Distributed floor control protocols for computer collaborative applications on overlay networks

Conference Paper · January 2005

DOI: 10.1109/COLCOM.2005.1651240 · Source: dx.doi.org

CITATIONS
8

READS
27

4 authors, including:

Sridhar Radhakrishnan
University of Oklahoma
163 PUBLICATIONS **1,637** CITATIONS

SEE PROFILE

Distributed Floor Control Protocols for Computer Collaborative Applications on Overlay Networks

Shankar M Banik, Sridhar Radhakrishnan, Tao Zheng
School of Computer Science
University of Oklahoma

Chandra N Sekharan
Department of Computer Science
Loyola University at Chicago

Overview

- Distributed Floor Control: What and Why?
- Overlay Networks: What and Why?
- Efficient Communication Channel
- Proposed Implementation of MAC protocols on Overlay Networks for Floor Control
- Causal Ordering of messages
- Analytical Model and Simulation
- Conclusion

Floor Control: What and Why?

- Providing exclusive access to communication network for a distributed application
- Three step process
  1) Obtain access to the floor
  2) Send data
  3) Relinquish access
- Mechanisms to gain exclusive access to the floor can be coordinated or distributed.
  - Explicit: wait for reserved-for-you message
  - Implicit: did not see any other request message for a period of time. That is no collisions.
- Sending data must follow ordering such as causal.
- Relinquishing access to the floor can be explicit or implicit based on whether special message is sent or not.

Floor Control: What and Why?

Examples of applications that can use this floor control

- Video Conferencing
- Collaborative Design and Simulation
- Online Games

Floor Control Mechanisms

- Centralized Floor Control
  - Each end-user sends a request to a controller
  - The controller grants floor to the end-users using some policy
  - Advantage
    - Simple and easy to implement
  - Disadvantage
    - Links to the controller will be congested
    - Failure of the controller will result in the failure of the total application
- Distributed Floor Control
  - Similar in spirit to the medium access control protocols.

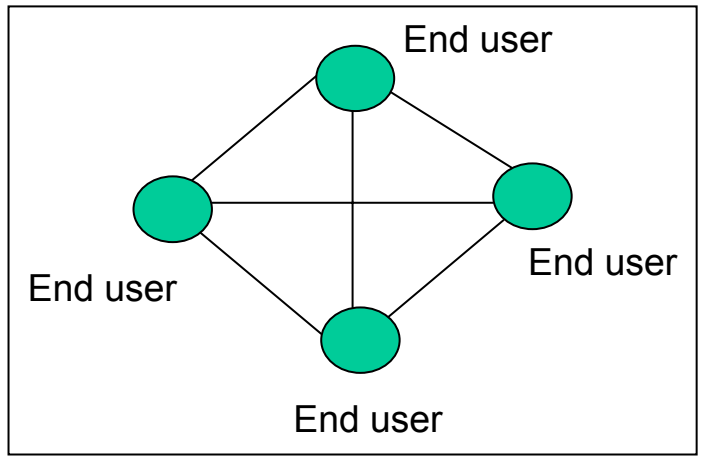
Overlay Network: What and Why?

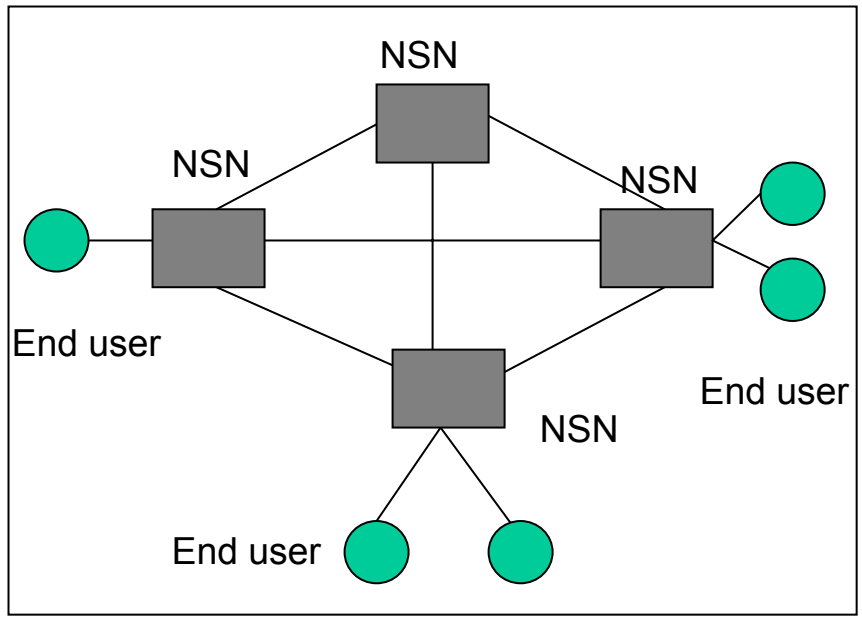
- Distributed applications require “designer networks” customized for the application.
- The applications require changes to the network protocols to allow for application dependent QoS requirements to be implemented.
- Impossible to change the underlying Internet to satisfy application requirements due to:
  - Changing Infrastructure
  - Scalability
  - Deployment
- Recent research trend
  - Develop and implement network layer protocol in the application layer
  - Application programs at end-users form a virtual network called *overlay network*

Architectures of Overlay Networks

- Peer-to-peer
  - Network functionality is pushed to the end-users
- Proxy-based
  - Removes additional burden from the end-users

We have considered proxy-based overlay in this paper





State-of-the-art of existing Floor Control Protocols

- Dommel, Garcia [Cluster Computing Journal 1999]
  - First one to introduce and analyze floor control protocols
  - Proposed a tree-based protocol using a logical control tree
  - Dynamically centralized
- Qiu et. al. [GlobeCom 2002]
  - Proposed a three channel rotation scheme
  - Interactive and scalable
  - Centralized controller
- Katrinis et. al. [Distributed Multimedia 2004]
  - Proposed an activity sensing protocol
  - Assumes that maximum inter-departure time between two consecutive data packets transmitted by a floor holder is fixed and globally known

Our Objectives

- To develop fully distributed floor control protocols
- To consider the delay of the underlying communication channel at the time of designing a floor control protocol
- To guarantee causal ordering of messages at the participating end-users

Causal Ordering: "If a message m is sent before m', then to all destinations common to m and m', m will be delivered before m'.

Distributed Floor Control

- Proposed to implement MAC protocols on Overlay Networks for floor control
- ALOHA
  - Randomized
  - Simple
- DQDB
  - Scheduled based
  - Maintains FIFO ordering

Efficient Communication Channel

- To implement MAC protocols on distributed networks:
  - Design an efficient communication channel among NSNs
    - Maximum end-to-end delay between all pairs of NSN is minimum
  - Develop techniques for collision detection in the case of randomized protocols.
  - Design data delivery mechanisms after the floor has been acquired
  - Develop techniques to relinquish the floor for scheduled protocols.

- Communication in any floor control application in the overlay network can be divided in three phases
  - Contention Phase
    - An end-user sends the floor request signal to its NSN
    - The NSN communicates with other NSNs to resolve the contention for floor acquisition and sends either a grant or reject signal to the requesting end-user
  - Data Delivery Phase
    - The end-user which was granted the floor by its NSN, sends data to all other participating end-users through its NSN
  - Floor release Phase
    - The end-user that gained access releases the floor.

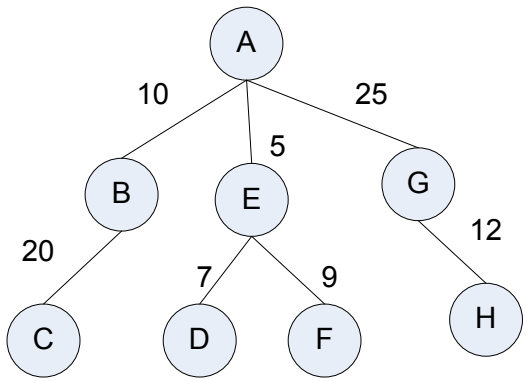
Construction of communication channel

- Participating NSN
  - If there exist at least one participating end-user attached to it
- Participating Chain
  - A communication channel that connects the set of participating NSNs
  - The length of a participating chain is defined as the summation of delay of all the links that belong to the participating chain
  - An ***Optimal Participating*** chain is a participating chain with minimum length

We develop an algorithm to obtain an optimal participating chain for a tree network

Properties of an Optimal Participating Chain of a Tree Network

- An optimal participating chain of a tree network will start at a node of degree 1 and end at a node of degree 1
- An optimal participating chain of a tree network will not traverse the links in the longest path of the tree more than once
- Different order of visiting the sub-trees of children (of the root) which are not contained in the longest path does not change the length of the optimal participating chain



Optimal Participating Chain:

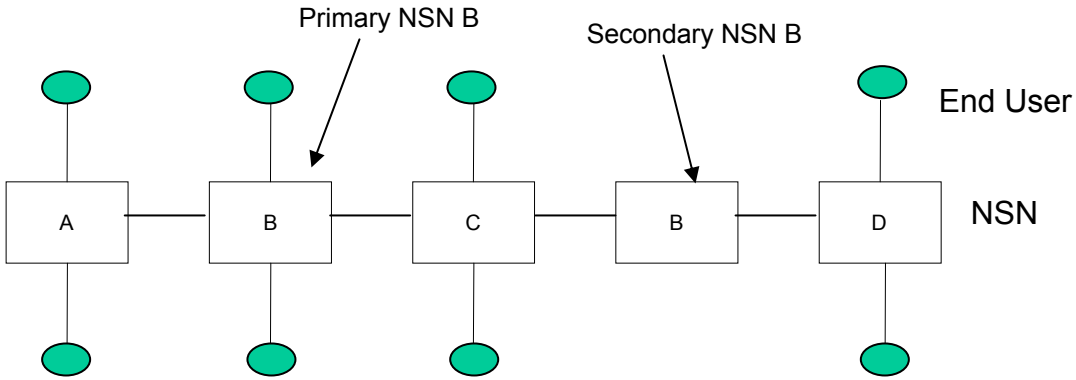
C – B – A – E – D – E – F – E – A – G – H

Length : 109

Constructing the Optimal Participating Chain of a Tree Network

- Main steps of finding the Optimal Participating chain of a tree T
  - Determine the root of T such that the root is the center of the tree (say r)
  - Let u and v be the two children of r which are contained in the longest path of T
    - Let $\{x_1, x_2, \ldots, x_k\}$ be the set of leaf nodes of the subtree rooted at x in T
    - Let $l(x, x_i)$ be the path length from node x to the leaf node x_i
    - Let $l(x) = \max \{ l(x, x_i), 1 \le i \le k\}$
  - Order the children of r in non-decreasing of their l values
    - First and second node in the sorted order will be v and u respectively
  - Rearrange the children of r such that u and v are the leftmost and rightmost child respectively
  - Rearrange the children of u (resp. v) in decreasing (resp. increasing) order of their l values
  - Perform an in order traversal of the rearranged tree

Example of a Participating Chain



- Assumptions:
  - Each NSN knows its left neighbor and right neighbor
  - Each NSN also knows the length of the chain (say *C*)
  - In a participating chain an NSN can appear more than once. In this situation, one of them will work as primary NSN and others will work as secondary NSNs. We assign the first NSN as primary NSN and the rest of them as secondary ones. A secondary NSN is not attached to any end-users, it just forwards the control signals

Protocol for Floor Acquisition at the End-user

- If an end-user wants to acquire a floor, it generates a *floor_request* message and sends the message to its NSN
- In Randomized Protocol
  - The NSN of the requesting end-user after getting the *floor_request* message tries to detect whether the request is collided with other *floor_request* messages and either sends *floor_grant* and *floor_reject* message to its requesting end-user depending on the outcome of collision detection mechanism
- In Scheduled Protocol
  - The NSN of the requesting end-user waits for its turn as in a token passing mechanism and sends *floor_grant* message to its end-user when it gets its turn
- If the end-user receives *floor_rejection* message, it waits for random amount of time and sends the *floor_request* again to its NSN
- If the end-user receives *floor_grant* signal, it goes to data delivery phase and sends data to all other participating end-user through its NSN

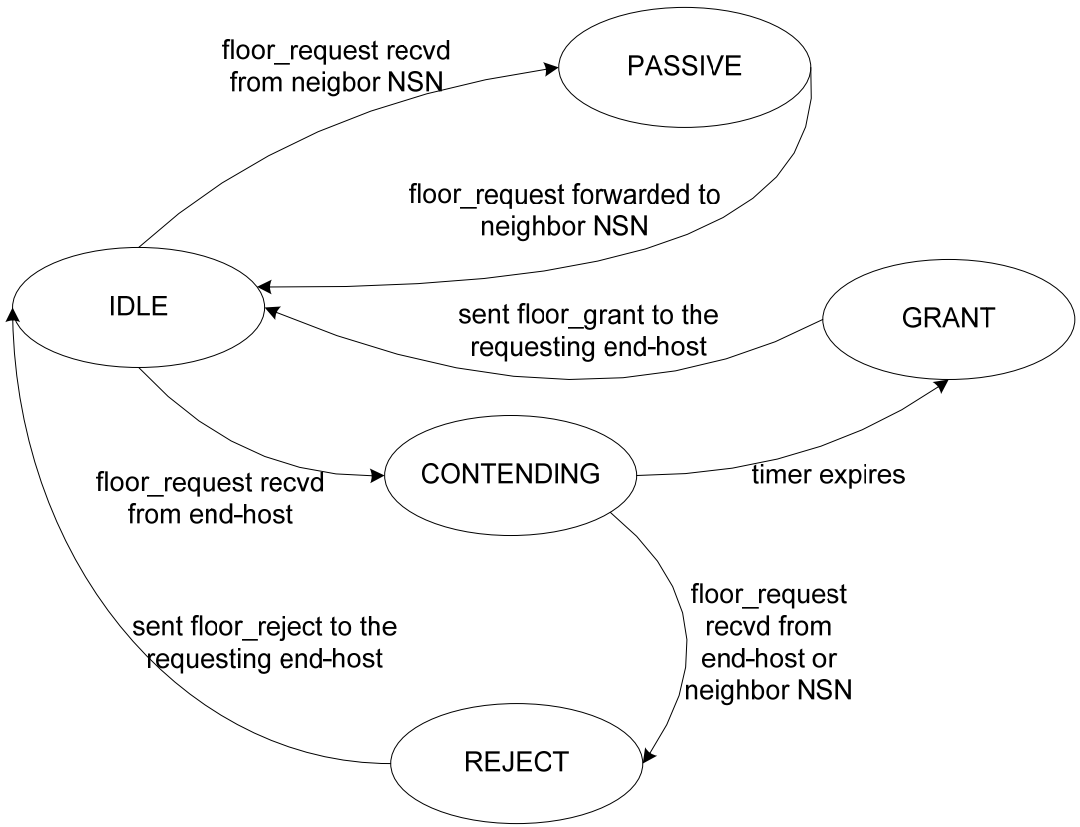
Generic Protocol for Floor Acquisition at the End-user

end-user_floor_acquisition( )
Begin
If end-user wants to acquire floor
Generate *floor_request* message
Send *floor_request* message to the NSN
End If
If end-user receives *floor_grant* message from the NSN
End-user goes to *Data Delivery Phase* for sending data to other end-users
Else If end-user receives *floor_reject* message from the NSN
End-user waits for random amount of time
Call *end-user_floor_acquisition*( )
End If
End

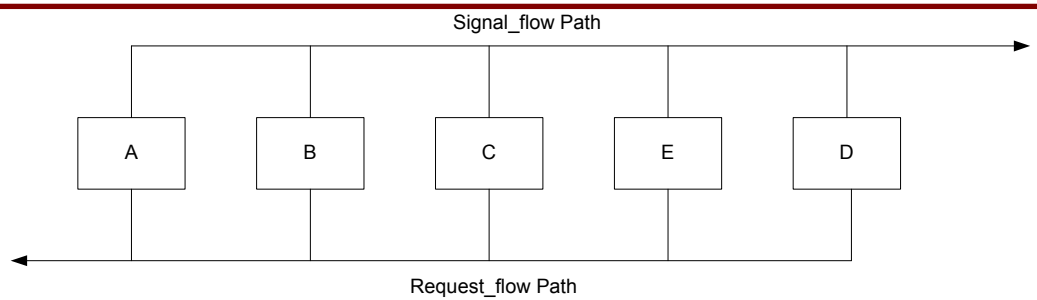
Implementing ALOHA for Floor Control among the participating NSNs

- Pure ALOHA
  - Nodes send data whenever they have data to send
  - When two nodes send data at the same time, a collision occurs
- How will an NSN detect a collision
  - NSN will wait for a certain period of time after forwarding the floor request signal (received from its end-user) to other NSNs
  - During the waiting period, if the NSN receives another floor request signal, it determines that a collision has occurred
- What will be the length of the waiting period
  - $2C$ *why?*

State Diagram of ALOHA



Implementing DQDB for Floor Control among the participating NSNs



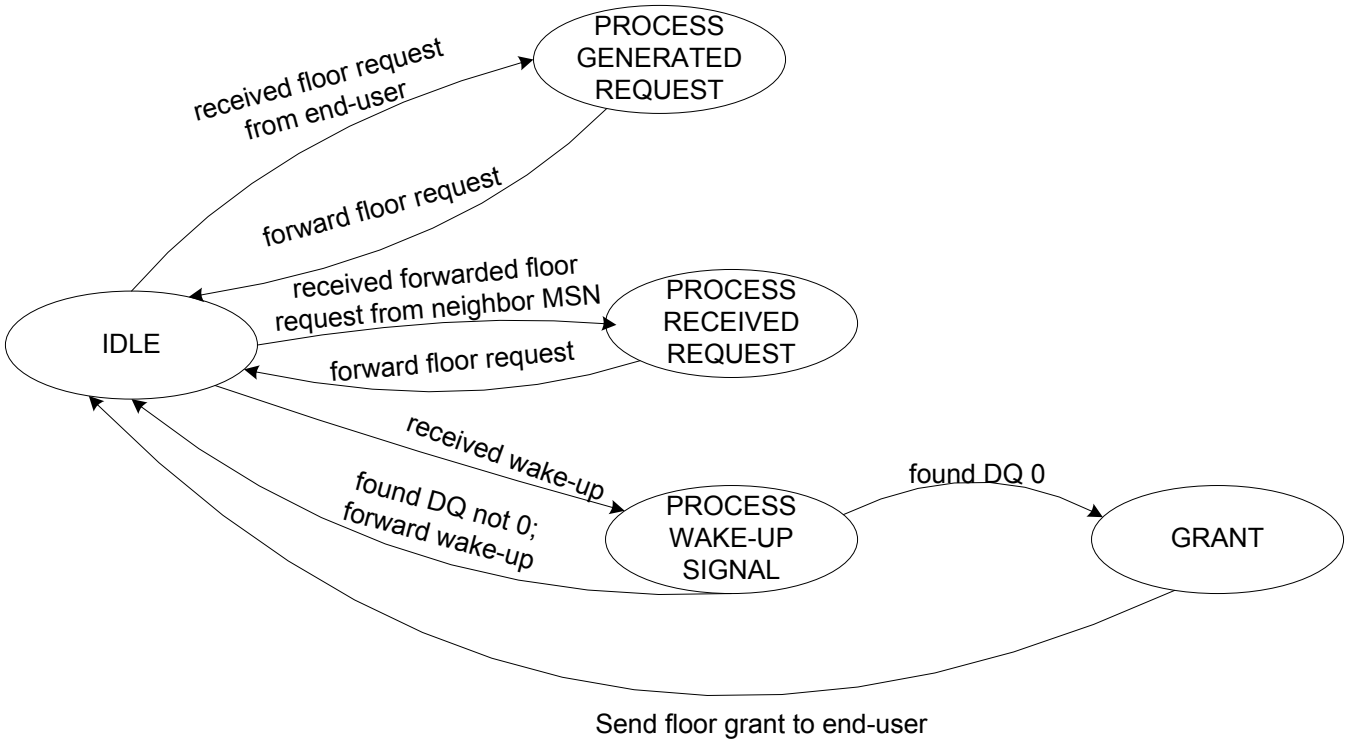
- Two logical paths are maintained among the NSNs
  - Request_flow_path
  - Signal_flow_path
- Each NSN knows its neighbors in each path
- First NSN in signal_flow_path is head_NSN
- First NSN in request_flow_path is tail_NSN
- Each NSN maintains two variables
  - RQ: keeps track of number of requests forwarded by the preceding NSNs in the request_flow path before the NSN gets its chance to acquire the floor for its end host
  - DQ: denotes the position of the NSN's end-user's request in the distributed queue

Implementing DQDB for Floor Control among the participating NSNs (contd.)

- Purpose of the request_flow path
  - When an NSN receives a request from its end-user
    - It forwards the request to its left neighbor in the request_flow path
    - It copies the value of RQ to DQ and sets RQ to 0
  - When an NSN receives a request from its neighbor NSNs
    - It increments its RQ value by 1
    - It forwards the request to its left neighbor in the request_flow path

- Purpose of the signal_flow path
  - The head_NSN generates wake_up signals periodically and forwards them to its right neighbor in the signal_flow path
  - When an NSN receives a wake_up signal in the signal_flow path
    - It checks the contents of the wake_up signal
    - If the wake_up signal does not contain end-host ID and the NSN's DQ value is 0
      - The NSN sends floor grant signal to its end-user and puts the end-user ID in the wake_up signal and forwards the wake_signal to its right neighbor in the signal_flow path
    - If the NSN's DQ value is not 0
      - The NSN decrements its DQ value and forwards the wake_up signal to its right neighbor in the signal_fow path

State Diagram of DQDB



Data Delivery Phase

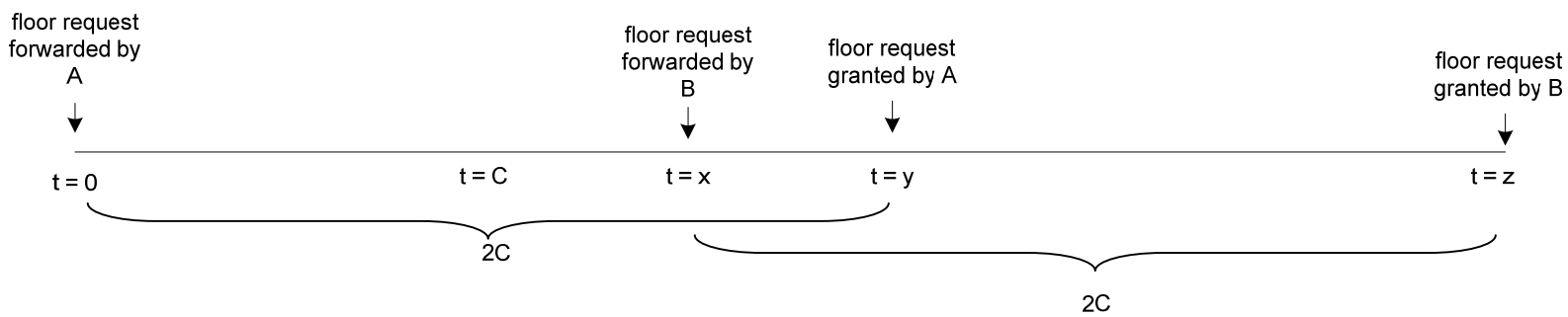
- Participating chain and optimal multicasting tree will be used in this phase
  - Optimal multicasting tree
    - Single source shortest path tree when we want to minimize the maximum end-to-end delay
    - Multicasting tree with minimum end-to-end delay and minimum delay variation when we want to ensure message should reach all the destination almost at the same time
- How to ensure Causal Ordering of messages
  - Connect the root of the multicasting tree to the last_NSN (tail_NSN) in the participating chain

Data Delivery Phase for ALOHA

- When an end-user enters into the data delivery phase, it sends data to its NSN
- The NSN forwards the data along the participating chain until it reaches the last NSN
- The last NSN forwards the data to the root of the multicasting tree which forwards it to all the participating end-users

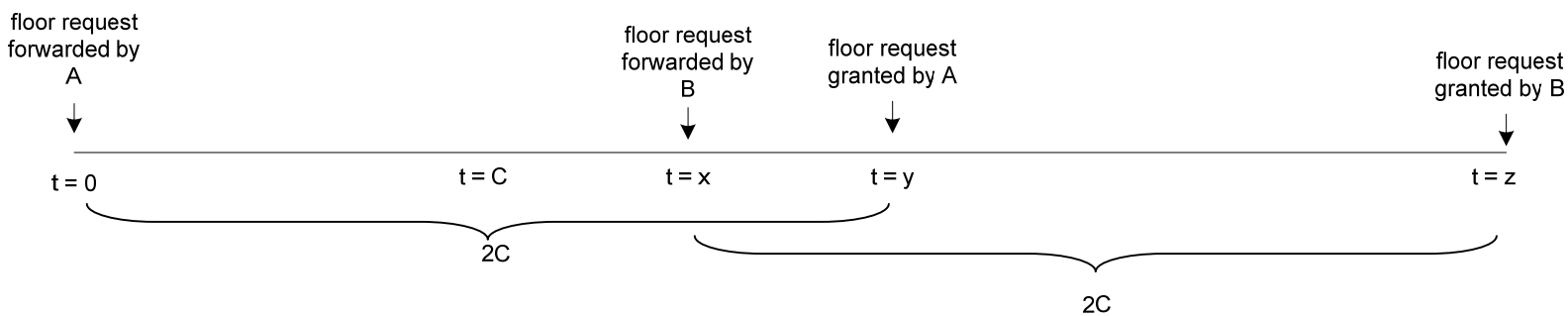
Data Delivery Phase for ALOHA

- How does this implementation ensure causal ordering
  - Consider the following worst case scenario
    - NSN A and B are the first and last NSN of the participating chain respectively
    - At t = 0, A forwards the floor request of its end-user to its neighbors
    - If A does not get any floor request in 2C time, it grants the floor to its end-user at t = y = 2C.
    - The end-user of A will send data to A and A will forward the data to the root of the multicasting tree
    - It will take C + δ units of time to reach the data from A to the root where δ is the path delay between B (last NSN) and the root of the multicasting tree



Data Delivery Phase for ALOHA

- How does this implementation ensure causal ordering
  - Consider the following worst case scenario
    - Now consider that B forwards a floor request (generated by its end-user) at t = x and clearly x > C
    - Assume that B does not get any floor request in 2C waiting period
    - So B grants floor to its end-user at t = z = x + 2C
    - Now the end-user of B will send data to B and B will forward it to the root
    - It will take δ units time to reach the data at the root
  - Need to show that the root receives the data of A before B
    - The root will receive the data of A at t = u = 2C + C + δ = 3C + δ
    - The root will receive the data of B at t = v = x + 2C + δ
    - It can be easily seen that v > u as x > C



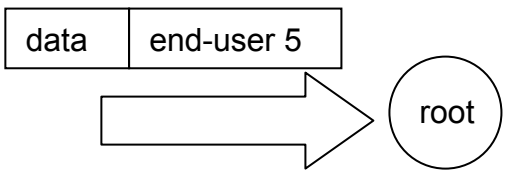
Data Delivery Phase for DQDB

- When the tail_NSN receives the wake_up signal in the signal_flow_path, it forwards the signal to the root of the multicasting tree
- The root keeps the sequence number of the wake_up signal in the database
- If the wake_up signal does not contain an end-user ID, then the root assumes that the wake_up signal was not used by any end-user
- If the wake_up signal contains an end-user ID, the root keeps both the sequence number and the end-user ID in the database

Wake_up signal	End-user ID
1	4
2	5
3	2

Data Delivery Phase for DQDB

- When an end-user receives the floor_grant signal from its NSN, it sends data along with its ID to its NSN
- The NSN forwards the data along the signal_flow_path until the data reaches the tail_NSN, then the tail_NSN forwards the data to the root of the multicasting tree
- The root checks the ID of the end-user in the database
- If the sequence number associated with the end-user is the smallest among the sequence numbers, the root forwards the data of the end-user to other participating end-users using the multicasting tree and removes the entry from the database, otherwise the root buffers the data



wake_up signal	end-user ID
1	4
2	5
3	2

Analytical Model

- Efficiency

$$\eta = \frac{U}{T} = \frac{U}{X + U}$$

- U: Floor Usage Time
- T: Turn Around Time
- X: Contention Time
- To derive the expression of X, we consider the worst case scenario for each protocol using communication infrastructure
- We assume the control messages in the protocol are transferred reliably over the network

Analytical Model

Parameter	Description
γ	Average processing time of control message at each NSN
δ	Duration of average activity period for each end-user
η	Efficiency of the protocol
λ	Floor request inter-arrival rate for end-users
σ	Average delay between an end-user and its NSN
n	Number of participating NSNs in the group communication
α	Frequency of wake-up signals generated by head NSN
C	Total Delay of the Communication Chain

Parameters used in the Analysis

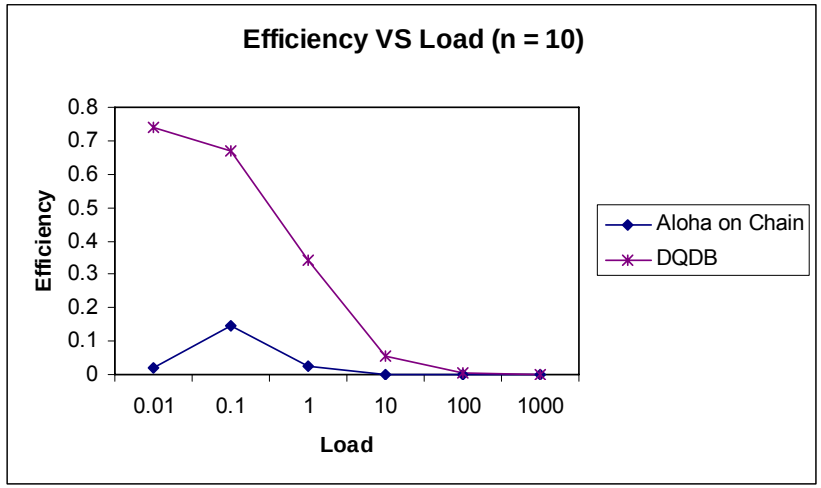
Efficiency of ALOHA

$$\eta = (\delta + \sigma + 2C + n\gamma)\,\lambda\, e^{-2(\delta + \sigma + 2C + n\gamma)\lambda}$$

Efficiency of DQDB

$$\eta = \delta / (((\sigma + C + n\gamma)\,\lambda) + C + \sigma + \delta)$$

Analytical Model



- On the average, DQDB outperforms ALOHA in terms of efficiency
- The efficiency of ALOHA reaches 0 when the request rate is 10 whereas the efficiency of DQDB reaches 0 when the request rate is near 1000

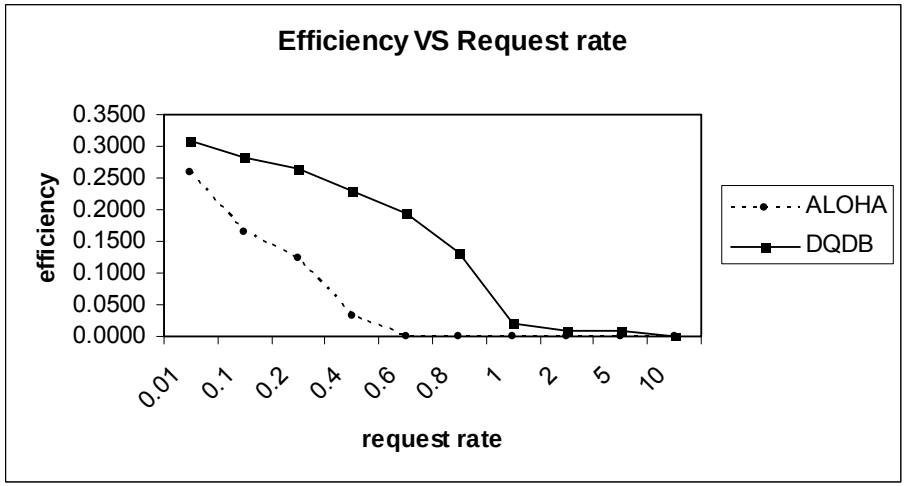
Simulation Experiments

- Both ALOHA and DQDB floor control protocols have been simulated using ns-2
- Considered a participating chain of 10 NSNs with 2 end-users connected to each NSN
- Simulation time set to 2000 seconds

Parameter	Value
request packet size	25 bytes
link delay	50 ms
bandwidth	1 Mbps
cell size for DQDB	44 bytes
contention window for ALOHA	20
cell generation rate for DQDB	1.0

Parameters used in the Simulations

Simulation Results



- DQDB performs better than ALOHA in terms of efficiency when the system request rate is high
- For ALOHA when the system request rate is low, the efficiency achieved from simulation experiments is higher than the efficiency calculated from the analytical model

Conclusion

- Proposed implementation of distributed MAC protocols on overlay networks for floor control
- Considered ALOHA and DQDB
- Constructed an efficient communication channel for floor control
- Implementation guarantees causal ordering
- Analytical and simulation results indicate that DQDB performs better than ALOHA in terms of efficiency