GROUP MUTUAL EXCLUSION IN OPPORTUNISTIC NETWORK

A thesis presented to the Department of Computer Science

African University of Science and Technology, Abuja

In partial fulfillment of the requirements for the award

MASTER OF SCIENCE DEGREE IN COMPUTER SCIENCE

By

ASULBA BARIKISU AHMED

Supervised by

Prof. Ousmane Thiare

African University of Science and Technology
www.aust.edu.ng
P.M.B 681, Garki, Abuja F.C.T
Nigeria

June, 2016

GROUP MUTUAL EXCLUSION IN OPPORTUNISTIC NETWORK

By

Asulba Barikisu Ahmed

A THESIS APPROVED BY THE COMPUTER SCIENCE DEPARTMENT

RECOMMENDED:

_____

Supervisor, Professor Ousmane Thiare

_____

Head, Department of Computer Science

APPROVED:

_____

Chief Academic Officer

_____

Date

ABSTRACT

The Opportunistic network is an interesting development in the Mobile Ad hoc Network (MANET) environment. It has no end-to-end connectivity among nodes.

Unlike MANETs, the nodes in Opportunistic network are independent on network topology. Resources are constrained, and nodes share resources in this type of network. Hence, to ensure the integrity of nodes wishing to access a shared resource, mutual exclusion is required to allow nodes to access shared resources exclusively.

In this thesis, we review an extension of the mutual exclusion problem known as, the Group Mutual Exclusion (GME) for MANETs, and evaluate their applicability to Opportunistic network. We further propose a token based Group Mutual Exclusion Algorithm for Opportunistic network. The MEOP algorithm in [20] is adapted for the proposed algorithm and to ensure concurrent execution of critical section, a similar approach is adopted from [21], [9]]. The algorithm ensures mutual exclusion, bounded delay and concurrent entering properties.

ACKNOWLEDGMENT

All praise and thanks to Almighty Allah, The all Knowing.

I am deeply indebted to Professor Ousmane Thiare, my thesis supervisor, for introducing me to this area of knowledge, for his persistent guidance and support throughout the research work. This thesis would not have come this far without his insightful comments, knowledge and experience. Indeed I have learned many skills useful for future research works.

My appreciation goes to the African Capacity Building Fund (ACBF), for granting me a scholarship to undertake a Master's degree in Computer Science. I may not have enrolled in the program without such assistance.

I would also like to thank all visiting faculties to the Computer Science department for impacting me with their knowledge, guidance and motivation to undertake this degree. And also, not forgetting my course mates for their massive contribution and encouragement, I am really grateful for their support. Also, to the management and staff of African University of Science and Technology, I am indeed grateful for their kind support and providing very helpful resources for research and a good atmosphere for sound learning.

Last but most important, I profoundly thank my family for their tremendous support. Every member of my family has been so supportive in building my career. Both parents have patiently supported and guided my pursuit of knowledge. My husband and daughter are still the best

DEDICATION

To my lovely Mum, Asulba Mamata.

Table of Contents

List of Figures

List of Tables

Table of Abbreviations

CS	**Critical section**
DAG	**Direct Acrylic graph**
GME	**Group Mutual Exclusion**
MEOP	**Mutual Exclusion for OppNet**
NCS	**Non Critical Section**
MANET	**Mobile Ad hoc Network Environment**
PDA	**Personal Digital Assistant**
OppNet	**Opportunistic Network**

CHAPTER 1

1.0 Introduction

Opportunistic Network is an interesting development in Mobile ad hoc Network (MANET) environment and a promising technology in achieving the vision of pervasive computing. The Opportunistic network [16] is designed from mobile wireless network devices that have good sensing capabilities, good memory, and short radio transmission functionality. These devices are usually carried by human, animal, vehicles, among many others.

The exceptions in Opportunistic Network such as network failure, node failure and infrequent node contact mostly result from battery failure and power management of these devices. Regardless, the mobility of these devices is used as an advantage to create an Opportunistic network. The mobility nature of these wireless devices is utilized to create communication between nodes when route connecting them never exited or there is no direct contact with the internet.

The network is usually partitioned into regions; nodes are interconnected by operating in a store-carry forward manner. A node can store carry and forward messages within the same region or different regions acting as a router or a gateway respectively, node can also be a host where data is finally stored [6].

The nodes in Opportunistic Network are independent of the network topology but not the case in MANET. Routes in Opportunistic Network are not predetermined. While a message is in route from source to destination, all nodes have the opportunity to serve as the next hop provided it is closer to the destination.

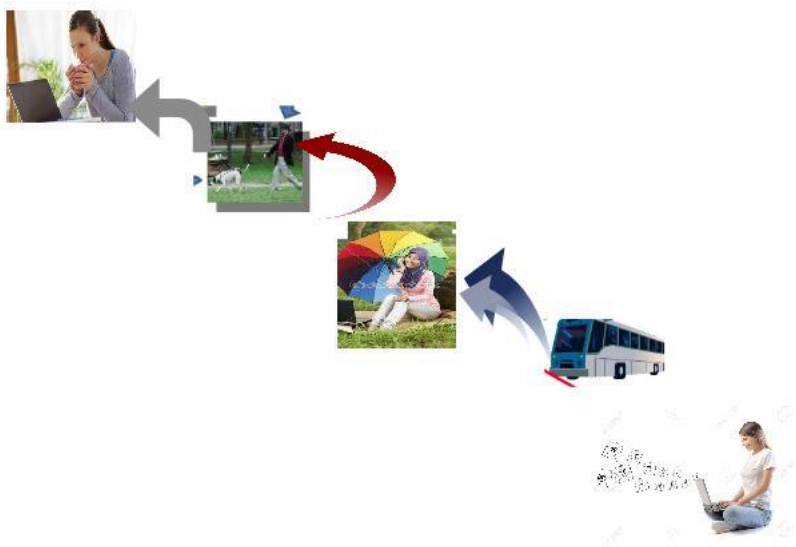

Figure 1.1: Sending an email using Opportunistic Network

Illustrated in Figure 1.1 [15], is an example that describes how opportunistic network operates in real life. A lady with a laptop at the bottom end in Figure 1.1 is trying to send a mail to a friend. The following steps ensure delivery of the message;

- The lady transfers the mail via Wi-Fi link to the bus passing within the area hoping the bus will transfer the message closer to her friend.
- A device somewhere in the bus then transfers the message to the lady with a phone along the way, also hoping it is closer to the intended destination.
- The lady's phone identifies a wireless network device carried by a pet and transfers the message to it.
- The man not far from the destination finally transfers the email to her friend via Wi-Fi link

The mobility nature of the network has drawn the attention of researchers to focus more on routing and data forwarding. The routing protocol in Opportunistic Network is classified into three categories: Mobility class, context–oblivious and social context–aware routing [5]. In [10] Lilien discussed the challenges in security and privacy in Opportunistic network.

1.1 Problem Formulation

An Opportunistic network is a typified distributed system and resources are constrained in distributed systems, thus processes or nodes share resources. To ensure the integrity of nodes wishing to access a shared resource, *mutual exclusion* is required. This allows nodes to exclusively access a shared resources, in other words to *execute critical section*. Mutual exclusion ensures that, at most one node executes the critical section at a time. Mutual exclusion algorithms are evaluated by the number of messages generated per critical section entry, synchronization delay, concurrency and size of information control.

Researchers have limited attention to this problem area in Opportunistic network. Tamhane in [20] proposed and simulated a novel token based Mutual Exclusion algorithm for Opportunistic network.

Over the past few decades, an interesting extension of the mutual exclusion problem known as the Group Mutual Exclusion (GME) problem, has been proposed by Joung. It is known as the ***Congenial Talking Philosopher*** [8]. Joung's major focus is to improve upon concurrent access to a critical section by, trying to avoid delays in processes waiting to access the same resources as those processes executing the critical section. This phenomenon allows resources to be shared by processes of same group but not processes of different groups. In other words, at most one group of processes executes their critical section concurrently.

A CD jukebox is a key example of the Group Mutual Exclusion problem. Data is stored on disk,

and only one disk is loaded for access at a time, therefore, processes wishing to access same loaded disk can do so concurrently. To the best of our knowledge, no GME algorithm has been proposed for Opportunistic network.

1.2 Research Objectives

The main objectives of this Master's project are to;

i. Evaluate GME algorithm for MANET and discuss their applicability to Opportunistic net-work.

ii. Propose a GME Algorithm for Opportunistic network based on some existing Algorithm.

iii. Prove the algorithm satisfies the GME Properties.

1.3 Approach Adopted

The scientific approach employed to solve the stated problem and to meet our objectives includes:

i. Evaluating the proposed GME algorithm for MANET, and based on the assumptions of each algorithm, we will determine if it is applicable to Opportunistic network.

ii. Proposing an existing Algorithm that will be modified to suit the GME problem in OPPNET.

1.4 Organization of Work

The rest of the report is organized as follows; Chapter 2 gives a survey of related work in Group Mutual Exclusion problems, and evaluation of existing Algorithms for MANET. Chapter 3 defines the proposed Algorithm. Proof of the Algorithm will be presented in chapter 4. Chapter 5 provides conclusion and future work.

CHAPTER 2

2.0 Literature Review

In this chapter, the basic concept of Opportunistic Network is presented, and various Group Mutual Exclusion Algorithms developed in literature is reviewed. Also, evaluation of existing Algorithms proposed for Mobile ad hoc Network (MANETs) and analysis of their implementation to Opportunistic Network (OppNet) is examined.

2.1 Mobile Ad hoc Network (MANET)

MANET is an infrastructure-less network of mobile nodes connected by wireless links. Nodes have limited communication range but communicate directly with each other. Nodes move randomly which results in change in network topology. Multi-hop paths are employed to, route data from source to destination when nodes are out of communication range. MANETs provide an end-to-end routing protocol.

2.2 Concepts of Opportunistic Network

An Opportunistic Network is an extension of MANETs. Mobility of nodes in MANETS generate problems in the network. Therefore, the idea was conceived to utilize the mobility of nodes to create opportunistic paths for data dissemination. Opportunistic Networks like MANETs, are self-configuring mobile wireless connected nodes.

2.2.1 Node Definition

A node [[13][15]] in Opportunistic network, is a device with wireless network capabilities. an example of such devices include; PDA, smart phones, etc. The nodes perform the following functionality in the network;

- Node discovery or search opportunities: A node has the ability to discover or identify other nodes with similar capabilities within its communication range. The communication range is estimated to be within 100-200 meter walking distance.
- After discovery of other nodes, a node can further exchange messages within its communication range

2.3 Difference between Mobile Ad hoc Network and Opportunistic Network

Table 1.1 outlines a summary of the major difference between Opportunistic Network and MANET.

Table 1.1 Comparisons between MANETs and OppNet

Mobile Ad hoc Network (MANET)	Opportunistic Network (OppNet)
Mobile ad hoc Network provide an end-to-end connectivity for communicate i.e. a complete path is defined in sending and receiving data	In opportunistic network there is no complete path complete path between two nodes wishing to communicate
MANET recites in network layer and has a well-defined routing proto-col for communication.	Unlike MANETs, opportunistic network recites on application layer [3] data forwarding and routing protocols are merged because routes are only built when messages are disseminated.
MANET recites in network layer and has a well-defined routing proto-col for communication.	It uses a store-carry-and-forward protocol in sending or receiving data [6]. Mobility [6] is an advantage to create a network using opportunistic contacts.

2.4 The Group Mutual Exclusion (GME) Problem

The Group Mutual Exclusion problem [8] is an extension of the Mutual Exclusion problem. The idea was conceived by Joung to improve on concurrent access to the critical section. It was modeled as the congenial talking philosopher.

It was motivated by considering a number of philosophers either thinking, or talking, and waiting to be in a meeting room. A set of N philosophers, M fora, and one meeting room were considered. The principle was that more than one philosopher can be in a forum but at most one forum can be in meeting at a time. This was proposed for static or shared models.

Hadzilacos [4] identified the fairness property in the Joung algorithm to be weak and proposed an algorithm that improved the fairness property of the Joung algorithm. His algorithm ensures that access to critical section is granted in order of request. In addition, processes do not require advance knowledge of the type of sessions before requesting for a session.

For applications with non-uniform groups i.e. some groups frequently accessing the critical section more than others, [14] proposed an algorithm to handle such case of non-uniformity. Simulation results showed that the algorithm out performs other traditional algorithms in application to non-uniform group scenarios.

The role GME proposed in [2] is an extension of GME problem. It ensures a process selects its role, either shared or exclusive, before selecting a group to join in order to execute the critical

section.

A cluster-based method was adopted to develop a hierarchical GME algorithm proposed in [18]. The algorithm improved on message complexity compared to traditional algorithms. In the algorithm, nodes are divided into clusters and each cluster has a cluster head known as a coordinator. The algorithm uses clusters to improve the message complexity than traditional algorithm. Thus, messages per access to critical section is dependent on a number of clusters, unlike existing algorithm that it is dependent on the entire nodes.

In literature, the GME algorithms for distributed systems can be categorized into, token based and permission-based. The permission-based algorithm such as the proposed algorithms by ([22], [12] and [1]) uses quorum based protocol. A request is sent to all processes in the same quorum and waits for permission to be granted before a process executes its critical section.

In the token based algorithms as that proposed in [11], a unique token is maintained with varying sub-tokens. A process executes critical section only if it has a token, then the token holder sends sub tokens to other processes requesting to enter the critical section.

2.5 Evaluation of MANET GME Algorithms

In this section, we give brief explanation of existing Group Mutual Exclusion Algorithms proposed for Mobile ad hoc network, and evaluate their applicability to Opportunistic network.

2.5.1 A Token–Based Group Mutual Exclusion Algorithm for MANETs

This section explains the GME algorithm proposed for MANET in **[21].**

The algorithm is token based and a direct acrylic graph (DAG) is maintained to, forward and request for token such that every non–token holder has a direct link to a token-holder. Each node i is identified by a height which is in a triple form as [a, r, i].

A link is outgoing from a high height to a low height. The higher the height of node, the higher the distance it is away from the token-holder. All requests are sent to the lowest node, usually the token-holder or leader. The algorithm circulates in three states as shown in Figure 2.

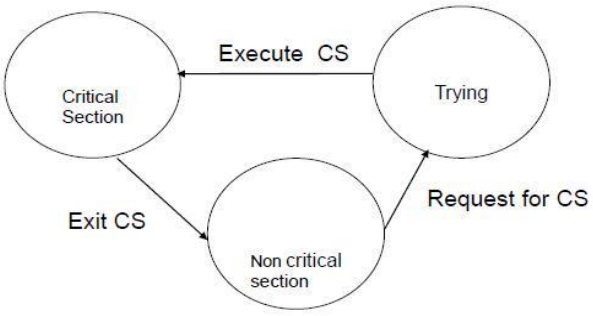

Figure 2: State diagram of Algorithm

A node in a Non-Critical Section (NCS) transit to trying state if it request for critical section and waits for feedback. When in trying state a node transit to critical section (CS) state if it receives token and becomes a leader or receives an okay from leader. When transiting from a critical section to non - critical section, if it is leader it forwards the token to the next in its re-quest queue else it sends a release message to its neighboring node indicating it exits the critical section.

The Token–Based Group Mutual Exclusion algorithm uses a unique token to ensure mutual exclusion. Concurrency is satisfied by ensuring the token-holder sends an okay message to neighboring nodes to allow them access critical section concurrently only if their request is the same as the resource in a critical section. A node, upon receiving token reduces its height value to become the lowest node in the DAG structure. It sends out an okay with resource id to all nodes in its request queue.

The DAG structure requires a frequent update of network topology, and this will pose a great challenge in applying to Opportunistic network due to its dynamic routing protocol.

The algorithm assumes no permanent partition of network and node failure is assumed not to occur. This is not applicable to Opportunistic network due to its mobility nature, even though the algorithm is tolerant to link failure.

2.5.2 A Group Mutual Exclusion Algorithm for Ad Hoc Mobile Networks

The algorithm proposed in [7], maintains a Direct acrylic graph(DAG) for message passing similar to algorithm in [21], but the algorithm implements a weight throwing mechanism to ensure concurrency. A node, with a token in critical section sends a sub-token with the stamp of re-source and weight to every requesting neighbor. Upon receiving the sub token, a node then sends fractions of the sub-token to all nodes in its queue. A sub-token holder can exit the critical section if only all fractions of sub-token are released. So for each sub-token, weight is incremented by one. A sub-token holder releases weight and weight is decremented by one.

Similar to algorithm in [21], to avoid starvation during link failure, the height information is used to up-date the DAG structure. The algorithm ensures that a node receiving a request checks, if there is link failure to the link, it ignores the node by not adding to its request queue.

The algorithm will not be best for Opportunistic network based on similar assumptions with Thiare's algorithm in [21]. Also, due to mobility and inconsistency of node contacts in Opportunistic network, the weight throwing mechanism cannot be applicable to Opportunistic network.

2.5.3 A Token based Distributed Group Mutual Exclusion Algorithm with Quorums for MANET

The algorithm [**19**] is token- based, and the network is partitioned in quorums. Nodes are partitioned in quorums and the quorums are further partitioned into coterie, and at least every two quorums share a node as shown in Figure 3. A main token, uniquely maintained in the network generates a varying number of sub-tokens.

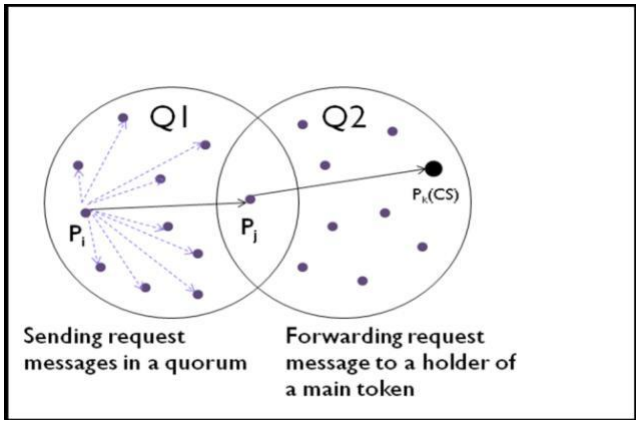

Figure 3: Node Request to Execute Critical Section

Shown in Figure 3, a node pi wishing to enter critical section sends request to each node pj in the quorum and waits to receive either of the tokens. A node Pj upon receiving the request, forwards the request if it identifies the token-holder as node Pk else it buffers the request. If the token-holder node Pk receives the request, it queues the same into its request list. The token is released if only it is not in critical section. If the resource request is same as the token-holder's resource, it sends a sub-token provided no other node in the queue has higher priority.

The Token based Distributed GME Algorithm avoids starvation during node failure by ensuring the main token is assigned to next node in its request list before it fails. Network simulator-2 is used to evaluate the performance of the Algorithm. The response time increases with increase in a number of nodes and decreases when it reaches a peak. The synchronization time varies but

decreases with increase in the number of nodes; whilst increase in mobility increases response time and synchronization delay.

The Algorithm was further compared with the Ricart–Agrawala algorithm for MANET and the Ricart–Agrawala algorithm for wired network. it out performs both algorithms in terms of synchronization delay and response time. The assumption then is, the Network is reliable but the Opportunistic network cannot be classified as a reliable network.

2.5.4 Arbitration Based Distributed Group Mutual Exclusion Algorithm for Mobile Ad hoc Network

In this algorithm proposed in **[17],** nodes are partitioned into quorums and coterie with similar assumption and operation as Talele et al Algorithm in [19]. The algorithm is an improvement of the Talele et al Algorithm in terms of message complexity. A look-ahead procedure is implemented to reduce message complex-ity .The look-ahead procedure ensures that, a node wishing to execute critical section sends request to only those nodes in its info-set.

At most, a node known as an **arbitrator node** intersects two quorums, and each node records other node data in its info-set; each info-set has information about the Arbitrator node. All requests are forwarded to the arbitrator node and only the arbitrator node forwards the request to the token-holder. The algorithm ensures that only the arbitrator node controls which node gets permission to access the critical section. Hence, to ensure mutual exclusion, only the arbitrator node forwards request to the token-holder. Upon, receiving a token, the token-holder then decides which node will be granted a token to access the critical section. A node, upon receiving the token sends sub-tokens to all nodes in its info-set, and only if the nodes in its info-set request for a similar resource, then they will execute critical section concurrently.

Simulation results show that the Algorithm outperforms the GME Algorithm proposed in [19], in terms of message complexity, response time and synchronization delay.

Due to node mobility in Opportunistic networking the concept of maintaining an arbitrator node cannot be applicable. To ensure starvation, the algorithm is assumed not to be fault tolerant such as node or link failure.

2.6 Summary

In summary, the algorithms explained above are all token based algorithm. This implies that a token is used to grant access to a critical section or shared resource. The Algorithms are categorized into:

- Direct Acrylic graph (DAG) based Group Mutual Exclusion Algorithm.
- Quorum Based Group Mutual Exclusion Algorithm.

2.6.1 Disadvantages of DAG based GME Algorithm

- Maintaining a DAG in the network for message passing cannot be applicable in opportunistic network due to random mobility of nodes.
- It requires frequent update of network topology but since routes from source to destination are not predetermined in opportunistic net-work topology update cannot be applicable.

2.6.2 Disadvantages of Quorum based GME Algorithms

- In opportunistic network nodes cannot be permanently positioned in a quorum due to random movement of nodes i.e, a node can be in any quorum at any time. Therefore, maintaining a quorum structure will be a challenge in opportunistic network.
- Also, maintaining a static arbitrator node cannot be applicable to Opportunistic network.

CHAPTER 3

3.0 Proposed Group Mutual Exclusion Algorithm for Opportunistic Network

In this chapter, we discussed the System model and proposed Algorithm. The model of the algorithm is modified from the MEOP algorithm in [20].It's comprised of four main parts; token generate, request generation, request propagation, and token propagation.

3.1 System Model

An Opportunistic network is an infrastructure-less wireless network where nodes are mobile. The system model and algorithm is adapted from the MEOP algorithm. In [20] a novel token based mutual exclusion algorithm known as MEOP (Mutual Exclusion for Opportunistic Net-work) was proposed. Similar assumptions proposed in the MEOP algorithm are adopted for the system model. The assumptions include:

- A nodei will be in communication range with a few nodes in the network as shown in Figure 4 Nodej is in communication range with node; e, f, g but not with node; i,c,d .

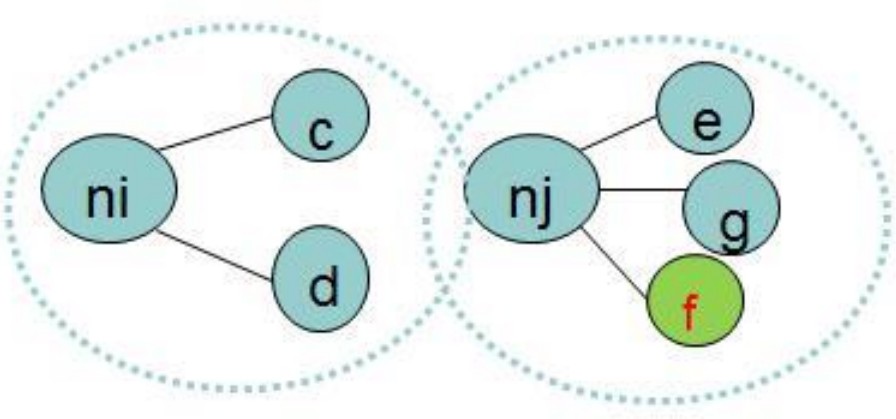

Figure 4: Node with same communication range communicate directly

- Nodes within the same communication range maintain a synchronous communication to detect message loss by any nodes and messages are transferred in First in First out (FIFO) order.
- To keep update with network topology, a bidirectional communication channel maintained. For example in Figure 5, assume nodei knows that nodef is token-holder and nodej had information that nodeg is token-holder. Upon opportunistic contact between nodei and nodej the exchange such information, i.e, bidirectional communication and based on the timestamp, one of the nodes updates its information about the token-holder location.
- Also, synchronous communication is maintained to ensure a message is delivered. Thus a

feedback is sent to indicate the token-holder, which is nodef is identified as in Figure 5.

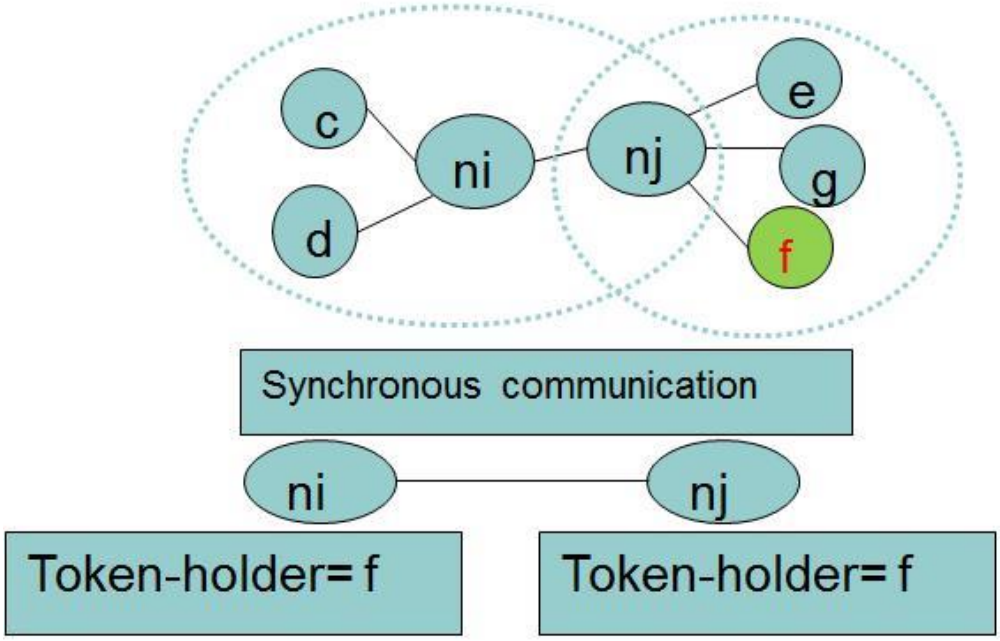

Figure 5: Synchronous communication between nodes.

- First In First Out (FIFO) order of message delivery is not preserved in multi-hop communication.
- The network is mostly partitioned but eventually, there will be a path between two nodes.
- A synchronization clock is implemented and clock drift is negligible.

In addition, we assume that nodes concurrently accessing the same resource terminate their tasks. This assumption is similar to one made in [7], for MANETs.

3.2 Working Principle of MEOP Algorithm

Figure 3.3 shows the state transition diagram of the MEOP Algorithm. If token-holder is known, a node initially in the **generated request state** transit to **spread request state** and waits for token. If a token is received, it transit to t**execute critical section (CS) state**. After terminating a task in critical section it can either **update the network** location of the token or transit to **transmit token state**. If the token-holder transmits token then, it can generate another request after a given time.

The MEOP algorithm implements a social context routing algorithm known as bubble Rap for message passing, using the inter contact time between two nodes as a criteria, a message from nodei to nodej will pass through nodek if and only if the inter contact time between nodek and nodej is less than the inter contact time between nodei and nodej .

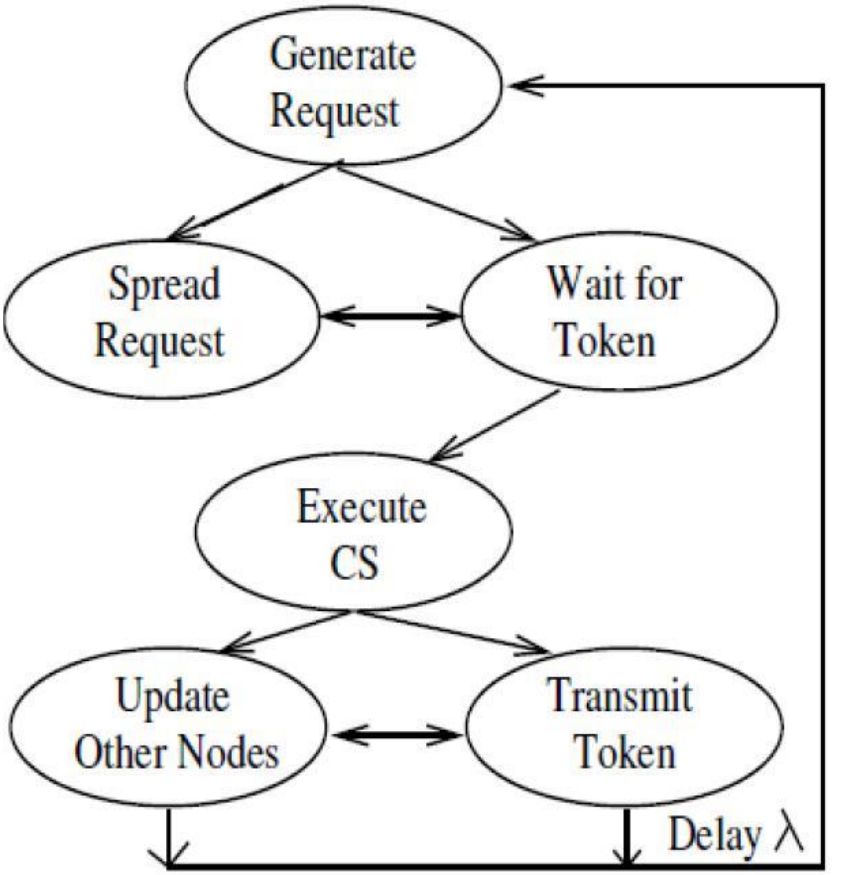

Figure 6: State diagram of MEOP Algorithm.

3.3 The GME Problem

Over the past few decades, Joung (2000) identified and solved the Group Mutual Exclusion problem as an interesting generalization of the Mutual exclusion problem. The problem Joung identified was how to avoid the delay of processes requesting to access the same shared resource. He therefore, proposed the solution to this problem by implementing Group Mutual Exclusion [8]. It ensures that processes are associated to a group, if and only if they request for the same resource. Processes in a group can concurrently execute the critical section.

In this section, we present a scenario of this problem in Opportunistic Network.

Considering an Opportunistic network with N number of mobile nodes moving randomly in the network competing for M number of resources where nodes communicate only through message passing.

To ensure concurrent execution of critical section, a similar approach is adopted from [[21], [9]].

In the GME problem, Figure 3.4 shows the state transition of each node in the proposed algorithm. A node continuously circulates the state transition diagram from a **non - critical section, trying section and critical section**. A shared resource is accessed only in the critical section. A node wishing to access a shared resource *Ri*, transits to trying state and waits to enter critical section state. It then exits from the critical section state back to the non-critical section state.

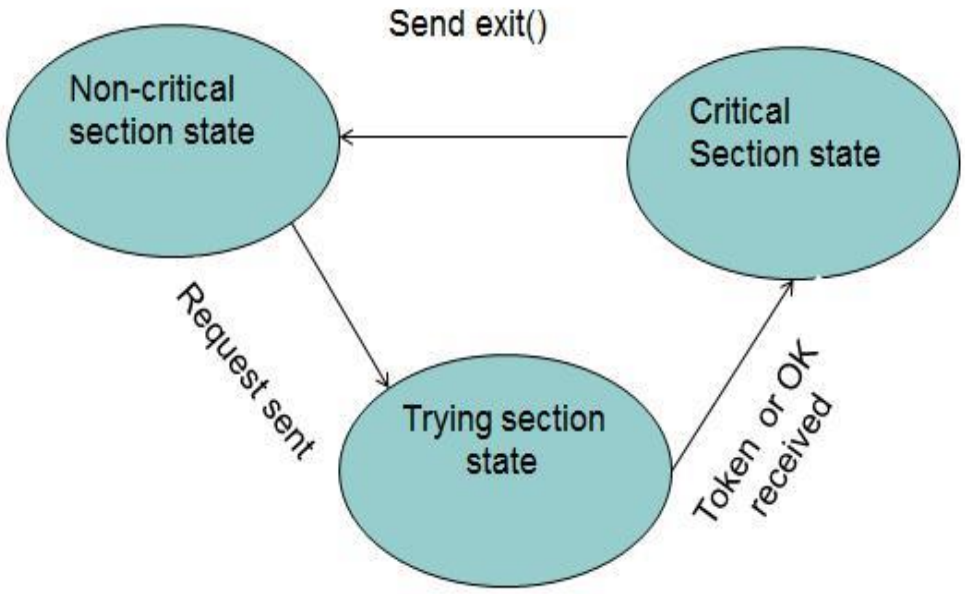

Figure 7:State diagram of GME Algorithm for OppNet.

The following properties must be satisfied in designing a GME algorithm [4];

- ***Mutual Exclusion:*** Two distinct nodes can execute the critical section simultaneously if and only if they are accessing same shared resource i.e. Ri = Rj.
- ***Concurrent Entering:*** Nodes requesting to access the same resource can do so concurrently if and only if no other node is accessing a different resource.
- ***Bonded Delay:*** A node in a trying section will eventually enter the critical section.

3.4 GME Algorithm for Opportunistic Network

3.4.1 Data Structure

- *id*: variable that stores ID of the node
- *Tid*: variable that stores timestamps of the node

- *Resource-id*: variable that stores type of resource a node is requesting for
- *State* : defines state of a node either Critical Section , Trying Or Non- Critical Section
    - *Token-Holder* : Node holding Token
    - *CurrentRes-id*: Type of resource currently in critical section
    - *Pending-Q*: Array of request(Id, Tid, Resource-id)
    - *Num*: keep records of number of nodes concurrently accessing Critical section

3.4.2 Messages

- *OK* (): message sent by Token-Holder to Allow concurrent access to critical section
- *Exit* (): Massage sent by non -Token holder exiting critical section.
- *Token*(): A message forward to node requesting for Token
- *Exit*(): Message sent to indicate node exit critical section

3.4.3 Initialization

- Token-Holder not Known.
- Pending-Q is empty
- Num = 0
- For all nodes in Network
- State.nodei = NCS

3.4.4 Pseodocode

Procedure Tokengenerate() {forward token id to all node}

1 Assume node 1 is Token-Holder
2 Token-id= 1
3 For every nodei in network
4 forwardToken-id() // flood the network with token-id

Procedure generateRequest() {Generate Request and Start Request Propagation}

1 if Token-id is known then
2 Request-id = (Tid, id, resource-id); // timestamp, node id and resource id
3 State = Trying
4 routeRequestTo(desination)

Procedure routeRequestTo(destination) Forward Request to Token-Holder

1 if destination == Token-id and state.token-holder==critical section then
2 if priority is high and resource-id== currentRes-id
3 forwardOK();
4 else Add request to list of pending-Q

5 else
6 Store local copy of the request
7 exchangeInfo ()
8 Use routing protocol to find next hop towards Token-Holder
9 Transfer request to next hop
10 end if

Procedure consumeToken () {forward OK to pending-Q with same resource as current resource, Execute CS and forward Token to successor}

1 if nodei receives Token
2 Token-Holder= nodei
3 TokenGenerate()
4 CurrentRes-id= nodei.Resource-id
5 State.nodei = Critical Section
6 Num= 1
7 if pending-Q not empty
8 for all nodej in pending-Q
9 if CurrentRes-id == nodej.Resource-id
10 Dequeue(nodej) // if in communication range
11 Forward OK(nodej)
12 State.nodej= Critical Section
13 Num = Num +1
14 FowardExit()
15 While Num = 0
16 If pending-Q not empty
17 successor := get next high priority node from request list
18 forwardToken(successor,pending-Q) // forward token to successor with list of pending-Q
19 Else exchangeInfo()

Procedure forwardExit()

1 if nodei.state = CS
2 forwardExit(nest hop)
3 State.nodei = Non Critical Section
4 Num = Num-1

Procedure forwardToken(destination, pending-Q) { Forward token and list of pending requests to destination}

1 if destination == id then
2 consumeToken()
3 else exchangeInfo()
4 Use routing protocol and transfer token to next hop towards destination
5 end if

Procedure exchangeInfo() {Update information with neighbors}

• Transmit ID of token holder to all neighbors

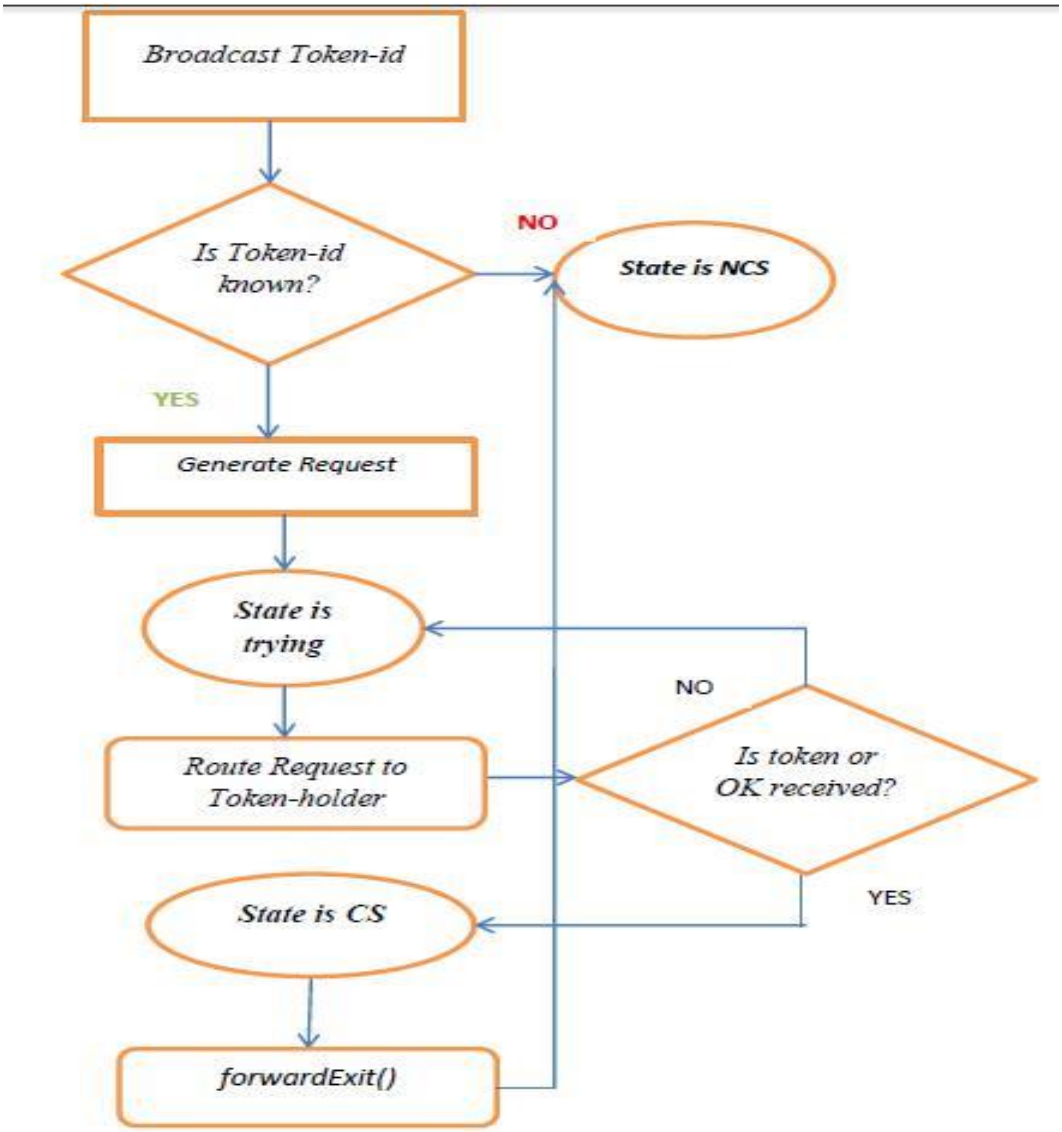

Figure 8: Flow chat of GME Algorithm for Opportunistic Network

3.5 Explanation of the GME algorithm for OppNet

The behavior of the GME algorithm in opportunistic networking is explained in the sections below.

3.5.1 Generating Request

Initially the state of all nodes in the network is in non-critical section and the Token-id is not known to all nodes in the network. The token-holder uses a flooding based routing protocol to inform all nodes of the token location. For example in Figure 9, *nodea* holding the token floods the network with token-holder id. Once the token-id is known, if any node wishes to access the critical section and it is not token-holder, it generates a request in the form of (id, Tid, Resource-id) indicating its identity, time request generated and the resource its requesting to access in critical section. The *node* then changes its state to *trying section*. Whilst in the trying section, it waits to receive either a token or an OK message to execute critical section. Another request can be generated only after exiting the critical section.

3.5.2 Forwarding Request

MEOP [20] implements a logical DAG where, nodes point to the token-holder but, the DAG needs no updating when the topology changes, thus independent of the underlying routing protocol.

In forwarding the request message, an OK message or Token, messages are forwarded over several nodes before it is finally delivered to the destination. A social context routing protocol such as bubble Rap or Prophet routing protocol could be implemented in forwarding such messages.

When a node receives a request and it is not the token-holder, it stores the request in its local list of pending request when it opportunistically contacts the token-holder it forwards the list to the token-holder. If the token-holder receives request from any *node*, a *node* can execute critical section concurrently with token-holder if only all the following conditions are satisfied:

- The Token-holder is in critical section.
- The currentRes-id is same as the Resource-id of nodej.
- *Node* has high priority than all nodes in pending queue.

If any of the above conditions are not satisfied the token-holder adds the request to its re-quest list. In Figure 9, the **nodes: g,b,d,f** in communication range with the token-holder route request to **nodea** and **nodec** identifies **nodef** is closer to token-holder route its request to **nodef**. In Figure 11 the Token was forwarded to *nodef*. **Nodef** is the new token-holder but **nodeg**

not updated with the new token-holder route request to *nodea*. Upon exchange of information **nodeg** will update its info about token-id and identify the best route to send request to token-holder. The actual routing will be performed by any defined routing protocol.

3.5.3 Forwarding Token

In Opportunistic network the path taken in forwarding the request might not be the reverse path to route Token or OK messages due to mobility of nodes. If a node receives a Token and List of pending request it floods the network using epidemic routing protocol with information of token-holder's identity. If the token-holder receives requests then, it routes OK messages to neighboring nodes requesting for the same resource, in order to execute critical section concurrently. After all nodes exit critical section the token-holder routes the token to the next in queue. If the queue is empty it keeps the token till it receives a request then it forwards the token to requested node.

An example is shown in Figure 10, the token-holder identifies node: g,d,c requesting for the same resource as that of the token-holder's. Since node d is not within its communication range, it will only invite **nodeg** and **nodec** by sending OK messages and then remove them from the request queue.

When all nodes terminate their tasks in the critical section, ***nodef*** is identified as next in queue with high priority based on timestamp. The token with the list of pending requests is routed to ***nodef*** . In Figure 11, **nodef** will again broadcast token identity and then invite others to execute critical section simultaneously.

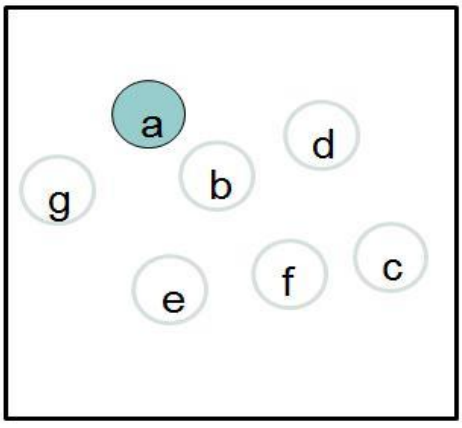

Figure 10: Node A Is Token – Holder

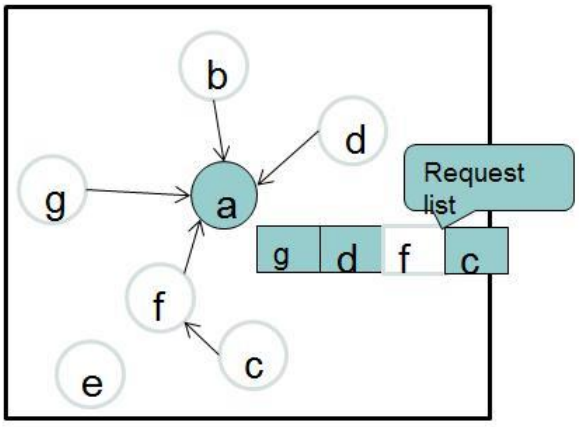

Figure 9: Nodes forwards request

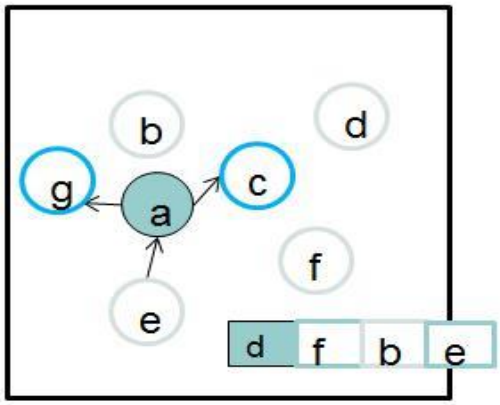

Figure 11: Token-Holder Invites Neighboring Nodes

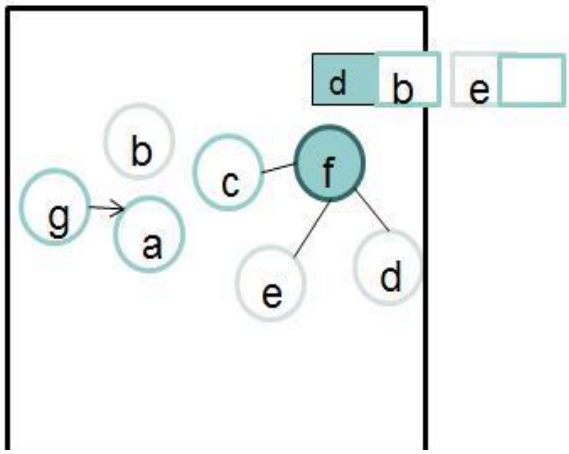

Figure 12: Successor receives token

CHAPTER 4

4.0 Proof of Algorithm Correctness

Proof of the proposed algorithm is presented in this chapter to ensure it 'satisfies' the properties of the Group Mutual Exclusion problem stated in chapter 3; Concurrent entering, Bounded delay (Starvation free), and Mutual exclusion (safety property).

4.1 The Mutual Exclusion Property

In the Group Mutual Exclusion Algorithm, mutual exclusion also known as safety property must be satisfied. In this property, two nodes can concurrently execute critical section, if and only if they request for the same type of resource. In other words, no two nodes can execute different resource types at the same time. Before proving this property, we present the following proposition which is easily verified from the algorithm in addition to the Algorithm notation presented in Table 2;

4.1.1 Proposition

There is exactly one Token-Holder in the network at a time.

Table 2: Notation of Algorithm.

When a node becomes Token-Holder

```
1   for all nodei in pending-Q
2   if CurrentRes-id == Resource-id
3   Dequeue(nodei)
4   Forward OK(nodei)
5   State.nodei= Critical Section
6   Num = Num +1;
7   forwardExit()
```

Definition of variables

Resource-id: variable that stores type of resource a node is requesting for.

State : defines state of a node either Critical Section , Trying Or Non- Critical Section.

CurrentRes-id: Type of resource the token - holder is currently executing in crit-ical section.

Pending-Q: Array of request(Id, Tid, Resorce-id) pending at Token-holders re-quest queue.

Num: keep records of number of nodes concurrently accessing Critical section.

In line 5 of the algorithm *state.node = Critical section* implies a node is executing crit-ical section. A node that receives a token becomes token-holder. A token-holder invites other nodes by forwarding an OK to only those nodes requesting for same resource type as its resource.

4.1.2 Theorem

*If state.nodei = critical section and State.nodej = critical section simultaneously implies m**odei's** Resourece-id = Resource-id of **nodej**.*

4.1.3 Proof by contradiction:

Assume at some instance *state.nodei = critical section and state.nodej = critical section simultaneously, and **nodei**; Resource-id ≠ Resource-id of **nodej**.*

In the proposed algorithm, a node can execute critical section if it is the token-holder or if it receives OK. Therefore, the assumption implies, either both nodes are token-holders or, one is a token-holder and forwards an OK to the other node.

In proposition 4.1.1 it indicates that both nodes can never be token-holders at the same time so the assumption is false. Also, from the notation if *node i* is token-holder, then *node j r*eceives OK from *node i* if and only if, *resource-id == current resource* as in line 2 of the algorithm.

Therefore, on the contrary *node i* and *node j* cannot be in critical section simultaneously. Hence, mutual exclusion is guaranteed in the proposed Algorithm.

4.2 The Concurrency Property

Concurrency is the key problem of the GME problem. Hence the proposed algorithm must satisfy the concurrency property. The algorithm must ensure that nodes requesting the same resource type can execute the resource concurrently

4.2.1 Theorem

The algorithm satisfies the concurrent entering property

4.2.2 Direct the proof:

With reference to the algorithm notation presented in Table 2, When a node holds token it sends OK to neighboring nodes, if and only if, the requesting resource-id is the same as that of the token-holder's resource-id i.e *resource-id == currentres-id* before it enters critical section. A node receiving OK enters critical section with the token-holder.

In summary, nodes can enter critical section if only their request is the same as the current resource in critical section. Therefore, the concurrent entering property is satisfied.

4.3 The Bounded Delay Property

In the Group Mutual Exclusion problem, the bounded delay property ensures that a node that is waiting to execute critical section should eventually access the critical section. In proving this

property in the GME algorithm presented, we use some notations from the algorithm in addition to proposition 4.1.1.

The algorithm notation in Table 3 shows that each token request is propagated till it reaches the token-holder; hence, the token - holder will eventually receive each request.

Table 3: Notation When a Node Receives Request

1	if nodei ≠ token-holder
2	Store local copy of the request
3	exchangeInfo ()
4	Use routing protocol to find next hop towards Token-holder
5	Transfer request to next hop

In Table 2 line 7 of algorithm, if a node forward exit, the node terminates its task in critical section to reduce the number of nodes in critical section. In Table 4, if every node terminates critical section including token-holder then number of nodes will sum to zero in critical section.

Table 4: Notation when Token-Holder is in non-critical section state

When Token-Holder is in non-critical section state	
1	While Num = 0
2	If pending-Q not empty
3	successor := get next high priority node from request list
4	forwardToken(successor,pending-Q)
5	Else exchangeInfo()
Definition of variables	
	Successor: Node with earliest generated request in pending-Q.
	forwardToken(successor,pending-Q) : Token is sent to successor with list of pending-Q.

4.3.1 Theorem

If every node in critical section state transits to non-critical section state, then a requesting node will eventually become a token-holder.

4.3.2 Proof:

In table 4, if number of nodes in critical section is **0**, the token-holder de-queues the successor

from the request queue and forwards a token with a list of pending requests to the successor. Therefore, based on the timestamp, every node will eventually become a successor to become a token-holder. Also, since a request is guaranteed to reach a token-holder every requesting node will eventually become a token-holder

CHAPTER 5
5.0 Conclusion and Future Work

5.1 Conclusion

In a typical distributed network such as Opportunistic network, Mutual Exclusion is a fundamental problem. To avoid nodes waiting to access the same type of resource; the Group Mutual Exclusion problem was conceived by Joung in [8]. This problem improves upon concurrency in the traditional mutual exclusion problem.

Nodes in Opportunistic network are mobile and resource constrained. Therefore, there is a need to ensure exclusive access to shared resources. In this thesis, we presented a variant of the Mutual Exclusion problem known as the Group Mutual Exclusion problem for Opportunistic network. A review of proposed GME algorithm for MANETs was presented to evaluate their applicability to Opportunistic network. A token based GME algorithm adapted from MEOP in [20] was also proposed.

The proposed algorithm assumed an integrated routing protocol as in [23] for Opportunistic network for message passing. The algorithm satisfied the bounded delay, concurrent entering and mutual exclusion property.

5.2 Future Work

Simulations to be extended for future work.

Bibliography

[1] Atreya, R. and Mittal, N. (2005). A dynamic group mutual exclusion algorithm using surrogate-quorums. In *Distributed Computing Systems, 2005. ICDCS 2005. Proceedings. 25th IEEE International Conference on*, pages 251–260. IEEE.

[2] Cenci, K. M. and Ardenghi, J. R. (2011). Group mutual exclusion-role processes. In *XVII Congreso Argentino de Ciencias de la Computación*.

[3] Conti, M., Kumar, M., et al. (2010). Opportunities in opportunistic computing. *Computer*, 43(1):42–50.

[4] Hadzilacos, V. (2001). A note on group mutual exclusion. In *Proceedings of the twentieth annual ACM symposium on Principles of distributed computing*, pages 100–106. ACM.

[5] Hoang Anh Nguyen, S. G. (2008). Routing in opportunistic networks. *Reliability Engineering and System Safety*, 93:1208–1217.

[6] Huang, C.-M., Lan, K.-c., and Tsai, C.-Z. (2008). A survey of opportunistic networks. In *Advanced Information Networking and Applications-Workshops, 2008. AINAW 2008. 22nd International Conference on*, pages 1672–1677. IEEE.

[7] Jiang, J.-R. (2002). A group mutual exclusion algorithm for ad hoc mobile networks. In *JCIS*, pages 266–270. Citeseer.

[8] Joung, Y.-J. (2000). Asynchronous group mutual exclusion. *Distributed computing*, 13(4):189–206.

[9] Keane, P. and Moir, M. (1999). A simple local-spin group mutual exclusion algorithm. In *Proceedings of the eighteenth annual ACM symposium on Principles of distributed computing*, pages 23–32. ACM.

[10] Lilien, L., Kamal, Z. H., Bhuse, V., and Gupta, A. (2006). Opportunistic networks: the concept and research challenges in privacy and security. *Proc. of the WSPWN*, pages 134–147.

[11] Mamun, Q. E. K. and Nakazato, H. (2006). A new token based protocol for group mutual exclusion in distributed systems. In *Parallel and Distributed Computing, 2006. ISPDC'06. The Fifth International Symposium on*, pages 34–41. IEEE.

[12] Manabe, Y. and Park, J. (2004). A quorum-based extended group mutual exclusion algorithm without unnecessary blocking. In *Parallel and Distributed Systems, 2004. ICPADS 2004.*

Proceedings. Tenth International Conference on, pages 341–348. IEEE.

[13] Manas, k. y. and vijayakranthi, c. (2014). A study of opportunistic networks for efficient ubiquitous computing. *International Journal of Advanced Research in Computer and Communication Engineering*, 3(1):20–26.

[14] Mittal, N. and Mohan, P. K. (2005). An efficient distributed group mutual exclusion algorithm for non-uniform group access. In *IASTED PDCS*, pages 367–372.

[15] Navneet, K. and Gauri, M. (2016). Opportunistic networking: A review. *Computer Engineer-ing (IOSR-JCE)*, 18(2):20–26.

[16] Pelusi, L., Passarella, A., and Conti, M. (2006). Opportunistic networking: data forwarding in disconnected mobile ad hoc networks. *Communications Magazine, IEEE*, 44(11):134–141.

[17] Priya, M., Krishna, C. R., and Saini, P. (2014). Arbitration based distributed group mutual exclusion algorithm for mobile ad hoc networks. In *Parallel, Distributed and Grid Computing (PDGC), 2014 International Conference on*, pages 147–152. IEEE.

[18] Swaroop, A. and Singh, A. K. (2009). A hierarchical approach to handle group mutual exclusion problem in distributed systems. In *Distributed Computing and Networking*, pages 462–467. Springer.

[19] Talele, P., Penurkar, M., Bhutada, S., and Talele, H. (2013). A token based distributed group mutual exclusion algorithm with quorums for manet. *International Journal of Emerging Sci-ence and Engineering*, 1(5):43–48.

[20] Tamhane, S. A. and Kumar, M. (2010). Token based algorithm for supporting mutual exclusion in opportunistic networks. In *Proceedings of the Second International Workshop on Mobile Opportunistic Networking*, pages 126–134. ACM.

[21] Thiare, O. (2012). A token-based group mutual exclusion algorithm for manets. In *Computer Applications for Communication, Networking, and Digital Contents*, pages 243–250. Springer.

[22] Toyomura, M., Kamei, S., and Kakugawa, H. (2003). A quorum-based distributed algorithm for group mutual exclusion. In *Parallel and Distributed Computing, Applications and Technologies, 2003. PDCAT'2003. Proceedings of the Fourth International Conference on*, pages 742–746. IEEE.

[23] Verma, A., Srivastava, D., et al. (2012). Integrated routing protocol for opportunistic networks. *arXiv preprint arXiv:1204.1658*.