

Case
2006
Thesis

**OUT OF BAND MESSAGE PASSING IN
WIRELESS TOKEN RING NETWORKS**

By

SIVAKUMAR CHINNATHAMBI JANAKIRAMAN

Bachelor of Engineering

University Of Madras

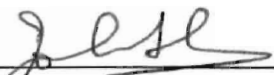
Madras, India

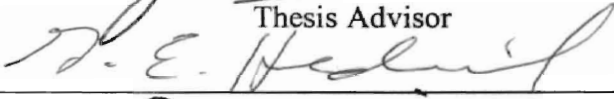
1998

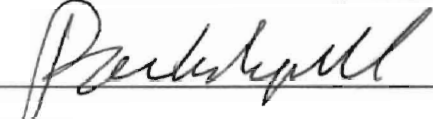
Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
in the partial fulfillment of
the requirements for
the Degree of
MASTER OF SCIENCE
December, 2003


OUT OF BAND MESSAGE PASSING IN WIRELESS TOKEN RING NETWORKS

Thesis Approved:



Thesis Advisor






Dean of the Graduate College

PREFACE

The Internet world is rapidly progressing from a wired era to a wireless era. New protocols are designed for this transition. Wireless Token Ring protocol (WTRP) is one such protocol that enables user to remain connected irrespective of the physical location in Ad-hoc networks. Key performance issues in WTRP is channel bandwidth is limited and the channel is shared among many stations. Under normal operating conditions, MAC protocol must provide the following guarantees to achieve Quality of Service (QOS). This is achieved by Minimum throughput for each station and Medium access time for each station is bounded. To improve the performance of WTRP an exceptional three different Out-Of-Band-Message passing technique are proposed. This enables QOS, uses the bandwidth effectively and minimizes the token rotation time. This further reduces the message delay in each node and increases the number of message passed.

ACKNOWLEDGEMENTS

I wish to express my sincere appreciation to Dr. Johnson Thomas for his guidance and assistance at Oklahoma State University. I would also like to thank my committee members, Dr. G. E. Hedrick and Dr. Nohpill Park, for their helpful contributions and advice.

A heart-felt thanks goes to my parents my brother and sister for their unending encouragement and emotional support throughout the years.

Finally I would like to thank all my friends who stood beside me with their unflinching and indispensable support.

ACKNOWLEDGEMENTS

I wish to express my sincere appreciation to Dr. Johnson Thomas for his guidance and assistance at Oklahoma State University. I would also like to thank my committee members, Dr. G. E. Hedrick and Dr. Nohpill Park, for their helpful contributions and advice.

A heart-felt thanks goes to my parents my brother and sister for their unending encouragement and emotional support throughout the years.

Finally I would like to thank all my friends who stood beside me with their unfailing and indispensable support.

TABLE OF CONTENTS

Chapter		Page
1	Introduction	
1.1	Networking World.....	1
1.2	Ad-hoc Network.....	2
1.3	Network Topology.....	5
1.4	Token Ring Network.....	7
1.4.1	Ring benefits	8
1.4.2	Token Ring Mechanism	8
1.4.3	Early Token Release Mechanism	9
1.4.4	Topology Media	10
1.4.5	Token Frame Format	10
1.4.6	Token Frame Fields	11
1.4.7	Data/Command Frame Fields	12
1.4.8	Ring Management	13
1.5	Token Bus Network.....	16
2	Overall Architecture and Deign of WTRP	
2.1	Medium Access Control.....	17
2.2	Channel Allocator..	18
2.3	Mobility Manager.....	19
2.4	Admission Control	20
2.5	Policer.....	20
2.6	Management Information Base.....	21
3	WTRP: Protocol Description	
3.1	Definitions.....	22
3.2	Observations	22
3.3	Frame Format	23
3.4	Connectivity Manager	24
3.5	Joining the Ring	26
3.6	Exiting the Ring	28
3.7	Multiple Ring	29

Chapter 4 Proposed Extension to WTRP

4.1	Limitations of WTRP	31
4.2	Proposed Extension	32
4.3	Out-Of-Band In-route Transmission	33
4.4	Out-Of-Band Off-route Transmission	34
4.5	Out-Of-Band Not-In-ring Transmission.....	35
4.6	State Diagram for WTRP Extension.....	37
4.6.1	Beginning State.....	38
4.6.2	Floating State.....	38
4.6.3	Offline State.....	39
4.6.4	Joining State.....	39
4.6.5	Soliciting State.....	40

Chapter 5 Simulation and Results

5.1	Performance Measures.....	42
5.2	Steps involved in simulation.....	42
5.3	Charts.....	43
5.3.1	Average Token Holding Time	44
5.3.1	Average Message Delay	45
5.3.1	Average Idle Time	46
5.4	Results	47

Chapter 6 Conclusions

6.1	Conclusions.....	48
6.2	Future Work.....	48
	References.....	49
	Appendix.....	51

List of Figures

Figure		Page
1.1	Ad-Hoc or Peer-to Peer Networking	3
1.2	Hardware Access Point	4
1.3	Software Access Point	4
1.4:	Network Topologies	7
1.5	Token Ring Network	9
1.6	IEEE 802.5 and Token Ring Specify Tokens and Data/Command Frames.....	11
1.7	Token Bus Network.....	16
2.1	Overall System Architecture	17
3.1	Frame Format.....	23
3.2	Connectivity Table	24
3.3	Joining.....	26
3.4	Exiting	28
3.5	Multiple Rings	29
4.1	Out-Of-Band In-route Transmission (OBIRT).....	33
4.2	Out-Of-Band Off-route Transmission (OBORT).....	35
4.3	Out-Of-Band Not-In-Ring Transmissions (OBNIRT).....	36
4.4	State diagram	37
5.1	Average Token Holding time.....	44
5.2	Average Message Delay.....	45
5.3	Average Idle time	46

List of Symbols

WTRP	-	Wireless Token Ring Protocol.
OBIRT	-	Out-Of-Band In-route Transmission
OBNIRT	-	Out-Of-Band Not-In-ring Transmission
OBORT	-	Out-Of-Band Off-route Transmission
TRT	-	Token Rotation Time
NoN	-	Number of Nodes
MAC	-	Medium Access Control
SA	-	Source Address
DA	-	Destination Address
RA	-	Ring Address
NIR	-	Not In Ring
TCP	-	Transmission Control Protocol.
QoS	-	Quality of Service
MIB	-	Management Information Base
THT	-	Token Holding Time
MTRT	-	Maximum Token Rotation Time

Chapter 1

Introduction

1.1 Networking World

Millions of people around the world use the Internet every day, to communicate with others, follow the stock market, keep up with the news, check the weather, make travel plans, conduct business, shop, entertain yourself and learn. Staying connected with the Internet has become so important to keep oneself updated of the various activities going on around the world. Online business take place within a time frame of seconds, a person who is poor this minute turns a billionaire the next. In order to keep up with this pace there is a need to stay connected. One may lose a lot of information if he happens to travel from one place to another. This scenario brings in the necessity of wireless networks.

Wireless networks allow the users the freedom to travel from one location to another without interruption to their computing resources. However wireless networks require the existence of wired base station in order for the wireless user to send/receive messages. The term wireless networking refers to technology that enables two or more computers to communicate using standard network protocols, but without network

Cabling. Strictly speaking, any technology that does this could be called wireless networking. The current buzzword however generally refers to wireless LANs.

This technology, fuelled by the emergence of cross-vendor industry standards such as IEEE 802.11, has produced a number of affordable wireless solutions that are growing in popularity with business and schools as well as sophisticated applications where network wiring is impossible, such as in warehousing or point-of-sale handheld equipment.

There are two kinds of wireless networks:

1. Ad-hoc Network
2. Wireless Network

1.2 Ad-hoc Network

An ad-hoc or peer-to-peer wireless network consists of a number of computers each equipped with a wireless networking interface card. Each computer can communicate directly with all of the other wireless enabled computers. They can share files and printers this way, but may not be able to access wired LAN resources, unless one of the computers acts as a bridge to the wired LAN using special software. (This is called "bridging"). Each computer with a wireless interface can communicate directly with all of the others.

Mobile hosts and wireless networking hardware are becoming widely available, and extensive work has been done recently in integrating these elements into traditional networks such as the Internet. Oftentimes, however, mobile users will want to communicate in situations in which no fixed wired infrastructure such as this is available, either because it may not be economically practical or physically possible to provide the necessary infrastructure or because the expediency of the situation does not permit its installation.

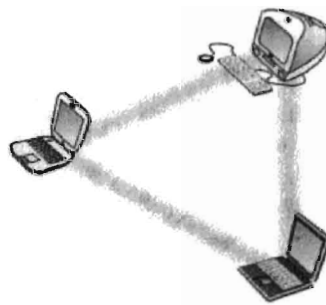


Figure 1.1: Ad-Hoc or Peer-to Peer Networking.

For example, a class of students may need to interact during a lecture, friends or business associates may run into each other in an airport terminal and wish to share files, or a group of emergency rescue workers may need to be quickly deployed after an earthquake or flood. In such situations, a collection of mobile hosts with wireless network interfaces may form a temporary network without the aid of any established infrastructure or centralized administration. This type of wireless network is known as an *ad hoc network*.

A wireless network can also use an access point, or base station. In this type of network the access point acts like a hub, providing connectivity for the wireless computers. It can

connect (or "bridge") the wireless LAN to a wired LAN, allowing wireless computer access to LAN resources, such as file servers or existing Internet Connectivity.

Wireless connected computers using a Hardware Access Point.

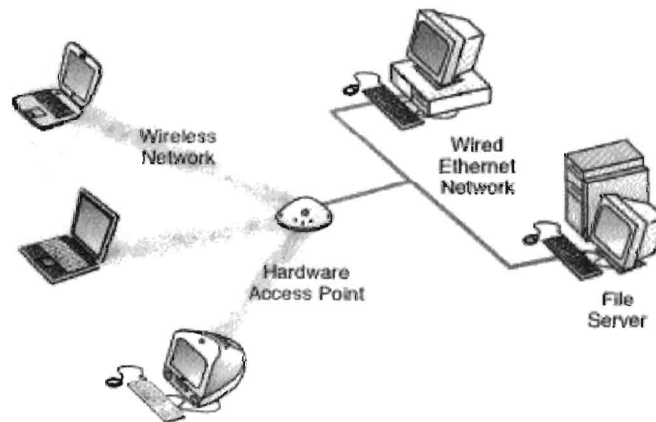


Figure 1.2: Hardware Access Point.

Wireless connected computers using a Software Access Point.

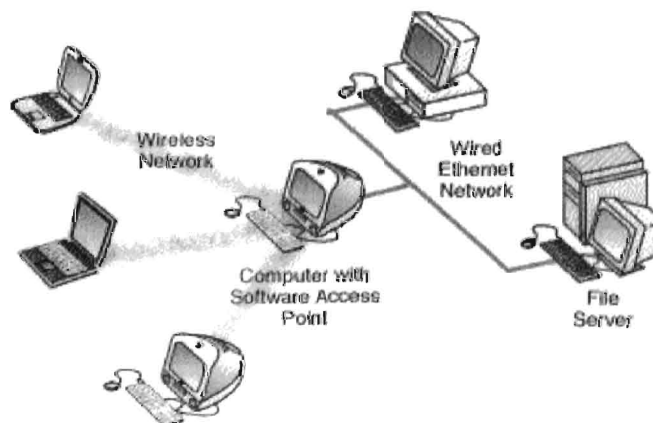


Figure 1.3: Software Access Point.

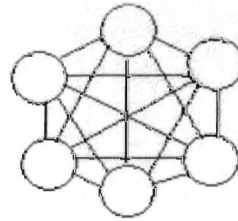
Mobile hosts and wireless networking hardware are becoming widely available, and extensive work has been done recently in integrating these elements into traditional networks such as the Internet. Oftentimes, however, mobile users will want to communicate in situations in which no fixed wired infrastructure such as this is available, either because it may not be economically practical or physically possible to provide the necessary infrastructure or because the expediency of the situation does not permit its installation. For example, a class of students may need to interact during a lecture, friends or business associates may run into each other in an airport terminal and wish to share files, or a group of emergency rescue workers may need to be quickly deployed after an earthquake or flood. In such situations, a collection of mobile hosts with wireless network interfaces may form a temporary network without the aid of any established infrastructure or centralized administration. This type of wireless network is known as an *ad hoc network*.

1.3 Network Topologies

Topology refers to the shape of a network, or the network's layout. How different nodes in a network are connected to each other and how they communicate is determined by the network's topology. Topologies are either physical or Logical. Below are diagrams of the five most common network topologies.

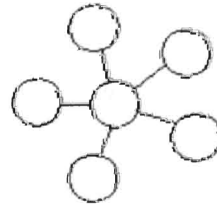
Mesh Topology

Devices are connected with many redundant interconnections between network nodes. In a true mesh topology every node has a connection to every other node in the network.



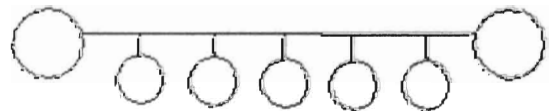
Star Topology

All devices are connected to a central hub. Nodes communicate across the network by passing data through the hub.



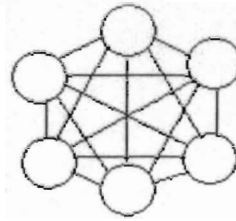
Bus Topology

All devices are connected to a central cable, called the bus or backbone.



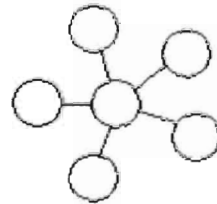
Mesh Topology

Devices are connected with many redundant interconnections between network nodes. In a true mesh topology every node has a connection to every other node in the network.



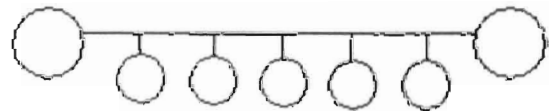
Star Topology

All devices are connected to a central hub. Nodes communicate across the network by passing data through the hub.



Bus Topology

All devices are connected to a central cable, called the bus or backbone.



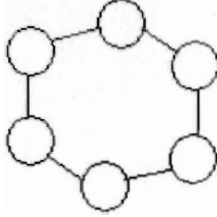
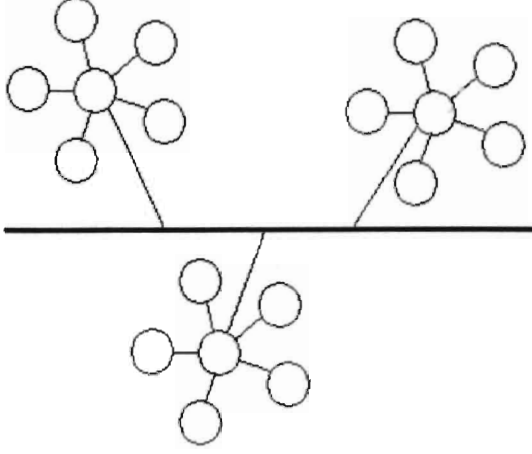
<p>Ring Topology</p> <p>All devices are connected to one another in the shape of a closed loop, so that each device is connected directly to two other devices, one on either side of it.</p>	
<p>Tree Topology</p> <p>A hybrid topology. Groups of star-configured networks are connected to a linear bus backbone.</p>	

Figure 1.4: Network Topologies.

1.4 Token Ring Network

Token Ring is a Local Area Network (LAN) protocol. The Token Ring protocol was first developed by IBM. Token Ring is standardized in IEEE 802.5 that was published in 1985. The protocol deals with the problem of collision, which is defined as a state, where two stations transmit at the same time. In order to avoid the situation of collision there was a need to control the access to the network. This kind of control is

possible by the use of a control (permission) called token. The token is passed from one station to another according to a set of rules. The ring consists of ring stations and transmission medium. Data travels sequentially from station to station. Only the station in possession of the token is allowed to transmit data. Each station repeats the data, checks for errors, and copies the data if appropriate. When the data is returned to the sending station, it removes it from the ring. The token Ring protocol supports priorities in transmission. It is implemented setting the priority bits in the Token Ring Frame.

Token Ring is a first and second layer protocol in the OSI (Open Systems Interconnection) seven layer model. The First release of Token Ring version was capable of 4Mbs data transmission rate, the transmission rate was improved later to 16Mbs.

1.4.1 Ring benefits:

- High reliability, the Ring can continue normal operation despite any single fault.
- Bypassing inactive stations.
- Effective use, 95% in Token Ring only whilst 30-40% in Ethernet.
- Excellent traffic handling (17.8 kb in TR, only 15kb in Ethernet.).
- Large maximum frame length.
- High bandwidth efficiency. 70% in Token Ring, 30% in Ethernet.
- Many media choices: UTP STP coax fiber.
- Supports transmission priority.

1.4.2 Token Ring Mechanism

Whenever a station wishes to send a frame, it first waits for the token. As soon as it receives the token, it initiates transmission of the frame, which includes the destination station address at its head. The frame is repeated (received and retransmitted) by each station on the network until it circulates back to the source station, where it is removed. In addition to repeating the frame, the destination station retains a copy of the frame and indicates that by setting the response bits (Copy Bit + Address Recognition Bit) at the tail of the frame. A station releases the token in one of the two ways depending on the ring rate. With slower rings (4Mbps), the token is released only after the response bits have been received. With higher speed rings (16Mbps), it is released after transmitting the last bit of the frame. This is known as early (token) release (ETR).

A typical token link mechanism is illustrated below.

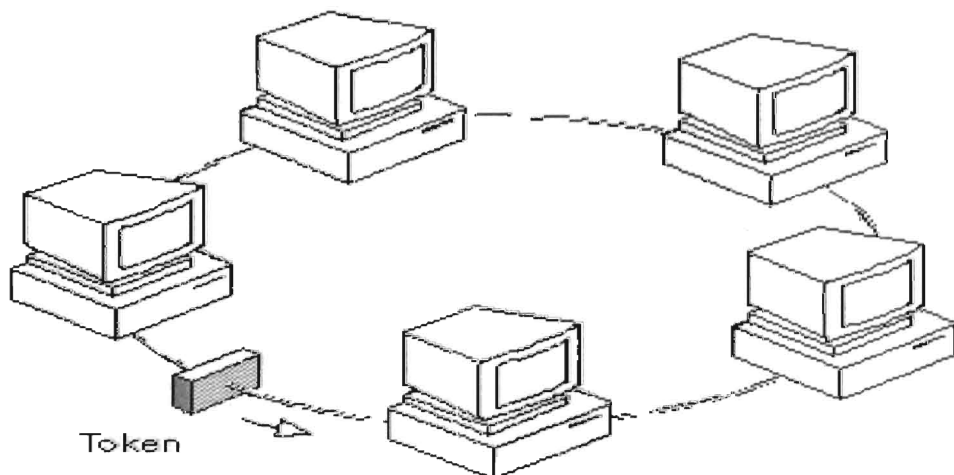


Figure 1.5: Token Ring Network.

1.4.3 Early Token Release mechanism (ETR):

A station that wants to transmit waits for a free token. The station transmits a frame and then releases the token before getting acknowledgment. The next station that wants to transmit waits for a free token and transmits a frame and then releases its token into the ring and so on.

The ETR mechanism enables multiple frames on the ring, and therefore the ring is more effective. When working in a large ring it improves performance, enabling a mixture of stations with ETR and stations without ETR.

1.4.4 Topology Media

Token ring is a logical ring topology, but can physically implement as:

- Ring
- Bus
- Star

Token Ring uses two counter-rotating rings like FDDI. One ring for main path and another for backup, this way it can bypass faulty parts. It enables continued operation with any single fault.

Token Ring can be operated on the following media's:

- Unshielded Twisted Pair (UTP).

- Shielded Twisted Pair (STP): Allowing a Max. of 260 stations at 16Mps rings.
- Coaxial cable (Thin\Thick\Broadband).
- Fiber Optics.

1.4.5 Token Frame Format

Token Ring and IEEE 802.5 support two basic frame types: tokens and data/command frames. Tokens are 3 bytes in length and consist of a start delimiter, an access control byte, and an end delimiter. Data/command frames vary in size, depending on the size of the Information field. Data frames carry information for upper-layer protocols, while command frames contain control information and have no data for upper-layer protocols. Both formats are shown in Figure 1.6.

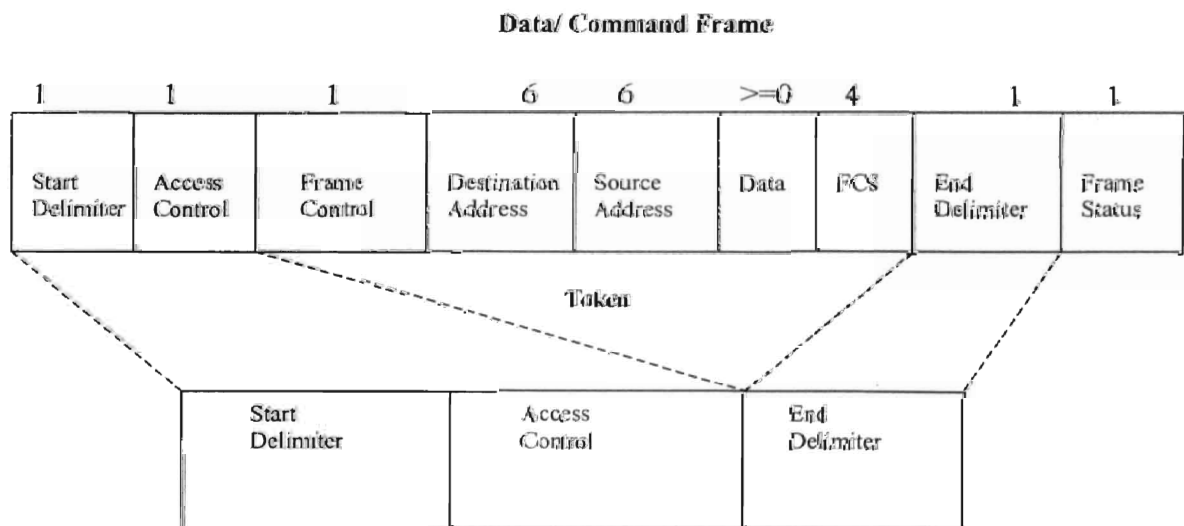


Figure 1.6: IEEE 802.5 and Token Ring Specify Tokens and Data/Command Frames

1.4.6 Token Frame Fields

The three token frame fields illustrated in Figure 1.6 are summarized in the descriptions that follow:

- **Start delimiter**—Alerts each station of the arrival of a token (or data/command frame). This field includes signals that distinguish the byte from the rest of the frame by violating the encoding scheme used elsewhere in the frame.
- **Access-control byte**—Contains the Priority field (the most significant 3 bits) and the Reservation field (the least significant 3 bits), as well as a token bit (used to differentiate a token from a data/command frame) and a monitor bit (used by the active monitor to determine whether a frame is circling the ring endlessly).
- **End delimiter**—Signals the end of the token or data/command frame. This field also contains bits to indicate a damaged frame and identify the frame that is the last in a logical sequence.

1.4.7 Data/Command Frame Fields

Data/command frames have the same three fields as Token Frames, plus several others. The Data/command frame fields illustrated in Figure 1.6 are described in the following summaries:

- **Start delimiter**—Alerts each station of the arrival of a token (or data/command frame). This field includes signals that distinguish the byte from the rest of the frame by violating the encoding scheme used elsewhere in the frame.

- **Access-control byte**—Contains the Priority field (the most significant 3 bits) and the Reservation field (the least significant 3 bits), as well as a token bit (used to differentiate a token from a data/command frame) and a monitor bit (used by the active monitor to determine whether a frame is circling the ring endlessly).
- **Frame-control bytes**—Indicates whether the frame contains data or control information. In control frames, this byte specifies the type of control information.
- **Destination and source addresses**—Consists of two 6-byte address fields that identify the destination and source station addresses.
- **Data**—indicates that the length of field is limited by the ring token holding time, which defines the maximum time a station, can hold the token.
- **Frame-check sequence (FCS)**—Is filed by the source station with a calculated value dependent on the frame contents. The destination station recalculates the value to determine whether the frame was damaged in transit. If so, the frame is discarded.
- **End Delimiter**—Signals the end of the token or data/command frame. The end delimiter also contains bits to indicate a damaged frame and identify the frame that is the last in a logical sequence.
- **Frame Status**—Is a 1-byte field terminating a command/data frame. The Frame Status field includes the address-recognized indicator and frame-copied indicator.

1.4.8 Ring Management

The mechanism of the network operation is consider being the mechanism in the steady state, but before this can take place the ring must be set up. Also if a new station wishes

to join operational rings it must first go through an initialization procedure to ensure that it does not interfere with the correct functioning of the current active ring. Also, during normal operation it is necessary for each active station on the ring to monitor its correct operation and if something is not working well it must take some action to try re-establishing a correctly functioning ring. Those functions and others, which meant to preserve the correct ring operation, are called ring management.

There are two types of stations in the ring Active Monitor (AM) station, and Standby Monitor (SBM) stations. There is only one Active Monitor station per ring. The Active Monitor is the ring manager. All other stations on the ring are Standby Monitor stations. Any station on the ring can be Active Monitor. The Active Monitor is chosen during a process called "Claim Token Process" after the Active Monitor is chosen all other stations become "Standby Monitors" (SBM)

- **Active Monitor Duties:** The Active Monitor Maintains the Master Clock it ensures proper ring delay. It initiates "Neighbor Notification" every 7 seconds it monitors Token and Frame transmission, Detects lost tokens and frames by setting the monitor bit, purging the ring.
- **Standby Monitor Duties:** To detect Active Monitor failures and to start Monitor Contention process to participate in the Neighbor Notification process

A list of the MAC frame types involved with the ring management function is:

- **Ring Poll:** Active Monitor sends Ring Poll every 7 sec. this process is used to learn the ring configuration. The Ring Poll routine is: Active Monitor sends an AMP (Active Monitor Present) frame. Each Downstream station sends a SMP (Standby Monitor Present) frame. Each Downstream Node learns its Next Active Upstream Neighbor (NAUN).
- **Ring Purge:** Takes place when token is lost. The purge frame is sent before the Active monitor initiates a new token. The Active Monitor broadcasts Ring Purge frame to all stations if 10ms elapsed. The Purge frame resets the stations to normal Repeat mode and cancels or restarts appropriate timers
- **Claim Token Process:** This is how the new active monitor is elected. It is initiated when the Active\Standby Monitor detects loss of signal, or a new station attaches and finds no Active Monitor.
- **Ring Insertion:** When a new station wants to enter the ring it performs a Ring insertion routine. First it makes a lobe media check - checks the lobe connections. In the second step it attaches the ring and searches for an Active Monitor if there is no Active Monitor for 18 sec, it initiates claim token process. In the third step it transmits a duplicate address test (DAT) frame; each active station checks the content of the frame to see that the new station address is different from its own, if it is not, the station sets a flag in the frame to indicate the error. After the frame has circulated back to the source station the later checks the error flag and acts

according to it. If there is no error the new station continues the init procedure by sending standby monitor present (SMP) frame.

- **Hardware error:** These are permanent faults, usually concerns hardware (equipment) that stops the ring's normal operation LAN: Local Area Network used to interconnect distributed computers (stations) located within a single building or a group of buildings.

1.5 Token Bus Network

A type of local area network that has a bus topology and uses a token-passing mechanism to regulate traffic on the bus. A token bus network is very similar to a token ring network, the main difference being that the endpoints of the bus do not meet to form a physical ring. Token bus network is defined by the IEEE 802.4 standard.

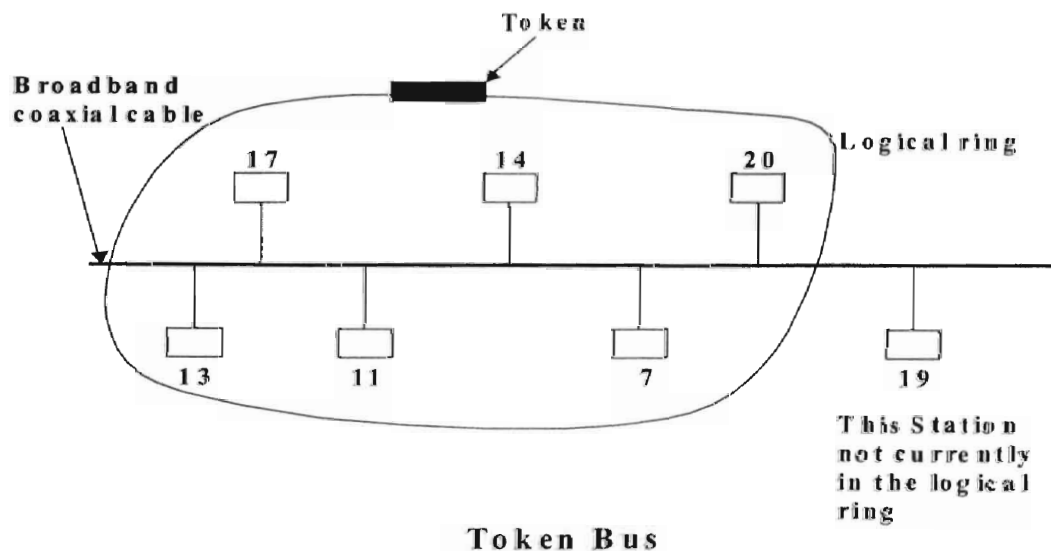


Figure 1.7: Token Bus Network.

Chapter 2

Overall Architecture and Design of WTRP

To put WTRP into a context in terms of its placement in the communication system, we describe the overall system architecture in Figure 2.1. In addition to the communication stack including the Datalink Layer where WTRP will be located, we need Mobility Manager, Channel Allocator, Management Information Base (MIB), and Admission Control Manager. We assume that multiple channels are available, and that different rings are on different channels. Different rings are assigned to different channels by a channel allocator.

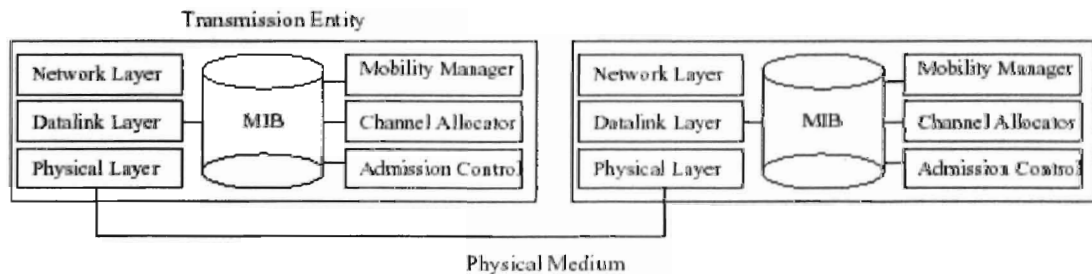


Figure 2.1: Overall System Architecture

2.1 Medium Access Control

Medium Access Control (MAC) enables multiple nodes to transmit on the same medium. This is where WTRP is located. The main function of MAC is to control the timing of the transmissions by different nodes to increase the chance of successful transmission.

In WTRP, the MAC layer performs ring management and timing of the transmissions.

The ring management involves:

1. Ensuring that each ring has a unique ring address.
2. Ensuring that one and only one token exists in a ring. No multiple tokens.
3. Ensuring that the rings are proper.
4. Managing the joining and the leaving operations.

We describe the operation of MAC layer in later section.

2.2 Channel Allocator

The channel allocator chooses the channel on which the station should transmit. If a large number of token rings exist in proximity, their efficiency can be increased by achieving spatial reuse through sensible channel allocation. The idea of spatial reuse is one of the core ideas of the wireless cellular community. The same channel (or a set of channels) can be reused in region A and B, if the two regions are separated by sufficient distance measured in terms of the signal to interference ratio. One way to increase spatial reuse is to reduce the cell size. Reducing the cell size (thus reducing the transmission power) has the following benefits:

1. Increase in capacity
2. Increase in battery life
3. Decrease in equipment costs

In addition, dividing the nodes into multiple rings would reduce the number of nodes in a ring. This would decrease the token rotation time which results in decreased maximum medium access time.

Finding a global optimal solution for the channel allocation in many mobile nodes is a challenging problem. Collecting and maintaining channel allocation information can be a difficult task. The collection and maintenance of information may involve frequent packet transmission. The problem of finding an optimal channel allocation is further complicated by the following factors in the wireless Ad-Hoc networks such as the transmission ranges of the nodes are limited, no stationary base station exists and the boundary of each channel is fluid. So greedy algorithm is the solution for the channel allocation, so each station can access the network topology through MIB. Each node decides on which channel to join in a distributed manner using the information collected.

2.3 Mobility Manager

The Mobility Manager decides when a station should join or leave the ring. The problem that the Mobility Manager has to solve is similar to the mobile hand-off problem. When a mobile node is drifting away from a ring and into the vicinity of another ring, at some threshold the Mobility Manager decides to move to the next ring. The level of connection of a node to a ring can be found from the connectivity table described in later section.

2.4 Admission Control

The Admission Control Manager limits the number of stations that can transmit on the medium. This is to ensure that a level of quality of service in terms of bounded latency and reserved bandwidth is maintained for stations already granted permission to transmit on the medium. There is an Admission Control Manager in each ring. The Admission Control Manager may move with the token but does not have to move every time the token moves. The Admission Control Manager periodically solicits other stations to join if there are “resources” available in the ring. The “resource” of the token ring can be defined in the following way. The MAX MTRT is the minimum of the maximum latency that each station in the ring can tolerate. RESV MTRT is the sum of token holding time (THT) of each station. MAX NoN is the maximum number of node (NoN) that is allowed in the ring.

The Admission Control Manager has to ensure the inequality:

$RESV\ MTRT < MAX\ MTRT$ and $NoN < MAX\ NoN$. Only if these inequalities are satisfied, may the Admission Control Manager solicit another station to join. During the solicitation, the Admission Control Manager also advertises the available resources. Only stations that require less resource than available in the ring may join.

2.5 Policer

The policer monitors the traffic generated by the application. It throttles the application when more traffic than reserved is produced. In the WTRP, because the token

holding timer polices the traffic generated by a station, no special policer module is necessary.

2.6 Management Information Base (MIB)

The Management Information Base holds all the information that each management module needs to manage the MAC module. Majority of this information is collected by the MAC module and stored there.

Chapter 3

WTRP: Protocol Description

3.1 Definitions

- WTRP refers to Wireless Token Ring Protocol, the topic of this report.
- The term “frame” refers to what we pass to the physical layer interface. A “frame” does not include the preambles, the start delimiter, the CRC check, and the end delimiter.
- The terms “station” and “node” are used interchangeably to describe the communication entities on the shared medium.
- The predecessor and the successor of station X describe the station that X receives the token from and the station that the X passes the token to respectively.
- “Incorrect state” means that a node’s view of the topology is wrong. For example node X may believe that node Y is its predecessor, but node Y does not.
- “Stable environment” refers to a state in which the topology of the network is fixed and there are no transmission errors.

3.2 Observations

- Not all stations need to be involved in token passing. Only those stations which desire to initiate data transmission need to be involved.
- Any station may detect multiple tokens and lost tokens. There is no special “monitor” station required to perform token recovery functions.
- Due to errors, stations may not have a consistent view of the ring.

3.3 Frame Format

In WTRP, the successor and the predecessor fields of each node in the ring define the ring and the transmission order. A station receives the token from its predecessor, transmits data, and passes the token to its successor. Here is an illustration of the token frame.

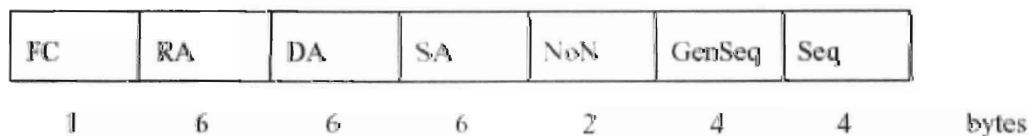


Figure 3.1: Frame Format

FC stands for Frame Control and it identifies the type of packet, such as Token, Solicit Successor, Set Predecessor, etc. In addition, the source address (SA), destination addresses (DA), ring address (RA), sequence number (Seq) and generation sequence (GenSeq) number are included in the token frame. The ring address refers to the ring to which the token belongs.

The sequence number is initialized to zero and incremented by every station that passes the token. The generation sequence number is initialized to zero and incremented at every rotation of the token by the creator of the token. The number of nodes (NoN) in the ring is represented in the token frame and calculated by taking the difference of sequence numbers in one rotation.

3.4 Connectivity Manager

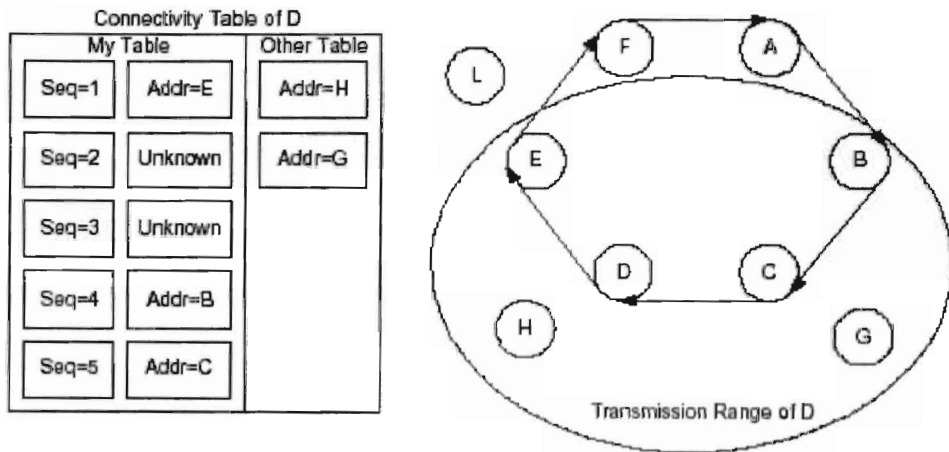


Figure 3.2: Connectivity Table

The Connectivity manager resident on each node tracks transmissions from its own ring and those from other nearby rings. By monitoring the sequence number of the transmitted tokens, the Connectivity Manager builds an ordered local list of stations in its own ring and an unordered global list of stations outside its ring. Station D monitors the

successive token transmission from E to F before the token comes back to D. At time 0, D transmits the token with sequence number 0, at time 1; E transmits the token with the sequence number 1, and so on. D will not hear the transmission from F and A, but when it hears transmission from B, D will notice that the sequence number has been increased by 3 instead of 1. This indicates to E that there were two stations that it could not hear between E and B.

The Ring Owner is the station that has the same MAC address as the ring address. A station can claim to be the ring owner by changing the ring address of the token that is being passed around. Stations rely on implicit acknowledgements to monitor the success of their token transmissions.

An implicit acknowledgement is any packet heard after token transmission that has the same ring address as the station. Another acceptable implicit acknowledgement is any transmission from a successive node regardless of the ring address in the transmission. A successive node is a station that was in the ring during the last token rotation. In other words, the successive stations are those present in the local connectivity table.

Each station resets its IDLE TIMER whenever it receives an implicit acknowledgement. If the token is lost in the ring, then no implicit acknowledgement will be heard in the ring, and the IDLE TIMER will expire. When the IDLE TIMER expires, the station generates a new token, thereby becoming the owner of the ring.

To resolve multiple tokens (to delete all tokens but one), the concept of priority is used. The generation sequence number and the ring address define the priority of a token.

A token with a higher generation sequence number has higher priority. When the generation sequence numbers of tokens are the same, ring addresses of each token are used to break the tie. The priority of a station is the priority of the token that the station accepted or generated. When a station receives a token with a lower priority than itself, it deletes the token and notifies its predecessor without accepting the token. With this scheme, it can be shown that the protocol deletes all multiple tokens in a single token rotation provided no more tokens are being generated.

3.5 Joining the Ring

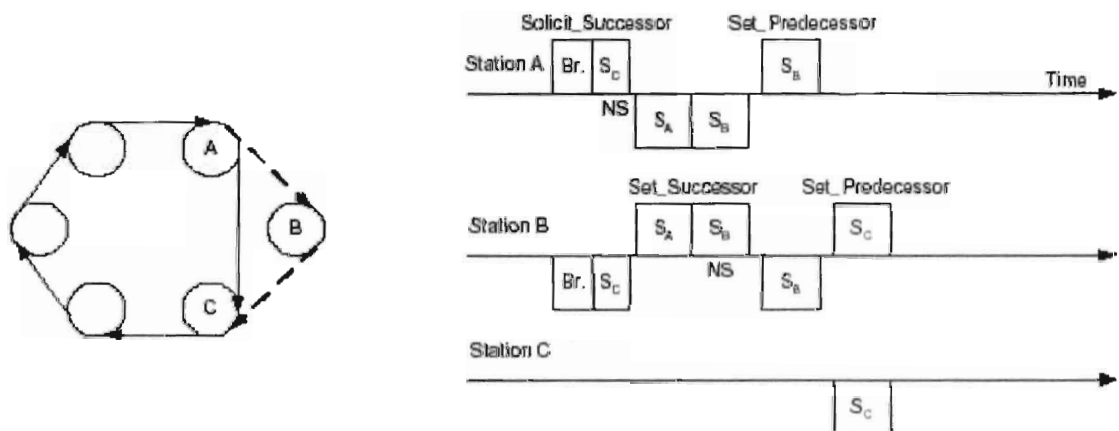


Figure 3.3: Joining

The ring recovery mechanism is invoked when the monitoring node decides that its successor is unreachable. In this case, the station tries to recover from the failure by forming the ring again. The strategy taken by the WTRP is to try to reform the ring by excluding as few stations as possible. Using the Connectivity Manager, the monitoring station is able to quickly find the next connected node in the transmission order. The

monitoring station then sends the SET PREDECESSOR token to the next connected node to close the ring.

WTRP allows nodes to join a ring dynamically, one at a time, if the token rotation time (sum of token holding times per node, plus overhead such as token transmission times) would not grow unacceptably with the addition of the new node. As illustrated in Figure 3.3, suppose station B wants to join the ring. Let us also say that the admission control manager on station A broadcasts (Br.) other nodes to join the ring by sending out a SOLICIT SUCCESSOR that includes successor(C) of A. The Admission Control Manager waits for the duration of the response window for interested nodes to respond. The response window represents the window of opportunity for a new node to join the ring.

The response window is divided into slots of the duration of the SET SUCCESSOR transmission time. When a node, such as B that wants to join the ring, hears a SOLICIT SUCCESSOR token, it picks a random slot and transmits a SET SUCCESSOR token. When the response window passes, the host node, A can decide among the slot winners. Suppose that B wins the contention, then the host node passes the SET PREDECESSOR token to B, and B sends the SET PREDECESSOR to node C, the successor of the host node A. The joining process concludes.

3.6 Exiting the Ring

As shown in Figure 3.4, suppose station B wants to leave the ring. First, B waits for the right to transmit. Upon receipt of the right to transmit, B sends the SET SUCCESSOR packet to its predecessor A with the MAC address of its successor, C. If A can hear C, A tries to connect with C by sending a SET PREDECESSOR token. If A cannot hear C, A will find the next connected node, in the transmission order, and send it the SET PREDECESSOR token.

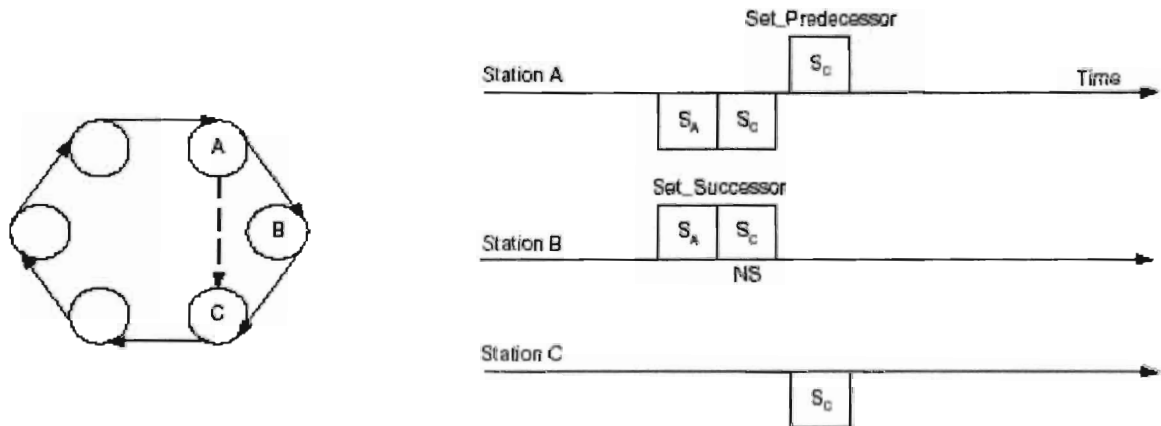


Figure 3.4: Exiting

Interference is eliminated by including NoN in the token packet. When a station detects a ring, it examines the NoN value in the token. If NoN is set to maximum, the station changes its channel and searches for another ring. Otherwise, the station either waits to become a ring member or changes its channel to search for another ring. If the station waits, it suspends transmission and waits for a SOLICIT SUCCESSOR token. As a result, a newcomer station never interferes with the ring.

3.7 Multiple Rings

In Figure 3.5, we can see that the ring address of a ring is the address of one of the stations in the ring, which is called the owner of the ring. In the example, the owner of ring A is station A. Because we assume that the MAC address of each station is unique the ring address is also unique. The uniqueness of the address is important, since it allows the stations to distinguish between messages coming from different rings. Multiple ring management is curial area for the out-of-band message transmission and it is discussed in detail further chapters.

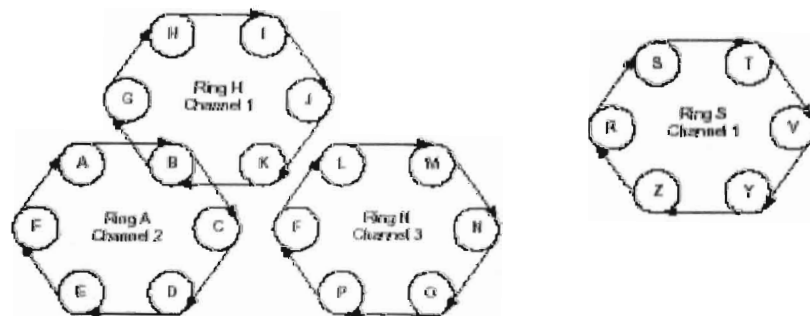


Figure 3.5: Multiple Rings

In Figure 3.5, we can see that the ring address of a ring is the address of one of the stations in the ring, which is called the owner of the ring. In the example, the owner of ring A is station A. Because we assume that the MAC address of each station is unique

the ring address is also unique. The uniqueness of the address is important, since it allows the stations to distinguish between messages coming from different rings. Multiple ring management is a crucial area for the out-of-band message transmission and it is discussed in detail in further chapters. There are possible schemes where a station can belong to more than one ring or a station may listen to more than one ring. To ensure that the ring owner is present in the ring, when the ring owner leaves the ring, the successor of the owner claims the ring address and becomes the ring owner. The protocol deals with the case where the ring owner leaves the ring without notifying the rest of the stations in the ring as follows. The ring owner updates the generation sequence number of the token every time it receives a valid token. If a station receives a token without its generation sequence number updated, it assumes that the ring owner is unreachable and it elects itself to be the ring owner.

Chapter 4

Proposed Extension to WTRP

4.1 Limitations of WTRP:

As we discussed in WTRP if the subset (number of station participating in transmission) of the channel is busy for most of time, this leads to most of the nodes to wait until their idle time expires. The probability of node getting any messages is low if the node waits for 75% of the idle time. This is a potential problem in Wireless Token Ring Network. Each station starts its idle timer as soon as it releases the token to the successor. This is similar to the cases we have in Wired Token Ring Protocol. One of the techniques discussed is early token release, but this may not be the feasible for the wireless ad-hoc networks because some of the nodes may be used as a communication path for the nodes to communicate. Moreover in Wireless Token Ring Protocol there is no explicit acknowledgement so the probability of many number of station's engaged in transmission is reduced.

As we said, WTRP is proposed to support mobility of nodes. This implies the following:

- Bandwidth is limited
- The channel is shared among many stations

Under normal operating conditions, MAC protocol must provide the following guarantees to achieve Quality of Service (QOS) through

- Minimum throughput for each station
- Medium access time for each station is bounded.

4.2 Proposed Extension:

The following exceptional message passing techniques are proposed. This enables QOS, effectively uses the bandwidth and minimizes the token rotation times. This improves the performance of WTRP:

- 1) Simultaneous Out-Of-Band message passing from source station (A) to any station X that is physically in the route of host transmission between source (A) and destination (B).
- 2) Simultaneous Out-Of-Band message passing between station X and Y, which is not in the path of host (original) transmission between source (A) and destination (B) nodes.
- 3) Simultaneous Out-Of-Band message passing between two different stations, which are not in the ring, but utilizing the ring bandwidth for its transmissions.

4.3 Out-Of-Band In-route Transmission (OBIRT):

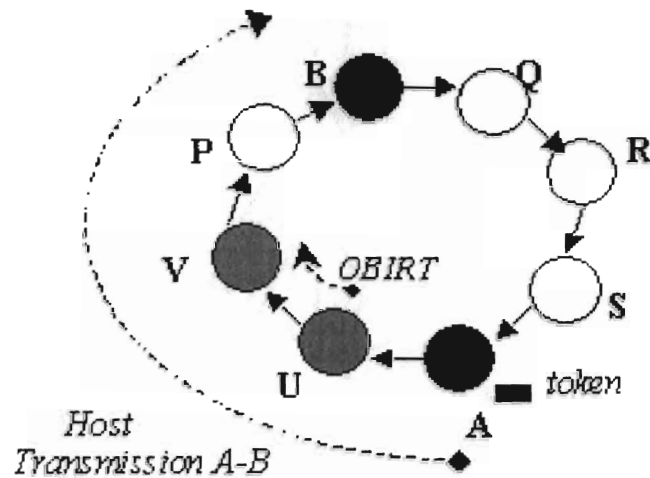


Figure 4.1: Out-Of-Band In-route Transmission (OBIRT)

Assume a well-formed ring for OBIRT. Later this assumption is relaxed as shown in Figure 4.1. As said above when station "A" has the token and wants to communicate to station B it communicates in the path "AUVPB", which is one of the shortest possible routes. Routers may be responsible for establishing the shortest possible route. Each station in the ring and out of the ring is responsible of maintaining the connectivity caches that gives station knowledge about other stations in the ring, which can also be used for determining the path.

Station "A" has the token and wants to send message to station "B", since the message is sent in form of packets. There is a propagation delay for each packet delivered into the medium from A to B and this time can be utilized by stations U and V to transmit messages between them. Once station "U" receives any packet from the host communication it terminates any Out-Of-Band-In-Route. This ensures medium access time for each station time for each station is bounded.

4.4 Out-Of-Band Off-route Transmission (OBORT):

Let's assume a well-formed ring for OBORT, later this assumption is relaxed. As shown in Figure4.2 above when the station "A" has the token and wants to communicate to the station D it communicates through the path of "ABCD". Simultaneously if the station "P" wants to transmit data to station "U" it can utilize the channel bandwidth and since the host transmission doesn't interfere with OBORT the medium access time for each station is bounded. This in turn decreases the total token rotation time and the channel bandwidth or medium is fully utilized.

OBORT is determined based on the connectivity caches maintained by each station. Since the connectivity table is refreshed for each token rotation time and each station knows the total number of station in the ring.

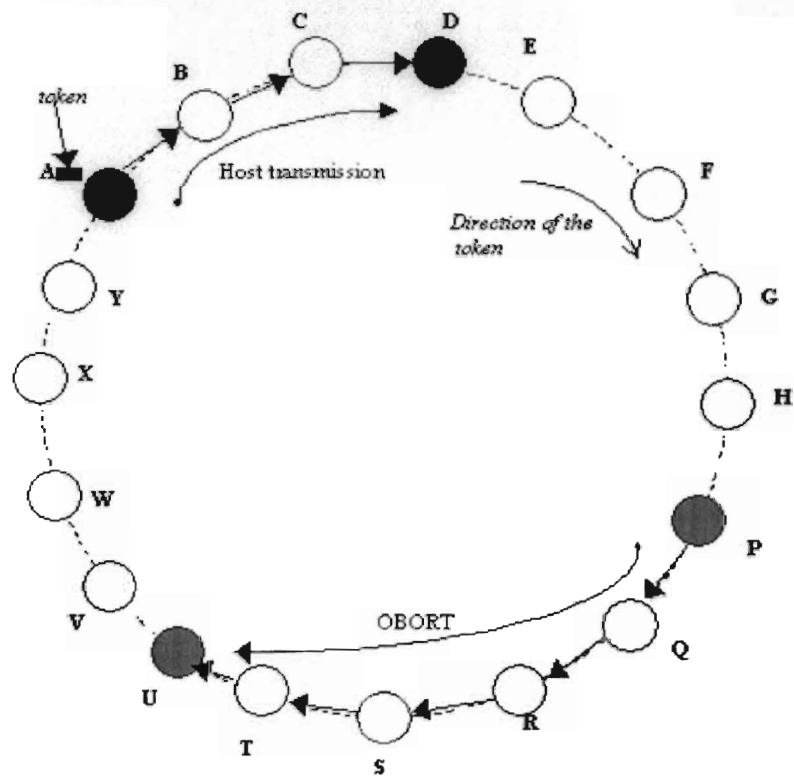


Figure 4.2: Out-Of-Band Off-route Transmission (OBORT)

4.5 Out-Of-Band Not-In-ring Transmission (OBNIRT):

As shown in Figure 4.3 if the station A has the token and wants to communicate to the station D it goes through the path of "ABCD". Simultaneously if the station I needs to communicate to station J it can utilize the channel bandwidth and since the host transmission doesn't interfere with this communication the medium access time for each station is still bounded. All the stations have two connectivity cache tables one for "My-Ring" cache table and "Not-In-My-Ring" cache table for the stations not in their host ring

(Ring- Address (RA) for that station is different). This can be used for making communication between two different stations that is not in the host ring.

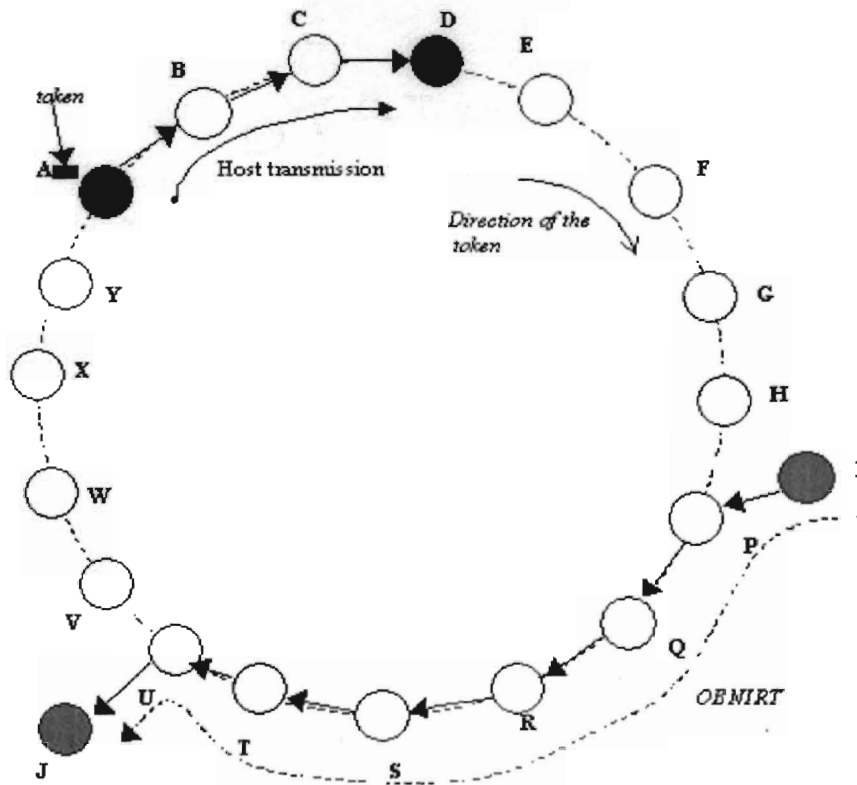


Figure 4.3: Out-Of-Band Not-In-Ring Transmissions (OBNIRT)

OBNIRT is determined based on “Not-In-My-Ring” connectivity caches maintained by each station. The connectivity table is refreshed for each token rotation time and each station knows the total number of station in the ring as well as the station not in the ring but in their transmission range.

4.6 State Diagram for WTRP Extension:

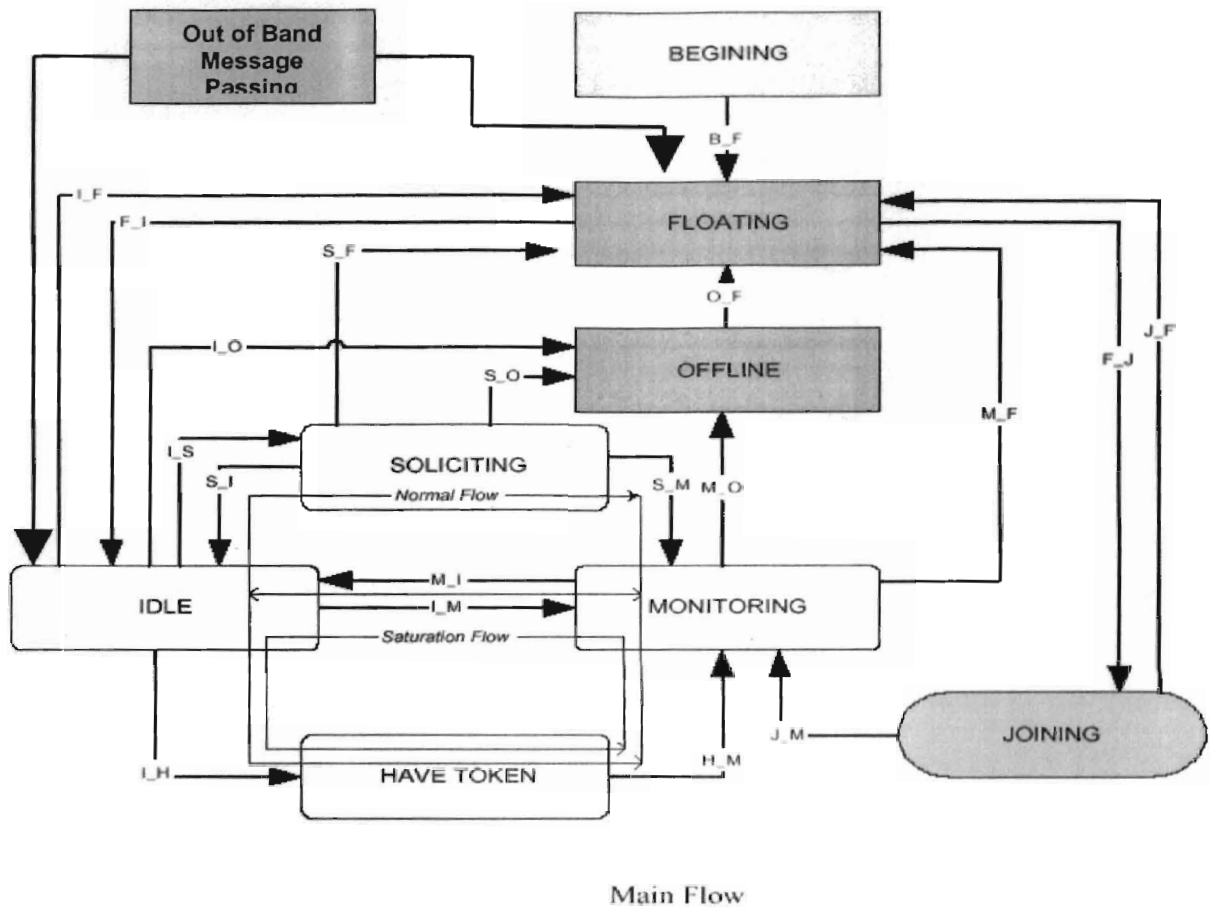


Figure 4.4: State diagram

4.6.1 Beginning State

Beginning state is a virtual state that represents the starting of the protocol. There is only one transition as seen in Figure 4.4 and the station directly goes to the floating state.

4.6.2 Floating State

Floating state is the state where a station resets its parameters and waits to join a ring. When a station passes to the floating state, it resets its station parameters, cleans up its packet queues, and initializes the claim token timer. A station passes to the floating state at the beginning and when there is a failure in the ring.

When the station is self ring (When the successor and predecessor of the station is itself and station is in idle state.) and detects another ring, it goes to floating state. If the station does not get the token in the idle state or can not join to a ring, idle timer expires and it goes to floating state. If the station detects a ring and is in the floating state, it waits to be invited from the ring and suspends its transmission in order not to interfere the ring transmission. If the station does not detect a ring, claim token timer expires and it goes to the idle state and creates a self ring. If the station gets a solicit successor token, and if it wants to join a ring, the station sends a set successor token and goes to joining state. The state transitions from or to the floating state can be seen in Figure 4.4 as F_I (Floating to Idle).

The proposed extension is implemented in the floating state and idle state. Any time there is a transmission primary transmission is given preference compared to that of

the out-of-band message transmission. One hop node is maintained between the original and secondary transmission. If a station receives a out-band transmission and primary transmission simultaneously then this hop node will shut down any secondary transmission. This way the worst bound case of this protocol will never goes over Wireless Token Ring Protocol performance.

4.6.3 Offline State

When a station goes to the offline state, it initializes the station, clears the packet queues, and adds offline timer to the scheduler. Since offline timer is twice the max token rotation time, the wait period in the offline state gives sufficient time to the former ring members to realize that the station is out of the ring. This prevents the station from joining a ring before the ring is closed. A station goes to the offline state if it belongs to a ring other than self ring when it detects another ring or when a station joins a ring but fails to pass the token to its successor. In the offline state, a station waits and does nothing until the offline timer expires. From the offline state, a station only goes to floating state. The state transitions from or to the floating state can be seen in Figure 4.4.

4.6.4 Joining State

When a station goes to the joining state, it initializes the contention timer. A station goes to joining state only from floating state. When a station receives solicit successor and decides to join the ring, the station sends set successor and goes to the joining state. If the station receives set predecessor, this means that joining is successful. Therefore, the station sends set predecessor and passes to the monitoring state. If the

station does not get set predecessor, contention timer expires and this means a failure in joining and the station goes back to the floating state as seen in Figure 4.4.

4.6.5 Soliciting State

When a station goes to the soliciting state, the station initializes solicit wait timer and sets the station->num node1 to MAX NoN+1 in order to suspend transmission of other stations if it is not a self ring. When the station is in the idle state and receives the token, it checks its queues and if they are empty, it decides to send solicit successor. If the decision is positive, then it ends solicit successor token and goes to the soliciting state. If the station is self ring, it adds solicit successor timer and when it expires, it sends solicit successor token and goes to the soliciting state.

If a station receives set successor in the soliciting state, this means that there is a station responding to it's solicit successor, then the station sends set predecessor and goes to the monitoring state. If there is no response to the invitation, solicit wait timer expires and the station goes to idle state if it is self ring and monitoring state if it is not self ring as seen in Figure 4.4.

When the station is either at the idle state or in the beginning state then there is a possibility that the station can send messages when it obeys the time constraints. If the station is in the idle state for 75% of the maximum idle time then it can start a out-of-band message, provided if the immediate upstream neighbor and the downstream neighbor is not transmitting any other messages. We make sure there is always one node

hop distance between the original transmission and the out-of-band message transmission. This ensures the collision avoidance between the two streams of messages.

When a station is in the process of sending out-of-band message and happens to receive a token for the original transmission. In this case the station will continue to send message, the only difference is changing the message type from out-of-band message to primary message. This will make sure the station will obey WTRP protocol.

Chapter 5

Simulation and Results

5.1 Performance Measures

To investigate the performance of the Out Of Band Message Passing over the Wireless Token Ring Protocol, both protocols are studied with the growing list of node. The scenario is created with the list of nodes in a circular link lists with random number of messages in their message queues. With the test case in place this is applied to both the protocol with and without out-of-band message passing.

The performance is measured by the comparing the results of the following measures

- Average token holding time by the stations
- Average Idle time by the ring
- Maximum token rotation time
- Simultaneous message passing between the station

5.2 Steps involved in simulation

- 1) Create the node in a circular link list (assume a well formed ring). The ring is fault tolerant.
- 2) Add message queues in a Poisson distribution with varying length of messages.

- 3) Start the simulation with the base case and record the sensitive parameters for performance measure.
- 4) With the same input test case, start WTRP with Out-Of-Band transmission and record the parameters.
 - a. Start the idle timer of each station when a node releases token.
 - b. If the idle time expires 75% of the total idle time then it can transmit any out-of-band messages if a node wishes to transmit messages.
 - c. Any node transmitting out-of-band message listen to any primary message transmission from its upstream neighbor will immediately inform the downstream successor to stop any secondary transmission thus allowing the original transmission.

5.3 Charts

To measure the performance of Out-Of-Band Message passing in wireless token ring protocol, average idle time, average total holding time and average message delay of stations. They are plotted against number of nodes as follows

1. Token holding time for each node is calculated for Out-Of-Band message and compared with WTRP.
2. Token holding time for each node is calculated for Out-Of-Band message and compared with WTRP.

3. Token holding time for each node is calculated for Out-Of-Band message and compared with WTRP.

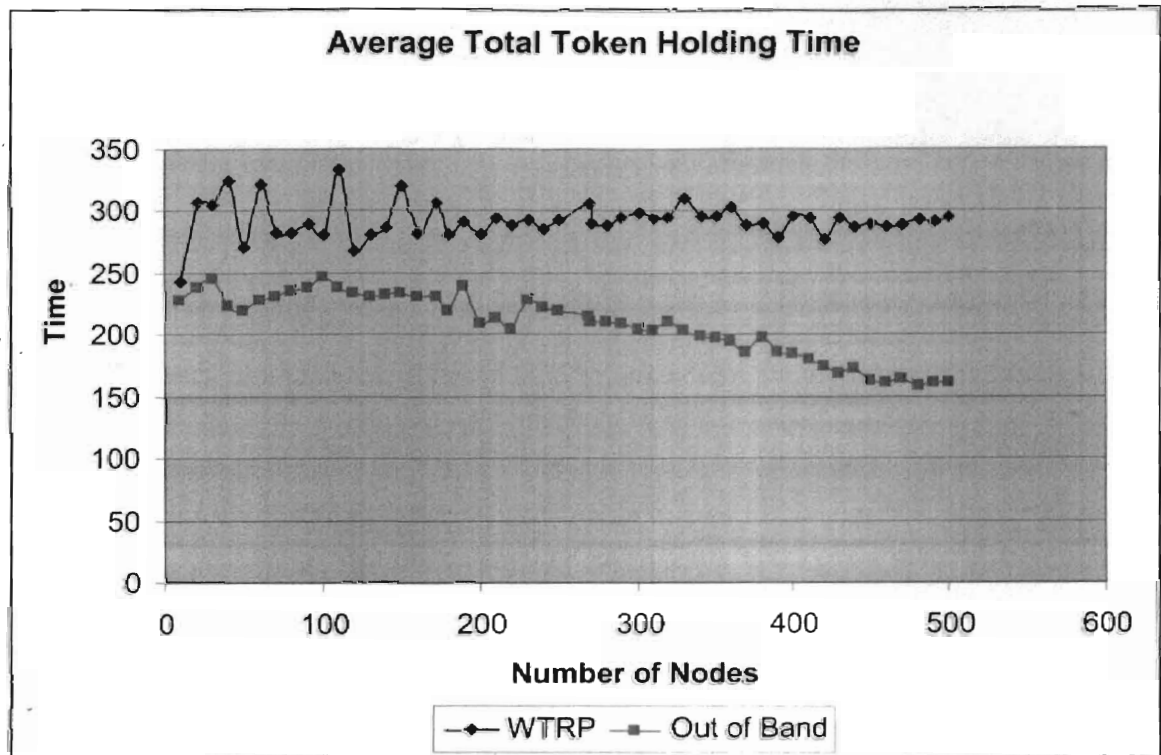


Figure 5.1: Average Total Token Holding time of Stations

From the figure 5.1 we can see that as the number of nodes increase the average total token holding time increases in WTRP protocol as expected. Since each station has to wait until the maximum of Token rotation time before it gets its chance to transmit message again. This causes all the other station to wait even if they are idle before transmitting its messages. In Out of band message passing we can see that Average Token Holding Time for the station decrease when the number of nodes increases. Since each station has already sent out some of its messages in out-of-band message transfer, it doesn't hold the

token to the Maximum Token Holding Time when it receives token. This reduces the time taken by stations to transfer its primary messages. The random nature of the Total Token Holding time is caused due to random distribution of the messages inserted in the message queue of each station.

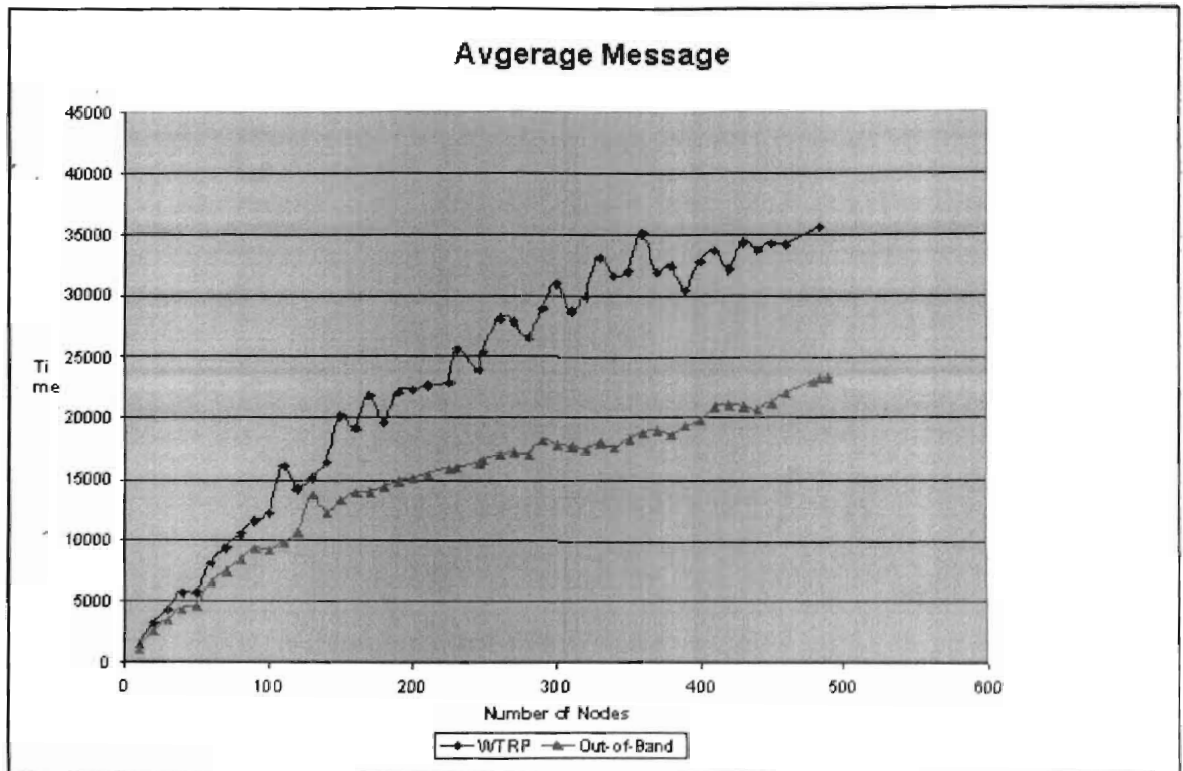


Figure 5.2: Average Message Delay of Stations

Figure 5.2 gives us the comparison of the average message delays of the station in wireless token ring protocol versus WTRP with Out-of-band-messages. When the stations are allowed to use only WTRP the average message delay increases with the number of the nodes. Message queue is populated with random message arrival time

using Poisson distribution. With Out-of-band-message passing in WTRP reduces the maximum message delay and also reduces the average message delay by the stations. As we can see from figure 5.2 as the number of nodes increases the percentage of decrease in message delay increases. This is due to more number of nodes being idle when the ring grows bigger. This is utilized by out-of-band message passing technique to pass more simultaneous messages. If the channel is utilized to its maximum then the protocol behaves as Wireless Token Ring Protocol. This makes sure the channel bandwidth is bounded and each station has a fair chance to utilize the channel bandwidth. Thus Out-of-Band message passing in WTRP enables Quality of Service.

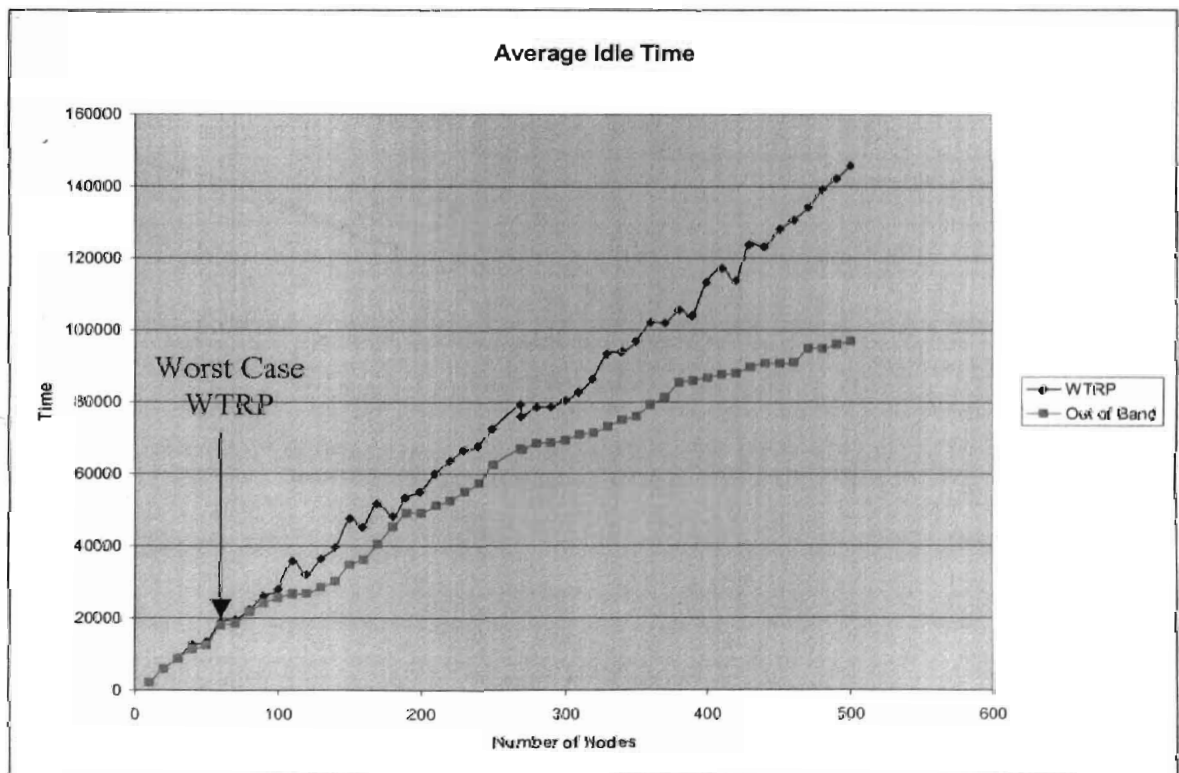


Figure 5.3: Average Idle time of the Stations

From figure 5.3 we can see that average idle time for stations increases when the number of nodes increases. Average idle time for stations reduces as they are busy sending out-of-band messages when they are idle for more than 75% of the maximum idle time. This in-turn utilizes the channel bandwidth to its maximum.

5.4 Results

Using the simulation the following observations were made on Out-Of-Band Message passing in Wireless Token Ring Network. Out-Of-Band message passing

- Minimizes the token holding time by the stations
- Utilizes more Channel Bandwidth
- Reduces Total Idle time for each station.
- Minimizes the Maximum Token Rotation time
- Simultaneous message passing is achieved

Chapter 6

Conclusions

6.1 Conclusions

Out-Of-Band message passing with the extension to Wireless Token Ring protocol increase the channel utilization and reduces the station idle time drastically when the activity in the ring is within the subset of the huge ring. This also reduces the message delay. More number of messages will be transmitted within the same window of time.

6.2 Future Work

1. Out-Of-Band message passing could be extended to multiple rings by implementing gateway nodes to increase the channel utilization effectively.
2. An effective protocol along with minimal explicit acknowledgement to the message can be devised.
3. A self-learning mechanism at node level can be implemented to learn more about the ring. This will reduce collision when transmitting out-of-band messages.
4. The above approach can be implemented in Network Simulator and the results can be studied.

Chapter 6

Conclusions

6.1 Conclusions

Out-Of-Band message passing with the extension to Wireless Token Ring protocol increase the channel utilization and reduces the station idle time drastically when the activity in the ring is within the subset of the huge ring. This also reduces the message delay. More number of messages will be transmitted within the same window of time.

6.2 Future Work

1. Out-Of-Band message passing could be extended to multiple rings by implementing gateway nodes to increase the channel utilization effectively.
2. An effective protocol along with minimal explicit acknowledgement to the message can be devised.
3. A self-learning mechanism at node level can be implemented to learn more about the ring. This will reduce collision when transmitting out-of-band messages.
4. The above approach can be implemented in Network Simulator and the results can be studied.

References

- [1] Akyildiz, I.F.; McNair, J.; Martorell, L.C.; Puigjaner, R.; Yesha, Y. Medium access control protocols for multimedia traffic in wireless networks. *IEEE Network*, vol.13, (no.4), IEEE, July-Aug. 1999. p.39-47.
- [2] Berkeley Web Over Wireless Group, <http://wow.eecs.berkeley.edu>.
- [3] D. Lee, B. Dunder, M. Ergen, A. Puri, *Socket Interface for User Applications in WOW Project*, <http://wow.eecs.berkeley.edu>.
- [4] D. Lee, *Wireless Token Ring Protocol*, Master's Thesis at Berkeley, 2001.
- [5] *Draft International Standard ISO IEC 8802-11—IEEE P802.11/D10*, 14 January 1999.
- [6] Eardley, P., Mihailovic, A. and Suihko, T., "A Framework for the Evaluation of IP Mobility Protocols", BT, Advanced Communications Technology Centre, King's College, London, Nokia / VTT Information Technology, The 11th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications, Volume 1, 2000, pp.451-457.
- [7] Haas, Z.J. On the performance of a medium access control scheme for the reconfigurable wireless networks. MILCOM 97. MILCOM 97 Proceedings, (vol.3), Monterey, CA, USA, p.1558- 64, 2-5 Nov. 1997
- [8] Hannikainen, M.; Knuutila, J.; Letonsaari, A.; Hamalainen, T.; Jokela, J.; Ala-Laurila, J.; Saarinen, J. *TUTMAC: a medium access control protocol for a new multimedia wireless local area network*, Ninth IEEE International Symposium on Personal, Indoor and Mobile Radio Communications, New York, NY, USA: IEEE, 1998. p.592-6 vol.2. 3 vol. 1574 pp.
- [9] H. Shim, T. J. Koo, F. Hoffmann, S. Sastry *A Comprehensive Study on Control Design of Autonomous Helicopter* In Proceedings of IEEE Conference on Decision and Control, Florida, December 1998
- [10] *High Performance Radio Local Area Network (HIPERLAN), Type 1; Functional specification*, ETS 300 652, Radio Equipment and systems (RES) October 1996
- [11] *International Standard ISO IEC8802-4: 1990—ANSI/IEEE Std. 802.4-1990*.
- [12] J. Postel *Internet Control Message Protocol RFC-792*

- [13] K. Nichols, S. Blake, F. Baker, D. Black, *RFC2474 Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers*, December 1998.
- [14] La Porta, T.F., Sabnani, K.K. and Gitlin, R.D., "*Challenges for Nomadic Computing: Mobility Management and Wireless Communications*", <www.bell-labs.com/user/tlp/nomad.pdf>
- [15] P.Varaiya. Smart Cars on Smart Roads: Problems of Control. IEEE Transactions on Automatic Control, 38(2):195-207, February 1993. Kim, H.Y. and Hwang, C.S., "*An Efficient Connection Scheme for Mobile IP*", Ninth IEEE International Conference on Networks, 2001, pp. 396-400.
- [16] Pomerantz, Ori. *Linux Kernel Module Programming Guide*
- [17] Rubini, Alessandro. *Linux Device Drivers* O'Reilly & Associates, Inc. Sebastopol, CA. 1998.
- [18] Salles,R.; Gondim, P. A new multiple access protocol for multimedia wireless networks. ISCAS '98. Proceedings of the 1998 IEEE International Symposium on Circuits and Systems, Monterey, CA, USA, 31 May-3 June 1998. p.538-41 vol.4.
- [19] S. Herzog, *RFC2750 RSVP Extensions for Policy Control*, January 2000.
- [20] Tanenbaum, S.A., "*Computer Networks* ", Third Edition, 1996. <<http://www.iprg.nokia.com/~charliep/txt/optim/optim.txt>>
- [21] *The Teja Technical Reference* — Teja Technologies
- [22] <http://www.bluetooth.com>
- [23] <http://www.wavelan.com>
- [24] Wave LAN/IEEE PCMCIA device driver for Linux (WVLAN49) Lucent Technologies Inc, 1998-1999.
- [25] Wireless Token Ring Protocol, Mustafa Ergen, Duke Lee, Anuj Puri, Pravin Varaiya, Roberto Attias. Raja Sengupta Stavros Tripakis. University of California Berkeley.

Appendix

```
/*
 * Simulation of Wireless Token Ring protocol
 *
 * FileName:          wtrp_main.cpp
 *
 * Purpose:
 *
 *          This program simulates Wireless Token ring protocol(WTRB),
 *          to calculate the Token Holding of each station,cumulative Message
 *          delays for each station,Token Rotation time.
 *
 *          Calculating the above said vaules with existing protocol and
 *          comparing them with Out-of-Bounds message passing in WTRB.
 *
 * Created:
 *
 *          Sivakumar C Janakiraman
 *
 * Last Modified:
 *
 *          04/18/2003
 */
```

```
#include "wtrp_main.h"
```

```
int main(int argv,char *argc[])
{
    FILE *fpout,*fpout1;
    //To give current time as seed for the rand
    srand((unsigned)time(NULL));

    //Intilize global variables
    tick = 1;
    max_TRT = 0;
```

```

fpout = fopen("out.txt","w");
fpout1 = fopen("out1.txt","w");
//Prepare Token ring with random amount of message list

/* Simulating by varying the number of nodes and the Maximum holding
   time of the token by each station linearly*/

fprintf(fpout,"Num_Nodes \t THT \t Idle_time \t Avg_MsgDelay\n");
fprintf(fpout1,"Num_Nodes \t THT \t Idle_time \t Avg_MsgDelay\n");

NUM_NODES =10;
MAX_TOK_HOLD_TIME = 10;

while(NUM_NODES <=500)
{
    prepare_tokenring();
    simulate_tokenring();
    print_node_info(fpout);
    tick =1;
    initialize_tokenring();
    simulate_out_of_band();
    print_node_info(fpout1);

    free_Memory();

    NUM_NODES += 10;
    MAX_TOK_HOLD_TIME += 10;
    head_node = NULL;
    tail_node = NULL;
}
}

/*-----*/
*
* Name : prepare_tokenring
*
* This function prepares the token ring for the Number of Nodes,
* Maximum-Token-Holding time and the Message length.
*
* 1. First create a message queue for each node. This creates messages with
*    in assigned maximum message length
* 2. Creates a Circular link node and attach this message queue to this node.

```

```

*-----*/

void prepare_tokenring()
{
    int i,node_id,num_msg,temp_des =0;

    LINK *head_msg,*temp_msglink;
    MESSAGE *tmpmsg;

    for(node_id=1;node_id<=NUM_NODES;node_id++)
    {
        head_msg = NULL;
        num_msg = get_random(MAX_NUM_MSG);

        for(i=0;i<num_msg;i++)
        {
            tmpmsg = (MESSAGE *)malloc(sizeof(MESSAGE));
            create_message(tmpmsg,node_id);

            temp_msglink = (LINK*) malloc(sizeof(LINK));
            temp_msglink->msg = tmpmsg;
            temp_msglink->next = NULL;

            addMsg(&head_msg,temp_msglink);
        }
        create_station(head_msg,node_id,1);// test for one ring
    }
}

int get_random(int range)
{
    int temp_rand=0;
    do
    {
        temp_rand = rand() % range;
    }while(temp_rand ==0);

    return temp_rand;
}

```



```

void print_node_info(FILE *fpout)
{
    LINK *current;
    NODE *current_node;
    int tot_msg_delay=0 ,num_msg =0;
    float avg_msg_delay=0;
    float tot_tok_hold_time=0,tot_idle_time=0, tot_avg_msg_delay=0;

    current_node = head_node;
    while(current_node ->next != head_node)
    {
        current = current_node->station.msg_head ;

        /*fprintf(fpout, "\n===== \n");
        fprintf(fpout, "**** Station Mac Addr %d \t", current_node->station.mac_addr);
        fprintf(fpout, "Token holding time \t\t: %d \n", current_node-
>station.tok_hold_time);
        fprintf(fpout, "Station Idle time \t\t: %d \n", current_node->station.idle_time );*/

        /*fprintf(fpout, "%d \t", current_node->station.mac_addr);
        fprintf(fpout, "\t\t %d", current_node->station.tok_hold_time);
        fprintf(fpout, "\t\t %d", current_node->station.idle_time );
        */

        while(current !=NULL)
        {
            tot_msg_delay += current->msg->delay;
            num_msg++;
            current = current->next;
        }
        avg_msg_delay = (float)tot_msg_delay / num_msg;

        /* Calculating the Total token holding time and total
        idle time of the station
        */
        tot_tok_hold_time += current_node->station.tok_hold_time;
        tot_idle_time += current_node->station.idle_time;
        tot_avg_msg_delay += avg_msg_delay;

        current_node = current_node->next;
    }
}

```

```

    }

    fprintf(fpout, "%d\t", NUM_NODES);
    fprintf(fpout, "%.3f\t", (float)(tot_tok_hold_time / NUM_NODES));
    fprintf(fpout, "%.3f\t", (float)(tot_idle_time / NUM_NODES));
    fprintf(fpout, "%.3f\n", (float)(tot_avg_msg_delay/NUM_NODES));
}

/*-----*
 *
 * Name : simulate_tokenring
 *
 * This function simulate the Wireless token ring network without out-of-band
 * message passing. This starts with a Token Owner and takes message from the
 * queue and starts processing each messages. Then this calculates the param
 * for performance measure.
 *
 *-----*/

void simulate_tokenring()
{
    NODE *curr_node = NULL;

    //Start the simulation with the head_node
    curr_node = head_node;
    curr_node->station.has_token = TRUE;

    while(some_node_msg_pending(curr_node) == TRUE)
    {
        //Traverse all the nodes in the ring
        curr_node->station.waiting_ack = FALSE;
        if(curr_node->station.has_token == TRUE)
        {
            /*-----
             * Calculate the token realsetime, Token holding time
             * idle time, Arrival time etc.
             -----*/

            calculate_params(curr_node);
            while(tick <= curr_node->station.tok_rel_time)
            {
                tick = tick +1;
                if(isMsgPending(curr_node)== TRUE)
                {

```

```

        send_msg( curr_node);
        curr_node->station.waiting_ack = TRUE;
        curr_node->station.next_msg = curr_node->station.next_msg->next;
        //tick increment is done in send message
    }
    else
    {
        if(curr_node->station.waiting_ack == TRUE)
            tick = tick +1;
        else
            break;
    }
}

curr_node->station.has_token = FALSE;
curr_node->station.waiting_ack = FALSE;
curr_node->station.last_tok_rel_time = tick;

curr_node->station.tok_hold_time += (tick - curr_node->station.tok_arr_time);

total_THT += (tick - curr_node->station.tok_arr_time);
curr_node->next->station.has_token = TRUE; //Give the token to the Next
Station;
curr_node = curr_node ->next ;
}
}
}

/*-----*/
*
* Name : sed_msg
*
* This takes the first available message from the message queue
* and sends to the destination address and calculate the parameters tied
* with the message.
*
* @param:
*     Input: Station_node
*     From where the message is sent to:
*-----*/
void send_msg(NODE *curr_node)
{
    LINK *curr_msg;
    curr_msg = curr_node->station.next_msg;

```

```

curr_node->station.tok_rel_time = ((tick + 5 * curr_msg->msg->len) < (curr_node-
>station.tok_arr_time + MAX_TOK_HOLD_TIME))
? (tick + 5 * curr_msg->msg->len) : (curr_node->station.tok_arr_time +
MAX_TOK_HOLD_TIME); // x = (a<b)?a:b

curr_msg->msg->sent = TRUE;
curr_msg->msg->delay = tick - curr_msg->msg->arr_time;

/*fprintf(stderr,"Station %d Arr Time %d current time %d\n",curr_node-
>station.mac_addr,
curr_msg->msg->arr_time,tick);

fprintf(stderr,"Token Release time %d\n",curr_node->station.tok_rel_time ); */

tick = tick + 5 * curr_msg->msg->len; //Processing time for sending each message
}
void calculate_outband_params(NODE *curr_node)
{
curr_node->station.idle_time += (tick - curr_node->station.last_tok_rel_time);
//curr_node->station.tok_rel_time = tick + (curr_node->station.msg_head->msg->len
}

/*-----*
*
* Name : calculate_params
*
* This calculate the parameters we needed for performance measure.
*
*
* @param:
* Input: Station_node
* From where the message is sent to:
*-----*/

void calculate_params(NODE *curr_node)
{
curr_node->station.tok_arr_time = tick;
curr_node->station.tok_rel_time = MAX_TOK_HOLD_TIME + tick;
curr_node->station.idle_time += (tick - curr_node->station.last_tok_rel_time);

if((max_TRT <(tick - curr_node->station.last_tok_rel_time))
{
max_TRT = tick - curr_node->station.last_tok_rel_time;
}
}

```

```

    }
}

/*-----
 * Returns if there is any message pending in the token ring
 *-----*/

int some_node_msg_pending(NODE *start_node)
{
/*****
 *   start node is not the first node always its just a
 *   marker for finding if the traversal reached the starting point
 *****/

    NODE *curr_node;
    int has_msg = 0;

    curr_node = start_node;
    while(curr_node->next != start_node)
    {
        if(curr_node->station.next_msg != NULL)
        {
            has_msg = 1;
            break;
        }
        else
            curr_node = curr_node->next;
    }

    return has_msg;
}

/*****
 *   Is there any message pending for the current station
 *****/

int isMsgPending(NODE *curr_node)
{
    // If there is any message to be sent and if that
    // message is pending with respect to the current time

    if(curr_node->station.next_msg != NULL &&
        curr_node->station.next_msg->msg->arr_time <= tick)
    {

```

```

    return TRUE;
}
else
    return FALSE;
}

```

```

/*-----*
 *
 * Name : simulate_tokenring
 * Simulating Out_of_band message passing in Wireless Token Ring protocol.
 *
 * This function simulate the Wireless token ring network with out-of-band
 * message passing. This starts with a Token Owner and takes message from the
 * queue and starts processing each messages. Then this calculates the param
 * for performance measure.
 *
 *-----*/

```

```

void simulate_out_of_band()
{
    NODE *curr_node = NULL;

    //Start the simulation with the head_node
    curr_node = start_node = head_node;
    curr_node->station.has_token = TRUE;

    while(some_node_msg_pending(curr_node) == TRUE){
        if(curr_node->station.has_token == TRUE){

            tick ++;
            /*-----
             Calculate the token realsetime, Token holding time
             idle time, Arrival time etc.
             -----*/
            calculate_params(curr_node);

            if(curr_node->station.waiting_ack == FALSE){
                if(tick <= curr_node->station.tok_rel_time){
                    if(isMsgPending(curr_node) == TRUE){
                        send_msg(curr_node);
                    }
                }
            }
        }
    }
}

```

```

curr_node->station.waiting_ack = TRUE;
curr_node->station.next_msg = curr_node->station.next_msg->next;
}
}
else{
curr_node->station.has_token = FALSE;
curr_node->station.waiting_ack = FALSE;
curr_node->station.last_tok_rel_time = tick;
curr_node->station.tok_hold_time += (tick - curr_node->station.tok_arr_time);

total_THT += (tick - curr_node->station.tok_arr_time);
//Give the token to the Next Station;
curr_node->next->station.has_token = TRUE;
start_node = curr_node->next;
/* This is a global variable set to determine who
holds the token */
station_has_token= curr_node->next->station.mac_addr;

}
}
else if(tick == curr_node->station.next_avail_time){
curr_node->station.waiting_ack = FALSE;
}
}
/* Other stations in the ring that doesn't have
Token but can still potetially send messages */
else{
int mac_addr = curr_node->station.mac_addr;

if((mac_addr + 1) == station_has_token ||
(mac_addr - 1) == station_has_token)
{
/*We cannot send a out of band messages if the stations
near by is already transmitting any messages */
break;
}
else{
int idle_time =0;
/*fprintf(stderr," Tick %d I don't know where am I in the ring now\n",tick);*/

idle_time = tick - curr_node->station.last_tok_rel_time;

if(idle_time >= (0.75 * MAX_TOK_HOLD_TIME)){
//calculate_params(curr_node);
if(curr_node->station.waiting_ack == FALSE){
if(tick >= curr_node->station.next_avail_time ){

```

```

    if(isMsgPending(curr_node)== TRUE){
        send_outofband_msg(curr_node);
        curr_node->station.waiting_ack = TRUE;
        curr_node->station.next_msg = curr_node->station.next_msg->next;
    }
}
else{
    curr_node->station.has_token = FALSE;
    curr_node->station.waiting_ack = FALSE;
}
}
}
if(tick >= curr_node->station.next_avail_time){
    curr_node->station.waiting_ack = FALSE;
}
}
}

//if(curr_node->next == start_node)
//    tick = tick + 1;

curr_node = curr_node->next ;
}
}

```

```

void send_outofband_msg(NODE *curr_node)

```

```

{
    LINK *curr_msg;
    curr_msg = curr_node->station.next_msg;

    curr_node->station.tok_rel_time = ((tick + 5 * curr_msg->msg->len) < (curr_node->station.tok_arr_time + MAX_TOK_HOLD_TIME))
    ? (tick + 5 * curr_msg->msg->len) : (curr_node->station.tok_arr_time +
    MAX_TOK_HOLD_TIME); // x = (a<b)?a:b

```

```

    curr_msg->msg->sent = TRUE;
    curr_msg->msg->delay = tick - curr_msg->msg->arr_time;

```

```

    curr_node->station.next_avail_time = tick + 5 * curr_msg->msg->len; //Processing time
    for sending each message

```



```

}

void free_Memory()
{
    NODE *curr_node = NULL, *temp_node= NULL;

    //Start the simulation with the head_node
    curr_node = head_node->next;

    /*
     To free the circular link list into single link list.
    */
    head_node ->next = NULL;

    while(curr_node !=NULL){
        free_messages(curr_node);
        temp_node = curr_node;
        curr_node = curr_node ->next ;
        free(temp_node);
    }
}

void free_messages(NODE *curr_node)
{
    LINK *curr_msg = NULL, *temp = NULL;

    curr_msg = curr_node->station.msg_head;

    while(curr_msg !=NULL){
        temp = curr_msg;
        curr_msg = curr_msg->next;
        free(temp);
    }
}

/*=====
  Initialize Token ring for Out-of_hand simulation
  =====*/

void initialize_tokenring()
{
    NODE *curr_node = NULL;
    LINK *msg_head = NULL;

    curr_node = head_node;

    /* Tail node is initiated to NULL to

```

```

    establish a link list then tail pointer
    is adjusted */
tail_node ->next = NULL;
msg_head = curr_node ->station.msg_head;
tick =1;
max_TRT =0;

while(curr_node !=NULL ){
    curr_node->station.idle_time = 0;
    curr_node->station.last_tok_rel_time =0;
    curr_node->station.has_token = FALSE;
    curr_node->station.tok_arr_time = 0;
    curr_node->station.tok_hold_time =0;
    curr_node->station.tok_rel_time =0;
    curr_node->station.last_tok_rel_time =0;
    curr_node->station.waiting_ack = FALSE;
    curr_node->station.next_msg = msg_head; //Initial msg to process
    curr_node->station.primary_msg = FALSE;
    curr_node->station.next_avail_time = 0;
    initialize_msg(msg_head);

    curr_node = curr_node ->next;

}
/* Establish the tail link */
tail_node ->next = head_node;
}

void initialize_msg(LINK *msg_head){
    LINK *curr_msg = msg_head;

    while(curr_msg != NULL){
        curr_msg->msg->delay = 0;
        curr_msg->msg->sent = FALSE;

        curr_msg = curr_msg ->next;
    }
}

```

FileName: wtrp_main.h

```
/*
 * Global declaration of functions and datastructure
 *
 */
```

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
```

```
#define TRUE 1
#define FALSE 0
#define MAX_MSG_LEN 5
#define MAX_NUM_MSG 10
/*Maximum token holdins time for each station */
#define MAX_TOK_HOLD_TIME 50
#define MAX_ARR_TIME 100
#define NUM_NODES 5
```

```
typedef struct message
{
    int src_addr; /* Source Address */
    int des_addr; /* Destination Address */
    int len; /* Message Length */
    int arr_time; /* Arrival time */
    int sent; /* Status of the Message PENDING or SENT */
    int delay; /* Message Delay (Sending time - Arrival time) */
} MESSAGE;
```

```
typedef struct link
{
    struct message *msg;
    struct link *next;
} LINK;
```

```

typedef struct node_info
{
    int ring_addr; /* Ring Address */
    int prev_node; /* Prev Station Addr */
    int next_node;
    int mac_addr;
    int tok_hold_time;
    int tok_arr_time;
    int tok_rel_time;
    int last_tok_rel_time;
    int idle_time; /*Idle time of the nod */
    int waiting_ack; /*Waiting for Acknowledgement
                    from the Next Node */
    int has_token; /* If node has token */
    struct link *msg_head; /* Message head in the node */
    struct link *next_msg; /* Next Message to process */

}NODE_INFO;

struct token_ring
{
    NODE_INFO station;
    struct token_ring *next;
};

typedef struct token_ring NODE;

//Global Variables
NODE *head_node;
NODE *tail_node;
FILE *fpout;
long tick;
/*Total Token Holding time by all the station*/
long total_THT;

/* Function Declarations */
void create_message(MESSAGE *tmpmsg,int node_id);
void addMsg(LINK **head,LINK *temp);
void create_station(LINK *msg_head, int mac_addr,int ring_addr);
void prepare_tokenring();
void simulate_tokenring();

//Function used as utility

```

```
int some_node_msg_pending(NODE *start_node);
void calculate_params(NODE *curr_node);
void send_msg(NODE *curr_node);
int isMsgPending(NODE *curr_node);

//Add-on funtions for the Token Ring Simulation
int get_random(int range);
void print_node_info();

//Check latter
NODE* traverse_station(int n);
```

FileName: msg_list.c

```
#include "wtrp_main.h"
```

```
/*
 * This function adds the message to the message list in
 * Ascending order of the message Arrival Time.
 */
void addMsg(LINK **head, LINK *temp)
{
    int n=0;
    LINK *current;
    current=*head ;

    if(current== NULL){
        *head = temp;
    }
    else{
        while(current!=NULL){
            if((temp->msg->arr_time<(*head)->msg->arr_time)){
                temp->next=*head;
                *head=temp;
            }

            else if((current->next==NULL) &&
                (temp->msg->arr_time<current->msg->arr_time)){
                temp->next=current->next;
                current->next=temp;
            }

            else if((temp->msg->arr_time>current->msg->arr_time)&&
                (temp->msg->arr_time<current->next->msg->arr_time)){
                temp->next=current->next;
                current->next=temp;
            }

            current=current->next;
        }
    }
}
```

```
void create_message(MESSAGE *tmpmsg,int node_id)
{
    int temp_des =0;

    tmpmsg->arr_time = get_random(MAX_ARR_TIME);
    tmpmsg->src_addr = node_id;
    tmpmsg->delay = 0;
    tmpmsg->sent = FALSE;

    do{
        temp_des = get_random(NUM_NODES);
    }while(temp_des == node_id);

    tmpmsg->des_addr = temp_des;
    tmpmsg->len = get_random(MAX_MSG_LEN);
}
```

FileName: circular.c

```
#include "wtrp_main.h"
```

```
void create_station(LINK *msg_head, int mac_addr, int ring_addr)
```

```
{  
    NODE *temp;  
  
    temp =(NODE*)malloc(sizeof(NODE));  
  
    temp->station.has_token = FALSE;  
    temp->station.idle_time =0;  
    temp->station.mac_addr = mac_addr;  
    temp->station.msg_head = msg_head;  
    temp->station.next_node = mac_addr +1 ;  
    temp->station.prev_node = mac_addr -1;  
    temp->station.ring_addr = ring_addr;  
    temp->station.tok_arr_time = 0;  
    temp->station.tok_hold_time =0;  
    temp->station.tok_rel_time =0;  
    temp->station.last_tok_rel_time =0;  
    temp->station.waiting_ack = FALSE;  
    temp->station.next_msg = msg_head; /*Initial msg to process*/  
  
    if(head_node ==NULL)  
    {  
        tail_node = head_node = temp;  
        temp->next = temp;  
    }  
    else  
    {  
        tail_node->next = temp;  
        temp->next = head_node;  
        tail_node = temp;  
    }  
}
```


VITA



Sivakumar Chinnathambi Janakiraman

Candidate for Degree of

Master of Science

Thesis: OUT OF BAND MESSAGE PASSING IN WIRELESS TOKEN RING
NETWORKS

Major Field: Computer Science

Biographical:

Education: Graduated from Santhom Higher Secondary School, Chennai, Tamilnadu, India in April 1994. Received Bachelor of Engineering in Mechanical Engineering from the University of Madras, Chennai, Tamilnadu, India in April 1998.

Completed the requirements for the Master of Science degree with a Major in Computer Science at Oklahoma State University, Stillwater, Oklahoma in December 2003.

Experience: Employed as Graduate Assistant, Department of Geography, Oklahoma State University, March 2000 to August 2002.