BITS Pilani
Hyderabad Campus

Distributed Computing (CS 5 – M5) Contd..

Distributed Mutual Exclusion Contd..

Prof. Geetha
Associate Professor, Dept. of Computer Sc. & Information Systems
BITS Pilani Hyderabad Campus
geetha@hyderabad.bits-pilani.ac.in

Suzuki–Kasami's Broadcast Algorithm

- if a site that wants to enter the CS does not have the token, it broadcasts a REQUEST message for the token to all other sites
- The site that possesses the token sends it to the requesting site upon the receipt of its REQUEST message
- if a site receives a REQUEST message when it is executing the CS, it sends the token only after it has completed the CS execution
- Two design issues exist:
  - how to distinguish an outdated REQUEST message from a current REQUEST message ?
  - how to determine which site has an outstanding request for the CS?

Suzuki–Kasami's Broadcast Algorithm

How to distinguish an outdated REQUEST message from a current REQUEST message ?

- a site may receive a token request message after the corresponding request has been satisfied
- if a site cannot determine if the request corresponding to a token request has been satisfied, it may dispatch the token to a site that does not need it

Suzuki–Kasami's Broadcast Algorithm

- **How to determine which site has an outstanding request for the CS?**

- after a site has finished the execution of the CS, it must determine what sites have an outstanding request for the CS so that the token can be dispatched to one of them
- after the corresponding request for the CS has been satisfied at S_j, an issue is how to inform site S_i and all other sites efficiently

Suzuki–Kasami's Broadcast Algorithm

- outdated REQUEST messages are distinguished from current REQUEST messages in the following manner:
  - a REQUEST message of site S_j has the form REQUEST(j, sn) where sn (sn = 1, 2…..) is a sequence number that indicates that site S_j is requesting its sn^{th} CS execution
  - a site S_i keeps an array of integers $RN_i[1, \ldots, n]$ where $RN_i[j]$ is the largest sequence number received in a REQUEST message so far from site S_j
  - when site S_i receives a REQUEST(j, sn) message, it sets $RN_i[j] = \max(RN_i[j], sn)$
  - when a site S_i receives a REQUEST(j,sn) message, the request is outdated if $RN_i[j] > sn$

Suzuki–Kasami's Broadcast Algorithm

- sites with outstanding requests for the CS are determined in the following manner:
  - the token consists of a queue of requesting sites, Q, and an array of integers LN[1, … ,n], where LN[j] is the sequence number of the request which site S_j executed most recently
  - after executing its CS, a site S_i updates LN[i] = RN_i[i] to indicate that its request corresponding to sequence number RN_i[i] has been executed
  - token array LN[1, … ,n] permits a site to determine if a site has an outstanding request for the CS
  - **at site S_i, if $RN_i[j] = LN[j] + 1$, then site S_j is currently requesting a token**
  - after executing the CS, a site checks this condition for all the j's to determine all the sites that are requesting the token and places their IDs in queue Q if these IDs are not already present in Q
  - finally, the site sends the token to the site whose ID is at the head of Q

Suzuki–Kasami's Broadcast Algorithm

Requesting the critical section:

(a) If requesting site S_i does not have the token, then it increments its sequence number, $RN_i[i]$, and sends a REQUEST(i, sn) message to all other sites. ("sn" is the updated value of $RN_i[i]$.)

(b) When a site S_j receives this message, it sets $RN_j[i]$ to max($RN_j[i]$, sn). If S_j has the idle token, then it sends the token to S_i if $RN_j[i] = LN[i]+1$.

Executing the critical section:

(c) Site S_i executes the CS after it has received the token

Suzuki–Kasami's Broadcast Algorithm

Releasing the critical section: Having finished the execution of the CS, site S_i takes the following actions:

(d) It sets LN[i] element of the token array equal to $RN_i[i]$.

(e) For every site S_j whose ID is not in the token queue, it appends its ID to the token queue if $RN_i[j] = LN[j] + 1$.

(f) If the token queue is non-empty after the above update, S_i deletes the top site ID from the token queue and sends the token to the site indicated by the ID

Suzuki–Kasami's Broadcast Algorithm

- ✓ **Correctness**
- ✓ mutual exclusion is guaranteed because there is only one token in the system and a site holds the token during the CS execution
- ✓ a requesting site enters the CS in finite time

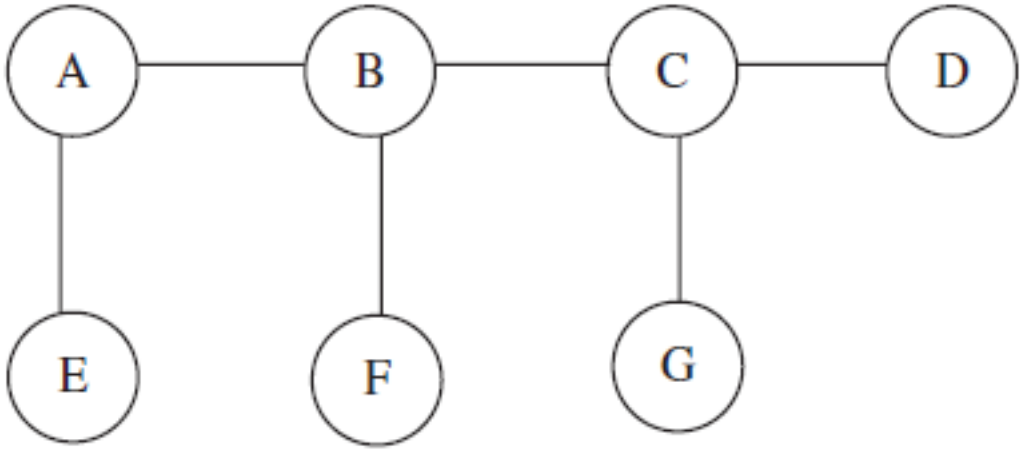
- ✓ **Performance**
- ✓ if a site holds the token, no message is required
- ✓ if a site does not hold the token, N messages are required
- ✓ synchronization delay is 0 or T

Raymond's Tree-Based Algorithm

- uses a spanning tree of the computer network to reduce the number of messages exchanged per critical section execution
- assumes that the underlying network guarantees message delivery
- time or order of message arrival cannot be predicted
- all nodes of the network are completely reliable
- a spanning tree of a network of N nodes will be a tree that contains all N nodes
- a minimal spanning tree is a spanning tree with minimum cost
- cost function is based on the network link characteristics

Raymond's Tree-Based Algorithm

- messages between nodes traverse along the undirected edges of the tree
- node needs to hold information about and communicate only to its immediate-neighboring nodes



- tree is a spanning tree of 7 nodes A, B, C, D, E, F, and G
- node C holds information about and communicates only to nodes B, D, and G
- node C does not need to know about the other nodes A, E, and F for the operation of the algorithm

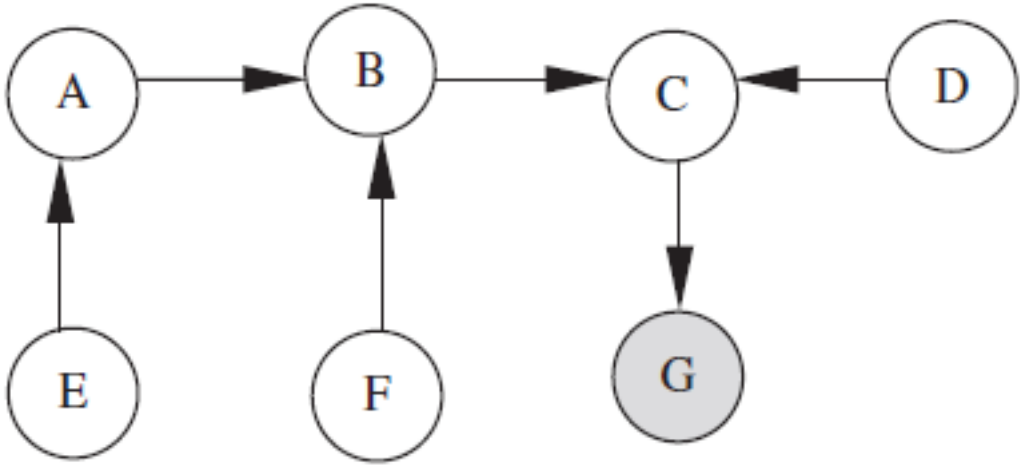
Raymond's Tree-Based Algorithm

- only one node can be in possession of the privilege (called the privileged node) at any time
  - except when the privilege is in transit from one node to another in the form of a PRIVILEGE message
- when there are no nodes requesting for the privilege, it remains in possession of the node that last used it

Raymond's Tree-Based Algorithm

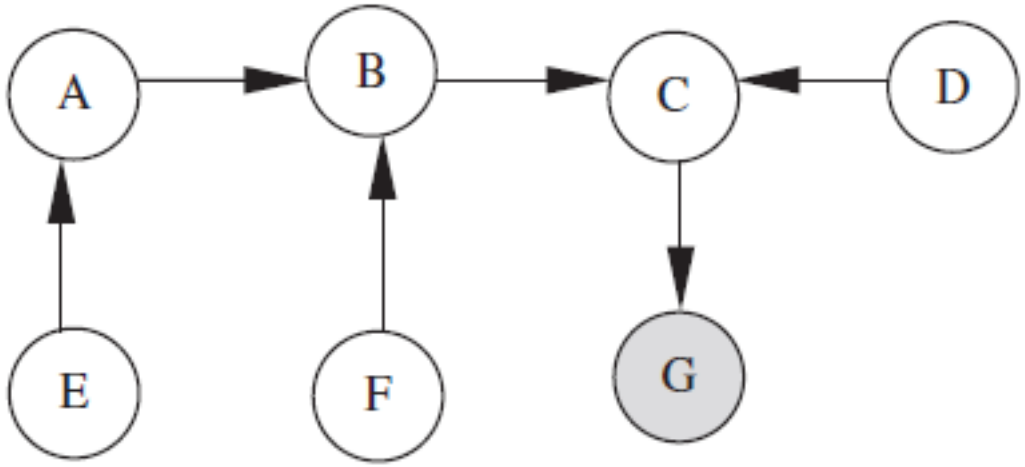
- each node maintains a HOLDER variable that provides information about the placement of the privilege in relation to the node itself
- a node stores in its HOLDER variable the identity of a node that it thinks has the privilege or leads to the node having the privilege
- HOLDER variables of all nodes maintain directed paths from each node to the node in possession of the privilege
- for two nodes X and Y, if $HOLDER_X$ = Y, the undirected edge between X and Y can be redrawn as a directed edge from X to Y

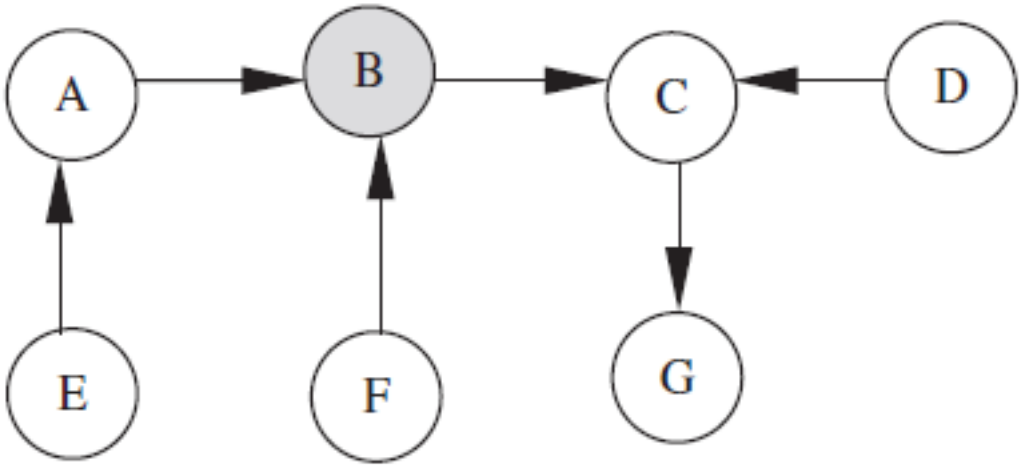
Raymond's Tree-Based Algorithm



- if node G holds the privilege, figure can be redrawn
- shaded node represents the privileged node
- $HOLDER_A$ = B (as the privilege is located in a sub-tree of A denoted by B)
- $HOLDER_B$ = C
- $HOLDER_C$ = G
- $HOLDER_D$ = C
- $HOLDER_E$ = A
- $HOLDER_F$ = B
- $HOLDER_G$ = self

Raymond's Tree-Based Algorithm





- node B does not hold the privilege and wants to execute the CS
- B sends a REQUEST message to $HOLDER_B$, i.e., C, which in turn forwards the REQUEST message to $HOLDER_C$, i.e., G
- privileged node G, if it no longer needs the privilege, sends the PRIVILEGE message to its neighbor C, which made a request for the privilege, and resets $HOLDER_G$ to C
- C forwards the PRIVILEGE to B, since it had requested the privilege on behalf of B
- C also resets $HOLDER_C$ to B

Raymond's Tree-Based Algorithm

- **Data Structures**
- HOLDER
  - possible values "self " or the identity of one of the immediate neighbors
- USING
  - possible values true or false
  - indicates if the current node is executing the critical section
- ASKED
  - possible values true or false
  - indicates if node has sent a request for the privilege
  - prevents the sending of duplicate requests for privilege

Raymond's Tree-Based Algorithm

Data Structures

- REQUEST_Q
  - FIFO queue that could contain "self " or the identities of immediate neighbors as elements
  - REQUEST_Q of a node consists of the identities of those immediate neighbors that have requested for privilege but have not yet been sent the privilege
  - maximum size of REQUEST_Q of a node is the number of immediate neighbors + 1 (for "self ")

Raymond's Tree-Based Algorithm

Correctness

The algorithm guarantees the following:

- **mutual exclusion**
- **deadlock is impossible**
- **starvation is impossible**

Raymond's Tree-Based Algorithm

<u>Cost and performance analysis</u>

The algorithm exchanges to execute the CS

- *O*(log N) messages under light load
- approximately four messages under heavy load,

N being the number of nodes in the network

Recap Quiz

Q1. The system throughput in distributed computing systems in terms of synchronization delay SD and average execution time E at the CS can be expressed as

(a) SD + E (b) 1/(SD + E) (c) 1/SD + 1/E (d) SD – E

Q2. The Lamport's mutual exclusion algorithm requires the channel to deliver messages in _____

(a) FIFO (b) non-FIFO (c) causal order (d) none of the above

Q3. The Lamport's mutual exclusion algorithm requires ____ messages per CS invocation

(a) 2(N-1) (b) (N-1) (c) 4(N-1) (d) 3(N-1)

Q4. The Ricart-Agrawala algorithm uses ____ to assign a timestamp to critical section requests

(a) Scalar clock (b) vector clock (c) NTP (d) none of the above

Q5. In which of the following mutual exclusion algorithms, each process maintains a deferred array?

(a) Maekawa (b)Lamport's (c) Ricart-Agrawala (d) Suzuki-Kasami

Q6. Consider the performance metrics of mutual exclusion algorithms. If there is always a pending request for critical section at a site, then it is called

(a) Low load (b) high load (c) medium load (d) none of the above

Q7. Which of the following is a quorum based distributed mutual exclusion algorithm?

(a) Maekawa (b)Lamport's (c) Ricart-Agrawala (d) Suzuki-Kasami

Q8. The name of the variable that provides information about the placement of the privilege in relation to the node itself in Raymond's tree based algorithm is called

(a) Marker (b) signal (c) assertion (d) holder

Q9. If the number of sites is N = 25, then according to the Maekawa algorithm, then the number of request sets of sites, K will be

(a) 5 (b) 10 (c) 15 (d) 25

Q10. Which of the following is a token based distributed mutual exclusion algorithm?

(a) Maekawa (b)Lamport's (c) Ricart-Agrawala (d) Suzuki-Kasami

Recap Quiz - Key

Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10
b	a	d	a	c	b	a	d	a	d

Reference

❖Ajay D. Kshemkalyani, and Mukesh Singhal, Chapter 9, “Distributed Computing: Principles, Algorithms, and Systems”, Cambridge University Press, 2008 (Reprint 2013).