Mechanical Verification of a Generalized Protocol for Byzantine Fault Tolerant Clock Synchronization

Reprint from J. Vytopil, editor: *Formal Techniques in Real-Time and Fault-Tolerant Systems*, Nijmegen, The Netherlands, January 1992, pages 217–236; Volume 571 of Springer Verlag *Lecture Notes in Computer Science*.

Natarajan Shankar*
Computer Science Laboratory
SRI International
Menlo Park, CA 94025 USA

Abstract

Schneider [Sch87] generalizes a number of protocols for Byzantine fault-tolerant clock synchronization and presents a uniform proof for their correctness. We present a mechanical verification of Schneider's protocol leading to several significant clarifications and revisions. The verification was carried out with the EHDM system [RvHO91] developed at the SRI Computer Science Laboratory. The mechanically checked proofs include the verification that the *egocentric mean* function used in Lamport and Melliar-Smith's Interactive Convergence Algorithm [LMS85] satisfies the requirements of Schneider's protocol. Our mechanical verification raises a number of issues regarding the verification of fault-tolerant, distributed, real-time protocols that are germane to the design of a special-purpose logic for such problems.

*This work was supported by NASA Contract NAS1-18226. John Rushby, Friedrich von Henke, Fred Schneider, and Rick Butler provided considerable guidance and encouragement. I also thank Paul Miner (NASA Langley Research Center) and the referees for their comments and clarifications.

1 Introduction

Synchronizing clocks in the presence of faults is a classic problem in distributed computing. Even the most accurate clocks do drift at significant rates, both with respect to a time standard and relative to each other. In order for independent processors to exhibit cooperative behavior, it is often required that their local clocks be synchronized. Such synchrony is the basis for distributed algorithms that use timeouts, time stamps, and rounds of message passing. Synchronization is also assumed when the same computation is executed on multiple, independent processors in order to mask processor failures.

Synchronizing clocks in the presence of faults is a difficult problem. Maintaining synchrony by periodically broadcasting a global clock has the drawback of creating a single point of failure. The basic way to achieve fault-tolerant synchronization is for each processor to periodically execute a protocol that involves exchanging clock values with the other processors, computing a consensus reading from these values, and appropriately adjusting its local clock to reflect the consensus. The difficulty is that processors can fail in arbitrary, unpredictable ways and can upset the consensus by communicating one clock value to one processor and a different clock value to another. An algorithm that can cope with such failures is said to be *Byzantine* fault-tolerant [LSP82]. There are a number of known algorithms for Byzantine fault-tolerant clock synchronization. These algorithms themselves are fairly simple to describe, but the reasoning required to establish their correctness is extremely delicate. The difficulty stems from having to simultaneously deal with relative and absolute clock drifts, processor failures, reading errors, and the complicated arithmetic that is involved. Correctly modelling the behavior of clocks can itself be a difficult problem under these conditions.

Schneider [Sch87] presents a clock synchronization scheme (abbreviated here as SCS) that captures the mathematics behind a number of individual synchronization algorithms. His scheme alleviates some of the complexity of reasoning about these protocols. Schneider regards each processor as maintaining a local clock by periodically adjusting its value to one computed by a *convergence function* applied to the readings of all of the clocks. Schneider places certain natural conditions on the behavior of suitable convergence functions and shows that these conditions are sufficient for demonstrating that at any time, the readings of two nonfaulty clocks are always within a fixed bound of each other. The convergence functions used by individual protocols can then be shown to meet Schneider's conditions.

The generality of Schneider's formulation made it an appropriate candidate for mechanical verification. Our verification employed the EHDM specification/verification environment developed at the Computer Science Laboratory of SRI International. The verification provides a rigorous formalization of the behavior of clocks in the presence of Byzantine faults, and careful and tight derivations of the conditions needed to achieve synchronization. The use of EHDM led to the clarification of a number of details from Schneider's original presentation. For instance, Schneider employs a monotonicity condition on convergence functions that was found to be inessential for the proof. The monotonicity condition actually fails for several protocols (see Section 5). The mechanized proof clears up some minor inaccuracies in Schneider's derivation of several of the inequality constraints on the various quantities. The powerful decision procedures for linear equalities and inequalities provided by EHDM were extremely useful for this proof.

There are some other related efforts aimed at mechanically verifying distributed protocols. Rushby and von Henke [RvH91] have used EHDM to check the proofs of Lamport and Melliar-Smith's interactive convergence clock synchronization algorithm (ICA) [LMS85]. This verification followed the original presentation of ICA in modelling local clocks as mapping clock time to real time. Our verification formalizes clocks as mapping real time to clock time. We compare these approaches in Section 6. Rushby [Rus91] uses EHDM to model and verify fault masking and recovery in a synchronous N-plex system for fault-tolerance. Bevier and Young [BY90] have used the Boyer-Moore theorem prover to verify the Oral Messages algorithm for Byzantine agreement [LSP82] assuming that the processors are already synchronized.

Schneider's results appear in a technical report and have not had the benefit of widespread scrutiny so it is not surprising that errors were discovered during verification. On the other hand, the interactive convergence algorithm appears in a widely read journal paper [LMS85] and the verification attempt by Rushby and von Henke [RvH91] discovered flaws in the proof that had, till then, safely survived the social process.

The above machine-assisted proofs of distributed protocols were formalized in general-purpose logics. EHDM, for example, uses a simply typed higher-order logic. Despite the mechanical assistance and the relative abstractness of the SCS protocol, our verification still required a significant amount of effort. It is an interesting challenge to devise a logic that is more specialized to the task of describing fault-tolerant distributed protocols and proving their correctness.

In this paper, we describe one outcome of our mechanized verification, namely, a precise description of Schneider's clock synchronization scheme. We also examine the issues relating to the formal description and verification of such protocols. In Section 2 we discuss the problem of Byzantine fault-tolerant clock synchronization. Section 3 is a careful outline of the SCS protocol as refined by the mechanized verification. Section 4 is a brief sketch of the proof that was mechanically checked. Section 5 illustrates how the egocentric mean function of the ICA protocol satisfies Schneider's conditions. Some observations on the proof are presented in Section 6. The Appendices present the informal proof and some highlights of the mechanized verification. An expanded description of the proof is available as a technical report [Sha91].

2 Byzantine Fault-tolerant Clock Synchronization

In any implementation of synchronized clocks, each processor has a *physical* clock that is typically a crystal clock. Such a physical clock drifts away from the fixed standard time ("real time") at a rate that can be bounded. By periodically applying an adjustment to the reading of the physical clock, each processor also maintains a *logical,* or *virtual,* clock. The adjustment is computed by a protocol involving the exchange of clock readings by the various processors. The primary requirement that any algorithm for clock synchronization must satisfy is that at any instant, the absolute difference, or the *skew*, between two virtual clock readings should be within some fixed, acceptable bound.

Processor failure adds a significant dimension of complexity to the problem of clock synchronization. As an illustration of the difficulty of synchronizing clocks in the presence of Byzantine failures, consider the case of three clocks a, b, and c, where only c is faulty, and nonfaulty clocks can gain or lose up to one minute during an hour. This means that two nonfaulty clocks could drift apart by two minutes over an hour since one clock can gain a minute while the other loses a minute. Suppose that the goal is to keep the nonfaulty clocks synchronized to within three minutes, where clock a gains a minute each hour and b loses a minute each hour. The clocks start synchronized at 12 noon. At 1pm, clock a reads 1:01pm, clock b reads 12:59pm, and clock c has failed. The clocks exchange their readings, and c maliciously communicates its reading as 1:03pm to a and as 12:57pm to b. At this point, a natural way for a clock to resynchronize would be to reset itself to the average of the acceptable clock readings, namely those readings that are within three minutes of its own reading. Then a resets itself to 1:01pm and b resets itself to 12:59pm, so that they remain two minutes apart. Continuing thus, at 2pm, clock a reads 2:02pm whereas clock b reads 1:58pm. The clocks a and b are now four minutes apart, thereby exceeding the acceptable bound on the skew between nonfaulty clocks.

The above scenario illustrates one of the early clock synchronization protocols capable of tolerating Byzantine processor failures: the Interactive Convergence Algorithm (ICA) of Lamport and Melliar-Smith [LMS85]. ICA tolerates up to F failures for N processors where $3F < N$, so that in the above case, at least four clocks are needed to tolerate a single failure. In ICA, a processor p resynchronizes for the i'th time when its clock reads iR. Processor p then reads the difference between the other clock readings and its own clock reading. By ignoring clock readings that differ from its own by more than a certain fixed value, a processor p computes the *egocentric mean* of the remaining clock readings as the required resynchronized clock reading. A number of functions can be used in place of egocentric mean function in order to compute the correction. For example, the fault-tolerant mean function takes the average of the clock readings that remain after the top F and the bottom F readings are discarded [LL84].

The following section describes the formalization of the SCS protocol arising from our verification.

3 Schneider's Schema for Clock Synchronization

The SCS protocol is described below in careful detail so that it can be compared with Schneider's original presentation [Sch87]. Section 3.1 describes how the logical clock is computed from the physical clock using the convergence function. Section 3.2 describes Schneider's conditions on the behavior of clocks and on suitable convergence functions. Note that we only deal with the case when the clocks are resynchronized with an instantaneous adjustment, whereas Schneider also deals with the situation when the adjustment is applied in a continuous manner. Since certain "clock ticks" might be lost or repeated in an instantaneous resynchronization, no critical events can be scheduled for these clock ticks. Another point to note is that both real time and clock readings range over real numbers in the formalization below, but the machine verification has also been carried out with natural number values for the time parameters.

3.1 Defining Clocks

The physical and logical clocks are presented as functions from real time (as given by some external standard) to clock readings. This real time thus forms the frame of reference and is often referred to simply as "time." The variable t ranges over this real time. Values ranging over real time are written in lower case and clock times are in upper case. Synchronization takes place in *rounds*. The time at which processor p adjusts its clock following the i'th round of synchronization is represented by t_p^i. The starting time t_p^0 is taken to be zero.

$PC_p(t)$ is the reading of p's physical clock at real time t, and $VC_p(t)$ is p's virtual or logical clock reading. The virtual clock reading at time t_p^i is computed by applying an adjustment adj_p^i to the physical clock reading $PC_p(t_p^i)$. In its i'th *interval* of operation, *i.e.*, when $t_p^i \leq t < t_p^{i+1}$, the virtual clock reading, $VC_p(t)$ is given by $PC_p(t) + adj_p^i$. The virtual clock in the interval between t_p^i and t_p^{i+1} is modelled by an abstraction called the interval clock whose value at time t is $IC_p^i(t)$.

At round 0, the adjustment adj_p^0 is taken to be 0 so that for $t < t_p^1$, the reading $VC_p(t)$ is just $PC_p(t)$. For $i > 0$, we let Θ_p^i be an array of clock readings so that $\Theta_p^i(q)$ is p's reading of $IC_q^{i-1}(t_p^i)$, *i.e.*, q's $(i-1)$'th interval clock reading at time t_p^i, since p is really estimating q's clock reading without taking into account the adjustment adj_q^i. The corrected value of the clock at time t_p^i, namely $VC_p(t_p^i)$, is computed by a *convergence function*, $cfn(p, \Theta_p^i)$.[1]

[1]In the EHDM formalization, the array of observed clock readings Θ_p^i, is actually represented as a function from clocks to readings. Since Θ_p^i is a function, *cfn* is a higher-order function.

The above description leads to following definitions where i ranges over the natural numbers and $t > 0$.

$$
\begin{aligned}
adj_p^{i+1} &= cfn(p, \Theta_p^{i+1}) - PC_p(t_p^{i+1}) && (3.1)\\
adj_p^0 &= 0 && (3.2)\\
IC_p^i(t) &= PC_p(t) + adj_p^i && (3.3)\\
VC_p(t) &= IC_p^i(t), \text{ for } t_p^i \le t < t_p^{i+1} && (3.4)
\end{aligned}
$$

It is easy to derive the following from Definitions (3.1), (3.3), and (3.4).

$$
\begin{aligned}
VC_p(t_p^{i+1}) &= IC_p^{i+1}(t_p^{i+1}) = cfn(p, \Theta_p^{i+1}) && (3.5)\\
IC_p^{i+1}(t) &= cfn(p, \Theta_p^{i+1}) + PC_p(t) - PC_p(t_p^{i+1}) && (3.6)
\end{aligned}
$$

In the next section, we enumerate the constraints on these quantities when p is a *nonfaulty* processor. The constraints on the behavior of the convergence function are particularly significant. The main result obtained from these constraints and the above definitions is a bound δ on the skew between the logical clocks of two correct processors p and q.

Theorem 3.1 (bounded skew) *For any two clocks p and q that are nonfaulty at time t,*

$$|VC_p(t) - VC_q(t)| \le \delta \tag{3.7}$$

The proof of Theorem 3.1 is outlined in Section 4.

3.2 Clock conditions

In formalizing the laws constraining the behavior of individual clocks, we must ensure that no assumptions are made regarding the faulty clocks since we are dealing with Byzantine failures. These laws which are conditions on the behavior of clocks are enumerated below. Individual protocols and clock implementations are expected to satisfy these conditions. Note that N is the total number of processors, and F is the maximum number of faulty clocks that the algorithm is expected to tolerate. To start, the following condition asserts that the nonfaulty clocks are synchronized to within the quantity δ_S at time 0.

Condition 1 (initial skew) *For nonfaulty processors p and q*

$$|PC_p(0) - PC_q(0)| \le \delta_S \tag{3.8}$$

The nonfaulty physical clocks must keep good enough time so that they do not drift away from real time by a rate greater than ρ.

Condition 2 (bounded drift) *If clock p is nonfaulty at time s, $s \ge t$, then*

$$(1-\rho)(s-t) \le PC_p(s) - PC_p(t) \le (1+\rho)(s-t) \tag{3.9}$$

A useful corollary to *bounded drift* is that two physical clocks p and q that are not faulty[2] at time s, for $s \geq t$, can drift further apart over the interval $s - t$ by $2\rho(s-t)$, since both p and q can drift by $\rho(s-t)$ with respect to real time, but in opposite directions.

$$|PC_p(s) - PC_q(s)| \leq |PC_p(t) - PC_q(t)| + 2\rho(s-t) \tag{3.10}$$

Each protocol has some mechanism for triggering the resynchronization of the clocks. Schneider postulates the existence of a global synchronization signal, t_G^i, which occurs at a period bounded from above and below. Our description dispenses with the notion of a global synchronization signal and bounds the period between the local synchronization signals and the range within which these signals occur.

Condition 3 (bounded interval) *For nonfaulty clock p*

$$0 < r_{min} \leq t_p^{i+1} - t_p^i \leq r_{max} \tag{3.11}$$

Condition 4 (bounded delay) *For nonfaulty clocks p and q*[3]

$$|t_q^i - t_p^i| \leq \beta \tag{3.12}$$

Condition 5 (initial synchronization) *For nonfaulty clock p*

$$t_p^0 = 0 \tag{3.13}$$

The following condition ensures that there is no overlap between synchronization periods, *i.e.*, by the time any nonfaulty processor is ready to synchronize for the $(i+1)$'th time, all nonfaulty processors have already synchronized for the i'th time.

Condition 6 (nonoverlap)

$$\beta \leq r_{min} \tag{3.14}$$

An important corollary of the *bounded interval* and *bounded delay* conditions is that for any two nonfaulty clocks p and q,

$$0 \leq t_p^{i+1} - t_q^i \leq r_{max} + \beta. \tag{3.15}$$

For a nonfaulty clock p, the value $\Theta_p^{i+1}(q)$ represents p's observation of q's i'th clock reading at time t_p^{i+1}, *i.e.*, it is p's estimate of $IC_q^i(t_p^{i+1})$. The error in this reading is assumed to be bounded by Λ.

---

[2] In the mechanized verification, great pains are taken to indicate the times at which the clocks are required to be nonfaulty. The informal discussion here makes the simplifying assumption that clocks are either faulty or nonfaulty, and often disregards the time at which clocks are asserted as being nonfaulty.

[3] Rushby [private communication] observes that this condition is a somewhat stringent one since for most protocols, it entails the assumption that the clocks are already synchronized. This is an important observation since it implies that the demonstration that a particular protocol meets this condition will have to be carried out by induction over the number of synchronization rounds.

Condition 7 (reading error) *For nonfaulty clocks p and q,*

$$|IC_q^i(t_p^{i+1}) - \Theta_p^{i+1}(q)| \leq \Lambda \tag{3.16}$$

Condition 8 (bounded faults) *At any time t, at most F processors are faulty. If p is nonfaulty at time t, then it is nonfaulty at any time s prior to t.*

The conditions below are mathematical constraints placed on the convergence function, e.g., clocks, drifts, and failures, do not play any role in the statements. The isolation of these constraints makes it possible to demonstrate that the egocentric mean function of ICA satisfies the conditions below independent of the context of its use. The condition of *translation invariance* indicates that adding x to the value of the convergence function should be the same as adding x to each clock reading instead.

Condition 9 (translation invariance) *For any function θ mapping clocks to clock values,*

$$cfn(p, (\lambda n\colon \theta(n) + x)) = cfn(p, \theta) + x \tag{3.17}$$

The next condition of *precision enhancement* facilitates a comparison between values of the convergence function based on the range of values of some subset C of the clock readings. C is to be intuitively interpreted as the subset of nonfaulty processors. *Precision enhancement* captures the convergence behavior of the convergence function by asserting that the "closer" two arrays of clock readings γ and θ are to each other, the closer are the results of the convergence function applied to γ and θ, respectively. In the statement of *precision enhancement*, the above intuitive interpretation of C as the subset of nonfaulty clocks is permissible by the *bounded faults* condition. The "closeness" of γ and θ is formalized by asserting that all the readings in γ and θ, respectively, of clocks in C lie in an interval of width y, and that the corresponding readings of γ and θ of any clock in C are no more than x apart. The function $\pi(x, y)$ captures the closeness of the values that result from applying the convergence function to γ and θ.[4]

Condition 10 (precision enhancement) *Given any subset C of the N clocks with $|C| \geq N - F$, and clocks p and q in C, then for any readings γ and θ satisfying the conditions*

1. *for any l in C, $|\gamma(l) - \theta(l)| \leq x$*
2. *for any l, m in C, $|\gamma(l) - \gamma(m)| \leq y$*
3. *for any l, m in C, $|\theta(l) - \theta(m)| \leq y$*

[4]Note that the order of arguments to π are reversed from their order in Schneider's description [Sch87].

there is a bound $\pi(x, y)$, such that

$$|cfn(p, \gamma) - cfn(q, \theta)| \leq \pi(x, y) \tag{3.18}$$

The final condition of *accuracy preservation* bounds the distance between the value of $cfn(p, \theta)$ and the nonfaulty readings in θ.[5]

Condition 11 (accuracy preservation) *Given any subset C of the N clocks with $|C| \geq N - F$, and clock readings θ such that for any l and m in C, the bound $|\theta(l) - \theta(m)| \leq x$ holds, there is a bound $\alpha(x)$ such that for any q in C*

$$|cfn(p, \theta) - \theta(q)| \leq \alpha(x) \tag{3.19}$$

Schneider also proposes a condition called *monotonicity* that is actually not satisfied by several clock synchronization protocols though it is used heavily in Schneider's proofs. Fortunately, this condition turns out to be unnecessary in the derivation. The monotonicity condition asserts that if for each processor l, $\theta(l) \geq \gamma(l)$, then $cfn(p, \theta) \geq cfn(p, \gamma)$. The failure of the monotonicity condition for ICA is demonstrated in Section 5.

4 The Correctness Proof

The formal arguments are extremely delicate to carry out carefully and correctly due to the additional consideration of processor failure. The phenomenon of processor failure is usually dealt with casually in informal presentations, but adds significantly to the complexity of the formalization as well as the proof. A brief sketch of the proof is given below, and a few further details are provided in Appendix A.

To establish the main result, Theorem 3.1, we must show that the skew, or absolute difference, between the readings of any two nonfaulty clocks p and q at time t, given by $|VC_p(t) - VC_q(t)|$, is bounded by a quantity δ. The first step (Theorem 4.1) is to bound the skew at the instant when both p and q have resynchronized their clocks for the i'th time. This time, $t^i_{p,q}$, is $max(t^i_p, t^i_q)$. The skew at this instant can be bounded by a quantity δ_S that can be computed using the conditions of *translation invariance*, *precision enhancement*, and *reading error*, and the proof is by induction on i.

Theorem 4.1 *There is a bound δ_S such that for synchronization round i and any two nonfaulty processors p and q*

$$|IC^i_p(t^i_{p,q}) - IC^i_q(t^i_{p,q})| \leq \delta_S \tag{4.20}$$

[5] Footnote 7 in Schneider [Sch87] explains the choice of the terms *precision enhancement* and *accuracy preservation*. 'Precision' is defined as the closeness with which a measurement can be reproduced, whereas 'accuracy' is the proximity of the measurement to the actual value being measured. The virtual clocks represent various measurements of real time. *Precision enhancement* characterizes the closeness of these measurements to each other. *Accuracy preservation* can be seen as bounding the drift rate of the virtual clock with respect to real time.

We can now compute the quantity δ that bounds the skew in the interval $t^i_{p,q} \leq t < t^{i+1}_{p,q}$. The proof of Theorem 4.2 has two cases according to whether $t < min(t^{i+1}_p, t^{i+1}_q)$ or $t \geq min(t^{i+1}_p, t^{i+1}_q)$. The first case follows easily from *bounded drift* and *bounded interval*, and the second case requires *accuracy preservation* as well.

Theorem 4.2 *For any two nonfaulty clocks p, q, and $t^i_{p,q} \leq t < t^{i+1}_{p,q}$,*

$$|VC_p(t) - VC_q(t)| \leq \delta. \tag{4.21}$$

Note that for any $t \geq 0$, there is an i such that $t < t^i_{p,q}$. The main theorem then follows from the fact that Theorem 4.2 yields a skew bound δ for any t such that $0 \leq t < t^i_{p,q}$. We note of the various constraints on δ and δ_S that arise from the proofs in Appendix A:

1. $\pi(2\Lambda + 2\beta\rho, \delta_S + 2\rho(r_{max} + \beta) + 2\Lambda) \leq \delta_S$
2. $\delta_S + 2\rho r_{max} \leq \delta$
3. $\alpha(\delta_S + 2\rho(r_{max} + \beta) + 2\Lambda) + \Lambda + 2\rho\beta \leq \delta$.

5 ICA as an instance of Schneider's scheme

To gain some intuition into the SCS protocol, we demonstrate that the egocentric mean function from the Interactive Convergence Algorithm of Lamport and Melliar-Smith [LMS85] satisfies Schneider's conditions of translation invariance, precision enhancement, and accuracy preservation.

With the interactive convergence algorithm, the convergence function cfn_I takes the *egocentric mean* of p's estimate of the readings of the N clocks numbered from 0 to $N-1$, *i.e.*, any readings that are more than Δ away from p's own reading are replaced by p's own reading. This yields the definition

$$cfn_I(p, \theta) = \frac{\sum_{l=0}^{N-1} fix_p(\theta(l))}{N} \tag{5.22}$$

where

$$fix_p(x) = \begin{cases} x & \text{if } |x - \theta(p)| \leq \Delta \\ \theta(p) & \text{otherwise.} \end{cases}$$

Translation invariance follows from the observation that

$$fix_p((\lambda l\colon \theta(l) + t)(q)) = fix_p(\theta(q)) + t \tag{5.23}$$

and

$$\frac{\sum_{l=0}^{N-1}(\theta(l) + t)}{N} = \frac{\sum_{l=0}^{N-1}(\theta(l))}{N} + t \tag{5.24}$$

To demonstrate *precision enhancement*, we start with a set of processors C of cardinality $|C|$ greater than $N - F$. Let f be $N - |C|$. The hypotheses for precision enhancement are that for any l and m in C, $|\gamma(l) - \theta(l)|$ is bounded by x, and $|\gamma(l) - \gamma(m)|$ and $|\theta(l) - \theta(m)|$ are bounded by y. When $y \leq \Delta$, it can be shown that

$$|cfn_I(p, \gamma) - cfn_I(q, \theta)| \leq \frac{(N-f)x}{N} + \frac{2f\Delta + fx + fy}{N}. \tag{5.25}$$

The right-hand side of (5.25) is the required $\pi(x, y)$ in this case.

In the typical situation when the egocentric mean is computed, the quantity x representing the reading error is negligible, and y representing the clock skew is bounded by Δ. Since the skew following synchronization should be smaller than Δ, we can see that in Equation (5.25), the number of failed processors f should be below $N/3$. The derivation of $\pi(x, y)$ for the case when $y > \Delta$, is carried out in the mechanized proof.

To show that cfn_I satisfies accuracy preservation, it is sufficient to observe that if all the clocks in C (the C-clocks) are within x of each other, then the C-clocks can cause the egocentric mean to be at most $(N - f)x/N$ away from any nonfaulty clock. The clocks outside C can cause the egocentric mean to be up to $f \times (x + \Delta)/N$ away from a good clock. The total thus yields

$$\alpha(x) = x + \frac{f\Delta}{N}.$$

The final step is to demonstrate the failure of the monotonicity condition for ICA. The monotonicity condition mentioned at the end of Section 3.2 asserts that if for each processor l, $\theta(l) \geq \gamma(l)$, then $cfn(p, \theta) \geq cfn(p, \gamma)$. Let $\theta(p) = \gamma(p)$. Observe now that if there is some l such that $\theta(l) + \Delta < \theta(p)$, but with $\gamma(p) > \gamma(l) \geq \gamma(p) - \Delta$, then $fix_p(\theta(l)) > fix_p(\gamma(l))$ holds. So, it is possible to have $fix_p(\theta(l)) > fix_p(\gamma(l))$, even though we have $\theta(l) < \gamma(l)$.

6 Discussion

The formal verification of the proof of the SCS protocol required considerable effort. Our own experience suggests that the difficulty would have been comparable using other verification systems. The following discussion highlights some of the complexity of reasoning about fault-tolerant clock synchronization protocols, and examines some issues that are relevant to managing this complexity.

Formalization itself is one source of complexity. A number of issues are simply swept under the rug in informal descriptions. For example, informal presentations of fault-tolerant protocols treat failure in a somewhat casual way. Clocks that fail are regarded as always having been faulty. Note that our informal presentation above makes essentially the same simplifying assumption. The mechanized proof is a great deal more precise. Clocks are allowed to fail at any point in time. A great deal of care is taken to not build in any assumptions regarding the behavior of failed clocks save that their clock readings are unspecified functions of the standard global time. Under the assumption that clocks can fail at any time, the precise forms of the axioms are considerably more difficult to formulate. For example, the axioms `rts0` and `rts1` in Figure 2 are not at all the obvious formulations of the condition of *bounded interval* since p is constrained to be correct at time t. The obvious form of these axioms would assert p to be correct at t_p^{i+1} and would thus say little about time points other than t_p^i and t_p^{i+1}. We initially employed the "obvious" forms of these axioms and quickly found them inadequate during the course of the mechanized proof. These two axioms turned out to have some other inadequate variations as well. Other axioms formalizing clock behavior also posed similar challenges.

Clock drifts are another obvious and inherent source of difficulty in the proof. Virtually the entire proof consists of determining real time bounds on certain intervals and using these to derive bounds on the clock drifts over these intervals. The algebra involved in these calculations is very complex and has to be done with some care. It is conceivable that these calculations could be further automated so that these bounds can be computed rather than supplied by the user.

A key difference between our proof of the SCS protocol and the Rushby/von Henke proof of the ICA protocol is that they follow Lamport and Melliar-Smith in formalizing local clocks as functions from clock time to real time whereas local clocks are formalized here as functions from real time to clock time. As a consequence, our main theorem asserts that the skew between any two correct clocks at the same time instant is bounded, whereas their main theorem asserts a bound in the real time interval separating one correct clock's reading of time T from another correct clock's reading of time T. We were unable to conclusively establish either approach to be clearly superior. It seems likely that the best approach would be to simultaneously employ both notions of a local clock. The internal behavior of the each processor is governed by the local clock, and this can be mapped to its real time behavior. The synchronization between various processors is best done in terms of a real time frame of reference and subsequently mapped back to timing requirements on the behavior of the individual processors.

The proofs here use a dense notion of time; the time variables range over ordered fields. The proof has been carried out with very minor changes with respect to a discrete notion of time. One significant point ignored by the formalization here is that the *bounded drift* axiom can only be guaranteed to hold over a large enough interval of measurement. It would be easy to incorporate such a notion into the formalization and redo the proof. (The ability to easily redo proofs with minor perturbations to the assumptions is one of the significant benefits of mechanized formal verification.)

Though there are a number of variants of temporal logic [Pnu77] that capture real-time notions [AH89, EMSS89, Koy89], none of these seem adequately sophisticated for dealing with faults and local clocks with any special felicity. It appears that what is required of such a logic is the ability to easily translate from local time to global time assertions, to treat faultiness as a time-varying property, and to reason about the cardinalities of processors satisfying certain constraints. We are currently investigating the design of such a logic. It would also be interesting to examine whether the model-checking approaches [CG87] to the verification of distributed protocols can be fruitfully applied here.

7 Conclusions

Rigorously proving the correctness of distributed protocols is an extremely difficult task, with or without mechanical assistance. Fault-tolerant clock synchronization is an excellent example of a problem where the algorithms, though often simple, are not at all easily verified. In such cases, it is extremely important to have certain organizing principles which capture the common features of the various protocols with convincing generality. Schneider's schema for Byzantine clock synchronization provides such principles.

The formalization here revises Schneider's presentation in some small ways. Schneider's notion of a global signal to trigger resynchronization has been dropped because such a notion is difficult to instantiate for many protocols. Though the quantities r_{max} and r_{min} have a different meaning from Schneider's, these differences ought not to matter in any of the bounds derived. The derivation we present is extremely tight, given the structure of the proof. Schneider's monotonicity condition is avoided the proofs here. This condition is used heavily by Schneider in his arguments, but it actually turns out to not hold for many protocols. The statement of accuracy preservation here is also slightly different here from that of Schneider.

The initial proof using EHDM and including the pencil-and-paper development took about a month. The conditions on clocks and convergence functions were fairly difficult to formalize due to the careful treatment of failure. Many of these statements were refined during the course of the proof. The proof itself has been considerably revised and improved since the first effort. Verifying that the egocentric mean function of ICA satisfied the conditions of *translation invariance, accuracy preservation*, and *precision enhancement*, took about two weeks. The complete proof involves about 182 theorems or lemmas.

The most useful feature of EHDM for this proof were the decision procedures for linear integer and rational inequalities and equalities. The proof is of course replete with long chains of inequality reasoning, and the decision procedures handled those steps in a fairly mechanical manner. The higher-order features of the language were also used to formalize the conditions of translation invariance, precision enhancement, and accuracy preservation, but such features were not essential to this proof.

Fault-tolerant distributed protocols are sufficiently delicate to warrant careful, formal, mechanized analysis. Such an analysis is possible with the existing technology for specification and verification. Schneider's presentation provides a valuable mathematical framework for the verification of synchronization protocols. The machine-checked proof of Schneider's protocol led to a precise formulation of the protocol and a closely reasoned proof. It is inconceivable that the same degree of logical rigor and preciseness could be achieved without computational assistance.

References

[AH89] R. Alur and T. A. Henzinger. A really temporal logic. In *30th IEEE Symposium on Foundations of Computer Science*, pages 164–169, 1989.

[BY90] W. R. Bevier and W. D. Young. Machine checked proofs of the design and implementation of a fault-tolerant circuit. NASA Contractor Report 182099, Computational Logic, Inc., 1990.

[CG87] E. M. Clarke and O. Grumberg. Research on automatic verication of finite state concurrent systems. In *Annual Review of Computer Science*, pages 269–290. Annual Reviews, Inc., 1987.

[EMSS89] E. A. Emerson, A. K. Mok, A. P. Sistla, and J. Srinivasan. Quantitative temporal reasoning. In *Computer-Aided Verification*, 1989.

[Koy89] R. Koymans. *Specifiying message passing and time-critical systems with temporal logic*. PhD thesis, Eindhoven Univ. of Technology, 1989.

[LL84] J. Lundelius and N. A. Lynch. A new fault-tolerant algorithm for clock synchronization. In *Proc. of the Third ACM Symp. on Principles of Distributed Computing*, pages 75–88, 1984.

[LMS85] L. Lamport and P. M. Melliar-Smith. Synchronizing clocks in the presence of faults. *Journal of the ACM*, 32(1):52–78, January 1985.

[LSP82] Leslie Lamport, Robert Shostak, and Marshall Pease. The Byzantine generals problem. *ACM Transactions on Programming Languages and Systems*, 4(3):382–401, July 1982.

[Pnu77] A. Pnueli. The temporal logic of programs. In *Proc. 18th Ann. IEEE Symp. on Foundations of Computer Science*, pages 46–57, 1977.

[Rus91] John Rushby. Formal specification and verification of a fault-masking and transient-recovery model for digital flight-control systems. Technical Report SRI-CSL-91-3, Computer Science Laboratory, SRI International, Menlo Park, CA, January 1991. Also available as NASA Contractor Report 4384, July 1991.

[RvH91] John Rushby and Friedrich von Henke. Formal verification of the Interactive Convergence clock synchronization algorithm using EHDM. Technical Report SRI-CSL-89-3R, Computer Science Laboratory, SRI International, Menlo Park, CA, February 1989 (Revised August 1991). Original version also available as NASA Contractor Report 4239, June 1989.

[RvHO91] John Rushby, Friedrich von Henke, and Sam Owre. An introduction to formal specification and verification using EHDM. Technical Report SRI-CSL-91-2, Computer Science Laboratory, SRI International, Menlo Park, CA, February 1991.

[Sch87] Fred B. Schneider. Understanding protocols for Byzantine clock synchronization. Technical Report 87-859, Department of Computer Science, Cornell University, Ithaca, NY, August 1987.

[Sha91] Natarajan Shankar. Mechanical verification of a schematic Byzantine fault-tolerant clock synchronization algorithm. Technical Report SRI-CSL-91-4, Computer Science Laboratory, SRI International, Menlo Park, CA, January 1991. Also available as NASA Contractor Report 4386, July 1991.

A Details of the Correctness Proof

The details of the proof of *bounded skew* are completed below by providing the proofs of Theorems 4.1 and 4.2.

Proof of Theorem 4.1. The proof is by induction on the round number i.

Base case: When $i = 0$, by (3.13) we have $t_p^0 = t_q^0 = 0$. The skew bound of δ_S follows from Definitions (3.3) and (3.1), and the *initial skew* condition.

Induction case: The induction hypothesis asserts that for every pair of nonfaulty processors, l and m

$$|IC_l^i(t_{l,m}^i) - IC_m^i(t_{l,m}^i)| \leq \delta_S \tag{A.26}$$

The goal is to establish for any pair of nonfaulty processors p and q, that

$$|IC_p^{i+1}(t_{p,q}^{i+1}) - IC_q^{i+1}(t_{p,q}^{i+1})| \leq \delta_S \tag{A.27}$$

Without loss of generality, assume that t_q^{i+1} precedes t_p^{i+1} so that $t_{p,q}^{i+1} = t_p^{i+1}$. Then Equation (3.6) and *translation invariance* yield

$$\begin{aligned} IC_q^{i+1}(t_p^{i+1}) &= cfn(q, \Theta_q^{i+1}) + PC_q(t_p^{i+1}) - PC_q(t_q^{i+1}) \\ &= cfn(q, (\lambda n\colon \Theta_q^{i+1}(n) + PC_q(t_p^{i+1}) - PC_q(t_q^{i+1}))) \end{aligned} \tag{A.28}$$

By Equation (3.5), we have

$$IC_p^{i+1}(t_p^{i+1}) = cfn(p, \Theta_p^{i+1}) \tag{A.29}$$

so that the required skew can be rewritten as

$$|cfn(q, (\lambda n\colon \Theta_q^{i+1}(n) + PC_q(t_p^{i+1}) - PC_q(t_q^{i+1}))) - cfn(p, \Theta_p^{i+1})|$$

This quantity can be bounded using *precision enhancement* with $(\lambda n\colon \Theta_q^{i+1}(n) + PC_q(t_p^{i+1}) - PC_q(t_q^{i+1}))$ for γ and Θ_p^{i+1} for θ. The set C in *precision enhancement* is taken to be the subset of nonfaulty clocks as permitted by *bounded faults*. To satisfy Hypothesis 1 of *precision enhancement*, we need an x such that for any nonfaulty l,

$$|(\Theta_q^{i+1}(l) + PC_q(t_p^{i+1}) - PC_q(t_q^{i+1})) - \Theta_p^{i+1}(l)| \leq x.$$

The value $2\rho\beta + 2\Lambda$ can be substituted for x since by *bounded drift*,

$$|(PC_q(t_p^{i+1}) - PC_q(t_q^{i+1})) - (t_p^{i+1} - t_q^{i+1})| \leq \beta\rho \tag{A.30}$$

and by *reading error* and *bounded drift*, we get

$$\begin{aligned} &|(\Theta_p^{i+1}(l) - \Theta_q^{i+1}(l)) - (t_p^{i+1} - t_q^{i+1})| \\ &\leq \ 2\Lambda + |(IC_l^i(t_p^{i+1}) - IC_l^i(t_q^{i+1})) - (t_p^{i+1} - t_q^{i+1})| \end{aligned}$$

$$\leq \quad 2\Lambda + \beta\rho \tag{A.31}$$

The induction hypothesis is needed in order to satisfy Hypotheses 2 and 3 of the relevant instance of *precision enhancement*. For both hypotheses, we need a y such that for any nonfaulty processors k, l and m,

$$|\Theta_k^{i+1}(l) - \Theta_k^{i+1}(m)| \leq y \tag{A.32}$$

By *reading error*, the induction hypothesis (A.26), and Equations (3.15) and (3.10), we have

$$\begin{aligned}
&|\Theta_k^{i+1}(l) - \Theta_k^{i+1}(m)| \\
&\leq \quad 2\Lambda + |IC_l^i(t_k^{i+1}) - IC_m^i(t_k^{i+1})| \\
&\leq \quad 2\Lambda + |IC_l^i(t_{l,m}^i) - IC_m^i(t_{l,m}^i)| + 2\rho(t_k^{i+1} - t_{l,m}^i) \\
&\leq \quad 2\Lambda + \delta_S + 2\rho(r_{max} + \beta)
\end{aligned} \tag{A.33}$$

so that the required y is $\delta_S + 2\rho(r_{max} + \beta) + 2\Lambda$. So by *precision enhancement*, if

$$\pi(2\Lambda + 2\beta\rho, \delta_S + 2\rho(r_{max} + \beta) + 2\Lambda) \leq \delta_S, \tag{A.34}$$

then

$$|IC_p^{i+1}(t_p^{i+1}) - IC_q^{i+1}(t_p^{i+1})| \leq \delta_S \tag{A.35}$$

thus completing the proof of Theorem 4.1. ∎

Proof of Theorem 4.2. Assume without loss of generality that $t_q^{i+1} \leq t_p^{i+1}$. The proof has two cases according to whether $t_{p,q}^i \leq t < t_q^{i+1}$ or $t_q^{i+1} \leq t < t_p^{i+1}$.

Case 1: Assuming $t_{p,q}^i \leq t < t_q^{i+1}$. From *bounded interval* we get $t - t_{p,q}^i \leq r_{max}$. By Equation (3.4), we get $VC_p(t) = IC_p^i(t)$ and $VC_q(t) = IC_q^i(t)$. Then by Equations (3.10), (3.3), and Theorem 4.1,

$$\begin{aligned}
&|VC_p(t) - VC_q(t)| \\
&\leq \quad |VC_p(t_{p,q}^i) - VC_q(t_{p,q}^i)| + 2\rho r_{max} \\
&\leq \quad \delta_S + 2\rho r_{max}
\end{aligned} \tag{A.36}$$

The bound δ should therefore be chosen so that

$$\delta_S + 2\rho r_{max} \leq \delta. \tag{A.37}$$

Case 2: Assuming $t_q^{i+1} < t < t_{p,q}^{i+1}$. In this interval, $VC_q(t) = IC_q^{i+1}(t)$, whereas $VC_p(t) = IC_p^i(t)$. The strategy here is to bound the skew at t_q^{i+1} and then compute the additional quantity by which the clocks can drift apart in the given interval. By

Equations (3.5) and (3.4), we have

$$|VC_p(t_q^{i+1}) - VC_q(t_q^{i+1})| = |IC_p^i(t_q^{i+1}) - cfn(q, \Theta_q^{i+1})|. \tag{A.38}$$

By *reading error* and *accuracy preservation*, the quantity $|IC_p^i(t_q^{i+1}) - cfn(q, \Theta_q^{i+1})|$ can by bound by $\alpha(x) + \Lambda$, where for any pair of nonfaulty clocks l and m,

$$|\Theta_q^{i+1}(l) - \Theta_q^{i+1}(m)| \leq x. \tag{A.39}$$

As already seen in the derivation of Equation (A.32), that (A.39) holds with $\delta_S + 2\rho(r_{max} + \beta) + 2\Lambda$ for x. We then have

$$|VC_p(t_q^{i+1}) - VC_q(t_q^{i+1})| \leq \alpha(\delta_S + 2\rho(r_{max} + \beta) + 2\Lambda) + \Lambda. \tag{A.40}$$

We can now bound the skew over the interval $t_q^{i+1} \leq t < t_p^{i+1}$, by observing that $t_p^{i+1} - t_q^{i+1} \leq \beta$ by (3.12), and applying Equation (3.10) to derive the inequality,

$$|VC_p(t) - VC_q(t)| \leq \alpha(\delta_S + 2\rho(r_{max} + \beta) + 2\Lambda) + \Lambda + 2\rho\beta. \tag{A.41}$$

Therefore δ has to be chosen to satisfy

$$\alpha(\delta_S + 2\rho(r_{max} + \beta) + 2\Lambda) + \Lambda + 2\rho\beta \leq \delta. \tag{A.42}$$

Both cases of the proof of Theorem 4.2 have been completed. ∎

This concludes the informal presentation of the proof.

B The EHDM Proof Highlights

This section contains the EHDM formalization of the conditions axiomatizing the behavior of clocks described in Section 3 and the statements of the key theorems. Figure 1 contains the type declarations for some of the variables and constants used in `clockassumptions`. The `clockassumptions` module makes use of the module `arith`, which contains the basic arithmetic facts, and `countmod`, which introduces a counting function. Nonfaultiness is expressed by the predicate `correct`.

The axioms constraining the physical behavior of the clock appear in Figure 2. Since we require μ to not exceed δ_S, the axiom `init` corresponds to *initial skew*. Axiom `correct_closed` asserts that a failed processor never recovers (see *bounded faults*). Axioms `rate_1` and `rate_2` together express the *bounded drift* condition. The axioms `rts0` and `rts1` capture the *bounded interval* condition. These axioms look strange because the variable t, needed to properly capture the correctness condition, appears in them but not in *bounded interval*. Most of the obvious ways of stating these axioms are either too restrictive or wrong. The axiom `rts2` captures *bounded delay*, and `synctime_0` is just *initial synchronization*. The condition of *nonoverlap* appears as an antecedent to the concluding theorem rather than as an axiom. In the LaTeX format below, multiplication is represented by $*$ as well clockassumptions: **Module**

Using arith, countmod

Exporting all with countmod, arith

Theory

process: **Type is** nat
event: **Type is** nat
time: **Type is** number
Clocktime: **Type is** number
$l, m, n, p, q, p_1, p_2, q_1, q_2, p_3, q_3$: **Var** process
i, j, k: **Var** event
x, y, z, r, s, t: **Var** time
X, Y, Z, R, S, T: **Var** Clocktime
γ, θ: **Var** function[process $\rightarrow$ Clocktime]
$\delta, \mu, \rho, r_{min}, r_{max}, \beta, \Lambda$: number
$PC_{\star 1}(\star 2), VC_{\star 1}(\star 2)$: function[process, time $\rightarrow$ Clocktime]
$t_{\star 1}^{\star 2}$: function[process, event $\rightarrow$ time]
$\Theta_{\star 1}^{\star 2}$: function[process, event $\rightarrow$ function[process $\rightarrow$ Clocktime]]
$IC_{\star 1}^{\star 2}(\star 3)$: function[process, event, time $\rightarrow$ Clocktime]
correct: function[process, time $\rightarrow$ bool]
cfn: function[process, function[process $\rightarrow$ Clocktime] $\rightarrow$ Clocktime]
π: function[Clocktime, Clocktime $\rightarrow$ Clocktime]
α: function[Clocktime $\rightarrow$ Clocktime]

Figure 1: Declarations from module clockassumptions

as $\star$. These are synonymous, but the latter represents the uninterpreted form of multiplication whereas the former is interpreted by the linear arithmetic decision procedures of EHDM.

The definitions of the virtual clock and the interval clock in terms of the physical clock appear in Figure 3. These correspond to (3.1), (3.4), and (3.3), respectively.

The conditions on the convergence function appear in Figure 4. The axiom `Readerror` corresponds to the condition *reading error*. The axiom `correct_count` corresponds to *bounded faults*. The remaining correspondences should be self-evident.

The conclusion corresponding to Theorem 3.1 is the theorem `agreement` in Figure 5. The verified version of Theorem 4.1 is given in Figure 6, and that of Theorem 4.2 in Figure 7. The expression $t^i_{(p \Uparrow q)[i]}$ is an alternative notation for $t^i_{p,q}$ since $(p \Uparrow q)[i]$ represents p if $t^i_p \geq t^i_q$, and q otherwise.

init: **Axiom** $\text{correct}(p,0) \supset PC_p(0) \geq 0 \wedge PC_p(0) \leq \mu$

correct_closed: **Axiom** $s \geq t \wedge \text{correct}(p,s) \supset \text{correct}(p,t)$

rate_1: **Axiom** $\text{correct}(p,s) \wedge s \geq t \supset PC_p(s) - PC_p(t) \leq (s-t) \star (1+\rho)$

rate_2: **Axiom** $\text{correct}(p,s) \wedge s \geq t \supset PC_p(s) - PC_p(t) \geq (s-t) \star (1-\rho)$

rts0: **Axiom** $\text{correct}(p,t) \wedge t \leq t_p^{i+1} \supset t - t_p^i \leq r_{max}$

rts1: **Axiom** $\text{correct}(p,t) \wedge t \geq t_p^{i+1} \supset t - t_p^i \geq r_{min}$

rts_0: **Lemma** $\text{correct}(p,t_p^{i+1}) \supset t_p^{i+1} - t_p^i \leq r_{max}$

rts_1: **Lemma** $\text{correct}(p,t_p^{i+1}) \supset t_p^{i+1} - t_p^i \geq r_{min}$

rts2: **Axiom** $\text{correct}(p,t) \wedge t \geq t_q^i + \beta \wedge \text{correct}(q,t) \supset t \geq t_p^i$

rts_2: **Axiom** $\text{correct}(p,t_p^i) \wedge \text{correct}(q,t_q^i) \supset t_p^i - t_q^i \leq \beta$

synctime_0: **Axiom** $t_p^0 = 0$

Figure 2: Physical clock axioms

VClock_defn: **Axiom**
$\text{correct}(p,t) \wedge t \geq t_p^i \wedge t < t_p^{i+1} \supset VC_p(t) = IC_p^i(t)$

Adj: function[process, event $\rightarrow$ Clocktime] =
$(\,\lambda\, p, i\!: (\ \textbf{if}\ i > 0\ \textbf{then}\ \mathit{cfn}(p, \Theta_p^i) - PC_p(t_p^i)\ \textbf{else}\ 0\ \textbf{end if}))$

IClock_defn: **Axiom** $\text{correct}(p,t) \supset IC_p^i(t) = PC_p(t) + \text{Adj}(p,i)$

Figure 3: Clock definitions

Readerror: **Axiom** $\text{correct}(p, t_p^{i+1}) \wedge \text{correct}(q, t_p^{i+1})$
$\supset |\Theta_p^{i+1}(q) - IC_q^i(t_p^{i+1})| \leq \Lambda$

translation_invariance: **Axiom**
$X \geq 0 \supset \mathit{cfn}(p, (\lambda\, p_1 \rightarrow \text{Clocktime}: \gamma(p_1) + X)) = \mathit{cfn}(p, \gamma) + X$

ppred: **Var** function[process → bool]
maxfaults: process
okay_Readpred: function[function[process → Clocktime], Clocktime,
function[process → bool] → bool] =
$(\lambda\, \gamma, Y, \text{ppred}: (\forall\, l, m: \text{ppred}(l) \wedge \text{ppred}(m) \supset |\gamma(l) - \gamma(m)| \leq Y))$
okay_pairs: function[function[process → Clocktime],
function[process → Clocktime], Clocktime,
function[process → bool] → bool] =
$(\lambda\, \gamma, \theta, X, \text{ppred}: (\forall\, p_3: \text{ppred}(p_3) \supset |\gamma(p_3) - \theta(p_3)| \leq X))$
N: process

N_0: **Axiom** $N > 0$

N_maxfaults: **Axiom** maxfaults $\leq N$

precision_enhancement_ax: **Axiom**
$\text{count}(\text{ppred}, N) \geq N - \text{maxfaults}$
$\wedge\ \text{okay\_Readpred}(\gamma, Y, \text{ppred})$
$\wedge\ \text{okay\_Readpred}(\theta, Y, \text{ppred})$
$\wedge\ \text{okay\_pairs}(\gamma, \theta, X, \text{ppred}) \wedge \text{ppred}(p) \wedge \text{ppred}(q)$
$\supset |\mathit{cfn}(p, \gamma) - \mathit{cfn}(q, \theta)| \leq \pi(X, Y)$

correct_count: **Axiom** $\text{count}((\lambda\, p: \text{correct}(p, t)), N) \geq N - \text{maxfaults}$

accuracy_preservation_ax: **Axiom**
$\text{okay\_Readpred}(\gamma, X, \text{ppred})$
$\wedge\ \text{count}(\text{ppred}, N) \geq N - \text{maxfaults} \wedge \text{ppred}(p) \wedge \text{ppred}(q)$
$\supset |\mathit{cfn}(p, \gamma) - \gamma(q)| \leq \alpha(X)$

Figure 4: Conditions on Logical Clocks

agreement: **Lemma** $\beta \leq r_{min}$
$\wedge\ \mu \leq \delta_S \wedge \pi(2 * \Lambda + 2 * \beta \star \rho, \delta_S + 2 * ((r_{max} + \beta) \star \rho + \Lambda)) \leq \delta_S$
$\wedge\ \delta_S + 2 * r_{max} \star \rho \leq \delta$
$\wedge\ \alpha(\delta_S + 2 * (r_{max} + \beta) \star \rho + 2 * \Lambda) + \Lambda + 2 * \beta \star \rho \leq \delta$
$\wedge\ t \geq 0 \wedge \text{correct}(p, t) \wedge \text{correct}(q, t)$
$\supset |VC_p(t) - VC_q(t)| \leq \delta$

Figure 5: Main Theorem

okaymaxsync: function[nat, Clocktime → bool] =

$(\lambda\, i, X\colon (\forall\, p, q\colon$

$\mathrm{correct}(p, t^i_{p,q}) \wedge \mathrm{correct}(q, t^i_{p,q})$

$\supset |IC^i_p(t^i_{p,q}) - IC^i_q(t^i_{p,q})| \leq X))$

lemma_2: **Lemma** $\beta \leq r_{min}$

$\wedge\, \mu \leq X \wedge \pi(2 * \Lambda + 2 * \beta \star \rho, X + 2 * ((r_{max} + \beta) \star \rho + \Lambda)) \leq X$

$\supset$ okaymaxsync(i, X)

Figure 6: Skew immediately following resynchronization

okayClocks: function[process, process, nat → bool] =

$(\lambda\, p, q, i\colon (\forall\, t\colon$

$t \geq 0 \wedge t < t^i_{(p \Uparrow q)[i]} \wedge \mathrm{correct}(p, t) \wedge \mathrm{correct}(q, t)$

$\supset |VC_p(t) - VC_q(t)| \leq \delta))$

lemma3_3: **Lemma** $\beta \leq r_{min}$

$\wedge\, \mu \leq \delta_S \wedge \pi(2 * \Lambda + 2 * \beta \star \rho, \delta_S + 2 * ((r_{max} + \beta) \star \rho + \Lambda)) \leq \delta_S$

$\wedge\, \delta_S + 2 * r_{max} \star \rho \leq \delta$

$\wedge\, \alpha(\delta_S + 2 * (r_{max} + \beta) \star \rho + 2 * \Lambda) + \Lambda + 2 * \beta \star \rho \leq \delta$

$\supset$ okayClocks(p, q, i)

Figure 7: Skew up to ith resynchronization