Study Guides on Computer Networks and Distributed Systems

Clock synchronization with Cristian Algorithm and NTP

Questionnaire

1. The Network Time Protocol (NTP) can be used to synchronize computer clocks. Explain why, even with this service, no bound is given for the resulting time difference between two clocks even right after synchronization.

   Computing the offset necessary to add to the requester clock (C_{Req}) which needs to be set to the responder clock (C_{Resp}) time entails determining $Delay_{Return}$ or the average of the return path delay over time. The approximation to $Delay_{Return}$ that we can compute is just $\frac{Rtt}{2}$, which, in reality can depart a lot from the actual $Delay_{Return}$ depending on the degree of asymmetry between the forward and the return paths and their respective delays, so the accuracy of this approximation can introduce a substantial error in the clock.

   Perfect synchronization would entail zero-Rtt which is impossible to achieve. You may want to skim the question about Cristian's method below.

2. What transport does NTP use?

   NTP is documented in RFC 5905. The transport used by NTP is UDP

3. When synchronizing computer clocks, can a clock be set back? Explain what has to be done in order for a clock to acquire the same real time as another clock.

   Computer clocks, in general, must not be brought back, since that might cause the repetition of certain actions configured by system administrators. The correct action to take upon a clock that must be brought behind consists in reducing its speed so that in a certain period of real time it will be in sync with a remote clock (One that, for some reason, has a good reference time). By decelerating the clock, it will continue generating a sequence of growing time values, though with a reduced accuracy. This so-called *clock monotonicity* will avoid the foregoing undesirable effects altogether. POSIX function adjtime() will accelerate or decelerate the clock to gain or lose some time monotonically.

4. Solve the synchronization example included in the lecture slides *(This is the solution of Nov-2019)*

- Clock A is assumed to present drift, consequently it has to be frequently synchronized with a quality clock source running at host B.

- The present exercise explains the algorithm executed by hosts A and B which allows A to synchronize its clock with that of B. Only one synchronization point is explained in this exercise. The considerations about the length to the next synchronization point are out of this exercise's scope.
- We assume no failure takes place on the communication channel, the processes bearing the synchronization at host A and at host B or the involved clocks.

The cells in green background represent the problem data that we are given and the gray-blue cells represent the calculations that we have to do in order to obtain the desired final result: the new time computer A must be set to so it be in sync with B's clock.

On the lectures we calculated the precision of Cristian clock synchronization method and established that it depends on the Rtt. We claim that the larger the Rtt, the less the precision we'll be able to achieve at any synchronization point. Cristian Algorithm is based on the claim that even though there exists a tradeoff between clock synchronization precision and Rtt, many repetitive clock readings of B can be accomplished by A and that that will allow A to improve the precision attained by discarding the readings having a large Rtt. The distribution of Rtt values *vs.* the number of sent messages is assumed to be long, thin-tailed.

On the following table, several time requests are undertaken by A by using the ICMP Timestamp Request/Response messages. Let's explain the meaning of each column by observing time request no. 3:

- **Originate** is the TS_{ORIG} included by A when it sent the ICMP Timestamp request to B
- **TimeOfDay** is A's clock reading at the time instant when it received the ICMP timestamp response from B
- **Receive TS** represents the timestamp applied by B when it received the timestamp request from A. Note, this timestamp is generated by clock B.
- **Transmit TS** represents the timestamp applied by B when it finished sending back the timestamp response to A.
- When A receives the response, it can calculate the observed Rtt by subtracting the Originate TS from the TimeOfDay. Check that the result (30 ms) is correct.
- A can also compute the residence time (Also known as Response Time) by computing:

  $T_{Residence}$:= Transmit TS – Receive TS

  This time represents the time taken by B in reading its clock and having it sent in the ICMP TS response.
- Host A can compute the Network Rtt (Rtt_N) by computing the difference between the overall Rtt and the residence time. Check that the residence time and the Rtt_N are both correct.

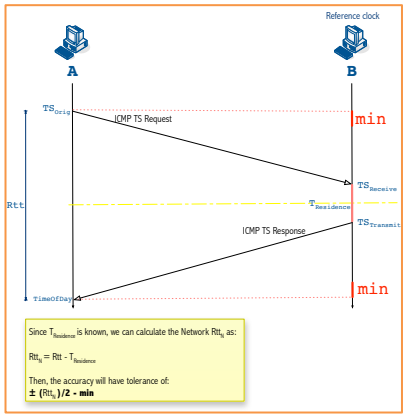


Figure 1. Synchronization point of host A clock with host B clock

After interpreting the contents of the timestamp table, we actually proceed to computing the time clock A must be set to in sync with clock B. According to the Cristian algorithm, we'll synchronize the A's clock with the data in the row having the least value of Rtt_N: row no. 7. The simple calculation process in included right below the table itself.

	Timestamps at host A		Timestamps at host B					
Request	Originate	TimeOfDay A	Receive TS	Transmit TS	Rtt (ms)	Residence time (ms)	RttN (ms)	
1	1:10:32.133	1:10:32.163	1:00:32.150	1:00:32.156	30	6	24	
2	1:10:34.025	1:10:34.056	1:00:34.040	1:00:34.045	31	5	26	
3	1:10:37.494	1:10:37.524	1:00:37.510	1:00:37.514	30	4	26	
4	1:10:40.305	1:10:40.334	1:00:40.320	1:00:40.326	29	6	23	
5	1:10:44.734	1:10:44.766	1:00:44.750	1:00:44.752	32	2	30	
6	1:10:49.005	1:10:49.039	1:00:49.020	1:00:49.027	34	7	27	
7	1:10:55.254	1:10:55.282	1:00:55.270	1:00:55.276	28	6	**22**	= 0,022 s
8	1:10:58.635	1:10:58.665	1:00:58.650	1:00:58.657	30	7	23	
9	1:11:02.325	1:11:02.357	1:01:02.340	1:01:02.344	32	4	28	
10	1:11:08.885	1:11:08.921	1:01:08.900	1:01:08.906	36	6	30	
			RttN/2 =	0,011 s				
		Target time for A in sync with B =		Transmit TS + RttN/2 =	1:00:55.276 + 0,011 =		1:00:55.287	
		Delta for adjtime() =		Target - TimeOfDay A =	1:00:55.287 - 1:10:55.282 =		-599,995 s	
		Delta for adjtime() =		-599,995 s				

Figure 2. Sequence of ICMP Timestamps for the synchronization of host A clock with host B's

5. Does TCP offer some facility to allow a transmitter to compute the *network Rtt* (Rtt_N) of a connection, *i.e.,* the Rtt that doesn't include the IRQ time (T_{RESP}) consumed by the destination host servicing the request received from the client.

   TCP's timestamps can only be used to calculate the *observed* Rtt, which includes T_{RESP}. In each segment sent, TCP includes the TSval timestamp, which represents the time of transmission according to the transmitter's monotonic clock (Not the *timeofday* clock). The receiving TCP will copy the received TSval in the TSecr field of the ACK segment. This mechanism based on TSval and its echo copy TSecr only permite the sending TCP to calculate the observed Rtt.

6. Describe the kind of operations caused by the reception of a layer-2 frame. Specifically we wish to obtain a perspective about what happens right after the microprocessor deals with the hardware interrupt which initially reports the full deserialization of a new frame.

   This exercise consists of having you review the lab practices of Computer Networks where you can find a detailed explanation of the encapsulation/multiplexing hierarchy over the TCP/IP protocol stack in an operating system like Linux.

7. Explain the basic principles underlying the Christian's synchronization algorithm

   Cristian's clock synchronization is based on a probabilistic algorithm. It assumes that messages sent onto the Internet may undergo *random unbounded delays* and that a host clock can not always be read with the desired precision, but that when it is read repeatedly one can achieve as good a precision as wished.

8. Clock synchronization over the internet suffers from a lack of precision, can you explain the cause?

   As mentioned in the preceding question, the precision in reading a host clock over the Internet depends on *random and unbounded message delays*. This inherent character of the delay is what

limits the precision in reading a clock and thereby on synchronizing another clock with it.

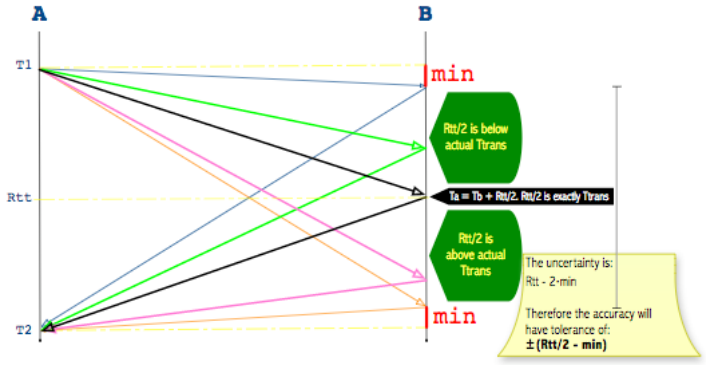
9. Tell several ways to have a host synchronized with a UTC source over the Internet

   By using the ICMP, NTP and SNTP protocols.

10. What is *clock drift?* What is its basic cause?

    Clock drift is a physical magnitude that represents the deviation of the real time offered by the clock *vs.* the actual real time. That means the clock speed varies over time. Particularly, the quartz-crystal based oscillators installed in most mainstream computer systems suffers from drift which has a heavy dependence on ambient temperature. Over time, drift causes the clock to skew from the actual real time which entails its resynchronization from a high quality reference clock.

11. Give an explanation of the following diagram:



    Compose the solution to this exercise on your own. You can skim your class notes and the presentation slides that we used in the lecture.

12. Study the solved exercise about Clock Synchro in paloalto.unileon.es/ds

    Refer to question no. 4 above, where you can find the problem solved by following the principles that we developed in the Lecture and in the Labs.

13. Solve exercises **14.1 – 14.5** from Dollimore/Kindberg/Coulouris/etc (You can access the problem statements on the last page of the Physical Clocks ppt lecture presentation)

    I will discuss these exercises and their solutions in the review class session that I am going to announce this week on the Agora Virtual Campus.

