IT UNIVERSITY OF COPENHAGEN

BMDS 2014

Designated Exercise Set 1

ASSIGNMENT 1

Anders Wind Steffensen - awis@itu.dk
Anders Fischer-Nielsen - afin@itu.dk
Christopher Blundell - cnbl@itu.dk
Daniel Varab - djam@itu.dk

September 21, 2014

CHAPTER 1

Answers

1.1 Exercise A.

A client attempts to synchronise 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)
(A) 22	10:54:23.674
(B) 25	10:54:25.450
(C) 20	10:54:28.342

A.1 - Which of these times should it use to set its clock?
Picking the time which had the fastest round trip (C) gives us a smaller margin for error. This means better accuracy. Therefore the answer is (A) 10:54:23.674. Since we do not know the distribution of the time from and to the server, the smaller the round trip, the smaller the difference is possible.

A.2 - To what time should it set it?
Since we picked C for setting the time we will use the result to calculate the time using the form time + (round trip/2)
With our data:

10:54:28:342+20/2 = 10:54:28:352
Therefore we should set the time to 10:54:28:352

A.3 - Estimate the accuracy of the setting with respect to the server's clock.
Since we add the roundtrip divided by two to the time, we have some inaccuracy. We are not sure if the first half of the roundtrip might take 0 ms or 20 and the trip back then respectively 20 or 0 ms. Therefore the accuracy of this clock may be

+-(TRound / 2 - min)

With our data (and with min removed since we do not know it):

+-(20ms / 2) = +-10ms

Therefore the accuracy of our clock will be +-10ms

A.4 - 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?

@A.1 - Even if we know that the round trip takes at least 8 ms, it still wouldn't make sense to pick a less accurate time such as (A) and (B).

@A.2 - The answer to our second question would not change either considering we don't change our answer to the first question.

@A.3 - The accuracy of (C) would change to 2 ms since we now know the minimum time we can use the fomular for accuracy one more time

+-(TRound / 2 - min)

With our data:

(20ms / 2 - 8) = +-2ms for (C)

Therefore with a minimum time of 8 ms and using the (C) roundtrip the accuracy of the clock would be 2ms.

1.2 Exercise B.

Exercise B. An NTP server B receives server A's message at 16:34:23.480 bearing a timestamp 16:34:13.430 and replies to it. A receives the message at 16:34:15.725, bearing B's timestamp 16:34:25.700 Estimate the offset between B and A and the accuracy of the estimate.

To begin with we are going to give the 4 events names:

T0 = 16:34:13.430 : the client's timestamp of the request packet transmission.

T1 = 16:34:23.480 : the server's timestamp of the request packet reception.

T2 = 16:34:25.700 : the server's timestamp of the response packet transmission.

T3 = 16:34:15.725 : the client's timestamp of the response packet reception.

Now that the values are named, we can use forms to get the offset round-trip delay and accuracy:

The offset is given by

((t1 - t0) + (t2 - t3))/2

With our data:

((00:00:10.050) + (00:00:10.025))/2 = 00:0010:037,5

Therefore the offset between the two clocks are 00:0010:037,5

The round-trip delay is computed as

(t3 - t0) - (t2 - t1)

With our data:

((00:00:02.295) - (00:00:02.220) = 00:00:00.075

Therefore the round-trip delay between the two clocks is 00:00:00.075

The accuracy is computed as

+-(TRound / 2 - min)

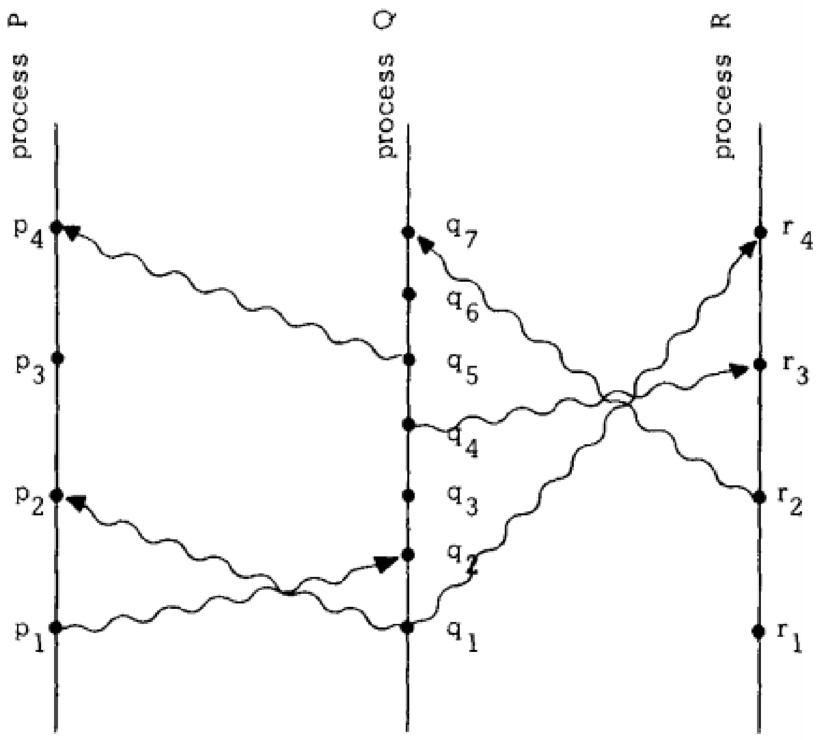
With our data, and since min is not known to us

00:00:00.075/2 = 00:00:00.037,5

Therefore the accuracy between the two clocks is 00:00:00.037,5

1.3 Exercise C.

Consider this diagram:



1. Write out the happens-before relation in the following diagram. (Immediate successors is fine; you don't have to write the entire transitive closure.
2. Identify 4 consistent cuts.
3. Identify 4 inconsistent cuts.
4. Write 2 different linearizations of the events in this diagram.

1) Happens before successors.

q1->q2->q3->q4->q5->q6->q7	HB statement 1
p1->p2->p3->p4	HB statement 1
r1->r2->r3->r4	HB statement 1
p1->q2	HB statement 2
q1->p2	HB statement 2
q1->r4	HB statement 2
q4->r3	HB statement 2
q5->p4	HB statement 2
r2->q7	HB statement 2

2) Consistent cuts

Process P	Procces Q	Process R
p1	q1	r1
p1-p3	q1-q4	r1-r4
p1	q1-q7	r1-r2
p1-p4	q1-q7	r1-r4

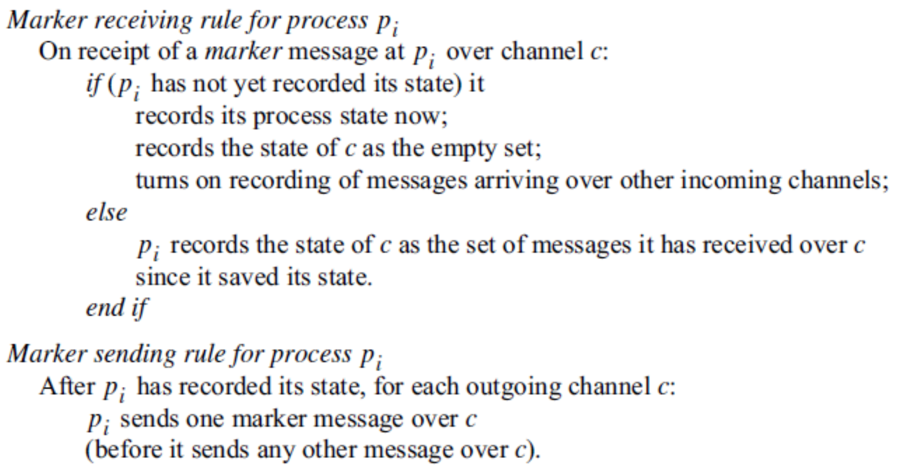
2) Inconsistent cuts

Process P	Procces Q	Process R	Explanation
p1-p4	q1	r1	q5->p4
p1-p3	q1-q2	r1-r4	q4->r3 , vr3-> r4
p1	q1-q7	r1	r2->q7
p1-2	none	r1-r4	q1->p2, q1->r4

1.4 Exercise D.

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.

The snapshot algorithm as described in *Distributed Systems — Concepts & Design* p. 616



```
Marker receiving rule for process $p_i$
  On receipt of a marker message at $p_i$ over channel $c$:
    if ($p_i$ has not yet recorded its state) it
        records its process state now;
        records the state of $c$ as the empty set;
        turns on recording of messages arriving over other incoming channels;
    else
        $p_i$ records the state of $c$ as the set of messages it has received over $c$
        since it saved its state.
    end if
Marker sending rule for process $p_i$
  After $p_i$ has recorded its state, for each outgoing channel $c$:
    $p_i$ sends one marker message over $c$
    (before it sends any other message over $c$).
```

Since P initiates the algorithm we look at the 'Marker sending rule for process' first, therefore P records its state to begin with, and since it has received the message M 101 times, this value will be stored. P only has one channel and therefore it sends a marker message along it to Q, and afterwards turns on recording of messages arriving over other incoming channels.

Q at this time receives M and therefore raises its state to 102 and sends the message on again. Shortly after Q receives the Marker message from P and therefore starts the "Marker receiving rule for process". Since it has not yet recorded its state it does so, and then records the incoming channel as empty. No other incoming channels exists so the algorithm is done.

1.5 Exercise E.

Consider this distributed system: There are 3 processes; each process has a state consisting of a single number k. Whenever a process receives a message, it updates its state k to a randomly chosen new number. If the state k of a process is even, it will transmit messages at random intervals; if odd, it does nothing.

Questions

1. Explain how this system may deadlock.
2. Explain how this system may deadlock even if you can make no assumptions on the initial state.
3. Explain how the snapshot algorithm can be used to discover if the system has deadlocked. (NB! Assume that a process in odd state will nonetheless send markers as required by the algorithm. Moreover, a process which receives a message containing only a marker does not update its state k.)
4. Summarise the steps of the snapshot algorithm when executed on the system with initial global states 3, 4, 7 and no messages in transit. Will the resulting snapshot allow you to conclude that the system is deadlocked?
5. Summarise the steps of the snapshot algorithm when executed on the system with initial global states 3, 5, 7 and no messages in transit. Will the resulting snapshot allow you to conclude that the system is deadlocked?

Answers

1. The system will deadlock if all processes are have k as an uneven number. If the system starts with every process with k equal to an uneven number, nothing will happen at all as the system will be locked, unable to retrieve.
2. The system can still freeze. Since every process gets a random number, all three processes might get an uneven number again. Imagine a given point in time where p1 = uneven, p2 = uneven, p3 even. P3 is sendings messages at an unknown interval, which pings p2 whichs random outcome sets its k-value to an even number. Now, simultaneously p3 and p2 send each other a message,

which unluckily both result in uneven k-evaluations, effectively deadlocking the system.

3. As the snapshot algorithm can be used to monitor the traffic of a distributed system one may inspect the states to disclose deadlocks. If processes are not omitting messages and the channels are empty it is safe to say that a deadlock has occurred.

4. *S0: p1:<3>, p2:<4>, p3<7>, c1:<>, c2:<>, c3:<>, c4:<>, c5:<>, c6:<>*
   As the snapshot algorithm is initiated on an empty system the first step is to emit the sending rule by sending a mark from p2 to its outgoing channels (p1 and p3) which are consequently pushed to record their state as a result of the receiving rule. Assuming for the sake of this question, p2 sends its message (m) to process p1 via channel c2 after its mark, p1 will set c2<m> as specified to the algorithm. At this point we are in the final global state which isn't a deadlock due to the factor of p2s continuation to send messages. This is visible through the snapshot by the state of c<m>.
   Final global state: p1:<3>, p2:<4>, p3<7>, c1:<>, c2:<m>, c3:<>, c4:<>, c5:<>, c6:<>

5. *S0 & final global state: p1:<3>, p2:<5>, p3<7>, c1:<>, c2:<>, c3:<>, c4:<>, c5:<>, c6:<>*
   Just as the previous answer, the initial marks will be broadcasted to the different processes, but in contrary there is no message to be sent and we find ourself in the definition of a deadlock: all processes are waiting. This is aligned with the snapshot as it shows that no messages are in any channels.