Lecture on Sensor Networks

Synchronization in sensor networks

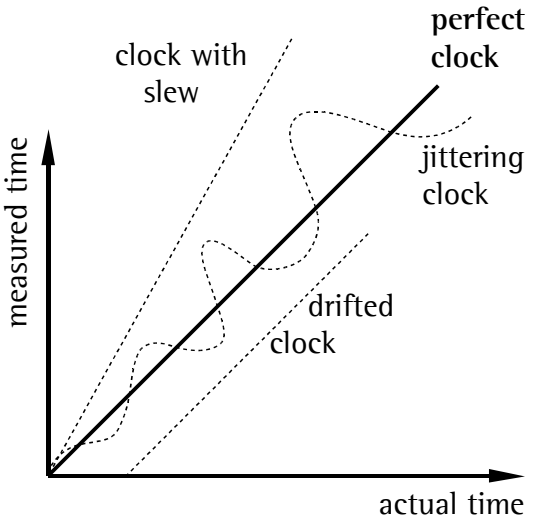
Motivation

Systems and metrics for time measurement

TAI:
Temps Atomique International is an averaged time from many cesium atomic clocks, distributed all over the world. The cesium atom changes its electric state 9.192.631.770 times per second. These changes of the state are counted in order to measure the duration of a second. Here, a deviation of about 10^{-12} per clock occurs - the time averaged over many clocks is more precise, respectively.

UTC:
Temps Universel Coordonné is the coordinated worldwide time, which is based on TAI. However, in particular years it should be adjusted by one leap second, because the tides slow down the Earth's rotation, the Moon moves away from Earth, the Sun becomes smaller etc.

NTP:
Network Time Protocol with the NTP time stamp, which consists of 64 Bits. The first 32 bits represent a seconds counter, from 1. January 0:00 o'clock in 1900, the last 32 bits indicate the split second, that means with accuracy of $1/2^{32}$ seconds.



Synchronization in sensor networks

Motivation

Trivial solution for synchronization

Requests are directed to a node, which is well synchronized. We adjust our own clock accordingly. If the expected duration of the message is short, particularly in comparison to the required accuracy of the clock, then one simple request will be enough. Otherwise, latencies between requester and the node with the reference clock can be subdivided as follows:

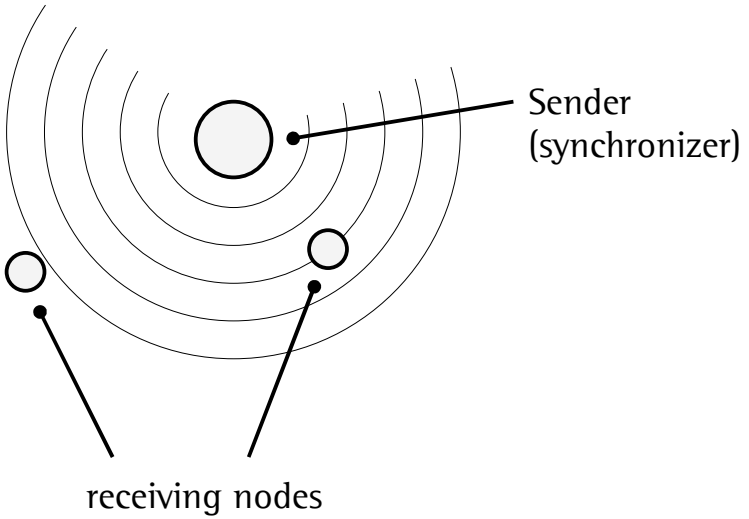
Time to send:	The time, which the sender requires to generate the message, a multitasking OS to get a time slice, to process further requests from other processes etc.
Access time:	The required time until access upon the medium can be allowed. (see CSMA method etc.) Problems occur with all methods which require a contention phase, as S-MAC, T-MAC etc.
Propagation time:	The time a message travels over the channel. Often negligible between senders, however, long latencies can accumulate over the network, especially with a great number of hops.
Receive time:	In analogy to the time for sending the message, i.e. the latency until the message reaches the application level.

Synchronization in sensor networks

Motivation

Reference broadcast synchronization (RBS)

Principle: Everyone gets the same sync signal over the broadcast channel. Despite the different signal propagation times for nodes with varying distances to the synchronizer every node is triggered at the same time. The sync signal does not have to be meaningful. A pulse or any kind of event can be used.



The signal propagation time is (almost) negligible. The source for variance in the measurement is the time different nodes need to receive the signal. All kinds of delays at the sender side do not play an important role since they are the same for every receiver.

Synchronization in sensor networks

Motivation

Reference broadcast synchronization (RBS)

When a sync signal is received every receiver stores its own local time. No synchronization took place so far. However, when exchanging timestamps with peer nodes next time the drift between clocks can be estimated.

Example: Node A receives a broadcast signal at 17:05. Later B tells A about an event that happened at 17:35 and adds in addition that the last sync signal was received at 17:02 according to B's clock. So A can conclude that its own clock is 3 minutes ahead of B and that the event took place at 17:38 according to its local time.

Alternatively, A could adjust its clock to be compliant with B if B's time is more credible.

Advantage: The broadcast signal can be any event which can be received by a group of nodes, e.g., even an RTS or CTS. But it has to be clear that everyone is talking about the same event when calculating the drift. An increasing counter for the synchronization can be helpful in this context.

Disadvantage: All receivers have to be in the same domain (listening to the same channel).

Synchronization in sensor networks

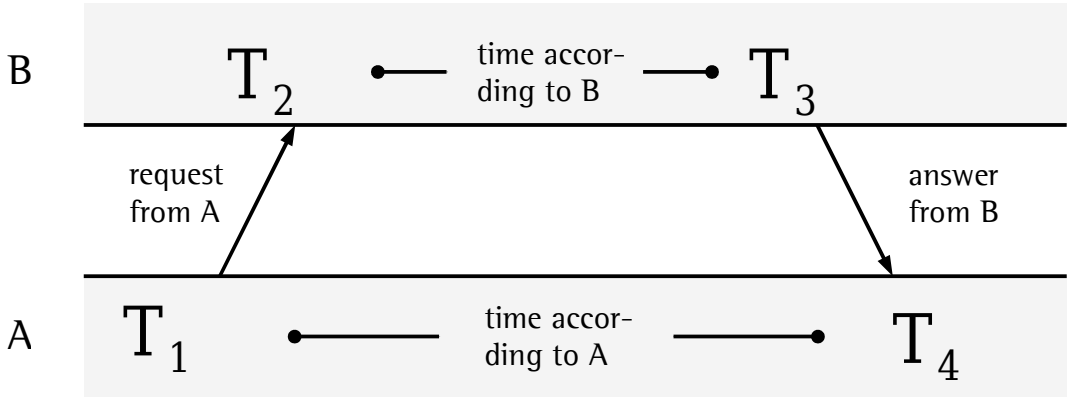
The Network Time Protocol (NTP) - Round Trip Time (RTT) based

Estimation of the latency delta (packet delay between two nodes) and the clock drift theta.

$$\delta = \frac{(T_4 - T_1) - (T_3 - T_2)}{2}$$

$$\theta = \frac{(T_2 - T_1) + (T_3 - T_4)}{2}$$

theta > 0 means that clock B is ahead of clock A



Synchronization in sensor networks

The Network Time Protocol (NTP)

Estimation of the round trip time delta and clock drift theta

$$\delta = \frac{(T_4 - T_1) - (T_3 - T_2)}{2} \qquad \theta = \frac{(T_2 - T_1) + (T_3 - T_4)}{2}$$

Verification: The clock shall deviate about e time units and the single packet delay (1 hop communication) is assumed to be d time units.

$$T_2 = T_1 + d + e$$
$$T_4 = T_3 + d - e$$

$$\delta = \frac{(T_3 + d - e - T_1) - (T_3 - (T_1 + d + e))}{2} = d$$

$$\theta = \frac{(T_1 + d + e - T_1) + (T_3 - (T_3 + d - e))}{2} = e$$

Synchronization in sensor networks

Error analysis of clock drift estimation between RTT and RBS

RTT based approach: The round trip time (time to send a packet and get an answer) is used to estimate the true delay of a simple one-way communication and the clock drift.

RBS based approach: A number of receivers are synchronized by the same event which is used to estimate the clock drift. Once the drift is known the packet delay can also be derived.

D_K = **Drift** between K's clock and real time

S_K = Time to **Send** (to prepare the packet at the sender's side, wait for empty queue etc. but not the time to access the medium).

A_K = Waiting time to gain **Access** to the medium (an issue of the MAC protocol)

$P_{K,L}$ = Signal **Propagation** time between node K and L

R_L = Time to **Receive** a packet (basically to hand it through the network stack to the application layer.

T_1 = The time as it is perceived by node 1.

t_1 = The actual realtime belonging to T_1. So the drift is obtained by T_1-t_1.

D_1 = Drift of node 1.

The length of the message itself and the time to send the entire message is not an issue of the above estimation. We only consider the arrival of the first bit.

Synchronization in sensor networks

Error analysis of clock drift estimation between RTT and RBS

Error in drift estimation of RTT based approaches:

$$T_1 = t_1 + D_A \qquad T_2 = T_1 + S_A + A_A + P_{A,B} + R_B$$
$$T_3 = t_3 + D_B \qquad T_4 = T_3 + S_B + A_B + P_{B,A} + R_A$$

Drift:
$$\theta_1 = \frac{(T_2 - T_1) + (T_3 - T_4)}{2}$$

Error in drift estimation: We substitute the measured times T with the actual and true real time t, however extended by all errors which can occur in the communication. In the end we want to know which kinds of errors influence the drift calculation most and which drop out.

$$e_{\theta_1} = \frac{(t_1 + D_A + S_A + A_A + P_{A,B} + R_B - (t_1 + D_A)) + (t_3 + D_B - (t_3 + D_B + S_B + A_B + P_{B,A} + R_A))}{2}$$

$$e_{\theta_1} = \frac{S_A + A_A + P_{A,B} + R_B - S_B - A_B - P_{B,A} - R_A}{2}$$

We can assume that the propagation $P_{A,B}$ from node A to B is approximately equal to the inverse path $P_{B,A}$ so that the propagation times compensate one another.

$$e_{\theta_1} = \frac{S_A + A_A + R_B - S_B - A_B - R_A}{2}$$

When nodes of the same kind are used the expectation values of the errors should have the same degree of magnitude. So the degree to which they cancel out one another depends on their variance. A smaller variance means better error cancellation.

Synchronization in sensor networks

Error analysis of clock drift estimation between RTT and RBS

Error in drift estimation of RBS based approaches:

$T_1 = t_1 + D_S$

$T_2 = T_1 + S_S + A_S + P_{S,A} + R_A$

$T_3 = T_1 + S_S + A_S + P_{S,B} + R_B$

Drift: $\theta_2 = T_2 - T_3$

Node S sends a sync signal at time T_1 (the time S measured with its clock). A receives the signal at T_2, B at T_3 (according to their local clock respectively).

Error in drift estimation:

$$e_{\theta_2} = T_2 - T_3 = t_1 + D_S + S_S + A_S + P_{S,A} + R_A - (t_1 + D_S + S_S + A_S + P_{S,B} + R_B)$$

$$e_{\theta_2} = P_{S,A} + R_A - P_{S,B} - R_B$$

As compared to the error in the RTT based approach:

$$e_{\theta_1} = \frac{S_A + A_A + R_B - S_B - A_B - R_A}{2}$$

All latencies at the sender's side cancel off themselves at the receiver's side. In contrast, the signal propagation time is not equal here anymore since the receivers may have different distances to the synchronizer. Furthermore, the term is not divided by 2. On the other hand many erroneous influences are missing and the signal propagation time is very small.

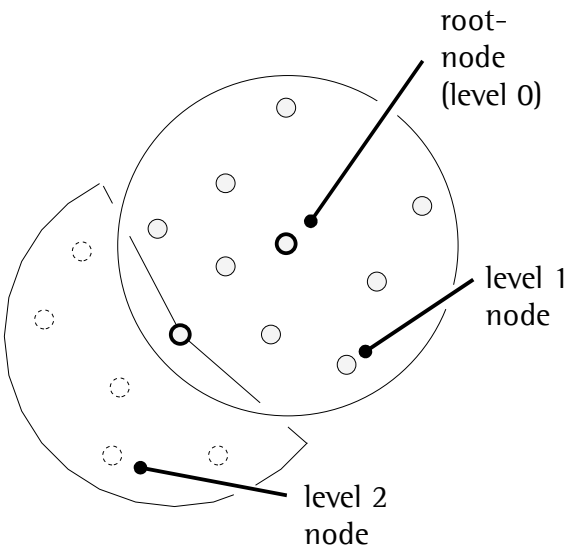
Synchronization in sensor networks

TPSN: Timing-sync Protocol for Sensor Networks

(according to the publication of S. Ganeriwal, R. Kumar, M. B. Srivastava)

Initialization phase

The initialization origins in TPSN from the highest level, which is the root node in the beginning. It sends a level_discovery packet including its own level.



All nodes within the range receive this announcement an adopt the level after increasing it by one. Then every node starts its own random timer.

After a timer's expiration a node will also send a level_discovery packet which is directed to all nodes in the range that did not yet hear an announcement. The others will ignore it. At the end of the flooding process every node has obtained a level which is increasing with an increasing number of hops to the root node.

Note that it is not guaranteed that a node has obtained its level because the level_discovery messages of more than one sender could have collided without the senders knowing it (remember the hidden station problem). A node with no valid level can send a level_request packet which is answered by everyone after the expiration of a random timer on the senders' sides. The requesting node will then choose the smallest level and increase it by 1.

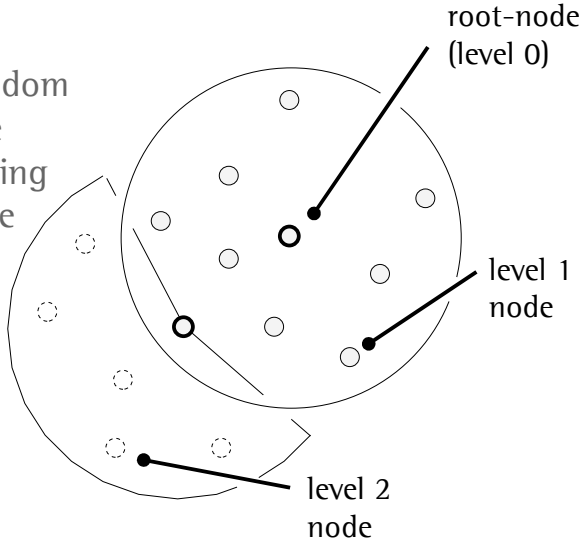
Synchronization in sensor networks

Motivation

Time-Sync Protocol for sensor Networks (TPSN)

Synchronization phase

The root node issues a time_sync packet which starts a random timer at the level 1 nodes. On expiration the node with the shortest random timer asks the root for synchronization using a synchronization pulse. Synchronization always takes place between a child and a parent node.



After the synchronization pulse arrives at the parent node it will reply with an ACK. Now the requesting node knows the round trip time and can thus estimate the single packet delay in the 1-hop communication.

The nodes on a higher level overhear the synchronization pulse for the time being and use it to start a random timer themselves. It has to be chosen large enough so that the synchronization between parent and grand parent nodes can be finished. On expiration of the first timer on the second level the resp. node will send a sync pulse to its father node and so on. Sync pulse and ACK have the same role as ping and pong in NTP.

The sync pulse has the side effect that a node can recognize if it has not received one and that child nodes may not be synchronized. In this case a parent node will simply send a time_sync packet itself (this is usually only done by the root) to trigger its children. The time_sync packet is however not necessary in the regular operation.

Synchronization in sensor networks

Estimation of time intervals in sensor networks

Precise synchronization and consistency of clocks are not always realistic in sensor networks due to the increased overhead or

- No sync source is available.
- Clocks are not precise due to large temperature deviations.
- Timing errors accumulate over several clocks.
- Clocks may not only experience a drift but also a so-called slew (clock to fast about a factor).

An event is detected in the time domain of the observer. Rather than trying to adapt it to a global time the inaccuracy is accepted. When telling another node N the event is simply transformed into the time domain of N. So the knowledge propagates through the local time domain of a number of hops.

Synchronization in sensor networks

Estimation of time intervals in sensor networks

Example An observer detects an event at 19:00 and tells it to B. B's clock shows 18:00. So B knows about the same event at approximately the same time but calls that time 18:00 rather than 19:00. Then the message is even propagated further to C which calls the time 19:00 again according to its clock. The fact the B called it 18:00 in between is not a problem.

Problem When forwarding the message between hops some additional time passes. These latencies have to be considered in the above example. One of the most important latencies is the packet delay which is often estimated as 0.5 x round trip time. Note that RTT means the way to another node and the way back. We do not know how much time is actually spent on a single hop. The lower bound is zero (unrealistic? -> find a better lower bound) or the whole RTT for one hop as upper bound.

The remaining uncertainty can be accounted for by defining an interval with a lower bound and an upper bound for the event .

Synchronization in sensor networks

Estimation of time intervals in sensor networks

Approach for estimating time intervals for events according to „Time Synchronization in Ad Hoc Networks“ by Kay Römer

Node 1:

$$[r_1, r_1] = [T_{Messung}, T_{Messung}]$$

Node 2:

$$\left[r_2 - (s_1 - r_1)\frac{1+p_2}{1-p_1} - \left(rtt_2 - idle_1 \frac{1-p_2}{1+p_1} \right), r_2 - (s_1 - r_1)\frac{1-p_2}{1+p_1} \right]$$

Exceptional: rtt_3 does not have to be transformed. Because it was measured by node 3 itself so it already is in the time domain of node 3.

Node 3:

$$\left[r_3 - (s_1 - r_1)\frac{1+p_3}{1-p_1} - (s_2 - r_2)\frac{1+p_3}{1-p_2} - \left(rtt_2 \frac{1+p_3}{1-p_2} - idle_1 \frac{1-p_3}{1+p_1} \right) - \left(rtt_3 - idle_2 \frac{1-p_3}{1+p_2} \right), \ldots \right.$$
$$\left. \ldots r_3 - (s_1 - r_1)\frac{1-p_3}{1+p_1} - (s_2 - r_2)\frac{1-p_3}{1+p_2} \right]$$

Node N:

$$\left[r_N - (1+p_N)\sum_{i=1}^{N-1} \frac{s_i - r_i}{1-p_i} - (1+p_N)\sum_{i=1}^{N-1} \frac{rtt_i}{1-p_i} + (1-p_N)\sum_{i=2}^{N} \frac{idle_i}{1+p_{i-1}} + rtt_N, r_N - (1-p_N)\sum_{i=1}^{N-1} \frac{s_i - r_i}{1+p_i} \right]$$

Synchronization in sensor networks

Estimation of time intervals in sensor networks

Node 2:

$$\left[r_2 - (s_1 - r_1)\frac{1+p_2}{1-p_1} - \left(rtt_2 - idle_1 \frac{1-p_2}{1+p_1} \right), r_2 - (s_1 - r_1)\frac{1-p_2}{1+p_1} \right]$$

In contrast to the lower bound the upper should remain as large (=as young) as possible. We do not want to subtract more than necessary. So we chose the lower estimate for the packet delay which is 0*[1]. The message's waiting time in node 1 should also be as small as possible which means a fast clock at node 1 and a slow one at node 2.

For this one: confer the explanation on the next but one slide. (rtt_2-$idle_1$ x fraction) stands for the round trip time between node 1 and 2 and exhibits an upper bound for the delay in a one-hop communication (the expectation value however, is RTT/2). The fraction is chosen such as to yield a large overall latency. Because the lower bound of the interval should become as small as possible. Therefore the clock of node 1 has to be as fast and the clock of node 2 has to be as slow as possible – to the maximum degree which is still permitted by the clock's known inaccuracies.

The waiting time the message had to pass at node 1 is based on the clock (or time domain) of node 1 which exhibits an inaccuracy of +/- p_1. So in the "slowest possible case" the clock is too slow by the factor ($1-p_1$) and in the "fastest" case it is too fast by the factor ($1+p_1$). The quotient results in the longest possible waiting time at node 1 – based on the time domain of node 2.

Node 1 also told us that it got to know about the event at r_1 and that it informed us at s_1. Note that both r_1 and s_1 are based on the (possibly inaccurate) clock of node 1. We could say that node 1 let the time span (s_1-r_1) pass (or whiled it away).

At r_2 node 2 gets to know about the event. So r_2 is a first estimate for the time of the event. In any case it can not have happened later than r_2.

*[1] Why can the packet delay simply be omitted? Because we only know the round trip time. However, we do not know how this overall RTT splits into the way to another node and the way back. The only obvious lower bound is that one way did take no time (even though this is not realistic) and the only obvious upper bound is that the packet took the full RTT for one way (leaving no time for the way back. Any better ideas?).

Synchronization in sensor networks

Estimation of time intervals in sensor networks

What's the matter with the calculation (rtt-idle)?

The receiver E wants to estimate an upper bound for the one-way packet delay between itself and the sender. Therefore, it could simply send the NTP typical ping and pong message. But this would require two additional packets which is not a good option in terms of energy efficiency.

Solution: Often E has sent S a message before for other reasons at time T_1. This message can in a sense already be interpreted as an NTP like ping message. Both E and S stored the local time when the message was sent (T_1) resp. when it was received (T_2).

Later at time T_3 node S sends a message to E. Prior to preparing the message it looks up in its neighborhood table when it received the last message from E, which is the time T_2 in our case. So it adds to the message that the idle period between itself and node E lasted for T_3-T_2 time units. This message is then received by E at time T_4. E also looks up in its table that it did communicate with S at last time at T_1. So the round trip time is (T_4-T_1)-(T_3-T_2). The process does not differ much from the NTP ping and pong messages.

Disadvantage: S and E have to do some simple book keeping for every neighbor they have. Furthermore a long idle period can introduce an additional error which can even be in the order of magnitude of the RTT itself for large idle times.

Synchronization in sensor networks

Estimation of time intervals in sensor networks

Comparison of time intervals

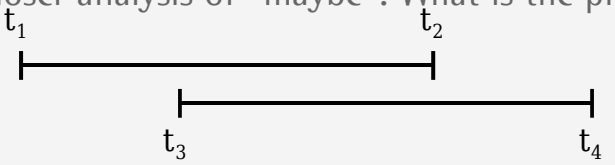
Points in time can be compared easily, e.g., to evaluate whether one event occurred before another. For intervals the comparison becomes slightly more complicated.

We assume that E_1 lies in the interval $[t_1,t_2]$, E_2 in the interval $[t_3,t_4]$.

Did E_1 happen before E_2?

yes	if $t_2 < t_3$
no	if $t_1 > t_4$
maybe	else

Closer analysis of “maybe”: What is the probability of E_1 happening before E_2?



We can distinguish three cases. The required order can be ensured if either $E_1 < t_3$ or $E_2 > t_2$. If both happen in the common time span $[t_3,t_2]$ there is a 50% chance of the required order.

Synchronization in sensor networks

Estimation of time intervals in sensor networks

Comparison of time intervals

But it is much easier to evaluate the inverse question: Is $E_1 > E_2$? This is the case if $E_1 > t_3$...

$$P(E_1>t_3)=\frac{t_2-t_3}{t_2-t_1}$$

... and $E_2 < t_2$:

$$P(E_2<t_2)=\frac{t_2-t_3}{t_4-t_3}$$

Since both events have to happen at the same time the product of the two probabilities have to be computed. Finally, if E_1 and E_2 are in the same interval there can be two different orders, but only one is valid in this context. Dividing by 2 takes this into account.

$$P(E_1>E_2)=\frac{(t_2-t_3)(t_2-t_3)}{2(t_2-t_1)(t_4-t_3)}$$

In the beginning we aimed at obtaining the probability of $E_1 < E_2$.

$$P(E_1<E_2)=1-P(E_1>E_2)$$

Synchronization in sensor networks

Estimation of time intervals in sensor networks

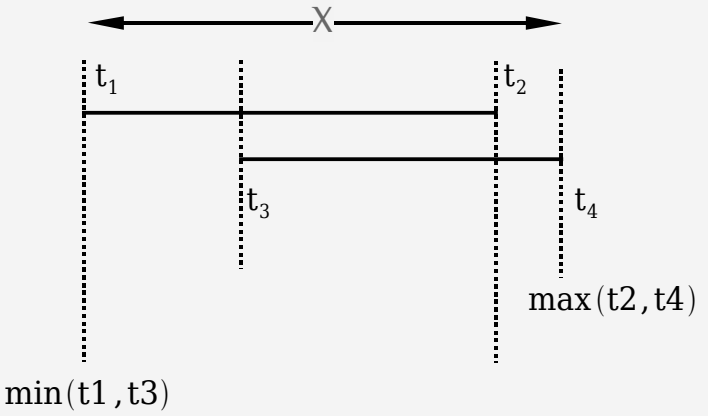
Comparison of time intervals

Are E_1 and E_2 further apart than X time units?

yes	if max(t4,t2) – min(t3,t1) < X (1-p)
no	if max(t3,t1) – min(t2,t4) > X (1+p)
maybe	else

If it is known in advance that the clock's deviation come from the interval [1-p,1+p] the real time X can be adjusted according to the question.

Case: Both events are less than X time units apart



Case: Both events are more than X time units apart