**Notes on Global Snapshot protocol**

* Notes
  + For CSMA time sync, one message suffices to sync time, assuming same clock speeds & clock drift.
  + Additional messages help calculate relative speed / drift.
* Assumptions:
  + On Sensor node, execution of snapshot is (a) same for all Sensor nodes and (b) is deterministic in time. This avoids issues wrt when taken vs. when sent.
  + At most one snapshot request is pending at a time.
  + Only fault is link error.
  + Using CSMA with time sync features.
  + Clocks are normalized for all nodes so that CPU clock speed rate differences are not an issue; and clock skew is small for small time spans. This means the CSMA message received time diff is accurate.
* Messages:
  + Base
    - Broadcast timestamped RequestSnapshot.
  + Sensor
    - Unicast timestamped SnapshotData
* Base
  + Variables
    - SnapshotId. Integer, initially 0. Unique id.
    - SnapshotPending: bool, initially false
  + On demand or periodically, with SnapshotTime as argument
    - If SnapshotPending is true: return
    - If SnapshotTime is not greater than CurrentTime + delta (to allow time for replies): return
    - Increment SnapshotId
    - Broadcast RequestSnapshot time-stamped message (LocalTime); payload contains LocalTime, SnapshotTime & SnapshotId
    - Set SnapshotPending = true
  + On receipt of SnapshotData unicast timestamped message
    - Check if for this app; if not, return
    - If message.SnapshotId <> SnapshotId: return
    - Save message.SensedTime, message.SnapshotId & message.SensedData
    - Set SnapshotPending = false
* Sensor
  + Variables
    - BaseAddress: int: initially -1
    - InitialTime: (long, long)
    - LastTime: (long, long)
    - SnapshotTime: long
  + On receipt of any broadcast message
    - Check if for this app; if not, return.
    - If first message [BaseAddress < 0]
      * Set BaseAddress = message.Src address
      * Set InitialTime(Base, Sensor) = (msg.SenderEventTimeStamp, currTime)
    - Set LastTime(Base, Sensor) = (msg.SenderEventTimeStamp, currTime)
    - If RequestSnapshot message:
      * Set SnapshotId = message.SnapshotId
      * Set Skew =
        + Skew = 1 iff Base and Sensor clocks running at same rate
        + Skew > 0 iff Base clock is faster
        + Skew < 0 iff Sensor clock is faster
      * Set SnapshotTime = InitialTime.Sensor + (message.SnapshotTime – InitialTime.Base) \* Skew
      * Set SnapshotInterval = SnapshotTime – currTime
      * Start one-shot timer TakeSnapshot(SnapshotInterval)
  + On tick of TakeSnapshot timer
    - Set SensorData as sensed data
    - Send CurrentData as timestamped (currTime) message. Payload consists of SnapshotId and SensorData