Journal of Innovative Research in Engineering Sciences ; Vol.4, No.3; 2018
ISSN 2476-7611

Mobility Models and Middleware Support for Mobile AdHoc Networks

Motahhareh Moravvej Hamedani
University of Tehran, Electrical and Computer Engineering Master Graduated,
m.moravej@ut.ac.ir

Received: 14, June , 2018 *Accepted: 15, July, 2018* *Online Published: 25, September, 2018*

Abstract
Embedded systems are increasingly expected to provide good performance at low cost. As the characteristics of compiled code can have an impact on the overall cost of an embedded system, a compilation strategy must be cost aware as well as achieve high performance. As one major factor for system cost is their execution time. On the other hand, code size is an important issue due to limited memory of some embedded systems. Thus, cost-effective optimization strategies that are able to perform a good trade-off between code size and performance improvement are needed. Mobile ad-hoc networks (MANETs) are random, self-configurable and rapidly deployable networks without any infrastructure. The data is moved between the memory and these registers by means of Load and Store instructions. The word length of the Nios II processor is 32 bits. All registers are 32 bits long. Byte addresses in a 32-bit word are assigned in little-endian style, in which the lower byte addresses are used for the less significant bytes of the word. The Nios II architecture uses separate instruction and data buses, which is often referred to as the Harvard architecture. In this paper, we review some existed mobility models of ad-hoc network and compare such methods with each other. Also, at the end of this paper, we will suggest new novel model, which has more optimality in comparison with the existed models.

Keywords: Ad-Hoc, Mobility, Middleware support

1. INTRODUCTION

In order to thoroughly simulate a new protocol for an ad hoc network, it is imperative to use a mobility model that accurately represents the mobile nodes (MNs) that will eventually utilize the given protocol. Only in this type of scenario it is possible to determine whether or not the proposed protocol will be useful when implemented. Currently there are two types of mobility models used in the simulation of networks: traces and synthetic models. Traces are those mobility patterns that are observed in real life systems [1, 2]. Traces provide accurate information, especially when they involve a large number of participants and an appropriately long observation period. However, new network environments (e.g. ad hoc networks) are not easily modeled if traces have not yet been created. In this type of situation it is necessary to use synthetic models. Synthetic models attempt to realistically represent the behaviors of MNs without the use of traces. In this paper, we present several synthetic mobility models that have been proposed for (or used in) the performance evaluation of ad hoc network protocols. A mobility model should attempt to mimic the movements of real MNs. Changes in speed and direction must occur and they must occur in reasonable time slots. For example, we would not want MNs to travel in straight lines at constant speeds throughout the course of the entire simulation because real MNs would not travel in such a restricted manner [2]. In the following, we discuss six different synthetic entity mobility models for ad hoc networks:
1. Random Walk Mobility Model (including its many derivatives): A simple mobility model based on random directions and speeds.
2. Random Waypoint Mobility Model: A model that includes pause times between changes in destination and speed.
3. Random Direction Mobility Model: A model that forces MNs to travel to the edge of the simulation area before changing direction and speed.
4. A Probabilistic Version of the Random Walk Mobility Model: A model that utilizes a set of probabilities to determine the next position of an MN [3].
5. City Section Mobility Model: A simulation area that represents streets within a city.
6. Obstacle Mobility Model: A real simulation area which includes obstacles.

2. PROBLEM DEFINITION

Mobile ad-hoc networks (MANETs) are random, self- configurable and rapidly deployable networks with out any infrastructure. MANETs typically exhibit high variability in network topology and communication quality. The dynamic (and temporary) paths are used by the nodes to transmit packets (messages). In such a network, the nodes communicate directly with each other within their wireless transmission ranges as mainly discussed in [10]. The main goal of developing the MANETs is not only obtaining better service, but also having networks that can serve in situations in which no other means of communications can operate. Examples include networks that are used in battlefields, in search-and

rescue operations. One of the most important features of any wireless network is the ability to provide some means of complete or partial connectivity among the communicating nodes. But

Corresponding Author. Mail. m.moravej@ut.ac.ir
Tell: +989128389521

this connectivity will changed according to mobile nodes movement, changing of bandwidth in arbitrary modes, Topology change may be frequent and Link capacity varies widely over time and etc[9].

In lots of MANET scenarios we want to have reliable network which can achieve its goal according to the tasks that are assigned to each mobile nodes. We define the death of a mobile node in the situation of consuming its battery power or facing attack from other enemy nodes and loosing the range of communication according to its mobility and direction. Following we highlight the most important parameters required for consideration in highly reliable and fault tolerant MANETs [4].

2.1. Resource Management

Participation in a MANET places a burden on a node, in addition to its primary functions. Each node is asked to also forward packets for the rest of the network. This consumes some or all of a node's resources, including processing time, energy stores, and network bandwidth. Depending on the amount of routing demand placed on a node, the additional routing load may reduce or drain a node's available resources. A node may then be forced to restrict its own operations, or even shut down, abandoning all functionality [5].

In mobile nodes the resource is limited so we should have some efficient fault tolerant mechanisms in adjusting transmission range and other mentioned parameters to better utilize our battery resource.

2.2. Transmission Range

Adjusting transmission range is important in battery usage and also in increasing or decreasing the probability of facing attacks. [1]

2.3. Challenges in Fault tolerant middleware design

The fundamental challenge in designing a middleware that provides an integrated support for high availability, fault tolerance, security, and attack tolerance is integrating the existing fault-tolerance techniques and security techniques. Integrating these techniques is an extremely difficult task. Some of the important reasons for this difficulty are [6, 7]:

1. Fault-tolerance techniques rely on redundancy in a system. For example, object replication, which is a popular technique to deal with object failures, results in redundant objects in a system [4]. The key idea these techniques exploit is that if a component fails, another component can take over the failed component's functionalities. However, redundancy in a system increases the vulnerability of that system to security violations. As a result, security techniques typically attempt to avoid redundancy in a system.

2. Security techniques typically rely on restricting access to a service and provide access to an entity only after extensive credential checks. For example, a mobile agent is allowed to run on a server only after it provides sufficient credentials in the form of certificates .Fault-tolerance techniques, on the other hand, rely on the open ness of a system. These techniques make decisions based on observing the behavior of a system. For example, failure detectors observe the external behavior of an object to determine object failures.

3. Security techniques often attempt to hide the identity of a system by purposely providing false information. Fault tolerance techniques; on the other hand rely on a cooperative interaction between redundant entities [8].

3. MOBILITY MODELS

3.1. Random Walk

3.1.1. Main Idea

The Random Walk Mobility Model was first described mathematically by Einstein in 1926. Since many entities in nature move in extremely unpredictable ways, the Random Walk Mobility Model was developed to mimic this erratic movement [1]. In this mobility model, an MN moves from its current location to a new location by randomly choosing a direction and speed in which to travel. The new speed and direction are both chosen from predefined ranges, [speedmin, speedmax] and $[0, 2\pi]$ respectively. Each movement in the Random Walk Mobility Model occurs in either a constant time interval t or a constant distance traveled d, at the end of which a new direction and speed are calculated. If an MN which moves according to this model reaches a simulation boundary, it “bounces” off the simulation border with an angle determined by the incoming direction. The MN then continues along this new path. Many derivatives of the Random Walk Mobility Model have been developed including the 1-D, 2-D, 3-D, and d-D walks. In 1921, Polya proved that a random walk on a one or two dimensional [9].

3.2. ACTIVE RESEARCH AREAS

During my long term survey I could find lots of new borne Research areas in designing fault tolerant middleware for systems in their special domains like Fault Tolerant Middleware for Agent Systems [6] or Byzantine-Fault- Tolerant Middleware for Web-Service Applications [7] and etc which you can find their mentioned papers through the references.

In the following I brought some of their summaries to highlight the desirability of this topic and the way of facing to these challenges for solving fault tolerant middleware design in each domain specific problem [6, 8].

The paper "A Middleware for Constructing Highly Available, Fault Tolerant, and Attack Tolerant Services"[2] describes the design of a middleware that provides support for constructing highly available, secure, fault-tolerant, and attack-tolerant services. The central component of this middleware is a group communication service that comprises of six network protocols: atomic broadcast, group membership, failure detection, attack detection, group access control, and secure inter member communication protocols including the motivation behind each of these protocols and discusses their functionalities.

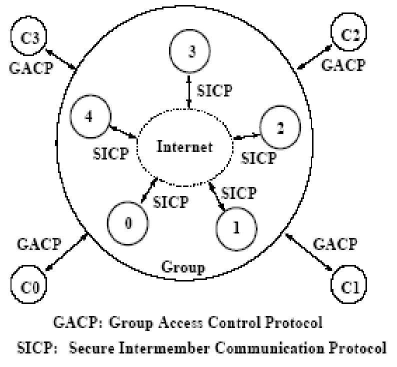
C3
GACP
C2
GACP
3
SICP
4
2
Internet
SICP
SICP
SICP
SICP
0
1
GACP
GACP
Group
C0
C1
GACP: Group Access Control Protocol
SICP: Secure Intermember Communication Protocol

Fig.1 Preliminary middleware architecture

Figure 1 shows the architecture of the proposed middleware. The central component of this middleware is a group communication service. Clients interact with a group (one of the group members) using a special protocol called the Group Access Control Protocol (GACP). This protocol is intended to allow secure communication between trusted clients and trusted groups. Group members interact with one another using a special protocol called the Secure Inter member Communication Protocol (SICP). Again, this protocol is intended to provide support for secure communication between trusted group members.

On the other hand in the area of embedded real time systems,The paper "Implementation of middleware fault tolerance support for real-time embedded applications" [3,4] is one of the papers that I have read. It explains an on going work to provide application fault tolerance by means of implementing middleware transparent support over the BOSS embedded operating system. The middleware uses a publisher subscriber protocol and enables the execution of several fault tolerance strategies with minimum burden to the application level software.

In the field of automotive software design, newly researches have been done in the paper "Designing Real-Time and Fault-Tolerant Middleware for Automotive Software" [5]. The contributions in this paper are two folds. First, they have identified five essential requirements that must be satisfied by automotive middleware. These are resource management [4];

communication model specialized for automotive network, fault-tolerance, a global time base, and resource frugality. Second, they have presented a middleware structure satisfying these requirements. They have designed the middleware supports the publish/subscribe communication model based on message transmission which also includes a QoS configuration and static resource allocation mechanism, a clock synchronization mechanism, and various fault-tolerance algorithms to achieve real-time data transmission, fault- tolerance, and a global time base. Figure NUM shows the structure of the proposed middleware for automotive systems. At the top level, there exists the publish/subscribe interface that is used by application components. Beneath the interface, there is QoS Configuration module for specifying real-time requirements, Resource Allocation module for guaranteeing timeliness operations, and Clock Synchronization module for providing a global time base. All incoming and outgoing messages pass through Fault-Tolerance Layer in order to guarantee reliability. Finally, messages are transmitted and received by Transport Layer [3].

All an all I did not find any research team working on the subject of designing fault tolerant middleware for MANETs. It seems a not followed road that worth noting on.

In the following sections I try to highlight my ideas for mapping one of fault tolerant strategies for MANETs(figure 2).

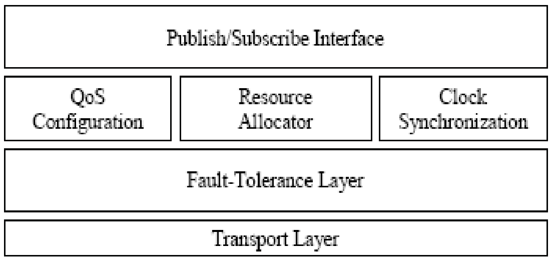

Fig.2 Proposed design of fault tolerant automotive middleware

E. FAULT TOLERANT STRATEGIES

Several FT strategies have been proposed and applied in the last 30 years. I explain some of the important ones. The simplest strategy is Rollback/Retry, also called"checkpoint and restart", which uses time redundancy. This strategy is effective only against transient faults, like hardware transient faults caused by electromagnetic radiation, and even some software transient faults like as race conditions. In order to deal with permanent software faults, other strategies have been proposed as Recovery Blocks (RB), Distributed Recovery Blocks (DRB) and N-Version Programming (NVP) [5].

RB and DRB perform backward error recovery like Rollback/Retry, but use different software versions, or variants, in each execution block. In these techniques there are at least two software versions in RB and two software versions in DRB, which must deliver similar correct results.

The main difference between RB and DRB is the distributed nature of the later, allowing concurrent running of variants in two distinct nodes and coordination between them to define what node will send the final output.

NVP is a FT strategy that users forward error recovery in which multiple variants (at least 3) run sequentially or concurrently and a decision mechanism selects the correct response usually by majority voting. In a multi-computer system, each variant runs in a different node and the decision mechanism (voter) may be replicated too. According to the specific characteristics of MANETs, NVP strategies with redundancy in mobile nodes executing different versions are considered. Trough this design, we should select different nodes dynamically to run desired versions in different situations [6].

For example in the specific situation of bandwidth or battery resource, some of the versions are desired to run; but according to mobility of nodes or Ad hoc situations some other versions in different mobile nodes are preferred to run. In my proposed design selecting mobile nodes for running desired tasks should be dynamic according to transmission range of each node and the level of criticality of tasks. It means I want each node reconfigure itself with each situation through adjusting transmission range and selecting the versions that should be run which satisfy the requirements of mobile Ad hoc network in the specific period of time. I include both dynamic selections of mobile nodes in a vicinity of a node in order to run N Copy programming which adds the benefits of hardware redundancy and including N version programming.

F. PERCOLATION MODEL FOR MANETS

In [1] Hossein Mohamadi et.al proposed a percolation model for studying the properties of the MANETs. As is well- known, percolation theory, describes the effect of the connectivity of the microscopic elements of a disordered system on its macroscopic properties.

The classical percolation models are mostly based on lattice models, in which a randomly-selected fraction of the nodes (or bonds) of a lattice are active (i.e., open to a given phenomenon, such as transport of electrical charges, or receiving messages through similar active sites) with probability p, or inactive or removed with probability $1-p$.

For $p > pc$, where pc is the percolation threshold, a SSC of the connected active sites is formed in which a macroscopic phenomenon can occur. Given this brief description of the percolation clusters on random lattices, their similarity with the MANETs becomes evident. The active sites in a percolation system may be thought of as the nodes in a MANET.

The model is based on a random network of sites, distributed in space, which represent the mobile nodes. Two nodes are linked if they are within each other’s transmission ranges. A node may be lost or become inactive if, for example, it runs out of energy (provided by its batteries). A link can be lost if, for example, one of its two end nodes moves outside of the other’s transmission range.

The dynamic (and temporary) paths are used by the nodes to transmit packets (messages). In such a network, the nodes communicate directly with each other within their wireless transmission ranges.

Sometimes the goal of the network is not necessarily interconnecting the nodes, but to convey some control commands to a few particular actors. For example, consider a military network of sensors guiding an unmanned autonomous vehicle (UAV), or a civil network of monitoring potential fires, using mobile sensors. The common feature of all such cases is the need to doing the task that is assigned to the special group of nodes that are present in the desired part of area.

For increasing reliability against faults that happen during the changing behavior of mobile nodes in Ad hoc networks, it is highly desirable that each node configures itself and run the desired versions. On the other hand with including diversity with N copy programming ideas we can achieve availability when one node attacks or uses hole of battery resource if we could run the mentioned task on the other mobile node that is in the vicinity of this node.

So I choose the percolation model for modeling the behavior of MANETs to be able to achieve global goals in the network based on setting local behaviors of mobile nodes through selection of desired versions that highly match with happened situation, because we can achieve global evaluation trough local data using this model.

G. PROPOSED MODEL

As described in the previous sections, the main idea is supporting fault tolerance strategies in middleware layer for minimizing the overhead and burden that may appear for the application.

Its mobile node architecture is based on my previous kernel design for mobile node in Ad hoc networks. This architecture is based on Exo-kenrel and has cross layer design architecture as shown in figure 3. This model is designed according to special requirements of fourth generation mobile nodes that these days' operating systems can not satisfy them. It also has some level of adaptability and configurability according to library OSs and user defined processes.

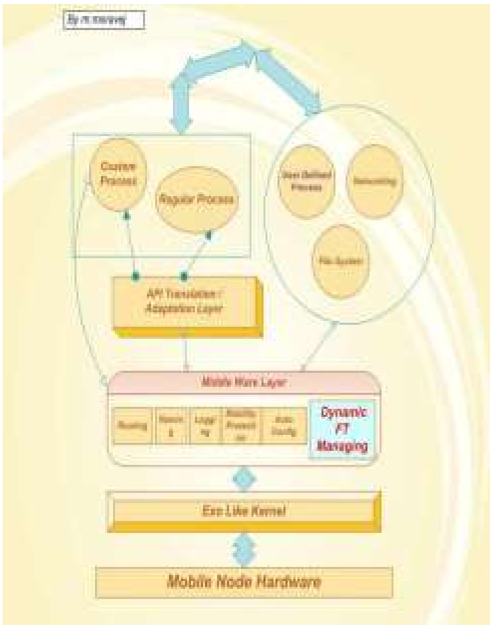

Fig. 3 kernel design for mobile node in Ad hoc network

In this designed model I added a new module with name Dynamic FT Managing to the pre designed model that is a coordinator and runner of fault tolerant strategies in mobile nodes and NVP in a single node or with diversity between other nodes. Setting deadline time for receiving the result from other node is required, because according to mobility of nodes, one node may become outside of transmission or receiving range of the other. So it can not send its computed result to the runner node of NVP with diversity. In the following this subject is explained in detail.

H. ANALYTICAL MESSURMENTS

In the proposed model we need an analytical evaluation for the reliability measurements of this dynamic NVP with consideration of including deadline for receiving the result when we have diversity.

A. Reliability analysis of N-version programming with deadline mechanism

In this section, the developed reliability model for N- version programming with deadline mechanism. The versions are independent and there is a voting mechanism that labels the majority as correct output. The maximum time available for a version to complete execution is t. Let us assume that the execution time of version i is a random variable with density function fi (t).Version that fails to complete execution by time t is discarded and is considered a failure [9].

Now the reliability of N-version programming with deadline mechanism can be derived as follows.

Let:

P (Ga) = P (two or more outputs agree). P (Gc) = P (recurring output is correct).

P (Dn) = P (all the outputs are different).

Then the reliability of N-version programming with deadline mechanism is given by:

$$R_n = (1 - P(D_n))P(G_c)$$

For simplicity let us assume that P(Gc) = 1, then:

$$R_n = 1 - P(D_n)$$

Expression for P(Dn) can be derived as follows.

From the definition an N-version programming fails when all the outputs are different. This can happen in the following two ways:

(1) Only one version produces correct output by time t.

(2) All the outputs, which are provided by time t, are different.

Let:

$$\alpha_i = p_i F_i(t) + (1 - F_i(t))$$

$$P(D_n) = \sum_{i=1}^{n} \frac{(1-p_i)\,\mathrm{F}_i(t)}{\alpha_i}\left[\prod_{k=1}^{n}\alpha_k\right] + \prod_{i=1}^{n}\alpha_i$$

Where, Fi(t) denote the probability that version I produces the output by time t. The reliability of an N-version programming with deadline mechanism is given by:

$$R_n = 1 - \sum_{i=1}^{n} \frac{(1-p_i)\,\mathrm{F}_i(t)}{\alpha_i}\left[\prod_{k=1}^{n}\alpha_k\right] + \prod_{i=1}^{n}\alpha_i$$

In the basic mechanism, we assume a free space propagation model [10], where the received signal strength solely depends on its distance to the transmitter. We also assume that all nodes in the network have their clock synchronized; therefore, if the motion parameters of two neighbors (such as speed, direction, and radio propagation range) are known, we can determine the duration of time these two nodes will remain connected. Assume two nodes i and j are within the transmission range of each other [8. Let (xi;yi) be the coordinate of mobile host I and (xj;yj) be that of mobile host j. also let vi and vj be the speeds and θi and θj where:

$$\alpha_i = p_i F_i(t) + (1 - F_i(t)), \qquad (0 \le \theta_i, \theta_j \prec$$

Be the moving directions of nodes i and j, respectively.Then,th amount of time two mobile hosts will stay connected, Dt is predicted by:

$$D_t = \frac{-(ab+cd) + \sqrt{(a^2+c^2)r^2 - (ad-bc)^2}}{a^2+c^2}$$

Where:

$$a = \upsilon_i \cos\theta_i - \upsilon_j \cos\theta_j$$

$$b = \chi_i - \chi_j$$

$$c = \upsilon_i \sin\theta_i - \upsilon_j \sin\theta_j, and$$

$$d = y_i - y_j$$

The predicted value is the link expiration time (LET)

between the two nodes.

I. CONCLUSIONS

In the previous sections I tried to highlight the importance of having high reliability and availability in MANETs. Achieving these goals and other specific considerations of such networks was a motivation for me to focus on the design principles and modeling of fault tolerant strategies. I selected dynamic NVP with diversity of mobile nodes both because of hardware redundancy and software redundancy. I proposed a model to support the above strategies in middleware layer because of reducing complexity of management and ease control and use of predefined modules in that layer.

Ideas on finding the system reliability trough component reliability can be applied in this topic to find the whole reliability of network whit reliability of dynamic selected mobile nodes as a cluster.

According to lack of simulation software for MANETs and in middleware layer, I will focus on finding applicable way of evaluating this design model and compare it with previous one which did not support any fault tolerance strategies.

FUNDING/SUPPORT
Not mentioned any Funding/Support by authors.

ACKNOWLEDGMENT
Not mentioned.

AUTHORS CONTRIBUTION
This work was carried out in collaboration among all authors.

CONFLICT OF INTEREST
The author (s) declared no potential conflicts of interests with respect to the authorship and/or publication of this paper.

REFERENCES

1. Shen WL, Chen CS, Lin KC, Hua KA. Autonomous mobile mesh networks. IEEE Transactions on Mobile Computing. 2014 Feb;13(2):364-76.[Scholar]
2. Akyildiz IF, Wang X, Wang W. Wireless mesh networks: a survey. Computer networks. 2005 Mar 15;47(4):445-87.[Scholar]
3. Afonso F, Silva CA, Montenegro S, Tavares A. Implementation of middleware fault tolerance support for real-time embedded applications.[Scholar]
4. Afonso F, Silva C, Montenegro S, Tavares A. Middleware Fault Tolerance Support for the BOSS Embedded Operating System. InIntelligent Solutions in Embedded Systems, 2006 International Workshop on 2006 Jun 30 (pp. 1-12). IEEE.. [Scholar]
5. Park J, Kim S, Yoo W, Hong S. Designing real-time and fault-tolerant middleware for automotive software. InSICE-ICASE, 2006. International Joint Conference 2006 Oct 18 (pp. 4409-4413). IEEE. [Scholar]
6. Laibinis L, Troubitsyna E, Iliasov A, Romanovsky AB. Fault tolerant middleware for agent systems: a refinement approach. In12th European Workshop on Dependable Computing, EWDC 2009 2009 May 14 (pp. 7-pages). [Scholar]
7. Merideth MG, Iyengar A, Mikalsen T, Tai S, Rouvellou I, Narasimhan P. Thema: Byzantine-fault-tolerant middleware for web-service applications. InReliable Distributed Systems, 2005. SRDS 2005. 24th IEEE Symposium on 2005 Oct 26 (pp. 131-140). IEEE.. [Scholar]
8. Su W, Lee SJ, Gerla M. Mobility prediction in wireless networks. InMILCOM 2000. 21st Century Military Communications Conference Proceedings 2000 (Vol. 1, pp. 491-495). IEEE. [Scholar]
9. Dinesh Kumar U. Reliability analysis of N-version programming with deadline mechanism. International Journal of Quality & Reliability Management. 2000 Apr 1;17(3):276-84. [Scholar]
10. Hadim S, Al-Jaroodi J, Mohamed N. Trends in middleware for mobile ad hoc networks. JCM. 2006 Jul;1(4):11-21. [Scholar]