DISTRIBUTED FLOOR CONTROL PROTOCOLS FOR SUPPORTING PRIORITIES IN COLLABORATIVE APPLICATIONS

By

SURYAPRAKASH SINGIREDDY

Bachelor of Science in Computer Science and Information Technology

Jawaharlal Nehru Technological University

Hyderabad, India

2003

Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
MASTER OF SCIENCE
July, 2006

DISTRIBUTED FLOOR CONTROL PROTOCOLS

FOR SUPPORTING PRIORITIES IN

COLLABORATIVE APPLICATIONS

Thesis Approved:

Dr.Venkatesh Sarangan
Thesis Advisor

Dr. John Chandler
Committee Member

Dr. H. K. Dai
Committee Member

A. Gordon Emslie
Dean of the Graduate College

ACKNOWLEDGEMENTS

I would like to express my sincere thanks to my advisor Dr. Venkatesh Sarangan for his guidance and motivation that he has provided me throughout my thesis work. I would also like to express my sincere appreciation to Dr. John Chandler and Dr. H. K. Dai for their suggestions and support.

I would like to thank my parents, my brother and my sister for their love, support and guidance that they have given me all the time. Finally I would also thank all my friends for their continuous support.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Collaborative applications are software applications that allow a group of users to cooperate with each other when geographically dispersed. Collaborative applications are becoming efficient means for group communication. The increase in number of broadband networks elevates the demand for this kind of applications. Collaborative applications could play vital role for communication, which could range from a computer user to corporations, military, and government. The very useful applications of collaborative applications would be video-conferencing, collaborative real-time editing, computer controlled priority right auctioning, collaborative simulation, and multi-player online games.

In collaborative applications it is critical that at any specific time of a collaborative session only a single user should gain the access to the communication network. In order to provide a user with exclusive access to the communication channel, floor control protocols are used [1]. The user who ever wins the floor can have the right to access the communication channel, otherwise have to wait until it gets free again. The user who has won the floor can multicast the messages to others in the group [2]. The floor control protocols and multicasting protocols when implemented at the network layer bring in major drawbacks such as scalability, network layer modification, changes in

router functions, and network management [2]. So to remove all these complications at the network layer, researchers are implementing them at the application layer.

As the floor control protocols and multicast protocols reside in the application layer, an end host has the responsibility of transferring messages to other end hosts in the collaborative session. So networking responsibilities are deployed to end hosts, they use the concept of virtual networks known as overlay networks. In the peer to peer based overlay networks, multicasting has to be handled by the end hosts. The peer to peer based architectures introduces limitation on scalability as the end-host finds it difficult to handle large group sizes. Because of this increasing burden on end hosts [6] has proposed to introduce Network Service Nodes (also known as multicast service nodes), which will communicate with end hosts and with each other using unicast paths. This kind of architecture is known as proxy based overlay networks. This approach will reduce the overall burden on the end-hosts.

The floor control protocols can be categorized into two groups, centralized and distributed floor control protocols. In the centralized floor control protocols the decision of allocating floor is taken by a centralized controller. End host which needs to send message to other end hosts has to request the centralized controller. Centralized controller processes the requests, gives the permission to the end host to access the channel according to the scheduling algorithm. In centralized floor control protocols, it becomes easier to construct algorithms to implement priorities, to maintain causal ordering of messages and to dynamically add additional end-hosts. But main drawback of centralized approach is when large number of requests that come to centralized controller it becomes tough task to process all those requests and also if centralized controller fails it will bring

down whole collaborative session. In distributed floor control protocols, the decision of allocating floor is not taken by any single node. Management of floor becomes difficult in distributed floor control protocols as every end host have to be aware of the current situation of the network and it has to know when end host has the permission to access the channel. Implementing priorities in allocating floor to an end user in distributed floor control protocols is complicated.

Priority in distributed floor control protocols makes user with higher priority to gain floor more number of times than lower priority users. Inclusion of priority in distributed floor control protocols is useful in many applications such as computer controlled priority bidding and multiplayer online games. For example, a group of people in a company using computer supported collaborative application. Obviously the users contain few high priority users and low priority users. In order to give high priority users more access to the application it will be useful to use priority based distributed floor control protocols.

The goal of this paper was firstly to evaluate the efficiency of distributed floor control protocols such as ALOHA and DQDB under different environments. Secondly to implement priorities in the ALOHA and DQDB based distributed floor control protocols. Thirdly, to implement CSMA MAC protocol for floor control and test the performance of this protocol with the other two protocols. Fourthly, to implement priority in the CSMA distributed floor control protocol, and finally to preserve causal ordering of the messages even after implementing the priority, as it should be taken care that a node sending message first has to be received first.

CHAPTER 2

BACKGROUND

2.1 Floor Control

Floor control is playing an important role in collaborating applications; it acts as a synchronizing primitive among the group of users in the collaborative session. Floor control works on similar principle to that of semaphore. Semaphore gives permission for only a single process to access critical section at any time and eliminates simultaneous access of critical section by two or more processes. In the same way floor control gives permission to only a single user at a time to access the communication channel in a collaborative session.

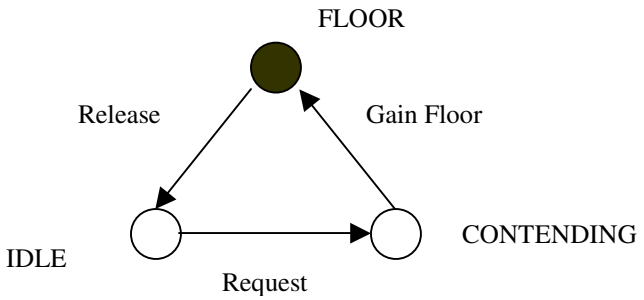

Figure 1. State Diagram of Floor Control Protocol

Figure 1 shows the finite state diagram of floor control protocol. Every user in the collaborative session will be in one of the three states. In Idle state user just receives the multicast messages sent by other users. In Contending state user has message to send to

other users, so user is contending to gain floor in order to access the network. In Floor state, user has gained access to the network, it can now multicast messages over the network to other users in the collaborative session.

2.2 Causal Ordering

Causal ordering of messages means to receive messages in the order they were sent by the stations. Causal ordering of messages is one of the important criteria that need to be protected in a multicast network. In most occasions the meaning of the whole situation changes if causal ordering of the messages is not preserved.

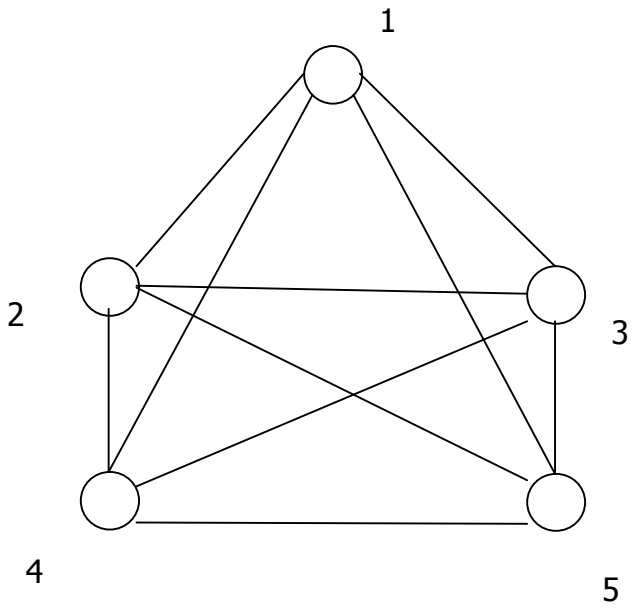

Figure 2: Causal Ordering

For example, in figure 2 stations 1, 2, 3, 4 and 5 are participating in a collaborative session. If station 1 and station 2 multicast messages to other stations in that order, then there will be a chance that message from station 1 is received after message

from station 2, which would change the whole scenario of collaboration. In order to preserve causal ordering of messages end-to-end delays in the communication should be considered.

2.3 Overlay Networks

In overlay networks the participating hosts share the responsibility of forwarding packets to all other hosts in the session. There are two kinds of overlay networks identified till now they are peer to peer based overlay networks and proxy based overlay networks. In peer-to-peer based overlay networks end hosts share the responsibility of network functionality. Because end-hosts are handling all the tasks, this kind of architecture does not scale well for large scale networks. In proxy-based overlay networks all the functionality is moved to Network Service Nodes (NSN). For Efficient group communication it is required to construct a Multicast tree such as a Steiner tree. In proxy-based overlay networks, the NSN's communicate with all the NSN's and end-hosts attached to it. As it forms a complete graph consisting of NSN's, it will become easier to construct a spanning tree from network of NSN's. Also this kind of overlay networks reduces the burden on the end hosts.

2.4 DQDB

DQDB is a MAC Layer protocol works on distributed queuing algorithm. DQDB consists of two unidirectional buses which can transfer packets in opposite directions. Each NSN is connected to both the unidirectional buses. One bus is used to transmit data to the nodes on the left and other bus is used to transmit data to the nodes on the right.

Every node can transmit data to the nodes in the downstream. For both the buses, one end consists of slot cell generator and the other end consists of sink. These end nodes act as master stations which have responsibility of generating slot cells and absorbing them finally when they reach to the end. Every NSN contains a queue and maintains two variables request counter (RQ) and countdown counter (DQ).

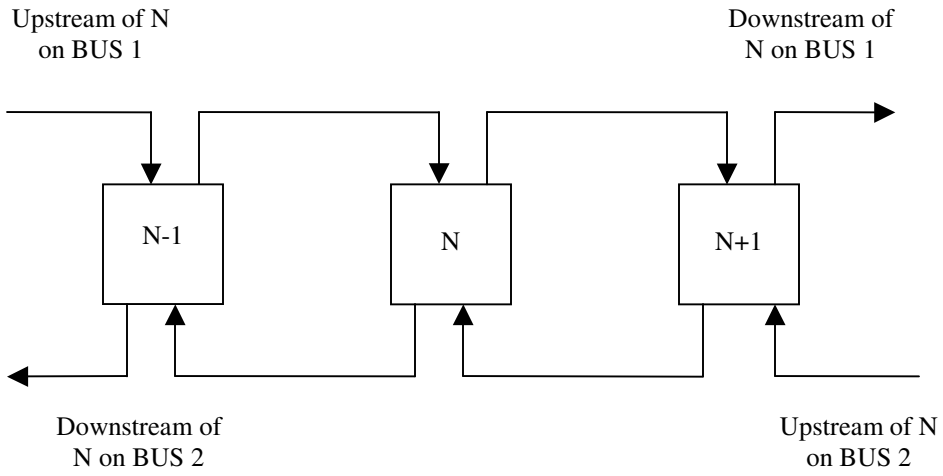

Figure 3: shows a fragment of DQDB network from DQDB.ppt

The distributed queue maintained in the every NSN allows them to access the bus in an orderly manner. RQ indicates the number of NSN's ahead of the current NSN when it is inactive i.e. not wanting to transmit data. Every inactive NSN counts the requests passing on BUS 1 and decrements the RQ value when it sees an empty slot on BUS 2 and vice versa. When a NSN wants to transmit data it waits for an available request bit slot, it then sets the request bit indicating that it needs to transmit data. The NSN copies the RQ value into the DQ and resets the RQ value to zero. DQ indicates the number of NSN's ahead of the current NSN when it is waiting to transmit data. Every active NSN decrements the

DQ value it finds an available slot passing on the Bus. When the DQ value becomes zero, indicates that NSN has reached the top of the queue and it can transmit the data. When transmitting data NSN acquires slot, sets the header, copies the data into the slot, the cells propagate to the end of the bus.

2.5 ALOHA

ALOHA protocol is considered to be a random access, contention type and distributed protocol. The idea is that whenever a station has packets to transmit then it transmits them on the common transmission channel with out checking the condition of the transmission channel. The station after transmitting packet waits for Round Trip Time (RTT) to receive an acknowledgment. If it does not receive an acknowledgment within that time then it retransmits the packet after waiting a random amount of time. It repeats this process few number of times and if it still failed to send the packet then it gives up. If the packet is received by the receiver then it checks the FCS field and then returns the acknowledgment packet back to the sender. The ALOHA protocol depends on the ability of a station to detect the collisions. The ALOHA protocol gives better utilization if the traffic within the network is very low, as the number of collisions in the network will be less thereby infrequent number of retransmissions. The concept of the ALOHA protocol is simple and can be applied to other situations with little modification.

2.6 CSMA

Collision Sense Multiple Access protocol works on the concept of sensing the channel before attempting to transmit data. By sensing the channel if a station finds it idle

then the station can send the frames otherwise it waits for certain amount of time and again senses the channel. It does this until it realizes that channel is idle, and then it transmits frames. But even after sensing the channel to be idle, collisions may still occur, since two or more stations can sense the channel idle at same time. When collision occurs, stations have to wait for whole frame transmission time in order to send frames again. The CSMA protocol has better utilization when compared with other Random MAC protocols.

2.7 Participating Chain

In overlay networks, the efficiency of communication channels is directly proportional to the maximum end-to-end delay between all pairs of end hosts. In proxy-based overlay networks, as all the end-hosts are connected to the NSN's, it is required to construct an efficient spanning sub-network for coordinating floor. Constructing participating chain is useful in quickly resolving the contention for floor in contention phase and useful for maintaining causal ordering of the data packets in the data delivery phase. The NSN's connected to end host participating in the collaborative session are called as participating NSN's. A participating chain is defined as a path that connects all the participating NSN's. An optimal participating chain is a participating chain where the maximum end-to-end delay is least.

CHAPTER 3

LITERATURE REVIEW

Many researchers have developed several centralized protocols for floor control. In these protocols, centralized controllers are used to resolve control of the floor among the contending stations. Few centralized approaches are discussed below.

Reference [4] proposed a conference control protocol for a highly interactive video conferencing. The protocol uses three-channel rotation scheme for floor control to avoid race conditions and to resolve the contention for shared media channels among the participants. Each conference participant can acquire any of the three roles: current speaker, previous speaker and listener. In order to get the role of speaker, a participant needs to send a request signal to the centralized controller. The controller and the participants send the control information on the control channel. The participant can turn into a speaker if the centralized controller acknowledges the request. The centralized controller elects a speaker depending on the access policy. The main drawback of this protocol is the use of centralized controller.

Reference [1] presents a tree-based floor control protocol referred as the Hierarchical Group Coordination protocol (HGCP). It uses a logical control tree derived from acknowledgment tree used in the reliable multicast tree. In order to reduce traffic at the source, the control tree uses cascaded acknowledgments and negative acknowledgments which prevent the receivers to directly communicate with the source.

The control tree can shrink and expand dynamically during session lifetime. In HGCP the roles of floor handler (FH) and floor controller (FC) are unified to a single node in order to remove triangle communication between the nodes if separated to three different nodes. The root of the control tree is FH and tree rotates when the FH changes, with root shifting to the new FH. A node can act as FH if it acquires the shared resource. There are three kinds of nodes in the control tree: control node, relay nodes and leaf nodes. Every node in the control tree communicates only with the neighboring nodes. Every node maintains local picture of its neighborhood, to know the direction towards the root of the tree (FH). There are three phases in the HGCP: setup phase, active phase and termination phase. In the setup phase, protocol allows new nodes to join the session. Each station maintains a floor state table and updates them regularly from its neighbors to be consistent. In the active phase of the protocol, station sends request control directive (CD) to its neighbor close to the FH. The direction towards the FH is computed by taking the prefix of the location of the floor in the floor state table. Floor can be assigned to a node based on priority, queuing, timestamp, reception order, frequency and holding time of floor. FH passes the GRANT message to its successor based on source label information. Upon confirmation, FH relinquishes the floor by multicasting the position of new FH to the session. From then all the nodes send their requests to the new FH. In the termination phase, floor state information is deleted from records of the station. Any node in the session can withdraw from the session and appropriate action is taken. The main drawbacks of HGCP protocol firstly, its partiality to nodes nearer to the root. Secondly, it has drawbacks of centralized approaches as it is based on dynamically centralized

approach (i.e. controller for floor handler is dynamically selected). Thirdly, requires lot of communication to coordinate floor.

Centralized protocols are simple to implement but they will have several problems like:

Controller Failure: Controller failure brings down entire collaborative session.

Traffic: As every request needs to be handled by the controller, it gets congested.

Scalability: The centralized protocols do not scale well for large group sizes as the controller gets congested.

Due to deficiencies in the centralized protocols, researchers have diverted towards distributed protocols. However the concept of distributed approach is relatively new in this field. Some of the existing distributed floor control techniques are summarized below.

The work done in this paper is inspired from the Reference [2], this paper presented distributed floor control protocols using MAC layer protocols such as ALOHA and Distributed Queue Dual Bus (DQDB) on overlay networks. The complete graph of NSN's is used to construct an efficient communication channel. The communication channel will ensure quick floor acquisition by end-host in contention phase and causal ordering of data packet in data delivery phase. The communication channel also known as participating chain is constructed using a polynomial time algorithm. The end-to-end delay will be small in the optimal participating chain. The properties of optimal participating chain are

- An optimal participating chain of tree network will start at node of degree 1 and end at a node of degree 1.
- An optimal participating chain of tree network will not traverse the links in the longest path of tree more than once.
- The length of participating chain does not change if sub-trees not having longest path are visited in different order.

The algorithm for constructing participating chain is as follows:

Notations:

$T(r)$: Tree with root r.

$ST(c)$: Subtree of T rooted at c.

Ordering Symbol: *{<=, >=}*

Order($T(r)$, [ordering symbol]): Arrange k children of r in T such that $l(c_i)$ [ordering symbol] $l(c_{i+1})$ and c_i is a child of r for $1 <= i <= k$.

Algorithm find chain *PC* **()**

Input: $T(r)$

Output: chain *PC*

Begin

Compute $l(x)$ for each non-leaf node x in T.

Sort the children of r in non-decreasing order of their l values.

Let v and u be the first and second child of r in the sorted order.

Rearrange T such that u and v are the first and last child of r

For each non-leaf node i in $ST(u)$

Order($ST(i)$,>=).

End For

For each non-leaf node l in theST (v)

Order($ST(l)$,<=).

End For

PC = In-order traversal on *T*

End

Protocol End-host_floor_acquisition()

Begin

If end-host wants to acquire floor

Generate *floor_request* message

Send *floor_request* message to NSN

End If

If end-host receives *floor_grant* message from NSN

End-host goes to *Data Delivery Phase*

Else If end-host receives *floor_reject* message from NSN

End-host waits for random amount of time

End-host_floor_acquisition()

End If

End

ALOHA: If a end-host needs to send a message to other end-hosts then sends floor request message to the NSN with which it is attached. NSN then forwards this message to all other NSN's. The requested NSN waits for 2C time (where C is the length of participating chain). If an NSN receives a floor request message from other NSN's within this time period then it understands that a collision has occurred, the requesting end host then will wait for random amount of time. If NSN does not receive any floor request message from other NSN's then it means that no other end host has requested for floor. Now this end-host can sends data to its NSN. This NSN forwards the data to last NSN in the participating, which then forwards it to root of the multicasting tree. The root then sends the message to all the participating end-hosts. This algorithm maintains the causal ordering of the messages.

DQDB: Using the NSN's participating in the collaboration a participating chain is constructed. Participating chain consists of two unidirectional logical paths request flow path and signal flow path. The first NSN in the request flow path is known as tail NSN and the first NSN in the signal flow path is known as head NSN. When an end-host needs to send message to other end-hosts then it sends floor request signal to its NSN. The requesting NSN copies the RQ value into DQ and resets the RQ value to zero. The NSN then forwards the request to the left neighbor NSN. The left neighboring NSN increments its RQ value and similarly forwards it to its left neighbor until it reaches leftmost NSN. The head NSN generates wake up signal which is forwarded using the signal flow path. Until the first encountered NSN which has DQ value of zero, it traverses on the signal flow path and decrements the DQ value of the NSN's encountered. The NSN which has DQ value of zero checks whether any of the end-hosts attached to it requested for floor. Then it sends a request grant signal to the requesting end host. The end-host sends messages to its NSN with its ID. The NSN forwards it to the tail NSN. The tail NSN forwards it to root of the multicasting tree. The root checks whether the sequence number is the smallest among the received, if it is the smallest then it multicasts the messages to all the participating end-hosts. Otherwise buffers them until smallest is received. This algorithm also maintains the causal ordering of the messages.

Reference [3] presents a distributed floor control protocol known as Activity Sensing Floor Control protocol that works on the concept of sensing activity on every shared resource. The monitoring mechanism in the ASFC is similar to that of 802.3 Ethernet protocol CSMA/CD. The shared resource is said to be unoccupied if any user monitoring the resource do not observe any data packet for a certain threshold time

known as maximum inter-departure time. If the time between the two data packets exceeds this maximum inter-departure time then also resource is assumed to be unoccupied. When a user senses that a resource is occupied then it contends for resource. If the floor attempt is unsuccessful then it backs off for certain amount of time depending on the number of unsuccessful attempts. The main drawback of this protocol is that it assumes that one way transit delay experienced by any packet is fixed. The protocol also does not make sure that data packets are received in the order they were sent to the end users.

Reference [5] proposed a protocol for distributed internet conferencing using the technique known as full mesh conferencing. In this technique, each member is directly connected to other members in the conference. Without any central point of control protocol allows any number of users to participate in the conference. All the members in the conference are given equal preference without any topological partiality. The protocol also allows a member to join and leave the conference at any time, and intimates all members in the conference about the new members. The protocol uses various messages in order to establish communication between users, to maintain the conference network and to join and leave a member from the conference. As this protocol uses direct communication between members, it is well suited for small scale conferences. The message complexity increases exponentially, so this protocol is not suitable for medium and large scale conferences.

CHAPTER 4

PROPOSED APPROACH

Implementing priority based distributed floor control protocols was the main aim of this paper. The paper presents an approach in implementing distributed floor control protocol using CSMA protocol. The implementation of priority in distributed floor control protocols using ALOHA, DQDB and CSMA are explained.

4.1 ALOHA

The ALOHA protocol works on the idea of transmitting data when it has data to send. Every station if it needs to send data then it does so. If there is collision then it retransmits for waiting certain period of time. Reference [2] implemented the ALOHA based distributed floor control protocol. However the ALOHA MAC protocol implemented in the Reference [2] do not consider the case of priority. The issue in implementing priority based ALOHA protocol for floor control on overlay networks is same as that of the ALOHA protocol i.e. detecting the collisions. But upon collisions the waiting time of the NSN differs from the ALOHA protocol however it is proportional to the length of the participating chain. The waiting time in ALOHA is random and not proportional to the priority of the end nodes.

Contending Phase:

A participating NSN maintains the information about its neighboring NSN's in the collaborative session. From this information the participating NSN identifies the NSN that is to the left side of it and NSN that is to the right side of it. If an end-host attached to the NSN needs to send data to other NSN's then it sends a request message to its NSN. The NSN forwards the request message to its left and right neighboring NSN's with first available slot. The slot time increases from high priority to low priority. The NSN after sending request waits for '2C' amount of time period. A NSN when it receives a floor request message from its left neighbor NSN it forwards the request message to the right neighbor NSN and vice versa. Within this time period if the current NSN receives any request message from its neighboring NSN's then it means that a collision has occurred otherwise no collision occurred. If there was no collision, it means that no other NSN is contending for the floor at that moment. The NSN sends floor_grant signal to the requesting end-host. The end-host enters into the data delivery phase. If a collision has occurred, it means other NSN's are also contending in order to gain the floor. The NSN sends floor_reject signal to end-host when a collision occurs. The end host after receiving floor_reject signal waits for random amount of time depending on the length of the participating chain. The end_host with high priority will have small slot-time. This differentiation between high priority and low priority end-hosts will give more opportunity for high priority end-hosts to gain the floor. The waiting time period for NSN is fixed at '2C' because the worst case when the first NSN in the participating chain is competing for the floor against last NSN in the participating chain, as it will take 'C' amount of time to transfer a message from one end to the other end. Waiting for '2C'

time period will determine the worst-case possible collision between first and last NSN in the participating chain. The priority based Aloha protocol for distributed floor control is as follows:

NSN_PRIORITY_ALOHA_on_*PC*
Begin
If NSN receives *floor_request* from its end-host
floor_control(*floor_request*)
End If
If NSN receives *floor_request* from *leftneigh bor*
NSN forwards *floor_request* to *right neighbor*
End If
If NSN receives *floor_request* from *right neighbor*
NSN forwards the *floor_request* to *leftn eighbor*
End If
End

floor_control(*floor request*)
Begin
Wait for Slot_Time
Send *floor_request* to *left neighbor* and *right neighbor*
Set *timer* = 2*C*
While *timer !*= 0
If *floor_request* is received from *leftneigh bor* OR *right neighbor*
Send *floor_reject* to end-host
break
End If
timer- -
End While
Send *floor_grant* to end-host

End

ENDHOST_PRIORITY_ALOHA

Begin

If end-host receives *floor_grant* from its NSN

Start sending data to NSN

End If

If end-host receives *floor_reject* from its NSN

wait(priority)

End If

End

Data Delivery Phase:

The end-host after gaining the floor in contention phase is now ready to transmit data to all other participating end-hosts. The data delivery phase uses participating chain and optimal multicasting tree to transfer data. Participating chain is used to ensure causal ordering of messages and multicasting tree is used to reduce end-to-end delay. The end-host transfers the data packets to its NSN, NSN then it forwards it last NSN in the chain. The last NSN forwards the packets to the root of the multicasting tree. The root forwards the packets to all the end-hosts in the session. The NSN's while forwarding data packets do not transfer the packets to its attached participating end-hosts. The working of ALOHA protocol in maintaining the causal ordering of messages can be proved by considering the worst case scenario. Let us consider the case where the first NSN(F) and last NSN(L) would like to transfer packets in sequence. The NSN F first sends floor request message to all other NSN's. It then waits for '2C' amount of time period to receive floor requests from other NSN's. Assuming that no collisions occur in the floor requests. After waiting for '2C' amount of time the NSN F sends grant message to the

requesting end-host. The end-host will send data to NSN F and NSN F forwards the data to the NSN L and NSN L forwards it to the root of multicasting tree. The total time taken for NSN F to send data can be estimated as '2C+C+d' where 2C is waiting time to gain floor, 'C' is time taken transfer data from F to L and 'd' is time taken for L to transfer to the root of the multicasting tree. If NSN L would like transfer floor request then it has to send request message after C amount of time other wise collision will occur. The NSN L will wait for 2C amount of time period in order to gain floor. Assuming no collisions the NSN L sends grant message to its requesting end-host. The end-host will send data to NSN L and NSN L forwards the data to the root of the multicasting tree. The total time taken for NSN L to transfer data can be estimated as 'x+2C+d' where $x > C$ it is necessary condition to avoid collision, 2C is waiting time to gain floor, and 'd' is the time taken for L to transfer data to the root of the multicasting tree. So it is clear that in the worst case situation also the protocol is maintaining the causal ordering of messages.

4.2 DQDB

In the Distributed Queue Dual Bus (DQDB) protocol, every station maintains the global information of the order of preference by regularly updating the distributed queue. Each station is connected to two parallel buses and is aware of the relative position of all other stations. Although it is possible to implement priority in DQDB, but only a maximum of four priority levels can be implemented. Also, in order to implement priority in DQDB protocol, every station has to handle few additional overheads. Every station has to maintain separate queues for each priority class. To differentiate each priority level, different request bits are set in the access control field. Separate request

counter (RQ) and countdown counter (DQ) are maintained for each queue. The high priority queues only count the high priority requests. The low priority queues count all the requests of equal or higher priority. However in the DQDB protocol for floor control, every station needs to send data only to the last station (also called as tail_NSN) and needs to send request signal towards first station (also called as head_NSN) so unidirectional paths are used.

Contending Phase:

In DQDB, the participating chain consists of two unidirectional logical paths maintained among the NSN's: request_flow and signal_flow. As shown in the figure below the request_flow path is E-D-C-B-A and signal flow path is A-B-C-D-E. The first NSN in request_flow path is known as tail_NSN and last NSN in request_flow path is known as head_NSN. The request_flow path carries requests in the direction of tail_NSN to head_NSN. The signal_flow path carries signals in the opposite direction.

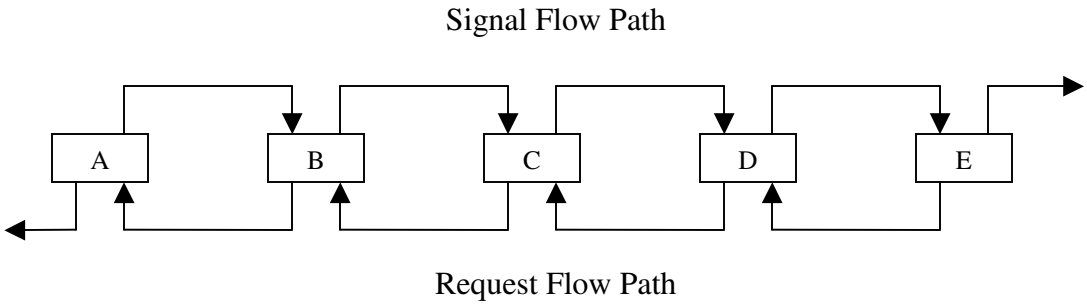

Figure 4 representing participating chain with request_flow path and signal_flow path from [2]

As in Aloha, each NSN in participating chain maintains information about its left neighbor and right neighbor in each path. Every NSN maintains separate queues for every priority level. The priority of messages can be known from the bits in the Access Control Field. Each NSN maintains request counter (RQ) and countdown counter (DQ) variables for each distributed queue i.e. if there are 3 priority levels then each NSN needs to maintain request counter (RQ) and countdown counter (DQ) variables in each of the 3 queues, each value indicating different priority level. Request counter (RQ) indicates number of requests forwarded by the upstream NSN's in the request_flow path ahead of the NSN. In each NSN the different request counter (RQ) values represent the number of requests of that priority forwarded by the upstream NSN's. Countdown counter (DQ) represents the position of NSN's request in the distributed queue. If an end-host wants to transmit data to other participants in the collaborative session then it has to obtain the floor. In order to gain the floor, the end-host sends floor_request signal to its NSN. The NSN forwards the floor_request signal to its left neighboring NSN in the request_flow path and copies its current request counters (RQ) values into the countdown counters (DQ) respectively. The countdown counters (DQ) now contain the number of NSN's ahead of this NSN in the queues. The request counters (RQ) are reset to zero. Each NSN after receiving floor_request signal in the request_flow path increments their RQ values and forwards the floor_request signal to its left neighbor in the request_flow path. If the receiving NSN sees a request of higher priority, it increments the request counters (RQ) of equal and lower priority queues and if it sees lower priority request then it only increment the request counter (RQ) of lower priority queue. Furthermore, the countdown counters (DQ) of the receiving NSN has to be increased if it sees a request message of

higher priority. Through this approach all the downstream NSN's in the request_flow path will come to know that the requesting NSN wants to acquire floor. The head_NSN generates wake-up signal at regular intervals. The wake-up signal flows through all the NSN's in the downstream along the signal_flow path and reaches tail_NSN. Each wake-up signal generated contains unique sequence number and it is incremented for every signal. The head_NSN after generating wake-up signal checks its countdown counters (DQ) in the order of priority. If its DQ value is 0, the head_NSN checks whether it has any requesting end-host with that priority. The head_NSN accepts the wake-up signal in case it has requesting end-host, it sends floor_grant signal to the requesting end-host. The head_NSN sets the end-host ID in the wake-up signal and forwards it to the right neighboring NSN in the signal_flow path. If the DQ value is greater than 0, it decrements its value by 1. Every NSN receiving the wake-up signal checks whether wake-up signal already contains any ID. If it contains ID, it just forwards the wake-up signal to the right neighbor in the signal flow path indicating that it is already occupied. If the wake-up signal is free, the NSN checks its DQ value in the order of priority and sets its requesting end-host ID in the wake-up signal if DQ value is 0, otherwise it just decrements its DQ values and forwards the wake-up signal. The end-host after receiving floor_grant signal enters into data delivery phase. The priority based DQDB protocol for distributed floor control is as follows:

NSN_PRIORITY_DQDB_on_*PC*()

Begin

Set *flag* ← FALSE

If NSN receives $floor\_request_p$ from its end-host

Forward $floor\_request_p$ to *left neighbor* in request_flow path

While i <= Number of Queues in the NSN

$DQ_i \leftarrow RQ_i$

$RQ_i \leftarrow 0$

End While

End If

If $floor\_request_p$ is received in request_flow path

While i <= $priority_p$

$RQ_i \leftarrow RQ_i + 1$

$DQ_i \leftarrow DQ_i + 1$

End While

Forward $floor\_request_p$ to *left neighbor* in request_flow path

End If

If *wake_up* is received in signal_flow path

If *wake_up* does not contain end-host ID

While i <= Number of Queues in the NSN

If $DQ_i \leftarrow 0$

If requesting end-host with priority exists

Send *floor_grant* to requesting end-host

Put end-host ID in the *wake_up* signal

Forward the *wake_up* signal to *right neighbor* in signal_flow path

flag ←FALSE

End If

End If

End While

If *flag* ← TRUE

While i <= Number of Queues in the NSN

If DQ_i *!= 0*

$DQ_i \leftarrow DQ_i - 1$

Forward *wake_up* to *right neighbor* in signal_flow path

End If

End While

End If

Else

Forward *wake_up* to *right neighbor* in signal_flow path

End If

End If

End

Data delivery Phase:

The tail_NSN after receiving the wake-up signal forwards it to root of the multicasting tree. The root checks if the wake-up signal is consumed (contains end-host ID in wake-up signal). If the wake-up signal is utilized then root stores both the sequence number and end-host ID in the database. In the data delivery phase, the end-host sends data along with ID to its NSN. The NSN forwards the data to its right neighbor along the signal_flow path. The receiving NSN forwards data until it reaches the tail_NSN. The tail_NSN forwards the data to root of the multicasting tree. The root checks the ID of the end-host with the ID in the data base for correctness and checks if the sequence number received is smallest. Otherwise it temporarily buffers the data until it receives data with smallest sequence number. This will ensure that data is transferred in the order of floor acquisition which in turn guarantees causal ordering of data.

4.3 CSMA

The Carrier Sense Multiple Access (CSMA) protocol has mentioned in the previous section works on the idea of “listen before talk”. Every station before transmitting the data, need to make sure whether the channel is idle. If the channel is idle

then the station can transmit the packets other wise has to wait for certain amount of time before scheduling for retransmission. Implementing CSMA protocol for floor control on overlay networks need to handle certain issues. In LANs it is easier to verify whether channel is idle, but in floor control protocols the NSN's might be spread across larger area. The NSN's also need to detect the collisions of the request signals. If a NSN receives a request signal then the channel is said to be occupied for '2C' where C is maximum end-to-end delay which is proportional to the length of participating chain. If a NSN receives request message when it is in waiting period then it certifies that a collision has occurred. The efficiency of the solution entirely depends on the waiting time which is proportional to the length of the participation chain. An optimal participation chain will generate excellent results.

Contending Phase:

A participating NSN maintains the information about its neighboring NSN's in the collaborative session. From this information the participating NSN identifies the NSN that is to the left side of it and NSN that is to the right side of it. Every NSN keeps track of the requests sent by other NSN's in the session. Every NSN gets a chance to send request depending on the priority of the end_host. If an end-host attached to the NSN wants to acquire floor in order to send data to other NSN's then it sends a request message to its NSN. The NSN checks whether it has received any request from other NSN's, if it has received a request, it checks if it has received within 2C amount of time period. If it has received a request within time period then the NSN will wait for 2C amount of time indicating that the bus is busy. Otherwise it forwards the request message

to its left and right neighboring NSN's. The NSN waits for '2C' amount of time period. A NSN when it receives a floor request message from its left neighbor NSN it updates its latest request time and forwards the request message to the right neighbor NSN and vice versa. Within this time period if the requesting NSN receives any request message from its neighboring NSN's then it means that a collision has occurred otherwise no collision occurred. If there was no collision, it means that no other NSN is contending for the floor at that moment. The NSN sends floor_grant signal to the requesting end-host. The end-host enters into the data delivery phase. If a collision has occurred, it means other NSN's are also contending in order to gain the floor. The NSN sends floor_reject signal to end-host when a collision occurs. The end-host enters into back-off mode. The priority of an end-host comes into action when an end-host receives the floor_reject signal i.e. when there is contention to gain floor. The end-host with higher priority back-offs for less time compared to the end-host with lower priority. This differentiation between high priority and low priority end-hosts will give more opportunity for high priority end-hosts to gain the floor. The waiting time period for NSN is fixed at '2C' because the worst case when the first NSN in the participating chain is competing for the floor against last NSN in the participating chain, as it will take 'C' amount of time to transfer a message from one end to the other end. Waiting for '2C' time period will determine the worst-case possible collision between first and last NSN in the participating chain. The priority based CSMA protocol for distributed floor control is as follows:

NSN_PRIORITY_CSMA_on_*PC*

Begin

If NSN receives *floor_request* from its end-host

If NSN has received as floor_request from other NSN within 2C time period

While *current_time-latest_request_time* < 2C

Wait()

End While

floor_control(*floor_request*)

End If

If NSN receives *floor_request* from *leftneigh bor*

Update *latest_request_time*

NSN forwards *floor_request* to *right neighbor*

End If

If NSN receives *floor_request* from *right neighbor*

Update *latest_request_time*

NSN forwards the *floor_request* to *leftn eighbor*

End If

End

floor_control(*floor request*)

Begin

Send *floor_request* to *left neighbor* and *right neighbor*

Set *timer* = 2*C*

While *timer !*= 0

If *floor_request* is received from *leftneigh bor* OR *right neighbor*

Send *floor_reject* to end-host

Break

End If

timer- -

End While

Send *floor_grant* to end-host

End

ENDHOST_PRIORITY_CSMA

Begin

If end-host receives *floor_grant* from its NSN

Start sending data to NSN

End If

If end-host receives *floor_reject* from its NSN

wait(priority)

End If

End

Data Delivery Phase:

The data delivery phase in CSMA protocol is similar to that of Aloha protocol. The end-host after gaining the floor in contention phase is now ready to transmit data to other participating end-hosts. The data delivery phase uses participating chain and optimal multicasting tree to transfer data. Participating chain is used ensure causal ordering of messages and multicasting tree is used to reduce end-to-end delay. The end-host transfers the data packets to its NSN, NSN then it forwards it to last NSN in the chain. The last NSN forwards the packets to the root of the multicasting tree. The root forwards the packets to all the end-hosts in the session. The NSN's while forwarding data packets do not transfer the packets to its attached participating end-hosts. The working of CSMA protocol in maintaining the causal ordering of messages can be proved by considering the worst case scenario. Let us consider the case where the first NSN (F) and last NSN (L) would like to transfer packets in sequence. The NSN F first sends floor request message to all other NSN's. It then waits for '2C' amount of time period to receive floor requests from other NSN's. Assuming that no collisions occur in the floor requests. After waiting for '2C' amount of time the NSN F sends grant message to the

requesting end-host. The end-host will send data to NSN F and NSN F forwards the data to the NSN L and NSN L forwards it to the root of multicasting tree. The total time taken for NSN F to send data can be estimated as '2C+C+d' where 2C is waiting time to gain floor, 'C' is time taken transfer data from F to L and 'd' is time taken for L to transfer to the root of the multicasting tree. If NSN L would like transfer floor request then it has to send request message after C amount of time other wise collision will occur. The NSN L will wait for 2C amount of time period in order to gain floor. Assuming no collisions the NSN L sends grant message to its requesting end-host. The end-host will send data to NSN L and NSN L forwards the data to the root of the multicasting tree. The total time taken for NSN L to transfer data can be estimated as 'x+2C+d' where $x > C$ it is necessary condition to avoid collision, 2C is waiting time to gain floor, and 'd' is the time taken for L to transfer data to the root of the multicasting tree. So it is clear that in the worst case situation also the CSMA protocol is maintaining the causal ordering of messages.

CHAPTER 5

SIMULATION

The protocols were developed to evaluate the behavior of priorities in distributed floor control. The protocols were simulated using ns-2.27. The simulations of the protocols were done extensively for different conditions. The various input parameters that are used in the simulations are:

Simulation Time: Every simulation was made to run for 2000 seconds.

End Nodes: Every NSN is connected to 2 end nodes.

Link Delay: The delay between one NSN to other NSN is fixed at 50 milliseconds and the delay between the NSN and its end node is 30milliseconds.

Bandwidth: The bandwidth of the links is 1 Mbps.

Request Packet Size: The request packets are 25 bytes.

Contention Window: The contention window in ALOHA is 20.

Back off Time: The back off time in CSMA is varied according to the priority of end node. The back off time in ALOHA is random.

Seed: All the three distributed floor control protocols use the random functions to generate the requests. A different set of results are obtained for a particular request rate and length of participating by varying the seed value.

Cell Generation Rate: The slot and request cells which are required in DQDB are varied proportional to the size of the participating chain. The cell generation rate is kept constant for a fixed participating chain.

Cell Size: The cells are 44 bytes.

*Participating Chain: T*he simulations were done by varying the length of the participating chain for every protocol. The length of the participating chain is varied from 5 NSN's to 20 NSN's. By varying the length of the participating chain, the behavior of ALOHA, CSMA and DQDB protocols under large and small group collaborations can be evaluated.

Request Rate: The request packet generation rate at end nodes is varied on whole network as well as varied depending on the priority. For every length of participating chain, simulations were done by varying the request rates. The request rates are varied from 0.0125req/sec to 0.05req/sec. By varying the request rate, the performance of all the three protocols under heavy loads can be tested.

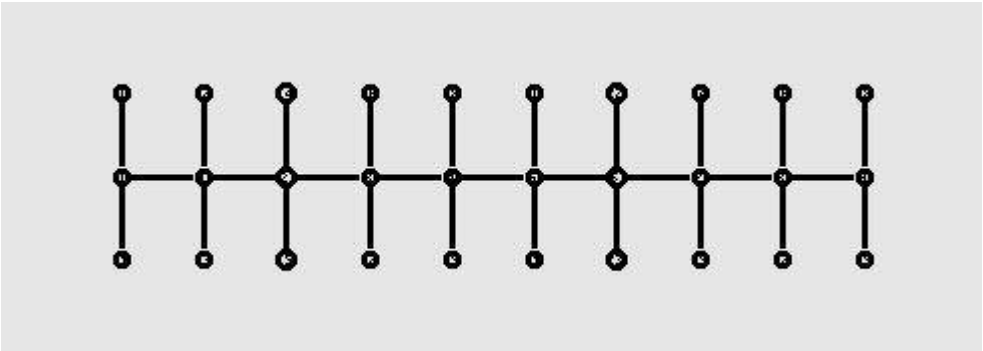

Figure 5: Shows the alignment of NSNs and End Nodes. The length of participating chain is 15 NSNs

In priority based simulations, the number of end nodes were taken proportional to the priority. The number of end nodes taken in simulations are presented in the following table:

Length of Participating Chain	Priority 0	Priority 1	Priority 2	Priority 3
40	6	6	11	17
30	4	4	9	13
20	3	3	6	8
10	1	2	3	4

The simulations were done for various request rates, seeds and lengths of participating chain. The average delay occurred for a request to succeed is calculated for all the protocols.

CHAPTER 6

RESULTS

The simulations were performed both on priority based and non-priority based distributed floor control protocols. The average delay taken by the requests are calculated and the results of the simulations are presented in the graphs.

The graph in figure 6 shows the performance of CSMA and ALOHA protocols without priorities, when requests are generated at 0.05req/sec at every end node.

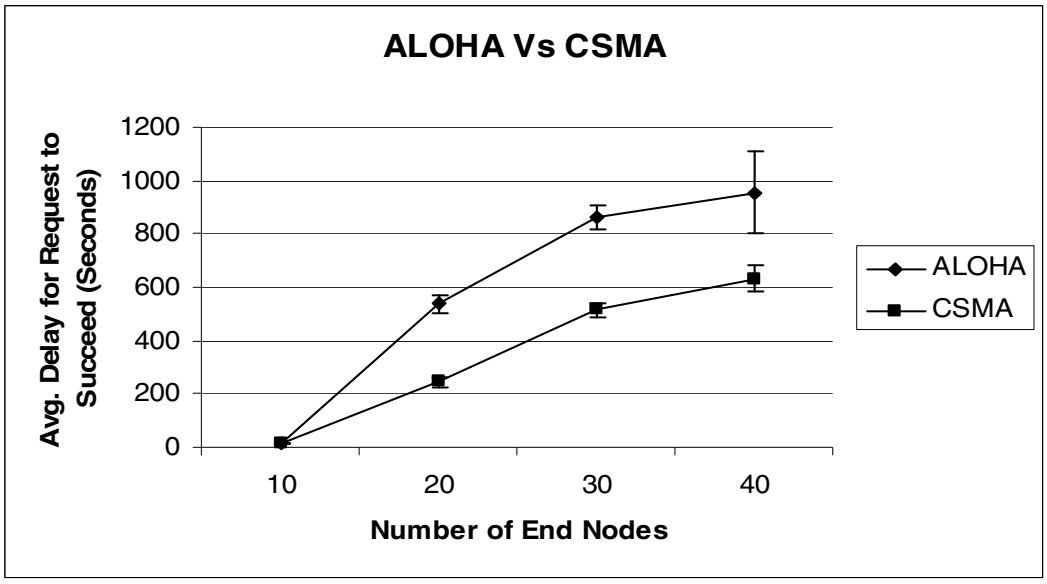

Figure 6: shows the average delay for CSMA and ALOHA protocols

From the results, it can be observed that CSMA protocol performs better than ALOHA protocol. This is because ALOHA protocol transmits the requests immediately whereas the CSMA protocol transmits the packets only when channel is predicted as idle.

As a result the number of collisions in ALOHA increases more rapidly than CSMA. But when the network is small, then both the protocols consume same amount of time for a request to succeed. Since the increase in the average delay in CSMA is less, the CSMA protocol can be used for large group communications.

The graph in figure 7 shows the average delay for request to succeed verses number of end nodes for ALOHA protocol, when requests are generated at 0.025req/sec at every end node.

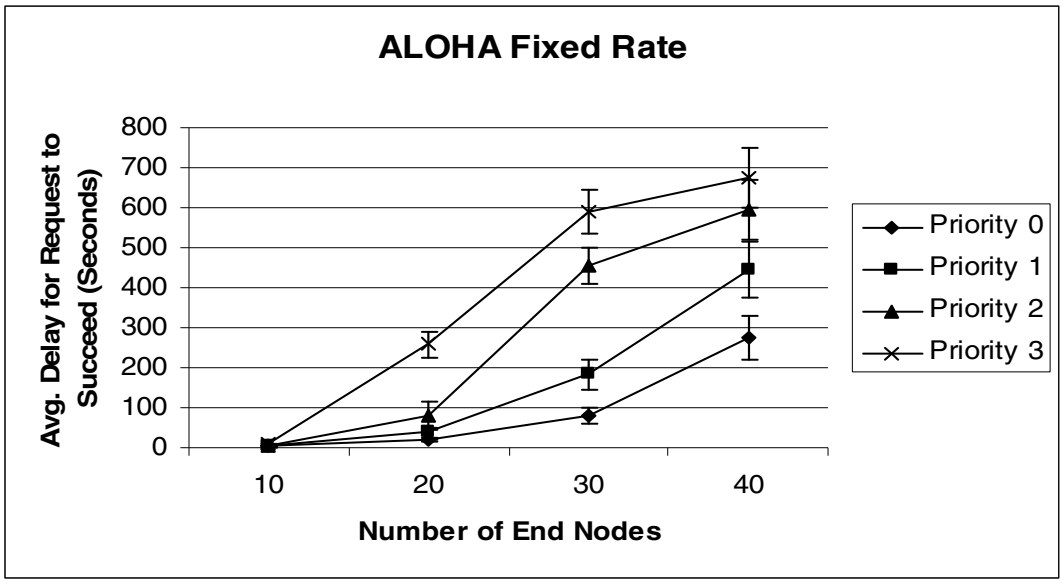

Figure 7: shows average delay for ALOHA protocol

In this simulation all the priorities have the same request rate. We can observe that the delay for priority '0' requests is less compared to the delay taken by other priority requests. This is because the priority '0' packets wait for smaller slot time than all other priorities. When the number of end nodes are less, then the difference in time consumed for each priority request is not clear. But as the number of end nodes increase, the difference in time taken by request is clearly visible.

The graph in figure 8 shows the average delay for request to succeed verses number of end nodes for ALOHA protocol, when requests are generated according to the priority of end nodes.

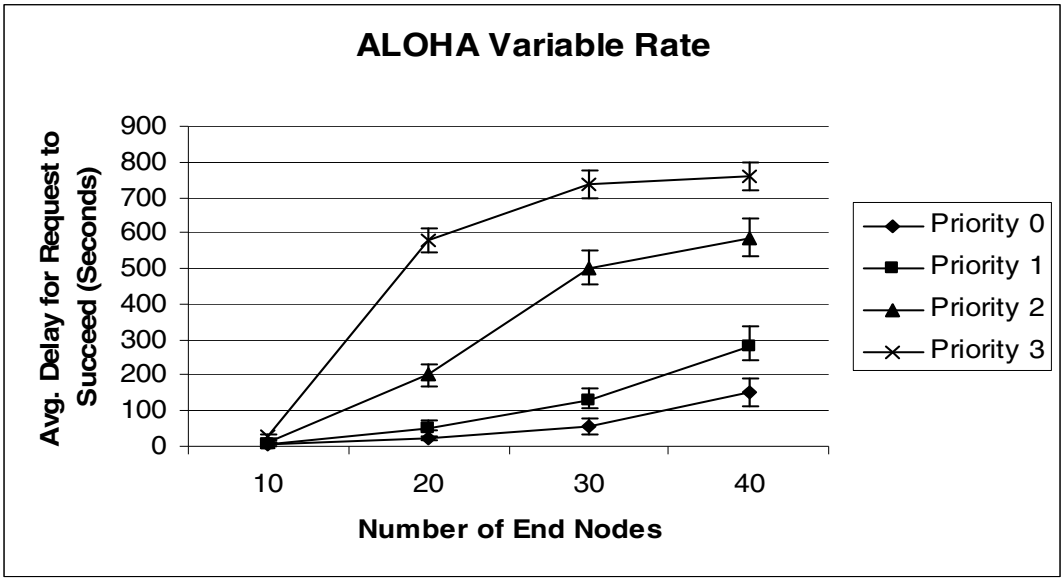

Figure 8: shows average delay for ALOHA protocol

We observe from the graph that the time taken by the priority '0' requests is less compared to other priorities. As the size of the network increase the average time taken for request to succeed increases. We can also observe that there is sharp increase in delays for lower priority requests when the number of nodes increases. Both simulations of ALOHA protocol behave similarly even though the request rates are changed.

The graph in figure 9 shows the average delay for request to succeed verses number of end nodes for CSMA protocol, when requests are generated at 0.025req/sec.

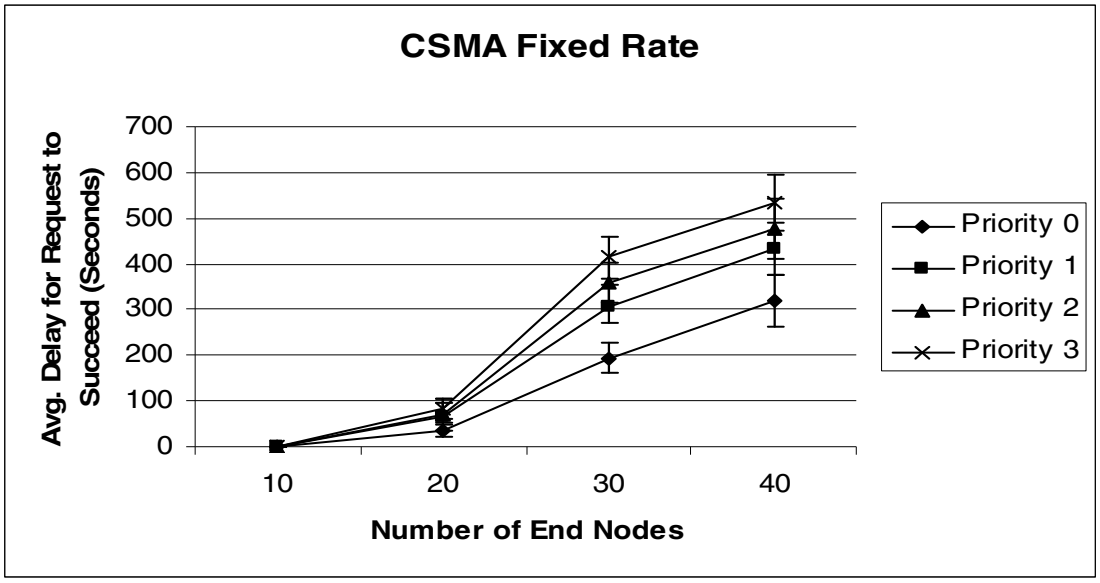

Figure 9 shows the average delay CSMA protocol.

The graph in figure 10 shows the performance of CSMA protocol when the requests are generated according to the priority of end nodes.

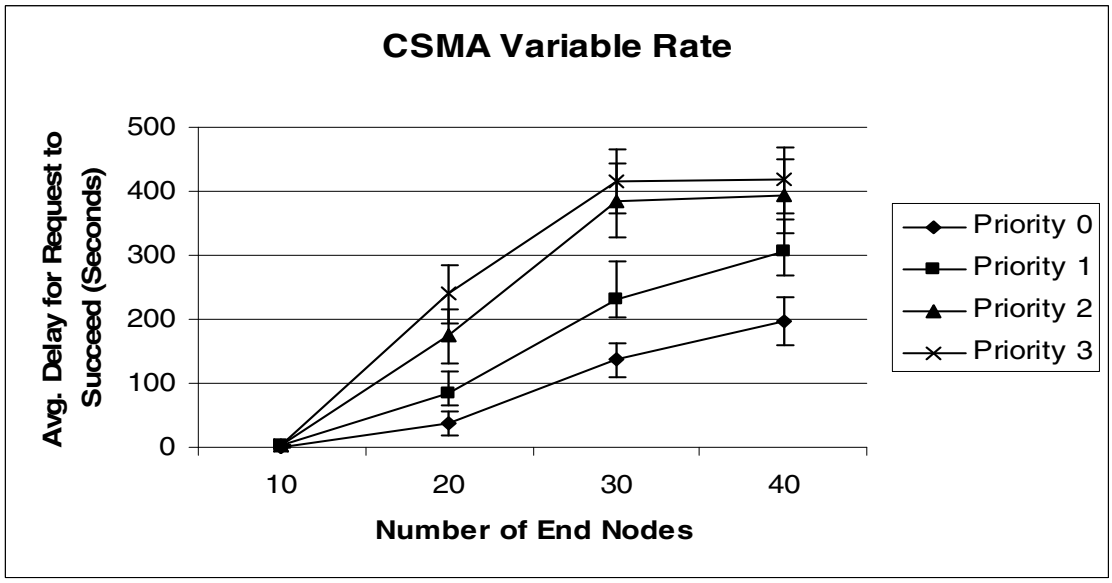

Figure 10 shows the average delay CSMA protocol.

Both CSMA simulations behave similar to that of ALOHA. The average time taken by the higher priority requests is less compared to that of other priorities.

The graph in figure 11 shows the average delay for request to succeed verses number of end nodes for DQDB protocol, when the requests are generated at 0.025req/sec at every end node.

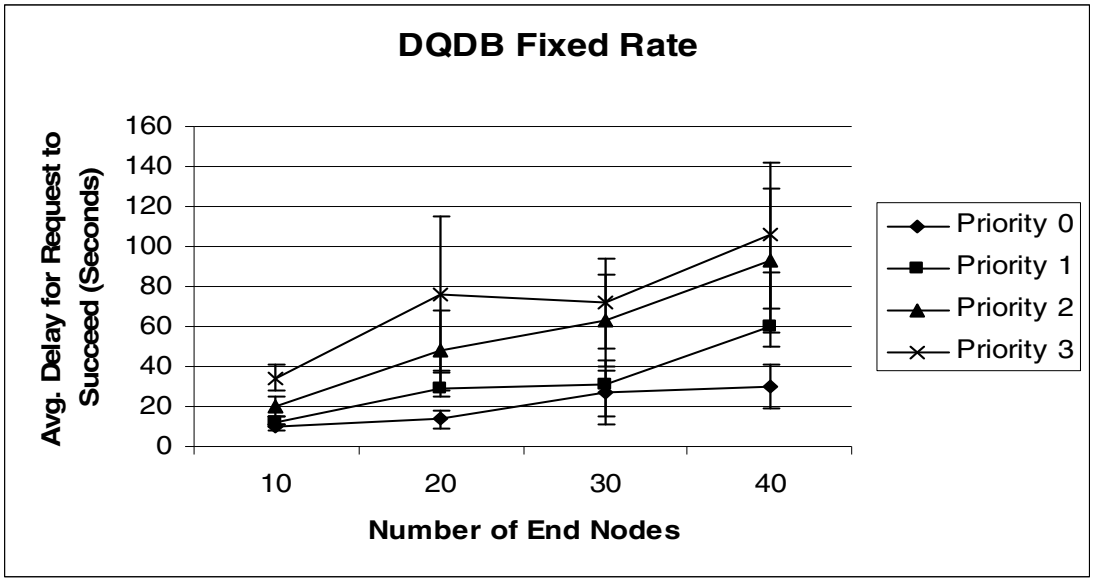

Figure 11 shows the average delay for DQDB protocol.

We observe from figures 11 and 12 that the time taken by higher order priority is less compared to lower order priorities.

In all the three protocols, average delay for higher priority request to succeed is less compared to lower priority requests. However, the average time taken by the DQDB request is very less compared to that of CSMA and ALOHA. This is because in DQDB there are no retransmissions and no collisions. The DQDB protocol performs better than other two protocols in terms of average delay for request but the only problem in DQDB is that head NSN has to constantly generate slot cells even when end nodes has no data to transmit.

The graph in figure 12 shows the average delay for request to succeed verses number of end nodes for DQDB protocol when the requests are generated based on the priority of the end node.

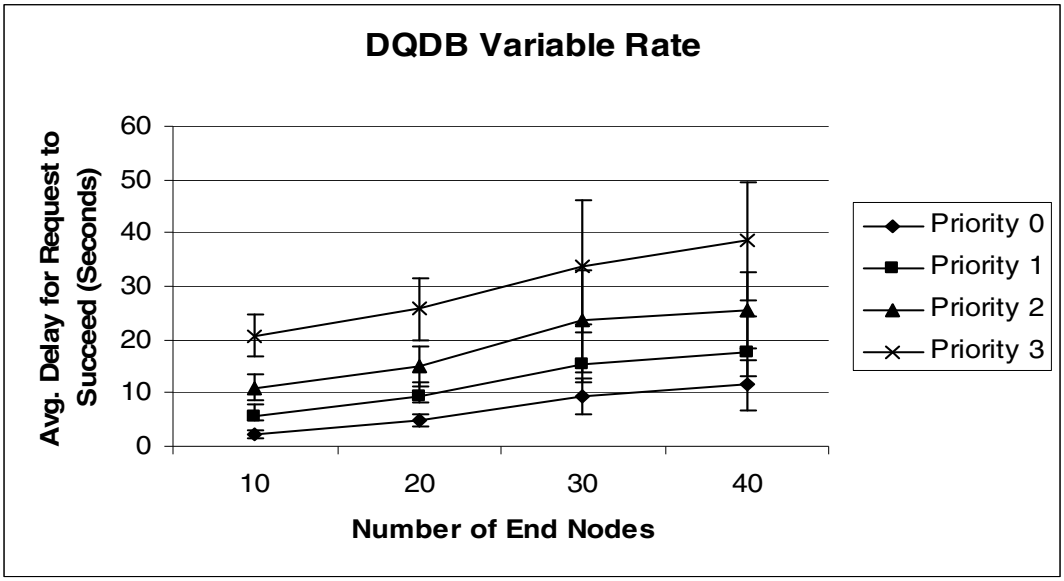

Figure 12. Average delay for DQDB protocol

CHAPTER 7

CONCLUSION AND FUTURE WORK

The proposed CSMA protocol for distributed floor control was implemented and the results showed that the protocol performs more efficiently than ALOHA protocol. Also we proposed to implement priorities in CSMA, ALOHA and DQDB protocols. The graphs in previous chapter showed that the protocols serve higher priority requests more often than lower priority requests. Among the three distributed floor control protocols DQDB performs efficiently than other two protocols. We have also shown that all the three distributed floor control protocols preserve the causal ordering of messages even with priorities. The results thus justify the use of priority based distributed floor control protocols in order to serve priority end nodes efficiently.

The distributed floor control protocols implemented so far are still in the elementary stage. The protocols need to be developed in order to handle the dynamism of the networks. The protocols proposed in this paper introduce priority in the collaborative session. Following extensions can be implemented to handle the dynamism of the collaborative session.

The protocols currently handle a fixed number of priorities, it can be further expanded to handle a number of priorities. Variable number of priorities can be easily implemented in CSMA and ALOHA but it is complicated to implement it in DQDB. Furthermore priority for an end node can be allowed to change dynamically

during the simulation. Protocols can be expanded to allow end nodes to join and leave a collaborative session dynamically. By allowing this feature, users get flexibility to join and leave a collaborative session at any moment. Adding the above mentioned functionalities to distributed floor control protocols will make collaborative applications more user friendly.

REFERENCES

[1] H.-Peter. Dommel, J.J. Garcia-Luna-Aceves, "Efficacy of Floor Control Protocols in Distributed Multimedia Environment," in *Cluster Computing Journal, Special Issue on Multimedia Collaborative Environments*, Vol 2, No 1,1999.

[2] S. M. Banik, S. Radhakrishnan, Tao Zheng, C. N. Sekharan, "Distributed Floor Control Protocols for Computer Collaborative Applications on Overlay Networks", in IEEE CollaborateCom, San Jose, December 2005.

[3] K. Katrinis, G. Parissidis, B. Plattner, "Activity Sensing Floor Control in Multimedia Collaborative Applications," in *10th International Conference on Distributed Multimedia Systems*, San Francisco Bay, September, 2004.

[4] R. Qiu, F. Kuhns, J. R. Cox, "A Conference Control Protcol for Highly Interactive Video-conferencing," in *IEEE Global Telecommunications Conference*, November 2002, pp. 2021-2025.

[5] J. Lennox, H. Schulzrinne, "A Protocol for Reliable Decentralized Conferencing" in *13th International Workshop in Network and Operating Systems,* Monterey, California, June 2003.

[6] Sherlia Y. Shi, Jonathan. S. Turner,"Routing in Overlay Networks," in *IEEE INFOCOMM 2002*,June 2002, pp. 1200-1208.

[7] A. S. Tanenbaum, *Computer Networks*, 3rd edition, Prentice Hall, Upper Saddle River, NJ, 1996.

[8] Distributed Queue Dual Bus protocol, www.cse.dmu.ac.uk/~amp/**ppt**/Dqdb.**ppt** Last accessed: 28 Feb 2006

VITA

SuryaPrakash Singireddy

Candidate for the Degree of

Master of Science

Thesis: DISTRIBUTED FLOOR CONTROL PROTOCOLS FOR SUPPORTING PRIORITIES IN COLLABORATIVE APPLICATIONS

Major Field: Computer Science

Biographical:

Personal Data: Born in Hyderabad, India, on May 22nd 1982, the son of Mr. Kista Reddy Singireddy and Mrs. Sujatha Singireddy.

Education: Received my Bachelors Degree in Computer Science and Information Technology from JNTU, May 2003. Completed the requirements for Master of Science degree at Oklahoma State University, July 2006.

Experience: Graduate Research Assistant, Sept 2004 – May 2006.

Name: SuryaPrakash Singireddy Date of Degree: July, 2006

Institution: Oklahoma State University Location: Stillwater, Oklahoma

Title of Study: DISTRIBUTED FLOOR CONTROL PROTOCOLS FOR SUPPORTING PRIORITIES IN COLLABORATIVE APPLICATIONS

Pages in Study: 43 Candidate for the Degree of Master of Science

Major Field: Computer Science

Scope and Method of Study:

Collaborative applications are becoming popular with the availability of broadband wide area networks. There are several collaborative applications such as computer controlled priority right auctioning, multiplayer online video games, collaborative editing and collaborative simulations that are used for cost and time efficient communication. Important issues in distributed floor controls are to provide exclusive access to communication channel for every single user at any time and to maintain causal ordering of messages. In many situations it is also necessary to have priorities in group collaboration to give higher access to users who deserve it, but the distributed floor control protocols implemented till now do not provide priority based group collaborations. This paper presents an implementation of CSMA MAC protocol to solve floor control problem and to implement priority based distributed floor control protocols using ALOHA, DQDB and CSMA MAC protocols on overlay networks.

Findings and Conclusions:

I compared the efficiencies of ALOHA, DQDB and CSMA MAC protocols with and without priority using network simulator-2. The results shown that higher priority requests are served more often than other requests. We have also shown that these three distributed floor control protocols preserve the causal ordering of messages even with priorities. Among the three MAC protocols, DQDB protocol performs efficiently for distributed floor control with or without priority. However, in the DQDB protocol slot cells are generated even when there are no requests to be sent by NSN's. This is an additional burden on network even if end nodes has small amount of data to be transferred. In conclusion we can say that collaborative applications with distributed floor control can be prioritized giving higher access to the end users with high priority.

ADVISER'S APPROVAL: Dr. Venkatesh Sarangan