Probabilistic Internal Clock Synchronization[*]

Flaviu Cristian and Christof Fetzer
fetzer@christof.org
www.christof.org

May 1, 2003

Abstract

We propose an improved probabilistic method for reading remote clocks in systems subject to unbounded communication delays and use this method to design a family of fault-tolerant probabilistic internal clock synchronization protocols. The members of this family differ in the failure classes they tolerate, from crash to arbitrary. Because of probabilistic reading, our protocols achieve better synchronization precisions than those achievable by previously known deterministic algorithms. Another advantage of the proposed protocols is that they use a linear, instead of quadratic, number of messages, and that message exchanges are staggered in time instead of all happening in narrow synchronization intervals. The envelope and drift rates of the synchronized clocks are proven to be optimal.

1 Introduction

Most distributed systems encountered in practice are asynchronous, in that they do not guarantee a bound on message communication delays. Traditional deterministic, fault-tolerant clock synchronization algorithms such as those of [2, 3] assume bounded communication delays. Thus, they cannot be directly used to synchronize clocks in asynchronous systems. Moreover, these protocols typically require the transmission of at least N^2 messages each time N clocks are synchronized and all messages are exchanged in a bursty manner within a narrow re-synchronization real-time interval. This makes them difficult to scale to larger networks. Given the practical importance of large, asynchronous distributed systems, such as those based on Unix and similar systems, the idea of probabilistic clock synchronization was proposed in [1] as a means to synchronize clocks in the presence of unbounded communication delays. However, [1] has discussed the use of probabilistic

[*]An earlier version of this paper was published in the proceedings of the Thirteenth Symposium on Reliable Distributed Systems, Oct, 1994. This work was done at the Department of Computer Science & Engineering, University of California San Diego (UCSD), La Jolla, CA 92093−0114. It was partially supported by grants from the Air Force Office of Scientific Research, the German Academic Exchange Service (DAAD), the Powell Foundation, Sun Microsystems, and the Microelectronics Innovation and Computer Research Opportunities in California

remote clock reading only to achieve *external* clock synchronization. The goal of this paper is to show how one can use probabilistic remote clock reading to achieve fault-tolerant *internal* clock synchronization.

Probabilistic remote clock reading yields likely reading errors much smaller than classical, deterministic reading. The price for this increased precision is a small probability of failing to read a remote correct clock with a reading error specified in advance. Different correct time servers can experience such remote clock reading failures independently. This can cause different time servers to successfully read the values of different sets of correct clocks while attempting to achieve internal synchronization. Classical deterministic internal synchronization algorithms are not design to deal with such behavior, since they all rely on the hypothesis that each reading of a correct clock by a correct time server succeeds.

This paper proposes a family of fault-tolerant internal clock synchronization protocols which can successfully mask independent remote clock reading failures. The members of the family differ in the failure classes they tolerate, from crash to arbitrary. Because our protocols rely on a probabilistic remote clock reading method, they can be used to synchronize clocks despite unbounded communication delays. Since probabilistic reading achieves higher precisions than deterministic reading, the protocols achieve synchronization precisions better than those achievable by previously known deterministic algorithms, such as those of [2, 3, 7, 8]. Our algorithms use several new midpoint convergence functions, derived from the original fault-tolerant midpoint convergence function of [3]. These new convergence functions achieve optimal accuracy: the drift rate of the synchronized clocks is bounded by the maximum drift rate of correct hardware clocks. In contrast to clock synchronization algorithms which are based on statistical clock reading methods, such as [4], our algorithms allow a process to detect the possibility that its local clock might be out of synch with the other correct clocks.

Internal clock synchronization algorithms typically require that each time server process reads the clocks of all other time servers periodically. Therefore, their message complexity tends to be at least quadratic in the number of processes. Moreover, the processes send the synchronization messages in narrow real-time intervals, increasing the risk of congestion. We propose an improved probabilistic remote clock reading method which only requires to send a linear number of unreliable broadcast messages. Our method staggers the message traffic in time, thereby avoiding bursty message transfers. We further reduce the number of synchronization messages by proposing a new transitive remote clock reading method that allows a process to read the clock of a second process if it learns of the reading by a third process of the second process' clock. This transitive reading method allows us to decrease the number of messages down to $N + 1$ in the best case, where N is the number of time server processes.

2 System Model

We consider distributed systems consisting of nodes connected by a communication network. Nodes do not share any storage, but each node has access to a local hardware clock. The processes that run on network nodes communicate with each other only by exchanging messages or by observing the passage of time on their local clock. We assume that the network provides a datagram communication service, such as UDP, that allows any process to send unreliable broadcast messages. To synchronize its clock, each node runs a time server.

2.1 Communication

To synchronize their clocks, time servers must exchange messages. We assume that if a process q receives a message m from process p, m was indeed sent by p. Between the sending and the reception of a message there is an arbitrary, random, real-time delay. A positive lower bound min on this delay exists, because of the existence of a positive processing overhead at both the sender and the receiving nodes and the existence of a positive network signal propagation time. In principle the delay min can be computed by adding the times required to transmit an empty message between two neighboring network nodes in the absence of any other load or transmission failures. In practice, it is often the case that this bound is not known a priori and must be estimated using empirical methods.

The magnitude of actual communication delays depends on the amount of computation and communication going on in parallel in the network, on the possibility that transmission errors will cause messages to be retransmitted several times before being successfully received, and on other random conditions, such as the occurrence of page faults and process switches or the establishments of new communication routes. Measurement of delays in current distributed environments, indicate that their distribution has a typical shape resembling that illustrated in figure 1. This delay graph was experimentally measured on an Ethernet based cluster of Sun IPX workstations in our Dependable Systems Laboratory. The measurement is based on 200,000 round-trip message exchanges. Most messages arrive very fast: while the minimum round-trip delay observed was above 1,700 μs, the average round-trip delay was below 1,900 μs and more than 50% of round-trip messages returned within 1,750 μs. However, a few messages take a much longer time: the 99% timeout delay observed was about 4,175 μs. The maximum round trip delay observed in this experiment was about 156,000 μs. Even though for any finite experiment there will be a maximum delay, it is not clear how such experimentally observed maximum delays can be used to determine some real upper bound on delays for all possible experiments. Moreover, the significant gap that exists between any experimentally determined max and the minimum delay min make the bound max impractical for the purpose of synchronizing N clocks, since all deterministic synchronization algorithms that depend on an upper bound for the maximum communication delay cannot achieve a maximum deviations better than $(max - min)(1 - 1/N)$ [3]. In what follows, we do not assume any upper bound on communication delays.

2.2 Clocks

A hardware clock consists of an oscillator and a counting register that is incremented by the ticks of the oscillator. To simplify our presentation, we assume hardware clocks have a much higher resolution than the time intervals (such as process to process communication delays) that have to be measured. For example if the delays observable are of the order of milliseconds, we assume hardware clocks have a microsecond resolution. The hardware clock H_p of any process p is a total mapping from real-time $\mathcal{RT}$ to clock time $\mathcal{CT}$: $H_p(t)$ is the value displayed by the counter of the hardware clock of p at real-time t. We say that H_p is correct if it measures the passage of any real-time interval $[s, t]$ with an error of at most $\rho(t-s)$, where ρ is the maximum clock drift rate specified by the clock manufacturer:

(bounded drift) $$(1-\rho)(t-s) \leq H_p(t) - H_p(s) \leq (1+\rho)(t-s)$$

In the above formula, it is implicitly assumed that the delay $t - s$ is long enough so that the clock granularity is negligible compared to the maximum error caused by the drift of the clock. For most

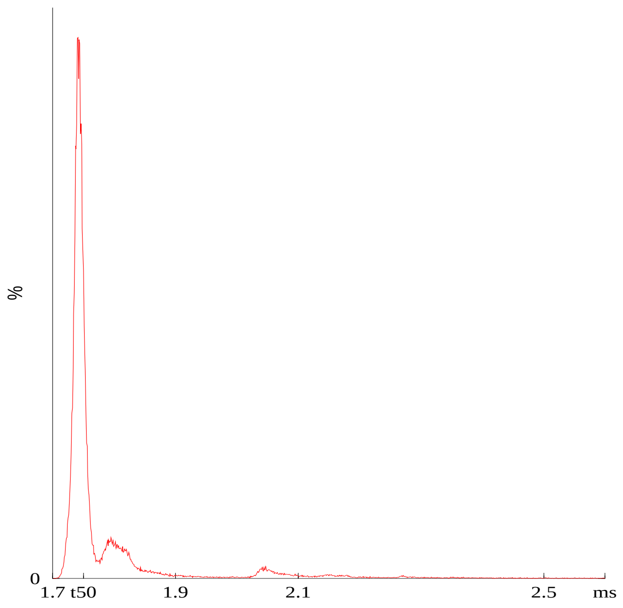

Figure 1: Transmission delay density

quartz clocks available in modern computers, the constant ρ is of the order of 10^{-5} or 10^{-6}, and for high precision quartz clocks ρ is of the order of 10^{-7} or 10^{-8}. Since ρ is such a small quantity, in this paper we ignore terms of the order of ρ^2 or smaller, for example we will equate $(1+\rho)^{-1}$ with $(1-\rho)$ and $(1-\rho)^{-1}$ with $(1+\rho)$.

A clock failure occurs if condition *bounded drift* is violated. We distinguish between two clock failure classes. A clock *crash* occurs when the counter of a clock stops progressing. Any other clock failure will be classified as *arbitrary*. Example of arbitrary clock failures are a clock that ticks faster or slower than allowed by the drift bound ρ or a clock that displays a non-monotonic sequence of time values because some of the counter bits are stuck at 0 or 1,

Time servers do not, in general, directly manipulate the value or the speed of hardware clocks. Instead, a time server maintains a *virtual clock* by adding to the underlying hardware clock an *adjustment function*. This function can be a step function of time, in which case the virtual clock is called *discrete*, or it can be a continuous function of time, in which case the clock is called *continuous*. For simplicity, we only consider discrete virtual clocks in this paper, knowing that we can substitute later continuous clocks to our discrete clocks without worsening the precision of the clock synchronization algorithms [6]. Like a hardware clock, a virtual clock C_p is a total mapping from real-time $\mathcal{RT}$ to clock time $\mathcal{CT}$. In what follows, we often refer to virtual clocks simply as clocks.

2.3 Processes

Time server processes undergo state transitions in response to message receptions and timeout events generated by local hardware clocks. The delay between the moment an event on which a process waits occurs and the moment the process is awaken will be referred to as the *process scheduling delay*. For a correct process, we assume the existence of an upper bound σ on the maximum scheduling delay. For example, if a correct process sets a timer at t to measure W time units, we assume the underlying operating system will awake the process in the real-time interval [t+(1-ρ)W,t+(1+ρ)W+σ].

Similarly, if a correct process waits for a message, we assume that it is awaken within σ real-time units of the message arrival.

We distinguish between three classes of process failures: crash, performance and arbitrary. A process suffers a *crash failure* when after a first failure to react to a trigger event it systematically omits to react to all subsequent trigger events until its restart. A process suffers a *performance failure* when it reacts too slowly to a trigger event, possibly because the underlying operating system exceeds the scheduling delay bound σ. All other possible process failures are classified as *arbitrary*. For example, a process that reacts too fast to a trigger event or undergoes erroneous state transitions, including the sending of conflicting information to other processes, will be classified as suffering an arbitrary failure.

We assume each process p checks that successive readings of its local clock yield different values: a clock crash will then result in a process crash. An arbitrary clock failure will result in an arbitrary process failure. A correct process has by definition a correct clock.

3 Remote Clock Reading

We first recall the basic idea of probabilistic remote clock reading and we describe its four characteristic properties. We then propose a new probabilistic clock reading method optimized for achieving internal clock synchronization. The main advantages of this method are its reduced number of messages and its increased probability of successfully reading a remote clock.

3.1 Probabilistic Remote Clock Reading

The basic idea behind probabilistic clock reading [1] is as follows. To read the clock of a remote process q, a process p must measure the round trip delay elapsed between the sending of the time request and the arrival of the reply on its local clock. Knowledge of the round trip delay allows p to *estimate* q's clock as well as to *bound* the error it makes in reading q's clock. If the bound on the reading error is smaller than or equal to a desired constant Λ, p declares success, otherwise it retries the remote clock reading procedure. We call constant Λ the *maximum acceptable clock reading error*. If the goal is to successfully read q's clock within at most D time units, p's probability of success can be made arbitrarily close to one by ensuring that p has time for a sufficient number of retries k before the D time units expire. If at the expiration of D time units no successful reading has occurred, a remote *clock reading failure* occurs.

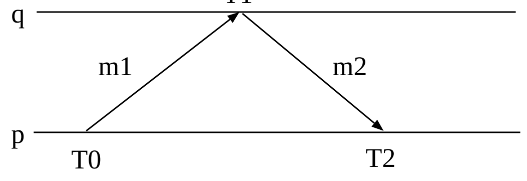

Figure 2: Probabilistic Clock Reading Method

For concreteness, let m_1 be the message that p sends and m_2 be q's reply containing q's current clock value (see figure 2). Let T_0 be p's clock value when m_1 is sent (figure 2), T_1 be q's clock value when m_2 is sent and T_2 be the time on p's clock when m_2 is received. Process p uses the measurement of the round trip delay (m_1, m_2) to calculate an upper bound for the clock reading error and to approximate $q's$ clock value at p's local time T_2 as follows. Process $p's$ clock can drift from real time by at most ρ. Thus, the round-trip delay is not greater than $(T_2 - T_0)(1 + \rho)$. This upper bound for the round-trip delay can be used to define an upper bound $max(m_2)$ for the one-way transmission delay $t(m_2)$ of message m_2:

$$max(m_2) \triangleq (T_2 - T_0)(1 + \rho) - min. \tag{1}$$

The condition $t(m_2) \leq max(m_2)$ holds, because $(T_2 - T_0)(1 + \rho) \geq t(m_1) + t(m_2)$ and $min \leq t(m_1)$, by definition of the minimum message transmission delay min. The lower bound min and the upper bound $max(m_2)$ of the message transmission delay $t(m_2)$ can now be used to approximate $q's$ clock at time T_2: $q's$'s clock value increases by $min(1-\rho)$ at the least and by $max(m_2)(1+\rho)$ at the most during the transmission of m_2. To minimize the worst case error that p makes in estimating q's clock, p's estimate of q's clock at T_2, denoted $\mathcal{C}_q(T_2, p)$, is defined as the midpoint of the interval $[T_1$+$min(1 - \rho)$,T_1+$max(m_2)(1 + \rho)]$:

$$\mathcal{C}_q(T_2, p) \triangleq T_1 + \frac{max(m_2)(1 + \rho) + min(1 - \rho)}{2} \tag{2}$$

This limits the worst case clock reading error $\mathcal{E}_q(T_2, p)$ that p can make in approximating q's clock at T_2 to half of the size of the above interval:

$$\mathcal{E}_q(T_2, p) \triangleq \frac{max(m_2)(1 + \rho) - min(1 - \rho)}{2} \tag{3}$$

Process p waits for a certain duration after sending a message and before sending the next message. When choosing this waiting time one has to consider that in some network protocol implementations, such as UDP on SunOS, messages can be coarsely classified as *cold* and *hot* messages. A cold message is one that follows a message that is sent a sufficient long time ago. A hot message is one that follows the previous message immediately or after a short delay. The transmission delay of a cold messages is substantially longer than the transmission delay of hot messages. Hence, a clock reading method wants to generate hot messages to decrease the clock reading error. For example, a clock reading method could use short waiting times so that almost every message is a hot message, or it could issue short bursts of request messages followed by a longer waiting time.

3.2 Characteristic Properties of a Probabilistic Reading Method

The goal of a probabilistic clock reading method is to allow a process p to estimate the clock of a remote process q with some known error. We denote Λ the maximum acceptable clock reading error, T_0 the time when p starts its attempt at estimating q's clock, t_0 the earliest point in real-time such that $C_p(t_0) = T_0$, D the maximum time p is willing to allow for its attempts at reading q's clock with the desired error, S a positive constant, and T any time between $T_0 + D$ and $T_0 + D + S$ by which p needs to have read q's clock with an error of at most Λ. Term t denotes the earliest point in real-time when process p's virtual clock shows T, i.e. $T = C_p(t)$. Probabilistic reading is required to yield p and estimate $\mathcal{C}_q(T, p)$ of q's clock as well as an error function $\mathcal{E}_q(T, p)$ such that the following conditions are satisfied.

- **(timeliness)** Remote clock reading takes at most D time units on p's clock.
- **(error bound)** If p or q do not suffer arbitrary failures between the start T_0 and the end T of remote reading, the reading error $\mathcal{E}_q(T,p)$ returned to p is a bound on the distance between the actual value of q's clock and the estimated value of q's clock:
$$|C_q(t) - \mathcal{C}_q(T,p)| \leq \mathcal{E}_q(T,p) \tag{4}$$
- **(crash handling)** If p is correct throughout the reading experiment $[t_0, t]$ but q crashed before t_0, the error is infinite $\mathcal{E}_q(T,p) = \infty$.
- **(likely success)** If p and q are both correct throughout the reading experiment $[t_0, t]$, the probability that the error $\mathcal{E}_q(T,p)$ is less than or equal to Λ is strictly positive:
$$prob(\mathcal{E}_q(T,p) \leq \Lambda) > 0. \tag{5}$$

The error bound condition implies that if processes p or q suffer a performance failure, $\mathcal{E}_q(T,p)$ still yields a bound of the clock reading error, although this might be infinite. It does not constrain the value of $\mathcal{E}_q(T,p)$ if one of the processes suffers an arbitrary failure. We require that if p crashes, the error bound condition should be true by definition. If q suffers a crash failure during probabilistic clock reading (after time t_0 and before time t) the error $\mathcal{E}_q(T,p)$ can be finite. The *error bound* condition is still valid, because C_q is totally defined and we assume that the drift rate of C_q is still bounded by ρ. Note that if two correct processes p and r approximate q's clock for time t and q crashes during these two readings, the distance between the two approximations is still bounded by $|\mathcal{C}_q(T_p,p) - \mathcal{C}_q(T_r,r)| \leq \mathcal{E}_q(T_p,p) + \mathcal{E}_q(T_r,r)$, where $T_p = C_p(t)$ and $T_r = C_r(t)$. This property is of importance for fault-tolerant internal clock synchronization: it allows correct processes p and q to approximately agree on $q's$ clock value even when q crashes.

The *likely success* condition forbids a probabilistic clock reading method from always delivering an infinite remote clock error bound. Remote clock reading methods which guarantee that $Prob(\mathcal{E}_q(T,p) \leq \Lambda) = 1$ are *deterministic* [1]. A remote clock reading method, such as [4], that does not provide the reading process p with a bound on the clock reading error is *statistical*. The advantage of probabilistic clock reading compared to deterministic clock reading is the possibility to achieve much smaller clock reading errors. The advantage with respect to statistical clock reading is that a process p *knows* when its clock is within a specified constant maximum deviation from other correct clocks.

The *timeliness* property implies that a correct process that invokes a probabilistic remote clock reading method at local time T_0 is guaranteed a return before its clock reads $T_0 + D$. This condition cannot be guaranteed in non real-time systems in general. Nevertheless, timeliness can be closely approximated in a system with a non real-time scheduler. A performance failure with respect to the timeliness condition can be detected and can be transformed - in the worst case - into a process crash failure.

3.3 Probabilistic Reading Method for Internal Clock Synchronization

Internal clock synchronization algorithms typically use a round-based approach to synchronize clocks. Before the end of every clock synchronization round, each process reads all other remote clocks at about the same time. The new probabilistic clock reading method optimizes the reading of remote clocks by decreasing the number of messages exchanged and by increasing the probability of reading all remote clocks successfully.

We use several techniques that optimize probabilistic clock reading for internal clock synchronization. First, given the popularity of local area networks that allow broadcasting, we use broadcast messages whenever possible, to reduce the number of messages exchanged in a round. In what follows processes send unreliable broadcast messages, but they could also use a sequence of (unreliable) unicast messages to emulate a broadcast message. Second, we use all potentially non-concurrent message pairs between two processes p and q to approximate $p's$ or $q's$ clocks. We use a heuristic that selects the approximately fastest message from p to q and from q to p. Hence, we select from all message pairs between two processes the message pair that provides approximately the lowest upper bound for the clock reading error. Third, the processes *stagger* the sending of messages in time, to reduce network congestion and message concurrency. Since staggering increases the number of non-concurrent messages, the probability of successfully reading a remote clock increases. Moreover, if the failure assumptions exclude arbitrary failures, any process p can make use of *transitive remote clock reading:* p can use process $q's$ remote clock approximations to obtain its own estimates of other remote clocks. In the best case, this allows all processes to read all remote clocks with a total number of $|\mathcal{P}|+1$ messages, where $\mathcal{P}$ denotes the set of all time server processes. Since the maximum number of messages sent per process and round is limited by k, our probabilistic clock reading method uses at most $k|\mathcal{P}|$ messages per round to perform $(|\mathcal{P}|-1)^2$ remote clock readings.

3.3.1 Staggering Messages

In round-based internal clock synchronization, processes determine the start and end of a synchronization round on their local clocks. Let T_0 denote the start of the first round and P the duration of a round. The i-th round starts at local time $T_0 + iP$ and must end by local time $T_0 + (i+1)P$ with all processes having read all other remote clocks.

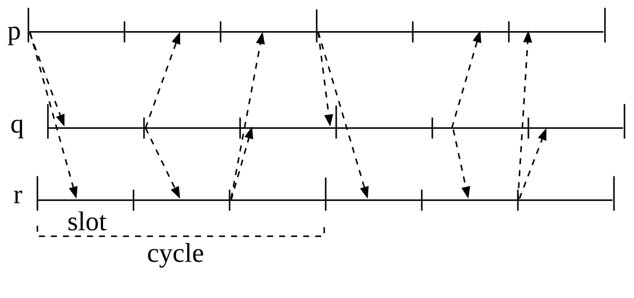

Figure 3: Staggering of messages

The messages exchanged in a round is staggered by dividing a round into k cycles and each cycle into $|\mathcal{P}|$ slots (see figure 3). Exactly one slot is assigned to each process per cycle. The size Z of a slot should be large enough so that all messages sent by a process p to read remote clocks are delivered to the other processes in the same slot with high probability. Every process in $\mathcal{P}$ has a unique rank. The rank of process p is denoted by $rank(p)$ and has a value between 0 and $|\mathcal{P}|-1$. The slot with number $rank(p)$ is assigned to process p. Because there are k cycles per round, each process can send up to k messages per round. Let T denote the end of the i-th round, $N \triangleq |\mathcal{P}|$, and

U denote the duration of a cycle, i.e. $U \triangleq ZN$. The first slot of the first cycle in round i starts at local time $T - kU$, and therefore a process p can send its first message at time $T - kU + rank(p)Z$.

3.3.2 Approximating Remote Clocks

A process executing the proposed probabilistic clock reading method sends a part of its current state with every message. Each message contains among other things the round number, the send timestamp, and the current approximations and error bounds of the sender for all remote clocks. Every sender timestamps every message m with a send timestamp $S(m)$. Every receiver q timestamps any received message m with a receive timestamp $R(m)$ (see figure 4). A process p records the send and receive timestamps of all messages sent and received in a round. Hence, if process q receives a message m from p, q gets all receive timestamps belonging to messages which q has sent and p has received before p sent m. This allows processes to have a partial knowledge of the receive timestamps of their own messages.

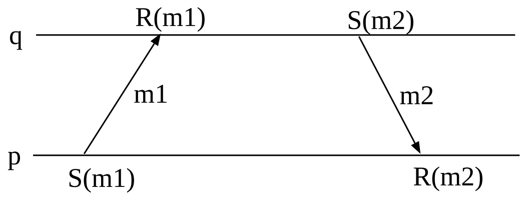

Figure 4: Improved Probabilistic Clock Reading

A process p uses the send and receive timestamps $S(m)$ and $R(m)$ to approximate the clock of any other process q (see figure 4). Let m_1 be a message sent by process p and let m_2 be a message sent by q. The message pair (m_1,m_2) is called *non-concurrent* if either q receives m_1 before it sends m_2 or p receives m_2 before it sends m_1, i.e.

$$R(m_1) \leq S(m_2) \vee R(m_2) \leq S(m_1)$$

To bound the transmission delay $t(m_2)$ of m_2, process p can use any non-concurrent message pair (m_1, m_2) sent between p and q. The bound $max^{m_1}(m_2)$ on the transmission delay of m_2 for a non-concurrent message pair (m_1, m_2) is given by

$$max^{m_1}(m_2) = \begin{cases} (R(m_2) - S(m_1))(1+\rho) - (S(m_2) - R(m_1))(1-\rho) - min & if\ R(m_1) \leq S(m_2) \\ (R(m_1) - S(m_2))(1+\rho) - (S(m_1) - R(m_2))(1-\rho) - min & if\ R(m_2) \leq S(m_1) \end{cases} \quad (6)$$

When $R(m_1) \leq S(m_2)$ is true, $t(m_2) \leq max^{m_1}(m_2)$ holds because the duration of the round-trip (m_1, m_2) is bounded by $(R(m_2) - S(m_1))(1+\rho)$ and we can subtract the time q waits before it sends m_2 (at least $(S(m_2) - R(m_1))(1-\rho)$) and the transmission delay of m_1 (at least min). Process p can use $max^{m_1}(m_2)$ to approximate $q's$ clock at time T (denoted by $\mathcal{C}_q^{m_1,m_2}(T,p)$) and to bound the clock reading error (denoted by $\mathcal{E}_q^{m_1,m_2}(T,p)$):

$$\mathcal{C}_q^{m_1,m_2}(T,p) = T - R(m_2) + S(m_2) + \frac{max^{m_1}(m_2)(1+\rho) + min(1-\rho)}{2} \quad (7)$$

$$\mathcal{E}_q^{m_1,m_2}(T,p) = 2\rho|T - R(m_2)| + \frac{max^{m_1}(m_2)(1+\rho) - min(1-\rho)}{2} \quad (8)$$

The idea behind these definitions is similar to that for equations (2) and (3). Process p bounds the transmission delay of m_2 by $max(m_2)$ and min and uses the average of these values as an estimate for $t(m_2)$ to minimize the clock reading error. Because p has to add $T - R(m_2)$ to adjust its view of $q's$ clock for time T, the bound on the clock reading error increases by $2\rho|T - R(m_2)|$ (since the clocks of p and q can drift apart with a rate of up to 2ρ).

If each pair of processes p and q sends up to k messages to each other, the number of message pairs that are exchanged by them estimating each other's clock can be as great as k^2. We derive now a way to reduce the number of message pairs that have to be considered for the approximation of a remote clock. Consider the case when process p sends message m_a and m_b to process q, and q replies with message m_c. The difference between the bounds for the clock reading errors is approximately $\mathcal{E}_q^{m_a,m_c}(T) - \mathcal{E}_q^{m_b,m_c}(T) \approx (R(m_a) - S(m_a)) - (R(m_b) - S(m_b))$ (derived by discarding terms with factor ρ in (8)). Therefore, we only consider the non-concurrent message pair (m_i, m_j) with $R(m_i) - S(m_i)$ and $R(m_j) - S(m_j)$ minimum. This allows us to select the message pair with the approximately smallest error bound without calculating the bound for the clock reading error for all message pairs.

Let us now derive a bound for the size Z of a slot. The slot size should be chosen so that a slot is large enough to ensure that a message broadcast to read remote clocks at the beginning of the slot is delivered during the same slot. If a process p schedules to send message m for time T_1, p cannot assume that m is sent exactly at time T_1. The slot size should reflect this variation in scheduling by the addition of the maximum scheduling delay σ. Since the maximum deviation between clocks is bounded by δ, the difference between the start of a round seen by two different processes is also bounded by δ. Hence, δ should also be added to the size of a slot. For $\mathcal{E}_q^{m_1,m_2}(T,p) \leq \Lambda$ to be true, the transmission delay of $t(m_2)$ has to be smaller than $2\Lambda + min$. Thus, we must chose the size of a slot so that $Z \geq \delta + 2\Lambda + min + \sigma$ holds.

We now propose a new *transitive remote clock reading method* to reduce the number of messages exchanged for reading remote clocks. The new method (see figure 5) allows a process q to estimate the clock of another process r when q learns of the approximations of r's and q's clocks by a third process p. Transitive reading reduces the number of messages per round to $N + 1$ in the best case and decreases the probability of clock reading failures.

To explain how transitive reading works, let us assume that p, q, r are correct processes. Assume that p has approximations of the clocks of r and q at real time t_p such that $C_p(t_p) = T$ and q has learned of $\mathcal{C}_r(T,p), \mathcal{E}_r(T,p)$, $\mathcal{C}_q(T,p), \mathcal{E}_q(T,p)$. The goal of transitive reading is to allow q to compute an estimate $\mathcal{C}_r(T,q)$ of r's clock at time $T = C_q(t_q)$ and bound the maximum error that it makes by $\mathcal{E}_r(T,q)$.

The remote clock estimate $\mathcal{C}_r(T,q)$ is determined as follows. In general, the real times t_p and t_q at which the clocks of p and q display local time T are different (see figure 5). To account for the time t_q-t_p elapsed between p's reading of r's clock and q's estimate of r's clock, q adds to p's estimate $\mathcal{C}_r(T,p)$ the time $T - \mathcal{C}_q(T,p)$ that approximates t_q-t_p.

$$\mathcal{C}_r(T,q) \triangleq \mathcal{C}_r(T,p) + T - \mathcal{C}_q(T,p) \tag{9}$$

Given that the clocks of p and q are at most δ apart, this estimate yields an error that is bounded by the sum of the errors $\mathcal{E}_r(T,p)$ and $\mathcal{E}_q(T,p)$ that p has made in reading the clocks of r and q, respectively, plus the error $2\rho\delta$, which must be added because the clocks of p and q can drift apart during time $|t_p - t_q|$ by at most $2\rho\delta$.

$$\mathcal{E}_r(T,q) \triangleq \mathcal{E}_r(T,p) + \mathcal{E}_q(T,p) + 2\rho\delta \tag{10}$$

The proof of Theorem (T1) in the Appendix shows that (10) is an upper bound on the error made by using estimate (9). Note that this kind of transitive knowledge cannot be used when arbitrary failures can occur, since a process suffering an arbitrary failure could send erroneous remote clock approximations to other processes.

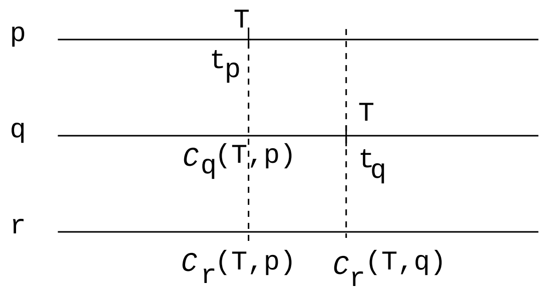

Figure 5: Transitive remote clock reading mechanism.

3.3.3 Round Message Exchange Protocol

A process executing the proposed probabilistic clock reading method can be in three *modes*: request, reply and finish. At the start of a new round a process is in *request* mode. A process p stays in request mode for as long as it has not yet successfully read all other remote clocks. While in request mode, p sends in each of its time slots a *request message* containing its current approximations and error bounds for all other clocks. Each process p that receives a request message checks if the sender q has successfully read $p's$ clock. If q has not done this, p has to send a message in its next slot. When a process has successfully read all remote clocks it changes its mode to reply or finish mode. A process p stays in *reply* mode for as long as it is not certain that all other processes have successfully read p's clock or until p changes to finish mode. While in reply mode, p will only send messages in reply to clock reading requests it receives from other processes.

A process p switches to *finish* mode when p has read all remote clocks successfully and p knows that all remote processes can use its remote clock approximations and error bounds to read in their turn all remote clocks successfully. Thus, any two error bounds eb_1 and eb_2 calculated by p must satisfy $eb_1 + eb_2 + 2\rho\delta \leq \Lambda$ before p can switch to finish mode. When p sends a finish message m, any receiver q can use m to get $p's$ approximation of any clock in $\mathcal{P}$. A process that sends or receives a finish message in a round normally stops sending further messages in that round. However, since we do not assume a reliable broadcast, it is possible for some processes to fail to receive a finish message. Thus, a process must be prepared to reply to request messages even when it receive them after sending or receiving a finish message.

When a process p receives a message m from process q, it locally stores the send and receive timestamps of m. Furthermore, p also extracts all receive time stamps belonging to messages sent by p

and stores the fastest message pairs between p and any other process observed so far. If the message pair for process q is changed to (m_1, m_2) by the reception of message m, p recalculates its best approximation and error bound of $q's$ clock to $\mathcal{C}_q^{m_1,m_2}(T)$ and $\mathcal{E}_q^{m_1,m_2}(T)$, where T is the end time of the current round. If the failure hypotheses excludes arbitrary failures, p uses the remote approximations of q to improve its own error bounds (by use of equations (9),(10)). If m is a finish message, p can use m to read all clocks successfully.

At the end of a round process p automatically switches to request mode. The probabilistic clock reading procedure executed by process p returns a result to its client when p switches to reply or finish mode or after the last slot in the last cycle of the current round has elapsed. The results returned by this procedure consist of the approximations and error bounds determined from the fastest message pair between p and each other process observed in the current round. In this way, the remote clock reading procedure guarantees the timeliness and the bounded error conditions. The lower bound for clock reading errors achievable by probabilist reading is $3\rho min$ [1]. If the maximum reading error is chosen so that $\Lambda > 3\rho min + 2\rho(D+S)$, k is chosen to be at least 2 and Z is chosen greater than $\delta + 2\Lambda + min + \sigma$, the likely success condition is also satisfied.

The message complexity of the round protocol described above is kN messages in the worst case, since each process can in the worst case send a message in each of its k slots of a round. In the best case, every process sends one message in the first cycle and the process with rank zero switches to finish mode after receiving the messages sent by all other processes in the first cycle. This process will send a finish message and all other processes will stop sending messages. Thus, the best case message complexity is $N + 1$.

4 Internal Clock Synchronization

4.1 Requirements

The main goal of a clock synchronization algorithm is to bound the distance between virtual clocks of correct time server processes by a constant δ (*maximum deviation*). If t^0 denotes the earliest point in real time for which virtual clocks must be synchronized, this requirement can be expressed as

(**bounded deviation**) *For any processes p, q correct at time $t \geq t^0$:*

$$|C_p(t) - C_q(t)| \leq \delta.$$

A second requirement is that the drift rate of a correct virtual clock be bounded by a constant $\rho_v < 1$. Synchronization algorithms for which $\rho_v = \rho$ are called *optimal* [9] If G denotes the maximum discontinuity of virtual clocks, this second requirement is expressed as:

(**clock drift**) *For any process p that is correct in interval $[t, u]$, $t^0 \leq t \leq u$:*

$$(1-\rho_v)(u-t) - G \leq C_p(u) - C_p(t) \leq (1+\rho_v)(u-t) + G.$$

A clock synchronization algorithm is *correct* when it satisfies the *bounded deviation* and *clock drift* requirements. A consequence of the *clock drift* requirement is the *linear envelope* condition: correct virtual clocks are within a linear envelope of real time.

(**linear envelope**) *For any process p that is correct in interval $[t^0, t]$, $t^0 \leq t$:*

$$(1-\rho_e)(t-t^0) - R \leq C_p(t) - C_p(t^0) \leq (1+\rho_e)(t-t^0) + R$$

```
ClockValue     A;                          // current adjustment value of local virtual clock
ClockValue     T;                          // end of current round

void $init$() {                            // schedule synchronizer
    A,T = InitialAdjustement();            // determine initial adjustment and end of first round
    schedule (/*function*/ $synchronizer$, /*every*/ $P$, /*start at*/ $T$);
                                           // schedule $synchronizer$ process
}

void $synchronizer$() {                    // called every $P$
    ClockValue Clocks[$N$];                // stores approximations for all $N$ round clocks
    ClockValue Errors[$N$];                // stores error bounds for all $N$ round clocks

    ReadClocks(Clocks, Errors);            // get approximations from probabilistic reading method
    A = A + cfn(rank(), Clocks, Errors) - T;
                                           // calculate adjustment for next round
    $T = T + P$;                           // set $T$ to end of next round
}
```

Figure 6: Clock Synchronization Protocol

The envelope rate ρ_e is smaller than or equal to the drift rate ρ_v of virtual clocks. A synchronization algorithm has an optimal envelope rate if ρ_e is not greater than the maximum hardware clock drift rate ρ.

4.2 Outline of Algorithms

A pseudocode description of a clock synchronization algorithm is given in figure 6. Variable T denotes the time at which the remote clocks are read probabilistically by another thread in process p and variable T also refers to the time at which the procedure $synchronizer$ is scheduled for execution. Time server process p executes the *synchronizer* procedure about every P clock time units. The procedure *ReadClocks* only fetches the approximations and error bounds provided by the parallel thread which executes the round message exchange protocol described earlier. The value of a virtual clock is determined as the sum of the hardware clock and a periodically computed *adjustment value* A. The value T of the virtual clock is replaced by the value computed by the *convergence function*, i.e. $cfn(rank(), Clocks, Errors)$. A new round starts at the time process p changes its adjustment value A.

To analyze our algorithms, we use the following notation: t_p^k denotes the start of $p's$ k-th synchronization round. Time t^k is the real time when all correct processes have just started their k-th synchronization round: $t^k = max\{t_p^k \mid p\ correct\ at\ time\ t_p^k\}$ (t^k is called the start of the k-th synchronization round). For every round k and every correct process p, the clock synchronization algorithm defines a new adjustment value which we denote by A_p^k. A virtual clock in round k is defined by the hardware clock and the adjustment value A_p^k:

$$C_p(t) \triangleq H_p(t) + A_p^k \; when \; t_p^k \leq t < t_p^{k+1}.$$

At the end of every round all correct processes try to estimate the values of all clocks. The rounds can overlap, i.e. a process can start round $k+1$ while another process is still in round k. Process p tries to approximate the remote virtual clocks with respect to the adjustment values of round k and not with respect to the adjustment values of round $k+1$. Therefore, we define the concept of a round clock. The *round clock* C_p^k of process p for round k is defined by

$$C_p^k(t) \triangleq H_p(t) + A_p^k.$$

Our clock synchronization algorithms use the improved probabilistic clock reading method to read remote round clocks (see section 3.3). Let T_p^{k+1} denote the end of round $k+1$ with respect to clock C_p^k, i.e. $T_p^{k+1} \triangleq C_p^k(t_p^{k+1})$. We assume that the reading error of a local clock is negligible, i.e. $\mathcal{E}_p^k(T_p^{k+1}, p) = 0$.

4.3 Assumptions

To prove that the proposed internal clock synchronization algorithms satisfy the bounded deviation condition, we make several assumptions; these must be guaranteed by any implementation of the clock synchronization algorithms. Since we use a probabilistic clock reading method and so processes can fail to read correct clocks within a given maximum error, our assumptions are somewhat different from those of [7, 8, 5]. The main differences are in the failure assumptions which limit the number of independent clock reading failures.

4.3.1 Initialization

We require that at the start of the first round all correct clocks be within a constant δ_S of each other, where δ_S will be determined later. Constant δ_S is also used to denote the maximum deviation between two correct clocks at the start of all rounds.

(**initial deviation**) *For any processes p and q that are correct at real time t^0:*

$$|C_p^0(t^0) - C_q^0(t^0)| \leq \delta_S. \quad \text{(A1)}$$

4.3.2 Interval Constraints

The real-time length of a clock synchronization round has to be bounded by constants r_{min} and r_{max}:

(**bounded interval**) *For any process p that is correct in real time interval $[t_p^k, t_p^{k+1}]$:*

$$r_{min} \leq t_p^{k+1} - t_p^k \leq r_{max}. \quad \text{(A2)}$$

The real-time delay between the beginning of the same synchronization round for different processes has to be bounded by a constant β:

(**bounded delay**) *For any processes p and q correct at t_p^k and t_q^k, respectively:*

$$|t_p^k - t_q^k| \leq \beta \tag{A3}$$

The overlap of rounds is restricted by:

(**nonoverlap**)

$$\beta \leq r_{min} \tag{A4}$$

Let us now derive some bounds for r_{min}, r_{max}, and β with respect to the clock synchronization algorithm of figure 6. Since the maximum scheduling delay for thread *synchronizer* is σ and a clock can obviously be adjusted by at most δ, we set $r_{max} \triangleq (1+\rho)P + \sigma + \delta$ and $r_{min} \triangleq (1-\rho)P - \delta$. All correct clocks synchronize their clocks at the same local time, i.e. $T_p^k = T_q^k$. Since the deviation between correct virtual clocks is bounded by δ, we define $\beta \triangleq (1+\rho)\delta$. Assumption nonoverlap is aquivalent to $(1+\rho)\delta \leq (1-\rho)P - \delta$, that is, it is sufficient to assume that $P \geq (2+3\rho)\delta$ to satisfy conditions (A2) to (A4).

4.3.3 Remote Clock Reading Requirements

We assume the use of a remote clock reading method that possesses the *error bound, crash handling, likely success,* and *timeliness* properties defined in section 3.2:

(**reading method**) Conditions *error bound, crash handling, likely success,* and *timeliness* are valid. (A5)

Since deterministic clock reading possesses all the above properties, our algorithms also work if deterministic remote clock reading is used.

4.3.4 Constraints for Constants

The correctness proofs of our clock synchronization algorithms require that constants δ and δ_s satisfy constraints (A6) and (A7), where $v = 0$ if only crash failures can occur, else $v = 2$.

$$\delta_S \geq (2+v)\Lambda + 2\rho(r_{max} + \beta) \tag{A6}$$

$$\delta \geq (2+v)\Lambda + 4\rho r_{max} + 2\rho\beta \tag{A7}$$

4.4 Midpoint Convergence Functions

Assume each correct process p would somehow be able to 1) read the value $C_q(t)$ of each clock q at a given time t without an error and 2) know which clocks are correct at t. Let L and R be the minimum and maximum values displayed at t by correct clocks and let $I(t) = [L, R]$ be the interval *spanned* by the correct clocks. If all processes would set their virtual clocks at the same time t to the midpoint of $I(t)$, then all correct clocks would be exactly synchronized at that point in time. Unfortunately, hardware clock drift prevents correct processes to agree on t, the randomness of communication delays prevents them from exactly reading remote clocks and the randomness that characterizes failure occurrences prevents them from knowing which clocks are correct and which clocks are faulty. Thus, correct processes can only *approximate* the interval spanned by the correct clocks. This causes the deviation between correct clocks to be positive. However, this deviation can be bounded by using the following simple ideas. First, we have to guarantee that any approximation $\mathcal{I}_p$ made by correct process p of the interval of correct clocks I is included in a bounded extension of I. Second, we must bound the distance between the approximations $\mathcal{I}_p$ and $\mathcal{I}_q$ computed by different correct processes p and q of the interval of correct clocks. This allows us to bound the deviation between the midpoints of all the approximation intervals computed by correct clocks, and hence, bound the deviation between correct clocks. Third, we must ensure that the deviation between successive round clocks C_p^k and C_q^{k+1} is bounded. This allows us to bound the deviation between correct virtual clocks at any point in real time.

We define the interval $I^k(t)$ determined by correct round clocks for a round k in accordance with the ideas above.

Since the clock of a process p that crashed in round k could have been read by some correct process q, we add to $I^k(t)$ the values of all clocks that belong to processes crashed in round k. This amounts to “shifting” the time of a crash to the end of the round in which the crash occurs. Thus, we will consider a process s to be ok at time t, if it is either correct at t or has crashed in the current round:

$$ok(s,t) \triangleq s\ correct\ at\ t \vee \exists k \exists u : t_s^k \leq u \leq t < t_s^{k+1} \wedge s\ correct\ before\ u \wedge s\ crashes\ at\ u.$$

We define the interval of ok clocks $I^k(t)$ to be

$$I^k(t) \triangleq [min\{C_s^k(t) \mid s \in \mathcal{P} \wedge ok(s,t)\}, max\{C_s^k(t) \mid s \in \mathcal{P} \wedge ok(s,t)\}]$$

A crash of p in round k causes a reading failure to occur whenever a correct process q attempts to read p's clock in rounds that follow k.

Since a correct process q estimates a correct remote clock s only with a certain error (which is at most Λ if q and s are correct and no reading failure occurs), we also define the Λ-extended interval of ok clocks $I_\Lambda^k(t)$ to be

$$I_\Lambda^k(t) \triangleq [min\{C_s^k(t) - \Lambda \mid s \in \mathcal{P} \wedge ok(s,t)\}, max\{C_s^k(t) + \Lambda \mid s \in \mathcal{P} \wedge ok(s,t)\}] \quad (11)$$

As the deviation of ok clocks is bounded by δ, the length of $I_\Lambda^k(t)$ is bounded by $\delta + 2\Lambda$.

We define the midpoint $mid([x,y])$ of an interval [x,y] as

$$mid([x,y]) \triangleq \frac{x+y}{2}. \quad (12)$$

The length of interval $[x,y]$, denoted $||[x,y]||$, is defined as

$$||[x,y]|| \triangleq y - x. \quad (13)$$

The distance $dist(X,Y)$ between two intervals X and Y is

$$dist(X,Y) \triangleq \begin{cases} min(Y) - max(Y) & if\ max(X) < min(Y) \\ min(X) - max(X) & if\ max(Y) < min(X) \\ 0 & otherwise \end{cases} \quad (14)$$

We denote process p's approximation of the interval of ok clocks by $\mathcal{I}_p^k$. A *midpoint convergence* function sets the clock of process p in round k to the midpoint of $\mathcal{I}_p^k$: $C_p(t_p^{k+1}) \triangleq mid(\mathcal{I}_p^k)$. Different midpoint convergence functions allow to mask different failures by using different approximations for the interval of ok clocks.

4.5 Design And Correctness Principles

The design and correctness proofs of our clock synchronization algorithms have been guided by the following principles.

The first principle, states that to bound the distance between the midpoints of the approximations $\mathcal{I}_p^k$ and $\mathcal{I}_q^k$, it is sufficient to ensure that 1) each approximation $\mathcal{I}_p^k$ made by a correct process p is included in the corresponding Λ-extended interval of ok clocks, 2) the distance between any two approximations $\mathcal{I}_p^k$ and $\mathcal{I}_q^k$ made by correct processes p and q is at most d, and 3) that any two correct processes p and q set their clocks at the same point in real time:

Principle P1: $t = t_p^{k+1} = t_q^{k+1} \wedge p \; and \; q \; correct \; at \; t \wedge \|I_\Lambda^k(t)\| \leq \delta + 2\Lambda \wedge \mathcal{I}_p^k \subseteq I_\Lambda^k(t) \wedge \mathcal{I}_q^k \subseteq I_\Lambda^k(t) \wedge dist(\mathcal{I}_p^k, \mathcal{I}_q^k) \leq d \rightarrow \; |mid(\mathcal{I}_p^k) - mid(\mathcal{I}_q^k)| \leq \frac{\delta+2\Lambda+d}{2}$

The requirement that any two correct processes set their clocks at the same point in real time is unfortunately too strong to be implementable; the most that can be guaranteed in practice is that the maximum distance between t_p^{k+1} and t_q^{k+1} is bounded by constant β. If the distance between $mid(\mathcal{I}_p^k)$ and $mid(\mathcal{I}_q^k)$ is bounded by e under the assumption of simultaneity (i.e. $t_p^{k+1} = t_q^{k+1}$) and the maximum distance between t_p^{k+1} and t_q^{k+1} is bounded by β, then the distance between the adjusted clocks C_p^{k+1} and C_q^{k+1} at time $t = max\{t_p^{k+1}, t_q^{k+1}\}$ is bounded by $e + 2\rho\beta$:

Principle P2: (p and q correct at t^{k+1}, and $|mid(\mathcal{I}_p^k) - mid(\mathcal{I}_q^k)| \leq e$ for assumption $t_p^{k+1} = t_q^{k+1}$) $\rightarrow$ $|C_p^{k+1}(t) - C_q^{k+1}(t)| \leq e + 2\rho\beta$ for $max\{t_p^{k+1}, t_q^{k+1}\} \leq t \leq t^{k+1}$.

When a process p already uses its new round clock C_p^{k+1} at t while another process q still uses its old round clock C_q^k t can in principle increase the distance between the virtual clocks C_p and C_q. To show that the maximum deviation between any two virtual clocks C_p and C_q is bounded by δ, it is sufficient to prove that $C_p^{k+1}(t_p^{k+1}) \in I^k(t_p^{k+1})$:

Principle P3: ($t_p^{k+1} \leq t < t_q^{k+1} \wedge p, q$ correct in $[t_p^{k+1}, t] \wedge C_p^{k+1}(t_p^{k+1}) \in I^k(t_p^{k+1}) \wedge \|I^k(t)\| \leq \delta) \rightarrow$ $|C_p^{k+1}(t) - C_q^k(t)| \leq \delta$.

We prove in the Appendix (theorem *Generic Proof*) that the bounded deviation requirement holds for midpoint convergence functions whenever assumptions (A1-A7) and the following conditions (C1-C3) hold:

$$\begin{aligned}
(C1) \;&\triangleq\; \mathcal{I}_p^k \subseteq I_\Lambda^k(t_p^{k+1}) \\
(C2) \;&\triangleq\; dist(\mathcal{I}_p^k, \mathcal{I}_q^k) \leq v\Lambda \; for \; t_p^{k+1} = t_q^{k+1} \\
(C3) \;&\triangleq\; C_p^{k+1}(t_p^{k+1}) \in I^k(t_p^{k+1})
\end{aligned}$$

5 Algorithms

Our clock synchronization algorithms differ in the failure assumptions they make. The first algorithm CSA_{Crash} masks up to F process or clock crash failures. CSA_{Crash} needs at least $F+1$ processes to guarantee the bounded deviation condition. Algorithm CSA_{Read} masks up to F failures, where each failure can be either a remote clock reading failure or a process or clock crash failure. It requires at least $2F + 1$ processes. The third algorithm $CSA_{Arbitrary}$ masks up to F failures, where each failure can be an arbitrary process or clock failure or a remote clock reading failure. It requires $3F + 1$ processes. Algorithm CSA_{Hybrid} is designed for a hybrid failure assumption: the maximum number of crash, reading and arbitrary failures are separately bounded by constants F_C, F_R and F_A. Algorithm CSA_{Hybrid} needs $3F_A + 2F_R + F_C + 1$ processes to mask these failures.

We denote by f_C^k the number of crashed processes and by f_A^k the number of processes suffering arbitrary failures at the end of round k, i.e. at time t^k. We define f_R^k as the maximum number of clock reading failures a correct process has suffered in round k when reading correct remote clocks. Hence, the failure assumptions for $CSA_{Arbitrary}$ restricts the number of arbitrary, clock reading and crash failures by $f_C^k + f_R^k + f_A^k \leq F$, while the failure assumptions for CSA_{Hybrid} assume that $f_C^k \leq F_C \wedge f_R^k \leq F_R \wedge f_A^k \leq F_A$.

5.1 Algorithm CSA_{Crash}

Algorithm CSA_{Crash} is based on the following failure assumptions:

- The number of processes suffering crash failures is at most F, that is, $f_C^k \leq F$ for all rounds k. (FA 1)
- No clock reading failures when reading a correct clock and no arbitrary failures occur: $f_R^k = f_A^k = 0$ for all rounds k. (FA 2)

When process r crashes in round k, the error bound computed by the clock reading method will be infinite for all following rounds and all processes can henceforth reject r's clock value. In round k a process may or may not observe a reading failure when attempting to read r's clock. For example, when in addition to r, process s also suffers a crash failure during round k, process p may read r's clock successfully and fail to read s's clock, while process q may read s's clock successfully and fail to read r's clock. (see figure 7). The crash failures of r and s increase the distance between the intervals $\mathcal{I}_p^k$ and $\mathcal{I}_q^k$ that p and q use to approximate the values of other correct clocks.

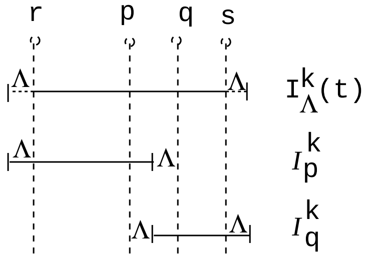

Figure 7: Processes r and s have crashed in the current interval.

We introduce an *interval extension* technique which allows to mask some clock reading errors without increasing the number N of needed processes: it allows a process p to use remote clock readings that yield an error greater than the maximum acceptable clock reading error Λ to extend the interval $\mathcal{I}_p^k$. We define the lower bound L of process p's approximation of the interval of ok clocks by

$$L \triangleq \ min\{\mathcal{C}_r^k(T_p^{k+1}, p) + \mathcal{E}_r^k(T_p^{k+1}, p) - \Lambda \mid r \in \mathcal{P}\}$$

To see that L is a lower bound, consider the process r for which $L = \mathcal{C}_r^k(T_p^{k+1}, p) + \mathcal{E}_r^k(T_p^{k+1}, p) - \Lambda$. By failure assumption (FA 2) and the *error bound* assumption of the clock reading method, $\mathcal{E}_r^k(T_p^{k+1}, p)$ bounds the clock reading error: $|C_r^k(t_p^{k+1}) - \mathcal{C}_r^k(T_p^{k+1}, p)| \leq \mathcal{E}_r^k(T_p^{k+1}, p)$. Thus, $L \geq C_r^k(t_p^{k+1}) - \Lambda \geq min\ I_\Lambda^k(t_p^{k+1})$ holds.

Similarly, we define the upper bound U of p's approximation of the interval of ok clocks as

$$U \triangleq \ max\{\mathcal{C}_r^k(T_p^{k+1}, p) - \mathcal{E}_r^k(T_p^{k+1}, p) + \Lambda \mid r \in \mathcal{P}\}$$

The bound U is smaller than the maximum of the Λ-extended interval of ok clocks: $U \leq max\ I_\Lambda^k(t_p^{k+1})$.

Process $p's$ approximation $\mathcal{I}_p^k$ of the ok clocks is thus defined by,

$$\mathcal{I}_p^k \triangleq \ [L, R]$$

From the above considerations we can conclude that $\mathcal{I}_p^k \subseteq I_\Lambda^k(t_p^{k+1})$ holds.

Figure 8 shows an example of an interval extension: $A_{q,p}^k$ denotes process p's approximation interval of process q's k-th round clock. We define $A_{q,p}^k \triangleq [\mathcal{C}_q^k(T_p^{k+1}, p) - \mathcal{E}_q^k(T_p^{k+1}, p), \mathcal{C}_q^k(T_p^{k+1}, p) + \mathcal{E}_q^k(T_p^{k+1}, p)]$. The length of interval $A_{p,p}^k$ is therefore 0 by definition. The correct value $C_q(t_p^{k+1})$ of q's clock is in $A_{q,p}^k$, because the failure assumptions exclude arbitrary failures. In this example process p suffers reading failures while reading the clocks of process r and s, because r and s suffer crash failures during round k. Nevertheless, p can use its approximations of r's and q's clock for the computation of $\mathcal{I}_p^k$.

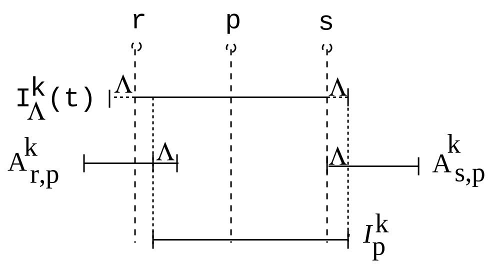

Figure 8: Process p extends $\mathcal{I}_p^k$ by using p's and q's clock readings.

We define a clock synchronization algorithm by its convergence function and the set of assumptions which have to be satisfied by an implementation of this clock synchronization algorithm at runtime. An implementation could consist of the code sketched in figure 6 together with an implementation of the convergence function, and a procedure that implements the improved probabilistic clock reading method discussed earlier. Algorithm CSA_{Crash} is defined by its convergence function $mid(\mathcal{I}_p^k)$ and it requires that the number of processes to be at least $F + 1$, assumptions (A1) - (A7), and (FA 1)

- (FA 2) to be true. Note that only F processes are necessary to guarantee the bounded deviation condition, but we require $F+1$ processes to ensure that at least one process remains correct.

Theorem I: Algorithm CSA_{Crash} guarantees the bounded deviation condition.

Informal Proof: We use the theorem *Generic Proof* of the Appendix to prove this theorem. For this purpose, we have to show that conditions (C1)-(C3) are valid. We already argued that condition (C1), i.e. $\mathcal{I}_p^k \subseteq I_\Lambda^k(t)$, is valid. We use the proof of theorem T2 in the Appendix to show condition (C3), i.e. that $C_p^{k+1}(t_p^{k+1}) \in I^k(t_p^{k+1})$ is valid. The proof of T2 shows that there exist two ok processes r and s so that $C_r(t_p^{k+1}) \leq C_p^{k+1}(t_p^{k+1}) \leq C_s(t_p^{k+1})$, i.e. $C_p^{k+1}(t_p^{k+1}) \in I^k(t_p^{k+1})$ holds.

Condition (C2) is holds for $v = 0$, i.e. $dist(\mathcal{I}_p^k, \mathcal{I}_q^k) = 0$ for $t_p^{k+1} = t_q^{k+1}$. When only one process is correct, then (C2) is trivially true. Consider now that processes p and q are correct. Processes p and q successfully read each others clock, because we exclude clock reading failures by the failure assumption (FA2). If $|C_p^k(t_p^{k+1}) - C_q^k(t_p^{k+1})| > 2\Lambda$, intervals $\mathcal{I}_p^k$ and $\mathcal{I}_q^k$ overlap each other, because p and q read each others clock with an error of at most Λ and they of course successfully read their own clock. If $|C_p^k(t_p^{k+1}) - C_q^k(t_p^{k+1})| \leq 2\Lambda$ the intervals $\mathcal{I}_p^k$ and $\mathcal{I}_q^k$ overlap, because $[C_p^k(t_p^{k+1}) - \Lambda, C_p^k(t_p^{k+1}) + \Lambda] \subseteq \mathcal{I}_p^k$ and $[C_q^k(t_q^{k+1}) - \Lambda, C_q^k(t_q^{k+1}) + \Lambda] \subseteq \mathcal{I}_q^k$ by definition of $\mathcal{I}_p^k$ and $\mathcal{I}_q^k$. $\square$

5.2 Algorithm CSA_{Read}

We make the following failure assumptions for algorithm CSA_{Read}:

- The number of processes suffering crash failures plus the number of reading failures when reading a correct clock is at most F: $f_C^k + f_R^k \leq F$ for all rounds k. (FA 3)
- No arbitrary failures occur: $f_A^k = 0$ for all rounds k. (FA 4)

These failure assumptions allow that, in any round k, a correct process fails to read up to $F - f_C^k$ correct clocks. Reading failures can be *independent*. For example, when $F = 1$, p can fail to read $q's$ clock and r can fail to read s' clock. To bound the deviation between two correct clocks at the start t^{k+1} of a round $k+1$ by δ_s, we have to bound the deviation between the approximations that different processes make of the interval $I_\Lambda^k(t)$. We use the pigeonhole principle to bound this deviation. Let S_p, S_q denote two sets of processes such that p has successfully read all clocks in S_p and q has successfully read all clocks in S_q. We call S_p the *success set* of process p. The pigeonhole principle, ensures that when $|S_p| + |S_q| \geq N + c$ then $|S_p \cap S_q| \geq c$. We require that every clock reads at least $\lfloor N/2 \rfloor + 1$ clocks successfully. Thus, for every pair (p,q) of correct processes there exists at least one common clock which both processes have successfully read. This limits the distance between their approximations $\mathcal{I}_p^k$ and $\mathcal{I}_q^k$ by 2Λ (assuming $t_p^{k+1} = t_q^{k+1}$). Figure 9 illustrates an example of two processes p and q which have successfully read s's clock, but they have failed to read each others clock and shows that $\mathcal{I}_p^k$ and $\mathcal{I}_q^k$ can be at at most 2Λ apart.

A clock synchronization algorithm needs at least $2F+1$ processes to mask F clock reading failures. Otherwise, we could partition the set of correct processes $\mathcal{P}$ in two sets P_1 and P_2 such that all processes in P_1 successfully read all clocks in P_1 and fail to read the clocks in P_2 and that the processes in P_2 successfully read all clocks in P_2 and fail to read the clocks in P_1. In such a case, the deviation between the clocks in P_1 and P_2 could get arbitrarily large. Algorithm CSA_{Read} requires therefore the number of processes to be at least $2F+1$.

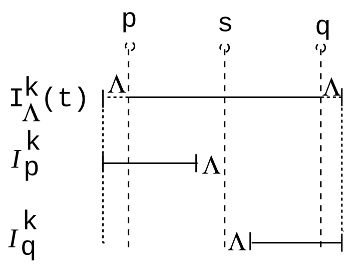

Figure 9: Processes p and q have read s's clock successfully.

Although we restrict the number of clock reading failures to be smaller than F for a correct process p, a process suffering a performance failure could observe more than F reading failures. Processes can use the error bounds of the probabilistic clock reading method to determine if they can synchronize their clock, because these error bounds are correct when no arbitrary failures can occur. A process p can synchronize its clock when it has successfully read at least $\lfloor N/2 \rfloor + 1$ clocks. If a process p fails to read that number of clocks successfully, the deviation between $p's$ clock and the correct clocks could get larger than δ. Hence, p will crash itself. Algorithm CSA_{Read} uses the same approximation $\mathcal{I}^k_p$ of $I^k_\Lambda(t^{k+1}_p)$ as CSA_{Crash} does.

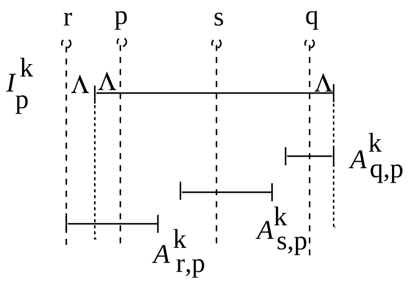

Figure 10: Processes p,q,r, and s are in the success set of process p.

We can improve the robustness of algorithm CSA_{Read} by refining the definition of the success set S_p. The idea hereby is, that the midpoint of $\mathcal{I}^k_p$ depends only on two clock readings and when the approximation interval $A^k_{q,p}$ of any other process q is a subset of $\mathcal{I}^k_p$ then this clock does not affect the midpoint of $\mathcal{I}^k_p$ and so process p can assume that it has read q's clock successfully. We generalize and formalize this idea by introducing the concept of the *effective* reading error: let $\mathcal{I}^k_p$ = [L,R]; process p's effective reading error e_r of r's clock in round k is defined by,

$$e_r \triangleq \begin{cases} L - C_r(t^{k+1}_p) & if\ C_r(t^{k+1}_p) < L \\ C_r(t^{k+1}_p) - R & if\ R < C_r(t^{k+1}_p) \\ 0 & otherwise \end{cases}$$

Process p fails to read $q's$ clock when the effective reading error is greater than Λ, that is, q's clock value is more than Λ apart from the interval $\mathcal{I}^k_p$. Let us define the *success set* of process p as

$$S_p \triangleq \{q \in \mathcal{P} \mid [\mathcal{C}^k_q(T^{k+1}_p, p) - max\{0, \mathcal{E}^k_q(T^{k+1}_p, p) - \Lambda\}, \mathcal{C}^k_q(T^{k+1}_p, p) + max\{0, \mathcal{E}^k_q(T^{k+1}_p, p) - \Lambda\}] \subseteq \mathcal{I}^k_p\}.$$

We can cut the edges of interval $A^k_{q,p}$ by Λ, because the effective reading error of q is still at most Λ. Figure 10 shows an example of a success set S_p. Process p is required to read at least $\lfloor N/2 \rfloor + 1$

clocks with an effective reading error of at most Λ, i.e. $|S_p| \geq \lfloor N/2 \rfloor + 1$. When $|S_p| < \lfloor N/2 \rfloor + 1$ process p cannot synchronize its clock and therefore must leave the group of correct processes.

Algorithm CSA_{Read} is defined by its convergence function $mid(\mathcal{I}_p^k)$, requires that the number N of processes be at least $2F + 1$, and that assumptions (A1) - (A7), and (FA 3) - (FA 4) hold.

Theorem II: CSA_{Read} guarantees the bounded deviation condition.

Informal Proof: The proof of conditions (C1) and (C3) is the same as for theorem I. We show that condition (C2) is true for $v = 2$. Let p and q be correct processes such that they have successfully read at least $\lfloor N/2 \rfloor + 1$ clocks: $|S_p|, |S_p| \geq \lfloor N/2 \rfloor + 1$. Because $|S_p| + |S_q| \geq 2(\lfloor N/2 \rfloor + 1) > N$, there exists at least one clock r which p and q have both read successfully: $r \in S_p$ and $r \in S_q$ (pigeonhole principle). The distance $dist(\mathcal{I}_p^k, \mathcal{I}_q^k)$ is therefore bounded by 2Λ for assumption $t_p^{k+1} = t_q^{k+1}$.

6 Algorithm $CSA_{Arbitrary}$

Algorithm $CSA_{Arbitrary}$ can mask remote clock reading failures as well as arbitrary process and clock failures. We make the following failure assumption (FA 5):
At any time t at most F processes are faulty. When F_A processes are faulty at time t^{k+1}, a correct process fails to read at most $F - F_A$ correct processes in round k.

We adapt the fault-tolerant midpoint function proposed in [3] to mask remote clock reading failures and to provide an optimal drift rate for the virtual clocks. Let us first recall the basic idea behind the fault-tolerant midpoint function of [3]. Consider first that no reading failures can occur. Let the number of processes N be at least $3F+1$. A process p estimates the virtual clocks of all processes at the end of every round and sorts these clock readings. Let array Y contain p's sorted clock readings of round k. Process p rejects the first F and the last F values, and it accepts the remaining $N - 2F$ clock readings. Interval $\mathcal{I}_p^k$ denotes the interval spanned by the clock values accepted by process p in round k: $\mathcal{I}_p^k = [Y[F], Y[N - F - 1]]$. Process p sets its clock at time t_p^{k+1} to the midpoint of $\mathcal{I}_p^k$: $C_p(t_p^{k+1}) \triangleq \frac{Y[F]+Y[N-F-1]}{2}$. The interval $\mathcal{I}_p^k$ is a subset of the Λ-extended interval of ok clocks $I_\Lambda^k(t_p^{k+1})$, because at most F clock readings belong to faulty clocks. Thus, there exist indices $0 \leq i, j \leq F$ so that $Y[i]$ and $Y[N - 1 - j]$ are clock readings of correct clocks. Because Y is sorted, it follows that $Y[i] \leq Y[F] \wedge Y[N-1-j] \geq Y[N-F-1]$ and hence $\mathcal{I}_p^k \subseteq I_\Lambda^k(t_p^{k+1})$ holds.

The distance between the intervals $\mathcal{I}_p^k$ and $\mathcal{I}_q^k$ of the correct processes p and q is bounded. Consider for simplicity that $t_p^{k+1} = t_q^{k+1}$. In the worst case, only $2F + 1$ clocks are correct and p and q reject the first F and the last F correct clock values. But p and q do not reject the F+1-th correct clock value. Because of clock reading errors, p and q can have a different view with respect to the ordering of the correct clocks. The two $F + 1$-th correct clock values of p and q are nevertheless at most 2Λ apart (for $t_p^{k+1} = t_q^{k+1}$), because the reading errors are bounded by Λ. The distance between $\mathcal{I}_p^k$ and $\mathcal{I}_q^k$ for $t_p^{k+1} \neq t_q^{k+1}$ is therefore bounded by $2\Lambda + 2\beta\rho$, because $|t_p^{k+1} - t_q^{k+1}| \leq \beta$ and p's and q's clocks can drift apart from each other by at most 2ρ.

The drift rate of virtual clocks synchronized by the above sketched fault-tolerant midpoint algorithm is not optimal: consider the case when all hardware clocks drift at their maximum drift rate ρ, all processes read remote clocks with a positive clock reading error of Λ, initially all clocks show the same clock value and all clocks are adjusted at the same points in real-time. When process p's clock shows T at the end of round k, then p approximates all other clocks by $T + \Lambda$. Process p rejects its own clock value, because p has to reject the first F and the last F clock values. Therefore, p sets its clock to $T + \Lambda$. The drift rate of the virtual clocks is hence greater than ρ, because the clocks have an additional drift introduced by the periodic increment by Λ.

The fault-tolerant midpoint function of [3] is not applicable for failure assumption (FA 5), because under (FA 5) reading failures can be independent from each other. To show this, let us consider $F = 1$ and four correct processes p, q, r, s. We will show that the maximum deviation between the clocks of these processes is not necessarily bounded by δ. Assume p's and q's clocks show always the same clock value, i.e. $C_p(t) = C_q(t)$, and also r's and q's clocks show always the same value. Let the drift rate of p's and q's hardware clocks be ρ and the drift rate of r's and s's hardware clocks be 0. Furthermore, assume that p and q fail to read r's clock and r and s fail to read p's clock and that the other reading errors are zero. Thus, we can assume that for all rounds k conditions $T^{k+1} = \mathcal{C}_q^k(T^{k+1}, p) = \mathcal{C}_p^k(T^{k+1}, q) = \mathcal{C}_r^k(T^{k+1}, p) = \mathcal{C}_r^k(T^{k+1}, q)$ and $T^{k+1} = \mathcal{C}_s^k(T^{k+1}, r) = \mathcal{C}_r^k(T^{k+1}, s) = \mathcal{C}_p^k(T^{k+1}, r) = \mathcal{C}_p^k(T^{k+1}, s)$ hold. In this case no process changes the value of its virtual clock and the deviation between the virtual clocks will eventually become greater than δ.

The basic idea behind the midpoint function proposed in this paper is as follows: a process must restrict the number of clock values that the midpoint function of [3] would reject, in order to be able to mask clock reading failures. We show later a lower bound for the number of accepted clock readings that is sufficient to bound the deviation between the approximations of the interval of ok clocks and that ensures that these approximations are subsets of the Λ-extended interval of ok clocks. Furthermore, a process should always include its own clock value in its approximation of the interval of ok clocks. This allows us to demonstrate that the drift rate of the virtual clocks is optimal.

Let us assume that process p is correct at time t_p^{k+1}. To calculate how many clock values process p can reject, we define two sorted arrays L and U with indexes ranging from 0 to $N-1$. Array L contains for every ok clock r an estimate $\mathcal{C}_r^k(T_p^{k+1}, p) + \mathcal{E}_r^k(T_p^{k+1}, p) - \Lambda$ of r's clock value $C_r^k(t_p^{k+1})$ such that the difference between the correct clock value $C_r(t_p^{k+1})$ and this estimate is at most Λ whenever the estimate is less than $C_r(t_p^{k+1})$:

$$L \triangleq sort\{\mathcal{C}_r^k(T_p^{k+1}, p) + \mathcal{E}_r^k(T_p^{k+1}, p) - \Lambda \mid r \in \mathcal{P}\}$$

Note that the definition of L also includes clock readings for which process p has suffered clock reading failures.

Similarly, array U contains for every ok clock r another estimate $\mathcal{C}_r^k(T_p^{k+1}, p) - \mathcal{E}_r^k(T_p^{k+1}, p) + \Lambda$ of r's clock value $C_r^k(t_p^{k+1})$ such that the difference between the correct clock value $C_r(t_p^{k+1})$ and this estimate is at most Λ whenever the estimate is greater than $C_r(t_p^{k+1})$.

$$U \triangleq sort\{\mathcal{C}_r^k(T_p^{k+1}, p) - \mathcal{E}_r^k(T_p^{k+1}, p) + \Lambda \mid r \in \mathcal{P}\}$$

Next, we define the interval J_v spanned by the lowest and highest values of the arrays L and U after removing at most the first v elements of L and at most the last v elements of U. When process p is correct, then $[T_p^{k+1} - \Lambda, T_p^{k+1} + \Lambda]$ is a subset of the Λ-extended interval of ok clocks $I^k(t_p^{k+1})$. We extend therefore J_v by the interval $[T_p^{k+1} - \Lambda, T_p^{k+1} + \Lambda]$:

$$J_v \triangleq [min\{L[v], T_p^{k+1} - \Lambda\}, max\{U[N-v-1], T_p^{k+1} + \Lambda\}]$$

To simplify the correctness proof of algorithm $CSA_{Arbitrary}$, we will assume that process p always rejects (with respect to J_v) the first v readings in L and the last v readings in U, that is, p rejects $L[0], .., L[v-1]$ and $U[N-v], .., U[N-1]$. We define the set of rejected clocks $R_v \subseteq \mathcal{P}$ to be set of all clock readings rejected by p. The set R_v contains $2v$ elements.

A process p accepts all clock readings in J_v that are not in R_v with an effective clock reading error of at most Λ, i.e. the correct clock value of an ok clock is at most Λ to the left or right of interval J_v. We define the *acceptance set* S_v with respect to J_v as follows:

$$\begin{aligned} S_v \;\triangleq\; & \{q \in \mathcal{P} - R_v \mid \\ & J_v \supseteq [\mathcal{C}_q^k(T_p^{k+1}, p) - max\{0, \mathcal{E}_q^k(T_p^{k+1}, p) - \Lambda\}, \mathcal{C}_q^k(T_p^{k+1}, p) + max\{0, \mathcal{E}_q^k(T_p^{k+1}, p) - \Lambda\}] \end{aligned}$$

Processes have to restrict the number of rejected clock values to guarantee that the distance between two approximations of the interval of ok clocks is bounded. We have shown previously that at least $2F+1$ processes are necessary to mask up to F independent reading failures per process. To mask up to F arbitrary process and clock failures at least $3F+1$ processes are needed. We exploit this fact to mask more than the $F - F_A$ permitted clock reading failures. We will in fact show that clock synchronization algorithm $CSA_{Arbitrary}$ is not only correct for failure assumption (FA 5), but it is also correct for the weaker failure assumption (FA 6): *The number of independent reading failures per round and process F_R is bounded by $\lfloor \frac{3}{2}(F - F_A) \rfloor \geq F_R$ and $F \geq F_A$.*

Let us now explain how a process p selects the number n so that J_n can be used as the approximation $\mathcal{I}_p^k$ of the interval of ok clocks $I^k(t_p^{k+1})$. Obviously, when p rejects too many clock values, p can get out of synch and also when p rejects too few clock values, its approximation $\mathcal{I}_p^k$ is not necessarily a subset of $I_\Lambda^k(t_p^{k+1})$ and thus p could also get out of synch. We will show that n can be chosen as

$$n \triangleq max\{0 \leq v \leq F \mid |S_v| \geq N - 2v - \lfloor \frac{3}{2}(F - v) \rfloor \}$$

When too many clock reading failures occur, n is undefined. In this case it is not guaranteed that process p can synchronize its clock and p should remove itself from the group of correct processes. When n is defined, process p approximates the interval of ok clocks by

$$\mathcal{I}_p^k \triangleq J_n$$

Note that the set of rejected processes R_n and the acceptance set S_n can contain processes for which process p has suffered clock reading failures. Process p has effectively suffered reading failures for processes which are not in S_n and R_n. Therefore, the number of effective reading failures is $N - |S_n| - |R_n| = N - |S_n| - 2n$.

Figure 11 shows an example for $F = 4$. Process p has not read any remote clock with a reading error smaller than Λ. Nevertheless, process p effectively suffers only three reading failures and it rejects the first two and the last two clock readings.

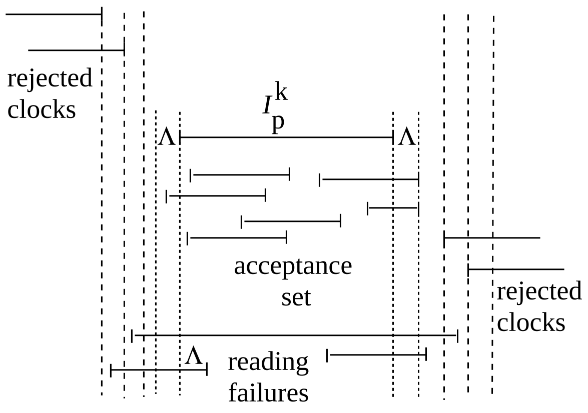

Figure 11: An example of process p's approximation of I_Λ^k.

Algorithm $CSA_{Arbitrary}$ is defined by its convergence function $mid(\mathcal{I}_p^k)$, it assumes that the number of processes is at least $3F+1$ and requires assumptions (A1) to (A7), and (FA 5) to be true.

Theorem III: $CSA_{Arbitrary}$ guarantees the bounded deviation condition.

Informal Proof: We prove this theorem for the relaxed failure assumption (FA 6). Therefore, theorem III is also valid for (FA 5). We show first that $\mathcal{I}_p^k \subseteq I_\Lambda^k(t_p^{k+1})$ for a correct process p. Second, we show that the distance between $\mathcal{I}_p^k$ and $\mathcal{I}_p^k$ is bounded by $2\Lambda + 2\beta\rho$ for correct processes p and q.

To prove that $\mathcal{I}_p^k \subseteq I_\Lambda^k(t_p^{k+1})$, we show that $n \geq F_A - f$ (L1), where f is the number of faulty processes which p effectively fails to read. When $n = F$, condition $n \geq F_A - f$ holds, because $F \geq F_A$ by (FA 6). Consider now that $n < F$. Let $F_p = c + f$ be the number effective reading failures process p has suffered in round k, where c is the number of correct clocks which p has failed to read. Hence, $f \leq F_A \wedge c \leq F_R$ (L2) by (FA 6). We use failure assumption (FA 6) to derive that $\frac{2}{3}F_R \leq F - F_A$ (L3). With the definition of n, we can conclude that

$$\lfloor \frac{3}{2}(F-n-1) \rfloor < F_p \leq \lfloor \frac{3}{2}(F-n) \rfloor \tag{15}$$

Therefore, $n = \lfloor F - \frac{2}{3}F_p \rfloor$ and

$$\begin{array}{lll} & \lfloor F_A - f \rfloor \geq F_A - f & \\ \Rightarrow & \lfloor F_A + \frac{2}{3}F_R - \frac{2}{3}F_p \rfloor \geq F_A - f & [F_R \geq F_p - f] \\ \Rightarrow & \lfloor F_A + F - F_A - \frac{2}{3}F_p \rfloor \geq F_A - f & [F - F_A \geq \frac{2}{3}F_R] \\ \Rightarrow & n \geq F_A - f & [n = \lfloor F - \frac{2}{3}F_p \rfloor] \end{array}$$

We now show that the distance between $\mathcal{I}_p^k$ and $\mathcal{I}_q^k$ is bounded by 2Λ for correct p and q (L5) for the simplified assumptions that $t_p^{k+1} = t_q^{k+1}$ and that all processes see the same order order of correct processes, that is, when p and q have successfully read correct clocks r and s then $\mathcal{C}_r^k(T^{k+1},p) \leq \mathcal{C}_s^k(T^{k+1},p) \iff \mathcal{C}_r^k(T^{k+1},q) \leq \mathcal{C}_s^k(T^{k+1},q)$. Let $F_p = c_p + f_p$ and $F_q = c_q + f_q$ be the number of effective reading failures suffered by p and q. In the worst case, it is possible that p does not accept the first $n_p + c_p$ correct clocks and q does not accept the last $n_q + c_q$ clocks (wrt. to the common ordering of correct clock values). The number of correct processes is $N - F_A$. We can conclude that (L5) is valid, because we show $n_p + c_p + n_q + c_q < N - F_A$, i.e. there exists one correct process r so that $\mathcal{I}_p^k$ is at most Λ to the right of r's clock value and $\mathcal{I}_q^k$ is at most Λ to the left of r's clock value for $t_p^{k+1} = t_q^{k+1}$:

$$\begin{array}{lll}
c_p + n_p + c_q + n_q & & \\
= & F_p - f_p + n_p + F_q - f_q + n_q & [F = c + f] \\
\leq & N - |S_{n_p}| - 2n_p - f_p + n_p + N - |S_{n_q}| - 2n_q - f_q + n_q & [F = N - |S_n| - 2n] \\
\leq & \lfloor \frac{3}{2}(F - n_p) \rfloor - f_p + n_p + \lfloor \frac{3}{2}(F - n_q) \rfloor - f_q + n_q & [|S_n| \geq N - 2n - \lfloor \frac{3}{2}(F - n) \rfloor] \\
\leq & \frac{3}{2}F - \frac{1}{2}n_p - f_p + \frac{3}{2}F - \frac{1}{2}n_q - f_q & [\textit{simple transformations}] \\
\leq & \frac{3}{2}F - \frac{1}{2}(F_A - f_p) - f_p + \frac{3}{2}F - \frac{1}{2}(F_A - f_q) - f_q & [n \geq F_A - f] \\
\leq & 3F - F_A & [\textit{simple transformations}] \\
< & N - F_A & [N \geq 3F + 1]
\end{array}$$

□

6.1 Algorithm CSA_{Hybrid}

We combine combine the techniques used in the first three clock synchronization algorithms to derive a hybrid algorithm. We assume that for any time t,

- The number of processes suffering crash failures is at most F_C: $f_C^k \leq F_C$. (FA 7)
- A correct process suffers at most F_R reading failures for correct clocks per round: $f_R^k \leq F_R$. (FA 8)
- The number of processes suffering arbitrary failures is at most F_A: $f_A^k \leq F_A$. (FA 9)

To mask these failures we need F_C processes to mask crash failures, $2F_R$ processes to mask remote clock reading failures and $3F_A$ processes to mask arbitrary clock or process failures. Thus, we have to assume that the number of processes N is at least $F_C + 2F_R + 3F_A + 1$. In a manner similar to that explained previously, we define:

$$\begin{array}{rcl}
L & \triangleq & sort\{\mathcal{C}_r^k(T_p^{k+1}, p) + \mathcal{E}_r^k(T_p^{k+1}, p) - \Lambda \mid r \in \mathcal{P}\} \\
U & \triangleq & sort\{\mathcal{C}_r^k(T_p^{k+1}, p) - \mathcal{E}_r^k(T_p^{k+1}, p) + \Lambda \mid r \in \mathcal{P}\} \\
\mathcal{I}_p^k & \triangleq & [min\{L[F_A], T_p^{k+1} - \Lambda\}, max\{U[N - F_A - 1], T_p^{k+1} + \Lambda\}] \\
R_p & \triangleq & \{r \in \mathcal{P} \mid r\ is\ rejected\ by\ p\ in\ round\ k\} \\
S_p & \triangleq & \{q \in \mathcal{P} \backslash R_p \mid \mathcal{I}_p^k \supseteq \\
& & [\mathcal{C}_q^k(T_p^{k+1}, p) - max\{0, \mathcal{E}_q^k(T_p^{k+1}, p) - \Lambda\}, \mathcal{C}_q^k(T_p^{k+1}, p) + max\{0, \mathcal{E}_q^k(T_p^{k+1}, p) - \Lambda\}]\}
\end{array}$$

A process p has to read at least $F_A + F_R + 1$ clocks successfully to synchronize its clock: $|S_p| \geq F_A + F_R + 1$. This is sufficient to show that the distance between two approximations of the interval of correct clocks is bounded, since there are at least $2F_A + 2F_R + 1$ correct processes and a correct process does not accept at most the left $F_A + F_R$ and the right $F_A + F_R$ clock values of the correct clocks.

Algorithm CSA_{Hybrid} is defined by its convergence function $mid(\mathcal{I}_p^k)$, its requirement that the number of processes to be at least $F_C + 2F_R + 3F_A + 1$ and the assumptions (A1) - (A7) and (FA 6) - (FA 9).

Theorem IV: CSA_{Hybrid} guarantees the bounded deviation condition.

Proof similar to the previous proofs.

6.2 General properties of the algorithms

For all our algorithms, the maximum deviation between correct clocks is bounded by

$$(2+v)\Lambda + 4\rho r_{max} + 2\rho\beta,$$

where v=0 if only crash failures can occur (FA 1-2), and v=2 otherwise. This maximum deviation is better than the best maximum deviations we are aware of. For example, the bound for the fault-tolerant midpoint function derived with the constraints for δ given in [8] and properties of the fault-tolerant midpoint function given in [7] is

$$9\Lambda + 10\rho\beta + 4\rho r_{max}.$$

Let us now derive an upper bound K for the maximum adjustment of a virtual clock. For K the following condition holds: $K \geq max\{|C_p^k(t_p^{k+1}) - C_p^{k+1}(t_p^{k+1})| \mid p \in \mathcal{P}\ correct\ at\ t_p^{k+1}\}$. The length of the interval $\mathcal{I}_p^k$ is bounded by $\delta + 2\Lambda$, because at time t_p^{k+1} the clocks can be up to δ apart, and the extension of this interval by clock reading errors is at most 2Λ. By definition of $\mathcal{I}_p^k$, process $p's$ clock value T_p^{k+1} is at least Λ apart from the bounds of $\mathcal{I}_p^k$. Hence, p changes its clock by at most $K = \frac{(\delta+2\Lambda)}{2} - \Lambda = \frac{\delta}{2}$.

The envelope rate ρ_e and the drift rate ρ_v of the virtual clocks are *optimal*, in the sense that they are bounded by the maximum drift rate ρ of the hardware clocks. Conditions *linear envelope* and *clock drift* introduce discontinuity values R and G to cope with the discrete adjustments of the virtual clocks. We can derive a better upper bound for the discontinuity of the envelope rate, because the bound for the initial deviation between correct clocks δ_S is better than the worst case deviation δ which we have to consider for condition *clock drift*. The following theorem (T1) states that the envelope rate is at most $\rho_e \triangleq \rho$ and that the discontinuity is at most $R \triangleq \delta_S$.

Theorem (T1):$\forall p \in \mathcal{P}$: *p correct at* $t \geq t_p^0$:

$$(1-\rho)(t-t_p^0) - R \leq C_p(t) - C_p(t_p^0) \leq (1+\rho)(t-t_p^0) + R$$

The next theorem shows that the drift rate of the virtual clocks is at most $\rho_v \triangleq \rho$ and the discontinuity is at most $G \triangleq K + R = \frac{\delta}{2} + \delta_S$. The proofs of these two theorems are in the Appendix.

Theorem (T2): *For any process* $p \in \mathcal{P}$ *that is correct in interval* $[t_0, t_1]$, *where* $t_p^0 \leq t_0 \leq t_1$:

$$(1-\rho)(t_1-t_0) - G \leq C_p(t_1) - C_p(t_0) \leq (1+\rho)(t_1-t_0) + G$$

7 Conclusion

We have proposed a family of new probabilistic internal clock synchronization algorithms. The members of this family differ in the failure classes tolerated, from crash to arbitrary. Because these algorithms rely on probabilistic remote clock reading, they achieve synchronization precisions better than those achievable by previously known deterministic internal fault-tolerant clock synchronization algorithms. Another advantage of the proposed protocols is that they use a linear, instead of a quadratic, number of messages, and that message exchanges are staggered in time instead of all happening in narrow synchronization intervals.

We have proposed a general specification for a probabilistic remote clock reading method: requirements error bound, crash handling, best effort and timeliness. The improved probabilistic clock reading method proposed in this paper satisfies all the above conditions, reduces the number of messages and improves the probability that remote clocks are read successfully. It uses disjoint time slots to send unreliable broadcast messages. This increases the number of non-concurrent messages and hence decreases the number of messages needed to read all remote clocks successfully. Two processes can use the information in all non-concurrent message pairs between them to read each others clock. The proposed transitive clock reading method allows a first process to estimate the clock of a second process by using a third process's approximation of the clock of the second process. This can reduce the number of message, in the best case to $N+1$, and, in the worst case, to kN messages per synchronization round.

The proposed clock synchronization algorithms use different kind of midpoint functions. We used the error bounds provided by the probabilistic clock reading method to increase the robustness of our clock synchronization algorithms by also making use of some failed clock readings. This increases the probability that a process can successfully synchronize its clock. Furthermore, the error bounds provided by probabilistic clock reading allow a process to determine if it has successfully synchronized its clock or it has failed to synchronize.

The drift rate of the virtual clocks is optimal and the derived upper bound for the maximum deviation is better than any other upper bound derived for convergence functions we are aware of. The proposed convergence function provides therefore also advantages when it is applied with a deterministic clock reading method.

8 Appendix

8.1 General Assumptions

Since ρ is such a small quantity, we ignore terms of the order of ρ^2 or smaller, for example we will equate $(1+\rho)^{-1}$ with $(1-\rho)$ and $(1-\rho)^{-1}$ with $(1+\rho)$.

8.2 Inequalities

We use in our proofs the following well known inequalities:

$$\begin{array}{ll} (W1) & |a+b| \leq |a|+|b| \\ (W2) & |a-b| \leq c \Rightarrow b-c \leq a \leq b+c \end{array}$$

8.3 Transitive Remote Clock Readings

⟨1⟩1. ASSUME: 1. p, q, r are correct clocks and $T = C_p(t_p) = C_q(t_q) = C_r(t_r)$.
2. $|C_r(T,p) - C_r(t_p)| \leq \mathcal{E}_r(T,p)$.
3. $|C_q(T,p) - C_q(t_p)| \leq \mathcal{E}_q(T,p)$.
4. We assume that the drift of clocks is bounded by $\rho < 1$.
5. The deviation between correct clocks is bounded by δ.
6. $\mathcal{C}_r(T,q) \triangleq \mathcal{C}_r(T,p) - \mathcal{C}_q(T,p) + T$.
7. $\mathcal{E}_r(T,q) \triangleq \mathcal{E}_r(T,p) + \mathcal{E}_q(T,p) + 2\rho\delta$.

PROVE: $|C_r(T,q) - C_r(t_q)| \leq \mathcal{E}_r(T,q)$

PROOF:

⟨2⟩1. ASSUME: 1. drift rate of p's hardware clock is ρ^* ($-\rho \leq \rho^* \leq \rho$) in $[t_p, t_q]$.

PROVE: $|t_q - t_p| \leq (1+\rho)\delta$

PROOF: with assumption ⟨1⟩1.5

$$
\begin{array}{lll}
 & |C_p(t_q) - C_r(t_q)| \leq \delta & \\
\Rightarrow & |C_p(t_p) + (t_q - t_p)(1+\rho*) - C_r(t_q)| \leq \delta & [\langle 1\rangle 1.4] \\
\Rightarrow & |T + (t_q - t_p)(1+\rho*) - T| \leq \delta & [\langle 1\rangle 1.1] \\
\Rightarrow & |t_q - t_p|(1+\rho*) \leq \delta & [\langle 1\rangle 1.4] \\
\Rightarrow & |t_q - t_p| \leq (1+\rho)\delta & [\langle 2\rangle 1.1]
\end{array}
$$

⟨2⟩2. Q.E.D.

PROOF:

$$
\begin{array}{lll}
|C_r(T,q) - C_r(t_q)| & & \\
= & |\mathcal{C}_r(T,p) - \mathcal{C}_q(T,p) + T - C_r(t_q)| & [\langle 1\rangle 1.6] \\
\leq & \mathcal{E}_r(T,p) + \mathcal{E}_q(T,p) + |(C_r(t_p) - C_r(t_q)) - (C_q(t_p) - T)\ | & [\langle 1\rangle 1.2, \langle 1\rangle 1.3] \\
\leq & \mathcal{E}_r(T,p) + \mathcal{E}_q(T,p) + |(t_p - t_q)(1+\rho) - (t_p - t_q)(1-\rho)| & [\langle 1\rangle 1.4] \\
\leq & \mathcal{E}_r(T,p) + \mathcal{E}_q(T,p) + 2\rho\delta & [\langle 2\rangle 1] \\
= & \mathcal{E}_r(T,q) & [\langle 1\rangle 1.7]
\end{array}
$$

8.4 Envelope and Drift Rate of Virtual Clocks

⟨1⟩1. ASSUME: 1. $T_u^p \triangleq max\{C_q(t_p^i) | q \in \mathcal{P}$ correct at $t_p^i\}$.
2. $T_l^p \triangleq min\{C_q(t_p^i) | q \in \mathcal{P}$ correct at $t_p^i\}$.
3. $R \triangleq \delta_S$.
4. $I_\Lambda^i(t) \triangleq [min\{C_s^i(t) - \Lambda \mid s\ ok\ at\ t\}, max\{C_s^i(t) + \Lambda \mid s\ ok\ at\ t\}]$.
5. $\mathcal{I}_p^i$ as definied in section 5.1,5.2,6, or 6.1.
6. $T_u^p - T_l^p \leq \delta_S$.
7. p is correct at time $t \geq t_p^i$.
8. $C_p^i(t) = H_p(t) + A_p^i$.
9. The drift rate of H_p and so also of $C_p^i(t)$ is bounded by ρ.
10. $C_p(t) = C_p^i(t)$ for $t_p^i \leq t \leq t_p^{i+1}$.
11. $\mathcal{I}_p^i \subseteq I_\Lambda^i(t_p^{i+1})$.

PROVE: $\forall p \in \mathcal{P}$: p correct at $t \geq t_p^i$:
$(t - t_p^i)(1-\rho) - R \leq C_p(t) - C_p(t_p^i) \leq (t - t_p^i)(1+\rho) + R$.

PROOF:

⟨2⟩1. PROVE: $\forall j \forall p \in \mathcal{P}\ correct\ at\ t_p^{i+j}$:
$T_l^p + (t_p^{i+j} - t_p^i)(1-\rho) \leq C_p(t_p^{i+j}) \leq T_u^p + (t_p^{i+j} - t_p^i)(1+\rho)$

PROOF: by induction over j.

CASE: $j = 0$

PROOF: holds by definition of T_l^p and T_u^p.

CASE: $j \to j+1$

PROOF:

Because $\mathcal{I}_p^{i+j} \subseteq I_\Lambda^{i+j}(t_p^{i+j+1})$ ($\langle 1\rangle 1.11$), there exists two ok clocks q and r with

$\langle 4\rangle 1.$ $C_q^{i+j}(t_p^{i+j+1}) \geq max(\mathcal{I}_p^{i+j}) - \Lambda$ [$\langle 1\rangle 1.11$]

$\langle 4\rangle 2.$ $C_r^{i+j}(t_p^{i+j+1}) \leq min(\mathcal{I}_p^{i+j}) + \Lambda$ [$\langle 1\rangle 1.11$]

$\langle 4\rangle 3.$ $min(\mathcal{I}_p^{i+j}) + \Lambda \leq mid(\mathcal{I}_p^{i+j}) \leq max(\mathcal{I}_p^{i}) - \Lambda$ [$\langle 1\rangle 1.5$: $\|I_p^{i+j}\| \geq 2\Lambda$]

$\langle 4\rangle 4.$ $C_r^{i+j}(t_p^{i+j+1}) \leq mid(\mathcal{I}_p^{i+j}) \leq C_q^{i+j}(t_p^{i+j+1})$ [$\langle 4\rangle 1$,$\langle 4\rangle 2$,$\langle 4\rangle 3$]

$\langle 4\rangle 5.$ Q.E.D.

$T_l^p + (t_p^{i+j+1} - t_p^i)(1-\rho) \leq C_p^{i+j+1}(t_p^{i+j+1}) \leq T_u^p + (t_p^{i+j+1} - t_p^i)(1+\rho)$ [$\langle 4\rangle 4$]

$\langle 2\rangle 2.$ PROVE: $T_l^p + (t - t_p^i)(1-\rho) \leq C_p(t) \leq T_u^p + (t - t_p^i)(1+\rho)$

PROOF: $\langle 2\rangle 1$ implies $\langle 2\rangle 2$

$\langle 2\rangle 3.$ Q.E.D.

PROOF:

$\langle 3\rangle 1.$ $C_p(t) - C_p(t_p^i) \leq T_u^p + (t - t_p^i)(1+\rho) - T_l^p \leq (t - t_p^i)(1+\rho) + R$ [$\langle 2\rangle 2$,$\langle 1\rangle 1.6$]

$\langle 3\rangle 2.$ $C_p(t) - C_p(t_p^i) \geq T_l^p + (t - t_p^i)(1-\rho) - T_u^p \geq (t - t_p^i)(1-\rho) - R$ [$\langle 2\rangle 2$,$\langle 1\rangle 1.6$]

$\langle 3\rangle 3.$ Q.E.D.

$(t - t_p^i)(1-\rho) - R \leq C_p(t) - C_p(t_p^i) \leq (t - t_p^i)(1+\rho) + R$ [$\langle 3\rangle 1$,$\langle 3\rangle 2$]

$\langle 1\rangle 2.$ ASSUME: 1. K is the maximum adjustments of virtual clocks.

2. The discontinuity G is defined by $G \triangleq K + R$;

PROVE: $\forall p \in \mathcal{P}, \forall s, t\ with\ t_p^0 \leq s \leq t$: p correct in $[s, t]$:

$(1-\rho)(t-s) - G \leq C_p(t) - C_p(s) \leq (1+\rho)(t-s) + G.$

PROOF:

$\langle 2\rangle 1.$ We choose i so that $t_p^i \leq s < t_p^{i+1}$.

The adjustment of clock C_p at time t_p^{i+1} is at most $K = G - R$.

$\langle 2\rangle 2.$ $(t_p^{i+1} - s)(1-\rho) - (G - R) \leq C_p(t_p^{i+1}) - C_p(s) \leq (t_p^{i+1} - s)(1+\rho) + (G - R)$ [$\langle 1\rangle 1.9$,$\langle 1\rangle 1.10$]

$\langle 2\rangle 3.$ $(t - t_p^{i+1})(1-\rho) - R \leq C_p(t) - C_p(t_p^{i+1}) \leq (t - t_p^{i+1})(1+\rho) + R$ [$\langle 1\rangle 1$]

$\langle 2\rangle 4.$ $(t-s)(1-\rho) - G \leq C_p(t) - C_p(s) \leq (t-s)(1+\rho) + G$ [$\langle 2\rangle 2$,$\langle 2\rangle 3$]

8.5 Principle P1

$\langle 1\rangle 1.$ ASSUME: 1. $I \triangleq [k_0, k_1], k_1 \geq k_0, k \triangleq k_1 - k_0.$

2. $X \triangleq [x_0, x_1], x_1 \geq x_0, x \triangleq x_1 - x_0, X \subseteq I.$

3. $Y \triangleq [y_0, y_1], y_1 \geq y_0, y \triangleq y_1 - y_0, Y \subseteq I.$

4. $d \triangleq y_0 - x_1.$

5. $dist(X, Y) = d \vee dist(X, Y) = 0$, *$X$ and Y overlap or Y is right of X.*

PROVE: $|midX - midY| \leq \frac{k + dist(X,Y)}{2}$

PROOF:

$\langle 2\rangle 1.$ PROVE: $x + y \leq k - d$

PROOF:

$$\begin{array}{lll} x+y & & \\ = & (x_1-x_0)+(y_1-y_0) & [\langle 1\rangle 1.2, \langle 1\rangle 1.3] \\ = & y_1-x_0-(y_0-x_1) & [\text{reorder terms}] \\ \leq & k-d & [\langle 1\rangle 1.2, \langle 1\rangle 1.3, \langle 1\rangle 1.4] \end{array}$$

$\langle 2\rangle 2.$ Q.E.D.

PROOF:

$\langle 3\rangle 1.$ $midX = \frac{x_0+x_1}{2} = \frac{x_0+x_0+x}{2} = x_0 + \frac{x}{2}$ [$\langle 1\rangle 1.2$]

$\langle 3\rangle 2.$ $midY = \frac{y_0+y_1}{2} = \frac{x_1+d+x_1+d+y}{2} = x_0 + x + d + \frac{y}{2}$ [$\langle 1\rangle 1.2$,$\langle 1\rangle 1.3$]

$\langle 3\rangle 3.$ $d = y_0 - x_1 \leq dist(X,Y)$ [$\langle 1\rangle 1.5$]

$\langle 3\rangle 4.$ Q.E.D.

$$\begin{array}{lll} mid\,Y - mid\,X & & \\ = & x_0+x+d+\frac{y}{2}-(x_0+\frac{x}{2}) & [\langle 3\rangle 1, \langle 3\rangle 2] \\ = & \frac{x}{2}+d+\frac{y}{2} & \\ = & \frac{x+y}{2}+d & \\ \leq & \frac{k-d}{2}+d & [\langle 2\rangle 1] \\ = & \frac{k+d}{2} & \\ \leq & \frac{k+dist(X,Y)}{2} & [\langle 3\rangle 3] \end{array}$$

8.6 Correctness Proof

We use the definition of (A1)-(A7) from section 4.3 and (C1)-(C3) from section section 4.4.

Lemma (L1): When (C1-3) and (A1-A7) are valid, then

$$\forall k \forall p,q \in \mathcal{P} \ : \ p,q \ correct \ at \ t^k \rightarrow |C_p(t^k) - C_q(t^k)| \leq \delta_S$$

Proof: We show this lemma by induction over k.
$k=0$: (L1) is valid by assumption *initial deviation* (A1).
$k \rightarrow k+1$: By the induction assumption $\|I^k_\Lambda(t^k)\| \leq \delta_S + 2\Lambda$ holds (L2). We can assume that $t^{k+1}_p \leq t^{k+1}_q$. Let $\mathcal{I}^k_p = [x_0, x_1]$, $\mathcal{I}^k_q = [y_0, y_1]$, and $c \triangleq t^{k+1}_q - t^{k+1}_p$. Let $\mathcal{I}^{k'}_p \triangleq [x_0 + H_p(t^{k+1}_q) - H_p(t^{k+1}_p), x_1 + H_p(t^{k+1}_q) - H_p(t^{k+1}_p)]$. Hence, $C_p(t^{k+1}_q) = mid(\mathcal{I}^{k'}_p)$ and $\mathcal{I}^{k'}_p \subseteq [x_0 + c(1-\rho), x_1 + c(1+\rho)]$, because the drift of p's hardware clocks is bounded by ρ. Furthermore, the length of interval $I \triangleq [min\{y_0, x_0 + c(1-\rho)\}, max\{y_0, x_0 + c(1+\rho)\}]$ is bounded by $\delta_S + 2\Lambda + 2\rho(t^{k+1}_q - t^k)$, because of (C1) and (L2). The distance between $\mathcal{I}^k_q$ and $\mathcal{I}^{k'}_p$ is bounded by $v\Lambda + 2\rho(t^{k+1}_q - t^{k+1}_p)$, because of (C2). We can conclude with the proof of principle P1 that

$$\begin{array}{lll} |C_p(t^{k+1}_q) - C_q(t^{k+1}_q)| & & \\ (1.1) & = & |mid(\mathcal{I}^{k'}_p) - mid(\mathcal{I}^k_q)| \\ (1.2) & \leq & \frac{\delta_S + 2\rho(t^{k+1}_q - t^k) + 2\Lambda + v\Lambda + 2\rho(t^{k+1}_q - t^{k+1}_p)}{2} \quad |C_p(t^k) - C_q(t^k)| \\ (2.1) & \leq & |C_p(t^{k+1}_q) - C_q(t^{k+1}_q)| + 2\rho(t^{k+1} - t^{k+1}_q) \\ (2.2) & \leq & \frac{\delta_S + 2\rho(t^{k+1}_q - t^k) + 2\Lambda + v\Lambda + 2\rho(t^{k+1}_q - t^{k+1}_p)}{2} + 2\rho(t^{k+1} - t^{k+1}_q) \\ (2.3) & = & \frac{\delta_S + 2\rho(t^{k+1} - t^k) + 2\Lambda + v\Lambda + 2\rho(t^{k+1} - t^{k+1}_p)}{2} \\ (2.4) & \leq & \frac{\delta_S + (2+v)\Lambda + 2\rho(r_{max} + \beta)}{2} \\ (2.5) & = & \frac{\delta_S + \delta_S}{2} \\ (2.6) & = & \delta_S \end{array}$$

Theorem *Generic Proof: When (C1-3), (A1-7) are valid and processes p,q are correct at t,*

$$|C_p(t) - C_q(t)| \leq \delta$$

Proof: There exists an k so that $t^k \leq t < t^{k+1}$. We can assume that $t_p^{k+1} \leq t_q^{k+1}$. We consider first the case that $t^k \leq t < t_p^{k+1}$. With lemma (L1) we can conclude that $|C_p(t) - C_q(t)| \leq |C_p(t^k) - C_q(t^k)| + 2\rho r_{max} \leq \delta$. Second, we consider the case that $t_p^{k+1} \leq t < t_q^{k+1}$. $C_p^{k+1}(t_p^{k+1}) \in I^k(t_p^{k+1})$ by (C3). Therefore, $|C_p(t) - C_q(t)| \leq |C_p(t^k) - C_q(t^k)| + 2\rho(t - t^k) \leq \delta$ holds. Third, we consider the case that $t_q^{k+1} \leq t$. Similarly to the proof of lemma (L1) one can show that $|C_p(t) - C_q(t)| \leq \delta$. $\square$

References

[1] F. Cristian. Probabilistic clock synchronization. *Distributed Computing*, 3:146–158, 1989.

[2] L. Lamport and P. M. Melliar-Smith. Synchronizing clocks in the presence of faults. *Journal of the ACM*, 32(1):52–78, Jan 1985.

[3] J. Lundelius-Welch and N. Lynch. A new fault-tolerant algorithm for clock synchronization. *Information and Computation*, 77(1):1–36, 1988.

[4] D. L. Mills. Internet time synchronization: the network time protocol. *IEEE Trans. Communications*, 39(10):1482–1493, Oct 1991.

[5] P. Miner. Verification of fault-tolerant clock synchronization systems. Technical Report TP-3349, NASA, Nov 1993.

[6] F. Schmuck and F. Cristian. Continuous clock amortization need not affect the precision of a clock synchronization algorithm. In *Proceedings of Ninth Annual ACM Symposium on Distributed Computing*, 1990.

[7] F. Schneider. Understanding protocols for Byzantine clock synchronization. Technical Report 87-859, Dept of Computer Science, Cornell University, Aug 1987.

[8] N. Shankar. Mechanical verification of a schematic byzantine clock synchronization algorithm. Technical Report CR-4386, NASA, July 1991.

[9] T. K. Srikanth and S. Toueg. Optimal clock synchronization. *Journal of the ACM*, 34(3):626–645, Jul 1987.

Symbol	Meaning
A_p^i	adjustment to p's hardware clock in round i
β	maximum difference between t_p^i and t_q^i
C_p	virtual clock of process p
$\mathcal{C}_p(T, q)$	q's estimate of $p's$ clock at local time T
$\mathcal{C}_p^{m_1,m_2}$	approximation of $p's$ clock given message pair (m_1, m_2)
C_p^i	virtual clock of process p in round i
$cfn(p, \Theta_p^i, E_p^i)$	convergence function used by p
$\mathcal{CT}$	set of clock time values
D	maximum adjustment of virtual clocks
δ	maximum deviation between virtual clocks
δ_S	maximum deviation between virtual clocks at the start of a round
Λ	a priori given maximum reading error
$\mathcal{E}_p$	error bound for $\mathcal{C}_p$
$\mathcal{E}_p^{m_1,m_2}$	error bound for $\mathcal{C}_p^{m_1,m_2}$
E_p^{i+1}	error bounds on clock readings Θ_p^{i+1}
G	maximum virtual clock discontinuity
H_p	hardware clock of process p
$I^k(t)$	interval spanned by correct virtual clocks at time t
$\mathcal{I}_p^k$	approximation of I^k by process p
k	max. number of messages sent by a process per round
$\|J\|$	length of interval J
m, m_1, m_2	messages sent by processes in $\mathcal{P}$
$max(m)$	calculated maximum transmission delay of m
$max^{m_1}(m_2)$	maximum transmission delay for m_2 given message pair (m_1, m_2)
min	minimum (real-time) transmission delay
N	number of processes, i.e. $N = \vert\mathcal{P}\vert$
P	clock time duration of a round
$\mathcal{P}$	set of (time-server) processes
p, q, r, s	processes, i.e. $p, q, r, s \in \mathcal{P}$
r_{min}	minimum (real-time) duration of a round
r_{max}	maximum (real-time) duration of a round
R	maximum enveloppe discontinuity
$R(m)$	receive time stamp of message m
$rank(p)$	rank of process p
ρ	maximum drift rate of hardware clocks
ρ_e	envelope rate of virtual clocks
ρ_v	maximum drift rate of virtual clocks
$\mathcal{RT}$	set of real time values
σ	maximum scheduling delay
$S(m)$	send time stamp of message m
T, T_0, T_1, T_2	virtual clock times
t, u, t_0, t_1	real time values
t^i	start of round i for all correct processes
t_p^i	start of round i for process p
T_p^{i+1}	local time at end of round i of process p
Θ_p^{i+1}	$p's$ approximation of all virtual clocks at start of round i
U	duration of a cycle
Z	duration of a slot