

[5] A. Luis,"Safety-critical Automotive Systems New developments in CAN", in W4: Design Issues in Distributed, Communication-Centric Systems, University of Aveiro, LiU-Tek-Lic-2006:58, pp1,2006.

[6] Y. Minsky, R. Renesse, F.B. Schneider and S.D. Stoller, "Cryptographic Support for Fault-Tolerant Distributed Computing," *Proc. 7th. ACM SIGOPS European Workshop*, pp109-114, 1996.

[7] H. Pfeifer, "Formal Verification of the TTP Group Membership Algorithm". In IFIP TC6/WG6.1 International Conference on Formal Description Techniques for Distributed Systems and Communication protocols and Protocol Specification, Testing and Verification, FORTE/PSTV 2000, Pisa, Italy, pp3-18, 2000.

[8] S. Sarika and P. Verghese, "Distributed Software agents for antiphishing", in IJCSI International Journal of Computer Science Issues, Vol. 10, Issue 3, No 2, pp1-4, 2013.

[9] A. Suhayya, T. Mansour and W. Tony, "Multi-Agent Systems Approach for Fraud Detection in Personal Communication Systems", in AAAI Technical Report WS-97-07, Institute for Information Technology National Research Council of Canada Ottawa, Canada KIA 0R6, pp3-4, 1997.

[10] I. Viacheslav, "Scheduling and Optimization of Fault-Tolerant Embedded Systems", in Linköping Studies in Science and Technology, Thesis No. 1277, ISSN 0280-797, pp1, 2000.

[11] M. McCarthy, "Fault-Tolerant," Tech Target, 3(1) pp13-21

- Adeosun Olajide Olusegun has a doctorate degree in Computer Science and works as a lecturer in the Computer Science and Engineering Department of Ladoke Akintola University of Technology, Ogbomoso, Nigeria. E-.mail: ooadeosun@lautech.edu.ng.
- Olajide Blessing Olajide has a B.Tech in Computer Engineering and currently pursuing his masters degree in Computer Science in Ladoke Akintola University of Technology, Ogbomoso, Nigeria. E-mail: olajideblessing55@gmail.com
- Adeosun Titilayo Helen is currently pursuing her Doctorate Degree in the Department of Management and Accounting of Ladoke Akintola University of Technology, Ogbomoso, Nigeria. E-mail: hootadeosun@yahoo.com