CS 499 – Distributed Systems  
Wednesday, November 04, 2009

TODO:

Class Notes:

* Quiz 6
  + (1) The mutual exclusion algorithms we know from operating systems don’t work in distributed systems. Why?
  + (2) Why is it essential for the bully algorithm to be run in synchronous systems (only)?
  + (3) Consensus: is it possible to agree on simple facts in an unreliable (concerning message), asynchronous system? How about reliable (again concerning message transfer) asynchronous systems?
    - (a) no -
    - (b) yes – this is possible by assuming the system is synchronous.
* Schedule
  + Friday – Lecture: Multicast Ordering
  + Monday – Project Due/Presentation
  + Wednesday – no class
  + Friday – Grad presentation
  + Monday - (possibly Replication lecture)

Lecture Notes:

* Why can the Internet have consensus?
  + Given that we know, the Internet is an asynchronous system.
  + Answer: Timeouts
    - The rate of packets that are thrown out are high and reaches a drop limit (timeouts)
  + Answer: We assume the system is synchronous.
* Multicast-Communication
  + Introduction
    - Def: Multicast – variation on Broadcast.
      * We are not interested in the technical information on how to do Multicasting
      * We are interested in the implementation (not the technical)
    - Multicast vs. Broadcast
    - Closed Group vs. Open Group Multicast
      * Closed Group:
        + Message can be sent in within the group but no processes outside the group can send messages.
      * Open Group:
        + Outside processes can send messages to the group
        + But also inner group processes send messages to each other
      * BOTH can send message from inside the group to outside processes.
    - Distinction between Receiving & Delivering messages:
      * Question: how are you going to order the message at the receiver site(s)?
        + // pretty much all we will discuss about
        + understand: in order to understand a message ordering

Basically learn that you do not always deliver a message as soon as that is ready.

Rather you wait till all messages are ready then delivered to receiver in a set order.

* + - General Technique

Incoming messages

Hold-back queue

When delivery guarantees are met

Delivery queue

Message processing

Delivery

* + - * “hold-back queue”
        + sets the order of messages to be delivered
        + // waits for messages to be completed before sending.
      * Delivery Queue
        + set of messages ready to deliver
        + then delivered
    - What delivery guarantees are we waiting for?
      * FIFO ordering:
        + applies to a single sending process
        + If a process sends a message m before m’, then every process that delivers m’ will deliver m before m’
      * Causal Ordering
        + can be applied to multiple process

FIFO is implied

* + - * + If multicast(g, m) 🡪 multicast(g, m’)
        + Every process that delivers m’ will deliver m before m’.
      * Total ordering
        + If a process delivers message m before m’ then any other process that delivers m’ will deliver m before m’.
        + All messages will arrive in order, everywhere.