# An Optimal Internal Clock Synchronization Algorithm*

Christof Fetzer and Flaviu Cristian
Department of Computer Science & Engineering
University of California, San Diego
La Jolla, CA 92093–0114

## Abstract

We propose an optimal convergence function for achieving fault-tolerant, internal clock synchronization in the presence of arbitrary process and clock failures. The *differential fault-tolerant midpoint* convergence function guarantees an optimal maximum correction, an optimal maximum drift rate, and an optimal maximum deviation.

The proposed convergence function is simple and easy to compute. It bounds the maximum drift rate of correct clocks by the maximum drift rate of a correct hardware clock. The maximum correction is limited by the maximum drift between two correct hardware clocks during one round. The maximum deviation is approximately $4\Lambda + 4\rho r_{max}$, where $\Lambda$ is the maximum remote clock reading error, $\rho$ is the maximum drift rate of a correct hardware clock and $r_{max}$ is the maximum duration of a synchronization round.

## 1 Introduction

Tight internal clock synchronization is essential for many real-time and fault-tolerant applications. Internal clock synchronization requires that (1) at any time the deviation between two correct clocks be bounded by a constant $\delta$ (called the *maximum deviation*) and (2) the drift rate of clocks with respect to real-time be bounded by a constant $\rho_v$. Clock synchronization is a non-trivial problem because of the need to tolerate failures. Since in this paper we are only interested in internal clock synchronization algorithms capable of masking arbitrary clock and process failures, when we talk about a synchronization algorithm, we mean an internal clock synchronization algorithm tolerant of arbitrary failures.

Most synchronization algorithms can be described as instances of a single abstract, generic clock synchronization algorithm by using the notion of a convergence function introduced in [6]. This generic algorithm can be succintly described as follows: at the end of each synchronization round each process reads the clocks of all processes and then adjusts its clock value for the next round by applying a *convergence function* to the clock readings of the current round. A synchronization algorithms which can be obtained from the above generic algorithm by instantiating some concrete function for the abstract notion of a convergence function will be termed a *convergence function based* algorithm.

At any point in time, the value of a synchronized clock is defined as the sum of the current value of a hardware clock and an adjustment variable. Because hardware clocks drift apart from real-time and from each other, the adjustment variable of a clock has to be updated periodically. The maximum change of a clock value that a synchronization algorithm can cause to occur during an adjustment is called the *maximum correction*. The adjustments of the clocks together with the drift of the hardware clocks introduce an error when a synchronized clock is used to measure a time interval. This error can be bounded by a constant part, which is called the *maximum discontinuity*, and a part which depends on the length of the time interval which is measured. This second part is called the *maximum drift rate* of the synchronized clocks.

When clocks are adjusted discretely, that is, the adjustment variables are only changed once per round, the synchronized clocks cannot be guaranteed to be monotonic. When using a convergence function with an optimal maximum correction the non-monotonicity of tightly synchronized clocks becomes in general negligible, since the time to read a clock is typically longer than the optimal maximum correction. The proposed convergence function is also useful for synchronization

*This work 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 algorithms based on statistical remote clock reading [5]. Since such statistical reading methods do not provide any a priori or computed upper bound on the error made when reading a remote clock, the synchronized clocks can be repeatedly adjusted forth and back with large corrections, because of excessive clock reading errors. By minimizing the correction of the clocks this undesired behavior can be reduced.

We propose a *differential fault-tolerant midpoint* convergence function which provides an optimal maximum correction, an optimal maximum drift rate, and an optimal maximum deviation between clocks. The differential fault-tolerant midpoint convergence function is based on the fault-tolerant midpoint function proposed in [4], and improves upon it, since the fault-tolerant midpoint function, as proposed in [4], does not provide an optimal maximum correction, an optimal maximum drift, or an optimal maximum deviation.

## 2 System Model

We consider a distributed system consisting of nodes hosting time server processes. Each such process has access to the local hardware clock of its node. In this paper we assume the existence of a remote clock reading method which can always read a correct remote clock with an error not greater than an a priori given *maximum reading error*. This allows us to abstract from the communication mechanisms used by the time server processes to read each other's clocks. We also require the existence of an upper bound $r_{max}$ on the maximum time between two successive clock synchronizations. This allows us to abstract from considering execution time and scheduling delay issues. We denote the set of time server processes by $\mathcal{P}$ and the number of such processes by $N \triangleq |\mathcal{P}|$.

### 2.1 Clocks and Clock Readings

We represent a hardware clock as a total mapping from real-time to clock time: $H_p$ denotes process $p$'s hardware clock. All hardware clocks have a finite granularity, but we assume that this granularity is negligible. A hardware clock can drift apart from real time, but the drift of correct hardware clocks is bounded by the *maximum drift rate* $\rho$. Let constant $t^0$ refer to the earliest point in real time for which the clocks must be synchronized. Process $p$'s hardware clock is correct at time $t_1$, when for all intervals $[s,t] \subseteq [t_0, t_1]$ the *bounded drift* condition holds:

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

A *hardware clock failure* occurs when the bounded drift condition is violated. For most quartz clocks available in modern computers, the maximum drift rate $\rho$ is of the order of $10^{-6}$. Since $\rho$ is such a small quantity, we will ignore terms of the order of $\rho^2$ or higher, for example we will equate $(1+\rho)^{-1}$ with $(1-\rho)$ and $(1-\rho)^{-1}$ with $(1+\rho)$.

In general, processes do not directly manipulate the value or the speed of hardware clocks. Instead, each process maintains a *virtual clock* by adding to the underlying hardware clock an *adjustment function*. In this paper we consider only discrete adjustment functions: these are step functions of time. We use the term *clock* to denote a virtual clock. Process $p$'s clock is represented by a total mapping $C_p$ from real time to clock time.

To achieve synchronization, processes estimate the clocks of other processes by using a *remote clock reading method*. Such a method provides for any process $p$ an *approximation* of any process' clock. We denote the approximation that $p$ computes at its local time $T$ of the clock of process $q$ by $\mathcal{C}_q(T,p)$. We assume that the remote clock reading method bounds the clock reading error by $\Lambda$, the *maximum reading error*: when $T = C_p(t)$ and processes $p$ and $q$ are correct at time $t$, then

$$|C_q(t) - \mathcal{C}_q(T,p)| \leq \Lambda.$$

We assume the local reading error is negligible, i.e. when process $p$ is correct at time $t$, then

$$C_p(t) = \mathcal{C}_p(T,p).$$

### 2.2 Failure Hypotheses

Each correct process has by definition a correct hardware clock. The total number of processes participating in the clock synchronization must be at least $3F+1$ [2], where $F$ denotes the maximum number of processes that can be faulty. For simplicity, we assume that no failed process recovers. We assume that processes and clocks fail in arbitrary ways. The reading error for approximating a clock of a process which has suffered an arbitrary failure is not bounded.

## 3 Requirements

An internal clock synchronization service can be specified by two requirements. The first, called the *bounded deviation* requirement, bounds the deviation between correct clocks by a constant $\delta$ (*maximum deviation*): *for any processes $p$ and $q$ correct at time $t \geq t^0$,*

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

Recall that constant $t^0$ refers the earliest point in real time for which the clocks must be synchronized.

The second, the *clock drift* requirement, bounds the drift rate of correct clocks by a constant $\rho_v \ll 1$: *for any process $p$ correct in interval $[t, u]$, where $t_0 \leq t \leq u$,*

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

The constant $G$ is called the *maximum discontinuity* of a clock. It accounts for the granularity and the adjustments of the clocks. Synchronization algorithms with $\rho_v = \rho$ have an *optimal* clock drift rate [7].

A synchronization algorithm is *correct* when it satisfies the *bounded deviation* and *clock drift* requirements above.

## 4 Overview

Clocks are synchronized in rounds, where correct processes approximately agree on the time when a round starts. At the end of each round, after reading all clocks, each correct clock is adjusted so that it becomes approximately synchronized with the others. The *clock readings* of a process $p$ are the approximations of the values displayed by all clocks by $p$ at the end of a round. These are represented by a mapping from processes to clock values. For example, for a clock reading $\Theta$ the approximation of process $q$'s clock is $\Theta(q)$.

The adjustment applied by a process to its clock at the end of a round is computed by applying a convergence function to the clock readings obtained in that round. Thus, a convergence function $cfn$ maps a process and a clock readings mapping to a clock value. When $\Theta$ represents the clock readings mapping of process $p$ at the end of the current round, then $p$'s clock value at the start of the next round is $cfn(p, \Theta)$.

### 4.1 Algorithm

We first recall how a convergence function can be used to synchronize clocks. A pseudocode description of a clock synchronization algorithm that uses a generic convergence function $cfn$ is given in figure 1. Each process executes a copy of this algorithm. The function $C$ (line 4) represents the clock of the executing process. Process $p$'s clock is defined by the sum of $p$'s current adjustment value stored in the variable

```
(1)   const  Time D, R;
(2)   var    Time A, T;
(3)
(4)   function Time C() { return H() + A; }
(5)
(6)   function void init () {
(7)       (A,T) = InitialAdjustement();
(8)       schedule (synchronizer, R, T − D);
(9)   }
(10)
(11)  function void synchronizer() {
(12)      Time Θ[N];
(13)
(14)      parbegin {
(16)          ∀q ∈ P : Θ[rank(q)] = C_q(T, myid());
(17)      } parend
(18)      A = A + (cfn(myid(), Θ) - T);
(19)      T = T + R;
(20)  }
```

Figure 1: Clock Synchronization Protocol.

$A$ (line 2) and the current value of the hardware clock which is returned by the function $H$ (line 4).

The length of a round is denoted by the constant $R$. The function *init* determines the initial adjustment value and the first time the clock has to be readjusted (line 7). The clocks are periodically adjusted by the execution of the *synchronizer* function (lines 11-20). This function is scheduled for execution every $R$ time units starting at local time $T - D$ (line 8), where constant $D$ denotes the maximum clock time needed to read a remote clock. In other words, when we neglect the execution time except for the time needed to read the remote clocks, the clock is adjusted at local time $T$ or before $T$. The scheduling is based on the clock represented by the local function $C$. Function *myid* returns the id of the executing process and the function call $C_q(T, myid())$ returns the approximation of process $q$'s clock at time $T$. Recall that the system model bounds the maximum error approximating a correct remote clock by $\Lambda$. Array $\Theta$ contains the approximations of the values of all clocks at local time $T$ (line 16). The convergence function calculates the value of the clock of a process at the start of the next round (line 18): the old value $T$ is replaced by $cfn(myid(), \Theta)$. This is achieved by changing the adjustment value $A$ to A+$cfn(myid(), \Theta) - T$. Finally, $T$ is incremented and refers now to the end of the next round (line 19).

## 4.2 Notation

To analyze this algorithm, we use the following well-known notations: $t_p^k$ denotes the start of $p's$ $k$-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 $A_p^k$. The clock in round $k$ of process $p$ is defined as the sum of the current value of $p$'s hardware clock and the adjustment value $A_p^k$:

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

At the end of each round each process tries to estimate the values of all clocks. Two successive rounds can overlap, i.e. a process can start round $k+1$ while another process is still in round $k$. At the end of round $k$ process $p$ tries to approximate the remote clocks with respect to the adjustment values of round $k$ and not with respect to the adjustment values of round $k+1$. This is achieved by using the concept of a round clock. The *round clock* $C_p^k$ of process $p$ for round $k$ is defined as:

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

We denote $T_p^{k+1}$ the end of round $k$ with respect to round clock $C_p^k$: $T_p^{k+1} \triangleq C_p^k(t_p^{k+1})$.

## 4.3 Assumptions

To prove that a convergence function based internal clock synchronization algorithm is correct one has to make the following standard assumptions [6] in addition to those described in our system model section.

### 4.3.1 Initialization

At the start of the first round, all correct clocks must be within $\delta_S$ of each other. Let term $t^k$ denote the real time when all correct processes have just started their $k$-th synchronization round: $t^k \triangleq max\{t_p^k \mid p \ correct \ at \ time \ t_p^k\}$. Formally, the first assumption **(A1)** can be expressed as: *For any processes p and q that are correct at real time* $t^0$,

$$|C_p^0(t^0) - C_q^0(t^0)| \leq \delta_S.$$

### 4.3.2 Interval Constraints

Assumption **(A2)** bounds the length of a clock synchronization round by given constants $r_{min}$ and $r_{max}$: *For any process p that is correct at time* $t_p^{k+1}$,

$$r_{min} \leq t_p^{k+1} - t_p^k \leq r_{max}.$$

Assumption **(A3)** bounds the real-time delay between the beginning of the same round for different processes by an a priori given constant $\beta$. *For any processes p and q correct at times* $t_p^k$ *and* $t_q^k$, *respectively:*

$$|t_p^k - t_q^k| \leq \beta.$$

The overlap of rounds is restricted by assumption **(A4)**:

$$\beta \leq r_{min}.$$

## 5 Differential Midpoint Function

In this section we introduce the *differential fault-tolerant midpoint* convergence function (DFTM). This convergence function improves the fault-tolerant midpoint function proposed in [4] by providing optimal maximum correction, optimal maximum drift rate, and optimal maximum deviation.

### 5.1 Definitions

Let $\mathcal{CT}$ refer to the set of clock values. A clock reading returns a clock value for each process, thus, it is a mapping with signature $\mathcal{P} \rightarrow \mathcal{CT}$. We refer to the set of all clock readings by $\mathcal{CR}$. Function *sort* takes a clock reading as input and returns a sorted array of the given clock values. Since an array of clock values is represented by a function with signature $\{0,..,N-1\} \rightarrow \mathcal{CT}$, the signature of *sort* is:

$$sort : \mathcal{CR} \rightarrow (\{0,..,N-1\} \rightarrow \mathcal{CT})$$

Function *sort* guarantees for each clock reading $\Theta$ that $sort(\Theta)$ is a permutation of $\Theta$ and that the values in the returned array are nondecreasing:

$$\forall \Theta \in \mathcal{CR} : \forall i \in \{0,..,N-2\} : \\ sort(\Theta)(i) \leq sort(\Theta)(i+1).$$

The *sign* function is defined by:

$$sign(x) \triangleq \begin{cases} 1 & if \ x > 0 \\ 0 & if \ x = 0 \\ -1 & if \ x < 0 \end{cases}$$

The midpoint of an interval $[x, y]$ is defined by:

$$mid(x,y) \triangleq \frac{x+y}{2}.$$

A convergence function takes as input parameters the identity of a process $p$ and a clock reading and returns the new value of $p$'s clock. Thus, the signature of a convergence function $cfn$ is:

$$cfn : (\mathcal{P} \times \mathcal{CR}) \rightarrow \mathcal{CT}.$$

## 5.2 Fault-Tolerant Midpoint Function

Because the proposed convergence function is based on the fault-tolerant midpoint convergence function (FTM) described in [4], we first recall the basic ideas of FTM and also give examples illustrating that FTM is not optimal. The FTM function is defined as:

$$FTM(p, \Theta) \triangleq mid(sort(\Theta)(F), sort(\Theta)(N - F - 1))$$

The intuition behind this convergence function is as follows; because at most $F$ clocks are faulty and the clock reading error for correct clocks is at most $\Lambda$, a process $p$ can reject the $F$ smallest and the $F$ greatest clock values to ensure that the length of the interval $I_p \triangleq [sort(\Theta)(F), sort(\Theta)(N-F-1)]$ spanned by the remaining clock values is bounded. In particular, the following *correct interval condition* holds: there exist two correct processes $q$ and $r$ such that the $F+1$st smallest approximated clock value $sort(\Theta)(F)$ is at most $\Lambda$ smaller than $q$'s clock value and the (N-F-1)th greatest clock value $sort(\Theta)(N-F-1)$ is at most $\Lambda$ greater than $r$'s clock value because at most $F$ of the $F+1$ smallest (or greatest) clock values can belong to faulty clocks. This implies that the length of the interval $I_p$ is bounded by the maximum deviation between correct clocks $\delta$ plus $2\Lambda$, where the last term accounts for the clock reading errors. Recall that the number of clocks participating in synchronization must be at least $3F+1$. Thus, there exists at least $2F+1$ correct clocks and a process can reject at most the $F$th smallest and the $F$th greatest correct clock values, but it cannot rejecting the $F+1$th smallest correct clock value. This bounds the distance between the intervals $I_p$ and $I_s$ of two correct processes $p$ and $s$ by about $2\Lambda$, and this in turn can be used together with the bounded length of the two intervals to show that $FTM$ bounds the deviation between any correct clocks.

The FTM does not provide an optimal drift rate for the synchronized clocks, because it can happen that all clocks have initially the same value, all hardware clocks drift with the maximum drift rate $\rho$, and the reading error is always $+\Lambda$. In this case all clocks are incremented in each round by $\Lambda$. In other words, the effective drift rate during each round of length $r$ is $\rho + \frac{\Lambda}{r}$. The maximum drift rate of clocks synchronized by FTM is therefore at least:

$$\rho_v \geq \rho + \frac{\Lambda}{r_{min}}.$$

Recall that $\delta_S$ denotes the maximum deviation of the clocks of two correct processes at the start of a round. At the end of a round the clocks of two correct processes can be up to $\delta_S + 2\rho r_{max}$ apart and the reading error can be up to $\Lambda$. A process $p$ can thus adjust its clock by up to $c \triangleq \delta_S + 2\rho r_{max} + \Lambda$ because $p$'s own clock value can be $c$ smaller than the approximations of all other clocks and $p$ will hence increase its clock by $c$. The optimal maximum correction is $2\rho r_{max}$ [3]. Thus, the FTM does not provide optimal maximum correction.

Let us now show why the maximum deviation provided by the fault-tolerant midpoint convergence function is not optimal. We partition the set of processes into three sets $L$, $M$, and $U$ with $|L| = |U| = F$ and $|M| = N - 2F > F$, such that the values of all hardware clocks in each set are the same at all times. Let us assume that the hardware clocks of processes in $M$ drift with maximum allowable drift rate $\rho$, the hardware clocks of processes in $L$ drift with minimum allowable drift rate $-\rho$, and the hardware clocks of processes in $U$ are faulty and they drift with $2\rho$. Furthermore, suppose that at the start of the first round, the clocks of processes in $M$ show a value which is exactly $\delta_S$ greater than the ones in $L$ and the clocks of processes in $U$ have initially a value which is exactly $\delta_S$ greater than the ones in $M$. Let now the processes in $M$ read each others clocks with a maximum error of $+\Lambda$, let them adjust their clocks before the processes in $L$, and recall that rounds are at most $r_{max}$ long. The clocks in $M$ are adjusted by $\frac{\Lambda}{2}$ because they reject the clocks of processes in $L$ and $U$. Thus, the maximum deviation between correct clocks is at least $4.5\Lambda + 4\rho r_{max}$. The tight lower bound for the maximum deviation between clocks is $\delta_{opt} = 4\Lambda + 4\rho r_{max}$ [3]. Therefore, the maximum deviation of the fault-tolerant midpoint convergence function FTM is not optimal.

## 5.3 Differential Fault-Tolerant Midpoint Function

To avoid non optimal drift rate and non optimal maximum deviation, a convergence function has to ensure that the new clock value of a process $p$ is neither greater nor smaller than the minimum and maximum of the old correct clock values. That is, we must ensure that there exists two correct processes so that the round clocks which they use during the last round show a value not smaller, respectively not greater, than $p$'s new clock value at the time $p$ adjusts its clock (we call this the *nested adjustment condition*). Because of condition correct interval, there exists two correct processes $q$ and $r$ such that $q$'s clock value is at most $\Lambda$ greater than $min(I_p)$ and $r$'s clock value is at most $\Lambda$ smaller than $max(I_p)$. When process $p$ would only adjust its clock to values which are at least $\Lambda$ apart from the boundaries of the interval $I_p$, then $p$ could ensure the nested adjustment condition. The problem is that the length of $I_p$ could be smaller than $2\Lambda$. Let us assume that $p$ adjusts its clock at local time $T$. Because a process $p$ can read its own clock with a negligible error, $p$ can extend $I_p$ by $[T-\Lambda, T+\Lambda]$ without violating the correct interval condition. Therefore, the extended interval has always a length of $2\Lambda$ and the midpoint of this interval guarantees the nested adjustment condition (see [1] for more details).

We use this idea to define an extended midpoint function *emid*. It computes, for a given time $T$ and an array $Y$ of $N$ clock values, the midpoint of the union of the intervals $[Y(F), Y(N-F-1)]$ and $[T-\Lambda, T+\Lambda]$:

$$emid(T,Y) \triangleq \quad mid(min\{T-\Lambda, Y(F)\},\ max\{T+\Lambda, Y(N-F-1)\}).$$

To bound the correction of clocks, we use the fact that two correct clocks can drift apart during one round by at most $2\rho r_{max}$. This implies that each process has to adjust its clock by up to $2\rho r_{max}$, because the following situation could occur: the clocks are partitioned into three sets $L$, $M$, and $U$ like above; it is then possible that no process in $M$ using *emid* changes its clock and hence the processes in $L$ have to adjust their clocks by $2r_{max}$. The direction towards which a process should adjust its clock is determined by the extended midpoint function. We use this idea in the definition of function *dmp* which has the same parameters as the function *emid*.

$$dmp(T,Y) \triangleq \quad if\ |emid(T,Y)-T| \leq 2\rho r_{max}\ then\ emid(T,Y)\ \text{else}\ T + sign(emid(T,Y)-T)2\rho r_{max}.$$

The differential fault-tolerant midpoint convergence function DFTM makes use of *dmp* to calculate the new clock value for a given process $p$ and a clock reading $\Theta$, where process $p$'s own clock shows $\Theta(p)$ at the end of the round and the function *sort* returns a sorted array of the clock values in $\Theta$:

$$DFTM(p,\Theta) \triangleq dmp(\Theta(p), sort(\Theta)).$$

## 5.4 Correctness Proofs

We show in the Appendix why the differential fault-tolerant midpoint convergence function is correct, i.e. a clock synchronization algorithm using DFTM and which guarantees assumptions (A1)-(A4) satisfies the clock drift and bounded deviation requirements. First, we prove that the extended midpoints calculated by two correct processes are at most $\delta_S$ apart, where $\delta_S = 4\Lambda + 2\rho r_{max} + 2\rho\beta$. Second, we use this proof to show that it is sufficient to adjust a clock by $2\rho r_{max}$. The problem here is that the restriction of the correction of the clocks by $2\rho r_{max}$ does not allow to use the bound for the deviation between the extended midpoints directly, because the clocks are not necessarily adjusted to the extended midpoints. The proof given in the Appendix is based on the following idea: let $p$ and $q$ be two correct processes and be $M_p$, $M_q$ the extended midpoints calculated by $p$, $q$ at the end of round $k$. Furthermore, let us assume that $p$ and $q$ change their clocks from their old values $O_p, O_q$ to the new values $N_p$, $N_q$ at the same point in real-time and that $M_p \leq M_q$. The deviation between $N_p$ and $N_q$ could only be greater than $\delta_S$ when $N_p$ or $N_q$ are not in interval $[M_p, M_q]$, because the deviation between $M_p$ and $M_q$ is bounded by $\delta_S$. In the worst case $N_p < M_p < M_q < N_q$ holds. The deviation between $N_p$ and $N_q$ is nevertheless bounded by $\delta_S$, because $O_p < N_p$, $N_q < O_q$, and $O_q - O_p \leq \delta_S + 2\rho r_{max}$ and $p$ adjusts its clock by $2\rho r_{max}$ towards $M_p$.

## 6 Properties

We summarize in this section the properties of the proposed convergence functions. In [3] the following lower bounds for the optimal maximum deviation $\delta_{opt}$ and the optimal maximum correction $K_{opt}$ are derived:

$$\delta_{opt} \geq 4\Lambda + 4\rho r_{max} \tag{1}$$
$$K_{opt} \geq 2\rho r_{max} \tag{2}$$

By definition of the function *dmp*, a clock is at most changed by $2\rho r_{max}$ and thus the maximum correction of the DFTM is optimal. Because $\rho$ is very small and $\beta$ is in general bounded by $\delta(1+\rho)$, [3] neglects summands of the form $2\rho\beta$. We derive in the Appendix that the deviation of correct clocks synchronized by the DFTM is bounded by $4\Lambda + 4\rho r_{max} + 2\rho\beta$. Hence the maximum deviation is optimal, because we neglect summands of the form $2\rho\beta$. The drift rate of correct clocks is limited by $\rho$ (see Appendix) and therefore the maximum drift rate of clocks is optimal [7]. We derive in the Appendix that the maximum discontinuity $G$ of the DFTM is bounded by $4\Lambda + 4\rho r_{max} + 2\rho\beta$.

## 7 Conclusion

This paper proposes a new differential fault-tolerant midpoint convergence function for fault-tolerant internal clock synchronization in the presence of arbitrary process and clock failures. This function achieves an optimal maximum correction, an optimal maximum drift rate, and an optimal maximum deviation. The function is simple and easy to compute.

The proposed convergence function bounds the drift rate $\rho_v$ of correct clocks by $\rho$, where $\rho$ is the maximum drift rate of correct hardware clocks. The maximum correction is limited by $2\rho r_{max}$, where $r_{max}$ is the maximum duration of a clock synchronization round. The maximum deviation and maximum discontinuity achieved are both $4\Lambda+4\rho r_{max}+2\rho\beta$, where $\Lambda$ is the maximum clock reading error.

## References

[1] F. Cristian and C. Fetzer. Fault-tolerant internal clock synchronization. In *Proceedings of the Thirteenth Symposium on Reliable Distributed Systems*, Dana Point, Ca., Oct 1994.

[2] D. Dolev, J. Y. Halpern, and R. Strong. On the possibility and impossibility of achieving clock synchronization. *Journal of Computer and System Science*, 32(2):230–250, 1986.

[3] C. Fetzer and F. Cristian. Optimal convergence function based clock synchronization. In *Proceedings of Fourtheenth ACM Symposium on Principles of Distributed Computing*, Ottawa, CA, Aug 1995.

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

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

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

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

## 8 Appendix

### 8.1 General Assumptions

Since $\rho$ is in practice a very small quantity, we ignore terms of the order of $\rho^2$ or higher, for example we equate $(1+\rho)^{-1}$ with $(1-\rho)$ and $(1-\rho)^{-1}$ with $(1+\rho)$.

### 8.2 Constraints for Constants

The correctness proof of a clock synchronization algorithm based on the DFTM convergence function requires that constants $\delta$ and $\delta_S$ satisfy the constraints (A5) and (A6) given below.

$$\delta_S \geq 4\Lambda + 2\rho r_{max} + 2\rho\beta. \qquad \text{(A5)}$$

$$\delta \geq 4\Lambda + 4\rho r_{max} + 2\rho\beta. \qquad \text{(A6)}$$

### 8.3 Definitions

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

$$I^k(t) \triangleq [min\{C_s^k(t) \mid s\ correct\ at\ t\}, \\ max\{C_s^k(t) \mid s\ correct\ at\ t\}].$$

Since a correct process $q$ estimates a correct remote clock $s$ with an error of up to $\Lambda$, we define the $\Lambda$-extended interval of correct clocks $I_\Lambda^k(t)$ to be

$$I_\Lambda^k(t) \triangleq [min\{C_s^k(t) - \Lambda \mid s\ correct\ at\ t\}, \\ max\{C_s^k(t) + \Lambda \mid s\ correct\ at\ t\}].$$

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

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

$$dist(X,Y) \triangleq \begin{array}{l} if\ (max(X) < min(Y))\ then \\ \quad min(Y) - max(X) \\ else\ if\ (max(Y) < min(X))\ then \\ \quad min(X) - max(Y) \\ \text{else} \\ \quad 0. \end{array}$$

We denote process $p$'s approximation of the interval of correct clocks by $\mathcal{I}_p^k$. Let $\Theta$ denote $p$'s clock readings mapping at the end of round $k$: $\Theta(q) \triangleq C_q^k(T_p^{k+1}, p)$. The interval $\mathcal{I}_p^k$ is then:

$$\mathcal{I}_p^k \triangleq [min\{T_p^{k+1} - \Lambda, sort(\Theta)(F)\}, \\ max\{T_p^{k+1} + \Lambda, sort(\Theta)(N-F-1)\}].$$

## 8.4 Outline

A first lemma states that to bound the distance between the midpoints of the approximations $\mathcal{I}_p^k$ and $\mathcal{I}_q^k$ (assuming that processes $p$ and $q$ would adjust their clocks at the same point in real-time) it is sufficient to ensure that 1) each approximation $\mathcal{I}_p^k$ made by a correct process $p$ is included in the corresponding $\Lambda$-extended interval of correct 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 bounded by a constant $d$:

*Lemma L1:* $t = t_p^{k+1} = t_q^{k+1} \wedge p$ *and* $q$ *correct at* $t \wedge \|I_\Lambda^k(t)\| \leq \delta + 2\Lambda \wedge \mathcal{I}_p^k \subseteq I_\Lambda^k(t) \wedge \mathcal{I}_q^k \subseteq I_\Lambda^k(t) \wedge dist(\mathcal{I}_p^k, \mathcal{I}_q^k) \leq d \rightarrow |mid(\mathcal{I}_p^k) - mid(\mathcal{I}_q^k)| \leq \frac{\delta+2\Lambda+d}{2}$.

Theorem T1 states that the bounded deviation requirement holds for each of the differential midpoint convergence functions whenever assumptions (A1)-(A6) and the following conditions (C1)-(C3) hold for any two correct processes $p$ and $q$:

$$(C1) \qquad \mathcal{I}_p^k \subseteq I_\Lambda^k(t_p^{k+1})$$
$$(C2) \qquad dist(\mathcal{I}_p^k, \mathcal{I}_q^k) \leq 2\Lambda \ for\ t_p^{k+1} = t_q^{k+1}$$
$$(C3) \qquad C_p^{k+1}(t_p^{k+1}) \in I^k(t_p^{k+1})$$

We first show that conditions (C1) and (C2) are valid. Condition (C3) obviously implies that the drift rate of clocks is bounded by the drift rate of hardware clocks. We show (C3) in the proof that the drift rate of clocks is bounded by $\rho$.

Let $\Theta$ denote $p$'s readings of all clocks at the end of round $k$ and let $T_p^{k+1} = C_p^k(t)$. Because at most $F$ clocks are faulty and the reading error is at most $\Lambda$, there exist two correct processes $s$, $r$ such that $C_s(t) - \Lambda \leq sort(\Theta)(F)$ and $C_r(t) + \Lambda \geq sort(\Theta)(N - F - 1)$. Hence, condition (C1) holds.

The failure assumption guarantees that at least $2F + 1$ clocks are correct. Hence, processes $p$ and $q$ can reject at most the $F$ smallest and the $F$ greatest correct clock values, but both do not reject the $F + 1$th smallest correct clock value. Therefore, intervals $\mathcal{I}_p^k$ and $\mathcal{I}_q^k$ are at most $2\Lambda$ apart and condition (C2) holds, because the reading error for correct clocks is bounded by $\Lambda$.

## 8.5 Drift Rate and Discontinuity

ASSUME:
1. $T_u^p \triangleq max\{C_q(t_p^k) \mid q$ correct at $t_p^k\}$.
2. $T_l^p \triangleq min\{C_q(t_p^k) \mid q$ correct at $t_p^k\}$.
3. $I_\Lambda^k(t)$ and $\mathcal{I}_p^k$ defined as above, i.e. $\|\mathcal{I}_p^{k+j}\| \geq 2\Lambda$
4. $T_u^p - T_l^p \leq \delta_S$.
5. $p$ is correct at time $t \geq t_p^k$.
6. $C_p^k(t) = H_p(t) + A_p^k$.
7. The drift rate of $H_p$ and also of $C_p^k(t)$ is bounded by $\rho$.
8. $C_p(t) = C_p^k(t)$ for $t_p^k \leq t \leq t_p^{k+1}$.
9. $\mathcal{I}_p^k \subseteq I_\Lambda^k(t_p^{k+1})$.
10. $K \triangleq 2\rho r_{max}$; maximum correction.
11. $R \triangleq \delta_S$.
12. $G \triangleq K + R$; maximum discontinuity.

PROVE: $\forall p \in \mathcal{P} \forall s, t : t_p^0 \leq s \leq t \wedge p$ correct at $t$:
$(1 - \rho)(t - s) - G \leq C_p(t) - C_p(s) \leq (1 + \rho)(t - s) + G$.

PROOF:

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

PROOF:

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

PROOF: by induction over $j$.

CASE: $j = 0$

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

CASE: $j \rightarrow j + 1$

PROOF:

Because $\mathcal{I}_p^{k+j} \subseteq I_\Lambda^{k+j}(t_p^{k+j+1})$ (9), there exist two correct clocks $q$ and $r$ with

$\langle 4 \rangle 1$. $C_q^{k+j}(t_p^{k+j+1}) \geq max(\mathcal{I}_p^{k+j}) - \Lambda$ [9]

$\langle 4 \rangle 2$. $C_r^{k+j}(t_p^{k+j+1}) \leq min(\mathcal{I}_p^{k+j}) + \Lambda$ [9]

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

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

$\langle 4 \rangle 5$. Q.E.D.
$T_l^p + (t_p^{k+j+1} - t_p^k)(1 - \rho) \leq C_p^{k+j+1}(t_p^{k+j+1}) \leq T_u^p + (t_p^{k+j+1} - t_p^k)(1 + \rho)$ [$\langle 4 \rangle$4]

$\langle 2 \rangle 2$. PROVE: $\forall p \in \mathcal{P} : p$ correct at $t \geq t_p^k$:
$T_l^p + (t - t_p^k)(1 - \rho) \leq C_p(t) \leq T_u^p + (t - t_p^k)(1 + \rho)$

PROOF: $\langle 2 \rangle$1 and 7 implies $\langle 2 \rangle$2

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

PROOF:

$\langle 3 \rangle 1$. $C_p(t) - C_p(t_p^k) \leq T_u^p + (t - t_p^k)(1 + \rho) - T_l^p \leq (t - t_p^k)(1 + \rho) + R$ [4,$\langle 2 \rangle$2,11]

$\langle 3 \rangle 2$. $C_p(t) - C_p(t_p^k) \geq T_l^p + (t - t_p^k)(1 - \rho) - T_u^p \geq (t - t_p^k)(1 - \rho) - R$ [4,$\langle 2 \rangle$2,11]

$\langle 3 \rangle 3$. Q.E.D.
$(t - t_p^k)(1 - \rho) - R \leq C_p(t) - C_p(t_p^k) \leq (t - t_p^k)(1 +$

$\rho) + R$ [⟨3⟩1,⟨3⟩2]

⟨1⟩2. Q.E.D.

⟨2⟩1. We choose $k$ such that $t_p^k \leq s < t_p^{k+1}$.
The adjustment of clock $C_p$ at time $t_p^{k+1}$ is at most $K = G - R$.

⟨2⟩2. $(t_p^{k+1}-s)(1-\rho)-(G-R) \leq C_p(t_p^{k+1})-C_p(s) \leq (t_p^{k+1} - s)(1 + \rho) + (G - R)$ [7,8]

⟨2⟩3. $(t - t_p^{k+1})(1 - \rho) - R \leq C_p(t) - C_p(t_p^{k+1}) \leq (t - t_p^{k+1})(1 + \rho) + R$ [⟨1⟩1]

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

## 8.6 Lemma L1

⟨1⟩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:

⟨2⟩1. PROVE: $x + y \leq k - d$

PROOF:

$$\begin{array}{lll} x + y & & \\ = & (x_1 - x_0) + (y_1 - y_0) & [⟨1⟩1.2\text{-}3] \\ = & y_1 - x_0 - (y_0 - x_1) & [\text{reorder}] \\ \leq & k - d & [⟨1⟩1.2\text{-}4] \end{array}$$

⟨2⟩2. Q.E.D.

PROOF:

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

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

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

⟨3⟩4. Q.E.D.

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

## 8.7 Lemma L2

*Lemma L2:* When conditions (C1)-(C3) and assumptions (A1)-(A6) are valid, then
$\forall k \forall p, q$: *$p, q$ correct at $t^k$* $\rightarrow |C_p(t^k) - C_q(t^k)| \leq \delta_S$.

*Proof.* We show this lemma by induction over $k$.
Case $k = 0$: (L2) is valid by assumption *initial deviation* (A1).
Case $k \rightarrow k + 1$: First, we show that the extended midpoints defined by function *emid* of two correct processes is at time $t^{k+1}$ at most $\delta_S$ apart. Second, we show that the deviation between two correct clocks at the start of a round is bounded by $\delta_S$. The induction assumption guarantees that (L3): $||I_\Lambda^k(t^k)|| \leq \delta_S + 2\Lambda$. Without loss of generality, we can assume that $t_p^{k+1} \leq t_q^{k+1}$. Let $\mathcal{I}_p^k = [x_0, x_1]$, $\mathcal{I}_q^k = [y_0, y_1]$, and $c \triangleq t_q^{k+1} - t_p^{k+1}$. Moreover, let $\mathcal{I}_p^{k'} \triangleq [x_0 + H_p(t_q^{k+1}) - H_p(t_p^{k+1}), x_1 + H_p(t_q^{k+1}) - H_p(t_p^{k+1})]$, $M_p \triangleq mid(\mathcal{I}_p^{k'})$, and $M_q = mid(\mathcal{I}_q^k)$. Hence, $\mathcal{I}_p^{k'} \subseteq [x_0+c(1-\rho), x_1+c(1+\rho)]$, because the drift of $p$'s hardware clocks is bounded by $\rho$. Furthermore, the length of interval $I \triangleq [min\{y_0, x_0 + c(1 - \rho)\}, max\{y_1, x_0 + c(1 + \rho)\}]$ is bounded by $\delta_S + 2\Lambda + 2\rho(t_q^{k+1} - t^k)$, because of (C1) and (L3). The distance between $\mathcal{I}_q^k$ and $\mathcal{I}_p^{k'}$ is bounded by $2\Lambda + 2\rho(t_q^{k+1} - t_p^{k+1})$, because of (C2). We can conclude with lemma L1 that

$$\begin{array}{lll} |M_p - M_q| & & \\ (1.1) & = & |mid(\mathcal{I}_p^{k'}) - mid(\mathcal{I}_q^k)| \\ (1.2) & \leq & \frac{\delta_S+2\Lambda+2\rho(t_q^{k+1}-t^k)+2\Lambda+2\rho(t_q^{k+1}-t_p^{k+1})}{2} \end{array}$$

Second, we show that $|C_p(t^k)-C_q(t^k)| \leq \delta_S$. The idea hereby is that we can use (1.2) to bound $|C_p(t_q^{k+1}) - C_q(t_q^{k+1})|$ by $\delta_S$. When $p$ and $q$ can adjust their clock to their extended midpoint this is obviously true. Otherwise, $p$'s and/or $q$'s extended midpoint is more than $2\rho r_{max}$ apart from $p$'s/$q$'s clock value at $t_p^{k+1}/t_q^{k+1}$. We only consider the case where $q$ cannot adjust its clock to $M_q$. The proof when $p$ cannot adjust its clock to its extended midpoint is similar. We have to distinguish between two cases: (a) When $C_q(t_q^{k+1})$ has a value between $M_q$ and $C_p(t_q^{k+1})$ then the deviation $|C_p(t_q^{k+1}) - C_q(t_q^{k+1})|$ is obviously bounded by the bound given in (1.2). Therefore,

$$\begin{array}{lll} |C_p(t^k) - C_q(t^k)| & & \\ (2.1) & \leq & |C_p(t_q^{k+1}) - C_q(t_q^{k+1})| + 2\rho(t^{k+1} - t_q^{k+1}) \\ (2.2) & = & \frac{\delta_S+2\rho(t^{k+1}-t^k)+4\Lambda+2\rho(t^{k+1}-t_p^{k+1})}{2} \\ (2.3) & \leq & \frac{\delta_S+4\Lambda+2\rho r_{max}+2\rho\beta}{2} \\ (2.4) & = & \frac{\delta_S+\delta_S}{2} \\ (2.5) & = & \delta_S \end{array}$$

(b) When $M_q$ has a value between $C_p(t_q^{k+1})$ and $C_q(t_q^{k+1})$ then, because of condition (C3), $|C_p(t_q^{k+1}) - C_q(t_q^{k+1})| \leq \delta_S + 2\rho(t_q^{k+1} - t^k) - 2\rho r_{max}$ and hence

$$\begin{array}{lll} |C_p(t^k) - C_q(t^k)| & & \\ (3.1) & \leq & \delta_S + 2\rho(t_q^{k+1}\text{-}t^k)\text{-}2\rho r_{max}+2\rho(t^{k+1}\text{-}t_q^{k+1}) \\ (3.2) & \leq & \delta_S \end{array}$$

## 8.8 Theorem T1

Theorem *T1: When (C1)-(C3) and (A1)-(A6) are valid and processes p,q are correct at t,*

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

Proof: There exists a $k$ so that $t^k \leq t < t^{k+1}$. Without loss of generality, we can assume $t_p^{k+1} \leq t_q^{k+1}$. We consider first the case $t^k \leq t < t_p^{k+1}$. With lemma (L2) 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 $t_p^{k+1} \leq t < t_q^{k+1}$. $C_p^{k+1}(t_p^{k+1}) \in I^k(t_p^{k+1})$ 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 (L2) one can show that $|C_p(t) - C_q(t)| \leq \delta$. □

| Symbol | Meaning |
|---|---|
| $A_p^k$ | $p$'s adjustment value in round k. |
| $\beta$ | maximum difference between $t_p^k$ and $t_q^k$. |
| $C_p$ | virtual clock of process p. |
| $C_p(T, q)$ | $q$'s estimate of $p's$ virtual clock at local time $T$. |
| $C_p^k$ | round clock of process $p$. |
| $\mathcal{CT}$ | set of clock time values. |
| $R$ | maximum execution time of a remote clock reading method. |
| $\delta$ | maximum internal deviation between virtual clocks. |
| $\delta_S$ | max. internal deviation between virtual clocks at the start of a round. |
| $\Lambda$ | maximum clock reading error. |
| $G$ | maximum discontinuity of virtual clocks. |
| $H_p$ | hardware clock of process $p$. |
| $N$ | number of time servers, i.e. $N = \lvert\mathcal{P}\rvert$. |
| $\mathcal{P}$ | set of time-server processes. |
| $p, q, r, s$ | processes, i.e. $p, q, r, s \in \mathcal{P}$. |
| $r_{min}$ | minimum duration of a round. |
| $r_{max}$ | maximum duration of a round. |
| $rank(p)$ | rank of process $p$ ; $rank(p) \in \{0, .., N-1\}$. |
| $\rho$ | max. drift rate of hardware clocks. |
| $\rho_v$ | max. drift rate of virtual clocks. |
| $t^k$ | start of the $k$-th round for all correct processes. |
| $t_p^k$ | start of the $k$-th of process $p$. |
| $T_p^i$ | local time at the start of the $k$-th round of process $p$. |