CS 425 Distributed Systems, Fall 2010, University of Illinois at Urbana-Champaign

Homework 2: Solution

1. In class we discussed how varying the waiting timeout (in heartbeat and ping/ack protocols) trades off between detection time and false positive rate. What parameter in these failure detector protocols would you vary in order to trade off between bandwidth and detection time? Explain briefly why (1-2 sentences).

   Solution:

   In heartbeat and ping/ack protocols the frequency of heartbeat and ping can be varied respectively to trade off between bandwidth and detection time. With increased heartbeat/ping frequency, the detection time reduces – while the bandwidth usage increases.

2. Calibrations on a recent version of the Linux operating system showed that on the client side, there is a delay of at least 1.3 ms for a packet to get from an application to the network interface, and a similar minimum delay for the opposite path (network interface to application buffer). The corresponding minimum delays for the server are 0.15 ms and 0.35 ms respectively. What would be the accuracy of a run of the Cristian's algorithm between a client and server, both running this version of Linux, if the round trip time measured at the client is 5 ms?

   Solution:

   The earliest point at which the server could have placed the time in message m_t was min_1 = $1.3 + 0.35 = 1.65$ ms after the client dispatched message m_r. The latest point at which the server could have done this was min_2 = $0.15 + 1.3 = 1.45$ ms before message m_t arrived at the client. The time by the server's clock when m_t arrives at the client is therefore in the range: $[t + min_2, t + RTT - min_1]$, where t is the time placed by the server in m_t.

   The width of this range is $(RTT - min_1 - min_2)$ = 1.9 ms, and the accuracy is thus: $\pm(RTT - min_1 - min_2)/2 = \pm 0.95 ms$.

3. Problem 11.2 from text.

   Solution:

   It is undesirable to set the clock back to right time because a process may implicitly depend on monotonically increasing values for time. For example, the make command checks if recompilation is necessary based on the the update timestamps of its dependencies.

   Current time according to the clock: 10:27:54.0 (hr:min:sec)
   Current actual time: 10:27:50.0

   After 8 seconds both clocks should read: 10:27:58.0

   Therefore, to adjust the clock we need to gain only 4 seconds in the software clock versus 8 seconds in the hardware clock. So, we need to move at $\frac{4}{8} = 0.5$ the pace of the hardware clock over the next 8 seconds.

4. In Figure 11.6 in the textbook, list *all* pairs of concurrent events.

   Solution:

   The set of concurrent events is: $\{a \| e, b \| e, c \| e, d \| e\}$.

5. Problem 11.14 from text.

   Solution:

(a) P sends message m.
(b) P records state: 101.
(c) P sends marker.
(d) Q receives m, making its state 102.
(e) Q receives the marker and by marker-receiving rule, records its state: 102 and the state of the channel from P to Q as: $\{\}$.
(f) Q sends marker (marker-sending rule). (Q sends m again at some point later).
(g) P receives marker and records the state of the channel from Q to P as set of messages received since it saved its state: $\{\}$ (marker-receiving rule).

6. Explain why the impossibility of consensus proof (proofs of Lemmas 2 and 3 in the FLP paper) would break if the system were synchronous. Specifically, give at least one statement in the proof that may not hold in a synchronous system.

   Solution:

   Lemma 2 is not violated in a synchronous system. Lemma 3, however, uses the assumption that there maybe arbitrary message delays. While this is true in an asynchronous system, message delays are bounded in a synchronous system.

7. In the given figure (see next page), if all processes start with sequence numbers or vector timestamps (as applicable) containing all zeroes, for each of the following algorithms, mark the timestamps at the point of each multicast send and each multicast receipt. Also mark multicast receipts that are buffered, along with the points at which they are delivered to the application.

   (a) FIFO Ordering algorithm discussed in class.
   (b) Causal Ordering algorithm discussed in class.

   Solution:

   (a) FIFO Ordering:

   - **P1:**
     Send, [1, 0, 0]
     Send, [2, 0, 0]
     Accept, [2, 0, 1]
     Send, [3, 0, 1]
     Accept, [3, 0, 2]
     Accept, [3, 1, 2]

   - **P2:**
     Accept, [0, 0, 1]
     Buffered as [2, 0, 1], Timestamp still [0, 0, 1]
     Send, [0, 1, 1]
     Accept, [1, 1, 1]
     Accept, [2, 1, 1] (Previously buffered)
     Accept, [2, 1, 2]
     Accept, [3, 1, 2]

   - **P3:**
     Accept, [1, 0, 0]
     Send, [1, 0, 1]

Accept, [2, 0, 1]
Accept, [2, 1, 1]
Send, [2, 1, 2]
Accept, [3, 1, 2]

(b) Causal Ordering:

- **P1:**
  Send, [1, 0, 0]
  Send, [2, 0, 0]
  Accept, [2, 0, 1]
  Send, [3, 0, 1]
  Buffered as [2, 1, 2], Timestamp still [3, 0, 1]
  Accept, [3, 1, 1]
  Accept, [3, 1, 2] (Previously buffered)

- **P2:**
  Buffered as [1, 0, 1], Timestamp still [0, 0, 0]
  Buffered as [2, 0, 0], Timestamp still [0, 0, 0]
  Send, [0, 1, 0]
  Accept, [1, 1, 0]
  Accept, [1, 1, 1] (Previously buffered)
  Accept, [2, 1, 1] (Previously buffered)
  Accept, [2, 1, 2]
  Accept, [3, 1, 2]

- **P3:**
  Accept, [1, 0, 0]
  Send, [1, 0, 1]
  Accept, [2, 0, 1]
  Accept, [2, 1, 1]
  Send, [2, 1, 2]
  Accept, [3, 1, 2]