Distributed Systems: Concepts and Design

Edition 5

By George Coulouris, Jean Dollimore, Tim Kindberg and Gordon Blair

Chapter 14 Exercise Solutions

14.1 What is clock skew? Also, explain the concepts of International Atomic Time and Coordinated Universal Time.

14.1 Ans.

The instantaneous difference between the readings of any two clocks is called their skew. A clock's drift rate is the change in the offset (difference in reading) between the clock and a nominal perfect reference clock per unit of time measured by the reference clock. The output of these atomic clocks is used as the standard for elapsed real time, known as International Atomic Time. Since 1967, the standard second has been defined as 9,192,631,770 periods of transition between the two hyperfine levels of the ground state of Caesium-133 (Cs133).

Coordinated Universal Time – abbreviated as UTC (from the French equivalent) – is an international standard for timekeeping.

14.2 A clock is reading 11:34:26.0 (hr:min:sec) when it is discovered to be 6 seconds fast. Explain why it is undesirable to set it back to the right time at that point, and show (numerically) how it should be adjusted so as to be correct after 12 seconds have elapsed.

14.2 Ans.

Hint: It is undesirable to set it back to the right time at this point, because it can create a major problem as the events which are occurring in the node differ in time by 6 seconds.

14.3 A scheme for implementing at-most-once reliable message delivery uses synchronized clocks to reject duplicate messages. Processes place their local clock value (a 'timestamp') in the messages they send. Each receiver keeps a table giving, for each sending process, the largest message timestamp it has seen. Assume that clocks are synchronized to within 100 ms, and that messages can arrive at most 50 ms after transmission.

(i) When may a process ignore a message bearing a timestamp T, if it has recorded the last message received from that process as having timestamp T' ?

(ii) When may a receiver remove a timestamp 175,000 (ms) from its table? (Hint: use the receiver's local clock value.)

(iii) Should the clocks be internally synchronized or externally synchronized?

14.3 Ans.

i) If T ð T' then the message must be a repeat.

ii) The earliest message timestamp that could still arrive when the receiver's clock is r is r - 100 - 50. If this is to be at least 175,000 (so that we cannot mistakenly receive a duplicate), we need r -150 = 175,000, i.e. r = 175,150.

iii) Internal synchronisation will suffice, since only time *differences* are relevant.

14.4 A client attempts to synchronize with a time server. It records the round-trip times and timestamps returned by the server in the table below.

Which of these times should it use to set its clock? To what time should it set it? Estimate the accuracy of the setting with respect to the server's clock. If it is known that the time between sending and receiving a message in the system concerned is at least 8 ms, do your answers change?

Round-trip (ms)	*Time (hr:min:sec)*
22	*10:54:23.674*
25	*10:54:25.450*
20	*10:54:28.342*

14.4 Ans.

The client should choose the minimum round-trip time of 20 ms = 0.02 s. It then estimates the current time to be 10:54:28.342 + 0.02/2 = 10:54:28.352. The accuracy is ± 10 ms.

If the minimum message transfer time is known to be 8 ms, then the setting remains the same but the accuracy improves to ± 2 ms.

14.5 Synchronization of clocks is necessary for communication between nodes. Explain the two modes of synchronization.

14.5 Ans.

External synchronization: For a synchronization bound $D>0$, and for a source S of UTC time, $|S(t)-C_i(t)|< D$, *for* $i=1,2,\ldots N$ and for all real times t in I. Another way of saying this is that the clocks C_i are *accurate* to within the bound D.

Internal synchronization: For a synchronization bound $D>0$, $|C_i(t)-C_j(t)|<D$ for $i,j=1,2\ldots N$, and for all real times t in I. Another way of saying this is that the clocks C_i *agree* within the bound D.

14.6 What is the main disadvantage of Cristian's method?

14.6 Ans.

Cristian's method suffers from the problem associated with all services implemented by a single server, which is that the single time server might fail and thus render synchronization impossible temporarily.

14.7 An NTP server B receives server A's message at 18:47:56.76, bearing a timestamp 18:46:33.46, and replies to it. A receives the message at 18:49:15.72, bearing B's timestamp, 18:48:25.72. Estimate the offset between B and A, and the accuracy of the estimate.

14.7 Ans.

Hint: To estimate the offset between B and A, the difference between the timestamp and receiving time of the message must be calculated.

14.8 How many modes do NTP servers use to synchronize with one another? Describe each mode.

14.8 Ans.

NTP servers synchronize with one another in three modes. They are multicast, procedure-call and symmetric mode.

Multicast mode is intended for use on a high-speed LAN. One or more servers periodically multicasts the time to the servers running in other computers connected by the LAN, which set their clocks assuming a small delay.

Procedure-call mode is similar to the operation of Cristian's algorithm, described above. In this mode, one server accepts requests from other computers, which it processes by replying with its timestamp (current clock reading).

Finally, symmetric mode is intended for use by the servers that supply time information in LANs and by the higher levels (lower strata) of the synchronization subnet, where the highest accuracies are to be achieved. A pair of servers operating in symmetric mode exchange messages bearing timing information.

14.9 Lamport stated that clocks cannot be synchronized perfectly across a distributed system. In accordance with this statement, what action can be taken?

14.9 Ans.

In general, we can use a scheme that is similar to physical causality, but that applies in distributed systems, to order some of the events that occur at different processes. This ordering is based on two simple and intuitively obvious points:

- If two events occurred at the same process p_i (i=1,2,…,N), then they occurred in the order in which p_i observes them.
- Whenever a message is sent between processes, the event of sending the message occurred before the event of receiving the message.

14.10 By considering a chain of zero or more messages connecting events e and e' and using induction, show that $e \rightarrow e' \Rightarrow L(e) < L(e')$.

14.10 Ans.

If e and e' are successive events occurring at the same process, or if there is a message m such that $e = send(m)$ and $e' = rcv(m)$, then the result is immediate from LC1 and LC2. Assume that the result to be proved is true for all pairs of events connected in a sequence of events (in which either *HB1* or *HB2* applies between each neighbouring pair) of length N or less (N Š 2). Now assume that e and e' are connected in a series of events $e_1, e_2, e_3, .., e_{N+1}$ occurring at one or more processes such that $e = e_1$ and $e' = e_{N+1}$. Then $e \rightarrow e_N$ and so C(e) < C(e_N) by the induction hypothesis. But by LC1 and LC2, C(e_N) <C(e'). Therefore C(e) < C(e').

14.11 What is total ordering of a logical clock? Explain with an example.

14.11 Ans.

Total ordering is a procedure by which, if two events occur at the same time, they can be put into an order. Let event *e1* occur at *t1* at *p1* (process 1) and event *e2* occur at *t2* at *p2* (process 2) and *e1* = *e2*. Then the process identifier of process *p1* and *p2* is used to represent the clock time, *e1.p1* and *e2.p2* where *e1* = *e2*.

14.12 In a similar fashion to Exercise 14.10, show that $e \rightarrow e' \Rightarrow V(e) < V(e')$.

14.12 Ans.

If e and e' are successive events occurring at the same process, or if there is a message m such that $e = send(m)$ and $e' = rcv(m)$, then the result follows from VC2–VC4. In the latter case the sender includes its timestamp value and the recipient increases its own vector clock entry; all of its other entries remain at least as great as those in the sender's timestamp.

Assume that the result to be proved is true for all pairs of events connected in a sequence of events (in which either *HB1* or *HB2* applies between each neighbouring pair) of length N or less (N Š 2). Now assume that e and e' are connected in a series of events $e_1, e_2, e_3, .., e_{N+1}$ occurring at one or more processes such that $e = e_1$ and $e' = e_{N+1}$. Then $e \rightarrow e_N$ and so V(e) < V(e_N) by the induction hypothesis. But by VC2–VC4, V(e_N) < V(e'). Therefore V(e) < V(e').

14.13 Using the result of Exercise 14.11, show that if events e and e' are concurrent then neither $V(e) \leq V(e')$ nor $V(e') \leq V(e)$. Hence show that if $V(e) < V(e')$ then ? $\rightarrow e$.

14.13 Ans.

Let e and e' be concurrent and let e occur at p_i and e' at p_j. Because the events are concurrent (not related by happened-before) we know that no message sent from p_i at or after event e has propagated its timestamp to p_j by the time e' occurs at p_j, and *vice versa*. By the reasoning for Exercise 10.11, it follows that $V_j[i] < V_i[i]$ and $V_i[j] < V_j[j]$ (strict inequalities) and therefore that neither $V(e) \leq V(e')$ nor $V(e') \leq V(e)$.

Therefore if $V(e) < V(e')$ the two events are not concurrent – they must be related by happened-before. Of the two possibilities, it obviously must be that $e \rightarrow e'$.

14.14 Two processes *P* and *Q* are connected in a ring using two channels, and they constantly rotate a message *m*. At any one time, there is only one copy of *m* in the system. Each process's state consists of the number of times it has received *m*, and *P* sends *m* first. At a certain point, *P* has the message and its state is 101. Immediately after sending *m*, *P* initiates the snapshot algorithm. Explain the operation of the algorithm in this case, giving the possible global state(s) reported by it.

14.14 Ans.

P sends msg m

P records state (101)

P sends marker (see initiation of algorithm described on p. 406)

Q receives m, making its state 102

Q receives the marker and by marker-receiving rule, records its state (102) and the state of the channel from P to Q as { }

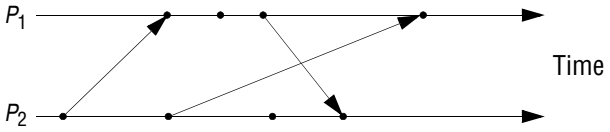
Q sends marker (marker-sending rule)

(Q sends m again at some point later)

P receives marker

P records the state of the channel from Q to P as set of messages received since it saved its state = { } (marker-receiving rule).

14.15

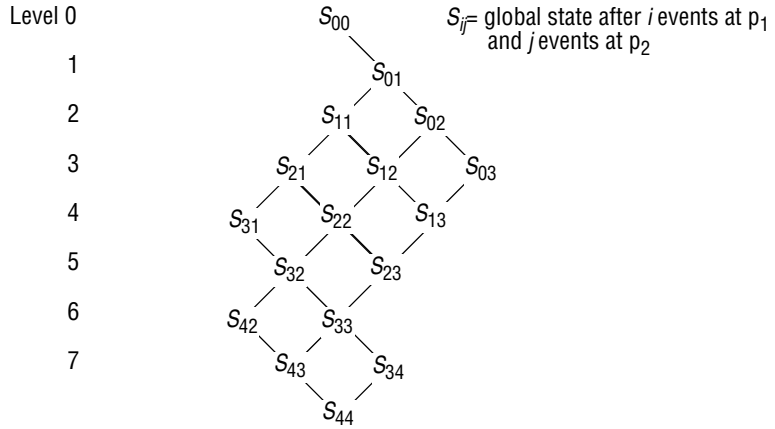


The figure above shows events occurring for each of two processes, p_1 and p_2. Arrows between processes denote message transmission.

Draw and label the lattice of consistent states (p_1 state, p_2 state), beginning with the initial state (0,0).

14.15 Ans.

The lattice of global states for the execution of Figure of exercise 11.15



14.16 Jones is running a collection of processes $p_1, p_2, \ldots, p_N$. Each process p_i contains a variable v_i. She wishes to determine whether all the variables $v_1, v_2, \ldots, v_N$ were ever equal in the course of the execution.

(i) Jones' processes run in a synchronous system. She uses a monitor process to determine whether the variables were ever equal. When should the application processes communicate with the monitor process, and what should their messages contain?

(ii) Explain the statement *possibly* ($v_1 = v_2 = \ldots = v_N$). How can Jones determine whether this statement is true of her execution?

14.16 Ans.

(i) communicate new value when local variable v_i changes;

with this value send: current time of day *C(e)* and vector timestamp *V(e)* of the event of the change, *e*.

(ii) *possibly* (...): there is a consistent, potentially simultaneous global state in which the given predicate is true.

Monitor process takes potentially simultaneous events which correspond to a consistent state, and checks predicate $v_1 = v_2 = \ldots = v_N$.

Simultaneous: estimate simultaneity using bound on clock synchronization and upper limit on message propagation time, comparing values of *C* (see p. 415).

Consistent state: check vector timestamps of all pairs of potentially simultaneous events e_i, e_j: check $V(e_i)[i] \geq V(e_j)[i]$.