CS 499- Distribute System  
Friday, November 06, 2009

TODO:

Class Notes:

* Quiz 6 graded and returned.
  + (1) OS Algorithms cannot be applied to DS because you cannot have the same access to the same place in memory
    - TSL – cannot be used
  + (2) Bully depends on Sync because we need timeouts
  + (3)
    - (a) yes
    - (b) no
* Projects Due – Monday
* Wednesday – No class
* Friday – Grad Presentation, Project Documentation Due

Lecture Notes:

* Multicasting
  + // last lecture, unless graduates finish up early
  + Three ordering processes
    - FIFO
    - Causal
      * When a process responds to another process then those processes must maintain order across the all transactions.
    - Total
      * all messages, across the whole group, across all processes - have the same order
      * unambiguously
  + Implementation of Ordering Algorithms
    - FIFO Ordering
      * Sequence #’s
      * Process P has two variables:
        + S 🡪 counts how many messages P has sent

// each other process knows this number when multicasted

* + - * + Rgq - sequence # of the latest message P has delivered from process q (sent to g)

g – Group

q – place holder of processes in g

* + - * On Multicasting, message from P to group g:
        + P piggybacks S on message m.
        + P multicast m
        + P increments S
      * On Delivering message m from q:
        + P checks S contained in the message == Rgq + 1

If this is true 🡪 then deliver!

Let R 🡪 S

else keep it back

If S > R + 1

// too new, missing messages in between

Message place in hold-back queue, until all intermediate messages have been delivered & S = R + 1.

* + - Total Ordering
      * // sequence number used to keep messages in a group
      * sequencer – used to keep count and broadcast to group
      * (1) Algorithm for group member p
        + On initialization: rg := 0;
        + To TO-Multicast (total ordering) message m to group g

B-mulitcast ( g U { sequencer(g) }, <m, i> );

* + - * + On B-deliver( <m, i> ) with g = group(m)

Place <m, i> in hold-back queue;

* + - * + On B-deliver( morder = < “order”, i, S>) with g = group( morder )

wait until <m, i> in hold-back queue and S = rg;

TO-deliver m;

rg = S + 1;

* + - * (2) Algorithm for sequence of g:
        + On initialization: sg := 0;
        + On B-delver( <m, i> with g = group(m)

B-multicast( g, < “order”, i, sg> );

sg := sg + 1;

* + - Causal Ordering
      * Algorithm for group member pi( i = 1, 2, …, N )
      * ….
      * // save for beginning of Monday