The University of Melbourne
School of Computing and Information Systems
COMP90020 Distributed Algorithms

Tutorial Week 2: Distributed Clocks

Solutions

1.

(a) In order to satisfy the *Monotonicity Condition* $t' > t \Rightarrow C(t') > C(t)$ where t is the time and $C(t)$ is the clock time. This can affect applications which use timestamps, e.g. logging across distributed servers.

(b) Denote E to be the errant clock (E is initially 10:27:54.0). We assume that H the hardware clock of E advances at a perfect rate, to a first approximation, over the next 12 seconds. In order to satisfy the monotonic condition, we adjust the software clock (S) tick rate so that it will be correct after 12 seconds, as follows:
$S = c(E - T_{skew}) + T_{skew}$ where $T_{skew} = 10:27:54$ and c is to be found.
$S = T_{skew} + 8$, (the correct time) when $E = T_{skew} + 12$
$T_{skew} + 8 = c(T_{skew} + 12 - T_{skew}) + T_{skew}$, we get $c = \frac{2}{3}$
$S = \frac{2}{3}(E - T_{skew}) + T_{skew}$

Note that if this level of detail is certainly not necessary to answer the question. Instead, an informal answer follows: The clock E is 4 second fast. So the current, time is $E - 4$. We want this clock to be correct in 12 seconds, whilst relatively for clock E, this time will be $E + 8$. So clock E has to tick 8 seconds over 12 seconds of real time. Hence we can adjust its rate to be $\frac{8}{12}$ of the hardware clock, assuming it has neglible drift over that period.

2.

(a) $T \leq T'$, *...the largest message timestamp it has seen.*

(b) We know the earliest possible timestamp that could arrive when the receivers clock is T_r is $T_r - 50 - 101$. We therefore need $T_r - 151 = 175,000$. Therefore $T_r = 175,151$

(c) In this example, all times are relative to the sending and receiving processes. Therefore internal synchronization is sufficient. A follow up to this is - under what new condition, would introducing external synchronization be necessary?

3.

(a) The client should estimate the current time to be $t + \frac{4}{2}$ = 10:55:24. The accuracy is the entire range of the round trip time, as the theoretical minimum transmission time is 0. So ± 2s.

(b) The client should estimate the current time to be $t + \frac{3}{2}$ = 10:55:23.5. The accuracy is $\pm\frac{(T_{round} - 2 \cdot T_{min})}{2} = \pm 0.5$s

4. We know that the minimum delay is 1s. We can use Christian's algorithm to determine the round trip time we must receive, in order to achieve the desired accuracy range of 2s. accuracy_range = $T_{round} - 2 * delay_{\min}$. Which substituting the appropriate values we get $T_{round} = 2 + 2 = 4$s. *Note the previous iteration of this problem stated a minimum round trip delay. This is inappropriate*

without having a method for calculating the proportion that represents the client to server request, as we cannot assume symmetry e.g. $\frac{request}{response} = 1$ *unless otherwise stated.*

5. let $a = T_{i-2} - T_{i-3} = 23.480 - 13.430 = 10.05$; $b = T_{i-1} - T_i = 25.7 - 15.725 = 9.975$. Then the estimated offset $o_i = (a + b)/2 = 10.013$s, with estimated accuracy $\pm\frac{d_i}{2} = \pm\frac{a-b}{2} = 0.038$s

6. From the definition for internal and external synchronization:

i) *External Synchronization*: For a synchronization bound $D > 0$ and a source S of UTC time, $|S(t) - C_i(t)| < D$, for $i = 1, 2, ...N$, and for all real times t.

ii) *Internal Synchronization*: For a synchronization bound $D > 0, |C_i(t) - C_j(t)| < D$ for all $i, j = 1, 2, ...N$, and for all real times t.

It is helpful to know the triangle inequality: $|x + y| \leq |x| + |y|$.

$$
\begin{aligned}
internal &= |C_i(t) - C_j(t)| && \text{from } ii \\
&= |C_i(t) - C_j(t) + S(t) - S(t)| && \text{adding } S(t) - S(t) = 0 \\
&= |C_i(t) - S(t) + S(t) - C_j(t)| && \text{rearranging for } i + i \\
&\leq |S(t) - C_i(t)| + |S(t) - C_j(t)| && \text{using triangle inequality} \\
&\leq 2 \cdot D
\end{aligned}
$$

7. If we know that the drift rate is constant, then we need only measure it between synchronization points with an accurate source and compensate for it. For example, if the clock loses a second every hour, then we can add a second every hour, in smooth increments, to the value returned to the user. The difficulty is that the clock's drift rate is liable to be variable – for example, it may be a function of temperature. Therefore we need an adaptive adjustment method, which guesses the drift rate, based on past behaviour, but which compensates when the drift rate is discovered to have changed by the next synchronisation point.

8. The main factors to take into account are the intrinsic reliability of the server as a source of time values, and the quality of the time information as it arrives at the destination. Sanity checks are needed, in case servers have bugs or are operated maliciously and emit spurious time values. Assuming that servers emit the best time values known to them, servers with lower stratum numbers are closest to UTC, and therefore liable to be the most accurate. On the other hand, a large network distance from a source can introduce large variations in network delays. The choice involves a trade-off between these two factors, and servers may synchronize with several other servers (peers) to seek the highest quality data.

9. A server may fail or become unreachable. Servers that synchronize with it will then attempt to synchronize to a different server. As a result, they may move to a different stratum. For example, a stratum 2 peer (server) loses its connection to a stratum 1 peer, and must thenceforth use a stratum 2 peer that has retained its connection to a stratum 1 peer. It becomes a stratum 3 peer.