NASA
Technical
Paper
3349

1993

# Verification of Fault-Tolerant Clock Synchronization Systems

Paul S. Miner
*Langley Research Center*
*Hampton, Virginia*

National Aeronautics and
Space Administration
Office of Management
Scientific and Technical
Information Program

# Contents

# List of Tables



# List of Figures

## Summary

A critical function in a fault-tolerant computer architecture is the synchronization of the redundant computing elements. One means of accomplishing this is for each computing element to maintain a local clock that is periodically synchronized with the other clocks in the system. The synchronization algorithm must include safeguards to ensure that failed components do not corrupt the behavior of good clocks. Reasoning about fault-tolerant clock synchronization is difficult because of the possibility of subtle interactions involving failed components. Therefore, mechanical proof systems are used to ensure that the verification of the synchronization system is correct.

In 1987, Schneider (Tech. Rep. 87-859, Cornell Univ.) presented a general proof of correctness for several fault-tolerant clock synchronization algorithms. Subsequently, Shankar (NASA CR-4386) verified Schneider's proof by using the mechanical proof system EHDM. This proof ensures that any system satisfying its underlying assumptions will provide Byzantine fault-tolerant clock synchronization. This paper explores the utility of Shankar's mechanization of Schneider's theory for the verification of clock synchronization systems.

In the course of this work, some limitations of Shankar's mechanically verified theory were encountered. These limitations include one assumption that is too strong and also insufficient support for reasoning about recovery from transient faults. With minor modifications to the other assumptions, a mechanically checked proof is provided that eliminates the overly strong assumption. In addition, the revised theory allows for proven recovery from transient faults.

Use of the revised theory is then illustrated with the verification of an abstract design of a fault-tolerant clock synchronization system. The fault-tolerant midpoint convergence function is proven with EHDM to satisfy the requirements of the theory. Then a design using this convergence function is shown to satisfy the remaining constraints.

# Chapter 1

# Introduction

At first glance, the development of fault-tolerant computer architectures does not appear to be a difficult problem. Clearly, three computers should be sufficient to survive a single fault. A simple majority vote should mask any errors caused by a failed component. However, to determine when to vote, the computers must be synchronized. This synchronization is easy with a perfect clock that coordinates actions among the redundant computing elements. Unfortunately, clocks also fail. Thus, each redundant computing element must maintain its own clock. No clock keeps perfect time; all drift with respect to some reference standard time. Similarly, clocks drift with respect to each other. Therefore, regular synchronization of the clocks of the redundant computing elements is necessary. An obvious algorithm for synchronizing clocks of three computers is for each to periodically *read* the clocks of the other two and then set its own clock to equal the mid value of the three observed values. Intuitively, this algorithm should work, but consider what happens if one clock fails so that it behaves in an arbitrary fashion. The classic example is given by Lamport and Melliar-Smith (ref. 1). Suppose that the clock for computer $A$ shows 1:00, the clock for computer $B$ shows 2:00, and the clock for computer $C$ has failed in such a way that when $A$ reads $C$'s clock it shows 0:00 and when $B$ reads $C$'s clock it shows 3:00. Clearly, neither $A$ nor $B$ has a compelling reason to adjust its clock and they may continue to drift apart. The presentation of Lamport and Melliar-Smith continues with a formal statement of the clock synchronization problem and presents three verified solutions. Subsequently, a number of other solutions to problems related to clock synchronization were developed, including those in references 2 through 7. A survey of the various approaches is given by Ramanathan, Shin, and Butler (ref. 8).

Schneider (ref. 9) recognized that the many approaches to clock synchronization can be presented as refinements of a single, verified paradigm. Shankar (ref. 10) provides a mechanical proof (using EHDM (ref. 11)) that Schneider's schema achieves Byzantine fault-tolerant clock synchronization, provided that 11 constraints are satisfied. (A failure that exhibits arbitrary or malicious behavior is called a Byzantine fault, in reference to the Byzantine Generals problem of Lamport, Shostak, and Pease (ref. 12).) One goal of this paper is to examine the utility of Shankar's mechanically checked version of Schneider's theory in the verification of a particular clock synchronization system.

The field of fault-tolerant computing is replete with examples of intuitively correct approaches that were later shown to be insufficient. In one system, the design of the fault-tolerance mechanism was cited as a major contributor to the unreliability of the system (ref. 13). Because of the extreme level of reliability required for many fault-tolerant systems, employing rigorous verification techniques is necessary. (An often quoted requirement for critical systems employed for civil air transport is a probability of catastrophic failure less than $10^{-9}$ for a 10-hour flight (ref. 14).) One such technique is the use of formal proof to establish that a design has certain properties. Additional certainty is gained by confirming the verification with a mechanical proof system, such as EHDM. Another benefit of machine-checked proofs is that the underlying assumptions are made explicit to help to clearly define the necessary verification conditions.

Shankar's verification of Schneider's protocol provides a trusted formal specification of a clock synchronization system. Many of the difficult aspects of the proof have been verified in a generic manner; all that is required to verify a synchronization system is to demonstrate that it meets the requirements of the general theory. This paper is a result of the first attempt to verify a design using Shankar's machine-checked theory (ref. 10). In the course of the verification, some difficulties were encountered with the underlying assumptions. The most significant problem was that one of the assumptions, bounded delay, was too strong. Bounded delay asserts that there is a bound on the elapsed time between synchronization events on any two good clocks. For some protocols, this property is the key required to maintain synchronization. The proof of bounded delay can be as difficult as the general synchronization property. This paper revises Shankar's general theory by modifying the remaining constraints to enable a general proof of bounded delay.

In an effort to demonstrate the applicability of formal proof techniques to the verification of highly reliable systems, the Langley Research Center is currently involved in the development of a formally verified Reliable Computing Platform (RCP) for real-time digital flight control (refs. 15, 16, and 17). The fault-tolerant clock synchronization circuit is intended to be part of a verified hardware base for the RCP. The primary intent of the RCP is to provide a verified fault-tolerant system that is proven to recover from a bounded number of transient faults. The current model of the system assumes (among other things) that the clocks are synchronized within a bounded skew (ref. 16). The clock synchronization circuitry also should be able to recover from transient faults. Originally, the interactive convergence algorithm (ICA) of Lamport and Melliar-Smith (ref. 1) was to be the basis for the clock synchronization system, the primary reason being the existence of a mechanical proof that the algorithm is correct (ref. 18). However, modifications to ICA to achieve transient-fault recovery are complicated. The fault-tolerant midpoint algorithm of Welch and Lynch (ref. 2) is more readily adapted to transient recovery.

Even though the clock synchronization circuit was designed to recover from transient faults, there was no support in the machine-checked theory for proven recovery from such failures. When the machine-checked theory was revised to remove the assumption of bounded delay, additional modifications were made to expand the theory to accommodate proven recovery from a bounded number of transient faults.

The synchronization circuit is designed to tolerate arbitrarily malicious permanent, intermittent, and transient hardware faults. A fault is defined as a physical perturbation altering the function implemented by a physical device. Intermittent faults are permanent physical defects that do not continuously alter the function of a device (e.g., a loose wire). A transient fault is caused by a one-shot, short-duration physical perturbation of a device (e.g., a cosmic ray or electromagnetic effect). This perturbation can result in any of the following situations:

1. Permanent damage to the device
2. No damage with a persistent error induced
3. No damage with the system recovering from the erroneous state

The first situation is classified as a permanent fault; the second and third are transient faults. A good design can eliminate the second situation by establishing a recovery path from all possible system states. Such a design is called self-stabilizing (ref. 19). Once the physical source of the fault is removed, the device can function correctly. The synchronization circuit is designed to automatically recover from a bounded number of transient failures.

Most proofs of fault-tolerant clock synchronization algorithms are by induction on the number of synchronization intervals. Usually, the base case of the induction, the initial skew, is assumed. The descriptions in references 1, 9, 10, and 18 all assume initial synchronization with no mention of how it is achieved. Others, including references 2, 4, 6, and 20, address the issue of initial synchronization and give descriptions of how it is achieved in varying degrees of detail. In proving an implementation correct, the details of initial synchronization cannot be ignored. If the initialization scheme is robust enough, it can also serve as a recovery mechanism from multiple correlated transient failures (as noted in ref. 20).

The chapters in this paper are arranged by decreasing generality. The most general results are presented first and are applicable to a number of designs. The use of the theory is then illustrated by application to a specific design. In Chapter 2, the definitions and constraints required by the general clock synchronization theory are presented. Chapter 3 presents the main revision made to Shankar's theory, which is removing the assumption of bounded delay. Chapter 4 presents mechanically checked proofs that the fault-tolerant midpoint convergence function satisfies the constraints required by the theory. In Chapter 5, a hardware realization of a fault-tolerant clock synchronization circuit is introduced and shown to satisfy the remaining constraints of the theory. Finally in section 6, the mechanisms for achieving initial synchronization and transient recovery are presented. Modifications to the theory to support the transient recovery arguments are also presented.

The information presented in this report was included in a thesis offered in partial fulfillment of the requirements for the Degree of Master of Science, The College of William and Mary in Virginia, Williamsburg, Virginia, 1992.

# Chapter 2

# Clock Definitions

A clock synchronization system ensures that the readings of two synchronized clocks differ by no more than a small amount $\delta$ for all time $t$. In addition, a fault-tolerant collection of clocks should maintain synchrony, even if a limited number of clocks have failed. Figure 2.1 illustrates a possible four-clock system that is designed to tolerate the failure of no more than one clock. Each nonfaulty clock provides a synchronized time reference $VC_p$ to local processing element $p$. This reference is guaranteed to be approximately synchronized with the corresponding value on any other good clock in the system. This guarantee is provided by an internal physical clock $PC_p$ and a distributed fault-tolerant clock synchronization algorithm executing in each of the redundant channels. A generalized view of the algorithm employed is

```
do forever {
   exchange clock values
   determine adjustment for this interval
   determine local time to apply correction
   when time, apply correction}
```

A system that implements this algorithm and satisfies the definitions and conditions presented in this chapter possesses the following property (presented in (ref. 10)):

**Theorem 2.1 (bounded skew)** *For any two clocks $p$ and $q$ that are nonfaulty at time $t$,*

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

In other words, the skew between good clocks is bounded by $\delta$.

## 2.1 Notation

A fault-tolerant clock synchronization system is composed of an interconnected collection of physically isolated clocks. Each redundant clock incorporates a physical oscillator that marks passage of time. Each oscillator drifts with respect to real time by a small amount. Physical clocks derived from these oscillators similarly drift with respect to each other. Following reference 1, the discussion of clocks involves two views of time. Real time

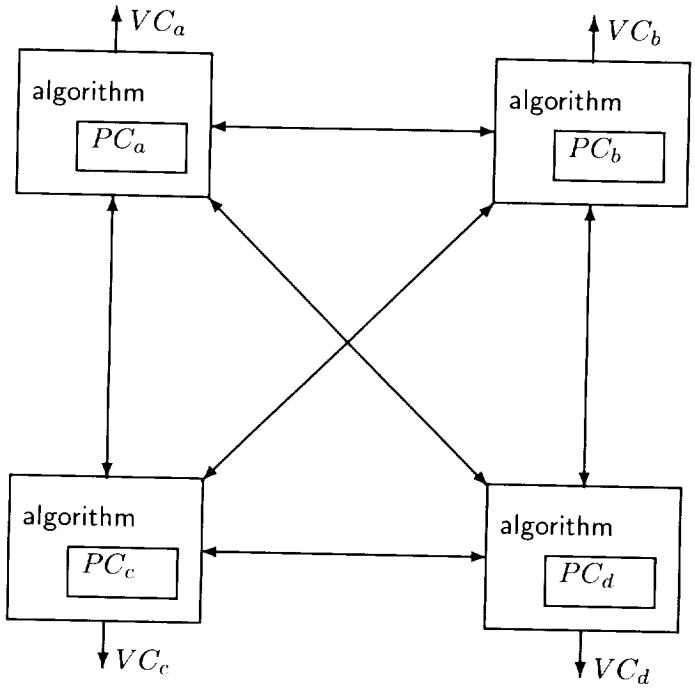



Figure 2.1: Four-clock system.

corresponds to an assumed Newtonian time frame; clock time is the measurement of this time frame by some clock. Identifiers representing real-time quantities will be denoted by lower case letters, e.g., $t, s$: **Var** time. Here, $t$ and $s$ are variables (in the logical theory) of type time. A declaration without the keyword **Var** defines a constant, e.g., $t_1$: time defines the constant $t_1$ of type time. Typically, time is taken as ranging over the real numbers. Clock time will be represented by upper case letters, e.g., $T, S$: **Var** Clocktime. Although Clocktime is often treated as ranging over the reals (refs. 2, 10, and 18), a physical realization of a clock marks time in discrete intervals. In this presentation Clocktime is assumed to range over the integers. The unit for both time and Clocktime is the tick. There are two sets of functions associated with the physical clocks[1]: functions mapping real time to clock time for each process $p$,[2]

$$PC_p : \mathsf{time} \rightarrow \mathsf{Clocktime}$$

and functions mapping clock time to real time,

$$pc_p : \mathsf{Clocktime} \rightarrow \mathsf{time}$$

[1]Shankar's presentation includes only the mappings from time to Clocktime. The mappings from Clocktime to time are added here because they are more natural representations for some of the proofs.

[2]Declarations of the form $f : \alpha \rightarrow \beta$ define a function $f$ with domain $\alpha$ and range $\beta$.

The notation $PC_p(t)$ represents the reading of $p$'s physical clock at real time $t$, and $pc_p(T)$ denotes the earliest real time that $p$'s clock reads $T$. By definition, $PC_p(pc_p(T)) = T$ for all $T$. In addition, we assume that $pc_p(PC_p(t)) \leq t < pc_p(PC_p(t) + 1)$.

The purpose of a clock synchronization algorithm is to make periodic adjustments to local clocks to keep a distributed collection of clocks within a bounded skew of each other. This periodic adjustment makes analysis difficult; therefore an interval clock abstraction is used in the proofs. Each process $p$ has an infinite number of interval clocks associated with it, each of these is indexed by the number of intervals since the beginning of the protocol. An interval corresponds to the elapsed time between adjustments to the virtual clock. These interval clocks are equivalent to adding an offset to the physical clock of a process. As with the physical clocks, they are characterized by two functions: $IC_p^i$ : time $\rightarrow$ Clocktime and $ic_p^i$ : Clocktime $\rightarrow$ time. If we let $adj_p^i$ : Clocktime denote the cumulative adjustment made to a clock as of the $i$th interval, we get the following definitions for the $i$th interval clock:

$$\begin{aligned} IC_p^i(t) &= PC_p(t) + adj_p^i \\ ic_p^i(T) &= pc_p(T - adj_p^i) \end{aligned}$$

From these definitions, it is simple to show $IC_p^i(ic_p^i(T)) = PC_p(pc_p(T - adj_p^i)) + adj_p^i = T$ for all $T$. Sometimes it is more useful to refer to the incremental adjustment made in a particular interval than to use a cumulative adjustment. By letting $ADJ_p^i = adj_p^{i+1} - adj_p^i$, we get the following equations relating successive interval clocks:

$$\begin{aligned} IC_p^{i+1}(t) &= IC_p^i(t) + ADJ_p^i \\ ic_p^{i+1}(T) &= ic_p^i(T - ADJ_p^i) \end{aligned}$$

A virtual clock, $VC_p$ : time $\rightarrow$ Clocktime, is defined in terms of the interval clocks by the equation

$$VC_p(t) = IC_p^i(t) \qquad (t_p^i \leq t < t_p^{i+1})$$

The symbol $t_p^i$ denotes the instant in real time that process $p$ begins the $i$th interval clock. Notice that there is no mapping from Clocktime to time for the virtual clock because $VC_p$ is not necessarily monotonic; the inverse relation might not be a function for some synchronization protocols. The definition of $VC_p(t)$ from the equations for $IC$ is illustrated in figure 2.2.

Synchronization protocols provide a mechanism for processes to read each other's clocks. The adjustment is computed as a function of these readings. In Shankar's presentation, the readings of remote clocks are captured in function $\Theta_p^{i+1}$ : process $\rightarrow$ Clocktime, where $\Theta_p^{i+1}(q)$ denotes process $p$'s estimate of $q$'s $i$th interval clock at real time $t_p^{i+1}$ (i.e., $IC_q^i(t_p^{i+1})$). Each process executes the same (higher order) convergence function, $cfn$ : (process, (process $\rightarrow$ Clocktime)) $\rightarrow$ Clocktime, to determine the proper correction to apply.[3] Shankar defines the cumulative adjustment in terms of the convergence function as follows:

[3]The domain of a higher order function can include functions. In this case, the second argument of $cfn$ is itself a function with domain process and range Clocktime.

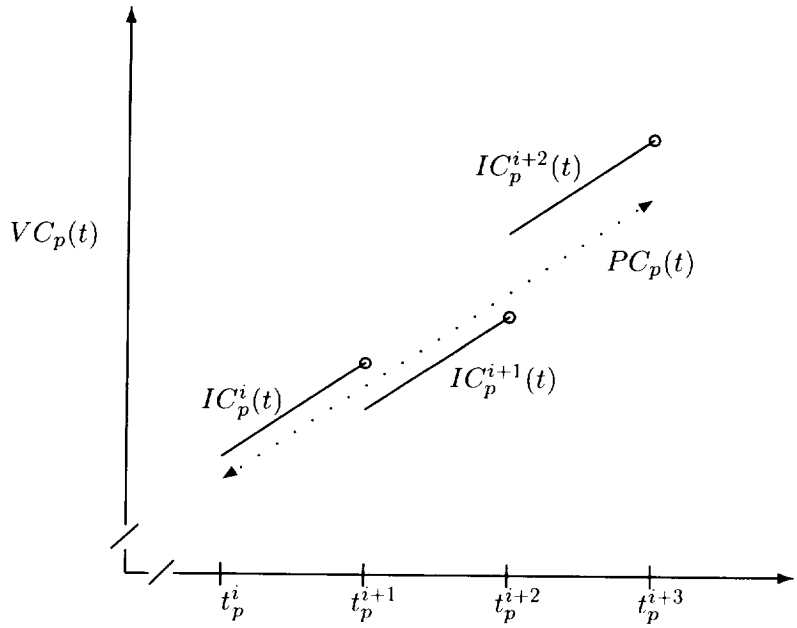



Figure 2.2: Determining $VC_p(t)$. Scale does not permit display of $IC_p$ as step function.

$$\begin{aligned} adj_p^{i+1} &= cfn(p, \Theta_p^{i+1}) - PC_p(t_p^{i+1}) \\ adj_p^0 &= 0 \end{aligned}$$

The following can be simply derived from the preceding definitions:

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

Using some of these equations and the conditions presented in section 2.2, Shankar mechanically verified Schneider's paradigm. Chapter 3 presents a general argument for satisfying one of the assumptions of Shankar's proof. The argument requires some modifications to Shankar's constraints and introduces a few new assumptions; in addition, some of the existing constraints are rendered unnecessary.

A new constant, $R$ : Clocktime, is introduced which denotes the expected duration of a synchronization interval as measured by clock time. (That is, in the absence of drift and jitter, no correction is necessary for the clocks to remain synchronized. In this case, the duration of an interval is exactly $R$ ticks.) We also introduce a collection of distinguished clock times $S^i$ : Clocktime, such that $S^i = iR + S^0$ and $S^0$ is a particular clock time in the first synchronization interval. We also introduce the abbreviation $s_p^i$ defined

as equal to $ic_p^i(S^i)$. The only constraints on $S^i$ are that, for each nonfaulty clock $p$ and real times $t_1$ and $t_2$,

$$(VC_p(t_1) = S^i) \wedge (VC_p(t_2) = S^i) \supset t_1 = t_2$$

and some real time $t$ exists, such that

$$VC_p(t) = S^i$$

The rationale for these constraints is that we want to unambiguously define a clock time in each synchronization interval to simplify the arguments necessary to bound separation of good clocks. If we choose a clock time near the instant that an adjustment is applied, it is possible that the $VC$ will never read that value because the clock has been adjusted ahead or that the value will be reached twice because of the clock being adjusted back. In reference 2, the chosen unambiguous event is the clock time that each good processor uses to initiate the exchange of clock values. For other algorithms, any clock time sufficiently removed from the time of the adjustment will suffice. A simple way to satisfy these constraints is to ensure that for all $i$,

$$S^i + ADJ_p^i < T_p^{i+1} < S^{i+1} - ADJ_p^i$$

where $T_p^{i+1} = IC_p^i(t_p^{i+1})$.

Table 2.1 summarizes the notation for the key elements required for a verified clock synchronization algorithm. Table 2.2 presents the many constants used in section 2.2. They are described when they are introduced in the text but are included here as a convenient reference.

## 2.2 Conditions

This section presents the assumptions required in the proof of theorem 2.1. The conditions can be separated into three main classes: abstract properties required of the convergence function, physical properties of the system, and various constraints on the length of the synchronization interval. Additional constraints are also determined by the proof of theorem 2.1. Some of these properties are taken directly from Shankar's presentation, whereas others are revised in order to facilitate verification of a clock synchronization system. Additional modifications are made to enable proofs of transient-fault recovery.

### 2.2.1 Properties of Convergence Function

Synchronization algorithms use a convergence function $cfn(p, \theta)$ to determine the adjustment required to maintain synchrony. The general theory requires that the convergence function satisfy three properties: *translation invariance*, *precision enhancement*, and *accuracy preservation*. Shankar mechanically proves that the interactive convergence function of Lamport and Melliar-Smith (ref. 1) satisfies these three conditions. A mechanically checked proof that the fault-tolerant midpoint function used by Welch and Lynch (ref. 2) satisfies these conditions is presented in Chapter 4 and was previously reported

Table 2.1: Clock Notation

| Notation | Definition |
|---|---|
| $PC_p(t)$ | Reading of $p$'s physical clock at real time $t$ |
| $pc_p(T)$ | Earliest real time that $p$'s physical clock reads $T$ |
| $IC_p^i(t)$ | Reading of $p$'s $i$th interval clock at real time $t$ |
| $ic_p^i(T)$ | Earliest real time that $p$'s $i$th interval clock reads $T$ |
| $VC_p(t)$ | Reading of $p$'s virtual clock at time $t$ |
| $T^0$ | Clocktime at beginning of protocol (for all good clocks) |
| $T_p^{i+1}$ | Clocktime for $VC_p$ to switch from $i$th to $(i+1)$th interval clock |
| $t_p^i$ | Real time that processor $p$ begins $i$th synchronization interval ($t_p^{i+1} = ic_p^i(T_p^{i+1})$) |
| $R$ | Clocktime duration of synchronization interval |
| $S^0$ | Special Clocktime in initial interval |
| $S^i$ | Unambiguous clock time in interval $i$; $S^i = iR + S^0$ |
| $s_p^i$ | Abbreviation for $ic_p^i(S^i)$ |
| $adj_p^i$ | Cumulative adjustment to $p$'s physical clock up through $t_p^i$ |
| $ADJ_p^i$ | Abbreviation for $adj_p^{i+1} - adj_p^i$ |
| $\Theta_p^{i+1}$ | Array of clock readings (local to $p$) such that $\Theta_p^i(q)$ is $p$'s reading of $q$'s $i$th interval clock at $t_p^{i+1}$ |
| $cfn(p, \Theta_p^{i+1})$ | Convergence function executed by $p$ to establish $VC_p(t_p^{i+1})$ |

Table 2.2: Constants

| Constant | Definition |
|---|---|
| $\delta_S$ : Clocktime | Bound on skew at beginning of protocol |
| $\delta$ : Clocktime | Bound on skew for all time |
| $\rho$ : number | Allowable drift rate for a good clock, $0 \leq \rho \ll 1$ |
| $\beta'$ : time | Maximum elapsed time from $s_p^i$ to $s_q^i$ ($p$ and $q$ working) |
| $\beta$ : time | Maximum elapsed time from $t_p^i$ to $t_q^i$ ($p$ and $q$ working) |
| $\beta_{\text{read}}$ : time | Maximum separation between $s_p^i$ and $s_q^i$, for $p$ to accurately read $q$, $\beta' \leq \beta_{\text{read}} < R/2$ |
| $r_{min}$ : time | Minimum elapsed time from $t_p^i$ to $t_p^{i+1}$ for good $p$ |
| $r_{max}$ : time | Maximum elapsed time from $t_p^i$ to $t_p^{i+1}$ for good $p$ |
| $\Lambda$ : Clocktime | Bound on error reading a remote clock |
| $\Lambda'$ : number | Reformulated error bound for reading a remote clock |
| $\alpha(\beta' + 2\Lambda')$ : number | Bound on $ADJ_p^i$ for good $p$ and all $i$ |

in reference 21. Schneider presents proofs that a number of other protocols satisfy these properties in reference 9. The conditions in this section are unchanged from Shankar's presentation.

The constraints on the convergence function assume a bound on the number of faults to be tolerated. This condition is stated here as condition 1; in Shankar's presentation, this was condition 8.

**Condition 1 (bounded faults)** *At any time $t$, the number of faulty processes is at most $F$.*

*Translation invariance* means that the value obtained by adding $X$ : Clocktime to the result of the convergence function should be the same as adding $X$ to each of the clock readings used in evaluating the convergence function. This was condition 9 in Shankar's presentation. The statement of this condition adapts notation from the lambda calculus. The symbol $\lambda$ is used to define an unnamed function. For example, $\lambda x.x + 2$ defines a function of one argument $x$ that returns the sum of $x$ and 2. For a detailed treatment of the lambda calculus, see reference 22.

**Condition 2 (translation invariance)** *For any function $\theta$ mapping clocks to clock values,*

$$cfn(p, (\lambda n : \theta(n) + X)) = cfn(p, \theta) + X$$

*Precision enhancement* is a formalization of the concept that, after executing the convergence function, the values of interest should be close together. Essentially, if the arguments presented to the convergence function are sufficiently similar, there is a bound on the difference of the results. In the proof of theorem 2.1, this condition ensures that if a large enough collection of good clocks is synchronized in one interval, then they will still be synchronized in the next. This was Shankar's condition 10.

**Condition 3 (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 $\gamma$ and $\theta$ satisfying the conditions*

1. *For any $l$ in $C$, $|\gamma(\ell) - \theta(\ell)| \leq X$*
2. *For any $l$, $m$ in $C$, $|\gamma(\ell) - \gamma(m)| \leq Y$*
3. *For any $l$, $m$ in $C$, $|\theta(\ell) - \theta(m)| \leq Y$*

*there is a bound $\pi(X, Y)$ such that*

$$|cfn(p, \gamma) - cfn(q, \theta)| \leq \pi(X, Y)$$

*Accuracy preservation* formalizes the notion that there should be a bound on the amount of correction applied in any synchronization interval. Accuracy preservation was condition 11 in Shankar's report.

**Condition 4 (accuracy preservation)** *Given any subset $C$ of the $N$ clocks with $|C| \geq N - F$ and clock readings $\theta$ such that, for any $l$ and $m$ in $C$, the bound $|\theta(\ell) - \theta(m)| \leq X$ holds, there is a bound $\alpha(X)$ such that for any $p$ and $q$ in $C$,*

$$|cfn(p, \theta) - \theta(q)| \leq \alpha(X)$$

For some convergence functions, the properties of precision enhancement and accuracy preservation can be weakened to simplify arguments for recovery from transient faults. Precision enhancement can be satisfied by many convergence functions even if $p$ and $q$ are not in $C$. Similarly, accuracy preservation can often be satisfied even when $p$ is not in $C$.

### 2.2.2 Physical Properties

Some of the conditions characterize the expected physical properties of the system. We rely on experimentation and engineering analysis to demonstrate these conditions.

The rate at which a good clock can drift from real time is bounded by a small positive constant $\rho$. Typically, $\rho < 10^{-5}$.

**Condition 5 (bounded drift)** *There is a nonnegative constant $\rho$ such that if $p$'s clock is nonfaulty during the interval from $T$ to $S (S \geq T)$, then*

$$\frac{S - T}{1 + \rho} \leq pc_p(S) - pc_p(T) \leq (1 + \rho)(S - T)$$

This condition replaces Shankar's condition 2. This assumption is stronger than Shankar's bound on drift, but the change is necessary to accommodate the integer representation of Clocktime. However, if the unit of time is taken to be a tick of Clocktime and Clocktime ranges over the integers, we can then derive the following bound on drift, which is sufficient for preserving Shankar's mechanical proof (with minor modifications):

**Corollary 5.1** *If $p$'s clock is not faulty during the interval from $t$ to $s$ then,*

$$\lfloor (s - t)/(1 + \rho) \rfloor \leq PC_p(s) - PC_p(t) \leq \lceil (1 + \rho)(s - t) \rceil$$

Note that with Shankar's algebraic relations defining various components of clocks, we can use these constraints to bound the drift of any interval clock ($ic_p^i$) for any $i$.

The following corollary to *bounded drift* limits the amount two good clocks can drift with respect to each other during the interval from $T$ to $S$.

**Corollary 5.2** *If clocks p and q are not faulty during the interval from T to S,*

$$|pc_p(S) - pc_q(S)| \leq |pc_p(T) - pc_q(T)| + 2\rho(S - T)$$

This corollary is used in bounding the amount of skew caused by drift during each synchronization interval.

We can also derive an additional corollary (adapted from lemma 2 of ref. 2).

**Corollary 5.3** *If clock p is not faulty during the interval from T to S,*

$$|(pc_p(S) - S) - (pc_p(T) - T)| \leq \rho|S - T|$$

This corollary recasts bounded drift into a form more useful for some proofs. A similar relation holds for $PC$.

All clock synchronization protocols require each process to obtain an estimate of the clock values for other processes within the system. The determination of this estimate is called *reading* the remote clock, even if there is no direct means to observe its value. Typically, some underlying communication protocol is employed which allows a fairly accurate estimate of other clocks in the system. Error in this estimate can be bounded but not eliminated. A discussion of different mechanisms for reading remote clocks can be found in Schneider (ref. 9). Shankar's statement of the bound on reading error is as follows:

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

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

This condition neglects an important point. In some protocols, the ability to accurately read another processor's clock is dependent on the clocks being already sufficiently synchronized. Therefore, we add a precondition stating that the real-time separation of $s_p^i$ and $s_q^i$ is bounded by some value of $\beta_{\text{read}}$. The precise value of $\beta_{\text{read}}$ required to ensure bounds on the reading error is determined by the implementation, but in all cases $\beta' \leq \beta_{\text{read}} < R/2$. Another useful observation is that an estimate of the value of a remote clock is subject to two interpretations. It can be used to approximate the difference in Clocktime that two clocks show at an instant of real time, or it can be used to approximate the separation in real time that two clocks show the same Clocktime.

**Condition 6 (reading error)** *For nonfaulty clocks p and q, if* $|s_p^i - s_q^i| \leq \beta_{\text{read}}$,

1. $|IC_q^i(t_p^{i+1}) - \Theta_p^{i+1}(q)| = |(\Theta_p^{i+1}(q) - IC_p^i(t_p^{i+1})) - (IC_q^i(t_p^{i+1}) - IC_p^i(t_p^{i+1}))| \leq \Lambda$
2. $|(\Theta_p^{i+1}(q) - IC_p^i(t_p^{i+1})) - (ic_p^i(T_p^{i+1}) - ic_q^i(T_p^{i+1}))| \leq \Lambda$
3. $|(\Theta_p^{i+1}(q) - IC_p^i(t_p^{i+1})) - (ic_p^i(S^i) - ic_q^i(S^i))| \leq \Lambda'$

The first clause just restates the existing read error condition to illustrate that the read error can also be viewed as the error in an estimate of the difference in readings of Clock-time, that is, the estimate allows us to determine approximately another clock's reading at a particular instant of time. The second clause recognizes that this difference can also be used to obtain an estimate of the time when a remote clock shows a particular Clocktime. For these relations, elements of type Clocktime and time are both treated as being of type number. Clocktime is a synonym for integer, which is a subtype of number, and time is a synonym for number. The third clause is the one used in this paper; it relates real-time separation of clocks when they read $S^i$ to the estimated difference when the correction is applied. A bound on this could be derived from the second clause, but it is likely that a tighter bound can be derived from the implementation. Since the guaranteed skew is derived, in part, from the read error, we wish this bound to be as tight as possible. For this reason, we add it as an assumption to be satisfied in the context of a particular implementation.

### 2.2.3 Interval Constraints

The conditions constraining the length of a synchronization interval are determined, in part, by the closeness of the initial synchronization. The following condition replaces Shankar's condition 1:

**Condition 7 (bounded delay init)** *For nonfaulty processes $p$ and $q$,*

$$|t_p^0 - t_q^0| \leq \beta' - 2\rho(S^0 - T^0)$$

A constraint similar to Shankar's can be easily derived from this new condition by using the constraint on clock drift. (Shankar's condition 1 is an immediate consequence of lemma 2.1.1 in appendix A.) An immediate consequence of this and condition 5 is that $|s_p^0 - s_q^0| \leq \beta'$.

Shankar assumes a bound on the duration of the synchronization interval.

**Shankar's Condition 3 (bounded interval)** *For nonfaulty clock $p$,*

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

The terms $r_{min}$ and $r_{max}$ are uninstantiated constants. In this formulation, a nominal duration ($R$) of an interval is assumed determined from the implementation. We set a lower bound on $R$ by placing restrictions on the events $S^i$. This restriction is done by bounding the amount of adjustment that a nonfaulty process can apply in any synchronization interval. In Chapter 3, the term $\boldsymbol{\alpha}(\beta' + 2\Lambda')$ is shown to bound $|ADJ_p^i|$ for nonfaulty process $p$. The function $\boldsymbol{\alpha}$ is introduced in condition 4, $\beta'$ is a bound on the separation of clocks at a particular Clocktime in each interval, and $\Lambda'$ bounds the error in estimating the value of a remote clock.

**Condition 8 (bounded interval)** *For nonfaulty clock p,*

$$S^i + \alpha(\beta' + 2\Lambda') < T_p^{i+1} < S^{i+1} - \alpha(\beta' + 2\Lambda')$$

By remembering that $S^i = iR + S^0$, it is easy to see that $R > 2\alpha(\beta' + 2\Lambda')$. Clearly, we can define $r_{min}$ as $(R - \alpha(\beta' + 2\Lambda'))/(1 + \rho)$ and $r_{max}$ as $(1 + \rho)(R + \alpha(\beta' + 2\Lambda'))$.

We need a condition to ensure that process $q$ does not start its $(i + 1)$th clock before process $p$ starts its $i$th clock. The following condition is sufficient to meet this requirement, which is a simple restatement of Shankar's condition 6, using the definition of $r_{min}$ from Shankar's condition 3.

**Condition 9 (nonoverlap)**

$$\beta \leq \frac{R - \alpha(\beta' + 2\Lambda')}{1 + \rho}$$

This condition essentially defines an additional constraint on $R$; namely, that $R \geq (1 + \rho)\beta + \alpha(\beta' + 2\Lambda')$, when $\beta$ bounds the maximum separation of $t_p^i$ and $t_q^i$.

### 2.2.4 Constraints on Skew

Shankar assumes the following additional conditions for an algorithm to be verified in this theory. These additional constraints were determined in the course of his proof of theorem 2.1.

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

These conditions relate the skew $\delta$ guaranteed by the theory with the properties of precision enhancement and accuracy preservation.

When Clocktime was changed to range over the integers, these conditions had to be modified. The bounds were altered to correspond to the revised version of bounded drift. Shankar's version of bounded drift was converted to correspond to corollary 5.1. (This is stated as axioms rate_1 and rate_2 in module clockassumptions (appendix A).) The mechanical proof was rerun and yielded the following constraints:

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

The arguments used are identical to those presented by Shankar. The only difference is that additional manipulations were needed with the floor and ceiling functions in order to complete the proof. Appendix A contains the proof chain analysis which confirms that these constraints are sufficient to prove theorem 2.1.

Since $\rho$ is typically very small ($< 10^{-5}$), the above reworked constraints appear overly conservative. It is possible to prove theorem 2.1 by assuming the following:

1. $4\rho r_{max} + \pi(\lfloor 2\Lambda' + 2 \rfloor, \lfloor \beta' + 2\Lambda' \rfloor) \leq \beta'$
2. $\lceil (1+\rho)\beta' + 2\rho r_{max} \rceil \leq \delta$
3. $\alpha(\lfloor \beta' + 2\Lambda' \rfloor) + \Lambda + \lceil 2\rho\beta \rceil + 1 \leq \delta$

A proof sketch can be found in appendix A.

### 2.2.5 Unnecessary Conditions

Two of the conditions presented in Shankar's report were found to be unnecessary. Shankar and Schneider both assume the following conditions in their proofs:

**Shankar's Condition 4 (bounded delay)** *For nonfaulty clocks p and q,*

$$|t_q^i - t_p^i| \leq \beta$$

The condition states that the elapsed time between two processes starting their $i$th interval clock is bounded. This property is closely related to the end result of the general theory (bounded skew) and should be derived in the context of an arbitrary algorithm.

The related property for nonfaulty clocks $p$ and $q$,

$$|s_q^i - s_p^i| \leq \beta'$$

is proven independently of the algorithm in Chapter 3. This gives sufficient information to prove bounded delay directly from the algorithm; however, this proof depends on the interpretation of $T_p^{i+1}$. Two interpretations and their corresponding proofs are also given in Chapter 3.

The next condition states that all good clocks begin executing the protocol at the same instant of real time (and defines that time to be 0):

**Shankar's Condition 5 (initial synchronization)** *For nonfaulty clock p,*

$$t_p^0 = 0$$

It is not possible to guarantee that all clocks start at the same instance of time; thus, no implementation can guarantee this property. This property is used, in conjunction with Shankar's condition 1, to ensure the base case of the induction required to prove

theorem 2.1. By defining $t_p^0 = ic_p^0(T^0)$, we can readily prove the base case with conditions 5 and 7. Some constant clock time known to all good clocks is represented by $T^0$ (i.e., $T^0$ is the clock time in the initial state). The definition of $t_p^0$ states that all nonfaulty clocks start the protocol at the same Clocktime.

# Chapter 3

# General Solution for Bounded Delay

The condition of bounded delay asserts that any two nonfaulty clocks begin each synchronization interval at approximately the same real time. This property is nearly as strong as theorem 2.1. In fact, the result follows immediately for some synchronization protocols. This chapter establishes, for many synchronization protocols, that the condition of bounded delay follows from the remaining conditions enumerated in Chapter 2.

Schneider's schema assumes that

$$|t_p^i - t_q^i| \leq \beta$$

for good clocks $p$ and $q$, where $t_p^i$ denotes the real time that clock $p$ begins its $i$th interval clock (this is condition 4 in Shankar's presentation). Anyone wishing to use the generalized proof to verify the correctness of an implementation must prove that this property is satisfied by their implementation. For the algorithm presented in reference 2, this is a nontrivial proof.

The difficulty stems, in part, from the inherent ambiguity in the interpretation of $t_p^{i+1}$. Relating the event to a particular clock time is difficult because it serves as a crossover point between two interval clocks. The logical clock implemented by the algorithm undergoes an instantaneous shift in its representation of time. Thus the local clock readings surrounding the time of adjustment may show a particular clock time twice or never. The event $t_p^{i+1}$ is determined by the algorithm to occur when $IC_p^i(t) = T_p^{i+1}$; that is, $T_p^{i+1}$ is the clock time for applying the adjustment $ADJ_p^i = (adj_p^{i+1} - adj_p^i)$. This also means that $t_p^{i+1} = ic_p^i(T_p^{i+1})$. In an instantaneous adjustment algorithm there are at least two possibilities:

1. $T_p^{i+1} = (i+1)R + T^0$,
2. $T_p^{i+1} = (i+1)R + T^0 - ADJ_p^i$

A more stable frame of reference is needed for bounding the separation of events. Welch and Lynch (ref. 2) exploit their mechanism for reading remote clocks to provide this frame

of reference. Every clock in the system sends a synchronization pulse when its virtual clock reads $S^i = iR + S^0$, where $S^0$ denotes the first exchange of clock values. Let $s_p^i$ be an abbreviation for $ic_p^i(S^i)$. If we ignore any implied interpretation of event $s_p^i$ and just select values of $S^i$ which satisfy condition 8, we have sufficient information to prove bounded delay for an arbitrary algorithm. These results were previously presented in reference 23.

## 3.1 Bounded Delay Offset

The general proof follows closely an argument given in reference 2. The proof adapted is that of theorem 4 of reference 2, section 6. We wish to prove for good clocks $p$ and $q$ that

$$|t_p^i - t_q^i| \leq \beta$$

To establish this, we must first prove the following theorem:

**Theorem 3.1** *(bounded delay offset) For nonfaulty clocks $p$ and $q$ and for $i \geq 0$,*

*(a) If $i \geq 1$, then $|ADJ_p^{i-1}| \leq \alpha(\beta' + 2\Lambda')$*
*(b) $|s_p^i - s_q^i| \leq \beta'$*

**Proof:** The proof of theorem 3.1 is by induction on $i$. The base case ($i = 0$) is trivial; part (a) is vacuously true and part (b) is a direct consequence of conditions 7 and 5.

By assuming that parts (a) and (b) are true for $i$, we proceed by showing they hold for $i + 1$.

To prove the induction step for theorem 3.1(a), we begin by recognizing that

$$ADJ_p^{(i+1)-1} = adj_p^{i+1} - adj_p^i = cfn(p, \Theta_p^{i+1}) - IC_p^i(t_p^{i+1})$$

Because $IC_p^i(t_p^{i+1}) = \Theta_p^{i+1}(p)$ (no error in reading own clock), we have an instance of accuracy preservation:

$$|cfn(p, \Theta_p^{i+1}) - \Theta_p^{i+1}(p)| \leq \alpha(X)$$

All that is required is to show that $\beta' + 2\Lambda'$ substituted for $X$ satisfies the hypotheses of accuracy preservation.

We need to establish that for good $\ell, m$,

$$|\Theta_p^{i+1}(\ell) - \Theta_p^{i+1}(m)| \leq \beta' + 2\Lambda'$$

We know from the induction hypothesis that for good clocks $p$ and $q$,

$$|s_p^i - s_q^i| \leq \beta'$$

By reading error and the induction hypothesis, we get for nonfaulty clocks $p$ and $q$[4]

$$|(\Theta_p^{i+1}(q) - IC_p^i(t_p^{i+1})) - (s_p^i - s_q^i)| \leq \Lambda'$$

[4]Recall that in this formulation, values of type time and Clocktime are both promoted to type number.

We proceed as follows:

$$\begin{aligned}
&|\Theta_p^{i+1}(\ell) - \Theta_p^{i+1}(m)| \\
&\quad = |(\Theta_p^{i+1}(\ell) - \Theta_p^{i+1}(m)) + (IC_p^i(t_p^{i+1}) - IC_p^i(t_p^{i+1})) \\
&\qquad + (s_p^i - s_p^i) + (s_\ell^i - s_\ell^i) + (s_m^i - s_m^i)| \\
&\quad \leq |s_\ell^i - s_m^i| + |(\Theta_p^{i+1}(\ell) - IC_p^i(t_p^{i+1})) - (s_p^i - s_\ell^i)| \\
&\qquad + |(\Theta_p^{i+1}(m) - IC_p^i(t_p^{i+1})) - (s_p^i - s_m^i)| \\
&\quad \leq \beta' + 2\Lambda'
\end{aligned}$$

We get the last step by substituting $\ell$ and $m$ for $p$ and $q$, respectively, in the induction hypothesis, then by using reading error twice, and by substituting first $\ell$ for $q$ and then $m$ for $q$.

The proof of the induction step for theorem 3.1(b) proceeds as follows. All supporting lemmas introduced in this section implicitly assume that theorems 3.1(a) and 3.1(b) are both true for $i$ and that theorem 3.1(a) is true for $i+1$. In the presentation of Welch and Lynch (ref. 2), they introduce a variant of precision enhancement. We restate it here in the context of the general protocol:

**Lemma 3.1.1** *For good clocks $p$ and $q$,*

$$|(s_p^i - s_q^i) - (ADJ_p^i - ADJ_q^i)| \leq \boldsymbol{\pi}(2\Lambda' + 2, \beta' + 2\Lambda')$$

**Proof:** We begin by recognizing that $ADJ_p^i = cfn(p, (\lambda\ell.\Theta_p^{i+1}(\ell) - IC_p^i(t_p^{i+1})))$ (and similarly for $ADJ_q^i$). A simple rearrangement of the terms gives us

$$|(s_p^i - s_q^i) - (ADJ_p^i - ADJ_q^i)| = |(ADJ_p^i - s_p^i) - (ADJ_q^i - s_q^i)|$$

We would like to use translation invariance to help convert this to an instance of precision enhancement. However, translation invariance only applies to values of type Clocktime (a synonym for integer). We need to convert the real values $s_p^i$ and $s_q^i$ to integer values while preserving the inequality. We do this via the integer floor and ceiling functions. Without loss of generality, assume that $(ADJ_p^i - s_p^i) \geq (ADJ_q^i - s_q^i)$. Thus,

$$\begin{aligned}
&|(ADJ_p^i - s_p^i) - (ADJ_q^i - s_q^i)| \\
&\quad \leq |(ADJ_p^i - \lfloor s_p^i \rfloor) - (ADJ_q^i - \lceil s_q^i \rceil)| \\
&\quad = |cfn(p, (\lambda\ell.\Theta_p^{i+1}(\ell) - IC_p^i(t_p^{i+1}) - \lfloor s_p^i \rfloor)) \\
&\qquad - cfn(q, (\lambda\ell.\Theta_q^{i+1}(\ell) - IC_q^i(t_q^{i+1}) - \lceil s_q^i \rceil))|
\end{aligned}$$

All that is required is to demonstrate that if

$$(\lambda\ell.\Theta_p^{i+1}(\ell) - IC_p^i(t_p^{i+1}) - \lfloor s_p^i \rfloor) = \gamma$$

and

$$(\lambda\ell.\Theta_q^{i+1}(\ell) - IC_q^i(t_q^{i+1}) - \lceil s_q^i \rceil) = \theta$$

they satisfy the hypotheses of precision enhancement.

We know from reading error and the induction hypothesis that

$$|(\Theta_p^{i+1}(\ell) - IC_p^i(t_p^{i+1})) - (s_p^i - s_\ell^i)| \leq \Lambda'$$

To satisfy the first hypothesis of precision enhancement, we notice that

$$\begin{aligned}
&|(\lambda\ell.\Theta_p^{i+1}(\ell) - IC_p^i(t_p^{i+1}) - \lfloor s_p^i \rfloor)(\ell) - (\lambda\ell.\Theta_q^{i+1}(\ell) - IC_q^i(t_q^{i+1}) - \lceil s_q^i \rceil)(\ell)| \\
&= |(\Theta_p^{i+1}(\ell) - IC_p^i(t_p^{i+1}) - \lfloor s_p^i \rfloor) - (\Theta_q^{i+1}(\ell) - IC_q^i(t_q^{i+1}) - \lceil s_q^i \rceil)| \\
&= |((\Theta_p^{i+1}(\ell) - IC_p^i(t_p^{i+1})) - (\lfloor s_p^i \rfloor - s_\ell^i)) \\
&\quad -((\Theta_q^{i+1}(\ell) - IC_q^i(t_q^{i+1})) - (\lceil s_q^i \rceil - s_\ell^i))| \\
&\leq 2\Lambda' + 2
\end{aligned}$$

Therefore, we can substitute $2\Lambda' + 2$ for $X$ to satisfy the first hypothesis of precision enhancement.

To satisfy the second and third hypotheses, we proceed as follows (the argument presented is for $(\lambda\ell.\Theta_p^{i+1}(\ell) - IC_p^i(t_p^{i+1}) - \lfloor s_p^i \rfloor) = \gamma$). We need a value of $Y$ such that

$$|(\lambda\ell.\Theta_p^{i+1}(\ell) - IC_p^i(t_p^{i+1}) - \lfloor s_p^i \rfloor)(\ell) - (\lambda\ell.\Theta_p^{i+1}(\ell) - IC_p^i(t_p^{i+1}) - \lfloor s_p^i \rfloor)(m)| \leq Y$$

We know that

$$\begin{aligned}
&|(\lambda\ell.\Theta_p^{i+1}(\ell) - IC_p^i(t_p^{i+1}) - \lfloor s_p^i \rfloor)(\ell) - (\lambda\ell.\Theta_p^{i+1}(\ell) - IC_p^i(t_p^{i+1}) - \lfloor s_p^i \rfloor)(m)| \\
&= |(\Theta_p^{i+1}(\ell) - IC_p^i(t_p^{i+1}) - \lfloor s_p^i \rfloor) - (\Theta_p^{i+1}(m) - IC_p^i(t_p^{i+1}) - \lfloor s_p^i \rfloor)| \\
&= |\Theta_p^{i+1}(\ell) - \Theta_p^{i+1}(m)|
\end{aligned}$$

The argument in theorem 3.1(a) shows that this value is bounded by $\beta' + 2\Lambda'$ which is the desired $Y$ for the remaining hypotheses of precision enhancement. ∎

Now we bound the separation of $ic_p^{i+1}(T)$ and $ic_q^{i+1}(T)$ for all $T$.

**Lemma 3.1.2** *For good clocks $p$ and $q$ and clock time $T$,*

$$|ic_p^{i+1}(T) - ic_q^{i+1}(T)| \leq 2\rho(|T - S^i| + \boldsymbol{\alpha}(\beta' + 2\Lambda')) + \boldsymbol{\pi}(2\Lambda' + 2, \beta' + 2\Lambda')$$

**Proof:** The proof is taken verbatim (with the exception of notational differences) from reference 2, lemma 10.

Note that

$$ic_p^{i+1}(T) = ic_p^i(T - ADJ_p^i) \text{ and } ic_q^{i+1}(T) = ic_q^i(T - ADJ_q^i)$$

Now

$$\begin{aligned}
&|ic_p^{i+1}(T) - ic_q^{i+1}(T)| \\
&\leq |ic_p^i(T - ADJ_p^i) - s_p^i - (T - ADJ_p^i - S^i)| \\
&\quad + |ic_q^i(T - ADJ_q^i) - s_q^i - (T - ADJ_q^i - S^i)| \\
&\quad + |(s_p^i - s_q^i) - (ADJ_p^i - ADJ_q^i)|
\end{aligned}$$

The three terms are bounded separately. By corollary 5.3 of bounded drift (condition 5), we get

$$\begin{aligned}
&|ic_p^i(T - ADJ_p^i) - s_p^i - (T - ADJ_p^i - S^i)| \\
&\leq \rho|T - S^i - ADJ_p^i| \\
&\leq \rho(|T - S^i| + \boldsymbol{\alpha}(\beta' + 2\Lambda'))
\end{aligned}$$

from theorem 3.1(a) for $i+1$. The second term is similarly bounded. Lemma 3.1.1 bounds the third term. Adding the bounds and simplifying gives the result. ∎

This leads to the desired result:

**Lemma 3.1.3** *For good clocks $p$ and $q$,*

$$|s_p^{i+1} - s_q^{i+1}| \leq 2\rho(R + \boldsymbol{\alpha}(\beta' + 2\Lambda')) + \boldsymbol{\pi}(2\Lambda' + 2, \beta' + 2\Lambda') \leq \beta'$$

**Proof:** This is simply an instance of lemma 3.1.2 with $S^{i+1}$ substituted for $T$. ∎
This completes the proof of theorem 3.1. Algebraic manipulations on the inequality

$$2\rho(R + \boldsymbol{\alpha}(\beta' + 2\Lambda')) + \boldsymbol{\pi}(2\Lambda' + 2, \beta' + 2\Lambda') \leq \beta'$$

give us an upper bound for $R$.

## 3.2 Bounded Delay for Two-Algorithm Schemata

We begin by noticing that both instantaneous adjustment schemata presented at the beginning of this chapter allow for a simple derivation of $\beta$ that satisfies the condition of bounded delay (Shankar's condition 4). Notice that knowledge of the algorithm is required in order to fully establish this property.

**Theorem 3.2 (bounded delay)** *For nonfaulty clocks $p, q$ employing either of the two instantaneous adjustment schemata presented, there is a $\beta$ such that,*

$$|t_p^i - t_q^i| \leq \beta$$

**Proof:** It is important to remember that $t_p^{i+1} = ic_p^i(T_p^{i+1}) = ic_p^{i+1}(T_p^{i+1} + ADJ_p^i)$.

1. When $T_p^{i+1} = (i+1)R + T^0$, let $\beta = 2\rho(R - (S^0 - T^0)) + \beta'$
   In this case, since $T_p^{i+1} = T_q^{i+1} = (i+1)R + T^0$, all that is required is a simple application of corollary 5.2 and expanding the definition of $S^i$; that is, $S^i = iR + S^0$.

$$|t_p^{i+1} - t_q^{i+1}| \leq |s_p^i - s_q^i| + 2\rho((i+1)R + T^0 - S^i) \leq \beta' + 2\rho(R - (S^0 - T^0))$$

2. When $T_p^{i+1} = (i+1)R + T^0 - ADJ_p^i$, let $\beta = \beta' - 2\rho(S^0 - T^0)$
   This case requires the observation that $T_p^{i+1} + ADJ_p^i = T_q^{i+1} + ADJ_q^i = ((i+1)R + T^0)$. By substituting $(i+1)R + T^0$ for $T$ in lemma 3.1.2 and remembering that $S^i = iR + S^0$, we get

$$|t_p^{i+1} - t_q^{i+1}| \leq 2\rho((R - (S^0 - T^0)) + \alpha(\beta' + 2\Lambda')) + \pi(2\Lambda' + 2, \beta' + 2\Lambda')$$

   We know that

$$2\rho(R + \alpha(\beta' + 2\Lambda')) - 2\rho(S^0 - T^0) + \pi(2\Lambda' + 2, \beta' + 2\Lambda') \leq \beta' - 2\rho(S^0 - T^0)$$

   Simple algebra completes the proof of this case.

Condition 7 establishes $|t_p^0 - t_q^0| \leq \beta$ for both of these schemata. ∎

This result has no impact on the proofs performed by Shankar. The only difference is that bounded delay is no longer an assumption. However, it is possible that some bounds and arguments can be improved.

## 3.3 EHDM Proofs of Bounded Delay

The EHDM (version 5.2) proofs and supporting definitions and axioms are in the modules delay, delay2, delay3, and delay4. LaTeX-formatted listings of these modules are in appendix B. A slightly modified version of Shankar's module clockassumptions is also included in appendix A for completeness. Some of the revised constraints presented in Chapter 2 are in module delay. The most difficult aspect of the proofs was determining a reasonable predicate to express *nonfaulty clocks*. Since we would like to express transient-fault recovery in the theory, it is necessary to avoid the axiom correct_closed from Shankar's module clockassumptions. This axiom has not yet been removed from the general theory. None of the proofs of *bounded delay offset* depend on it, however. The notion of nonfaulty clocks is expressed by the following from module delay:

correct_during: function[process, time, time → bool] =
  ( λ p, t, s : t ≤ s ∧ ( ∀ t₁ : t ≤ t₁ ∧ t₁ ≤ s ⊃ correct(p, t₁)))
wpred: function[event → function[process → bool]]
rpred: function[event → function[process → bool]]
wvr_pred: function[event → function[process → bool]] =
  ( λ i : ( λ p : wpred(i)(p) ∨ rpred(i)(p)))

wpred_ax: **Axiom** count(wpred(i), N) ≥ N − F

wpred_correct: **Axiom** wpred(i)(p) ⊃ correct_during(p, $t_p^i$, $t_p^{i+1}$)

wpred_preceding: **Axiom** $\mathsf{wpred}(i+1)(p) \supset \mathsf{wpred}(i)(p) \vee \mathsf{rpred}(i)(p)$

wpred_rpred_disjoint: **Axiom** $\neg(\mathsf{wpred}(i)(p) \wedge \mathsf{rpred}(i)(p))$

wpred_bridge: **Axiom**
$\mathsf{wvr\_pred}(i)(p) \wedge \mathsf{correct\_during}(p, t_p^{i+1}, t_p^{i+2}) \supset \mathsf{wpred}(i+1)(p)$

Also, module delay3 states the following axiom:

recovery_lemma: **Axiom**
$\mathsf{delay\_pred}(i) \wedge \mathsf{ADJ\_pred}(i+1)$
$\wedge\ \mathsf{rpred}(i)(p) \wedge \mathsf{correct\_during}(p, t_p^{i+1}, t_p^{i+2}) \wedge \mathsf{wpred}(i+1)(q)$
$\supset |s_p^{i+1} - s_q^{i+1}| \leq \beta'$

There are two predicates defined, wpred and rpred. Wpred is used to denote a working clock; that is, it is not faulty and is in the proper state. Rpred denotes a process that is not faulty but has not yet recovered proper state information. Correct is a predicate taken from Shankar's proof that states whether a clock is fault free at a particular instance of real time. Correct_during is used to denote correctness of a clock over an interval of time. In order to reason about transient recovery it is necessary to provide an rpred that satisfies these relationships. If we do not plan on establishing transient recovery, let $\mathsf{rpred}(i) = (\lambda p : \mathit{false})$. In this case, axioms recovery_lemma and wpred_rpred_disjoint are vacuously true, and the remaining axioms are analogous to Shankar's correct_closed. This reduces to a system in which the only correct clocks are those that have been so since the beginning of the protocol. This is precisely what should be true if no recovery is possible.

The restated property of bounded drift is captured by axioms RATE_1 and RATE_2. The new constraints for bounded interval are rts_new_1 and rts_new_2. Bounded delay initialization is expressed by bnd_delay_init. The third clause of the new reading error is reading_error3. The other two clauses are not used in this proof. An additional assumption not included in the constraints given in Chapter 2 is that there is no error in reading your own clock. This is captured by read_self. All these can be found in module delay. In addition, a few assumptions were included to define interrelationships of some of the constants required by the theory.

The statement of theorem 3.1 is bnd_delay_offset in module delay2. The main step of the inductive proof for theorem 3.1(a) is captured by good_Readclock, which with accuracy preservation, was sufficient to establish bnd_delay_offset_ind_a. Theorem 3.1(b) is more involved. Lemma delay_prec_enh in module delay2 is the machine-checked version of lemma 3.1.1. Module delay3 contains the remaining proofs for theorem 3.1(b). Lemma 3.1.2 is presented as bound_future. The first two terms in the proof are bounded by lemma bound_future1; the third, by delay_prec_enh. Lemma bound_FIXTIME completes the proof.

Module delay4 contains the proofs that each of the proposed substitutions for $\beta$ satisfy the condition of bounded delay. Option 1 is captured by option1_bounded_delay, and option 2 is expressed by option2_bounded_delay. The EHDM proof chain status, demonstrating

that all proof obligations have been met, can also be found in appendix B. The task of mechanically verifying the proofs also forced some revisions to some hand proofs in an earlier draft of this paper. The errors revealed by the mechanical proof included invalid substitution of reals for integers and arithmetic sign errors.

Module new_basics restates Shankar's condition 8 as rts0_new and rts1_new with the substitutions suggested in section 2.2.3 for $r_{max}$ and $r_{min}$. These substitutions are proven to bound $t_p^{i+1} - t_p^i$ for each of the proposed algorithm schemata in module rmax_rmin. The revised statement of condition 9 can also be found in module new_basics; it is axiom nonoverlap. The modules new_basics and rmax_rmin provide the foundations for a mechanically checked version of the informal proof of theorem 2.1 given in appendix A.

## 3.4 New Theory Obligations

This revision to the theory leaves us with a set of conditions that are much easier to satisfy for a particular implementation. Establishing that an implementation is an instance of this extended theory requires the following obligations:

1. Prove the properties of translation invariance, precision enhancement, and accuracy preservation for the chosen convergence function
2. Derive bounds for reading error from the implementation (condition 6, clauses 1 and 3)
3. Solve the derived inequalities listed at the end of Chapter 2 with values determined from the implementation and properties of the convergence function
4. Satisfy the conditions of bounded interval and nonoverlap by using the derived values.
5. Identify data structures in the implementation that correspond to the algebraic definitions of clocks; show that the structures used in the implementation satisfy the definitions
6. Show that the implementation correctly executes an instance of the following algorithm schema:

   $i \leftarrow 0$
   do forever {
      exchange clock values
      determine adjustment for this interval
      determine $T^{i+1}$ (local time to apply correction)
      when $IC^i(t) = T^{i+1}$ apply correction; $i \leftarrow i + 1$ }

7. Provide a mechanism for establishing initial synchronization ($|t_p^0 - t_q^0| \leq \beta' - 2\rho(S^0 - T^0)$); ensure that $\beta'$ is as small as possible within the constraints of the aforementioned inequalities

8. If the protocol does not behave in the manner of either instantaneous adjustment option presented, it will be necessary to use another means to establish $\forall i : |t_p^i - t_q^i| \leq \beta$ from $\forall i : |s_p^i - s_q^i| \leq \beta'$

Requirement 1 is established in Chapter 4; requirements 2, 3, 4, 5, and 6 are demonstrated for an abstract design in Chapter 5; and requirement 7 is established in Chapter 6. The inequalities used in satisfying requirement 3 are the ones developed in the course of this work, even though the proof has not yet been subjected to mechanical verification. The proof sketch in appendix A is sufficient for the current development. Requirement 8 is trivially satisfied because the design described herein uses one of the two verified schemata.

# Chapter 4

# Fault-Tolerant Midpoint as an Instance of Schneider's Schema

The convergence function selected for the design described in Chapter 5 is the fault-tolerant midpoint used by Welch and Lynch in reference 2. The function consists of discarding the $F$ largest and $F$ smallest clock readings[5], and then determining the midpoint of the range of the remaining readings. Its formal definition is

$$cfn_{MID}(p,\theta) = \left\lfloor \frac{\theta_{(F+1)} + \theta_{(N-F)}}{2} \right\rfloor$$

where $\theta_{(m)}$ returns the $m$th largest element in $\theta$. This formulation of the convergence function is different from that used in reference 2. A proof of equality between the two formulations is not needed because it is shown that this formulation satisfies the properties required by Schneider's paradigm. For this function to make sense, we want the number of clocks in the system to be greater than twice the number of faults, $N \geq 2F + 1$. In order to complete the proofs, however, we need the stronger assumption that $N \geq 3F + 1$. Dolev, Halpern, and Strong have proven that clock synchronization is impossible (without authentication) if there are fewer than $3F + 1$ clocks. (See ref. 3.) Consider a system with $3F$ clocks. If $F$ clocks are faulty, then it is possible for two clusters of nonfaulty clocks to form, each of size $F$. Label the clusters $C_1$ and $C_2$. Without loss of generality, assume that the clocks in $C_1$ are faster than the clocks in $C_2$. In addition, the remaining $F$ clocks are faulty and are in cluster $C_F$. If the clocks in $C_F$ behave in a manner such that they all appear to be fast to the clocks in $C_1$ and slow to the clocks in $C_2$, clocks in each of the clusters will only use readings from other clocks within their own cluster. Nothing will prevent the two clusters from drifting farther apart. The one additional clock ensures that for any pair of good clocks, the ranges of the readings used in the convergence function overlap.

This section presents proofs that $cfn_{MID}(p,\theta)$ satisfies the properties required by Schneider's theory. The EHDM proofs are presented in appendix C and assume that a deterministic sorting algorithm arranges the array of clock readings.

[5]Remember that condition 1 defines $F$ to be the maximum number of faults tolerated.

The properties presented in this chapter are applicable for any clock synchronization protocol that employs the fault-tolerant midpoint convergence function. All that is required for a verified implementation is a proof that the function is correctly implemented and proofs that the other conditions have been satisfied. The weak forms of precision enhancement and accuracy preservation are used to simplify the arguments for transient recovery given in Chapter 6.

## 4.1 Translation Invariance

Recall that *translation invariance* states that the value obtained by adding Clocktime $X$ to the result of the convergence function should be the same as adding $X$ to each of the clock readings used in evaluating the convergence function. The condition is restated here for easy reference exactly as presented in Chapter 2.

**Condition 2 (translation invariance)** *For any function $\theta$ mapping clocks to clock values,*

$$cfn(p, (\lambda n : \theta(n) + X)) = cfn(p, \theta) + X$$

*Translation invariance* is evident by noticing that for all $m$,

$$(\lambda l : \theta(l) + X)_{(m)} = \theta_{(m)} + X$$

and

$$\left\lfloor \frac{(\theta_{(F+1)} + X) + (\theta_{(N-F)} + X)}{2} \right\rfloor = \left\lfloor \frac{\theta_{(F+1)} + \theta_{(N-F)}}{2} \right\rfloor + X$$

## 4.2 Precision Enhancement

As mentioned in Chapter 2, *precision enhancement* is a formalization of the concept that, after executing the convergence function, the values of interest should be close together. The proofs do not depend on $p$ and $q$ being in $C$; therefore, the precondition was removed for the following weakened restatement of precision enhancement:

**Condition 3 (precision enhancement)** *Given any subset $C$ of the $N$ clocks with $|C| \geq N - F$, then for any readings $\gamma$ and $\theta$ 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 $C$, $|\theta(l) - \theta(m)| \leq Y$*

*there is a bound $\pi(X, Y)$ such that*

$$|cfn(p, \gamma) - cfn(q, \theta)| \leq \pi(X, Y)$$

**Theorem 4.1** *Precision enhancement is satisfied for* $cfn_{MID}(p, \vartheta)$ *if*

$$\pi(X, Y) = \left\lceil \frac{Y}{2} + X \right\rceil$$

One characteristic of $cfn_{MID}(p, \vartheta)$ is that it is possible for it to use readings from faulty clocks. If this occurs, we know that such readings are bounded by readings from good clocks. The next few lemmas establish this fact. To prove these lemmas, it was expedient to develop a pigeonhole principle.

**Lemma 4.1.1 (Pigeonhole Principle)** *If* $N$ *is the number of clocks in the system and* $C_1$ *and* $C_2$ *are subsets of these* $N$ *clocks,*

$$|C_1| + |C_2| \geq N + k \supset |C_1 \cap C_2| \geq k$$

This principle greatly simplifies the existence proofs required to establish the next two lemmas. First, we establish that the values used in computing the convergence function are bounded by readings from good clocks.

**Lemma 4.1.2** *Given any subset* $C$ *of the* $N$ *clocks with* $|C| \geq N - F$ *and any reading* $\theta$*, there exist* $p, q \in C$ *such that*

$$\theta(p) \geq \theta_{(F+1)} \text{ and } \theta_{(N-F)} \geq \theta(q)$$

**Proof:** By definition, $|\{p : \theta(p) \geq \theta_{(F+1)}\}| \geq F + 1$ (similarly, $|\{q : \theta_{(N-F)} \geq \theta(q)\}| \geq F + 1$). The conclusion follows immediately from the pigeonhole principle. ∎

Now we introduce a lemma that allows us to relate values from two different readings to the same good clock.

**Lemma 4.1.3** *Given any subset* $C$ *of the* $N$ *clocks with* $|C| \geq N - F$ *and readings* $\theta$ *and* $\gamma$*, there exist* $a$*,* $p \in C$ *such that*

$$\theta(p) \geq \theta_{(N-F)} \text{ and } \gamma_{(F+1)} \geq \gamma(p)$$

**Proof:** With $N \geq 3F + 1$, we can apply the pigeonhole principle twice: first, to establish that $|\{p : \theta(p) \geq \theta_{(N-F)}\} \cap C| \geq F + 1$ and second, to establish the conclusion. ∎

An immediate consequence of the preceding lemma is that the readings used in computing $cfn_{MID}(p, \theta)$ bound a reading from a good clock.

The next lemma introduces a useful fact for bounding the difference between good clock values from different readings.

**Lemma 4.1.4** *Given any subset* $C$ *of the* $N$ *clocks and clock readings* $\theta$ *and* $\gamma$ *such that for any* $l$ *in* $C$*, the bound* $|\theta(l) - \gamma(l)| \leq X$ *holds, for all* $p, q \in C$*,*

$$\theta(p) \geq \theta(q) \wedge \gamma(q) \geq \gamma(p) \supset |\theta(p) - \gamma(q)| \leq X$$

**Proof:** By cases,

1. If $\theta(p) \geq \gamma(q)$, then $|\theta(p) - \gamma(q)| \leq |\theta(p) - \gamma(p)| \leq X$
2. If $\theta(p) \leq \gamma(q)$, then $|\theta(p) - \gamma(q)| \leq |\theta(q) - \gamma(q)| \leq X$

■

From this lemma, we can establish the following lemma:

**Lemma 4.1.5** *Given any subset $C$ of the $N$ clocks and clock readings $\theta$ and $\gamma$ such that for any $l$ in $C$, the bound $|\theta(l) - \gamma(l)| \leq X$ holds, there exist $p, q \in C$ such that*

$$\theta(p) \geq \theta_{(F+1)}$$

$$\gamma(q) \geq \gamma_{(F+1)}$$

$$|\theta(p) - \gamma(q)| \leq X$$

**Proof:** We know from lemma 4.1.2 that there are $p_1, q_1 \in C$ that satisfy the first two conjuncts of the conclusion. Three cases to consider are

1. If $\gamma(p_1) > \gamma(q_1)$, let $p = q = p_1$
2. If $\theta(q_1) > \theta(p_1)$, let $p = q = q_1$
3. Otherwise, we have satisfied the hypotheses for lemma 4.1.4; therefore, we let $p = p_1$ and $q = q_1$

■

We are now able to establish precision enhancement for $cfn_{MID}(p, \vartheta)$ (theorem 4.1).

**Proof:** Without loss of generality, assume $cfn_{MID}(p, \gamma) \geq cfn_{MID}(q, \theta)$:

$$\begin{aligned} & |cfn_{MID}(p, \gamma) - cfn_{MID}(q, \theta)| \\ = & \left| \left\lfloor \frac{\gamma_{(F+1)} + \gamma_{(N-F)}}{2} \right\rfloor - \left\lfloor \frac{\theta_{(F+1)} + \theta_{(N-F)}}{2} \right\rfloor \right| \\ \leq & \left\lceil \frac{\left| \gamma_{(F+1)} + \gamma_{(N-F)} - (\theta_{(F+1)} + \theta_{(N-F)}) \right|}{2} \right\rceil \end{aligned}$$

Thus we need to show that

$$|\gamma_{(F+1)} + \gamma_{(N-F)} - (\theta_{(F+1)} + \theta_{(N-F)})| \leq Y + 2X$$

By choosing good clocks $p, q$ from lemma 4.1.5, $p_1$ from lemma 4.1.3, and $q_1$ from the right conjunct of lemma 4.1.2, we establish

$$\begin{aligned} & |\gamma_{(F+1)} + \gamma_{(N-F)} - (\theta_{(F+1)} + \theta_{(N-F)})| \\ \leq & \ |\gamma(q) + \gamma(p_1) - \theta(p_1) - \theta(q_1)| \\ = & \ |\gamma(q) + (\theta(p) - \theta(p)) + \gamma(p_1) - \theta(p_1) - \theta(q_1)| \\ \leq & \ |\theta(p) - \theta(q_1)| + |\gamma(q) - \theta(p)| + |\gamma(p_1) - \theta(p_1)| \\ \leq & \ Y + 2X \end{aligned}$$

(by hypotheses and lemma 4.1.5).

■

## 4.3 Accuracy Preservation

Recall that *accuracy preservation* formalizes the notion that there should be a bound on the amount of correction applied in any synchronization interval. The proof here uses the weak form of accuracy preservation. The bound holds even if $p$ is not in $C$.

**Condition 4 (accuracy preservation)** *Given any subset $C$ of the $N$ clocks with $|C| \geq N - F$ and clock readings $\theta$ 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 $C$,*

$$|cfn(p, \theta) - \theta(q)| \leq \alpha(X)$$

**Theorem 4.2** *Accuracy preservation is satisfied for $cfn_{MID}(p, \theta)$ if $\alpha(X) = X$.*

**Proof:** Begin by selecting $p_1$ and $q_1$ using lemma 4.1.2. Clearly, $\theta(p_1) \geq cfn_{MID}(p, \theta)$ and $cfn_{MID}(p, \theta) \geq \theta(q_1)$. Two cases to consider are

1. If $\theta(q) \leq cfn_{MID}(p, \theta)$, then $|cfn_{MID}(p, \theta) - \theta(q)| \leq |\theta(p_1) - \theta(q)| \leq X$
2. If $\theta(q) \geq cfn_{MID}(p, \theta)$, then $|cfn_{MID}(p, \theta) - \theta(q)| \leq |\theta(q_1) - \theta(q)| \leq X$ ∎

## 4.4 EHDM Proofs of Convergence Properties

This section presents the important details of the EHDM proofs that $cfn_{MID}(p, \theta)$ satisfies the convergence properties. In general, the proofs closely follow the presentation given previously. The EHDM modules used in this effort are given in appendix C. Supporting proofs, including the EHDM proof of the pigeonhole principle, are given in appendix D.

One underlying assumption for these proofs is that $N \geq 3F + 1$, which is a well-known requirement for systems to achieve Byzantine fault tolerance without requiring authentication (ref. 3). The statement of this assumption is axiom No_authentication in module ft_mid_assume. As an experiment, this assumption was weakened to $N \geq 2F + 1$. The only proof corrupted was that of lemma good_between in module mid3. This corresponds to lemma 4.1.3. Lemma 4.1.3 is central to the proof of precision enhancement. It establishes that for any pair of nonfaulty clocks, there is at least one reading from the same good clock in the range of the readings selected for computation of the convergence function. This prevents a scenario in which two or more clusters of good clocks continue to drift apart because the values used in the convergence function for any two good clocks are guaranteed to overlap.

Another assumption added for this effort states that the array of clock readings can be sorted. Additionally, a few properties one would expect to be true of a sorted array were assumed. These additional properties used in the EHDM proofs are (from module clocksort)

funsort_ax: **Axiom**
$i \leq j \wedge j \leq N \supset \vartheta(\mathsf{funsort}(\vartheta)(i)) \geq \vartheta(\mathsf{funsort}(\vartheta)(j))$

funsort_trans_inv: **Axiom**
$k \leq N \supset (\vartheta(\mathsf{funsort}(\lambda\, q : \vartheta(q) + X)(k)) = \vartheta(\mathsf{funsort}(\vartheta)(k)))$

cnt_sort_geq: **Axiom**
$k \leq N \supset \mathsf{count}((\lambda\, p : \vartheta(p) \geq \vartheta(\mathsf{funsort}(\vartheta)(k))), N) \geq k$

cnt_sort_leq: **Axiom**
$k \leq N \supset \mathsf{count}((\lambda\, p : \vartheta(\mathsf{funsort}(\vartheta)(k)) \geq \vartheta(p)), N) \geq N - k + 1$

Appendix C contains the proof chain analysis for the three properties. The proof for translation invariance is in module mid, precision enhancement is in mid3, and accuracy preservation is in mid4.

A number of lemmas were added to (and proven in) module countmod. The most important of these is the aforementioned pigeonhole principle. In addition, lemma count_complement was moved from Shankar's module ica3 to countmod. Shankar's complete proof was rerun after the changes to ensure that nothing was inadvertently destroyed. Basic manipulations involving the integer floor and ceiling functions are presented in module floor_ceil. In addition, the weakened versions of *accuracy preservation* and *translation invariance* were added to module clockassumptions. The restatements are axioms accuracy_preservation_recovery_ax and precision_enhancement_recovery_ax, respectively. The revised formulations imply the original formulation, but are more flexible for reasoning about recovery from transient faults because they do not require that the process evaluating the convergence function be part of the collection of *working* clocks. The proofs that $cfn_{MID}(p, \theta)$ satisfies these properties were performed with respect to the revised formulation. The original formulation of the convergence function properties is retained in the theory because not all convergence functions satisfy the weakened formulas.

Chapter 5 presents a hardware design of a clock synchronization system that uses the fault-tolerant midpoint convergence function. The design is shown to satisfy the remaining constraints of the theory.

# Chapter 5

# Design of Clock Synchronization System

This chapter describes a design of a fault-tolerant clock synchronization circuit that satisfies the constraints of the theory. This design assumes that the network of clocks is completely connected. Section 5.1 presents an informal description of the design, and section 5.2 demonstrates that the design meets requirements 2 through 6 from section 3.4.

## 5.1 Description of Design

As in other synchronization algorithms, this one consists of an infinite sequence of synchronization intervals $i$ for each clock $p$; each interval is of duration $R + ADJ_p^i$. All good clocks are assumed to maintain an index of the current interval (a simple counter is sufficient, provided that all *good* channels start the counter in the same interval). Furthermore, the assumption is made that the network of clocks contains a sufficient number of nonfaulty clocks and that the system is already synchronized. In other words, the design described in this chapter preserves the synchronization of the redundant clocks. The issue of achieving initial synchronization is addressed in Chapter 6. The major concern is when to begin the next interval; this consists of both determining the amount of the adjustment and when to apply it. For this, we require readings of the other clocks in the system and a suitable convergence function. As stated in Chapter 4, the selected convergence function is the fault-tolerant midpoint.

In order to evaluate the convergence function to determine the $(i+1)$th interval clock, clock $p$ needs an estimate of the other clocks when local time is $T_p^{i+1}$. All clocks participating in the protocol know to send a synchronization signal when they are $Q$ ticks into the current interval;[6] for example, when $LC_p^i(t) = Q$, where $LC$ is a counter measuring elapsed time since the beginning of the current interval. Our estimate, $\Theta_p^{i+1}$, of other clocks is

$$\Theta_p^{i+1}(q) = T_p^{i+1} + (Q - LC_p^i(t_{pq}))$$

[6]This is actually a simplification for the purpose of presentation. Clock $p$ sends its signal so that it will be received at the remote clock when $LC_p^i(t) = Q$.

where $t_{pq}$ is the time when $p$ recognizes the signal from $q$. The value $Q - LC_p^i(t_{pq})$ gives the difference between when the local clock $p$ expected the signal and when it observed a signal from $q$. The reading is taken in such a way that simply adding the value to the current local clock time gives an estimate of the other clock's reading at that instant. It is not important that $Q$ be near the end of the interval. For this system, we assume the drift rate $\rho$ of a good clock is less than $10^{-5}$; this value corresponds to the drift rate of commercially available oscillators. By selecting $R$ to be $\leq 10^4$ ticks (a synchronization interval of 1 msec for a 10-MHz clock), the maximum added error of $2\rho R \leq 0.2$ caused by clock drift does not appreciably alter the quality of our estimate of a remote clock's value. In this system, $p$ always receives a signal from itself when $LC_p^i(t) = Q$; therefore, no error is made in reading its own clock.

Chapter 3 presents two options for determining when to apply the adjustment. This design employs the second option, namely that

$$T_p^{i+1} = (i+1)R + T^0 - ADJ_p^i$$

Recalling that $t_p^{i+1} = ic_p^i(T_p^{i+1}) = ic_p^{i+1}(T_p^{i+1} + ADJ_p^i)$ makes it easy to determine from the algebraic clock definitions given in section 2.1 and the above expression, that

$$cfn_{MID}(p, \Theta_p^{i+1}) = IC_p^{i+1}(t_p^{i+1}) = (i+1)R + T^0$$

Since $T^0 = 0$ in this design, we just need to ensure that $cfn_{MID}(p, \Theta_p^{i+1}) = (i+1)R$. Using translation invariance and this definition for $\Theta_p^{i+1}$ gives

$$cfn_{MID}(p, (\lambda q.\Theta_p^{i+1}(q) - T_p^{i+1})) = (i+1)R - T_p^{i+1} = ADJ_p^i$$

Since $\Theta_p^{i+1}(q) - T_p^{i+1} = (Q - LC_p^i(t_{pq}))$, we have

$$ADJ_p^i = cfn_{MID}(p, (\lambda q(Q - LC_p^i(t_{pq}))))$$

In Chapter 4, the fault-tolerant midpoint convergence function was defined as follows:

$$cfn_{MID}(p, \theta) = \left\lfloor \frac{\theta_{(F+1)} + \theta_{(N-F)}}{2} \right\rfloor$$

If we are able to select the $(N-F)$th and $(F+1)$th readings, computing this function in hardware consists of a simple addition followed by an arithmetic shift right.[7] All that remains is to determine the appropriate readings to use. By assumption, there are a sufficient number $(N-F)$ of nonfaulty synchronized clocks participating in the protocol. Therefore, we know that we will observe at least $N-F$ pulses during the synchronization interval. Since $Q$ is fixed and $LC$ does not decrease during the interval, the readings $(\lambda qQ - LC_p^i(t_{pq}))$ are sorted into decreasing order by arrival time. Suppose $t_{pq}$ is when the $(F+1)$th pulse is recognized, then $Q - LC_p^i(t_{pq})$ must be the $(F+1)$th largest reading. A similar argument applies to the $(N-F)$th pulse arrival. A pulse counter gives us the

[7] An arithmetic shift right of a two's complement value preserves the sign bit and truncates the least significant bit.

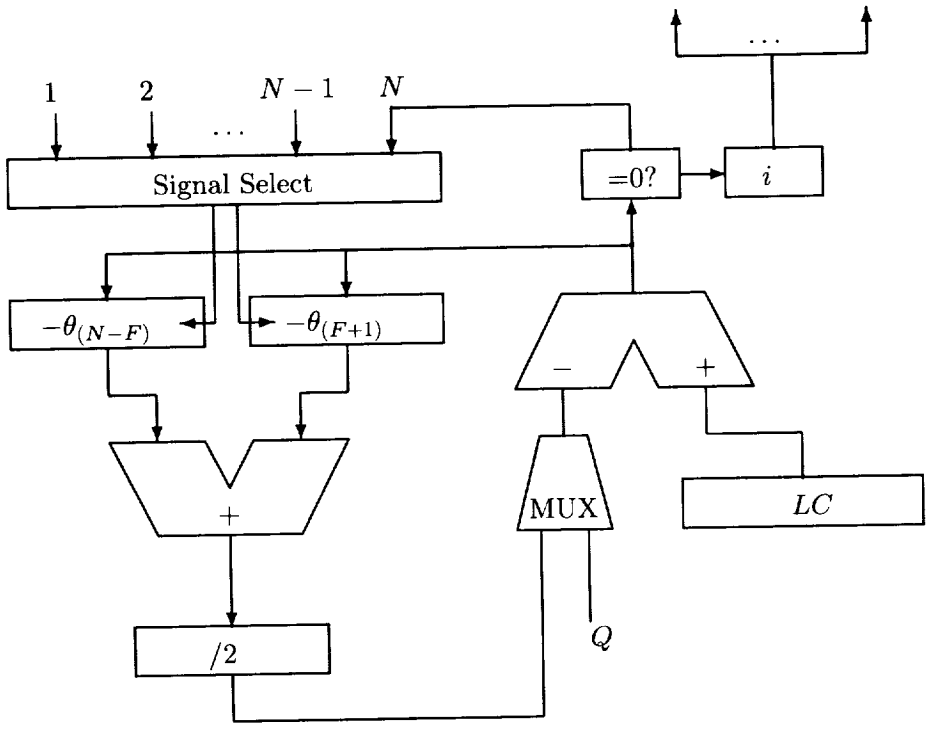



Figure 5.1: Informal block model of clock synchronization circuit.

necessary information to select appropriate readings for the convergence function. Once $N - F$ pulses have been observed, both the magnitude and time of adjustment can be determined. At this point, the circuit just waits until $LC_p^i(t) = R + ADJ_p^i$ to begin the next interval.

Figure 5.1 presents an informal block model of the clock synchronization circuit. The circuit consists of the following components:[8]

$N$ pulse recognizers (only one pulse per clock is recognized in any given interval)
Pulse counter (triggers events based on pulse arrivals)
Local counter $LC$ (measures elapsed time since beginning of current interval)
Interval counter (contains the index $i$ of the current interval)
One adder for computing the value $-(Q - LC_p^i(t_{pq}))$
One register each for storing $-\theta_{(F+1)}$ and $-\theta_{(N-F)}$
Adder for computing the sum of these two registers
A divide-by-2 component (arithmetic shift right)

The pulses are already sorted by arrival time, therefore, using a pulse counter is natural to select the time stamp of the $(F + 1)$th and the $(N - F)$th pulses for the computation

[8] In order to simplify the design, the circuit computes $-ADJ_p^i$ and then subtracts this value when applying the adjustment. Thus the readings captured are $-\theta$ rather than $\theta$.

of the convergence function. As stated previously, all that is required is the difference between the local and remote clocks. Let

$$\theta = (\lambda q.\Theta_p^{i+1}(q) - T_p^{i+1})$$

When the $(F+1)$th $(N-F)$th signal is observed, register $-\theta_{(F+1)}$ $(-\theta_{(N-F)})$ is clocked, saving the value $-(Q-LC_p^i(t))$. After $N-F$ signals have been observed, the multiplexor selects the computed convergence function instead of $Q$. When $LC_p^i(t)-(-cfn_{MID}(p,(\theta))) = R$, it is time to begin the $(i+1)$th interval. To do this, all that is required is to increment $i$ and reset $LC$ to 0. The pulse recognizers, multiplexor select, and registers are also reset at this time.

## 5.2 Theory Obligations

The requirements referred to in this section are from the list presented in section 3.4. Since this design was developed, in part, from the algebraic definitions given in section 2.1, it is relatively easy to see that it meets the necessary definitions as specified by requirement 5. The interval clock is defined as follows:

$$IC_p^i(t) = iR + LC_p^i(t)$$

From the description of the design given, we know that

$$IC_p^{i+1}(t) = IC_p^i(t) + ADJ_p^i$$

with $LC_p^0(t)$ corresponding to $PC_p(t)$ as described in Chapter 2. The only distinction is that, in the implementation, $LC$ is repeatedly reset. Even so, it is the primary mechanism for marking the passage of time. Clearly, this implementation of $IC$ ensures that this design provides a correct $VC$. The time reference provided to the local processing elements is the pair $(i,\ LC_p^i(t))$ with the expected interpretation that the current elapsed time since the beginning of the protocol is $iR + LC_p^i(t)$.

This circuit cycles through the following states:

1. From $LC_p^i(t) = 0$ until the $(N-F)$th pulse is received, it determines the readings needed for the convergence function
2. It uses the readings to compute the adjustment $ADJ_p^i$
3. When $LC_p^i(t)+ADJ_p^i = R$, it applies the correction by resetting for the next interval

In parallel with this sequence of states, when $LC_p^i(t) = Q$, it transmits its synchronization signal to the other clocks in the system. This algorithm is clearly an instance of the general algorithm schema presented as requirement 6 (section 3.4). State 1, in conjunction with the transmission of the synchronization signal, implements the exchange of clock values. State 2 determines both the adjustment for this interval and the time of application. State 3 applies the correction at the appropriate time.

Requirement 2 demands a demonstration that the mechanism for exchanging clock values introduces at most a small error to the readings of a remote clock. The best that can be achieved in practice for the first clause of condition 6 is for $\Lambda$ to equal 1 tick. The third clause, however, includes real-time separation and a possible value for $\Lambda'$ of approximately 0.5 tick. We assume these values for the remainder of this paper. A hardware realization of the above abstract design with estimates of reading error equivalent to these is presented in reference 24. These bounds have not been established formally. Preliminary research, which may enable formal derivation of such bounds, can be found in reference 25.

With these values for reading error, we can now solve the inequalities presented at the end of Chapter 2. The inequalities used for this presentation are those from the informal proof of theorem 2.1 given in appendix A. These inequalities are

1. $4\rho r_{max} + \pi(\lfloor 2\Lambda' + 2 \rfloor, \lfloor \beta' + 2\Lambda' \rfloor) \leq \beta'$
2. $\lceil (1+\rho)\beta' + 2\rho r_{max} \rceil \leq \delta$
3. $\alpha(\lfloor \beta' + 2\Lambda' \rfloor) + \Lambda + \lceil 2\rho\beta \rceil + 1 \leq \delta$

For the first inequality, we need to find the smallest value of $\beta'$ that satisfies the inequality. The bound $\beta'$ can be represented as the sum of an integer and a real between 0 and 1. Let the integer part be $B$ and the real part be $b$. We know that $\rho R \leq 0.1$ and that $r_{max}$ is not significantly more than $R$. Therefore, we can let $b = 4\rho r_{max} \approx 0.4$ and reduce the inequality to the following form:

$$\pi(\lfloor 2\Lambda' + 2 \rfloor, \lfloor \beta' + 2\Lambda' \rfloor) \leq B$$

The estimate for $\Lambda'$ is $\approx 0.5 < 1 - b/2$, therefore with $\lfloor 2\Lambda' + 2 \rfloor = 3$ and $\lfloor \beta' + 2\Lambda' \rfloor = B+1$. Using the $\pi$ established for $cfn_{MID}(p, \theta)$ in Chapter 4 gives

$$3 + \left\lceil \frac{B+1}{2} \right\rceil \leq B$$

The smallest value of $B$ that satisfies this inequality is 7, therefore, the above circuit can maintain a value of $\beta'$ that is $\approx 7.4$ ticks. By using this value in the second inequality, we see that $\delta \geq 8$. Because $\alpha$ is the identity function for $cfn_{MID}(p, \theta)$ and $\Lambda = 1$, we get $\delta \geq 11$ ticks from the third inequality. The bound from the third inequality does not seem tight, but it is the best proven result we have. By using these numbers with a clock rate of 10 MHz, this circuit will synchronize the redundant clocks to within about 1 $\mu$sec. Since the frame length for most flight control systems is on the order of 50 msec, this circuit provides tight synchronization with negligible overhead.

All that remains in this chapter is to show that this design satisfies requirement 4. This consists of satisfying conditions 8 and 9. We know that $\alpha(\beta' + 2\Lambda') < 9$ and that $T^0 = 0$. We can satisfy condition 8 by selecting $S^0$ such that $9 \leq S^0 \leq R - 9$. Since $R \approx 10^4$, this should be no problem. For simplicity, let $S^0 = Q$. Also, since $R \gg (1+\rho)\beta + \alpha(\beta' + 2\Lambda')$, condition 9 is easily met. Requirement 7, achieving initial synchronization, is addressed in the next chapter.

# Chapter 6

# Initialization and Transient Recovery

This chapter establishes that the design presented in Chapter 5 meets the one remaining requirement of the list given in section 3.4. This requirement is to satisfy condition 7, bounded delay initialization. Establishing this requirement in the absence of faults is sufficient because initialization is only required at system startup. A fault encountered at startup is not critical and can be remedied by repairing the failed component. However, a guaranteed automatic mechanism that establishes initial synchronization would provide a mechanism for recovery from correlated transient failures. Therefore, the arguments given for initial synchronization attempt to address behavior in the presence of faults also. These arguments are still in an early stage of development and are therefore presented informally unlike the proofs in earlier chapters.

Section 6.2 addresses guaranteed recovery from a bounded number of transient faults. The EHDM theory presented in section 3.3 presents sufficient conditions to establish theorem 3.1 while recovering from transient faults. Section 6.2 restates these conditions and adds a few more that may be necessary to mechanically prove theorem 2.1 and still allow transient recovery. Section 6.2 also demonstrates that the design presented in Chapter 5 meets the requirements of these transient recovery conditions.

A number of clock synchronization protocols include mechanisms to achieve initialization and transient recovery. An implicit assumption in all these approaches is a diagnosis mechanism that triggers the initialization or recovery action. One goal of this design is that these functions happen automatically by virtue of the normal operation of the synchronization algorithm. It appears that the fault-tolerant midpoint cannot be modified to ensure automatic initialization. However, with slight modification, the fault-tolerant midpoint algorithm allows for automatic recovery from transient faults without a diagnostic action.

## 6.1 Initial Synchronization

If we can get into a state that satisfies the requirements for *precision enhancement* (condition 3, repeated here for easy reference):

> **Condition 3 (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 $\gamma$ and $\theta$ satisfying the conditions*
>
> 1. *For any $l$ in $C$, $|\gamma(\ell) - \theta(\ell)| \leq X$*
> 2. *For any $l$, $m$ in $C$, $|\gamma(\ell) - \gamma(m)| \leq Y$*
> 3. *For any $l$, $m$ in $C$, $|\theta(\ell) - \theta(m)| \leq Y$*
>
> *there is a bound $\pi(X, Y)$ such that*
>
> $$|cfn(p, \gamma) - cfn(q, \theta)| \leq \pi(X, Y)$$

where $Y \leq \lfloor \beta_{\text{read}} + 2\Lambda' \rfloor$ and $X = \lfloor 2\Lambda' + 2 \rfloor$[9], then a synchronization system using the design presented in Chapter 5 will converge to the point where $|s_p^0 - s_q^0| \leq \beta'$ in approximately $\log_2(Y)$ intervals. Byzantine agreement is then required to establish a consistent interval counter. (For the purposes of this discussion, it is assumed that a verified mechanism for achieving Byzantine agreement exists. Examples of such mechanisms can be found in refs. 26 and 27.) The clocks must reach a state satisfying the above constraints. Clearly, we would like $\beta_{\text{read}}$ to be as large as possible. To be conservative, we set $\beta_{\text{read}} = (\min(Q, R - Q) - \alpha(\lfloor \beta' + 2\Lambda' \rfloor))/(1 + \rho)$. Figure 6.1 illustrates the relevant phases in a synchronization interval. If the clocks all transmit their synchronization pulses within $\beta_{\text{read}}$ of each other, the clock readings will satisfy the constraints listed above. By letting $Q = R/2$, we get the largest possible symmetric window for observing the other clocks. However, more appropriate settings for $Q$ may exist.

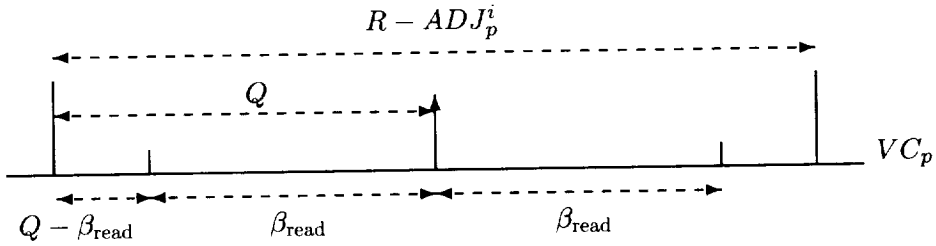



Figure 6.1: Key parts of synchronization interval.

[9]This condition is satisfied when for $p, q \in C$, $|s_p^i - s_q^i| \leq \beta_{\text{read}}$. During initialization, $i = 0$.

### 6.1.1 Mechanisms for Initialization

In order to ensure that we reach a state that satisfies these requirements, it is necessary to identify possible states that violate these requirements. Such states would happen because of the behavior of clocks prior to the time that enough good clocks are running. In previous cases, we knew we had a set $C$ of good clocks with $|C| \geq N - F$. This means a sufficient number of clock readings were available to resolve $\theta_{(F+1)}$ and $\theta_{(N-F)}$. This may not be true during initialization. We need to determine a course of action when we do not observe $N - F$ clocks. Two plausible options are as follows:

**Assumed perfection** — pretend all clocks are observed to be in perfect synchrony

**End of interval** — pretend that unobserved clocks are observed at the end of the synchronization interval; i.e., $LC_p^i(t_{pq}) - Q = R - Q$; compute the correction based on this value

The first option is simple to implement because no correction is necessary. When $LC = R$, set both $i$ and $LC$ to 0, and reset the circuit for the next interval. To implement the second option, perform the following action when $LC = R$: if fewer than $N - F$ $(F+1)$ signals are observed, then enable register $-\theta_{(N-F)}(-\theta_{(F+1)})$. This causes the unobserved readings to be $(R-Q)$ which is equivalent to observing the pulse at the end of an interval of duration $R$.

We discuss these two possibilities with respect to a four-clock system. The arguments for the general case are similar, but are combinatorially more complicated. We only consider cases in which at least one pair of clocks is separated by more than $\beta_{\text{read}}$. Otherwise, the conditions enumerated would be satisfied.

#### 6.1.1.1 Assumed Perfection

For assumed perfection, all operational clocks transmit their pulse within $(1 + \rho)R/2$ of every other operational clock. We present one scenario consisting of four nonfaulty clocks to demonstrate that this approach does not work. At least one pair of clocks is separated by more than $\beta_{\text{read}}$. A real implementation needs a certain amount of time to reset for the next interval; therefore, there is a short period of time $z$ at the end of an interval where signals will be missed. This enables a pathological case that can prevent a clock from participating in the protocol, even if no faults are present. If two clocks are separated by $(R - Q) - z$, only one of the two clocks is able to read the other. If additional clocks that are synchronous with the hidden clock are added, they too will be hidden. Figure 6.2 illustrates a four-clock system caught in this pathological scenario. The scale is exaggerated to clearly depict the window $z$ in which signals from other clocks cannot be observed. Typically, this window is quite small with respect to the length of the synchronization interval. In this figure, clock $a$ never sees the other clocks in the system, and therefore remains unsynchronized, even though it is not faulty. There are a number of options for remedying this deficiency, but all result in more difficult arguments for demonstrating recovery from transient faults. The presence of this window of invisibility is unfortunate, because it invalidates a simple probabilistic proof that this approach guarantees initial synchronization. Although the illustration shows $Q = R/2$, a similar pathological scenario exists for any setting of $Q$.

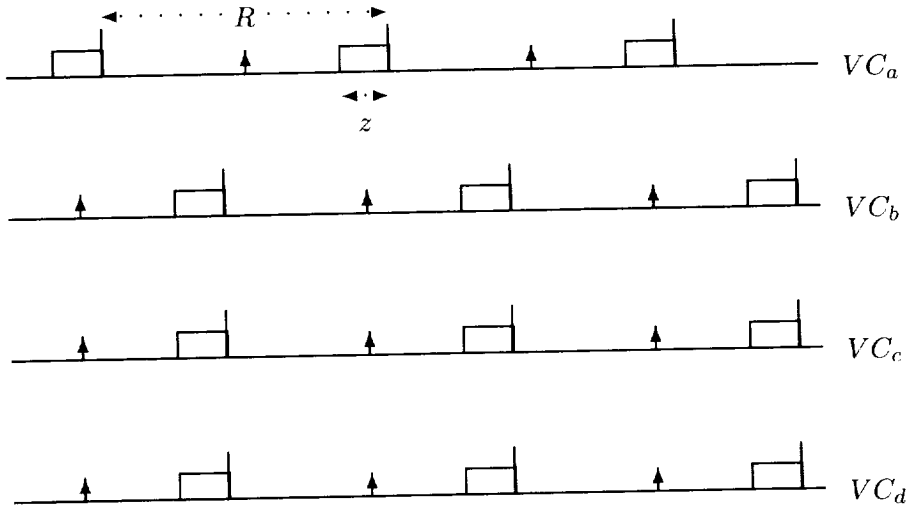


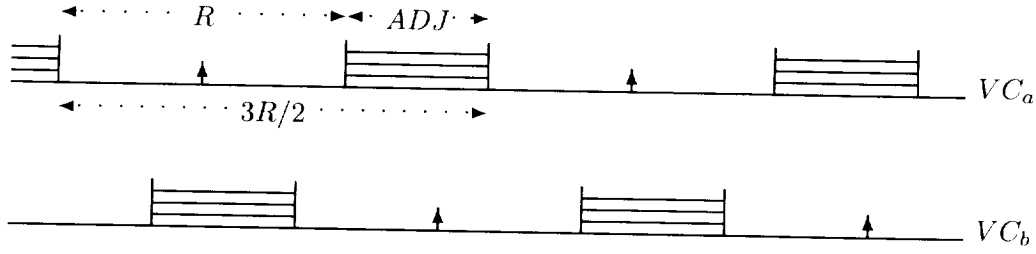

Figure 6.2: Pathological scenario—assumed perfection.

#### 6.1.1.2 End of Interval

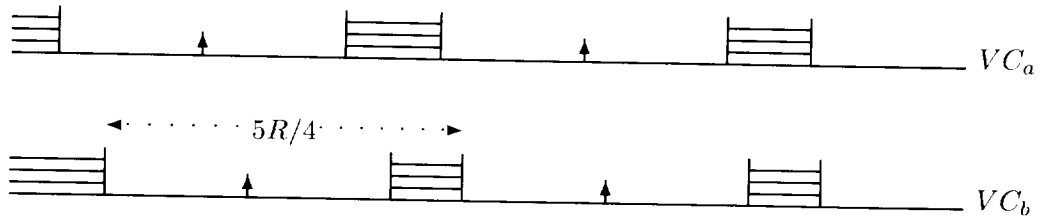
The end of interval approach is an attempt to avoid the pathological case illustrated in figure 6.2. We begin by considering the cases where only two clocks are actively participating. Assume for the sake of this discussion that $Q = R/2$ (to maximize $\beta_{\text{read}}$). There are two possibilities—the synchronization pulses are either separated by more than $R/2$ or less than $R/2$. The two cases are illustrated in figure 6.3. In case 1, each clock computes the maximum adjustment of $R/2$ and transmits a pulse every $3R/2$ ticks. In case 2, $VC_b$ computes an adjustment of $R/4$ and transmits a pulse every $5R/4$ ticks, whereas $VC_a$ computes an adjustment between $R/4$ and $R/2$ and converges to a point where it transmits a pulse every $5R/4$ ticks and is synchronized with $VC_b$. If we add a third clock to case 1, it must be within $R/2$ of at least one of the two clocks. If it is within $R/2$ of both, it will pull the two clocks together quickly. Otherwise, the pair within $R/2$ of each other will act as if they are the only two clocks in the system and will converge to each other in the manner of case 2. Since two clocks have an interval length of $5R/4$, and the third has an interval length of $3R/2$, the three clocks will shortly reach a point where they are within $\beta_{\text{read}}$ of each other. This argument also covers the case where we add a third clock to case 2. Once the three nonfaulty clocks are synchronized, we can add a fourth clock and use the transient recovery arguments presented in section 6.2 to ensure that it joins the ensemble of clocks. This provides us with a sound mechanism to ensure initial synchronization in the absence of failed clocks; we just power the clocks in order with enough elapsed time between clocks to ensure that they have stabilized. This mechanism is sufficient to satisfy the initialization requirement but does not address reinitialization due to the occurrence of correlated transient failures.

Unfortunately, if we begin with four clocks participating in the initialization scheme, a pathological scenario arises. This scenario is illustrated in figure 6.4. Clocks $VC_a$ and

$VC_b$ are synchronized with each other in the manner of case 2 of figure 6.3; likewise, $VC_c$ and $VC_d$ are synchronized. The two pairs are not synchronized with each other. This illustrates that even with no faulty clocks, the system may converge to a 2-2 split: two pairs synchronized with each other but not with the other pair. Once again, values for $Q$ other than $R/2$ were explored; in each case a 2-2 split was discovered. The next section proposes a means to avoid this pathological case, while preserving the existing means for achieving initial synchronization and transient recovery.




Case 1: $|s_a - s_b| \geq R/2$




Case 2: $|s_a - s_b| < R/2$

Figure 6.3: End of interval initialization.

#### 6.1.1.3 End of Interval—Time-Out

Inspection of figure 6.4 suggests that if any of the clocks were to arbitrarily decide not to compute any adjustment, the immediately following interval would have a collection of three clocks within $\beta_{\text{read}}$ of each other, as shown in figure 6.5. When clock $b$ decides not to compute any adjustment, it shifts to a point where its pulse is within $\beta_{\text{read}}$ of $c$ and $d$. Here the algorithm takes over, and the three values converge. Figure 6.5 illustrates the fault-free case. If $a$ were faulty, it could delay convergence by at most $\log_2(\beta_{\text{read}})$. Clock $a$ is also brought into the fold because of the transient recovery process. This process is explained in more detail in section 6.2. All that remains is to provide a means for the clocks not to apply any adjustment when such action is necessary.

Suppose each clock maintains a count of the number of elapsed intervals since it has observed $N - F$ pulses. When this count reaches 8, for example, it is reasonably safe

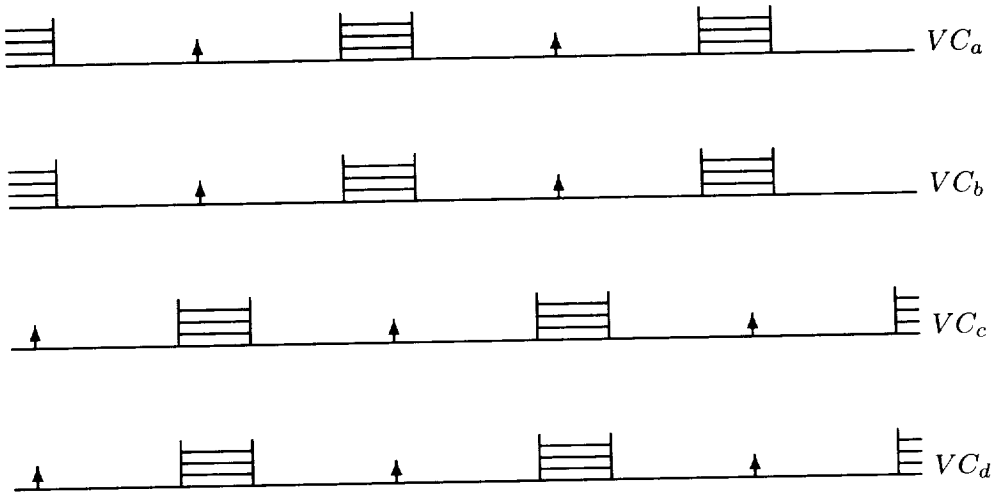



Figure 6.4: Pathological end of interval initialization.

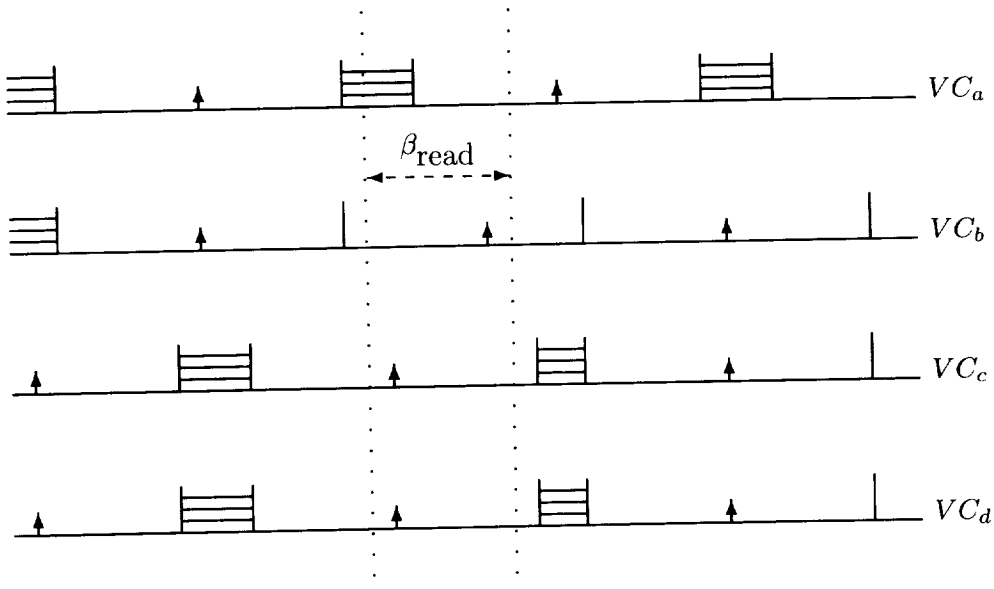



Figure 6.5: End of interval initialization—time-out.

to assume that either fewer than $N - F$ clocks are active or the system is caught in the pathological scenario illustrated in figure 6.4. In either case, choosing to apply no correction for one interval does no harm. Once this time-out expires, it is important to reset the counter and switch back immediately to the end of interval mode. This prevents the system from falling into the pathological situation presented in figure 6.2.

Now that we have a consistent mechanism for automatically initializing a collection of good clocks, we need to explore how a faulty clock could affect this procedure. First we note that figure 6.4 shows the only possible pathological scenario. Consider that an ensemble of unsynchronized clocks must have at least one pair separated by more than $\beta_{\text{read}}$, otherwise the properties of precision enhancement force the system to synchronize. In a collection of three clocks, at least one pair must be within $\beta_{\text{read}}$; figure 6.3 shows that in the absence of other readings, a pair within $\beta_{\text{read}}$ will synchronize to each other. The only way a fourth clock can be added to prevent system convergence is the pathological case in figure 6.4. If this fourth clock is fault free, the time-out mechanism will ensure convergence. Two questions remain: can a faulty clock prevent the time-out from expiring, and can a faulty clock prevent synchronization if a time-out occurs. We address the former first.

Recall from the description of the design that, in any synchronization interval, each clock recognizes at most one signal from any other clock in the system. The only means to prevent a time-out is for each nonfaulty clock to observe three pulses in an interval, at least once every eight intervals. In figure 6.6, $d$ is faulty in such a manner that it will be observed by $a$, $b$, and $c$ without significantly altering their computed corrections. This fault is considered benign because $d$ is regularly transmitting a synchronization pulse that is visible to all the other clocks in the system. Clock $d$ is considered faulty because it is not correctly responding to the signals that it observes. Clock $c$ is not visible to either $a$ or $b$, and neither of these is visible to $c$. Neither $a$ nor $b$ will reach a time-out, because they see three signals in every interval. However, except for very rare circumstances, $c$ will eventually execute a time-out, and the procedure illustrated in figure 6.5 will cause $a$, $b$, and $c$ to synchronize.

There is one unlikely scenario when $Q = R/2$ in which the good clocks fail to converge. It requires $c$ to observe $a$ at the end of its interval, with neither $a$ nor $b$ observing $c$. Only one of the symmetric cases is presented here. This is only possible if $c$ and $a$ are separated by precisely $R/2$ ticks. Even then, $a$ will more likely see $c$ than the other way around. This tendency can be exaggerated by setting $Q$ to be slightly more than $R/2$, ensuring that $a$ will see $c$ first. If $a$ observes $c$, the effect will be the same as if it had a time-out. Since $a$ is synchronized with $b$, observing $c$ at the beginning of the interval will cause the proper correction to be 0, and the system will synchronize.

The only remaining question is whether a faulty clock can prevent the others from converging if a time-out occurs. Unfortunately, a fault can exhibit sufficiently malicious behavior to prevent initialization. We begin by looking back at figure 6.5. If $a$ is faulty, and a time-out occurs for $b$, then $b$, $c$, and $d$ will synchronize. If, on the other hand, $d$ is faulty, we do not get a collection of good clocks within $\beta_{\text{read}}$. A possible scenario is

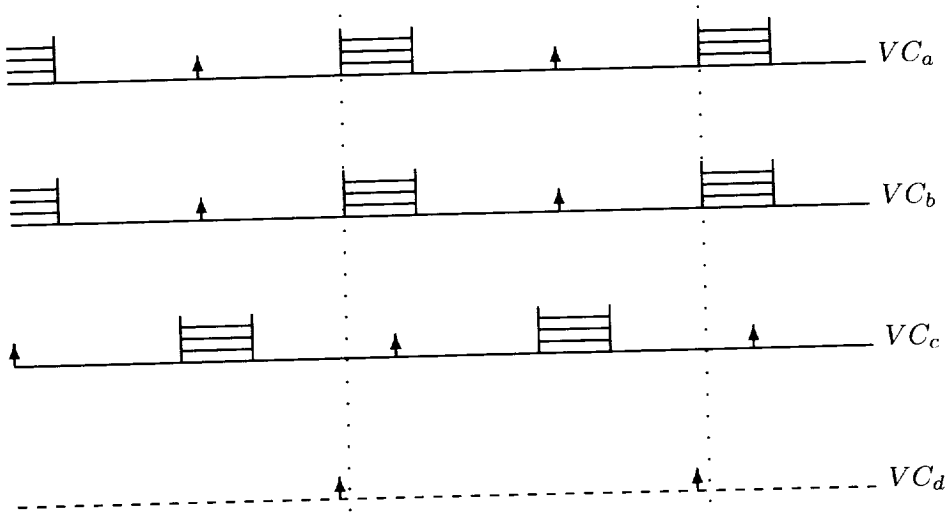



Figure 6.6: End of interval initialization: $d$ faulty—benign.

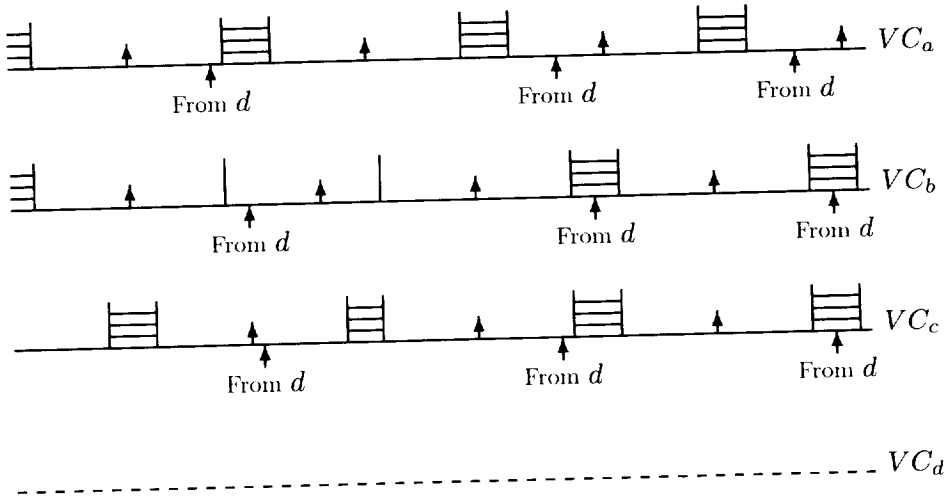



Figure 6.7: End of interval initialization: $d$ faulty—malicious.

shown in figure 6.7, where $d$ prevents $a$ from synchronizing and also causes the time-out for $a$ to reset. At some point, $d$ also sends a pulse at the end of an interval to either $b$ or $c$ to ensure that just one of them has a time-out. The process can then be repeated, preventing the collection of good clocks from ever becoming synchronized. This fault is malicious because the behavior of $d$ appears different to each of the other clocks in the system.

The attempt for a fully automatic initialization scheme has fallen short. A sound mechanism exists for initializing the clocks in the absence of any failures. Also, if a clock fails passive, the remaining clocks will be able to synchronize. Unfortunately, the technique is not robust enough to ensure initialization in the presence of malicious failures.

### 6.1.2 Comparison With Other Approaches

The argument that the clocks converge within $\log_2(\beta_{\text{read}})$ intervals is adapted from that given by Welch and Lynch (ref. 2). However, the approach given here for achieving initial synchronization differs from most methods in that first the interval clocks are synchronized, and then an index is decided on for the current interval. Techniques in references 2, 4, and 6 all depend on the good clocks knowing that they wish to initialize. Agreement is reached among the clocks wishing to join, and then the initialization protocol begins. It seems that this standard approach is necessary to ensure initialization in the presence of malicious faults. The approach taken here is similar to that mentioned in reference 20; however, details of that approach are not given.

## 6.2 Transient Recovery

The argument for transient recovery capabilities hinges on the following observation:

> *As long as there is power to the circuit and no faults are present, the circuit will execute the algorithm.*

With the fact that the algorithm executes continually and that pulses can be observed during the entire synchronization interval, we can establish that up to $F$ transiently affected channels will automatically reintegrate themselves into the set of good channels.

### 6.2.1 Theory Considerations

A number of axioms were added to the EHDM clock synchronization theory to provide sufficient conditions to establish transient recovery. Current theory provides an uninstantiated predicate rpred that must imply certain properties. To formally establish transient recovery, it is sufficient to identify an appropriate rpred for the given design and then show that a clock will eventually satisfy rpred if affected by a transient fault (provided that enough clocks were unaffected). The task is considerably simplified if the convergence function satisfies the *recovery* variants of precision enhancement and accuracy preservation. In Chapter 4, it was shown that the fault-tolerant midpoint function satisfies those conditions. The current requirements for rpred are the following:

1. From module delay3 –
   recovery_lemma: **Axiom**
   $\text{delay\_pred}(i) \wedge \text{ADJ\_pred}(i+1)$
   $\wedge\ \text{rpred}(i)(p) \wedge \text{correct\_during}(p, t_p^{i+1}, t_p^{i+2}) \wedge \text{wpred}(i+1)(q)$
   $\supset |s_p^{i+1} - s_q^{i+1}| \leq \beta'$

2. From module new_basics –
   delay_recovery: **Axiom**
   $\text{rpred}(i)(p) \wedge \text{wvr\_pred}(i)(q) \supset |t_p^{i+1} - t_q^{i+1}| \leq \beta$

3. From module rmax_rmin—
   ADJ_recovery: **Axiom** option1 $\wedge$ rpred$(i)(p) \supset |ADJ_p^i| \leq \boldsymbol{\alpha}(\lfloor \beta' + 2 * \Lambda' \rfloor)$

4. From module delay—
   wpred_preceding: **Axiom** wpred$(i+1)(p) \supset$ wpred$(i)(p) \vee$ rpred$(i)(p)$

   wpred_rpred_disjoint: **Axiom** $\neg$(wpred$(i)(p) \wedge$ rpred$(i)(p)$)

   wpred_bridge: **Axiom**
   wvr_pred$(i)(p) \wedge$ correct_during$(p, t_p^{i+1}, t_p^{i+2}) \supset$ wpred$(i+1)(p)$

The conditions from module delay define wpred; they ensure that a clock is considered working only if it was working or recovered in the previous interval. They were previously discussed in section 3.3. Arguments for transient recovery hinge on the first three constraints presented. In Chapter 3, two options were presented for determining when to apply the adjustment. These options are

1. $T_p^{i+1} = (i+1)R + T^0$
2. $T_p^{i+1} = (i+1)R + T^0 - ADJ_p^i$

Since the design presented in Chapter 5 uses the second option, the arguments for transient recovery are specific to that case. The argument for this option depends primarily on satisfying axiom recovery_lemma.

Axiom recovery_lemma is used in the inductive step of the machine-checked proof of theorem 3.1. To prove recovery_lemma, it is sufficient for rpred$(i)(p)$ to equal the following:

correct_during$(p, s_p^i, t_p^{i+1})$
wpred$(i)(q) \supset |s_p^i - s_q^i| \leq \beta_{\text{read}}$ and
$\neg$wpred$(i)(p)$

Using arguments similar to the proof of theorem 3.1, we can then establish that

$$|ADJ_p^i| \leq \boldsymbol{\alpha}(\beta_{\text{read}} + 2\Lambda')$$

$$|ic_p^{i+1}(T) - ic_q^{i+1}(T)| \leq 2\rho(|T - S^i| + \boldsymbol{\alpha}(\beta_{\text{read}} + 2\Lambda')) + \boldsymbol{\pi}(2\Lambda' + 2, \beta' + 2\Lambda')$$

The second of these is made possible by using the recovery version of precision enhancement. Since $\beta' \geq 4\rho r_{max} + \boldsymbol{\pi}(2\Lambda' + 2, \beta' + 2\Lambda')$, all that remains is to establish that $2\rho(|S^{i+1} - S^i| + \boldsymbol{\alpha}(\beta_{\text{read}} + 2\Lambda')) \leq 4\rho r_{max}$. Since $\beta_{\text{read}} < R/2$ and $\boldsymbol{\alpha}$ is the identity function, this relation is easily established. Axiom delay_recovery is easily established for implementations by using the second algorithm schema presented in Chapter 3. Because $T_p^{i+1} + ADJ_p^i = (i+1)R + T^0$ and $t_p^{i+1} = ic_p^{i+1}((i+1)R + T^0)$, all that is required is to substitute $(i+1)R + T^0$ for $T$ in item 2. Since the two options are mutually exclusive and the design employs the second, axiom ADJ_recovery is trivially satisfied.

### 6.2.2 Satisfying rpred

The only modification required to the design is that the synchronization signals include the sender's value for $i$ (the index for the current synchronization interval). By virtue of the maintenance algorithm, the $N - F$ good clocks are synchronized within a bounded skew $\delta \ll R$. A simple majority vote restores the index of the recovering clock. If the recovering clock's pulse is within $\beta_{\text{read}}$ of the collection of good clocks, rpred is satisfied. If not, we need to ensure that a recovering clock will always shift to a point where it is within $\beta_{\text{read}}$ of the collection of good clocks.

The argument for satisfying rpred is given for a four-clock system; the argument for the general case requires an additional time-out mechanism to avoid pathological cases. Consider the first full synchronization interval in which the recovering clock is not faulty. In a window of duration $R$, it will obtain readings of the good clocks in the system. If the three readings are within $\delta$ of each other, the recovering clock will use two of the three readings to compute the convergence function, restore the index via a majority vote, and will be completely recovered for the next interval. It is possible, however, that the pulses from the good clocks align closely with the edge of the synchronization interval. The recovering clock may see one or two clocks in the beginning of the interval and read the rest at the end. It is important to be using the end of interval method for resolving the absence of pulses. By using the end of interval method, it is guaranteed that some adjustment will be computed in every interval. If two pulses are observed near the beginning of the interval, the current interval will be shortened by no more than $R-Q$. If only one clock is observed in the beginning of the interval, then either two clocks will be observed at the end of the interval or the circuit will pretend they were observed. In either case, the interval will be lengthened by $(R-Q)/2$. It is guaranteed that in the next interval the recovering clock will be separated from the good clocks by $\approx (R-Q)/2$. Since $(R-Q)/2 < \beta_{\text{read}}$, the requirements of rpred have been satisfied. It is important to recognize that this argument does not depend on the particular value chosen for $Q$. This gives greater flexibility for manipulating the design to meet other desired properties.

### 6.2.3 Comparison With Other Approaches

A number of other fault-tolerant clock synchronization protocols allow for restoration of a lost clock. The approach taken here is very similar to the one proposed by Welch and Lynch (ref. 2). They propose that when a process awakens, it observes incoming messages until it can determine which round is underway and then waits sufficiently long to ensure that it has seen all valid messages in that round. It then computes the necessary correction to become synchronized. Srikanth and Toueg (ref. 6) use a similar approach modified to the context of their algorithm. Halpern et al. (ref. 4) suggest a rather complicated protocol which requires explicit cooperation of other clocks in the system. All these approaches have the common theme, namely, that the joining clock knows that it wants to join. This implies the presence of some diagnostic logic or time-out mechanism that triggers the recovery process. The approach suggested here happens automatically. By virtue of the algorithm's execution in dedicated hardware, there is no need to awaken a process to participate in the protocol. The main idea is for the recovering process to converge to a state where it will observe all other clocks in the same interval and then restore the correct interval counter.

# Chapter 7

# Concluding Remarks

Clock synchronization provides the cornerstone of many fault-tolerant computer architectures. To avoid a single point failure it is imperative that each processor maintain a local clock that is periodically resynchronized with other clocks in a fault-tolerant manner. Reasoning about fault-tolerant clock synchronization is complicated by the potential for subtle interactions involving failed components. For critical applications, it is necessary to prove that this function is implemented correctly. Shankar (NASA CR-4386) provides a mechanical proof (using EHDM) that Schneider's generalized protocol (Tech. Rep. 87-859, Cornell Univ.) achieves Byzantine fault-tolerant clock synchronization if 11 constraints are satisfied. This general proof is quite useful because it simplifies the verification of fault-tolerant clock synchronization systems. The difficult part of the proof is reusable; all that is required for a verified system is to show that the implementation satisfies the underlying assumptions of the general theory. This paper has revised the proof to simplify the verification conditions and illustrated the revised theory with a concrete example.

Both Schneider and Shankar assumed the property of bounded delay. (This terminology is from Shankar's report; Schneider called this property a reliable time source.) This property asserts that there is a bound on the elapsed time between synchronization actions of any two good clocks. For many protocols, it is easy to prove synchronization once bounded delay has been established. For these protocols, the difficult part of the proof has been left to the verifier. This paper presents a general proof of bounded delay from suitably modified versions of the remaining conditions. This revised set of conditions greatly simplifies the use of Schneider's theory in the verification of clock synchronization systems. In addition, a set of conditions sufficient for proving recovery from transient faults has been added to the theory. A design of a synchronization system, based on the fault-tolerant midpoint convergence function, was shown to satisfy the constraints of the revised theory.

One of the goals of the design was to develop a synchronization system that could automatically initialize itself, even in the presence of faults. Two approaches for a four-clock system were explored and shown to possess pathological scenarios that prevent reliable initialization. An informal sketch of a third approach was given that combines techniques from the two failed attempts. This technique ensures automatic initialization in the absence of failures or when the failures are benign. However, malicious behavior from a

failed clock can prevent good clocks from synchronizing. The standard approach of first reaching agreement and then synchronizing seems necessary for guaranteed initialization in the presence of arbitrary failures.

In keeping with the design philosophy of the Reliable Computing Platform (RCP), the clock synchronization system was designed to recover from transient faults. Sufficient conditions for transient recovery were embedded in the EHDM proofs. These conditions were based on the approach used by DiVito, Butler, and Caldwell for the RCP (NASA TM-102716). It was shown that a four-clock instance of the given design will satisfy the transient recovery assumptions. Furthermore, the recovery happens automatically; there is no need to diagnose occurrence of a transient fault.

In summary, a mechanically checked version of Schneider's paradigm for fault-tolerant clock synchronization was extended both to simplify verification conditions and to allow for proven recovery from transient faults. Use of the extended theory was illustrated with the verification of an abstract design of a fault-tolerant clock synchronization system. Some of the requirements of the theory were established via a mechanically checked formal proof using EHDM, whereas other constraints were demonstrated informally. Ultimately, a mechanically checked argument should be developed for all the constraints to help clarify the underlying assumptions and, in many cases, to correct errors in the informal proofs. Mechanical proof is still a difficult task because it is not always clear how to best present arguments to the mechanical proof system. For example, the arguments given for initial synchronization need to be revised considerably before a mechanically checked proof is possible. Nevertheless, even though some conditions were not proven mechanically, development of the design from the mechanically checked specification has yielded better understanding of the system than has been possible otherwise.

NASA Langley Research Center
Hampton, VA 23681-0001
July 19, 1993

# Appendix A

# Proof of Agreement

This appendix consists of two parts: The first part consists of an informal proof sketch that agreement can be established by using the revised constraints on $\delta$ and some of the intermediate results of Chapter 3 are presented. The second part consists of information extracted from EHDM that confirms that the mechanical proofs of agreement have been performed for the minor revisions to Shankar's theory. There are also revised versions of modules clockassumptions and lemma_final; lemma_final contains the EHDM statement of theorem 2.1, lemma agreement.

## A.1 Proof Sketch of Agreement

This section sketches the highlights of an informal proof that the following constraints are sufficient to establish theorem 2.1; these arguments have not yet been submitted to EHDM:

1. $4\rho r_{max} + \pi(\lfloor 2\Lambda' + 2 \rfloor \lfloor \beta' + 2\Lambda' \rfloor) \leq \beta'$
2. $\lceil (1+\rho)\beta' + 2\rho r_{max} \rceil \leq \delta$
3. $\alpha(\lfloor \beta' + 2\Lambda' \rfloor) + \Lambda + \lceil 2\rho\beta \rceil + 1 \leq \delta$

The first of these constraints is established in Chapter 3 and is used to ensure that $|s_p^i - s_q^i| \leq \beta'$. We can use an intermediate result of that proof (lemma 3.1.2) to establish the second of these constraints. The third constraint is obtained by substituting the revised bounds on the array of clock readings (established in the proof of part (a) of theorem 3.1) into Shankar's proof. This has not been done in the mechanical proof because Shankar's proof has not yet been revised to accommodate transient recovery.

We now prove the following theorem (from Chapter 2):

**Theorem 2.1 (bounded skew)** *For any two clocks $p$ and $q$ that are nonfaulty at time $t$,*

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

To do this, we first need the following two lemmas:

**Lemma 2.1.1** *For nonfaulty clocks p and q, and* $\max(t_p^i, t_q^i) \leq t < \min(t_p^{i+1}, t_q^{i+1})$,

$$|IC_p^i(t) - IC_q^i(t)| \leq \lceil (1+\rho)\beta' + 2\rho r_{max} \rceil$$

**Proof:** We begin by noticing that $IC_p^i(t) = IC_p^i(ic_p^i(IC_p^i(t)))$ (and similarly for $IC_q$). Assume without loss of generality that $ic_p^i(IC_p^i(t)) \leq ic_q^i(IC_q^i(t)) \leq t$, and let $T = IC_q^i(t)$. Clearly, $T \leq \max(T_p^{i+1}, T_q^{i+1})$. We now have

$$\begin{aligned} |IC_p^i(t) - IC_q^i(t)| &= |IC_p^i(ic_q^i(T)) - IC_q^i(ic_q^i(T))| \\ &= |IC_p^i(ic_q^i(T)) - IC_p^i(ic_p^i(T))| \\ &\leq \lceil (1+\rho)(|ic_q^i(T) - ic_p^i(T)|) \rceil \end{aligned}$$

The final step in the above derivation is established by corollary 5.1.

All that remains is to establish that $|ic_q^i(T) - ic_p^i(T)| \leq \beta' + 2\rho r_{max}/(1+\rho)$. Earlier, we defined $r_{max}$ to be $(1+\rho)(R + \boldsymbol{\alpha}(\beta' + 2\Lambda'))$. The proof is by induction on $i$. For $i = 0$,

$$\begin{aligned} |ic_q^i(T) - ic_p^i(T)| &\leq |t_q^0 - t_p^0| + 2\rho(\max(T_p^{i+1}, T_q^{i+1}) - T^0) \\ &\leq \beta' + 2\rho(R + \boldsymbol{\alpha}(\beta' + 2\Lambda')) \end{aligned}$$

For the inductive step, we use lemma 3.1.2 to establish that

$$|ic_q^{i+1}(T) - ic_p^{i+1}(T)| \leq 2\rho(|T - S^i| + \boldsymbol{\alpha}(\beta' + 2\Lambda')) + \boldsymbol{\pi}(2\Lambda' + 2, \beta' + 2\Lambda')$$

There are two cases to consider: if $T \leq S^{i+1}$, this is clearly less than $\beta'$; if $T > S^{i+1}$, this is bounded by $\beta' + 2\rho(\max(T_p^{i+1}, T_q^{i+1}) - S^{i+1})$. It is simple to establish that $(\max(T_p^{i+1}, T_q^{i+1}) - S^{i+1}) \leq (R + \boldsymbol{\alpha}(\beta' + 2\Lambda'))$. ∎

**Lemma 2.1.2** *For nonfaulty clocks p and q and* $t_q^{i+1} \leq t < t_p^{i+1}$,

$$|IC_p^i(t) - IC_q^{i+1}(t)| \leq \boldsymbol{\alpha}(\lfloor \beta' + 2\Lambda' \rfloor) + \Lambda + \lceil 2\rho\beta \rceil + 1$$

**Proof Sketch:** The proof follows closely the argument given in the proof of case 2 of theorem 2.3.2 in reference 10. The proof is in two parts. First, the difference at $t_q^{i+1}$ is bounded with accuracy preservation, and then the remainder of the interval is bounded. The difference in this presentation is that here the argument to $\boldsymbol{\alpha}$ is smaller. ∎

We can now prove theorem 2.1.

**Proof Sketch:** The proof consists of recognizing that $VC_p(t) = IC_p^i(t)$ for $t_p^i \leq t < t_p^{i+1}$. This, coupled with nonoverlap and the above two lemmas, assures the result. ∎

## A.2 EHDM Extracts

### A.2.1 Proof Chain Analysis

The following is an extract of the EHDM proof chain analysis for lemma agreement in module lemma_final.

```
⋮

   ================== SUMMARY ==================

The proof chain is complete

The axioms and assumptions at the base are:
  clockassumptions.IClock_defn
  clockassumptions.Readerror
  clockassumptions.VClock_defn
  clockassumptions.accuracy_preservation_recovery_ax
  clockassumptions.beta_0
  clockassumptions.correct_closed
  clockassumptions.correct_count
  clockassumptions.init
  clockassumptions.mu_0
  clockassumptions.precision_enhancement_recovery_ax
  clockassumptions.rate_1
  clockassumptions.rate_2
  clockassumptions.rho_0
  clockassumptions.rho_1
  clockassumptions.rmax_0
  clockassumptions.rmin_0
  clockassumptions.rts0
  clockassumptions.rts1
  clockassumptions.rts2
  clockassumptions.rts_2
  clockassumptions.synctime_0
  clockassumptions.translation_invariance
  division.mult_div_1
  division.mult_div_2
  division.mult_div_3
  floor_ceil.ceil_defn
  floor_ceil.floor_defn
  multiplication.mult_10
  multiplication.mult_non_neg
  noetherian[EXPR, EXPR].general_induction
 Total: 30
```

```
The definitions and type-constraints are:
  absmod.abs
  basics.maxsync
  basics.maxsynctime
  basics.minsync
  clockassumptions.Adj
  clockassumptions.okay_Reading
  clockassumptions.okay_Readpred
  clockassumptions.okay_Readvars
  clockassumptions.okay_pairs
  lemma3.okayClocks
  multiplication.mult
  readbounds.okaymaxsync
Total: 12

⋮
```

### A.2.2 Module lemma_final

lemma_final: **Module**

**Using** clockassumptions, lemma3, arith, basics

**Exporting all with** clockassumptions, lemma3

**Theory**

$p, q, p_1, p_2, q_1, q_2, p_3, q_3, i, j, k$: **Var** nat
$l, m, n$: **Var** int
$x, y, z$: **Var** number
posnumber: **Type from** number **with** $(\lambda\, x : x \geq 0)$
$r, s, t$: **Var** posnumber

correct_synctime: **Lemma** $\mathsf{correct}(p,t) \wedge t < t_p^i + r_{min} \supset t < t_p^{i+1}$

synctime_multiples: **Lemma** $\mathsf{correct}(p,t) \wedge t \geq 0 \wedge t < i \star r_{min} \supset t_p^i > t$

synctime_multiples_bnd: **Lemma** $\mathsf{correct}(p,t) \wedge t \geq 0 \supset t < t_p^{\lceil t/r_{min} \rceil + 1}$

agreement: **Lemma** $\beta \leq r_{min}$
$\wedge\, \mu \leq \delta_S \wedge \boldsymbol{\pi}(\lceil 2 * \Lambda + 2 * \beta \star \rho \rceil + 1,$
$\delta_S + \lceil 2 * ((r_{max} + \beta) \star \rho + \Lambda) \rceil + 1)$
$\leq \delta_S$
$\wedge\, \delta_S + \lceil 2 * r_{max} \star \rho \rceil + 1 \leq \delta$
$\wedge\, \boldsymbol{\alpha}(\delta_S + \lceil 2 * (r_{max} + \beta) \star \rho + 2 * \Lambda \rceil + 1) + \Lambda + \lceil 2 * \beta \star \rho \rceil + 1$
$\leq \delta$
$\wedge\, t \geq 0 \wedge \mathsf{correct}(p,t) \wedge \mathsf{correct}(q,t)$
$\supset |VC_p(t) - VC_q(t)| \leq \delta$

**Proof**

agreement_proof: **Prove** agreement **from**
lemma3_3 $\{i \leftarrow \lceil t/r_{min} \rceil + 1\}$,
okayClocks_defn_lr $\{i \leftarrow \lceil t/r_{min} \rceil + 1,\ t \leftarrow t@CS\}$,
maxsync_correct $\{s \leftarrow t,\ i \leftarrow \lceil t/r_{min} \rceil + 1\}$,
synctime_multiples_bnd $\{p \leftarrow (p \Uparrow q)[\lceil t/r_{min} \rceil + 1]\}$,
rmin_0,
div_nonnegative $\{x \leftarrow t,\ y \leftarrow r_{min}\}$,
ceil_defn $\{x \leftarrow (t/r_{min})\}$

synctime_multiples_bnd_proof: **Prove** synctime_multiples_bnd **from**
ceil_plus_mult_div $\{x \leftarrow t,\ y \leftarrow r_{min}\}$,
synctime_multiples $\{i \leftarrow \lceil t/r_{min} \rceil + 1\}$,
rmin_0,
div_nonnegative $\{x \leftarrow t,\ y \leftarrow r_{min}\}$,
ceil_defn $\{x \leftarrow (t/r_{min})\}$

correct_synctime_proof: **Prove** correct_synctime **from** rts1 $\{t \leftarrow t@CS\}$

synctime_multiples_pred: function[nat, nat, posnumber $\rightarrow$ bool] ==
$(\lambda\, i, p, t$ : correct$(p, t) \wedge t \geq 0 \wedge t < i * r_{min} \supset t_p^i > t)$

synctime_multiples_step: **Lemma**
correct$(p, t) \wedge t \geq t_p^i \wedge t \geq 0 \supset t_p^i \geq i \star r_{min}$

synctime_multiples_proof: **Prove** synctime_multiples **from**
synctime_multiples_step

synctime_multiples_step_pred: function[nat, nat, posnumber $\rightarrow$ bool] ==
$(\lambda\, i, p, t$ : correct$(p, t) \wedge t_p^i \leq t \wedge t \geq 0 \supset t_p^i \geq i \star r_{min})$

synctime_multiples_step_proof: **Prove** synctime_multiples_step **from**
induction {prop $\leftarrow (\lambda\, i$ : synctime_multiples_step_pred$(i, p, t))$},
mult_l0 $\{x \leftarrow r_{min}\}$,
synctime_0,
rts_1 $\{i \leftarrow j@P1\}$,
rmin_0,
correct_closed $\{s \leftarrow t,\ t \leftarrow t_p^{j@P1+1}\}$,
distrib $\{x \leftarrow j@P1,\ y \leftarrow 1,\ z \leftarrow r_{min}\}$,
mult_lident $\{x \leftarrow r_{min}\}$

**End** lemma_final

### A.2.3 Module clockassumptions

clockassumptions: **Module**

**Using** arith, countmod

**Exporting all with** countmod, arith

**Theory**

$N$: nat

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

process: **Type is** nat
event: **Type is** nat
time: **Type is** number
Clocktime: **Type is** integer
$l, m, n, p, q, p_1, p_2, q_1, q_2, p_3, q_3$: **Var** process
$i, j, k$: **Var** event
$x, y, z, r, s, t$: **Var** time
$X, Y, Z, R, S, T$: **Var** Clocktime
$\gamma, \theta$: **Var** function[process → Clocktime]
$\delta, \rho, r_{min}, r_{max}, \beta$: number
$\Lambda, \mu$: Clocktime
$PC_{\star 1}(\star 2), VC_{\star 1}(\star 2)$: function[process, time → Clocktime]
$t^{\star 2}_{\star 1}$: function[process, event → time]
$\Theta^{\star 2}_{\star 1}$: function[process, event → function[process → Clocktime]]
$IC^{\star 2}_{\star 1}(\star 3)$: function[process, event, time → Clocktime]
correct: function[process, time → bool]
$cfn$: function[process, function[process → Clocktime] → Clocktime]
$\pi$: function[Clocktime, Clocktime → Clocktime]
$\alpha$: function[Clocktime → Clocktime]

delta_0: **Axiom** $\delta \geq 0$

mu_0: **Axiom** $\mu \geq 0$

rho_0: **Axiom** $\rho \geq 0$

rho_1: **Axiom** $\rho < 1$

rmin_0: **Axiom** $r_{min} > 0$

rmax_0: **Axiom** $r_{max} > 0$

beta_0: **Axiom** $\beta \geq 0$

lamb_0: **Axiom** $\Lambda \geq 0$

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

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

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

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

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

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

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

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

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

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

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

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

$adj_{\star 1}^{\star 2}$: function[process, event $\rightarrow$ Clocktime] =
$(\lambda\, p, i : (\textbf{ if } i > 0 \textbf{ then } cfn(p, \Theta_p^i) - PC_p(t_p^i) \textbf{ else } 0 \textbf{ end if}))$

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

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

translation_invariance: **Axiom**
$cfn(p, (\lambda\, p_1 \rightarrow \mathsf{Clocktime} : \gamma(p_1) + X)) = cfn(p,\gamma) + X$

ppred: **Var** function[process $\rightarrow$ bool]
$F$: process
okay_Readpred: function[function[process $\rightarrow$ Clocktime], number,
function[process $\rightarrow$ bool] $\rightarrow$ bool] =
$(\lambda\, \gamma, y, \mathsf{ppred} : (\forall\, l, m : \mathsf{ppred}(l) \wedge \mathsf{ppred}(m) \supset |\gamma(l) - \gamma(m)| \leq y))$
okay_pairs: function[function[process $\rightarrow$ Clocktime],
function[process $\rightarrow$ Clocktime], number,
function[process $\rightarrow$ bool] $\rightarrow$ bool] =
$(\lambda\, \gamma, \theta, x, \mathsf{ppred} : (\forall\, p_3 : \mathsf{ppred}(p_3) \supset |\gamma(p_3) - \theta(p_3)| \leq x))$

okay_Readpred_floor: **Lemma**
$\mathsf{okay\_Readpred}(\gamma, y, \mathsf{ppred}) \supset \mathsf{okay\_Readpred}(\gamma, \lfloor y \rfloor, \mathsf{ppred})$

okay_pairs_floor: **Lemma**
$\mathsf{okay\_pairs}(\gamma, \theta, x, \mathsf{ppred}) \supset \mathsf{okay\_pairs}(\gamma, \theta, \lfloor x \rfloor, \mathsf{ppred})$

N_maxfaults: **Axiom** $F < N$

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

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

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

okay_Reading: $\mathsf{function}[\mathsf{function}[\mathsf{process} \to \mathsf{Clocktime}], \mathsf{number}, \mathsf{time} \to \mathsf{bool}] =$
$(\lambda\, \gamma, y, t : (\forall\, p_1, q_1 :$
$\mathsf{correct}(p_1, t) \wedge \mathsf{correct}(q_1, t) \supset |\gamma(p_1) - \gamma(q_1)| \leq y))$

okay_Readvars: $\mathsf{function}[\mathsf{function}[\mathsf{process} \to \mathsf{Clocktime}], \mathsf{function}[\mathsf{process} \to \mathsf{Clocktime}], \mathsf{number}, \mathsf{time} \to \mathsf{bool}] =$
$(\lambda\, \gamma, \theta, x, t : (\forall\, p_3 : \mathsf{correct}(p_3, t) \supset |\gamma(p_3) - \theta(p_3)| \leq x))$

okay_Readpred_Reading: **Lemma**
$\mathsf{okay\_Reading}(\gamma, y, t) \supset \mathsf{okay\_Readpred}(\gamma, y, (\lambda\, p : \mathsf{correct}(p, t)))$

okay_pairs_Readvars: **Lemma**
$\mathsf{okay\_Readvars}(\gamma, \theta, x, t) \supset \mathsf{okay\_pairs}(\gamma, \theta, x, (\lambda\, p : \mathsf{correct}(p, t)))$

precision_enhancement: **Lemma**
$\mathsf{okay\_Reading}(\gamma, Y, t_p^{i+1})$
$\wedge\ \mathsf{okay\_Reading}(\theta, Y, t_p^{i+1})$
$\wedge\ \mathsf{okay\_Readvars}(\gamma, \theta, X, t_p^{i+1})$
$\wedge\ \mathsf{correct}(p, t_p^{i+1}) \wedge \mathsf{correct}(q, t_p^{i+1})$
$\supset |cfn(p, \gamma) - cfn(q, \theta)| \leq \pi(X, Y)$

okay_Reading_defn_lr: **Lemma**
$\mathsf{okay\_Reading}(\gamma, y, t)$
$\supset (\forall\, p_1, q_1 : \mathsf{correct}(p_1, t) \wedge \mathsf{correct}(q_1, t) \supset |\gamma(p_1) - \gamma(q_1)| \leq y)$

okay_Reading_defn_rl: **Lemma**
$(\forall\, p_1, q_1 : \mathsf{correct}(p_1, t) \wedge \mathsf{correct}(q_1, t) \supset |\gamma(p_1) - \gamma(q_1)| \leq y)$
$\supset \mathsf{okay\_Reading}(\gamma, y, t)$

okay_Readvars_defn_lr: **Lemma**
$\mathsf{okay\_Readvars}(\gamma, \theta, x, t) \supset (\forall\, p_3 : \mathsf{correct}(p_3, t) \supset |\gamma(p_3) - \theta(p_3)| \leq x)$

okay_Readvars_defn_rl: **Lemma**
$(\forall\, p_3 : \mathsf{correct}(p_3, t) \supset |\gamma(p_3) - \theta(p_3)| \leq x) \supset \mathsf{okay\_Readvars}(\gamma, \theta, x, t)$

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

accuracy_preservation_recovery_ax: **Axiom**
$\mathsf{okay\_Readpred}(\gamma, X, \mathsf{ppred}) \wedge \mathsf{count}(\mathsf{ppred}, N) \geq N - F \wedge \mathsf{ppred}(q)$
$\supset |cfn(p, \gamma) - \gamma(q)| \leq \boldsymbol{\alpha}(X)$

## Proof

okay_Readpred_floor_pr: **Prove** okay_Readpred_floor **from**
okay_Readpred $\{l \leftarrow l@p2,\ m \leftarrow m@p2\}$,
okay_Readpred $\{y \leftarrow \lfloor y \rfloor\}$,
iabs_is_abs $\{X \leftarrow \gamma(l@p2) - \gamma(m@p2),\ x \leftarrow \gamma(l@p2) - \gamma(m@p2)\}$,
floor_mon $\{x \leftarrow \mathsf{iabs}(X@p3)\}$,
floor_int $\{i \leftarrow \mathsf{iabs}(X@p3)\}$

okay_pairs_floor_pr: **Prove** okay_pairs_floor **from**
okay_pairs $\{p_3 \leftarrow p_3@\mathsf{p2}\}$,
okay_pairs $\{x \leftarrow \lfloor x \rfloor\}$,
iabs_is_abs $\{x \leftarrow \gamma(p_3@\mathsf{p2}) - \theta(p_3@\mathsf{p2}),\ X \leftarrow \gamma(p_3@\mathsf{p2}) - \theta(p_3@\mathsf{p2})\}$,
floor_mon $\{x \leftarrow \mathsf{iabs}(X@p3),\ y \leftarrow x\}$,
floor_int $\{i \leftarrow \mathsf{iabs}(X@p3)\}$

precision_enhancement_ax_pr: **Prove** precision_enhancement_ax **from**
precision_enhancement_recovery_ax

accuracy_preservation_ax_pr: **Prove** accuracy_preservation_ax **from**
accuracy_preservation_recovery_ax

okay_Reading_defn_rl_pr: **Prove**
okay_Reading_defn_rl $\{p_1 \leftarrow p_1@\mathsf{P1S},\ q_1 \leftarrow q_1@\mathsf{P1S}\}$ **from** okay_Reading

okay_Reading_defn_lr_pr: **Prove** okay_Reading_defn_lr **from**
okay_Reading $\{p_1 \leftarrow p_1@\mathsf{CS},\ q_1 \leftarrow q_1@\mathsf{CS}\}$

okay_Readvars_defn_rl_pr: **Prove** okay_Readvars_defn_rl $\{p_3 \leftarrow p_3@\mathsf{P1S}\}$ **from**
okay_Readvars

okay_Readvars_defn_lr_pr: **Prove** okay_Readvars_defn_lr **from**
okay_Readvars $\{p_3 \leftarrow p_3@\mathsf{CS}\}$

precision_enhancement_pr: **Prove** precision_enhancement **from**
precision_enhancement_ax $\{$ppred $\leftarrow (\lambda\, q$ : correct$(q, t_p^{i+1}))\}$,
okay_Readpred_Reading $\{t \leftarrow t_p^{i+1},\ y \leftarrow Y\}$,
okay_Readpred_Reading $\{t \leftarrow t_p^{i+1},\ y \leftarrow Y,\ \gamma \leftarrow \theta\}$,
okay_pairs_Readvars $\{t \leftarrow t_p^{i+1},\ x \leftarrow X\}$,
correct_count $\{t \leftarrow t_p^{i+1}\}$

okay_Readpred_Reading_pr: **Prove** okay_Readpred_Reading **from**
okay_Readpred $\{$ppred $\leftarrow (\lambda\, p$ : correct$(p, t))\}$,
okay_Reading $\{p_1 \leftarrow l@P1S,\ q_1 \leftarrow m@P1S\}$

okay_pairs_Readvars_pr: **Prove** okay_pairs_Readvars **from**
okay_pairs $\{$ppred $\leftarrow (\lambda\, p$ : correct$(p, t))\}$, okay_Readvars $\{p_3 \leftarrow p_3$@P1S$\}$

rts_0_proof: **Prove** rts_0 **from** rts0 $\{t \leftarrow t_p^{i+1}\}$

rts_1_proof: **Prove** rts_1 **from** rts1 $\{t \leftarrow t_p^{i+1}\}$

**End** clockassumptions

# Appendix B

# Bounded Delay Modules

This appendix contains the EHDM proof modules for the extended clock synchronization theory. The proof chain analysis is taken from modules delay4, rmax_rmin, and new_basics. Module delay4 contains the proofs of bounded delay, whereas rmax_rmin and new_basics show that the new conditions are sufficient for establishing some of the old constraints from Shankar's theory. Several lines of the proof analysis have been deleted. The pertinent information concerning the axioms at the base of the proof chain remains.

## B.1 Proof Analysis

### B.1.1 Proof Chain for delay4

```
Terse proof chains for module delay4

⋮

                        SUMMARY

The proof chain is complete

The axioms and assumptions at the base are:
  clockassumptions.IClock_defn
  clockassumptions.N_maxfaults
  clockassumptions.accuracy_preservation_recovery_ax
  clockassumptions.precision_enhancement_recovery_ax
  clockassumptions.rho_0
  clockassumptions.translation_invariance
  delay.FIX_SYNC
  delay.RATE_1
  delay.RATE_2
  delay.R_FIX_SYNC_0
  delay.betaread_ax
```

```
delay.bnd_delay_init
delay.fix_between_sync
delay.good_read_pred_ax1
delay.read_self
delay.reading_error3
delay.rts_new_1
delay.rts_new_2
delay.synctime0_defn
delay.synctime_defn
delay.wpred_ax
delay.wpred_correct
delay.wpred_preceding
delay3.betaprime_ax
delay3.recovery_lemma
delay4.option1_defn
delay4.option2_defn
delay4.options_exhausted
division.mult_div_1
division.mult_div_2
division.mult_div_3
floor_ceil.ceil_defn
floor_ceil.floor_defn
multiplication.mult_non_neg
multiplication.mult_pos
noetherian[EXPR, EXPR].general_induction
Total: 36
```

⋮

### B.1.2 Proof Chain for rmax_rmin

```
Terse proof chains for module rmax_rmin
```

⋮

```
                        SUMMARY

The proof chain is complete

The axioms and assumptions at the base are:
  clockassumptions.IClock_defn
  clockassumptions.accuracy_preservation_recovery_ax
```

```
    clockassumptions.precision_enhancement_recovery_ax
    clockassumptions.rho_0
    clockassumptions.translation_invariance
    delay.FIX_SYNC
    delay.RATE_1
    delay.RATE_2
    delay.R_FIX_SYNC_0
    delay.betaread_ax
    delay.bnd_delay_init
    delay.fix_between_sync
    delay.good_read_pred_ax1
    delay.read_self
    delay.reading_error3
    delay.rts_new_1
    delay.rts_new_2
    delay.synctime0_defn
    delay.synctime_defn
    delay.wpred_ax
    delay.wpred_correct
    delay.wpred_preceding
    delay3.betaprime_ax
    delay3.recovery_lemma
    delay4.option1_defn
    delay4.option2_defn
    delay4.options_exhausted
    division.mult_div_1
    division.mult_div_2
    division.mult_div_3
    floor_ceil.ceil_defn
    floor_ceil.floor_defn
    multiplication.mult_non_neg
    multiplication.mult_pos
    noetherian[EXPR, EXPR].general_induction
    rmax_rmin.ADJ_recovery
Total: 36

⋮
```

### B.1.3 Proof Chain for new_basics

```
Terse proof chains for module new_basics

⋮
```

SUMMARY

```
The proof chain is complete

The axioms and assumptions at the base are:
  clockassumptions.IClock_defn
  clockassumptions.N_maxfaults
  clockassumptions.accuracy_preservation_recovery_ax
  clockassumptions.precision_enhancement_recovery_ax
  clockassumptions.rho_0
  clockassumptions.translation_invariance
  delay.FIX_SYNC
  delay.RATE_1
  delay.RATE_2
  delay.R_FIX_SYNC_0
  delay.betaread_ax
  delay.bnd_delay_init
  delay.fix_between_sync
  delay.good_read_pred_ax1
  delay.read_self
  delay.reading_error3
  delay.rts_new_1
  delay.rts_new_2
  delay.synctime0_defn
  delay.synctime_defn
  delay.wpred_ax
  delay.wpred_correct
  delay.wpred_preceding
  delay3.betaprime_ax
  delay3.recovery_lemma
  delay4.option1_defn
  delay4.option2_defn
  delay4.options_exhausted
  division.mult_div_1
  division.mult_div_2
  division.mult_div_3
  floor_ceil.ceil_defn
  floor_ceil.floor_defn
  multiplication.mult_non_neg
  multiplication.mult_pos
  new_basics.delay_recovery
  new_basics.nonoverlap
  noetherian[EXPR, EXPR].general_induction
  rmax_rmin.ADJ_recovery
Total: 39
```

## B.2 delay

delay: **Module**

**Using** arith, clockassumptions

**Exporting all with** clockassumptions

**Theory**

$p, q, p_1, q_1$: **Var** process
$i, j, k$: **Var** event
$X, S, T$: **Var** Clocktime
$s, t, t_1, t_2$: **Var** time
$\gamma$: **Var** function[process → Clocktime]
$\beta', \beta_{\text{read}}, \Lambda'$: number
$R$: Clocktime

betaread_ax: **Axiom** $\beta' \leq \beta_{\text{read}} \wedge \beta_{\text{read}} < R/2$

ppred, ppred1: **Var** function[process → bool]
$S^0$: Clocktime
$S^{\star 1}$: function[event → Clocktime] = $(\lambda\, i : i * R + S^0)$
$pc_{\star 1}(\star 2)$: function[process, Clocktime → time]
$ic^{\star 2}_{\star 1}(\star 3)$: function[process, event, Clocktime → time] =
$(\lambda\, p, i, T : pc_p(T - adj^i_p))$
$s^{\star 2}_{\star 1}$: function[process, event → time] = $(\lambda\, p, i : ic^i_p(S^i))$
$T^0$: Clocktime
$T^{\star 2}_{\star 1}$: function[process, event → Clocktime]

synctime_defn: **Axiom** $t^{i+1}_p = ic^i_p(T^{i+1}_p)$

synctime0_defn: **Axiom** $t^0_p = ic^0_p(T^0)$

FIX_SYNC: **Axiom** $S^0 > T^0$

R_FIX_SYNC_0: **Axiom** $R > (S^0 - T^0)$

R_0: **Lemma** $R > 0$

good_read_pred: function[event → function[process, process → bool]]
correct_during: function[process, time, time → bool] =
$(\lambda\, p, t, s : t \leq s \wedge (\forall\, t_1 : t \leq t_1 \wedge t_1 \leq s \supset \text{correct}(p, t_1)))$
wpred: function[event → function[process → bool]]
rpred: function[event → function[process → bool]]
wvr_pred: function[event → function[process → bool]] =
$(\lambda\, i : (\lambda\, p : \text{wpred}(i)(p) \vee \text{rpred}(i)(p)))$
working: function[process, time → bool] =
$(\lambda\, p, t : (\exists\, i : \text{wpred}(i)(p) \wedge t^i_p \leq t \wedge t < t^{i+1}_p))$

wvr_defn: **Lemma** wvr_pred$(i) = (\lambda\, p : \text{wpred}(i)(p) \vee \text{rpred}(i)(p))$

wpred_wvr: **Lemma** wpred$(i)(p) \supset$ wvr_pred$(i)(p)$

rpred_wvr: **Lemma** rpred$(i)(p) \supset$ wvr_pred$(i)(p)$

wpred_ax: **Axiom** count(wpred$(i), N) \geq N - F$

wvr_count: **Lemma** count(wvr_pred$(i), N) \geq N - F$

wpred_correct: **Axiom** wpred$(i)(p) \supset$ correct_during$(p, t_p^i, t_p^{i+1})$

wpred_preceding: **Axiom** wpred$(i+1)(p) \supset$ wpred$(i)(p) \vee$ rpred$(i)(p)$

wpred_rpred_disjoint: **Axiom** $\neg$(wpred$(i)(p) \wedge$ rpred$(i)(p)$)

wpred_bridge: **Axiom**
wvr_pred$(i)(p) \wedge$ correct_during$(p, t_p^{i+1}, t_p^{i+2}) \supset$ wpred$(i+1)(p)$

wpred_fixtime: **Lemma** wpred$(i)(p) \supset$ correct_during$(p, s_p^i, t_p^{i+1})$

wpred_fixtime_low: **Lemma** wpred$(i)(p) \supset$ correct_during$(p, t_p^i, s_p^i)$

correct_during_trans: **Lemma**
correct_during$(p, t, t_2) \wedge$ correct_during$(p, t_2, s)$
$\supset$ correct_during$(p, t, s)$

correct_during_sub_left: **Lemma**
correct_during$(p, t, s) \wedge t \leq t_2 \wedge t_2 \leq s \supset$ correct_during$(p, t, t_2)$

correct_during_sub_right: **Lemma**
correct_during$(p, t, s) \wedge t \leq t_2 \wedge t_2 \leq s \supset$ correct_during$(p, t_2, s)$

wpred_lo_lem: **Lemma** wpred$(i)(p) \supset$ correct$(p, t_p^i)$

wpred_hi_lem: **Lemma** wpred$(i)(p) \supset$ correct$(p, t_p^{i+1})$

correct_during_hi: **Lemma** correct_during$(p, t, s) \supset$ correct$(p, s)$

correct_during_lo: **Lemma** correct_during$(p, t, s) \supset$ correct$(p, t)$

clock_ax1: **Axiom** $PC_p(pc_p(T)) = T$

clock_ax2: **Axiom** $pc_p(PC_p(t)) \leq t \wedge t < pc_p(PC_p(t) + 1)$

iclock_defn: **Lemma** $ic_p^i(T) = pc_p(T - adj_p^i)$

iclock0_defn: **Lemma** $ic_p^0(T) = pc_p(T)$

iclock_lem: **Lemma** correct$(p, ic_p^i(T)) \supset IC_p^i(ic_p^i(T)) = T$

$ADJ_{\star 1}^{\star 2}$: function[process, event $\rightarrow$ Clocktime] $= (\lambda\, p, i : adj_p^{i+1} - adj_p^i)$

IClock_ADJ_lem: **Lemma** $\mathsf{correct}(p,t) \supset IC_p^{i+1}(t) = IC_p^i(t) + ADJ_p^i$

iclock_ADJ_lem: **Lemma** $ic_p^{i+1}(T) = ic_p^i(T - ADJ_p^i)$

rts_new_1: **Axiom** $\mathsf{correct}(p, t_p^{i+1}) \supset S^i + \boldsymbol{\alpha}(\lfloor \beta' + 2 * \Lambda' \rfloor) < T_p^{i+1}$

rts_new_2: **Axiom** $\mathsf{correct}(p, t_p^i) \supset T_p^i < S^i - \boldsymbol{\alpha}(\lfloor \beta' + 2 * \Lambda' \rfloor)$

FIXTIME_bound: **Lemma**
$\mathsf{correct}(p, t_p^{i+1}) \supset S^{i+1} > S^i + 2 * \boldsymbol{\alpha}(\lfloor \beta' + 2 * \Lambda' \rfloor)$

R_bound: **Lemma** $\mathsf{correct}(p, t_p^{i+1}) \supset R > 2 * \boldsymbol{\alpha}(\lfloor \beta' + 2 * \Lambda' \rfloor)$

RATE_1: **Axiom** $\mathsf{correct\_during}(p, pc_p(T), pc_p(S)) \wedge S \geq T$
$\supset pc_p(S) - pc_p(T) \leq (S - T) \star (1 + \rho)$

RATE_2: **Axiom** $\mathsf{correct\_during}(p, pc_p(T), pc_p(S)) \wedge S \geq T$
$\supset pc_p(S) - pc_p(T) \geq (S - T)/(1 + \rho)$

RATE_1_iclock: **Lemma**
$\mathsf{correct\_during}(p, ic_p^i(T), ic_p^i(S)) \wedge S \geq T$
$\supset ic_p^i(S) - ic_p^i(T) \leq (S - T) \star (1 + \rho)$

RATE_2_iclock: **Lemma**
$\mathsf{correct\_during}(p, ic_p^i(T), ic_p^i(S)) \wedge S \geq T$
$\supset ic_p^i(S) - ic_p^i(T) \geq (S - T)/(1 + \rho)$

rate_simplify: **Lemma** $S \geq T \supset (S - T)/(1 + \rho) \geq (S - T) \star (1 - \rho)$

rate_simplify_step: **Lemma** $S \geq T \supset (1 + \rho) \star (S - T) \star (1 - \rho) \leq S - T$

RATE_2_simplify: **Lemma**
$\mathsf{correct\_during}(p, pc_p(T), pc_p(S)) \wedge S \geq T$
$\supset pc_p(S) - pc_p(T) \geq (S - T) \star (1 - \rho)$

RATE_2_simplify_iclock: **Lemma**
$\mathsf{correct\_during}(p, ic_p^i(T), ic_p^i(S)) \wedge S \geq T$
$\supset ic_p^i(S) - ic_p^i(T) \geq (S - T) \star (1 - \rho)$

RATE_lemma1: **Lemma**
$\mathsf{correct\_during}(p, pc_p(T), pc_p(S))$
$\wedge \mathsf{correct\_during}(q, pc_q(T), pc_q(S)) \wedge S \geq T$
$\supset |pc_p(S) - pc_q(S)| \leq |pc_p(T) - pc_q(T)| + 2 * \rho \star (S - T)$

RATE_lemma1_iclock: **Lemma**
$\mathsf{correct\_during}(p, ic_p^i(T), ic_p^i(S))$
$\wedge \mathsf{correct\_during}(q, ic_q^i(T), ic_q^i(S)) \wedge S \geq T$
$\supset |ic_p^i(S) - ic_q^i(S)| \leq |ic_p^i(T) - ic_q^i(T)| + 2 * \rho \star (S - T)$

RATE_lemma2: **Lemma**
correct_during$(p, pc_p(T), pc_p(S)) \wedge S \geq T$
$\supset |(pc_p(S) - S) - (pc_p(T) - T)| \leq \rho \star (|S - T|)$

RATE_lemma2_iclock: **Lemma**
correct_during$(p, ic_p^i(T), ic_p^i(S)) \wedge S \geq T$
$\supset |(ic_p^i(S) - S) - (ic_p^i(T) - T)| \leq \rho \star (|S - T|)$

bnd_delay_init: **Axiom**
wpred$(0)(p) \wedge$ wpred$(0)(q)$
$\supset |t_p^0 - t_q^0| \leq \beta' - 2 * \rho \star (S^0 - T^0) \wedge \beta' - 2 * (\rho \star (S^0 - T^0)) \leq \beta$

bnd_delay_off_init: **Lemma** wpred$(0)(p) \wedge$ wpred$(0)(q) \supset |s_p^0 - s_q^0| \leq \beta'$

good_read_pred_ax1: **Axiom**
correct_during$(p, s_p^i, t_p^{i+1})$
$\wedge$ correct_during$(q, s_q^i, t_q^{i+1}) \wedge |s_p^i - s_q^i| \leq \beta_{\text{read}}$
$\supset$ good_read_pred$(i)(p, q)$

reading_error3: **Axiom**
good_read_pred$(i)(p, q)$
$\supset |(\Theta_p^{i+1}(q) - IC_p^i(t_p^{i+1})) - (s_p^i - s_q^i)| \leq \Lambda'$

ADJ_lem1: **Lemma** correct_during$(p, s_p^i, t_p^{i+1})$
$\supset (ADJ_p^i = cfn(p, (\lambda\, p_1 : \Theta_p^{i+1}(p_1) - IC_p^i(t_p^{i+1}))))$

ADJ_lem2: **Lemma** correct_during$(p, s_p^i, t_p^{i+1})$
$\supset (ADJ_p^i = cfn(p, \Theta_p^{i+1}) - IC_p^i(t_p^{i+1}))$

read_self: **Axiom** wpred$(i)(p) \supset \Theta_p^{i+1}(p) = IC_p^i(t_p^{i+1})$

fix_between_sync: **Axiom**
correct_during$(p, t_p^i, t_p^{i+1}) \supset t_p^i < s_p^i \wedge s_p^i < t_p^{i+1}$

rts_2_lo: **Lemma** wpred$(i)(p) \wedge$ wpred$(i)(q) \supset |t_p^i - t_q^i| \leq \beta$

rts_2_hi: **Axiom** wpred$(i)(p) \wedge$ wpred$(i)(q) \supset |t_p^{i+1} - t_q^{i+1}| \leq \beta$

## Proof

R_0_pr: **Prove** R_0 **from** R_FIX_SYNC_0, FIX_SYNC

FIXTIME_bound_pr: **Prove** FIXTIME_bound **from** rts_new_1, rts_new_2 $\{i \leftarrow i + 1\}$

R_bound_pr: **Prove** R_bound **from** FIXTIME_bound, $S^{\star 1}$ , $S^{\star 1}$ $\{i \leftarrow i + 1\}$

iclock_defn_pr: **Prove** iclock_defn **from** $ic_{\star 1}^{\star 2}(\star 3)$

wpred_fixtime_pr: **Prove** wpred_fixtime **from**
  fix_between_sync,
  wpred_correct,
  correct_during_sub_right $\{s \leftarrow t_p^{i+1},\ t \leftarrow t_p^i,\ t_2 \leftarrow s_p^i\}$

wpred_fixtime_low_pr: **Prove** wpred_fixtime_low **from**
  fix_between_sync,
  wpred_correct,
  correct_during_sub_left $\{s \leftarrow t_p^{i+1},\ t \leftarrow t_p^i,\ t_2 \leftarrow s_p^i\}$

correct_during_sub_left_pr: **Prove** correct_during_sub_left **from**
  correct_during $\{s \leftarrow t_2\}$, correct_during $\{t_1 \leftarrow t_1@p1\}$

correct_during_sub_right_pr: **Prove** correct_during_sub_right **from**
  correct_during $\{t \leftarrow t_2\}$, correct_during $\{t_1 \leftarrow t_1@p1\}$

correct_during_trans_pr: **Prove** correct_during_trans **from**
  correct_during,
  correct_during $\{s \leftarrow t_2,\ t_1 \leftarrow t_1@p1\}$,
  correct_during $\{t \leftarrow t_2,\ t_1 \leftarrow t_1@p1\}$

wpred_wvr_pr: **Prove** wpred_wvr **from** wvr_defn

rpred_wvr_pr: **Prove** rpred_wvr **from** wvr_defn

wvr_defn_hack: **Lemma**
  $(\forall p : \mathsf{wvr\_pred}(i)(p) = ((\lambda p : \mathsf{wpred}(i)(p) \vee \mathsf{rpred}(i)(p))p))$

wvr_defn_hack_pr: **Prove** wvr_defn_hack **from** wvr_pred $\{p \leftarrow p@c\}$

wvr_defn_pr: **Prove** wvr_defn **from**
  pred_extensionality
    {pred1 $\leftarrow$ wvr_pred$(i)$,
     pred2 $\leftarrow$ $(\lambda p : \mathsf{wpred}(i)(p) \vee \mathsf{rpred}(i)(p))$},
  wvr_defn_hack $\{p \leftarrow p@p1\}$

wvr_count_pr: **Prove** wvr_count **from**
  wpred_ax,
  count_imp
    {ppred1 $\leftarrow$ wpred$(i)$,
     ppred2 $\leftarrow$ $(\lambda p : \mathsf{wpred}(i)(p) \vee \mathsf{rpred}(i)(p))$,
     $n \leftarrow N$},
  wvr_defn,
  imp_pred_or {ppred1 $\leftarrow$ wpred$(i)$, ppred2 $\leftarrow$ rpred$(i)$}

$w, x, y, z$: **Var** number

bd_hack: **Lemma** $|w| \leq x - y \wedge |z| \leq |w| + y \supset |z| \leq x$

bd_hack_pr: **Prove** bd_hack

bnd_delay_off_init_pr: **Prove** bnd_delay_off_init **from**
bnd_delay_init,
RATE_lemma1_iclock $\{S \leftarrow S^0,\ T \leftarrow T^0,\ i \leftarrow 0\}$,
FIX_SYNC,
synctime0_defn,
synctime0_defn $\{p \leftarrow q\}$,
$s^{\star 2}_{\star 1}$ $\{i \leftarrow 0\}$,
$s^{\star 2}_{\star 1}$ $\{i \leftarrow 0,\ p \leftarrow q\}$,
wpred_fixtime_low $\{i \leftarrow 0\}$,
wpred_fixtime_low $\{p \leftarrow q,\ i \leftarrow 0\}$,
$S^{\star 1}$ $\{i \leftarrow 0\}$

mult_abs_hack: **Lemma** $x \star (1-\rho) \leq y \wedge y \leq x \star (1+\rho) \supset |y-x| \leq \rho \star x$

mult_abs_hack_pr: **Prove** mult_abs_hack **from**
mult_ldistrib $\{y \leftarrow 1,\ z \leftarrow \rho\}$,
mult_ldistrib_minus $\{y \leftarrow 1,\ z \leftarrow \rho\}$,
mult_rident,
abs_3_bnd $\{x \leftarrow y,\ y \leftarrow x,\ z \leftarrow \rho \star x\}$,
mult_com $\{y \leftarrow \rho\}$

RATE_1_iclock_pr: **Prove** RATE_1_iclock **from**
RATE_1 $\{S \leftarrow S - adj^i_p,\ T \leftarrow T - adj^i_p\}$,
iclock_defn,
iclock_defn $\{T \leftarrow S\}$

RATE_2_iclock_pr: **Prove** RATE_2_iclock **from**
RATE_2 $\{S \leftarrow S - adj^i_p,\ T \leftarrow T - adj^i_p\}$,
iclock_defn,
iclock_defn $\{T \leftarrow S\}$

RATE_2_simplify_iclock_pr: **Prove** RATE_2_simplify_iclock **from**
RATE_2_simplify $\{S \leftarrow S - adj^i_p,\ T \leftarrow T - adj^i_p\}$,
iclock_defn,
iclock_defn $\{T \leftarrow S\}$

RATE_lemma1_sym: **Lemma**
correct_during$(p, pc_p(T), pc_p(S))$
$\wedge$ correct_during$(q, pc_q(T), pc_q(S)) \wedge S \geq T \wedge pc_p(S) \geq pc_q(S)$
$\supset |pc_p(S) - pc_q(S)| \leq |pc_p(T) - pc_q(T)| + 2 * \rho \star (S-T)$

Rl1hack: **Lemma** $w \leq x \wedge y \leq z \wedge y \geq x \supset |y-x| \leq |z-w|$

Rl1hack_pr: **Prove** Rl1hack **from** $|\star 1|$ $\{x \leftarrow y-x\}$, $|\star 1|$ $\{x \leftarrow z-w\}$

RATE_lemma1_sym_pr: **Prove** RATE_lemma1_sym **from**
RATE_1,
RATE_2_simplify $\{p \leftarrow q\}$,
RI1hack
$\{x \leftarrow pc_q(S)$,
$y \leftarrow pc_p(S)$,
$w \leftarrow pc_q(T) + (S - T) \star (1 - \rho)$,
$z \leftarrow pc_p(T) + (S - T) \star (1 + \rho)\}$,
mult_ldistrib $\{x \leftarrow S - T,\ y \leftarrow 1,\ z \leftarrow \rho\}$,
mult_ldistrib_minus $\{x \leftarrow S - T,\ y \leftarrow 1,\ z \leftarrow \rho\}$,
abs_plus $\{x \leftarrow pc_p(T) - pc_q(T),\ y \leftarrow 2 * \rho \star (S - T)\}$,
mult_com $\{x \leftarrow \rho,\ y \leftarrow S - T\}$,
abs_ge0 $\{x \leftarrow 2 * \rho \star (S - T)\}$,
mult_non_neg $\{x \leftarrow \rho,\ y \leftarrow S - T\}$,
rho_0

RATE_lemma1_pr: **Prove** RATE_lemma1 **from**
RATE_lemma1_sym,
RATE_lemma1_sym $\{p \leftarrow q,\ q \leftarrow p\}$,
abs_com $\{x \leftarrow pc_p(S),\ y \leftarrow pc_q(S)\}$,
abs_com $\{x \leftarrow pc_p(T),\ y \leftarrow pc_q(T)\}$

RATE_lemma1_iclock_sym: **Lemma**
correct_during$(p, ic_p^i(T), ic_p^i(S))$
$\wedge$ correct_during$(q, ic_q^i(T), ic_q^i(S)) \wedge S \geq T \wedge ic_p^i(S) \geq ic_q^i(S)$
$\supset |ic_p^i(S) - ic_q^i(S)| \leq |ic_p^i(T) - ic_q^i(T)| + 2 * \rho \star (S - T)$

RATE_lemma1_iclock_sym_pr: **Prove** RATE_lemma1_iclock_sym **from**
RATE_1_iclock,
RATE_2_simplify_iclock $\{p \leftarrow q\}$,
RI1hack
$\{x \leftarrow ic_q^i(S)$,
$y \leftarrow ic_p^i(S)$,
$w \leftarrow ic_q^i(T) + (S - T) \star (1 - \rho)$,
$z \leftarrow ic_p^i(T) + (S - T) \star (1 + \rho)\}$,
mult_ldistrib $\{x \leftarrow S - T,\ y \leftarrow 1,\ z \leftarrow \rho\}$,
mult_ldistrib_minus $\{x \leftarrow S - T,\ y \leftarrow 1,\ z \leftarrow \rho\}$,
abs_plus $\{x \leftarrow ic_p^i(T) - ic_q^i(T),\ y \leftarrow 2 * \rho \star (S - T)\}$,
mult_com $\{x \leftarrow \rho,\ y \leftarrow S - T\}$,
abs_ge0 $\{x \leftarrow 2 * \rho \star (S - T)\}$,
mult_non_neg $\{x \leftarrow \rho,\ y \leftarrow S - T\}$,
rho_0

RATE_lemma1_iclock_pr: **Prove** RATE_lemma1_iclock **from**
RATE_lemma1_iclock_sym,
RATE_lemma1_iclock_sym $\{p \leftarrow q,\ q \leftarrow p\}$,
abs_com $\{x \leftarrow ic_p^i(S),\ y \leftarrow ic_q^i(S)\}$,
abs_com $\{x \leftarrow ic_p^i(T),\ y \leftarrow ic_q^i(T)\}$

RATE_lemma2_pr: **Prove** RATE_lemma2 **from**
RATE_1,
RATE_2_simplify,
mult_abs_hack $\{x \leftarrow S - T,\ y \leftarrow pc_p(S) - pc_p(T)\}$,
abs_ge0 $\{x \leftarrow S - T\}$

RATE_lemma2_iclock_pr: **Prove** RATE_lemma2_iclock **from**
RATE_lemma2 $\{S \leftarrow S - adj_p^i,\ T \leftarrow T - adj_p^i\}$,
iclock_defn $\{T \leftarrow S\}$,
iclock_defn

wpred_lo_lem_pr: **Prove** wpred_lo_lem **from**
wpred_correct,
correct_during $\{s \leftarrow t_p^{i+1},\ t \leftarrow t_p^i,\ t_1 \leftarrow t_p^i\}$

wpred_hi_lem_pr: **Prove** wpred_hi_lem **from**
wpred_correct,
correct_during $\{s \leftarrow t_p^{i+1},\ t \leftarrow t_p^i,\ t_1 \leftarrow t_p^{i+1}\}$

correct_during_hi_pr: **Prove** correct_during_hi **from** correct_during $\{t_1 \leftarrow s\}$

correct_during_lo_pr: **Prove** correct_during_lo **from** correct_during $\{t_1 \leftarrow t\}$

mult_assoc: **Lemma** $x \star (y \star z) = (x \star y) \star z$

mult_assoc_pr: **Prove** mult_assoc **from**
$\star 1 \star \star 2$ $\{y \leftarrow y \star z\}$,
$\star 1 \star \star 2$ ,
$\star 1 \star \star 2$ $\{x \leftarrow y,\ y \leftarrow z\}$,
$\star 1 \star \star 2$ $\{x \leftarrow x \star y,\ y \leftarrow z\}$

diff_squares: **Lemma** $(1 + \rho) \star (1 - \rho) = 1 - \rho \star \rho$

diff_squares_pr: **Prove** diff_squares **from**
distrib $\{x \leftarrow 1,\ y \leftarrow \rho,\ z \leftarrow 1 - \rho\}$,
mult_lident $\{x \leftarrow 1 - \rho\}$,
mult_ldistrib_minus $\{x \leftarrow \rho,\ y \leftarrow 1,\ z \leftarrow \rho\}$,
mult_rident $\{x \leftarrow \rho\}$

rate_simplify_step_pr: **Prove** rate_simplify_step **from**
mult_com $\{x \leftarrow (S-T),\ y \leftarrow (1-\rho)\}$,
mult_assoc $\{x \leftarrow 1+\rho,\ y \leftarrow 1-\rho,\ z \leftarrow S-T\}$,
diff_squares,
distrib_minus $\{x \leftarrow 1,\ y \leftarrow \rho \star \rho,\ z \leftarrow S-T\}$,
mult_lident $\{x \leftarrow S-T\}$,
pos_product $\{x \leftarrow \rho \star \rho,\ y \leftarrow S-T\}$,
pos_product $\{x \leftarrow \rho,\ y \leftarrow \rho\}$,
rho_0

rate_simplify_pr: **Prove** rate_simplify **from**
div_ineq
$\{z \leftarrow (1+\rho),$
$y \leftarrow (S-T),$
$x \leftarrow (1+\rho) \star (S-T) \star (1-\rho)\}$,
div_cancel $\{x \leftarrow (1+\rho),\ y \leftarrow (S-T) \star (1-\rho)\}$,
rho_0,
rate_simplify_step

RATE_2_simplify_pr: **Prove** RATE_2_simplify **from** RATE_2, rate_simplify

iclock_lem_pr: **Prove** iclock_lem **from**
iclock_defn, IClock_defn $\{t \leftarrow ic_p^i(T)\}$, clock_ax1 $\{T \leftarrow T - adj_p^i\}$

IClock_ADJ_lem_pr: **Prove** IClock_ADJ_lem **from**
IClock_defn, IClock_defn $\{i \leftarrow i+1\}$, $ADJ_{\star 1}^{\star 2}$

iclock_ADJ_lem_pr: **Prove** iclock_ADJ_lem **from**
iclock_defn $\{T \leftarrow T - ADJ_p^i\}$, iclock_defn $\{i \leftarrow i+1\}$, $ADJ_{\star 1}^{\star 2}$

ADJ_lem1_pr: **Prove** ADJ_lem1 **from**
ADJ_lem2,
translation_invariance $\{X \leftarrow -IC_p^i(t_p^{i+1}),\ \gamma \leftarrow \Theta_p^{i+1}\}$

ADJ_lem2_pr: **Prove** ADJ_lem2 **from**
$ADJ_{\star 1}^{\star 2}$ ,
$adj_{\star 1}^{\star 2}$ $\{i \leftarrow i+1\}$,
IClock_defn $\{t \leftarrow t_p^{i+1},\ i \leftarrow i\}$,
correct_during_hi $\{t \leftarrow s_p^i,\ s \leftarrow t_p^{i+1}\}$

**End** delay

## B.3 delay2

delay2: **Module**

**Using** arith, clockassumptions, delay

**Exporting all with** clockassumptions, delay

**Theory**

$p, q, p_1, q_1$: **Var** process
$i$: **Var** event
delay_pred: function[event → bool] =
$(\lambda\, i : (\forall\, p, q : \mathsf{wpred}(i)(p) \wedge \mathsf{wpred}(i)(q) \supset |s_p^i - s_q^i| \leq \beta'))$
ADJ_pred: function[event → bool] =
$(\lambda\, i : (\forall\, p : i \geq 1 \wedge \mathsf{wpred}(i-1)(p) \supset |ADJ_p^{i-1}| \leq \boldsymbol{\alpha}(\lfloor \beta' + 2 * \Lambda' \rfloor)))$

delay_pred_lr: **Lemma**
delay_pred$(i) \supset (\mathsf{wpred}(i)(p) \wedge \mathsf{wpred}(i)(q) \supset |s_p^i - s_q^i| \leq \beta')$

bnd_delay_offset: **Theorem** ADJ_pred$(i) \wedge$ delay_pred$(i)$

bnd_delay_offset_0: **Lemma** ADJ_pred$(0) \wedge$ delay_pred$(0)$

bnd_delay_offset_ind: **Lemma**
ADJ_pred$(i) \wedge$ delay_pred$(i) \supset$ ADJ_pred$(i+1) \wedge$ delay_pred$(i+1)$

bnd_delay_offset_ind_a: **Lemma** delay_pred$(i) \supset$ ADJ_pred$(i+1)$

bnd_delay_offset_ind_b: **Lemma**
delay_pred$(i) \wedge$ ADJ_pred$(i+1) \supset$ delay_pred$(i+1)$

good_ReadClock: **Lemma**
delay_pred$(i) \wedge \mathsf{wpred}(i)(p) \supset$ okay_Readpred$(\Theta_p^{i+1}, \beta' + 2 * \Lambda', \mathsf{wpred}(i))$

good_ReadClock_recover: **Axiom**
delay_pred$(i) \wedge \mathsf{rpred}(i)(p) \supset$ okay_Readpred$(\Theta_p^{i+1}, \beta' + 2 * \Lambda', \mathsf{wpred}(i))$

delay_prec_enh: **Lemma**
delay_pred$(i) \wedge \mathsf{wpred}(i)(p) \wedge \mathsf{wpred}(i)(q)$
$\supset |(s_p^i - s_q^i) - (ADJ_p^i - ADJ_q^i)| \leq \boldsymbol{\pi}(\lfloor 2 * \Lambda' + 2 \rfloor, \lfloor \beta' + 2 * \Lambda' \rfloor)$

delay_prec_enh_step1: **Lemma**
delay_pred$(i) \wedge \mathsf{wpred}(i)(p) \wedge \mathsf{wpred}(i)(q)$
$\supset |cfn(p, (\lambda\, p_1 : \Theta_p^{i+1}(p_1) - IC_p^i(t_p^{i+1}) - \lfloor s_p^i \rfloor))$
$- cfn(q, (\lambda\, p_1 : \Theta_q^{i+1}(p_1) - IC_q^i(t_q^{i+1}) - \lceil s_q^i \rceil))|$
$\leq \boldsymbol{\pi}(\lfloor 2 * \Lambda' + 2 \rfloor, \lfloor \beta' + 2 * \Lambda' \rfloor)$

delay_prec_enh_step1_sym: **Lemma**

$$\begin{aligned}&\mathsf{delay\_pred}(i) \wedge \mathsf{wpred}(i)(p) \wedge \mathsf{wpred}(i)(q) \wedge (ADJ_p^i - s_p^i \geq ADJ_q^i - s_q^i)\\ &\quad \supset |(ADJ_p^i - s_p^i) - (ADJ_q^i - s_q^i)|\\ &\qquad \leq |cfn(p, (\lambda\, p_1 : \Theta_p^{i+1}(p_1) - IC_p^i(t_p^{i+1}) - \lfloor s_p^i \rfloor))\\ &\qquad\qquad - cfn(q, (\lambda\, p_1 : \Theta_q^{i+1}(p_1) - IC_q^i(t_q^{i+1}) - \lceil s_q^i \rceil))|\end{aligned}$$

prec_enh_hyp1: **Lemma**

$$\begin{aligned}&\mathsf{delay\_pred}(i) \wedge \mathsf{wpred}(i)(p) \wedge \mathsf{wpred}(i)(q)\\ &\quad \supset \mathsf{okay\_pairs}((\lambda\, p_1 : \Theta_p^{i+1}(p_1) - IC_p^i(t_p^{i+1}) - \lfloor s_p^i \rfloor),\\ &\qquad\qquad (\lambda\, p_1 : \Theta_q^{i+1}(p_1) - IC_q^i(t_q^{i+1}) - \lceil s_q^i \rceil),\\ &\qquad\qquad 2 * \Lambda' + 2,\\ &\qquad\qquad \mathsf{wpred}(i))\end{aligned}$$

prec_enh_hyp_2: **Lemma**

$$\begin{aligned}&\mathsf{delay\_pred}(i) \wedge \mathsf{wpred}(i)(p)\\ &\quad \supset \mathsf{okay\_Readpred}((\lambda\, p_1 : \Theta_p^{i+1}(p_1) - IC_p^i(t_p^{i+1}) - \lfloor s_p^i \rfloor),\\ &\qquad\qquad \beta' + 2 * \Lambda',\\ &\qquad\qquad \mathsf{wpred}(i))\end{aligned}$$

prec_enh_hyp_3: **Lemma**

$$\begin{aligned}&\mathsf{delay\_pred}(i) \wedge \mathsf{wpred}(i)(q)\\ &\quad \supset \mathsf{okay\_Readpred}((\lambda\, p_1 : \Theta_q^{i+1}(p_1) - IC_q^i(t_q^{i+1}) - \lceil s_q^i \rceil),\\ &\qquad\qquad \beta' + 2 * \Lambda',\\ &\qquad\qquad \mathsf{wpred}(i))\end{aligned}$$

## Proof

delay_pred_lr_pr: **Prove** delay_pred_lr **from** delay_pred

delay_prec_enh_step1_pr: **Prove** delay_prec_enh_step1 **from**
precision_enhancement_ax
{ppred $\leftarrow$ wpred($i$),
$Y \leftarrow \lfloor \beta' + 2 * \Lambda' \rfloor$,
$X \leftarrow \lfloor 2 * \Lambda' + 2 \rfloor$,
$\gamma \leftarrow (\lambda\, p_1 : \Theta_p^{i+1}(p_1) - IC_p^i(t_p^{i+1}) - \lfloor s_p^i \rfloor)$,
$\theta \leftarrow (\lambda\, p_1 : \Theta_q^{i+1}(p_1) - IC_q^i(t_q^{i+1}) - \lceil s_q^i \rceil)$},
prec_enh_hyp1,
prec_enh_hyp_2,
prec_enh_hyp_3,
wpred_ax,
okay_Readpred_floor
{ppred $\leftarrow$ wpred($i$),
$y \leftarrow \beta' + 2 * \Lambda'$,
$\gamma \leftarrow \gamma$@p1},
okay_Readpred_floor
{ppred $\leftarrow$ wpred($i$),
$y \leftarrow \beta' + 2 * \Lambda'$,
$\gamma \leftarrow \theta$@p1},
okay_pairs_floor
{ppred $\leftarrow$ wpred($i$),
$x \leftarrow 2 * \Lambda' + 2$,
$\gamma \leftarrow \gamma$@p1,
$\theta \leftarrow \theta$@p1}

prec_enh_hyp_2_pr: **Prove** prec_enh_hyp_2 **from**
good_ReadClock,
okay_Readpred
{$\gamma \leftarrow (\lambda\, p_1 : \Theta_p^{i+1}(p_1) - IC_p^i(t_p^{i+1}) - \lfloor s_p^i \rfloor)$,
$y \leftarrow \beta' + 2 * \Lambda'$,
ppred $\leftarrow$ wpred($i$)},
okay_Readpred
{$\gamma \leftarrow \Theta_p^{i+1}$,
$y \leftarrow \beta' + 2 * \Lambda'$,
ppred $\leftarrow$ wpred($i$),
$l \leftarrow l$@$p2$,
$m \leftarrow m$@$p2$}

prec_enh_hyp_3_pr: **Prove** prec_enh_hyp_3 **from**
good_ReadClock $\{p \leftarrow q\}$,
okay_Readpred
$\{\gamma \leftarrow (\lambda\, p_1 : \Theta_q^{i+1}(p_1) - IC_q^i(t_q^{i+1}) - \lceil s_q^i \rceil)$,
$y \leftarrow \beta' + 2 * \Lambda'$,
ppred $\leftarrow$ wpred$(i)\}$,
okay_Readpred
$\{\gamma \leftarrow \Theta_q^{i+1}$,
$y \leftarrow \beta' + 2 * \Lambda'$,
ppred $\leftarrow$ wpred$(i)$,
$l \leftarrow l@p2$,
$m \leftarrow m@p2\}$

bnd_del_off_0_pr: **Prove** bnd_delay_offset_0 **from**
ADJ_pred $\{i \leftarrow 0\}$,
delay_pred $\{i \leftarrow 0\}$,
bnd_delay_off_init $\{p \leftarrow p@p2,\ q \leftarrow q@p2\}$

bnd_delay_offset_ind_pr: **Prove** bnd_delay_offset_ind **from**
bnd_delay_offset_ind_a, bnd_delay_offset_ind_b

bnd_delay_offset_pr: **Prove** bnd_delay_offset **from**
induction $\{$prop $\leftarrow (\lambda\, i :$ ADJ_pred$(i) \wedge$ delay_pred$(i))\}$,
bnd_delay_offset_0,
bnd_delay_offset_ind $\{i \leftarrow j@p1\}$

$a, b, c, d, e, f, g, h$: **Var** number

abs_hack: **Lemma** $|a - b|$
$\leq |e - f| + |(a - c) - (d - e)| + |(b - c) - (d - f)|$

abs_hack_pr: **Prove** abs_hack **from**
abs_com $\{x \leftarrow f,\ y \leftarrow e\}$,
abs_com $\{x \leftarrow (d - f),\ y \leftarrow (b - c)\}$,
abs_plus
$\{x \leftarrow (f - e)$,
$y \leftarrow ((a - c) - (d - e)) + ((d - f) - (b - c))\}$,
abs_plus $\{x \leftarrow ((a - c) - (d - e)),\ y \leftarrow ((d - f) - (b - c))\}$

abshack2: **Lemma** $|a| \leq b \wedge |c| \leq d \wedge |e| \leq d \supset |a| + |c| + |e| \leq b + 2 * d$

abshack2_pr: **Prove** abshack2

good_ReadClock_pr: **Prove** good_ReadClock **from**
  okay_Readpred
    $\{\gamma \leftarrow \Theta_p^{i+1}$,
     $y \leftarrow \beta' + 2 * \Lambda'$,
     ppred $\leftarrow$ wpred$(i)\}$,
  delay_pred $\{p \leftarrow l@p1,\ q \leftarrow m@p1\}$,
  delay_pred $\{q \leftarrow l@p1\}$,
  delay_pred $\{q \leftarrow m@p1\}$,
  reading_error3 $\{q \leftarrow l@p1\}$,
  reading_error3 $\{q \leftarrow m@p1\}$,
  abs_hack
    $\{a \leftarrow \Theta_p^{i+1}(l@p1)$,
     $b \leftarrow \Theta_p^{i+1}(m@p1)$,
     $c \leftarrow IC_p^i(t_p^{i+1})$,
     $d \leftarrow s_p^i$,
     $e \leftarrow s_{l@p1}^i$,
     $f \leftarrow s_{m@p1}^i\}$,
  abshack2
    $\{a \leftarrow e@p7 - f@p7$,
     $b \leftarrow \beta'$,
     $c \leftarrow ((a@p7 - c@p7) - (d@p7 - e@p7))$,
     $d \leftarrow \Lambda'$,
     $e \leftarrow ((b@p7 - c@p7) - (d@p7 - f@p7))\}$,
  good_read_pred_ax1 $\{q \leftarrow l@p1\}$,
  good_read_pred_ax1 $\{q \leftarrow m@p1\}$,
  wpred_fixtime,
  wpred_fixtime $\{p \leftarrow l@p1\}$,
  wpred_fixtime $\{p \leftarrow m@p1\}$,
  betaread_ax

bnd_del_off_ind_a_pr: **Prove** bnd_delay_offset_ind_a **from**
ADJ_pred $\{i \leftarrow i+1\}$,
ADJ_lem2 $\{p \leftarrow p@p1\}$,
accuracy_preservation_ax
$\{$ppred $\leftarrow$ wpred$(i)$,
$\gamma \leftarrow \Theta^{i+1}_{p@p1}$,
$p \leftarrow p@p1$,
$q \leftarrow p@p1$,
$X \leftarrow \lfloor \beta' + 2 * \Lambda' \rfloor\}$,
wpred_ax,
read_self $\{p \leftarrow p@p1\}$,
good_ReadClock $\{p \leftarrow p@p1\}$,
wpred_fixtime $\{p \leftarrow p@p1\}$,
okay_Readpred_floor
$\{$ppred $\leftarrow$ wpred$(i)$,
$\gamma \leftarrow \gamma$@p3,
$y \leftarrow \beta' + 2 * \Lambda'\}$

abshack4: **Lemma** $a - b \geq c - d$
$\supset |(a-b)-(c-d)| \leq |(a - \lfloor b \rfloor) - (c - \lceil d \rceil)|$

floor_hack: **Lemma** $a - \lfloor b \rfloor \geq a - b$

floor_hack_pr: **Prove** floor_hack **from** floor_defn $\{x \leftarrow b\}$

ceil_hack: **Lemma** $c - d \geq c - \lceil d \rceil$

ceil_hack_pr: **Prove** ceil_hack **from** ceil_defn $\{x \leftarrow d\}$

abshack4_pr: **Prove** abshack4 **from**
abs_ge0 $\{x \leftarrow (a-b)-(c-d)\}$,
abs_ge0 $\{x \leftarrow (a - \lfloor b \rfloor) - (c - \lceil d \rceil)\}$,
floor_hack,
ceil_hack

$X$: **Var** Clocktime

ADJ_hack: **Lemma** wpred$(i)(p)$
$\supset ADJ^i_p - X = cfn(p, (\lambda\, p_1 : \Theta^{i+1}_p(p_1) - IC^i_p(t^{i+1}_p) - X))$

ADJ_hack_pr: **Prove** ADJ_hack **from**
ADJ_lem1,
translation_invariance
$\{\gamma \leftarrow (\lambda\, p_1 \rightarrow$ Clocktime $: \Theta^{i+1}_p(p_1) - IC^i_p(t^{i+1}_p))$,
$X \leftarrow -X\}$,
wpred_fixtime

delay_prec_enh_step1_sym_pr: **Prove** delay_prec_enh_step1_sym **from**
ADJ_hack $\{X \leftarrow \lfloor s_p^i \rfloor\}$,
ADJ_hack $\{p \leftarrow q,\ X \leftarrow \lceil s_q^i \rceil\}$,
abshack4 $\{a \leftarrow ADJ_p^i,\ b \leftarrow s_p^i,\ c \leftarrow ADJ_q^i,\ d \leftarrow s_q^i\}$

abshack5: **Lemma** $|((a-b)-(\lfloor c \rfloor - d)) - ((e-f)-(\lceil g \rceil - d))|$
$\leq |(a-b)-(\lfloor c \rfloor - d)| + |(e-f)-(\lceil g \rceil - d)|$

abshack5_pr: **Prove** abshack5 **from**
abs_com $\{x \leftarrow e-f,\ y \leftarrow \lceil g \rceil - d\}$,
abs_plus $\{x \leftarrow (a-b)-(\lfloor c \rfloor - d),\ y \leftarrow (\lceil g \rceil - d)-(e-f)\}$

absfloor: **Lemma** $|a - \lfloor b \rfloor| \leq |a-b| + 1$

absceil: **Lemma** $|a - \lceil b \rceil| \leq |a-b| + 1$

absfloor_pr: **Prove** absfloor **from**
floor_defn $\{x \leftarrow b\}$, $|\star 1|$ $\{x \leftarrow a - \lfloor b \rfloor\}$, $|\star 1|$ $\{x \leftarrow a-b\}$

absceil_pr: **Prove** absceil **from**
ceil_defn $\{x \leftarrow b\}$, $|\star 1|$ $\{x \leftarrow a - \lceil b \rceil\}$, $|\star 1|$ $\{x \leftarrow a-b\}$

abshack6a: **Lemma** $|(a-b)-(\lfloor c \rfloor - d)| \leq |(a-b)-(c-d)| + 1$

abshack6b: **Lemma** $|(e-f)-(\lceil g \rceil - d)| \leq |(e-f)-(g-d)| + 1$

abshack6a_pr: **Prove** abshack6a **from**
absfloor $\{a \leftarrow (a-b)+d,\ b \leftarrow c\}$,
abs_plus $\{x \leftarrow (a-b)-(c-d),\ y \leftarrow 1\}$,
abs_ge0 $\{x \leftarrow 1\}$

abshack6b_pr: **Prove** abshack6b **from**
absceil $\{a \leftarrow (e-f)+d,\ b \leftarrow g\}$,
abs_plus $\{x \leftarrow (e-f)-(g-d),\ y \leftarrow 1\}$,
abs_ge0 $\{x \leftarrow 1\}$

abshack7: **Lemma** $|(a-b)-(c-d)| \leq h \wedge |(e-f)-(g-d)| \leq h$
$\supset |((a-b)-(\lfloor c \rfloor - d))-((e-f)-(\lceil g \rceil - d))| \leq 2*(h+1)$

abshack7_pr: **Prove** abshack7 **from** abshack5, abshack6a, abshack6b

prec_enh_hyp1_pr: **Prove** prec_enh_hyp1 **from**
okay_pairs
$\{\gamma \leftarrow (\lambda\, p_1 : \Theta_p^{i+1}(p_1) - IC_p^i(t_p^{i+1}) - \lfloor s_p^i \rfloor)$,
$\theta \leftarrow (\lambda\, p_1 : \Theta_q^{i+1}(p_1) - IC_q^i(t_q^{i+1}) - \lceil s_q^i \rceil)$,
$x \leftarrow 2 * (\Lambda' + 1)$,
ppred $\leftarrow$ wpred$(i)\}$,
delay_pred $\{q \leftarrow p_3$@p1$\}$,
delay_pred $\{p \leftarrow q,\ q \leftarrow p_3$@p1$\}$,
reading_error3 $\{q \leftarrow p_3$@p1$\}$,
reading_error3 $\{p \leftarrow q,\ q \leftarrow p_3$@p1$\}$,
good_read_pred_ax1 $\{q \leftarrow p_3$@p1$\}$,
good_read_pred_ax1 $\{p \leftarrow q,\ q \leftarrow p_3$@p1$\}$,
abshack7
$\{a \leftarrow \Theta_p^{i+1}(p_3$@p1$)$,
$b \leftarrow IC_p^i(t_p^{i+1})$,
$c \leftarrow s_p^i$,
$d \leftarrow s^i_{p_3}$@p1',
$e \leftarrow \Theta_q^{i+1}(p_3$@p1$)$,
$f \leftarrow IC_q^i(t_q^{i+1})$,
$g \leftarrow s_q^i$,
$h \leftarrow \Lambda'\}$,
wpred_fixtime,
wpred_fixtime $\{p \leftarrow q\}$,
wpred_fixtime $\{p \leftarrow p_3$@p1$\}$,
betaread_ax

abshack3: **Lemma** $|(a-b)-(c-d)| = |(c-a)-(d-b)|$

abshack3_pr: **Prove** abshack3 **from** abs_com $\{x \leftarrow a-b,\ y \leftarrow c-d\}$

delay_prec_enh_pr: **Prove** delay_prec_enh **from**
delay_prec_enh_step1,
delay_prec_enh_step1 $\{p \leftarrow q,\ q \leftarrow p\}$,
delay_prec_enh_step1_sym,
delay_prec_enh_step1_sym $\{p \leftarrow q,\ q \leftarrow p\}$,
abs_com $\{x \leftarrow ADJ_p^i - s_p^i,\ y \leftarrow ADJ_q^i - s_q^i\}$,
abshack3 $\{a \leftarrow s_p^i,\ b \leftarrow s_q^i,\ c \leftarrow ADJ_p^i,\ d \leftarrow ADJ_q^i\}$

**End** delay2

## B.4 delay3

delay3: **Module**

**Using** arith, clockassumptions, delay2

**Exporting all with** clockassumptions, delay2

**Theory**

$p, q, p_1, q_1$: **Var** process
$i$: **Var** event
$T$: **Var** Clocktime
good_interval: function[process, event, Clocktime → bool] =
$(\lambda\, p, i, T : (\text{correct\_during}(p, s_p^i, ic_p^{i+1}(T)) \wedge T - ADJ_p^i \geq S^i)$
$\vee\, (\text{correct\_during}(p, ic_p^{i+1}(T), s_p^i) \wedge S^i \geq T - ADJ_p^i))$

recovery_lemma: **Axiom**
$\text{delay\_pred}(i) \wedge \text{ADJ\_pred}(i+1)$
$\wedge\, \text{rpred}(i)(p) \wedge \text{correct\_during}(p, t_p^{i+1}, t_p^{i+2}) \wedge \text{wpred}(i+1)(q)$
$\supset |s_p^{i+1} - s_q^{i+1}| \leq \beta'$

good_interval_lem: **Lemma**
$\text{wpred}(i)(p) \wedge \text{wpred}(i+1)(p) \wedge \text{ADJ\_pred}(i+1) \supset \text{good\_interval}(p, i, S^{i+1})$

betaprime_ax: **Axiom**
$4 * \rho * (R + \boldsymbol{\alpha}(\lfloor \beta' + 2 * \Lambda' \rfloor)) + \boldsymbol{\pi}(\lfloor 2 * (\Lambda' + 1) \rfloor, \lfloor \beta' + 2 * \Lambda' \rfloor) \leq \beta'$

betaprime_ind_lem: **Lemma**
$\text{ADJ\_pred}(i+1) \wedge \text{wpred}(i)(p)$
$\supset 2 * \rho * (R + \boldsymbol{\alpha}(\lfloor \beta' + 2 * \Lambda' \rfloor)) + \boldsymbol{\pi}(\lfloor 2 * (\Lambda' + 1) \rfloor, \lfloor \beta' + 2 * \Lambda' \rfloor) \leq \beta'$

betaprime_lem: **Lemma**
$2 * \rho * (R + \boldsymbol{\alpha}(\lfloor \beta' + 2 * \Lambda' \rfloor)) + \boldsymbol{\pi}(\lfloor 2 * (\Lambda' + 1) \rfloor, \lfloor \beta' + 2 * \Lambda' \rfloor) \leq \beta'$

R_0_lem: **Lemma** $\text{wpred}(i)(p) \wedge \text{ADJ\_pred}(i+1) \supset R > 0$

bound_future: **Lemma**
$\text{delay\_pred}(i) \wedge \text{ADJ\_pred}(i+1)$
$\wedge\, \text{wpred}(i)(p)$
$\wedge\, \text{wpred}(i)(q) \wedge \text{good\_interval}(p, i, T) \wedge \text{good\_interval}(q, i, T)$
$\supset |ic_p^{i+1}(T) - ic_q^{i+1}(T)|$
$\leq 2 * \rho * (|T - S^i| + \boldsymbol{\alpha}(\lfloor \beta' + 2 * \Lambda' \rfloor))$
$+\, \boldsymbol{\pi}(\lfloor 2 * (\Lambda' + 1) \rfloor, \lfloor \beta' + 2 * \Lambda' \rfloor)$

bound_future1: **Lemma**
$\text{delay\_pred}(i) \wedge \text{ADJ\_pred}(i+1) \wedge \text{wpred}(i)(p) \wedge \text{good\_interval}(p, i, T)$
$\supset |(ic_p^i(T - ADJ_p^i) - s_p^i) - (T - ADJ_p^i - S^i)|$
$\leq \rho * (|T - S^i| + \boldsymbol{\alpha}(\lfloor \beta' + 2 * \Lambda' \rfloor))$

bound_future1_step: **Lemma**
delay_pred$(i)$ $\wedge$ ADJ_pred$(i+1)$ $\wedge$ wpred$(i)(p)$ $\wedge$ good_interval$(p,i,T)$
$\supset |(ic_p^i(T - ADJ_p^i) - s_p^i) - (T - ADJ_p^i - S^i)| \leq \rho \star (|T - ADJ_p^i - S^i|)$

bound_FIXTIME: **Lemma**
delay_pred$(i)$ $\wedge$ ADJ_pred$(i+1)$
$\wedge$ wpred$(i)(p)$
$\wedge$ wpred$(i)(q)$
$\wedge$ good_interval$(p,i,S^{i+1})$ $\wedge$ good_interval$(q,i,S^{i+1})$
$\supset |s_p^{i+1} - s_q^{i+1}| \leq \beta'$

bound_FIXTIME2: **Lemma**
delay_pred$(i)$ $\wedge$ ADJ_pred$(i+1)$ $\wedge$ wpred$(i)(p)$ $\wedge$ wpred$(i)(q)$
$\supset$ (wpred$(i+1)(p)$ $\wedge$ wpred$(i+1)(q)$ $\supset |s_p^{i+1} - s_q^{i+1}| \leq \beta'$)

delay_offset: **Lemma** wpred$(i)(p)$ $\wedge$ wpred$(i)(q)$ $\supset |s_p^i - s_q^i| \leq \beta'$

ADJ_bound: **Lemma** wpred$(i)(p)$ $\supset |ADJ_p^i| \leq \alpha(\lfloor \beta' + 2 * \Lambda' \rfloor)$

Alpha_0: **Lemma** wpred$(i)(p)$ $\supset \alpha(\lfloor \beta' + 2 * \Lambda' \rfloor) \geq 0$

## Proof

ADJ_pred_lr: **Lemma**
ADJ_pred$(i+1)$ $\supset$ (wpred$(i)(p)$ $\supset |ADJ_p^i| \leq \alpha(\lfloor \beta' + 2 * \Lambda' \rfloor)$)

ADJ_pred_lr_pr: **Prove** ADJ_pred_lr **from** ADJ_pred $\{i \leftarrow i+1\}$

betaprime_ind_lem_pr: **Prove** betaprime_ind_lem **from**
betaprime_ax,
pos_product $\{x \leftarrow \rho,\ y \leftarrow R + \alpha(\lfloor \beta' + 2 * \Lambda' \rfloor)\}$,
rho_0,
R_FIX_SYNC_0,
FIX_SYNC,
ADJ_pred_lr,
$|\star 1|$ $\{x \leftarrow ADJ_p^i\}$

betaprime_lem_pr: **Prove** betaprime_lem **from**
betaprime_ind_lem $\{p \leftarrow p@p4\}$,
bnd_delay_offset $\{i \leftarrow i+1\}$,
wpred_ax,
count_exists {ppred $\leftarrow$ wpred$(i@p1)$, $n \leftarrow N\}$,
N_maxfaults

delay_offset_pr: **Prove** delay_offset **from** bnd_delay_offset, delay_pred

ADJ_bound_pr: **Prove** ADJ_bound **from**
bnd_delay_offset $\{i \leftarrow i+1\}$, ADJ_pred $\{i \leftarrow i+1\}$

$a_1, b_1, c_1, d_1$: **Var** number

abs_0: **Lemma** $|a_1| \leq b_1 \supset b_1 \geq 0$

abs_0_pr: **Prove** abs_0 **from** $|\star 1|$ $\{x \leftarrow a_1\}$

Alpha_0_pr: **Prove** Alpha_0 **from** ADJ_bound, $|\star 1|$ $\{x \leftarrow ADJ_p^i\}$

R_0_hack: **Lemma** wpred$(i)(p) \wedge$ ADJ_pred$(i+1) \supset S^{i+1} - S^i > 0$

R_0_hack_pr: **Prove** R_0_hack **from**
ADJ_pred $\{i \leftarrow i+1\}$,
FIXTIME_bound,
wpred_hi_lem,
abs_0 $\{a_1 \leftarrow ADJ_p^i,\ b_1 \leftarrow \boldsymbol{\alpha}(\lfloor \beta' + 2 * \Lambda' \rfloor)\}$

R_0_lem_pr: **Prove** R_0_lem **from** R_0_hack, $S^{\star 1}$, $S^{\star 1}$ $\{i \leftarrow i+1\}$

abshack_future: **Lemma** $|(a_1 - b_1) - (c_1 - d_1)| = |(a_1 - c_1) - (b_1 - d_1)|$

abshack_future_pr: **Prove** abshack_future

abs_minus: **Lemma** $|a_1 - b_1| \leq |a_1| + |b_1|$

abs_minus_pr: **Prove** abs_minus **from**
$|\star 1|$ $\{x \leftarrow a_1 - b_1\}$, $|\star 1|$ $\{x \leftarrow a_1\}$, $|\star 1|$ $\{x \leftarrow b_1\}$

bound_future1_pr: **Prove** bound_future1 **from**
bound_future1_step,
abs_minus $\{a_1 \leftarrow T - S^i,\ b_1 \leftarrow ADJ_p^i\}$,
ADJ_pred $\{i \leftarrow i+1\}$,
mult_leq_2
$\{z \leftarrow \rho,$
$y \leftarrow |T - ADJ_p^i - S^i|,$
$x \leftarrow |T - S^i| + \boldsymbol{\alpha}(\lfloor \beta' + 2 * \Lambda' \rfloor)\}$,
rho_0

bound_future1_step_a: **Lemma**
correct_during$(p, ic_p^i(T - ADJ_p^i), s_p^i) \wedge S^i \geq T - ADJ_p^i$
$\supset |(ic_p^i(T - ADJ_p^i) - s_p^i) - (T - ADJ_p^i - S^i)| \leq \rho \star (|T - ADJ_p^i - S^i|)$

bound_future1_step_b: **Lemma**
correct_during$(p, s_p^i, ic_p^i(T - ADJ_p^i)) \wedge T - ADJ_p^i \geq S^i$
$\supset |(ic_p^i(T - ADJ_p^i) - s_p^i) - (T - ADJ_p^i - S^i)| \leq \rho \star (|T - ADJ_p^i - S^i|)$

bound_future1_step_a_pr: **Prove** bound_future1_step_a **from**
RATE_lemma2_iclock $\{T \leftarrow T - ADJ_p^i,\ S \leftarrow S^i\}$,
$s_{\star 1}^{\star 2}$ ,
abshack_future
$\{a_1 \leftarrow ic_p^i(T - ADJ_p^i)$,
$b_1 \leftarrow s_p^i$,
$c_1 \leftarrow T - ADJ_p^i$,
$d_1 \leftarrow S^i\}$,
abs_com $\{x \leftarrow a_1@p3 - c_1@p3,\ y \leftarrow b_1@p3 - d_1@p3\}$,
abs_com $\{x \leftarrow T@p1,\ y \leftarrow S@p1\}$

bound_future1_step_b_pr: **Prove** bound_future1_step_b **from**
RATE_lemma2_iclock $\{S \leftarrow T - ADJ_p^i,\ T \leftarrow S^i\}$,
$s_{\star 1}^{\star 2}$ ,
abshack_future
$\{a_1 \leftarrow ic_p^i(T - ADJ_p^i)$,
$b_1 \leftarrow s_p^i$,
$c_1 \leftarrow T - ADJ_p^i$,
$d_1 \leftarrow S^i\}$

bound_future1_step_pr: **Prove** bound_future1_step **from**
good_interval, bound_future1_step_a, bound_future1_step_b, iclock_ADJ_lem

good_interval_lem_pr: **Prove** good_interval_lem **from**
good_interval $\{T \leftarrow S^{i+1}\}$,
$s_{\star 1}^{\star 2}$ $\{i \leftarrow i + 1\}$,
wpred_fixtime,
wpred_fixtime_low $\{i \leftarrow i + 1\}$,
correct_during_trans $\{t \leftarrow s_p^i,\ t_2 \leftarrow t_p^{i+1},\ s \leftarrow s_p^{i+1}\}$,
wpred_hi_lem,
FIXTIME_bound,
ADJ_pred $\{i \leftarrow i + 1\}$,
$|\star 1|$ $\{x \leftarrow ADJ_p^i\}$

bound_FIXTIME2_pr: **Prove** bound_FIXTIME2 **from**
bound_FIXTIME, good_interval_lem, good_interval_lem $\{p \leftarrow q\}$

bound_FIXTIME_pr: **Prove** bound_FIXTIME **from**
bound_future $\{T \leftarrow S^{i+1}\}$,
$S^{\star 1}$ ,
$S^{\star 1}$ $\{i \leftarrow i + 1\}$,
abs_ge0 $\{x \leftarrow R\}$,
R_0_lem,
$s_{\star 1}^{\star 2}$ $\{p \leftarrow p@p1,\ i \leftarrow i + 1\}$,
$s_{\star 1}^{\star 2}$ $\{p \leftarrow q@p1,\ i \leftarrow i + 1\}$,
betaprime_ind_lem

bnd_delay_offset_ind_b_pr: **Prove** bnd_delay_offset_ind_b **from**
  bound_FIXTIME2 $\{p \leftarrow p@p2,\ q \leftarrow q@p2\}$,
  delay_pred $\{i \leftarrow i+1\}$,
  delay_pred $\{p \leftarrow p@p2,\ q \leftarrow q@p2\}$,
  recovery_lemma $\{p \leftarrow p@p2,\ q \leftarrow q@p2\}$,
  recovery_lemma $\{p \leftarrow q@p2,\ q \leftarrow p@p2\}$,
  abs_com $\{x \leftarrow s^{i+1}_{p@p2},\ y \leftarrow s^{i+1}_{q@p2}\}$,
  wpred_preceding $\{p \leftarrow p@p2\}$,
  wpred_preceding $\{p \leftarrow q@p2\}$,
  wpred_correct $\{i \leftarrow i+1,\ p \leftarrow p@p2\}$,
  wpred_correct $\{i \leftarrow i+1,\ p \leftarrow q@p2\}$

$a, b, c, d, e, f, g, h$, aa, bb: **Var** number

abshack: **Lemma** $|a-b|$
  $\leq |(a-e)-(c-f-d)| + |(b-g)-(c-h-d)|$
  $+ |(e-g)-(f-h)|$

abshack2: **Lemma** $|(a-e)-(c-f-d)| \leq$ aa
  $\wedge\ |(b-g)-(c-h-d)| \leq$ aa $\wedge\ |(e-g)-(f-h)| \leq$ bb
  $\supset |a-b| \leq 2 * \text{aa} + \text{bb}$

abshack2_pr: **Prove** abshack2 **from** abshack

abshack_pr: **Prove** abshack **from**
  abs_com $\{x \leftarrow b-g,\ y \leftarrow c-h-d\}$,
  abs_plus $\{x \leftarrow (a-e)-(c-f-d),\ y \leftarrow (c-h-d)-(b-g)\}$,
  abs_plus $\{x \leftarrow x@p2 + y@p2,\ y \leftarrow (e-g)-(f-h)\}$

bound_future_pr: **Prove** bound_future **from**
  bound_future1,
  bound_future1 $\{p \leftarrow q\}$,
  delay_prec_enh,
  iclock_ADJ_lem,
  iclock_ADJ_lem $\{p \leftarrow q\}$,
  abshack2
    $\{a \leftarrow ic^i_p(T - ADJ^i_p)$,
     $b \leftarrow ic^i_q(T - ADJ^i_q)$,
     $c \leftarrow T$,
     $d \leftarrow S^i$,
     $e \leftarrow s^i_p$,
     $f \leftarrow ADJ^i_p$,
     $g \leftarrow s^i_q$,
     $h \leftarrow ADJ^i_q$,
     aa $\leftarrow \rho * (|T - S^i| + \alpha(\lfloor \beta' + 2 * \Lambda' \rfloor))$,
     bb $\leftarrow \pi(\lfloor 2 * (\Lambda' + 1) \rfloor, \lfloor \beta' + 2 * \Lambda' \rfloor)\}$

**End** delay3

## B.5 delay4

delay4: **Module**

**Using** arith, clockassumptions, delay3

**Exporting all with** clockassumptions, delay3

**Theory**

$p, q, p_1, q_1$: **Var** process
$i$: **Var** event
$X, S, T$: **Var** Clocktime
$s, t, t_1, t_2$: **Var** time
$\gamma$: **Var** function[process $\rightarrow$ Clocktime]
ppred, ppred1: **Var** function[process $\rightarrow$ bool]
option1, option2: bool

option1_defn: **Axiom**
$\text{option1} \supset T_p^{i+1} = (i+1) * R + T^0 \wedge (\beta = 2 * \rho \star (R - (S^0 - T^0)) + \beta')$

option2_defn: **Axiom**
$\text{option2} \supset T_p^{i+1} = (i+1) * R + T^0 - ADJ_p^i$
$\wedge (\beta = \beta' - 2 * \rho \star (S^0 - T^0))$

options_disjoint: **Axiom** $\neg(\text{option1} \wedge \text{option2})$

option1_bounded_delay: **Lemma**
$\text{option1} \wedge \text{wpred}(i)(p) \wedge \text{wpred}(i)(q) \supset |t_p^{i+1} - t_q^{i+1}| \leq \beta$

option2_bounded_delay: **Lemma**
$\text{option2} \wedge \text{wpred}(i)(p) \wedge \text{wpred}(i)(q) \supset |t_p^{i+1} - t_q^{i+1}| \leq \beta$

option1_bounded_delay0: **Lemma**
$\text{option1} \wedge \text{wpred}(0)(p) \wedge \text{wpred}(0)(q) \supset |t_p^0 - t_q^0| \leq \beta$

option2_bounded_delay0: **Lemma**
$\text{option2} \wedge \text{wpred}(0)(p) \wedge \text{wpred}(0)(q) \supset |t_p^0 - t_q^0| \leq \beta$

option2_convert_lemma: **Lemma**
$(\beta = \beta' - 2 * \rho \star (S^0 - T^0))$
$\supset 2 * \rho \star ((R - (S^0 - T^0)) + \boldsymbol{\alpha}(\lfloor \beta' + 2 * \Lambda' \rfloor))$
$+ \boldsymbol{\pi}(\lfloor 2 * (\Lambda' + 1) \rfloor, \lfloor \beta' + 2 * \Lambda' \rfloor)$
$\leq \beta$

option2_good_interval: **Lemma**
$\text{option2} \wedge \text{wpred}(i)(p) \supset \text{good\_interval}(p, i, (i+1) * R + T^0)$

options_exhausted: **Axiom** option1 $\vee$ option2

## Proof

rts_2_hi_pr: **Prove** rts_2_hi **from**
options_exhausted, option1_bounded_delay, option2_bounded_delay

option1_bounded_delay0_pr: **Prove** option1_bounded_delay0 **from**
bnd_delay_init,
option1_defn,
pos_product $\{x \leftarrow \rho,\ y \leftarrow S^0 - T^0\}$,
pos_product $\{x \leftarrow \rho,\ y \leftarrow R - (S^0 - T^0)\}$,
R_FIX_SYNC_0,
FIX_SYNC,
rho_0

option2_bounded_delay0_pr: **Prove** option2_bounded_delay0 **from**
bnd_delay_init, option2_defn

option1_bounded_delay_pr: **Prove** option1_bounded_delay **from**
RATE_lemma1_iclock $\{S \leftarrow (i+1) * R + T^0,\ T \leftarrow S^i\}$,
$S^{\star 1}$ ,
delay_offset,
wpred_fixtime,
wpred_fixtime $\{p \leftarrow q\}$,
synctime_defn,
synctime_defn $\{p \leftarrow q\}$,
$s^{\star 2}_{\star 1}$ ,
$s^{\star 2}_{\star 1}$ $\{p \leftarrow q\}$,
option1_defn,
option1_defn $\{p \leftarrow q\}$,
R_FIX_SYNC_0,
option1_defn

option2_good_interval_pr: **Prove** option2_good_interval **from**
good_interval $\{T \leftarrow T_p^{i+1} + ADJ_p^i\}$,
wpred_fixtime,
wpred_hi_lem,
rts_new_1,
iclock_ADJ_lem $\{T \leftarrow T@p1\}$,
synctime_defn,
Alpha_0,
option2_defn

option2_convert_lemma_pr: **Prove** option2_convert_lemma **from**
betaprime_lem,
mult_ldistrib_minus
$\{x \leftarrow \rho$,
$y \leftarrow R + \alpha(\lfloor \beta' + 2 * \Lambda' \rfloor)$,
$z \leftarrow (S^0 - T^0)\}$

option2_bounded_delay_pr: **Prove** option2_bounded_delay **from**
option2_convert_lemma,
option2_good_interval,
option2_good_interval $\{p \leftarrow q\}$,
bound_future $\{T \leftarrow (i+1) * R + T^0\}$,
option2_defn,
option2_defn $\{p \leftarrow q\}$,
iclock_ADJ_lem $\{T \leftarrow T@p4\}$,
iclock_ADJ_lem $\{T \leftarrow T@p4,\ p \leftarrow q\}$,
synctime_defn,
synctime_defn $\{p \leftarrow q\}$,
$S^{\star 1}$,
R_0_lem,
bnd_delay_offset,
bnd_delay_offset $\{i \leftarrow i+1\}$,
abs_ge0 $\{x \leftarrow (R - (S^0 - T^0))\}$,
R_FIX_SYNC_0,
option2_defn

**End** delay4

## B.6 new_basics

new_basics: **Module**

**Using** clockassumptions, arith, delay3

**Exporting all with** clockassumptions, delay3

**Theory**

$p, q$: **Var** process
$i, j, k$: **Var** event
$x, y, y_1, y_2, z$: **Var** number
$r, s, t, t_1, t_2$: **Var** time
$X, Y$: **Var** Clocktime
$(\star 1 \Uparrow \star 2)[\star 3]$: **Definition** function[process, process, event $\rightarrow$ process] =
$(\lambda\, p, q, i : (\ \textbf{if}\ t_p^i \geq t_q^i\ \textbf{then}\ p\ \textbf{else}\ q\ \textbf{end if}))$

maxsync_correct: **Lemma** correct$(p, s) \wedge$ correct$(q, s) \supset$ correct$((p \Uparrow q)[i], s)$

minsync: **Definition** function[process, process, event $\rightarrow$ process] =
$(\lambda\, p, q, i : (\ \textbf{if}\ t_p^i \geq t_q^i\ \textbf{then}\ q\ \textbf{else}\ p\ \textbf{end if}))$

minsync_correct: **Lemma** correct$(p, s) \wedge$ correct$(q, s) \supset$ correct$((p \Downarrow q)[i], s)$

minsync_maxsync: **Lemma** $t^i_{(p \Downarrow q)[i]} \leq t^i_{(p \Uparrow q)[i]}$

$t^{\star 3}_{\star 1, \star 2}$: **Definition** function[process, process, event $\rightarrow$ time] =
$(\lambda\, p, q, i : t^i_{(p \Uparrow q)[i]})$

delay_recovery: **Axiom**
rpred$(i)(p) \wedge$ wvr_pred$(i)(q) \supset |t_p^{i+1} - t_q^{i+1}| \leq \beta$

rts0_new: **Axiom** wpred$(i)(p)$
$\supset t_p^{i+1} - t_p^i \leq (1 + \rho) \star (R + \boldsymbol{\alpha}(\lfloor \beta' + 2 * \Lambda' \rfloor))$

rts1_new: **Axiom** wpred$(i)(p)$
$\supset ((R - \boldsymbol{\alpha}(\lfloor \beta' + 2 * \Lambda' \rfloor))/(1 + \rho)) \leq t_p^{i+1} - t_p^i$

nonoverlap: **Axiom** $\beta < ((R - \boldsymbol{\alpha}(\lfloor \beta' + 2 * \Lambda' \rfloor))/(1 + \rho))$

lemma_1: **Lemma** wpred$(i)(p) \wedge$ wpred$(i)(q) \supset t_p^i < t_q^{i+1}$

lemma_1_1: **Lemma** wpred$(i)(p) \wedge$ wpred$(i + 1)(q) \supset t_p^i < t_q^{i+1}$

lemma_1_2: **Lemma** wpred$(i)(p) \wedge$ wpred$(i + 1)(q) \supset t_p^{i+1} < t_q^{i+2}$

lemma_2_1: **Lemma** correct$(q, t_q^{i+1})$
$\supset IC_q^{i+1}(t_q^{i+1}) = cfn(q, \Theta_q^{i+1})$

rts_2_lo_i: **Lemma**
wpred$(i+1)(p) \wedge$ wpred$(i+1)(q) \supset |t_p^{i+1} - t_q^{i+1}| \leq \beta$

rts_2_lo_i_recover: **Lemma**
rpred$(i)(p) \wedge$ wpred$(i+1)(q) \supset |t_p^{i+1} - t_q^{i+1}| \leq \beta$

synctime_monotonic: **Axiom** $i \leq j \supset t_q^i \leq t_q^j$

working_clocks_lo: **Lemma**
wpred$(i+1)(p) \wedge t_p^{i+1} \leq t \wedge$ wpred$(i)(q) \supset t_q^i < t$

working_clocks_hi: **Lemma**
wpred$(i)(p) \wedge t < t_p^{i+1} \wedge$ wpred$(i+1)(q) \supset t < t_q^{i+2}$

working_clocks_interval: **Lemma**
$i > 0 \wedge$ wpred$(i)(p)$
$\quad \wedge$ wpred$(j)(q) \wedge t_p^i \leq t \wedge t < t_p^{i+1} \wedge t_q^j \leq t \wedge t < t_q^{j+1}$
$\supset t_q^{i-1} < t_q^{j+1} \wedge t_q^j < t_q^{i+2}$

## Proof

working_clocks_lo_pr: **Prove** working_clocks_lo **from**
lemma_1_1 $\{p \leftarrow q,\ q \leftarrow p\}$

working_clocks_hi_pr: **Prove** working_clocks_hi **from** lemma_1_2

rts_2_lo_i_recover_pr: **Prove** rts_2_lo_i_recover **from**
delay_recovery, wpred_preceding $\{p \leftarrow q\}$, wvr_pred $\{p \leftarrow q\}$

rts_2_lo_i_pr: **Prove** rts_2_lo_i **from**
rts_2_lo_i_recover,
rts_2_lo_i_recover $\{p \leftarrow q,\ q \leftarrow p\}$,
abs_com $\{x \leftarrow t_p^{i+1},\ y \leftarrow t_q^{i+1}\}$,
rts_2_hi,
wpred_preceding,
wpred_preceding $\{p \leftarrow q\}$

rts_2_lo_pr: **Prove** rts_2_lo **from** rts_2_lo_i $\{i \leftarrow$ pred$(i)\}$, bnd_delay_init

maxsync_correct_pr: **Prove** maxsync_correct **from** $(\star 1 \Uparrow \star 2)[\star 3]$

minsync_correct_pr: **Prove** minsync_correct **from** minsync

minsync_maxsync_pr: **Prove** minsync_maxsync **from** minsync, $(\star 1 \Uparrow \star 2)[\star 3]$

lemma_1_proof: **Prove** lemma_1 **from**
rts_2_hi, rts1_new, $|\star 1|$ $\{x \leftarrow t_p^{i+1} - t_q^{i+1}\}$, nonoverlap



C.2

lemma_2_1_proof: **Prove** lemma_2_1 **from**
lClock_defn $\{p \leftarrow q,\ i \leftarrow i+1,\ t \leftarrow t_q^{i+1}\}$,
$adj_{\star 1}^{\star 2}$ $\{i \leftarrow i+1,\ p \leftarrow q\}$

lemma_1_1_proof: **Prove** lemma_1_1 **from**
rts_2_hi,
wpred_preceding $\{p \leftarrow q\}$,
delay_recovery $\{p \leftarrow q,\ q \leftarrow p\}$,
abs_com $\{x \leftarrow t_p^{i+1},\ y \leftarrow t_q^{i+1}\}$,
wvr_pred,
$|\star 1|$ $\{x \leftarrow t_p^{i+1} - t_q^{i+1}\}$,
rts1_new,
nonoverlap

lemma_1_2_proof: **Prove** lemma_1_2 **from**
rts_2_hi,
wpred_preceding $\{p \leftarrow q\}$,
delay_recovery $\{p \leftarrow q,\ q \leftarrow p\}$,
abs_com $\{x \leftarrow t_p^{i+1},\ y \leftarrow t_q^{i+1}\}$,
wvr_pred,
$|\star 1|$ $\{x \leftarrow t_p^{i+1} - t_q^{i+1}\}$,
rts1_new $\{p \leftarrow q,\ i \leftarrow i+1\}$,
nonoverlap

**End** new_basics

## B.7 rmax_rmin

rmax_rmin: **Module**

**Using** clockassumptions, arith, delay4, new_basics

**Exporting all with** clockassumptions, delay4

**Theory**

$p, q$: **Var** process
$i, j, k$: **Var** event
$x, y, y_1, y_2, z$: **Var** number
$r, s, t, t_1, t_2$: **Var** time
$X, Y$: **Var** Clocktime
rmax_pred: function[process, event → bool] =
  $(\lambda\, p, i$ : wpred$(i)(p)$
    $\supset t_p^{i+1} - t_p^i \leq (1+\rho) * (R + \boldsymbol{\alpha}(\lfloor \beta' + 2 * \Lambda' \rfloor)))$
rmin_pred: function[process, event → bool] =
  $(\lambda\, p, i$ : wpred$(i)(p)$
    $\supset ((R - \boldsymbol{\alpha}(\lfloor \beta' + 2 * \Lambda' \rfloor))/(1+\rho)) \leq t_p^{i+1} - t_p^i)$

ADJ_recovery: **Axiom** option1 $\wedge$ rpred$(i)(p) \supset |ADJ_p^i| \leq \boldsymbol{\alpha}(\lfloor \beta' + 2 * \Lambda' \rfloor)$

rmax1: **Lemma** option1 $\supset$ rmax_pred$(p, i)$

rmax2: **Lemma** option2 $\supset$ rmax_pred$(p, i)$

rmin1: **Lemma** option1 $\supset$ rmin_pred$(p, i)$

rmin2: **Lemma** option2 $\supset$ rmin_pred$(p, i)$

**Proof**

rts0_new_pr: **Prove** rts0_new **from** options_exhausted, rmax1, rmax2, rmax_pred

rts1_new_pr: **Prove** rts1_new **from** options_exhausted, rmin1, rmin2, rmin_pred

rmin2_0: **Lemma** option2 $\supset$ rmin_pred$(p, 0)$

rmin2_plus: **Lemma** option2 $\supset$ rmin_pred$(p, i+1)$

rmin2_pr: **Prove** rmin2 **from** rmin2_0, rmin2_plus $\{i \leftarrow \text{pred}(i)\}$

rmin2_0_pr: **Prove** rmin2_0 **from**
rmin_pred $\{i \leftarrow 0\}$,
synctime0_defn,
synctime_defn $\{i \leftarrow i@p1\}$,
option2_defn $\{i \leftarrow i@p1\}$,
R_0,
RATE_2_iclock $\{i \leftarrow i@p1,\ S \leftarrow T_p^{i@p1+1},\ T \leftarrow T^0\}$,
wpred_correct $\{i \leftarrow i@p1\}$,
div_ineq
$\{z \leftarrow (1+\rho)$,
$y \leftarrow R - ADJ_p^{i@p1}$,
$x \leftarrow R - \alpha(\lfloor \beta' + 2 * \Lambda' \rfloor)\}$,
rho_0,
ADJ_bound $\{i \leftarrow i@p1\}$,
$|\star 1|$ $\{x \leftarrow ADJ_p^{i@p1}\}$,
R_bound $\{i \leftarrow i@p1\}$,
wpred_hi_lem $\{i \leftarrow i@p1\}$,
Alpha_0 $\{i \leftarrow i@p1\}$

rmin2_plus_pr: **Prove** rmin2_plus **from**
rmin_pred $\{i \leftarrow i+1\}$,
synctime_defn,
synctime_defn $\{i \leftarrow i@p1\}$,
option2_defn $\{i \leftarrow i\}$,
option2_defn $\{i \leftarrow i@p1\}$,
R_0,
RATE_2_iclock
$\{i \leftarrow i@p1$,
$S \leftarrow T_p^{i@p1+1}$,
$T \leftarrow T_p^{i@p1} + ADJ_p^i\}$,
wpred_correct $\{i \leftarrow i@p1\}$,
div_ineq
$\{z \leftarrow (1+\rho)$,
$y \leftarrow R - ADJ_p^{i@p1}$,
$x \leftarrow R - \alpha(\lfloor \beta' + 2 * \Lambda' \rfloor)\}$,
rho_0,
ADJ_bound $\{i \leftarrow i@p1\}$,
$|\star 1|$ $\{x \leftarrow ADJ_p^{i@p1}\}$,
R_bound $\{i \leftarrow i@p1\}$,
wpred_hi_lem $\{i \leftarrow i@p1\}$,
Alpha_0 $\{i \leftarrow i@p1\}$,
iclock_ADJ_lem $\{i \leftarrow i,\ T \leftarrow T_p^{i@p1} + ADJ_p^i\}$

rmax2_0: **Lemma** option2 $\supset$ rmax_pred$(p, 0)$

rmax2_plus: **Lemma** option2 $\supset$ rmax_pred$(p, i+1)$

rmax2_pr: **Prove** rmax2 **from** rmax2_0, rmax2_plus $\{i \leftarrow \mathsf{pred}(i)\}$

rmax2_0_pr: **Prove** rmax2_0 **from**
rmax_pred $\{i \leftarrow 0\}$,
synctime0_defn,
synctime_defn $\{i \leftarrow i@p1\}$,
option2_defn $\{i \leftarrow i@p1\}$,
R_0,
RATE_1_iclock $\{i \leftarrow i@p1,\ S \leftarrow T_p^{i@p1+1},\ T \leftarrow T^0\}$,
wpred_correct $\{i \leftarrow i@p1\}$,
mult_leq_2
$\{z \leftarrow (1+\rho)$,
$y \leftarrow R - ADJ_p^{i@p1}$,
$x \leftarrow R + \boldsymbol{\alpha}(\lfloor \beta' + 2 * \Lambda' \rfloor)\}$,
mult_com $\{x \leftarrow (T_p^{i@p1+1} - T^0),\ y \leftarrow (1+\rho)\}$,
rho_0,
ADJ_bound $\{i \leftarrow i@p1\}$,
$|\star 1|$ $\{x \leftarrow ADJ_p^{i@p1}\}$,
R_bound $\{i \leftarrow i@p1\}$,
wpred_hi_lem $\{i \leftarrow i@p1\}$,
Alpha_0 $\{i \leftarrow i@p1\}$

rmax2_plus_pr: **Prove** rmax2_plus **from**
rmax_pred $\{i \leftarrow i+1\}$,
synctime_defn,
synctime_defn $\{i \leftarrow i@p1\}$,
option2_defn,
option2_defn $\{i \leftarrow i@p1\}$,
R_0,
RATE_1_iclock
$\{i \leftarrow i@p1$,
$S \leftarrow T_p^{i@p1+1}$,
$T \leftarrow T_p^{i@p1} + ADJ_p^i\}$,
wpred_correct $\{i \leftarrow i@p1\}$,
mult_leq_2
$\{z \leftarrow (1+\rho)$,
$y \leftarrow R - ADJ_p^{i@p1}$,
$x \leftarrow R + \boldsymbol{\alpha}(\lfloor \beta' + 2 * \Lambda' \rfloor)\}$,
mult_com $\{x \leftarrow (T_p^{i@p1+1} - (T_p^{i@p1} + ADJ_p^i)),\ y \leftarrow (1+\rho)\}$,
rho_0,
ADJ_bound $\{i \leftarrow i@p1\}$,
$|\star 1|$ $\{x \leftarrow ADJ_p^{i@p1}\}$,
R_bound $\{i \leftarrow i@p1\}$,
wpred_hi_lem $\{i \leftarrow i@p1\}$,
Alpha_0 $\{i \leftarrow i@p1\}$,
iclock_ADJ_lem $\{i \leftarrow i,\ T \leftarrow T_p^{i@p1} + ADJ_p^i\}$

rmin1_0: **Lemma** option1 $\supset$ rmin_pred$(p, 0)$

rmin1_plus: **Lemma** option1 $\supset$ rmin_pred$(p, i+1)$

rmin1_pr: **Prove** rmin1 **from** rmin1_0, rmin1_plus $\{i \leftarrow \mathsf{pred}(i)\}$

rmin1_0_pr: **Prove** rmin1_0 **from**
rmin_pred $\{i \leftarrow 0\}$,
synctime0_defn,
synctime_defn $\{i \leftarrow i@p1\}$,
option1_defn $\{i \leftarrow i@p1\}$,
R_0,
RATE_2_iclock $\{i \leftarrow i@p1,\ S \leftarrow T_p^{i@p1+1},\ T \leftarrow T^0\}$,
wpred_correct $\{i \leftarrow i@p1\}$,
Alpha_0 $\{i \leftarrow i@p1\}$,
div_ineq $\{z \leftarrow (1+\rho),\ y \leftarrow R,\ x \leftarrow R - \boldsymbol{\alpha}(\lfloor \beta' + 2 * \Lambda' \rfloor)\}$,
rho_0

rmin1_plus_pr: **Prove** rmin1_plus **from**
rmin_pred $\{i \leftarrow i+1\}$,
synctime_defn,
synctime_defn $\{i \leftarrow i@p1\}$,
option1_defn,
option1_defn $\{i \leftarrow i@p1\}$,
R_0,
RATE_2_iclock
$\{i \leftarrow i@p1,$
$S \leftarrow T_p^{i@p1+1},$
$T \leftarrow T_p^{i@p1} + ADJ_p^i\}$,
wpred_correct $\{i \leftarrow i@p1\}$,
Alpha_0 $\{i \leftarrow i@p1\}$,
div_ineq
$\{z \leftarrow (1+\rho),$
$y \leftarrow R - ADJ_p^i,$
$x \leftarrow R - \boldsymbol{\alpha}(\lfloor \beta' + 2 * \Lambda' \rfloor)\}$,
rho_0,
R_bound $\{i \leftarrow i@p1\}$,
wpred_hi_lem $\{i \leftarrow i@p1\}$,
$|\star 1|$ $\{x \leftarrow ADJ_p^i\}$,
ADJ_recovery,
ADJ_bound,
wpred_preceding,
iclock_ADJ_lem $\{T \leftarrow T_p^{i@p1} + ADJ_p^i\}$

rmax1_0: **Lemma** option1 $\supset$ rmax_pred$(p, 0)$

rmax1_plus: **Lemma** option1 $\supset$ rmax_pred$(p, i+1)$

rmax1_pr: **Prove** rmax1 **from** rmax1_0, rmax1_plus $\{i \leftarrow \mathsf{pred}(i)\}$

rmax1_0_pr: **Prove** rmax1_0 **from**
rmax_pred $\{i \leftarrow 0\}$,
synctime0_defn,
synctime_defn $\{i \leftarrow i@p1\}$,
option1_defn $\{i \leftarrow i@p1\}$,
R_0,
RATE_1_iclock $\{i \leftarrow i@p1,\ S \leftarrow T_p^{i@p1+1},\ T \leftarrow T^0\}$,
wpred_correct $\{i \leftarrow i@p1\}$,
Alpha_0 $\{i \leftarrow i@p1\}$,
mult_leq_2 $\{z \leftarrow (1+\rho),\ y \leftarrow R,\ x \leftarrow R + \boldsymbol{\alpha}(\lfloor \beta' + 2 * \Lambda' \rfloor)\}$,
mult_com $\{x \leftarrow (T_p^{i@p1+1} - T^0),\ y \leftarrow (1+\rho)\}$,
rho_0

rmax1_plus_pr: **Prove** rmax1_plus **from**
rmax_pred $\{i \leftarrow i+1\}$,
synctime_defn,
synctime_defn $\{i \leftarrow i@p1\}$,
option1_defn,
option1_defn $\{i \leftarrow i@p1\}$,
R_0,
RATE_1_iclock
$\{i \leftarrow i@p1,$
$S \leftarrow T_p^{i@p1+1},$
$T \leftarrow T_p^{i@p1} + ADJ_p^i\}$,
wpred_correct $\{i \leftarrow i@p1\}$,
Alpha_0 $\{i \leftarrow i@p1\}$,
mult_leq_2
$\{z \leftarrow (1+\rho),$
$y \leftarrow R - ADJ_p^i,$
$x \leftarrow R + \boldsymbol{\alpha}(\lfloor \beta' + 2 * \Lambda' \rfloor)\}$,
mult_com $\{x \leftarrow (T_p^{i@p1+1} - (T_p^{i@p1} + ADJ_p^i)),\ y \leftarrow (1+\rho)\}$,
rho_0,
R_bound $\{i \leftarrow i@p1\}$,
wpred_hi_lem $\{i \leftarrow i@p1\}$,
$|\star 1|$ $\{x \leftarrow ADJ_p^i\}$,
ADJ_recovery,
ADJ_bound,
wpred_preceding,
iclock_ADJ_lem $\{T \leftarrow T_p^{i@p1} + ADJ_p^i\}$

**End** rmax_rmin

# Appendix C

# Fault-Tolerant Midpoint Modules

This appendix contains the EHDM modules and proof chain analysis showing that the properties of translation invariance, precision enhancement, and accuracy preservation have been established for the fault-tolerant midpoint convergence function. In the interest of brevity, the proof chain status has been trimmed to show just the overall proof status and the axioms at the base.

## C.1 Proof Analysis

### C.1.1 Proof Chain for Translation Invariance

```
Terse proof chain for proof ft_mid_trans_inv_pr in module mid

⋮

   ================== SUMMARY ==================

The proof chain is complete

The axioms and assumptions at the base are:
  clocksort.funsort_trans_inv
  division.mult_div_1
  division.mult_div_2
  division.mult_div_3
  floor_ceil.floor_defn
  ft_mid_assume.No_authentication
Total: 6

⋮
```

### C.1.2 Proof Chain for Precision Enhancement

```
Terse proof chain for proof ft_mid_precision_enhancement_pr in module mid3

⋮

   ================== SUMMARY ==================

The proof chain is complete

The axioms and assumptions at the base are:
  clocksort.cnt_sort_geq
  clocksort.cnt_sort_leq
  division.mult_div_1
  division.mult_div_2
  division.mult_div_3
  floor_ceil.ceil_defn
  floor_ceil.floor_defn
  ft_mid_assume.No_authentication
  multiplication.mult_non_neg
  multiplication.mult_pos
  noetherian[EXPR, EXPR].general_induction
Total: 11

⋮
```

### C.1.3 Proof Chain for Accuracy Preservation

```
Terse proof chain for proof ft_mid_acc_pres_pr in module mid4

⋮

   ================== SUMMARY ==================

The proof chain is complete

The axioms and assumptions at the base are:
  clocksort.cnt_sort_geq
  clocksort.cnt_sort_leq
  clocksort.funsort_ax
  division.mult_div_1
```

```
  division.mult_div_2
  division.mult_div_3
  floor_ceil.floor_defn
  ft_mid_assume.No_authentication
  multiplication.mult_pos
  noetherian[EXPR, EXPR].general_induction
Total: 10

⋮
```

## C.2 mid

mid: **Module**

**Using** arith, clockassumptions, select_defs, ft_mid_assume

**Exporting all with** select_defs

**Theory**

process: **Type is** nat
Clocktime: **Type is** integer
$l, m, n, p, q$: **Var** process
$\vartheta$: **Var** function[process → Clocktime]
$i, j, k$: **Var** posint
$T, X, Y, Z$: **Var** Clocktime
$cfn_{MID}$: function[process, function[process → Clocktime] → Clocktime] =
$(\lambda\, p, \vartheta : \lfloor(\vartheta_{(F+1)} + \vartheta_{(N-F)})/2\rfloor)$

ft_mid_trans_inv: **Lemma** $cfn_{MID}(p, (\lambda\, q : \vartheta(q) + X)) = cfn_{MID}(p, \vartheta) + X$

**Proof**

add_assoc_hack: **Lemma** $X + Y + Z + Y = (X + Z) + 2 \star Y$

add_assoc_hack_pr: **Prove** add_assoc_hack **from** $\star 1 \star \star 2$ $\{x \leftarrow 2,\ y \leftarrow Y\}$

ft_mid_trans_inv_pr: **Prove** ft_mid_trans_inv **from**
$cfn_{MID}$ ,
$cfn_{MID}$ $\{\vartheta \leftarrow (\lambda\, q : \vartheta(q) + X)\}$,
select_trans_inv $\{k \leftarrow F + 1\}$,
select_trans_inv $\{k \leftarrow N - F\}$,
add_assoc_hack $\{X \leftarrow \vartheta_{(F+1)},\ Z \leftarrow \vartheta_{(N-F)},\ Y \leftarrow X\}$,
div_distrib $\{x \leftarrow (\vartheta_{(F+1)} + \vartheta_{(N-F)}),\ y \leftarrow 2 \star X,\ z \leftarrow 2\}$,
div_cancel $\{x \leftarrow 2,\ y \leftarrow X\}$,
ft_mid_maxfaults,
floor_plus_int $\{x \leftarrow x@p6/2,\ i \leftarrow X\}$

**End** mid

## C.3 mid2

mid2: **Module**

**Using** arith, clockassumptions, mid

**Exporting all with** mid

**Theory**

Clocktime: **Type is** integer
$m, n, p, q, p_1, q_1$: **Var** process
$i, j, k, l$: **Var** posint
$x, y, z, r, s, t$: **Var** time
$D, X, Y, Z, R, S, T$: **Var** Clocktime
$\vartheta, \theta, \gamma$: **Var** function[process → Clocktime]
ppred, ppred1, ppred2: **Var** function[process → bool]

good_greater_F1: **Lemma**
count(ppred, $N$) $\geq N - F \supset (\exists\, p$ : ppred($p$) $\wedge\ \vartheta(p) \geq \vartheta_{(F+1)})$

good_less_NF: **Lemma**
count(ppred, $N$) $\geq N - F \supset (\exists\, p$ : ppred($p$) $\wedge\ \vartheta(p) \leq \vartheta_{(N-F)})$

**Proof**

good_greater_F1_pr: **Prove** good_greater_F1 $\{p \leftarrow p@p3\}$ **from**
count_geq_select $\{k \leftarrow F + 1\}$,
ft_mid_maxfaults,
count_exists
$\{$ppred $\leftarrow (\lambda\, p_1$ : ppred1@p4($p_1$) $\wedge$ ppred2@p4($p_1$)),
$n \leftarrow N\}$,
pigeon_hole
$\{$ppred1 $\leftarrow$ ppred,
ppred2 $\leftarrow (\lambda\, p_1 : \vartheta(p_1) \geq \vartheta_{(F+1)})$,
$n \leftarrow N$,
$k \leftarrow 1\}$

good_less_NF_pr: **Prove** good_less_NF $\{p \leftarrow p@p3\}$ **from**
count_leq_select $\{k \leftarrow N - F\}$,
ft_mid_maxfaults,
count_exists
$\{$ppred $\leftarrow (\lambda\, p_1 :$ ppred1@p4$(p_1) \wedge$ ppred2@p4$(p_1))$,
$n \leftarrow N\}$,
pigeon_hole
$\{$ppred1 $\leftarrow$ ppred,
ppred2 $\leftarrow (\lambda\, p_1 : \vartheta_{(N-F)} \geq \vartheta(p_1))$,
$n \leftarrow N$,
$k \leftarrow 1\}$

**End** mid2

## C.4 mid3

mid3: **Module**

**Using** arith, clockassumptions, mid2

**Exporting all with** mid2

**Theory**

Clocktime: **Type is** integer
$m, n, p, q, p_1, q_1$: **Var** process
$i, j, k, l$: **Var** posint
$x, y, z, r, s, t$: **Var** time
$D, X, Y, Z, R, S, T$: **Var** Clocktime
$\vartheta, \theta, \gamma$: **Var** function[process $\rightarrow$ Clocktime]
ppred, ppred1, ppred2: **Var** function[process $\rightarrow$ bool]
ft_mid_Pi: function[Clocktime, Clocktime $\rightarrow$ Clocktime] ==
$(\lambda\, X, Z : \lceil Z/2 + X \rceil)$

exchange_order: **Lemma**
ppred$(p) \wedge$ ppred$(q)$
$\wedge\, \theta(q) \leq \theta(p) \wedge \gamma(p) \leq \gamma(q) \wedge$ okay_pairs$(\theta, \gamma, X,$ ppred$)$
$\supset |\theta(p) - \gamma(q)| \leq X$

good_geq_F_add1: **Lemma**
count(ppred, $N$) $\geq N - F \supset (\exists\, p :$ ppred$(p) \wedge \vartheta(p) \geq \vartheta_{(F+1)})$

okay_pair_geq_F_add1: **Lemma**
count(ppred, $N$) $\geq N - F \wedge$ okay_pairs$(\theta, \gamma, X,$ ppred$)$
$\supset (\exists\, p_1, q_1 :$
ppred$(p_1) \wedge \theta(p_1) \geq \theta_{(F+1)}$
$\wedge$ ppred$(q_1) \wedge \gamma(q_1) \geq \gamma_{(F+1)} \wedge |\theta(p_1) - \gamma(q_1)| \leq X)$

good_between: **Lemma**
count(ppred, $N$) $\geq N - F$
$\supset (\exists\, p :$ ppred$(p) \wedge \gamma_{(F+1)} \geq \gamma(p) \wedge \theta(p) \geq \theta_{(N-F)})$

ft_mid_precision_enhancement: **Lemma**
count(ppred, $N$) $\geq N - F$
$\wedge$ okay_pairs$(\theta, \gamma, X,$ ppred$)$
$\wedge$ okay_Readpred$(\theta, Z,$ ppred$) \wedge$ okay_Readpred$(\gamma, Z,$ ppred$)$
$\supset |cfn_{MID}(p, \theta) - cfn_{MID}(q, \gamma)| \leq$ ft_mid_Pi$(X, Z)$

ft_mid_prec_enh_sym: **Lemma**
$$\begin{aligned}&\mathsf{count}(\mathsf{ppred}, N) \geq N - F\\ &\quad \wedge\ \mathsf{okay\_pairs}(\theta, \gamma, X, \mathsf{ppred})\\ &\quad\quad \wedge\ \mathsf{okay\_Readpred}(\theta, Z, \mathsf{ppred})\\ &\quad\quad\quad \wedge\ \mathsf{okay\_Readpred}(\gamma, Z, \mathsf{ppred}) \wedge (cfn_{MID}(p, \theta) > cfn_{MID}(q, \gamma))\\ &\quad \supset |cfn_{MID}(p, \theta) - cfn_{MID}(q, \gamma)| \leq \mathsf{ft\_mid\_Pi}(X, Z)\end{aligned}$$

ft_mid_eq: **Lemma** $\mathsf{count}(\mathsf{ppred}, N) \geq N - F$
$$\begin{aligned}&\quad \wedge\ \mathsf{okay\_pairs}(\theta, \gamma, X, \mathsf{ppred})\\ &\quad\quad \wedge\ \mathsf{okay\_Readpred}(\theta, Z, \mathsf{ppred})\\ &\quad\quad\quad \wedge\ \mathsf{okay\_Readpred}(\gamma, Z, \mathsf{ppred}) \wedge (cfn_{MID}(p, \theta) = cfn_{MID}(q, \gamma))\\ &\quad \supset |cfn_{MID}(p, \theta) - cfn_{MID}(q, \gamma)| \leq \mathsf{ft\_mid\_Pi}(X, Z)\end{aligned}$$

ft_mid_prec_sym1: **Lemma**
$$\begin{aligned}&\mathsf{count}(\mathsf{ppred}, N) \geq N - F\\ &\quad \wedge\ \mathsf{okay\_pairs}(\theta, \gamma, X, \mathsf{ppred})\\ &\quad\quad \wedge\ \mathsf{okay\_Readpred}(\theta, Z, \mathsf{ppred})\\ &\quad\quad\quad \wedge\ \mathsf{okay\_Readpred}(\gamma, Z, \mathsf{ppred})\\ &\quad\quad\quad\quad \wedge\ ((\theta_{(F+1)} + \theta_{(N-F)}) > (\gamma_{(F+1)} + \gamma_{(N-F)}))\\ &\quad \supset |(\theta_{(F+1)} + \theta_{(N-F)}) - (\gamma_{(F+1)} + \gamma_{(N-F)})| \leq Z + 2 \star X\end{aligned}$$

mid_gt_imp_sel_gt: **Lemma**
$$\begin{aligned}&(cfn_{MID}(p, \theta) > cfn_{MID}(q, \gamma))\\ &\quad \supset ((\theta_{(F+1)} + \theta_{(N-F)}) > (\gamma_{(F+1)} + \gamma_{(N-F)}))\end{aligned}$$

okay_pairs_sym: **Lemma**
$$\mathsf{okay\_pairs}(\theta, \gamma, X, \mathsf{ppred}) \supset \mathsf{okay\_pairs}(\gamma, \theta, X, \mathsf{ppred})$$

## Proof

ft_mid_prec_sym1_pr: **Prove** ft_mid_prec_sym1 **from**
good_between,
okay_pair_geq_F_add1,
good_less_NF $\{\vartheta \leftarrow \gamma\}$,
abs_geq
$\{x \leftarrow (\gamma(q_1 @ \mathsf{p2}) - \gamma(p @ p3)) + (\theta(p @ p1) - \gamma(p @ p1))$
$\quad + (\theta(p_1 @ \mathsf{p2}) - \gamma(q_1 @ \mathsf{p2}))$,
$y \leftarrow (\theta_{(F+1)} + \theta_{(N-F)}) - (\gamma_{(F+1)} + \gamma_{(N-F)})\}$,
abs_plus
$\{x \leftarrow (\gamma(q_1 @ \mathsf{p2}) - \gamma(p @ p3)) + (\theta(p @ p1) - \gamma(p @ p1))$,
$y \leftarrow (\theta(p_1 @ \mathsf{p2}) - \gamma(q_1 @ \mathsf{p2}))\}$,
abs_plus $\{x \leftarrow (\gamma(q_1 @ \mathsf{p2}) - \gamma(p @ p3))$, $y \leftarrow (\theta(p @ p1) - \gamma(p @ p1))\}$,
okay_pairs $\{\gamma \leftarrow \theta,\ \theta \leftarrow \gamma,\ x \leftarrow X,\ p_3 \leftarrow p @ p1\}$,
okay_Readpred $\{\gamma \leftarrow \gamma,\ y \leftarrow Z,\ l \leftarrow q_1 @ \mathsf{p2},\ m \leftarrow p @ p3\}$,
distrib $\{x \leftarrow 1,\ y \leftarrow 1,\ z \leftarrow X\}$,
mult_lident $\{x \leftarrow X\}$

mid_gt_imp_sel_gt_pr: **Prove** mid_gt_imp_sel_gt **from**
  $cfn_{MID}$ $\{\vartheta \leftarrow \theta\}$,
  $cfn_{MID}$ $\{\vartheta \leftarrow \gamma,\ p \leftarrow q\}$,
  mult_div $\{x \leftarrow (\theta_{(F+1)} + \theta_{(N-F)}),\ y \leftarrow 2\}$,
  mult_div $\{x \leftarrow (\gamma_{(F+1)} + \gamma_{(N-F)}),\ y \leftarrow 2\}$,
  mult_floor_gt $\{x \leftarrow x@p3/2,\ y \leftarrow x@p4/2,\ z \leftarrow 2\}$

ft_mid_eq_pr: **Prove** ft_mid_eq **from**
  count_exists $\{n \leftarrow N\}$,
  ft_mid_maxfaults,
  okay_pairs $\{\gamma \leftarrow \theta,\ \theta \leftarrow \gamma,\ x \leftarrow X,\ p_3 \leftarrow p@p1\}$,
  okay_Readpred $\{\gamma \leftarrow \gamma,\ y \leftarrow Z,\ l \leftarrow p@p1,\ m \leftarrow p@p1\}$,
  $|\star 1|$ $\{x \leftarrow cfn_{MID}(p, \theta) - cfn_{MID}(q, \gamma)\}$,
  $|\star 1|$ $\{x \leftarrow \gamma(p@p1) - \gamma(p@p1)\}$,
  $|\star 1|$ $\{x \leftarrow \theta(p@p1) - \gamma(p@p1)\}$,
  ceil_defn $\{x \leftarrow Z/2 + X\}$,
  div_nonnegative $\{x \leftarrow Z,\ y \leftarrow 2\}$

ft_mid_prec_enh_sym_pr: **Prove** ft_mid_prec_enh_sym **from**
  $cfn_{MID}$ $\{\vartheta \leftarrow \theta\}$,
  $cfn_{MID}$ $\{\vartheta \leftarrow \gamma,\ p \leftarrow q\}$,
  div_minus_distrib
    $\{x \leftarrow (\theta_{(F+1)} + \theta_{(N-F)}),$
     $y \leftarrow (\gamma_{(F+1)} + \gamma_{(N-F)}),$
     $z \leftarrow 2\}$,
  abs_div
    $\{x \leftarrow (\theta_{(F+1)} + \theta_{(N-F)}) - (\gamma_{(F+1)} + \gamma_{(N-F)}),$
     $y \leftarrow 2\}$,
  ft_mid_prec_sym1,
  mid_gt_imp_sel_gt,
  div_ineq
    $\{x \leftarrow |(\theta_{(F+1)} + \theta_{(N-F)}) - (\gamma_{(F+1)} + \gamma_{(N-F)})|,$
     $y \leftarrow Z + 2 \star X,$
     $z \leftarrow 2\}$,
  div_distrib $\{x \leftarrow Z,\ y \leftarrow 2 \star X,\ z \leftarrow 2\}$,
  div_cancel $\{x \leftarrow 2,\ y \leftarrow X\}$,
  abs_floor_sub_floor_leq_ceil
    $\{x \leftarrow x@p3/2,$
     $y \leftarrow y@p3/2,$
     $z \leftarrow Z/2 + X\}$

okay_pairs_sym_pr: **Prove** okay_pairs_sym **from**
  okay_pairs $\{\gamma \leftarrow \theta,\ \theta \leftarrow \gamma,\ x \leftarrow X,\ p_3 \leftarrow p_3@p2\}$,
  okay_pairs $\{\gamma \leftarrow \gamma,\ \theta \leftarrow \theta,\ x \leftarrow X\}$,
  abs_com $\{x \leftarrow \theta(p_3@p2),\ y \leftarrow \gamma(p_3@p2)\}$

ft_mid_precision_enhancement_pr: **Prove** ft_mid_precision_enhancement **from**
  ft_mid_prec_enh_sym,
  ft_mid_prec_enh_sym
    $\{p \leftarrow q@p1$,
     $q \leftarrow p@p1$,
     $\theta \leftarrow \gamma$@p1,
     $\gamma \leftarrow \theta$@p1$\}$,
  ft_mid_eq,
  okay_pairs_sym,
  abs_com $\{x \leftarrow cfn_{MID}(p, \theta),\ y \leftarrow cfn_{MID}(q, \gamma)\}$

okay_pair_geq_F_add1_pr: **Prove**
  okay_pair_geq_F_add1
    $\{p_1 \leftarrow$ **if** $(\theta(p@p2) \geq \theta(p@p1))$
        **then** $p@p2$
        **elsif** $(\gamma(p@p1) \geq \gamma(p@p2))$ **then** $p@p1$ **else** $p@p3$
        **end if**,
     $q_1 \leftarrow$ **if** $(\theta(p@p2) \geq \theta(p@p1))$
        **then** $p@p2$
        **elsif** $(\gamma(p@p1) \geq \gamma(p@p2))$ **then** $p@p1$ **else** $q@p3$
        **end if**$\}$ **from**
  good_geq_F_add1 $\{\vartheta \leftarrow \theta\}$,
  good_geq_F_add1 $\{\vartheta \leftarrow \gamma\}$,
  exchange_order $\{p \leftarrow p@p1,\ q \leftarrow p@p2\}$,
  okay_pairs $\{\gamma \leftarrow \theta,\ \theta \leftarrow \gamma,\ x \leftarrow X,\ p_3 \leftarrow p@p1\}$,
  okay_pairs $\{\gamma \leftarrow \theta,\ \theta \leftarrow \gamma,\ x \leftarrow X,\ p_3 \leftarrow p@p2\}$

good_geq_F_add1_pr: **Prove** good_geq_F_add1 $\{p \leftarrow p@p1\}$ **from**
  count_exists
    {ppred $\leftarrow$ ( $\lambda\, p$ : ((ppred1@$p2$)$p$) $\wedge$ ((ppred2@$p2$)$p$)),
     $n \leftarrow N\}$,
  pigeon_hole
    $\{n \leftarrow N$,
     $k \leftarrow 1$,
     ppred1 $\leftarrow$ ppred,
     ppred2 $\leftarrow$ ( $\lambda\, p : \vartheta(p) \geq \vartheta_{((k@p3))}$)$\}$,
  count_geq_select $\{k \leftarrow F + 1\}$,
  ft_mid_maxfaults

good_between_pr: **Prove** good_between $\{p \leftarrow p@p1\}$ **from**
  count_exists
    $\{$ppred $\leftarrow$ $(\lambda\, p : ((\text{ppred1}@p2)p) \wedge ((\text{ppred2}@p2)p))$,
     $n \leftarrow N\}$,
  pigeon_hole
    $\{n \leftarrow N$,
     $k \leftarrow 1$,
     ppred1 $\leftarrow$ $(\lambda\, p : ((\text{ppred1}@p3)p) \wedge ((\text{ppred2}@p3)p))$,
     ppred2 $\leftarrow$ $(\lambda\, p : \theta(p) \geq \theta_{((k@p4))})\}$,
  pigeon_hole
    $\{n \leftarrow N$,
     $k \leftarrow k@p5$,
     ppred1 $\leftarrow$ ppred,
     ppred2 $\leftarrow$ $(\lambda\, p : \gamma_{((k@p5))} \geq \gamma(p))\}$,
  count_geq_select $\{\vartheta \leftarrow \theta,\ k \leftarrow N - F\}$,
  count_leq_select $\{\vartheta \leftarrow \gamma,\ k \leftarrow F + 1\}$,
  No_authentication

exchange_order_pr: **Prove** exchange_order **from**
  okay_pairs $\{\gamma \leftarrow \theta,\ \theta \leftarrow \gamma,\ x \leftarrow X,\ p_3 \leftarrow p\}$,
  okay_pairs $\{\gamma \leftarrow \theta,\ \theta \leftarrow \gamma,\ x \leftarrow X,\ p_3 \leftarrow q\}$,
  abs_geq $\{x \leftarrow (\theta(p) - \gamma(p)),\ y \leftarrow \theta(p) - \gamma(q)\}$,
  abs_geq $\{x \leftarrow (\gamma(q) - \theta(q)),\ y \leftarrow \gamma(q) - \theta(p)\}$,
  abs_com $\{x \leftarrow \theta(q),\ y \leftarrow \gamma(q)\}$,
  abs_com $\{x \leftarrow \theta(p),\ y \leftarrow \gamma(q)\}$

**End** mid3

## C.5 mid4

mid4: **Module**

**Using** arith, clockassumptions, mid3

**Exporting all with** clockassumptions, mid3

**Theory**

process: **Type is** nat
Clocktime: **Type is** integer
$m, n, p, q, p_1, q_1$: **Var** process
$i, j, k$: **Var** posint
$x, y, z, r, s, t$: **Var** time
$D, X, Y, Z, R, S, T$: **Var** Clocktime
$\vartheta, \theta, \gamma$: **Var** function[process → Clocktime]
ppred, ppred1, ppred2: **Var** function[process → bool]

ft_mid_accuracy_preservation: **Lemma**
ppred$(q) \wedge$ count(ppred, $N) \geq N - F \wedge$ okay_Readpred$(\vartheta, X,$ ppred$)$
$\supset |cfn_{MID}(p, \vartheta) - \vartheta(q)| \leq X$

ft_mid_less: **Lemma** $cfn_{MID}(p, \vartheta) \leq \vartheta_{(F+1)}$

ft_mid_greater: **Lemma** $cfn_{MID}(p, \vartheta) \geq \vartheta_{(N-F)}$

abs_q_less: **Lemma**
count(ppred, $N) \geq N - F \supset (\exists\, p_1 :$ ppred$(p_1) \wedge \vartheta(p_1) \leq cfn_{MID}(p, \vartheta))$

abs_q_greater: **Lemma**
count(ppred, $N) \geq N - F \supset (\exists\, p_1 :$ ppred$(p_1) \wedge \vartheta(p_1) \geq cfn_{MID}(p, \vartheta))$

ft_mid_bnd_by_good: **Lemma**
count(ppred, $N) \geq N - F$
$\supset (\exists\, p_1 :$ ppred$(p_1) \wedge |cfn_{MID}(p, \vartheta) - \vartheta(q)| \leq |\vartheta(p_1) - \vartheta(q)|)$

maxfaults_lem: **Lemma** $F + 1 \leq N - F$

ft_select: **Lemma** $\vartheta_{(F+1)} \geq \vartheta_{(N-F)}$

**Proof**

ft_select_pr: **Prove** ft_select **from**
select_ax $\{i \leftarrow F + 1,\ k \leftarrow N - F\}$, maxfaults_lem

maxfaults_lem_pr: **Prove** maxfaults_lem **from** ft_mid_maxfaults

ft_mid_bnd_by_good_pr: **Prove**
ft_mid_bnd_by_good
$\{p_1 \leftarrow$ ( **if** $cfn_{MID}(p, \vartheta) \geq \vartheta(q)$ **then** $p_1$@p1 **else** $p_1$@p2 **end if**)$\}$ **from**
abs_q_greater,
abs_q_less,
abs_com $\{x \leftarrow \vartheta(q),\ y \leftarrow \vartheta(p_1@c)\}$,
abs_com $\{x \leftarrow \vartheta(q),\ y \leftarrow cfn_{MID}(p, \vartheta)\}$,
abs_geq $\{x \leftarrow x@p3 - y@p3,\ y \leftarrow x@p4 - y@p4\}$,
abs_geq $\{x \leftarrow \vartheta(p_1@c) - \vartheta(q),\ y \leftarrow cfn_{MID}(p, \vartheta) - \vartheta(q)\}$

abs_q_less_pr: **Prove** abs_q_less $\{p_1 \leftarrow p@p1\}$ **from**
good_less_NF, ft_mid_greater

abs_q_greater_pr: **Prove** abs_q_greater $\{p_1 \leftarrow p@p1\}$ **from**
good_greater_F1, ft_mid_less

mult_hack: **Lemma** $X + X = 2 \star X$

mult_hack_pr: **Prove** mult_hack **from** $\star 1 \star \star 2$ $\{x \leftarrow 2,\ y \leftarrow X\}$

ft_mid_less_pr: **Prove** ft_mid_less **from**
$cfn_{MID}$ ,
ft_select,
div_ineq
$\{x \leftarrow (\vartheta_{(F+1)} + \vartheta_{(N-F)})$,
$y \leftarrow (\vartheta_{(F+1)} + \vartheta_{(F+1)})$,
$z \leftarrow 2\}$,
div_cancel $\{x \leftarrow 2,\ y \leftarrow \vartheta_{(F+1)}\}$,
mult_hack $\{X \leftarrow \vartheta_{(F+1)}\}$,
floor_defn $\{x \leftarrow x@p3/2\}$

ft_mid_greater_pr: **Prove** ft_mid_greater **from**
$cfn_{MID}$ ,
ft_select,
div_ineq
$\{x \leftarrow (\vartheta_{(N-F)} + \vartheta_{(N-F)})$,
$y \leftarrow (\vartheta_{(F+1)} + \vartheta_{(N-F)})$,
$z \leftarrow 2\}$,
div_cancel $\{x \leftarrow 2,\ y \leftarrow \vartheta_{(N-F)}\}$,
mult_hack $\{X \leftarrow \vartheta_{(N-F)}\}$,
floor_mon $\{x \leftarrow x@p3/2,\ y \leftarrow y@p3/2\}$,
floor_int $\{i \leftarrow X@p5\}$

ft_mid_acc_pres_pr: **Prove** ft_mid_accuracy_preservation **from**
ft_mid_bnd_by_good,
okay_Readpred $\{\gamma \leftarrow \vartheta,\ y \leftarrow X,\ l \leftarrow p_1@p1,\ m \leftarrow q@c\}$

**End** mid4

## C.6 select_defs

select_defs: **Module**

**Using** arith, countmod, clockassumptions, clocksort

**Exporting all with** clockassumptions

**Theory**

process: **Type is** nat
Clocktime: **Type is** integer
$l, m, n, p, q$: **Var** process
$\vartheta$: **Var** function[process → Clocktime]
$i, j, k$: **Var** posint
$T, X, Y, Z$: **Var** Clocktime
$\star 1_{(\star 2)}$: function[function[process → Clocktime], posint → Clocktime] ==
$(\lambda\, \vartheta, i : \vartheta(\text{funsort}(\vartheta)(i)))$

select_trans_inv: **Lemma** $k \leq N \supset (\lambda\, q : \vartheta(q) + X)_{(k)} = \vartheta_{(k)} + X$

select_exists1: **Lemma** $i \leq N \supset (\exists\, p : p < N \wedge \vartheta(p) = \vartheta_{(i)})$

select_exists2: **Lemma** $p < N \supset (\exists\, i : i \leq N \wedge \vartheta(p) = \vartheta_{(i)})$

select_ax: **Lemma** $1 \leq i \wedge i < k \wedge k \leq N \supset \vartheta_{(i)} \geq \vartheta_{(k)}$

count_geq_select: **Lemma** $k \leq N \supset \text{count}((\lambda\, p : \vartheta(p) \geq \vartheta_{(k)}), N) \geq k$

count_leq_select: **Lemma** $k \leq N \supset \text{count}((\lambda\, p : \vartheta_{(k)} \geq \vartheta(p)), N) \geq N - k + 1$

**Proof**

select_trans_inv_pr: **Prove** select_trans_inv **from** funsort_trans_inv

select_exists1_pr: **Prove** select_exists1 $\{p \leftarrow \text{funsort}(\vartheta)(i)\}$ **from**
funsort_fun_1_1 $\{j \leftarrow i\}$

select_exists2_pr: **Prove** select_exists2 $\{i \leftarrow i@p1\}$ **from** funsort_fun_onto

select_ax_pr: **Prove** select_ax **from** funsort_ax $\{i \leftarrow i@c,\ j \leftarrow k@c\}$

count_leq_select_pr: **Prove** count_leq_select **from** cnt_sort_leq

count_geq_select_pr: **Prove** count_geq_select **from** cnt_sort_geq

**End** select_defs

## C.7 ft_mid_assume

ft_mid_assume: **Module**

**Using** clockassumptions

**Exporting all with** clockassumptions

**Theory**

ft_mid_maxfaults: **Axiom** $N \geq 2 * F + 1$

No_authentication: **Axiom** $N \geq 3 * F + 1$

**Proof**

ft_mid_maxfaults_pr: **Prove** ft_mid_maxfaults **from** No_authentication

**End** ft_mid_assume

## C.8 clocksort

clocksort: **Module**

**Using** clockassumptions

**Exporting all with** clockassumptions

**Theory**

$l, m, n, p, q$: **Var** process
$i, j, k$: **Var** posint
$X, Y$: **Var** Clocktime
$\vartheta$: **Var** function[process → Clocktime]
funsort: function[function[process → Clocktime]
→ function[posint → process]]
`(* clock readings can be sorted *)`

funsort_ax: **Axiom** $i \leq j \wedge j \leq N \supset \vartheta(\mathsf{funsort}(\vartheta)(i)) \geq \vartheta(\mathsf{funsort}(\vartheta)(j))$

funsort_fun_1_1: **Axiom**
$i \leq N \wedge j \leq N \wedge \mathsf{funsort}(\vartheta)(i) = \mathsf{funsort}(\vartheta)(j) \supset i = j \wedge \mathsf{funsort}(\vartheta)(i) < N$

funsort_fun_onto: **Axiom** $p < N \supset (\exists\, i : i \leq N \wedge \mathsf{funsort}(\vartheta)(i) = p)$

funsort_trans_inv: **Axiom**
$k \leq N \supset (\vartheta(\mathsf{funsort}((\lambda\, q : \vartheta(q) + X))(k)) = \vartheta(\mathsf{funsort}(\vartheta)(k)))$

cnt_sort_geq: **Axiom** $k \leq N \supset \mathsf{count}((\lambda\, p : \vartheta(p) \geq \vartheta(\mathsf{funsort}(\vartheta)(k))), N) \geq k$

cnt_sort_leq: **Axiom**
$k \leq N \supset \mathsf{count}((\lambda\, p : \vartheta(\mathsf{funsort}(\vartheta)(k)) \geq \vartheta(p)), N) \geq N - k + 1$

**Proof**

**End** clocksort

# Appendix D

# Utility Modules

This appendix contains the EHDM utility modules required for the clock synchronization proofs. Most of these were taken from Shankar's theory (ref. 10). The induction modules are from Rushby's transient recovery verification (ref. 17). Module countmod was substantially changed in the course of this verification and is therefore much different from Shankar's module countmod. Also, module floor_ceil added a number of useful properties required to support the conversion of Clocktime from real to integer. In Shankar's presentation Clocktime ranged over the reals.

## D.1 multiplication

multiplication: **Module**

**Exporting all**

**Theory**

$x, y, z, x_1, y_1, z_1, x_2, y_2, z_2$: **Var** number
$\star 1 \star \star 2$: function[number, number → number] = $(\lambda\, x, y : (x * y))$

mult_ldistrib: **Lemma** $x \star (y + z) = x \star y + x \star z$

mult_ldistrib_minus: **Lemma** $x \star (y - z) = x \star y - x \star z$

mult_rident: **Lemma** $x \star 1 = x$

mult_lident: **Lemma** $1 \star x = x$

distrib: **Lemma** $(x + y) \star z = x \star z + y \star z$

distrib_minus: **Lemma** $(x - y) \star z = x \star z - y \star z$

mult_non_neg: **Axiom** $((x \geq 0 \wedge y \geq 0) \vee (x \leq 0 \wedge y \leq 0)) \Leftrightarrow x \star y \geq 0$

mult_pos: **Axiom** $((x > 0 \wedge y > 0) \vee (x < 0 \wedge y < 0)) \Leftrightarrow x \star y > 0$

mult_com: **Lemma** $x \star y = y \star x$

pos_product: **Lemma** $x \geq 0 \wedge y \geq 0 \supset x \star y \geq 0$

mult_leq: **Lemma** $z \geq 0 \wedge x \geq y \supset x \star z \geq y \star z$

mult_leq_2: **Lemma** $z \geq 0 \wedge x \geq y \supset z \star x \geq z \star y$

mult_l0: **Axiom** $0 \star x = 0$

mult_gt: **Lemma** $z > 0 \wedge x > y \supset x \star z > y \star z$

**Proof**

mult_gt_pr: **Prove** mult_gt **from**
  mult_pos $\{x \leftarrow x - y,\ y \leftarrow z\}$, distrib_minus

distrib_minus_pr: **Prove** distrib_minus **from**
  mult_ldistrib_minus $\{x \leftarrow z,\ y \leftarrow x,\ z \leftarrow y\}$,
  mult_com $\{x \leftarrow x - y,\ y \leftarrow z\}$,
  mult_com $\{y \leftarrow z\}$,
  mult_com $\{x \leftarrow y,\ y \leftarrow z\}$

mult_leq_2_pr: **Prove** mult_leq_2 **from**
  mult_ldistrib_minus $\{x \leftarrow z,\ y \leftarrow x,\ z \leftarrow y\}$,
  mult_non_neg $\{x \leftarrow z,\ y \leftarrow x - y\}$

mult_leq_pr: **Prove** mult_leq **from**
  distrib_minus, mult_non_neg $\{x \leftarrow x - y,\ y \leftarrow z\}$

mult_com_pr: **Prove** mult_com **from** $\star 1 \star\star 2$ , $\star 1 \star\star 2$ $\{x \leftarrow y,\ y \leftarrow x\}$

pos_product_pr: **Prove** pos_product **from** mult_non_neg

mult_rident_proof: **Prove** mult_rident **from** $\star 1 \star\star 2$ $\{y \leftarrow 1\}$

mult_lident_proof: **Prove** mult_lident **from** $\star 1 \star\star 2$ $\{x \leftarrow 1,\ y \leftarrow x\}$

distrib_proof: **Prove** distrib **from**
  $\star 1 \star\star 2$ $\{x \leftarrow x + y,\ y \leftarrow z\}$,
  $\star 1 \star\star 2$ $\{y \leftarrow z\}$,
  $\star 1 \star\star 2$ $\{x \leftarrow y,\ y \leftarrow z\}$

mult_ldistrib_proof: **Prove** mult_ldistrib **from**
  $\star 1 \star\star 2$ $\{y \leftarrow y + z,\ x \leftarrow x\}$, $\star 1 \star\star 2$ , $\star 1 \star\star 2$ $\{y \leftarrow z\}$

mult_ldistrib_minus_proof: **Prove** mult_ldistrib_minus **from**
  $\star 1 \star\star 2$ $\{y \leftarrow y - z,\ x \leftarrow x\}$, $\star 1 \star\star 2$ , $\star 1 \star\star 2$ $\{y \leftarrow z\}$

**End** multiplication

## D.2 division

division: **Module**

**Using** multiplication, absmod, floor_ceil

**Exporting all**

**Theory**

$x, y, z, x_1, y_1, z_1, x_2, y_2, z_2$: **Var** number

mult_div_1: **Axiom** $z \neq 0 \supset x \star y/z = x \star (y/z)$

mult_div_2: **Axiom** $z \neq 0 \supset x \star y/z = (x/z) \star y$

mult_div_3: **Axiom** $z \neq 0 \supset (z/z) = 1$

mult_div: **Lemma** $y \neq 0 \supset (x/y) \star y = x$

div_cancel: **Lemma** $x \neq 0 \supset x \star y/x = y$

div_distrib: **Lemma** $z \neq 0 \supset ((x + y)/z) = (x/z) + (y/z)$

ceil_mult_div: **Lemma** $y > 0 \supset \lceil x/y \rceil \star y \geq x$

ceil_plus_mult_div: **Lemma** $y > 0 \supset \lceil x/y \rceil + 1 \star y > x$

div_nonnegative: **Lemma** $x \geq 0 \wedge y > 0 \supset (x/y) \geq 0$

div_minus_distrib: **Lemma** $z \neq 0 \supset (x - y)/z = (x/z) - (y/z)$

div_ineq: **Lemma** $z > 0 \wedge x \leq y \supset (x/z) \leq (y/z)$

abs_div: **Lemma** $y > 0 \supset |x/y| = |x|/y$

mult_minus: **Lemma** $y \neq 0 \supset -(x/y) = (-x/y)$

div_minus_1: **Lemma** $y > 0 \wedge x < 0 \supset (x/y) < 0$

**Proof**

div_nonnegative_pr: **Prove** div_nonnegative **from**
mult_non_neg $\{x \leftarrow (\textbf{ if } y \neq 0 \textbf{ then } (x/y) \textbf{ else } 0 \textbf{ end if})\}$, mult_div

div_distrib_pr: **Prove** div_distrib **from**
mult_div_1 $\{x \leftarrow x+y,\ y \leftarrow 1,\ z \leftarrow z\}$,
mult_rident $\{x \leftarrow x+y\}$,
mult_div_1 $\{x \leftarrow x,\ y \leftarrow 1,\ z \leftarrow z\}$,
mult_rident,
mult_div_1 $\{x \leftarrow y,\ y \leftarrow 1,\ z \leftarrow z\}$,
mult_rident $\{x \leftarrow y\}$,
distrib $\{z \leftarrow (\ \textbf{if}\ z \neq 0\ \textbf{then}\ (1/z)\ \textbf{else}\ 0\ \textbf{end if})\}$

div_cancel_pr: **Prove** div_cancel **from**
mult_div_2 $\{z \leftarrow x\}$, mult_div_3 $\{z \leftarrow x\}$, mult_lident $\{x \leftarrow y\}$

mult_div_pr: **Prove** mult_div **from**
mult_div_2 $\{z \leftarrow y\}$, mult_div_1 $\{z \leftarrow y\}$, mult_div_3 $\{z \leftarrow y\}$, mult_rident

abs_div_pr: **Prove** abs_div **from**
$|\star 1|$ $\{x \leftarrow (\ \textbf{if}\ y \neq 0\ \textbf{then}\ (x/y)\ \textbf{else}\ 0\ \textbf{end if})\}$,
$|\star 1|$,
div_nonnegative,
div_minus_1,
mult_minus

mult_minus_pr: **Prove** mult_minus **from**
mult_div_1 $\{x \leftarrow -1,\ y \leftarrow x,\ z \leftarrow y\}$,
$\star 1 \star\star 2$ $\{x \leftarrow -1,\ y \leftarrow x\}$,
$\star 1 \star\star 2$ $\{x \leftarrow -1,\ y \leftarrow (\ \textbf{if}\ y \neq 0\ \textbf{then}\ (x/y)\ \textbf{else}\ 1\ \textbf{end if})\}$

div_minus_1_pr: **Prove** div_minus_1 **from**
mult_div,
pos_product $\{x \leftarrow (\ \textbf{if}\ y \neq 0\ \textbf{then}\ (x/y)\ \textbf{else}\ 0\ \textbf{end if}),\ y \leftarrow y\}$

div_minus_distrib_pr: **Prove** div_minus_distrib **from**
div_distrib $\{y \leftarrow -y\}$, mult_minus $\{x \leftarrow y,\ y \leftarrow z\}$

div_ineq_pr: **Prove** div_ineq **from**
mult_div $\{y \leftarrow z\}$,
mult_div $\{x \leftarrow y,\ y \leftarrow z\}$,
mult_gt
$\{x \leftarrow (\ \textbf{if}\ z \neq 0\ \textbf{then}\ (x/z)\ \textbf{else}\ 0\ \textbf{end if}),$
$y \leftarrow (\ \textbf{if}\ z \neq 0\ \textbf{then}\ (y/z)\ \textbf{else}\ 0\ \textbf{end if})\}$

ceil_plus_mult_div_proof: **Prove** ceil_plus_mult_div **from**
ceil_mult_div,
distrib
$\{x \leftarrow \lceil(\ \textbf{if}\ y \neq 0\ \textbf{then}\ (x/y)\ \textbf{else}\ 0\ \textbf{end if})\rceil,$
$y \leftarrow 1,$
$z \leftarrow y\}$,
mult_lident $\{x \leftarrow y\}$

ceil_mult_div_proof: **Prove** ceil_mult_div **from**
mult_div,
mult_leq
$\{x \leftarrow \lceil(\ \textbf{if}\ y \neq 0\ \textbf{then}\ (x/y)\ \textbf{else}\ 0\ \textbf{end if})\rceil,$
$y \leftarrow (\ \textbf{if}\ y \neq 0\ \textbf{then}\ (x/y)\ \textbf{else}\ 0\ \textbf{end if}),$
$z \leftarrow y\},$
ceil_defn $\{x \leftarrow (\ \textbf{if}\ y \neq 0\ \textbf{then}\ (x/y)\ \textbf{else}\ 0\ \textbf{end if})\}$

**End** division

## D.3 absmod

absmod: **Module**

**Using** multiplication

**Exporting all**

**Theory**

$x, y, z, x_1, y_1, z_1, x_2, y_2, z_2$: **Var** number
$X$: **Var** integer
$|\star 1|$: **Definition** function[number $\rightarrow$ number] =
  $(\lambda\, x : (\textbf{ if } x < 0 \textbf{ then } -x \textbf{ else } x \textbf{ end if}))$
iabs: **Definition** function[integer $\rightarrow$ integer] =
  $(\lambda\, X : (\textbf{ if } X < 0 \textbf{ then } -X \textbf{ else } X \textbf{ end if}))$

iabs_is_abs: **Lemma** $x = X \supset \mathsf{iabs}(X) = |x|$

abs_main: **Lemma** $|x| < z \supset (x < z \vee -x < z)$

abs_leq_0: **Lemma** $|x - y| \leq z \supset (x - y) \leq z$

abs_diff: **Lemma** $|x - y| < z \supset ((x - y) < z \vee (y - x) < z)$

abs_leq: **Lemma** $|x| \leq z \supset (x \leq z \vee -x \leq z)$

abs_bnd: **Lemma** $0 \leq z \wedge 0 \leq x \wedge x \leq z \wedge 0 \leq y \wedge y \leq z \supset |x - y| \leq z$

abs_1_bnd: **Lemma** $|x - y| \leq z \supset x \leq y + z$

abs_2_bnd: **Lemma** $|x - y| \leq z \supset x \geq y - z$

abs_3_bnd: **Lemma** $x \leq y + z \wedge x \geq y - z \supset |x - y| \leq z$

abs_drift: **Lemma** $|x - y| \leq z \wedge |x_1 - x| \leq z_1 \supset |x_1 - y| \leq z + z_1$

abs_com: **Lemma** $|x - y| = |y - x|$

abs_drift_2: **Lemma**
  $|x - y| \leq z \wedge |x_1 - x| \leq z_1 \wedge |y_1 - y| \leq z_2 \supset |x_1 - y_1| \leq z + z_1 + z_2$

abs_geq: **Lemma** $x \geq y \wedge y \geq 0 \supset |x| \geq |y|$

abs_ge0: **Lemma** $x \geq 0 \supset |x| = x$

abs_plus: **Lemma** $|x + y| \leq |x| + |y|$

abs_diff_3: **Lemma** $x - y \leq z \wedge y - x \leq z \supset |x - y| \leq z$

**Proof**

iabs_pr: **Prove** iabs_is_abs **from** $|\star 1|$ , iabs

abs_plus_pr: **Prove** abs_plus **from** $|\star 1|\ \{x \leftarrow x + y\}$, $|\star 1|$ , $|\star 1|\ \{x \leftarrow y\}$

abs_diff_3_pr: **Prove** abs_diff_3 **from** $|\star 1|\ \{x \leftarrow x - y\}$

abs_ge0_proof: **Prove** abs_ge0 **from** $|\star 1|$

abs_geq_proof: **Prove** abs_geq **from** $|\star 1|$ , $|\star 1|\ \{x \leftarrow y\}$

abs_drift_2_proof: **Prove** abs_drift_2 **from**
abs_drift,
abs_drift $\{x \leftarrow y,\ y \leftarrow y_1,\ z \leftarrow z_2,\ z_1 \leftarrow z + z_1\}$,
abs_com $\{x \leftarrow y_1\}$

abs_com_proof: **Prove** abs_com **from** $|\star 1|\ \{x \leftarrow (x - y)\}$, $|\star 1|\ \{x \leftarrow (y - x)\}$

abs_drift_proof: **Prove** abs_drift **from**
abs_1_bnd,
abs_1_bnd $\{x \leftarrow x_1,\ y \leftarrow x,\ z \leftarrow z_1\}$,
abs_2_bnd,
abs_2_bnd $\{x \leftarrow x_1,\ y \leftarrow x,\ z \leftarrow z_1\}$,
abs_3_bnd $\{x \leftarrow x_1,\ z \leftarrow z + z_1\}$

abs_3_bnd_proof: **Prove** abs_3_bnd **from** $|\star 1|\ \{x \leftarrow (x - y)\}$

abs_main_proof: **Prove** abs_main **from** $|\star 1|$

abs_leq_0_proof: **Prove** abs_leq_0 **from** $|\star 1|\ \{x \leftarrow x - y\}$

abs_diff_proof: **Prove** abs_diff **from** $|\star 1|\ \{x \leftarrow (x - y)\}$

abs_leq_proof: **Prove** abs_leq **from** $|\star 1|$

abs_bnd_proof: **Prove** abs_bnd **from** $|\star 1|\ \{x \leftarrow (x - y)\}$

abs_1_bnd_proof: **Prove** abs_1_bnd **from** $|\star 1|\ \{x \leftarrow (x - y)\}$

abs_2_bnd_proof: **Prove** abs_2_bnd **from** $|\star 1|\ \{x \leftarrow (x - y)\}$

**End** absmod

## D.4 floor_ceil

floor_ceil: **Module**

**Using** multiplication, absmod

**Exporting all**

**Theory**

$i, j$: **Var** integer
$x, y, z, x_1, y_1, z_1, x_2, y_2, z_2$: **Var** number
$\lceil \star 1 \rceil$: function[number $\rightarrow$ int]

ceil_defn: **Axiom** $\lceil x \rceil \geq x \wedge \lceil x \rceil - 1 < x$

$\lfloor \star 1 \rfloor$: function[number $\rightarrow$ int]

floor_defn: **Axiom** $\lfloor x \rfloor \leq x \wedge \lfloor x \rfloor + 1 > x$

ceil_geq: **Lemma** $\lceil x \rceil \geq x$

ceil_mon: **Lemma** $x \geq y \supset \lceil x \rceil \geq \lceil y \rceil$

ceil_int: **Lemma** $\lceil i \rceil = i$

floor_leq: **Lemma** $\lfloor x \rfloor \leq x$

floor_mon: **Lemma** $x \leq y \supset \lfloor x \rfloor \leq \lfloor y \rfloor$

floor_int: **Lemma** $\lfloor i \rfloor = i$

ceil_plus_i: **Lemma** $\lceil x \rceil + i \geq x + i \wedge \lceil x \rceil + i - 1 < x + i$

ceil_plus_int: **Lemma** $\lceil x \rceil + i = \lceil x + i \rceil$

int_plus_ceil: **Lemma** $i + \lceil x \rceil = \lceil i + x \rceil$

floor_plus_i: **Lemma** $\lfloor x \rfloor + i \leq x + i \wedge \lfloor x \rfloor + i + 1 > x + i$

floor_plus_int: **Lemma** $\lfloor x \rfloor + i = \lfloor x + i \rfloor$

neg_floor_eq_ceil_neg: **Lemma** $-\lfloor x \rfloor = \lceil -x \rceil$

neg_ceil_eq_floor_neg: **Lemma** $-\lceil x \rceil = \lfloor -x \rfloor$

ceil_sum: **Lemma** $\lceil x \rceil + \lceil y \rceil \leq \lceil x + y \rceil + 1$

abs_ceil_sum: **Lemma** $|\lceil x \rceil + \lceil y \rceil| \leq |\lceil x + y \rceil| + 1$

floor_sub_floor_leq_ceil: **Lemma** $x - y \leq z \supset \lfloor x \rfloor - \lfloor y \rfloor \leq \lceil z \rceil$

abs_floor_sub_floor_leq_ceil: **Lemma** $|x - y| \leq z \supset |\lfloor x \rfloor - \lfloor y \rfloor| \leq \lceil z \rceil$

floor_gt_imp_gt: **Lemma** $\lfloor x \rfloor > \lfloor y \rfloor \supset x > y$

mult_floor_gt: **Lemma** $z > 0 \wedge \lfloor x \rfloor > \lfloor y \rfloor \supset x \star z > y \star z$

## Proof

mult_floor_gt_pr: **Prove** mult_floor_gt **from** floor_gt_imp_gt, mult_gt

floor_gt_imp_gt_pr: **Prove** floor_gt_imp_gt **from**
floor_defn, floor_defn $\{x \leftarrow y\}$

floor_sub_floor_leq_ceil_pr: **Prove** floor_sub_floor_leq_ceil **from**
floor_defn, floor_defn $\{x \leftarrow y\}$, ceil_defn $\{x \leftarrow z\}$

abs_floor_sub_floor_leq_ceil_pr: **Prove** abs_floor_sub_floor_leq_ceil **from**
floor_defn,
floor_defn $\{x \leftarrow y\}$,
ceil_defn $\{x \leftarrow z\}$,
$|\star 1|$ $\{x \leftarrow x - y\}$,
$|\star 1|$ $\{x \leftarrow \lfloor x \rfloor - \lfloor y \rfloor\}$

int_plus_ceil_pr: **Prove** int_plus_ceil **from** ceil_plus_int

ceil_geq_pr: **Prove** ceil_geq **from** ceil_defn

ceil_mon_pr: **Prove** ceil_mon **from** ceil_defn, ceil_defn $\{x \leftarrow y\}$

floor_leq_pr: **Prove** floor_leq **from** floor_defn

floor_mon_pr: **Prove** floor_mon **from** floor_defn, floor_defn $\{x \leftarrow y\}$

ceil_eq_hack: **Sublemma** $i \geq x \wedge i - 1 < x \wedge j \geq x \wedge j - 1 < x \supset i = j$

ceil_eq_hack_pr: **Prove** ceil_eq_hack

ceil_plus_i_pr: **Prove** ceil_plus_i **from** ceil_defn

ceil_plus_int_pr: **Prove** ceil_plus_int **from**
ceil_plus_i,
ceil_defn $\{x \leftarrow x + i\}$,
ceil_eq_hack $\{x \leftarrow x + i,\ i \leftarrow \lceil x \rceil + i,\ j \leftarrow \lceil x + i \rceil\}$

floor_eq_hack: **Sublemma** $i \leq x \wedge i + 1 > x \wedge j \leq x \wedge j + 1 > x \supset i = j$

floor_eq_hack_pr: **Prove** floor_eq_hack

floor_plus_i_pr: **Prove** floor_plus_i **from** floor_defn

floor_plus_int_pr: **Prove** floor_plus_int **from**
  floor_plus_i,
  floor_defn $\{x \leftarrow x + i\}$,
  floor_eq_hack $\{x \leftarrow x + i,\ i \leftarrow \lfloor x \rfloor + i,\ j \leftarrow \lfloor x + i \rfloor\}$

neg_floor_eq_ceil_neg_pr: **Prove** neg_floor_eq_ceil_neg **from**
  floor_defn, ceil_defn $\{x \leftarrow -x\}$

neg_ceil_eq_floor_neg_pr: **Prove** neg_ceil_eq_floor_neg **from**
  floor_defn $\{x \leftarrow -x\}$, ceil_defn

ceil_sum_pr: **Prove** ceil_sum **from**
  ceil_defn $\{x \leftarrow x + y\}$, ceil_defn $\{x \leftarrow y\}$, ceil_defn

abs_ceil_sum_pr: **Prove** abs_ceil_sum **from**
  $|\star 1|$ $\{x \leftarrow \lceil x \rceil + \lceil y \rceil\}$,
  $|\star 1|$ $\{x \leftarrow \lceil x + y \rceil\}$,
  ceil_defn $\{x \leftarrow x + y\}$,
  ceil_defn $\{x \leftarrow y\}$,
  ceil_defn

ceil_int_pr: **Prove** ceil_int **from** ceil_defn $\{x \leftarrow i\}$

floor_int_pr: **Prove** floor_int **from** floor_defn $\{x \leftarrow i\}$

**End** floor_ceil

## D.5 natinduction

natinduction: **Module**

### Theory

$i, j, m, m_1, n$: **Var** nat
$p$, prop: **Var** function[nat $\rightarrow$ bool]

induction: **Theorem** (prop(0) $\wedge$ ( $\forall\, j$ : prop($j$) $\supset$ prop($j+1$))) $\supset$ prop($i$)

complete_induction: **Theorem**
( $\forall\, i$ : ( $\forall\, j$ : $j < i \supset p(j)$) $\supset$ $p(i)$) $\supset$ ( $\forall\, n$ : $p(n)$)

induction_m: **Theorem**
$p(m) \wedge$ ( $\forall\, i$ : $i \geq m \wedge p(i) \supset p(i+1)$) $\supset$ ( $\forall\, n$ : $n \geq m \supset p(n)$)

limited_induction: **Theorem**
$(m \leq m_1 \supset p(m)) \wedge$ ( $\forall\, i$ : $i \geq m \wedge i < m_1 \wedge p(i) \supset p(i+1)$)
$\supset$ ( $\forall\, n$ : $n \geq m \wedge n \leq m_1 \supset p(n)$)

### Proof

**Using** noetherian

less: function[nat, nat $\rightarrow$ bool] == ( $\lambda\, m, n$ : $m < n$)

instance: **Module is** noetherian[nat, less]
$x$: **Var** nat
identity: function[nat $\rightarrow$ nat] == ( $\lambda\, n$ : $n$)

discharge: **Prove** well_founded {measure $\leftarrow$ identity}

complete_ind_pr: **Prove** complete_induction $\{i \leftarrow d_1@p1\}$ **from**
general_induction $\{d \leftarrow n,\ d_2 \leftarrow j\}$

ind_proof: **Prove** induction $\{j \leftarrow \text{pred}(d_1@p1)\}$ **from**
general_induction $\{p \leftarrow$ prop, $d \leftarrow i,\ d_2 \leftarrow j\}$

```
(* Substitution for n in following could simply be n <- n-m
   but then the TCC would not be provable *)
```

ind_m_proof: **Prove** induction_m $\{i \leftarrow j@p1 + m\}$ **from**
induction
{prop $\leftarrow$ ( $\lambda\, x$ : p@c($x + m$)),
$i \leftarrow$ **if** $n \geq m$ **then** $n - m$ **else** 0 **end if**}

limited_proof: **Prove** limited_induction $\{i \leftarrow i@p1\}$ **from**
induction_m $\{p \leftarrow$ ( $\lambda\, x$ : $x \leq m_1 \supset$ p@c($x$))$\}$

```
(*

  (* These results can also be proved the other way about but the
      TCCs are more complex *)

  alt_ind_m_proof: PROVE induction_m {i <- d1@p1 + m - 1} FROM
    general_induction
      {d <- n - m,
       d2 <- i - m,
       p <- (LAMBDA x : p@c(x + m))}

  alt_ind_proof: PROVE induction {i <- i@p1 - m@p1} FROM
    induction_m {p <- (LAMBDA x : p@c(x - m)), n <- n@c + m}
*)
```

**End** natinduction

## D.6 noetherian

noetherian: **Module** [dom: **Type**, <: function[dom, dom → bool]]

**Assuming**

measure: **Var** function[dom → nat]
$a, b$: **Var** dom

well_founded: **Formula** ( ∃ measure : $a < b \supset$ measure($a$) < measure($b$))

**Theory**

$p, A, B$: **Var** function[dom → bool]
$d, d_1, d_2$: **Var** dom

general_induction: **Axiom**
$(\forall\, d_1 : (\forall\, d_2 : d_2 < d_1 \supset p(d_2)) \supset p(d_1)) \supset (\forall\, d : p(d))$

$d_3, d_4$: **Var** dom

mod_induction: **Theorem**
$(\forall\, d_3, d_4 : d_4 < d_3 \supset A(d_3) \supset A(d_4))$
$\wedge\, (\forall\, d_1 : (\forall\, d_2 : d_2 < d_1 \supset (A(d_1) \wedge B(d_2))) \supset B(d_1))$
$\supset (\forall\, d : A(d) \supset B(d))$

**Proof**

mod_proof: **Prove** mod_induction
$\{d_1 \leftarrow d_1@p1,$
$d_3 \leftarrow d_1@p1,$
$d_4 \leftarrow d_2\}$ **from** general_induction $\{p \leftarrow (\lambda\, d : A(d) \supset B(d))\}$

**End** noetherian

## D.7 countmod

countmod: **Module**

**Exporting all**

**Theory**

$i_1$: **Var** int
posint: **Type from** nat **with** $(\lambda\, i_1 : i_1 > 0)$
$l, m, n, p, q, p_1, p_2, q_1, q_2, p_3, q_3$: **Var** nat
$i, j, k$: **Var** nat
$x, y, z, r, s, t$: **Var** number
$X, Y, Z$: **Var** number
ppred, ppred1, ppred2: **Var** function[nat → bool]
$\vartheta, \theta, \gamma$: **Var** function[nat → number]
countsize: function[function[nat → bool], nat → nat] = $(\lambda$ ppred, $i : i)$
count: **Recursive** function[function[nat → bool], nat → nat] =
  $(\lambda$ ppred, $i :$ ( **if** $i > 0$
    **then** ( **if** ppred$(i-1)$
      **then** $1 +$ (count(ppred, $i-1$))
      **else** count(ppred, $i-1$)
      **end if**)
    **else** 0
    **end if**)) **by** countsize
(* Count Complement was moved from ica3 *)

count_complement: **Lemma** count($(\lambda\, q : \neg$ppred$(q)), n) = n -$ count(ppred, $n$)

count_exists: **Lemma** count(ppred, $n) > 0 \supset (\exists\, p : p < n \wedge$ ppred$(p))$

count_true: **Lemma** count($(\lambda\, p :$ true$), n) = n$

count_false: **Lemma** count($(\lambda\, p :$ false$), n) = 0$

imp_pred: function[function[nat → bool], function[nat → bool] → bool] =
  $(\lambda$ ppred1, ppred2 : $(\forall\, p :$ ppred1$(p) \supset$ ppred2$(p)))$

imp_pred_lem: **Lemma** imp_pred(ppred1, ppred2) $\supset$ (ppred1$(p) \supset$ ppred2$(p))$

imp_pred_or: **Lemma** imp_pred(ppred1, $(\lambda\, p :$ ppred1$(p) \vee$ ppred2$(p)))$

count_imp: **Lemma** imp_pred(ppred1, ppred2)
  $\supset$ count(ppred1, $n) \leq$ count(ppred2, $n)$

count_or: **Lemma** count(ppred1, $n) \geq k$
  $\supset$ count($(\lambda\, p :$ ppred1$(p) \vee$ ppred2$(p)), n) \geq k$

count_bounded_imp: **Lemma** count($(\lambda\, p : p < n \supset$ ppred$(p)), n) =$ count(ppred, $n)$

count_bounded_and: **Lemma** count((λ p : p < n ∧ ppred(p)), n) = count(ppred, n)

pigeon_hole: **Lemma**
count(ppred1, n) + count(ppred2, n) ≥ n + k
⊃ count((λ p : ppred1(p) ∧ ppred2(p)), n) ≥ k

pred1, pred2: **Var** function[nat → bool]

pred_extensionality: **Axiom** (∀ p : pred1(p) = pred2(p)) ⊃ pred1 = pred2

```
(* these are in the theory section so the tcc module won't complain *)
```
nk_type: **Type** = **Record** n : nat,
k : nat
**end record**
nk, nk1, nk2: **Var** nk_type
nk_less: function[nk_type, nk_type → bool] ==
(λ nk1, nk2 : nk1.n + nk1.k < nk2.n + nk2.k)

## Proof

**Using** natinduction, noetherian

imp_pred_lem_pr: **Prove** imp_pred_lem **from** imp_pred {p ← p@c}

imp_pred_or_pr: **Prove** imp_pred_or **from**
imp_pred {ppred2 ← (λ p : ppred1(p) ∨ ppred2(p))}

count_imp0: **Lemma**
imp_pred(ppred1, ppred2) ⊃ count(ppred1, 0) ≤ count(ppred2, 0)

count_imp_ind: **Lemma**
(imp_pred(ppred1, ppred2) ⊃ count(ppred1, n) ≤ count(ppred2, n))
⊃ (imp_pred(ppred1, ppred2)
⊃ count(ppred1, n + 1) ≤ count(ppred2, n + 1))

count_imp0_pr: **Prove** count_imp0 **from**
count {i ← 0, ppred ← ppred1}, count {i ← 0, ppred ← ppred2}

count_imp_ind_pr: **Prove** count_imp_ind **from**
count {ppred ← ppred1, i ← n + 1},
count {ppred ← ppred2, i ← n + 1},
imp_pred {p ← n}

count_imp_pr: **Prove** count_imp **from**
  induction
    {prop $\leftarrow$ ( $\lambda\, n$ :
          (imp_pred(ppred1, ppred2) $\supset$ count(ppred1, $n$) $\leq$ count(ppred2, $n$))),
     $i \leftarrow n@c$},
  count_imp0,
  count_imp_ind $\{n \leftarrow j@p1\}$

count_or_pr: **Prove** count_or **from**
  count_imp {ppred2 $\leftarrow$ ( $\lambda\, p$ : ppred1($p$) $\vee$ ppred2($p$))}, imp_pred_or

count_bounded_imp0: **Lemma**
  $k \geq 0 \supset$ count(( $\lambda\, p : p < k \supset$ ppred($p$)), 0) = count(ppred, 0)

count_bounded_imp_ind: **Lemma**
  ($k \geq n \supset$ count(( $\lambda\, p : p < k \supset$ ppred($p$)), $n$) = count(ppred, $n$))
    $\supset$ ($k \geq n + 1$
        $\supset$ count(( $\lambda\, p : p < k \supset$ ppred($p$)), $n + 1$) = count(ppred, $n + 1$))

count_bounded_imp_k: **Lemma**
  ($k \geq n \supset$ count(( $\lambda\, p : p < k \supset$ ppred($p$)), $n$) = count(ppred, $n$))

count_bounded_imp0_pr: **Prove** count_bounded_imp0 **from**
  count $\{i \leftarrow 0\}$, count {ppred $\leftarrow$ ( $\lambda\, p : p < k \supset$ ppred($p$)), $i \leftarrow 0$}

count_bounded_imp_ind_pr: **Prove** count_bounded_imp_ind **from**
  count $\{i \leftarrow n + 1\}$,
  count {ppred $\leftarrow$ ( $\lambda\, p : p < k \supset$ ppred($p$)), $i \leftarrow n + 1$}

count_bounded_imp_k_pr: **Prove** count_bounded_imp_k **from**
  induction
    {prop $\leftarrow$ ( $\lambda\, n$ :
          $k \geq n \supset$ count(( $\lambda\, p : p < k \supset$ ppred($p$)), $n$) = count(ppred, $n$)),
     $i \leftarrow n$},
  count_bounded_imp0,
  count_bounded_imp_ind $\{n \leftarrow j@p1\}$

count_bounded_imp_pr: **Prove** count_bounded_imp **from**
  count_bounded_imp_k $\{k \leftarrow n\}$

count_bounded_and0: **Lemma**
  $k \geq 0 \supset$ count(( $\lambda\, p : p < k \wedge$ ppred($p$)), 0) = count(ppred, 0)

count_bounded_and_ind: **Lemma**
  ($k \geq n \supset$ count(( $\lambda\, p : p < k \wedge$ ppred($p$)), $n$) = count(ppred, $n$))
    $\supset$ ($k \geq n + 1$
        $\supset$ count(( $\lambda\, p : p < k \wedge$ ppred($p$)), $n + 1$) = count(ppred, $n + 1$))

count_bounded_and_k: **Lemma**
$(k \geq n \supset \mathsf{count}((\lambda\, p : p < k \wedge \mathsf{ppred}(p)), n) = \mathsf{count}(\mathsf{ppred}, n))$

count_bounded_and0_pr: **Prove** count_bounded_and0 **from**
count $\{i \leftarrow 0\}$, count $\{\mathsf{ppred} \leftarrow (\lambda\, p : p < k \wedge \mathsf{ppred}(p)),\ i \leftarrow 0\}$

count_bounded_and_ind_pr: **Prove** count_bounded_and_ind **from**
count $\{i \leftarrow n + 1\}$,
count $\{\mathsf{ppred} \leftarrow (\lambda\, p : p < k \wedge \mathsf{ppred}(p)),\ i \leftarrow n + 1\}$

count_bounded_and_k_pr: **Prove** count_bounded_and_k **from**
induction
$\{\mathsf{prop} \leftarrow (\lambda\, n :$
$k \geq n \supset \mathsf{count}((\lambda\, p : p < k \wedge \mathsf{ppred}(p)), n) = \mathsf{count}(\mathsf{ppred}, n)),$
$i \leftarrow n\}$,
count_bounded_and0,
count_bounded_and_ind $\{n \leftarrow j@p1\}$

count_bounded_and_pr: **Prove** count_bounded_and **from**
count_bounded_and_k $\{k \leftarrow n\}$

count_false_pr: **Prove** count_false **from**
count_true,
count_complement $\{\mathsf{ppred} \leftarrow (\lambda\, p : \mathsf{true})\}$,
pred_extensionality
$\{\mathsf{pred1} \leftarrow (\lambda\, p : \neg\mathsf{true}),$
$\mathsf{pred2} \leftarrow (\lambda\, p : \mathsf{false})\}$

cc0: **Lemma** $\mathsf{count}((\lambda\, q : \neg\mathsf{ppred}(q)), 0) = 0 - \mathsf{count}(\mathsf{ppred}, 0)$

cc_ind: **Lemma** $(\mathsf{count}((\lambda\, q : \neg\mathsf{ppred}(q)), n) = n - \mathsf{count}(\mathsf{ppred}, n))$
$\supset (\mathsf{count}((\lambda\, q : \neg\mathsf{ppred}(q)), n + 1) = n + 1 - \mathsf{count}(\mathsf{ppred}, n + 1))$

cc0_pr: **Prove** cc0 **from**
count $\{\mathsf{ppred} \leftarrow (\lambda\, q : \neg\mathsf{ppred}(q)),\ i \leftarrow 0\}$, count $\{i \leftarrow 0\}$

cc_ind_pr: **Prove** cc_ind **from**
count $\{\mathsf{ppred} \leftarrow (\lambda\, q : \neg\mathsf{ppred}(q)),\ i \leftarrow n + 1\}$, count $\{i \leftarrow n + 1\}$

count_complement_pr: **Prove** count_complement **from**
induction
$\{\mathsf{prop} \leftarrow (\lambda\, n : \mathsf{count}((\lambda\, q : \neg\mathsf{ppred}(q)), n) = n - \mathsf{count}(\mathsf{ppred}, n)),$
$i \leftarrow n\}$,
cc0,
cc_ind $\{n \leftarrow j@p1\}$

instance: **Module is** noetherian[nk_type, nk_less]
nk_measure: function[nk_type $\rightarrow$ nat] == $(\lambda\, \mathsf{nk1} : \mathsf{nk1}.n + \mathsf{nk1}.k)$

nk_well_founded: **Prove** well_founded {measure ← nk_measure}

nk_ph_pred: function[function[nat → bool], function[nat → bool], nk_type
→ bool] =
( λ ppred1, ppred2, nk :
count(ppred1, nk.$n$) + count(ppred2, nk.$n$) ≥ nk.$n$ + nk.$k$
⊃ count((λ $p$ : ppred1($p$) ∧ ppred2($p$)), nk.$n$) ≥ nk.$k$)
nk_noeth_pred: function[function[nat → bool], function[nat → bool],
nk_type → bool] =
( λ ppred1, ppred2, nk1 :
(∀ nk2 : nk_less(nk2, nk1) ⊃ nk_ph_pred(ppred1, ppred2, nk2)))

ph_case1: **Lemma** count((λ $p$ : ppred1($p$) ∧ ppred2($p$)), pred($n$)) ≥ $k$
⊃ count((λ $p$ : ppred1($p$) ∧ ppred2($p$)), $n$) ≥ $k$

ph_case1_pr: **Prove** ph_case1 **from**
count {ppred ← (λ $p$ : ppred1($p$) ∧ ppred2($p$)), $i$ ← $n$}

ph_case2: **Lemma** count(ppred1, pred($n$)) + count(ppred2, pred($n$)) < pred($n$) + $k$
∧ count(ppred1, $n$) + count(ppred2, $n$) ≥ $n$ + $k$
∧ count((λ $p$ : ppred1($p$) ∧ ppred2($p$)), pred($n$)) ≥ pred($k$)
⊃ count((λ $p$ : ppred1($p$) ∧ ppred2($p$)), $n$) ≥ $k$

ph_case2a: **Lemma** count(ppred1, pred($n$)) + count(ppred2, pred($n$)) < pred($n$) + $k$
∧ count(ppred1, $n$) + count(ppred2, $n$) ≥ $n$ + $k$
⊃ ppred1(pred($n$)) ∧ ppred2(pred($n$))

ph_case2b: **Lemma** $n > 0$
∧ $k > 0$ ∧ count(ppred1, pred($n$)) + count(ppred2, pred($n$)) < pred($n$) + $k$
∧ count(ppred1, $n$) + count(ppred2, $n$) ≥ $n$ + $k$
⊃ count(ppred1, pred($n$)) + count(ppred2, pred($n$)) ≥ pred($n$) + pred($k$)

ph_case2a_pr: **Prove** ph_case2a **from**
count {ppred ← ppred1, $i$ ← $n$}, count {ppred ← ppred2, $i$ ← $n$}

ph_case2b_pr: **Prove** ph_case2b **from**
count {ppred ← ppred1, $i$ ← $n$}, count {ppred ← ppred2, $i$ ← $n$}

ph_case2_pr: **Prove** ph_case2 **from**
count {ppred ← (λ $p$ : ppred1($p$) ∧ ppred2($p$)), $i$ ← $n$}, ph_case2a

ph_case0: **Lemma** ($n = 0 \vee k = 0$)
⊃ (count(ppred1, $n$) + count(ppred2, $n$) ≥ $n$ + $k$
⊃ count((λ $p$ : ppred1($p$) ∧ ppred2($p$)), $n$) ≥ $k$)

ph_case0n: **Lemma** (count(ppred1, 0) + count(ppred2, 0) ≥ $k$
⊃ count((λ $p$ : ppred1($p$) ∧ ppred2($p$)), 0) ≥ $k$)

ph_case0n_pr: **Prove** ph_case0n **from**
  count $\{$ppred $\leftarrow$ ppred1, $i \leftarrow 0\}$,
  count $\{$ppred $\leftarrow$ ppred2, $i \leftarrow 0\}$,
  count $\{$ppred $\leftarrow (\lambda\, p$ : ppred1$(p) \wedge$ ppred2$(p))$, $i \leftarrow 0\}$

ph_case0k: **Lemma** count$((\lambda\, p$ : ppred1$(p) \wedge$ ppred2$(p)), n) \geq 0$

ph_case0k_pr: **Prove** ph_case0k **from**
  nat_invariant $\{$nat_var $\leftarrow$ count$((\lambda\, p$ : ppred1$(p) \wedge$ ppred2$(p)), n)\}$

ph_case0_pr: **Prove** ph_case0 **from** ph_case0n, ph_case0k

nk_ph_expand: **Lemma**
  $(\forall\, n, k$ : (count(ppred1, pred$(n)$) + count(ppred2, pred$(n)$) $\geq$ pred$(n)$ + pred$(k)$
      $\supset$ count$((\lambda\, p$ : ppred1$(p) \wedge$ ppred2$(p))$, pred$(n)) \geq$ pred$(k))$
    $\wedge$ (count(ppred1, pred$(n)$) + count(ppred2, pred$(n)$) $\geq$ pred$(n) + k$
      $\supset$ count$((\lambda\, p$ : ppred1$(p) \wedge$ ppred2$(p))$, pred$(n)) \geq k)$
  $\supset$ (count(ppred1, $n$) + count(ppred2, $n$) $\geq n + k$
    $\supset$ count$((\lambda\, p$ : ppred1$(p) \wedge$ ppred2$(p)), n) \geq k))$

nk_ph_expand_pr: **Prove** nk_ph_expand **from**
  ph_case0, ph_case1, ph_case2, ph_case2a, ph_case2b

nk_ph_noeth_hyp: **Lemma**
  $(\forall$ nk1 : nk_noeth_pred(ppred1, ppred2, nk1)
    $\supset$ nk_ph_pred(ppred1, ppred2, nk1))

nk_ph_noeth_hyp_pr: **Prove** nk_ph_noeth_hyp **from**
  nk_ph_pred $\{$nk $\leftarrow$ nk1$\}$,
  nk_noeth_pred $\{$nk2 $\leftarrow$ nk1 **with** $[(n) :=$ pred(nk1.$n)]\}$,
  nk_noeth_pred $\{$nk2 $\leftarrow$ nk1 **with** $[(n) :=$ pred(nk1.$n), (k) :=$ pred(nk1.$k)]\}$,
  nk_ph_pred $\{$nk $\leftarrow$ nk1 **with** $[(n) :=$ pred(nk1.$n)]\}$,
  nk_ph_pred $\{$nk $\leftarrow$ nk1 **with** $[(n) :=$ pred(nk1.$n), (k) :=$ pred(nk1.$k)]\}$,
  nk_ph_expand $\{n \leftarrow$ nk1.$n$, $k \leftarrow$ nk1.$k\}$,
  ph_case0 $\{n \leftarrow$ nk1.$n$, $k \leftarrow$ nk1.$k\}$,
  nat_invariant $\{$nat_var $\leftarrow$ nk1.$n\}$,
  nat_invariant $\{$nat_var $\leftarrow$ nk1.$k\}$

nk_ph_lem: **Lemma** nk_ph_pred(ppred1, ppred2, nk)

nk_ph_lem_pr: **Prove** nk_ph_lem **from**
  general_induction
    $\{p \leftarrow (\lambda$ nk : nk_ph_pred(ppred1, ppred2, nk)),
     $d_2 \leftarrow$ nk2@$p3$,
     $d \leftarrow$ nk@$c\}$,
  nk_ph_noeth_hyp $\{$nk1 $\leftarrow d_1$@$p1\}$,
  nk_noeth_pred $\{$nk1 $\leftarrow d_1$@$p1\}$

pigeon_hole_pr: **Prove** pigeon_hole **from**
nk_ph_lem $\{$nk $\leftarrow$ nk **with** $[(n) := n@c, (k) := k@c]\}$,
nk_ph_pred $\{$nk $\leftarrow$ nk$@p1\}$

exists_less: function[function[nat $\rightarrow$ bool], nat $\rightarrow$ bool] =
$(\lambda$ ppred, $n : (\exists\, p : p < n \wedge$ ppred$(p)))$

count_exists_base: **Lemma** count(ppred, 0) $> 0 \supset$ exists_less(ppred, 0)

count_exists_base_pr: **Prove** count_exists_base **from**
count $\{i \leftarrow 0\}$, exists_less $\{n \leftarrow 0\}$

count_exists_ind: **Lemma**
(count(ppred, $n$) $> 0 \supset$ exists_less(ppred, $n$))
$\supset$ (count(ppred, $n+1$) $> 0 \supset$ exists_less(ppred, $n+1$))

count_exists_ind_pr: **Prove** count_exists_ind **from**
count $\{i \leftarrow n+1\}$,
exists_less,
exists_less $\{n \leftarrow n+1,\ p \leftarrow$ ( **if** ppred$(n)$ **then** $n$ **else** $p@p2$ **end if**$)\}$

count_exists_pr: **Prove** count_exists $\{p \leftarrow p@p4\}$ **from**
induction
$\{$prop $\leftarrow (\lambda\, n :$ count(ppred, $n) > 0 \supset$ exists_less(ppred, $n)),$
$i \leftarrow n@c\}$,
count_exists_base,
count_exists_ind $\{n \leftarrow j@p1\}$,
exists_less $\{n \leftarrow i@p1\}$

count_base: **Sublemma** count(ppred, 0) $= 0$

count_base_pr: **Prove** count_base **from** count $\{i \leftarrow 0\}$

count_true_ind: **Sublemma**
(count$((\lambda\, p :$ true$), n) = n) \supset$ count$((\lambda\, p :$ true$), n+1) = n+1$

count_true_ind_pr: **Prove** count_true_ind **from**
count $\{$ppred $\leftarrow (\lambda\, p :$ true$),\ i \leftarrow n+1\}$

count_true_pr: **Prove** count_true **from**
induction $\{$prop $\leftarrow (\lambda\, n :$ count$((\lambda\, p :$ true$), n) = n),\ i \leftarrow n@c\}$,
count_base $\{$ppred $\leftarrow (\lambda\, p :$ true$)\}$,
count_true_ind $\{n \leftarrow j@p1\}$

**End** countmod

# References


[1] Lamport, Leslie; and Melliar-Smith, P. M.: Synchronizing Clocks in the Presence of Faults. *J. Assoc. Comput. Mach.*, vol. 32, no. 1, Jan. 1985, pp. 52–78.

[2] Welch, Jennifer Lundelius; and Lynch, Nancy: A New Fault-Tolerant Algorithm for Clock Synchronization. *Inf. & Comput.*, vol. 77, no. 1, Apr. 1988, pp. 1–36.

[3] Dolev, Danny; Halpern, Joseph Y.; and Strong, H. Raymond: On the Possibility and Impossibility of Achieving Clock Synchronization. *J. Comput. & Syst. Sci.*, vol. 32, 1986, pp. 230–250.

[4] Halpern, Joseph Y.; Simons, Barbara; Strong, Ray; and Dolev, Danny: Fault-Tolerant Clock Synchonization. *Proceedings of the Third ACM Symposium on Principles of Distributed Computing*, Assoc. for Computing Machinery, 1984, pp. 89–102.

[5] Dolev, Danny; Lynch, Nancy A.; Pinter, Shlomit S.; Stark, Eugene W.; and Weihl, William E.: Reaching Approximate Agreement in the Presence of Faults. *J. Assoc. Comput. Mach.*, vol. 33, no. 3, July 1986, pp. 499–516.

[6] Srikanth, T. K.; and Toueg, Sam: Optimal Clock Synchronization. *J. Assoc. Comput. Mach.*, vol. 34, no. 3, July 1987, pp. 626–645.

[7] Mahaney, Stephen R.; and Schneider, Fred B.: *Inexact Agreement: Accuracy, Precision, and Graceful Degradation.* TR 85-683, Cornell Univ., May 1985. (Presented at 4th ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing (Ontario, Canada), Aug. 1985.)

[8] Ramanathan, Parameswaran; Shin, Kang G.; and Butler, Ricky W.: Fault-Tolerant Clock Synchronization in Distributed Systems. *Computer*, vol. 23, no. 10, Oct. 1990, pp. 33–42.

[9] Schneider, Fred B.: *Understanding Protocols for Byzantine Clock Synchronization.* Tech. Rep. 87-859 (NSF Grant DCR-8320274 and Office of Naval Research Contract N00014-86-K-0092), Cornell Univ., Aug. 1987.

[10] Shankar, Natarajan: *Mechanical Verification of a Schematic Byzantine Clock Synchronization Algorithm.* NASA CR-4386, 1991.

[11] Rushby, John; Von Henke, Friedrich; and Owre, Sam: *An Introduction to Formal Specification and Verification Using* EHDM. SRI-CSL-91-02, SRI International, Feb. 1991.

[12] Lamport, Leslie; Shostak, Robert; and Pease, Marshall: The Byzantine Generals Problem. *ACM Trans. Program. Lang. & Syst.*, vol. 4, no. 3, July 1982, pp. 382–401.

[13] Mackall, Dale A.: *Development and Flight Test Experiences With a Flight-Crucial Digital Control System.* NASA TP-2857, 1988.

[14] *System Design and Analysis.* AC No. 25.1309-1A, Federal Aviation Adm., June 21, 1988.

[15] DiVito, Ben L.; Butler, Ricky W.; and Caldwell, James L.: *Formal Design and Verification of a Reliable Computing Platform for Real-Time Controls, Phase 1: Results.* NASA TM-102716, 1990.

[16] Butler, Ricky W.; and DiVito, Ben L.: *Formal Design and Verification of a Reliable Computing Platform for Real-Time Control, Phase 2: Results.* NASA TM-104196, 1992.

[17] Rushby, John: *Formal Specification and Verification of a Fault-Masking and Transient-Recovery Model for Digital Flight-Control Systems.* NASA CR-4384, 1991.

[18] Rushby, John; and von Henke, Friedrich: *Formal Verification of a Fault Tolerant Clock Synchronization Algorithm.* NASA CR-4239, 1989.

[19] Gouda, Mohamed G.; and Multari, Nicholas J.: Stabilizing Communication Protocols. *IEEE Trans. Comput.*, vol. 40, no. 4, Apr. 1991, pp. 448–458.

[20] Kieckhafer, Roger M.; Walter, Chris J.; Finn, Alan M.; and Thambidurai, Philip M.: The MAFT Architecture for Distributed Fault Tolerance. *IEEE Trans. Comput.*, vol. 37, no. 4, Apr. 1988, pp. 398–405.

[21] Miner, Paul S.: *A Verified Design of a Fault-Tolerant Clock Synchronization Circuit: Preliminary Investigations.* NASA TM-107568, 1992.

[22] Barendregt, H. P.: *The Lambda Calculus—Its Syntax and Semantics*, Revised ed. Elsevier Science Publ. Co., 1984.

[23] Miner, Paul S.: *An Extension to Schneider's General Paradigm for Fault-Tolerant Clock Synchronization.* NASA TM-107634, 1992.

[24] Miner, Paul S.; Padilla, Peter A.; and Torres, Wilfredo: A Provably Correct Design of a Fault-Tolerant Clock Synchronization Circuit. *Proceedings IEEE/AIAA 11th Digital Avionics Systems Conference*, IEEE Catalog No. 92CH3212-8, Inst. of Electrical and Electronics Engineers, Inc., 1992, pp. 341–346.

[25] Moore, J. Strother: *A Formal Model of Asynchronous Communication and Its Use in Mechanically Verifying a Biphase Mark Protocol.* NASA CR-4433, 1992.

[26] Srivas, Mandayam; and Bickford, Mark: *Verification of the FtCayuga Fault-Tolerant Microprocessor System. Volume 1: A Case Study in Theorem Prover-Based Verification.* NASA CR-4381, 1991.

[27] Bevier, William R.; and Young, William D.: *Machine Checked Proofs of the Design and Implementation of a Fault-Tolerant Circuit.* NASA CR-182099, 1990.

# REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188



| 1. AGENCY USE ONLY *(Leave blank)* | 2. REPORT DATE | 3. REPORT TYPE AND DATES COVERED |
|---|---|---|
| | November 1993 | Technical Paper |

**4. TITLE AND SUBTITLE**
Verification of Fault-Tolerant Clock Synchronization Systems

**5. FUNDING NUMBERS**
WU 505-64-50-03

**6. AUTHOR(S)**
Paul S. Miner

**7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)**
NASA Langley Research Center
Hampton, VA 23681-0001

**8. PERFORMING ORGANIZATION REPORT NUMBER**
L-17209

**9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)**
National Aeronautics and Space Administration
Washington, DC 20546-0001

**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**
NASA TP-3349

**11. SUPPLEMENTARY NOTES**
The information presented in this report was included in a thesis offered in partial fulfillment of the requirements for the Degree of Master of Science, The College of William and Mary in Virginia, Williamsburg, VA, 1992.

**12a. DISTRIBUTION/AVAILABILITY STATEMENT**
Unclassified–Unlimited

Subject Category 62

**12b. DISTRIBUTION CODE**

**13. ABSTRACT** *(Maximum 200 words)*

A critical function in a fault-tolerant computer architecture is the synchronization of the redundant computing elements. The synchronization algorithm must include safeguards to ensure that failed components do not corrupt the behavior of good clocks. Reasoning about fault-tolerant clock synchronization is difficult because of the possibility of subtle interactions involving failed components. Therefore, mechanical proof systems are used to ensure that the verification of the synchronization system is correct. In 1987, Schneider presented a general proof of correctness for several fault-tolerant clock synchronization algorithms. Subsequently, Shankar verified Schneider's proof by using the mechanical proof system EHDM. This proof ensures that any system satisfying its underlying assumptions will provide Byzantine fault-tolerant clock synchronization. This paper explores the utility of Shankar's mechanization of Schneider's theory for the verification of clock synchronization systems. In the course of this work, some limitations of Shankar's mechanically verified theory were encountered. With minor modifications to the theory, a mechanically checked proof is provided that removes these limitations. The revised theory also allows for proven recovery from transient faults. Use of the revised theory is illustrated with the verification of an abstract design of a clock synchronization system.

**14. SUBJECT TERMS**
Fault tolerance; Clock synchronization; Formal methods; Mechanized proof; Transient faults

**15. NUMBER OF PAGES**
142

**16. PRICE CODE**
A07

| 17. SECURITY CLASSIFICATION OF REPORT | 18. SECURITY CLASSIFICATION OF THIS PAGE | 19. SECURITY CLASSIFICATION OF ABSTRACT | 20. LIMITATION OF ABSTRACT |
|---|---|---|---|
| Unclassified | Unclassified | | |