

1992

Verification of Fault-Tolerant Clock Synchronization Systems

Paul S. Miner

College of William & Mary - Arts & Sciences

Follow this and additional works at: <https://scholarworks.wm.edu/etd>



Part of the [Computer Sciences Commons](#)

Recommended Citation

Miner, Paul S., "Verification of Fault-Tolerant Clock Synchronization Systems" (1992). *Dissertations, Theses, and Masters Projects*. Paper 1539625738.

<https://dx.doi.org/doi:10.21220/s2-a74q-7r46>

This Thesis is brought to you for free and open access by the Theses, Dissertations, & Master Projects at W&M ScholarWorks. It has been accepted for inclusion in Dissertations, Theses, and Masters Projects by an authorized administrator of W&M ScholarWorks. For more information, please contact scholarworks@wm.edu.

Verification of Fault-Tolerant Clock Synchronization Systems

A Thesis

Presented to

The Faculty of the Department of Computer Science

The College of William and Mary in Virginia

In Partial Fulfillment

Of the Requirements for the Degree of

Master of Science

by

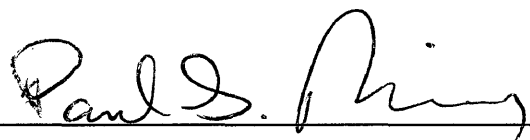
Paul S. Miner

1992

APPROVAL SHEET

This thesis is submitted in partial fulfillment of
the requirements for the degree of

Master of Science

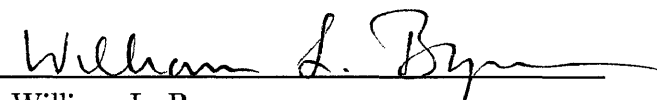


Author

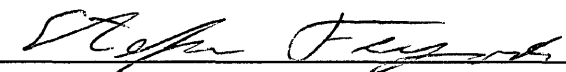
Approved, July 1992



Joe P. Kearns



William L. Bynum



Stefan Feyock

Contents

Acknowledgments	vii
List of Tables	viii
List of Figures	ix
Abstract	x
1 Introduction	1
2 Clock Definitions	4
2.1 Notation	5
2.2 The Conditions	9
3 A General Solution for Bounded Delay	20
3.1 Bounded Delay Offset	21
3.2 Bounded Delay for Two Algorithm Schemata	26
3.3 EHDM Proofs of Bounded Delay	27
3.4 New Theory Obligations	30
4 Fault-Tolerant Midpoint as an Instance of Schneider’s Schema	32
4.1 Translation Invariance	33
4.2 Precision Enhancement	33

4.3	Accuracy Preservation	37
4.4	EHDM Proofs of Convergence Properties	38
5	Design of a Clock Synchronization System	41
5.1	Description of Design	41
5.2	Theory Obligations	45
6	Initialization and Transient Recovery	49
6.1	Initial Synchronization	50
6.1.1	Mechanisms for Initialization	51
6.1.2	Comparison to Other Approaches	57
6.2	Transient Recovery	58
6.2.1	Theory Considerations	58
6.2.2	Satisfying <code>rpred</code>	60
6.2.3	Comparison with Other Approaches	61
7	Concluding Remarks	63
A	Proof of Agreement	66
A.1	Proof Sketch of Agreement	66
A.2	EHDM Extracts	69
A.2.1	Proof Chain Analysis	69
A.2.2	Module <code>lemma_final</code>	71
A.2.3	Module <code>clockassumptions</code>	73
B	Bounded Delay Modules	78
B.1	Proof Analysis	78
B.1.1	Proof Chain for <code>delay4</code>	78
B.1.2	Proof Chain for <code>rmax_rmin</code>	80
B.1.3	Proof Chain for <code>new_basics</code>	81

B.2	delay	83
B.3	delay2	92
B.4	delay3	100
B.5	delay4	106
B.6	new_basics	109
B.7	rmax_rmin	112
C	Fault-Tolerant Midpoint Modules	117
C.1	Proof Analysis	117
C.1.1	Proof Chain for Translation Invariance	117
C.1.2	Proof Chain for Precision Enhancement	118
C.1.3	Proof Chain for Accuracy Preservation	118
C.2	mid	120
C.3	mid2	121
C.4	mid3	123
C.5	mid4	128
C.6	select_defs	130
C.7	ft_mid_assume	131
C.8	clocksort	132
D	Utility Modules	133
D.1	multiplication	134
D.2	division	136
D.3	absmod	139
D.4	floor_ceil	141
D.5	natinduction	144
D.6	noetherian	146
D.7	countmod	147

Acknowledgments

I am grateful to my colleagues at NASA Langley Research Center for providing an environment in which this research was possible. In particular, I would like to thank Dr. Allan White for patiently listening to my ramblings, Dr. Ben DiVito for his insightful comments, and Ricky Butler for pointers on interacting with EHDm.

I would like to thank my adviser, Dr. J. Philip Kearns, for always asking the right questions. I am also indebted to Dr. William L. Bynum and Dr. Stefan Feyock for carefully reading the manuscript.

Finally, I would like to thank my wife Leah, without whom none of this would have been possible.

List of Tables

2.1	Clock Notation	10
2.2	Constants	10

List of Figures

2.1	Four Clock System	5
5.1	Informal Block Model	44
6.1	Synchronization Interval	50
6.2	Pathological Scenario	52
6.3	End of Interval Initialization	53
6.4	Pathological End of Interval Initialization	54
6.5	End of Interval Initialization—Time Out	55
6.6	End of Interval Initialization: d Faulty—benign	56
6.7	End of Interval Initialization: d Faulty—malicious	57

Abstract

An important function in the design of a fault-tolerant computer system is the synchronization of the clocks of the redundant processing elements. Due to the subtleties involved in reasoning about the behavior of failed components, it is necessary to prove that systems purporting to be fault-tolerant will survive an arbitrary failure.

In 1987, Schneider presented a general proof of correctness encompassing several fault-tolerant clock synchronization algorithms. Subsequently, Shankar verified Schneider's proof using the mechanical proof system EHDM. This proof ensures that any system satisfying its underlying assumptions will provide Byzantine fault-tolerant clock synchronization. This thesis explores the utility of Shankar's mechanization of Schneider's theory for the verification of clock synchronization systems.

A mechanically checked proof is presented which provides a general solution for one constraint of the existing theory. Also, the fault-tolerant midpoint convergence function is proven, using EHDM, to satisfy the requirements of the theory. Other constraints are modified to provide simpler verification conditions. Furthermore, the theory is extended to allow general proofs of transient fault recovery. Use of the revised theory is then illustrated with the verification of an abstract design of a fault-tolerant clock synchronization system.

Chapter 1

Introduction

NASA Langley Research Center is currently involved in the development of a formally verified Reliable Computing Platform (RCP) for real-time digital flight control systems [1, 2, 3]. 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 [4]. Since failure rates for digital devices are on the order of 10^{-6} per hour [5], hardware redundancy is required to achieve the desired level of reliability. While there are many ways of incorporating redundant hardware, the approach taken in the RCP is the use of identical redundant channels with exact match voting (see [1, 2] and [3]).

A critical function in a fault-tolerant system is that of synchronizing the clocks of the redundant computing elements. The clocks must be synchronized in order to provide coordinated action among the redundant sites. Although perfect synchronization is not possible, clocks can be synchronized within a small skew.

Schneider [6] demonstrates that many fault-tolerant clock synchronization algorithms can be represented as refinements of a single proven correct paradigm. Shankar [7] provides a mechanical proof (using EHDM [8]) that Schneider's schema achieves Byzantine fault-tolerant clock synchronization, provided that eleven constraints are satisfied. Some of the constraints are assumptions about physical properties of the system and can not be

established formally. This thesis explores the utility of Shankar’s mechanically verified theory as a top-level specification for a fault-tolerant clock synchronization system. First, some of the assumptions employed by Shankar are addressed in a general fashion, and then an abstract design of a fault-tolerant clock synchronization circuit is shown to satisfy the necessary constraints of the theory.

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 which 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 [2]. It is desirable that the clock synchronization circuitry also be able to recover from transient faults. Originally, Lamport and Melliar-Smith’s Interactive Convergence Algorithm (ICA) [9] 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 [10]. However, modifications to ICA to achieve transient fault recovery are unnecessarily complicated. The fault-tolerant midpoint algorithm of [11] is more readily adapted to transient recovery.

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 faults which do not constantly alter the function of a device (e.g. a loose wire). A transient fault is a one shot short duration physical perturbation of a device (e.g. caused by a cosmic ray or other electromagnetic effect). Once the source of the fault is removed, the device can function correctly.

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 [6, 7, 9, 10] all assume initial synchronization with no mention of how it is achieved. Others, including [11, 12, 13] and [14] 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 is noted in [14]).

The chapters in this thesis are ordered 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 Shankar's proof are presented. Also in Chapter 2, the additional definitions and constraints required for a general extension to the theory are introduced. Chapter 3 presents a general extension to the theory which should simplify future verification efforts. 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. It is shown that this circuit satisfies the remaining constraints of the theory. Finally, the mechanisms for achieving initial synchronization and transient recovery are presented. Modifications to the theory to support the transient recovery arguments are also presented.

Chapter 2

Clock Definitions

Any implementation that satisfies the definitions and constraints in Shankar's report will provide the following guarantee [7].

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

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

That is, the difference in time observed by two nonfaulty clocks is bounded by a small amount. This gives the leverage needed to reliably build a fault-tolerant system. Figure 2.1 illustrates a possible four clock system. Each of the nonfaulty clocks provides a time reference, VC_p , to its processing element. This reference is guaranteed to be approximately synchronized with the corresponding value on any other good clock in the system, for all time t . 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}
```

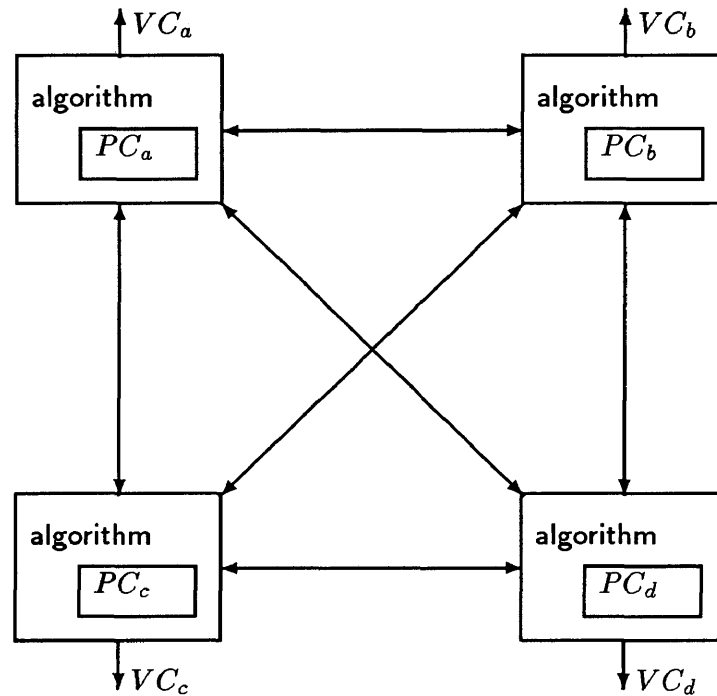


Figure 2.1: Four Clock System

This chapter presents the definitions and conditions to be met to verify this result. Much of it is taken from sections 2.1 and 2.2 of Shankar's report documenting his mechanization of Schneider's proof [7]. Modifications to the conditions needed for this revision of the theory are also presented.

2.1 Notation

A fault-tolerant clock synchronization system is composed of an interconnected collection of physically isolated clocks. Each redundant clock will incorporate a physical oscillator which marks passage of time. Each oscillator will drift with respect to real time by a small amount. Physical clocks derived from these oscillators will similarly drift with respect to each other. There are two different views of physical clocks relating different perceptions of time. Real time will be denoted by lower case letters, e.g. t, s : **Var** 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**. While **Clocktime** is often treated as ranging over the reals [11, 7, 10], a physical realization of a clock marks time in discrete intervals. It is more appropriate to treat values of type **Clocktime** as representing some integral number of ticks; 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 clocks¹: functions mapping real time to clock time for each process p ,

$$PC_p : \text{time} \rightarrow \text{Clocktime};$$

and functions mapping clock time to real time,

$$pc_p : \text{Clocktime} \rightarrow \text{time}.$$

The intended semantics are for $PC_p(t)$ to represent the reading of p 's clock at real time t , and for $pc_p(T)$ to denote 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, so an interval clock abstraction is used in the proofs. Each process p will have an infinite number of interval clocks associated with it, each of these will be 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 a process' physical clock plus an offset. As with the physical clocks, they are characterized by two functions: $IC_p^i : \text{time} \rightarrow \text{Clocktime}$; and $ic_p^i : \text{Clocktime} \rightarrow \text{time}$. If we let $adj_p^i : \text{Clocktime}$ denote the cumulative adjustment

¹Shankar's presentation includes only the mappings from time to **Clocktime**. The mappings from **Clocktime** to time are added here because it is a more natural representation for some of the proofs.

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, $VC_p : \text{time} \rightarrow \text{Clocktime}$, is defined in terms of the interval clocks by the equation

$$VC_p(t) = IC_p^i(t), \text{ for } t_p^i \leq t < t_p^{i+1}.$$

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. This is because VC_p is not necessarily monotonic; the inverse relation might not be a function for some synchronization protocols.

Synchronization protocols provide a mechanism for processes to read each others clocks. The adjustment is computed as a function of these readings. In Shankar's presentation, the readings of remote clocks are captured in function $\Theta_p^{i+1} : \text{process} \rightarrow \text{Clocktime}$, 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, $cf_n : (\text{process}, (\text{process} \rightarrow \text{Clocktime})) \rightarrow \text{Clocktime}$, to determine the proper correction to

apply. Shankar defines the cumulative adjustment in terms of the convergence function as follows:

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

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 the next section, 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 (i.e. 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 to equal $ic_p^i(S^i)$. 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 there exists some real time t , 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 (due to the clock being adjusted back). In [11], 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 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 the next section. They will be described when they are introduced in the text, but are included here as a convenient reference.

2.2 The Conditions

This section introduces the conditions required by Shankar's mechanical proof of Schneider's Theory. The changes needed for the general extension to the theory are also introduced here. The old conditions are those from Shankar's mechanization of Schneider's theory [7]. The order in which Shankar presented them is preserved for convenient reference to his report. However, this makes the presentation of the revised (new) conditions awkward. Much of the required notation for the revised conditions require a forward reference. Table 2.2 should provide an intuitive feel for some terms that have not yet been fully

$PC_p(t)$	The reading of p 's physical clock at real time t .
$pc_p(T)$	The earliest real time that p 's physical clock reads T .
$IC_p^i(t)$	The reading of p 's i th interval clock at real time t .
$ic_p^i(T)$	The earliest real time that p 's i th interval clock reads T .
$VC_p(t)$	The 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	The real time that processor p begins the i th synchronization interval ($t_p^{i+1} = ic_p^i(T_p^{i+1})$).
R	Clocktime duration of a 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 $adj_p^{i+1} - adj_p^i$.
Θ_p^{i+1}	An 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 2.1: Clock Notation

δ_S : Clocktime	Bound on skew at beginning of protocol.
δ : Clocktime	Bound on skew for all time.
ρ : number	Allowable drift rate for a good clock, $0 \leq \rho \ll 1$.
β' : time	Maximum elapsed time from s_p^i to s_q^i (p and q working).
β : time	Maximum elapsed time from t_p^i to t_q^i (p and q working).
β_{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 .
Λ : Clocktime	Bound on error reading a remote clock.
Λ' : 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 .

Table 2.2: Constants

developed in the text. Where possible, it will be shown how some of the old conditions can be derived from the new.

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

$$|PC_p(0) - PC_q(0)| \leq \delta_S$$

This condition will be replaced by the following:

New Condition 1 (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 the original condition can be easily derived from this new condition using the constraint on clock drift². An immediate consequence of this and the revised form of condition 2 is that $|s_p^0 - s_q^0| \leq \beta'$.

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

Old Condition 2 (bounded drift) *There is a nonnegative constant ρ such that if clock p is nonfaulty at time $s, s \geq t$, then*

$$(1 - \rho)(s - t) \leq PC_p(s) - PC_p(t) \leq (1 + \rho)(s - t)$$

This characterization of drift is not quite accurate, and is only valid if Clocktime ranges over the rationals or reals. If we treat Clocktime as an integer, the inequality does not hold for all s, t , or ρ . We restate the condition for the mapping from Clocktime to time. To allow for future modifications to the theory which allow for recovery from transient faults, we also remove the implicit assumption that nonfaulty clocks have been so since the beginning of the protocol.

²Old Condition 1 is an immediate consequence of Lemma 2.1.1 in Appendix A

New Condition 2 (bounded drift) *There is a nonnegative constant ρ such that if p 's clock is nonfaulty during the interval from T to S , ($S \geq T$), then*

$$(S - T)/(1 + \rho) \leq pc_p(S) - pc_p(T) \leq (1 + \rho)(S - T)$$

The benefit of changing the lower bound to $(S - T)/(1 + \rho)$ is that we can derive the following constraint on the mapping from time to Clocktime:

Corollary 2.1 *If p 's clock is nonfaulty during the interval from $pc_p(T)$ to $pc_p(S)$, $S \geq T$,*

$$(pc_p(S) - pc_p(T))/(1 + \rho) \leq PC_p(pc_p(S)) - PC_p(pc_p(T)) \leq (1 + \rho)(pc_p(S) - pc_p(T))$$

This is not as strong an assumption as Shankar's original condition. 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 that is sufficient for preserving Shankar's mechanical proof (with minor modifications):

Corollary 2.2 *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 using 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 2.3 *If clocks p and q are not faulty during the interval from T to S ,*

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

Shankar stated the above corollary with respect to the original formulation of bounded drift.

We can also derive an additional corollary (this adapted from lemma 2 of [11]).

Corollary 2.4 *If clock p is not faulty during the interval from T to S ,*

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

A similar relation holds for PC .

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

Old 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 our formulation, we assume that a nominal duration (R) of an interval is determined from the implementation. We set a lower bound on R by placing restrictions on the events S^i . This is done by bounding the amount of adjustment that a nonfaulty process can apply in any synchronization interval. The term $\alpha(\beta' + 2\Lambda')$ will be shown to bound $|ADJ_p^i|$ for nonfaulty process p . The function α is introduced in condition 11, β' is a bound on the separation of clocks at a particular **Clocktime** in each interval, and Λ' bounds the error in estimating the value of a remote clock.

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

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

A trivial consequence is that $R > 2\alpha(\beta' + 2\Lambda')$. Clearly, we can let $r_{min} = (R - \alpha(\beta' + 2\Lambda'))/(1 + \rho)$ and $r_{max} = (1 + \rho)(R + \alpha(\beta' + 2\Lambda'))$. The values for ρ , R , Λ' , β' , and $\alpha()$ will be determined by the implementation. The constraints on these values will be presented later.

Shankar and Schneider both assume the following in their proofs. 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.

Old Condition 4 (bounded delay) *For nonfaulty clocks p and q*

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

The related property, that 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 upon 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).

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

$$t_p^0 = 0$$

This is clearly unsatisfiable, and will be discarded. It is used in proving the base case of the induction proof which establishes that good clocks are within δ_S of other good clocks, immediately following applying a correction. By defining $t_p^0 = ic_p^0(T^0)$ we gain sufficient leverage for that proof. T^0 is some constant clock time known to all good clocks (i.e. T^0 is the clock time in the initial state). This just states that all nonfaulty clocks start the protocol at the same Clocktime.

Since we do not want process q to start its $(i + 1)$ th clock before process p starts its

i th, Shankar states a nonoverlap condition

Old Condition 6 (nonoverlap)

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

This, with *bounded interval* and *bounded delay*, ensures that for good clocks p and q , $t_p^i \leq t_q^{i+1}$. We restate the condition in terms related to this presentation

New Condition 6 (nonoverlap)

$$\beta \leq (R - \alpha(\beta' + 2\Lambda')) / (1 + \rho)$$

This essentially defines an additional constraint on R ; namely, that $R \geq (1 + \rho)\beta + \alpha(\beta' + 2\Lambda')$.

All clock synchronization protocols require each process to obtain an estimate of the clock values for other processes within the system. Error in this estimate can be bounded, but not eliminated.

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

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

However, in stating this condition an important consideration was overlooked. In some protocols, the ability to accurately read another processor's clock is dependent upon those 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 β_{read} . The precise value of β_{read} required to ensure bounds on the reading error is determined by the implementation, but in all cases $\beta' \leq \beta_{read} < R/2$. Another useful observation is that an estimate of a remote clock's value 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.

New Condition 7 (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 Clocktime, i.e. the estimate allows us to approximately determine another clocks 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 that a remote clock shows a particular Clocktime.³ 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.

The remaining constraints are unaltered in this presentation. They are exactly as Shankar stated them. The first of these is that there is a bound to the number of faults which can be tolerated.

Old Condition 8 (bounded faults) At any time t , the number of faulty processes is at most F .

³For 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.

Synchronization algorithms execute a convergence function $cf_n(p, \theta)$ which must satisfy the conditions of *translation invariance*, *precision enhancement*, and *accuracy preservation* irrespective of the physical constraints on the system. Shankar mechanically proves that Lamport and Melliar-Smith's Interactive Convergence function [9] satisfies these three conditions. A mechanically checked proof that the fault-tolerant midpoint function used by Welch and Lynch [11] satisfies these conditions is presented in Chapter 4, and was previously reported in [15]. Schneider presents proofs that a number of other protocols satisfy these properties in [6].

Translation invariance states 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.

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

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

Precision enhancement is a formalization of the concept that, after executing the convergence function, the values of interest should be close together.

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

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

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

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

Accuracy preservation formalizes the notion that there should be a bound on the amount

of correction applied in any synchronization interval.

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

In the course of his proof of Theorem 2.1, Shankar derives the following additional conditions for an algorithm to be verified in this theory.

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 have been modified to account for differences introduced by restricting Clocktime to the integers. The bounds need to be altered to correspond to the revised version of bounded drift. Shankar's version of bounded drift was converted to correspond to Corollary 2.2.⁴ The mechanical proof has been re-run, yielding the following constraints. 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 confirming that the following are sufficient to prove Theorem 2.1.

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

⁴This is stated as axioms rate_1 and rate_2 in module clockassumptions.

$$2. \delta_S + \lceil 2\rho r_{max} \rceil + 1 \leq \delta$$

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

Since ρ is typically very small ($< 10^{-5}$), it appears that the above reworked constraints are overly conservative. It should be possible to prove Theorem 2.1 assuming the following:

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

$$2. \lceil (1 + \rho)\beta' + 2\rho r_{max} \rceil \leq \delta$$

$$3. \alpha(\lfloor \beta' + 2\Lambda' \rfloor) + \Lambda + \lceil 2\rho\beta \rceil + 1 \leq \delta.$$

An informal proof sketch can be found in Appendix A. Chapter 3 uses the new conditions presented here, as well as the existing constraints on the convergence function to provide a general proof of bounded delay (condition 4).

Chapter 3

A General Solution for Bounded Delay

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 an implementation correct must prove that this property is satisfied in the context of their implementation. In the case of the algorithm presented in [11], this is a non-trivial 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 $IC_p^i(t) = T_p^{i+1}$, i.e. 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. $T_p^{i+1} = (i + 1)R + T^0$, or

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

A more stable frame of reference is needed for bounding the separation of events. Welch and Lynch 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 S^i which satisfy condition 3 we have sufficient information to prove bounded delay for an arbitrary algorithm.

3.1 Bounded Delay Offset

The general proof follows closely an argument given in [11]. The proof adapted is that of Theorem 4 of [11, 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 first prove the following:

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

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

$$(b) |s_p^i - s_q^i| \leq \beta'.$$

Proof: By induction on i . The base case ($i = 0$) is trivial; part (a) is vacuously true and (b) is a direct consequence of new conditions 1 and 2.

Assuming that (a) and (b) are true for i we proceed by showing they hold for $i + 1$

(a)

We begin by recognizing that (a) is an instance of *accuracy preservation*. $ADJ_p^{(i+1)-1} = adj_p^{i+1} - adj_p^i = cfn(p, \Theta_p^{i+1}) - IC_p^i(t_p^{i+1})$. Since $IC_p^i(t_p^{i+1}) = \Theta_p^{i+1}(p)$ (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 ℓ, 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 ,

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

Using reading error and the induction hypothesis we get for nonfaulty clocks p and q ¹

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

We proceed as follows:

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

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

¹Recall that in this formulation, values of type time and Clocktime are both promoted to type number.

(b)

All supporting lemmas introduced in this section implicitly assume both the induction hypothesis and part (a) for $i + 1$. In Welch and Lynch's presentation 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 ,*

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

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

$$\begin{aligned} & |(s_p^i - s_q^i) - (ADJ_p^i - ADJ_q^i)| \\ &= |(ADJ_p^i - s_p^i) - (ADJ_q^i - s_q^i)| \end{aligned}$$

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 $(ADJ_p^i - s_p^i) \geq (ADJ_q^i - s_q^i)$.

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

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 hypothesis 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 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 part (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 ,*

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

Proof: The proof is taken verbatim (modulo notational differences) from [11, Lemma 10].

Note that $ic_p^{i+1}(T) = ic_p^i(T - ADJ_p^i)$ 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 2.4 of bounded drift (Condition 2), we get

$$\begin{aligned} & |ic_p^i(T - ADJ_p^i) - s_p^i - (T - ADJ_p^i - S^i)| \\ & \leq \rho|T - S^i - ADJ_p^i| \\ & \leq \rho(|T - S^i| + \alpha(\beta' + 2\Lambda')), \text{ from part (a) for } i + 1. \end{aligned}$$

The second term is similarly bounded. Lemma 3.1.1 bounds the third term. Adding the bounds and simplifying gives the result. \blacksquare

This leads to the desired result:

Lemma 3.1.3 *For good clocks p and q ,*

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

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

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

give us an upper bound for R .

3.2 Bounded Delay for Two Algorithm Schemata

We begin by noticing that both instantaneous adjustment schemes presented at the beginning of this chapter allow for a simple derivation of a β that satisfies the condition of bounded delay (old 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 β 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 2.3 (page 12) and expanding the definition of S^i , i.e. $S^i = iR + S^0$.

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

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.

New condition 1 establishes $|t_p^0 - t_q^0| \leq \beta$ for both of the above schemata. ■

All down stream proofs performed by Shankar need not be altered. 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 the appendix.² 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`³ The notion of nonfaulty clocks is expressed by the following from module `delay`.

²A slightly modified version of Shankar's module `clockassumptions` is also included in the appendix for completeness.

³This axiom has not yet been removed from the general theory. None of the proofs of *bounded delay offset* depend on it, however.

correct_during: function[process, time, time \rightarrow 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 \rightarrow function[process \rightarrow bool]]
rpred: function[event \rightarrow function[process \rightarrow bool]]
wvr_pred: function[event \rightarrow function[process \rightarrow bool]] =
 $(\lambda i : (\lambda p : \text{wpred}(i)(p) \vee \text{rpred}(i)(p)))$
wpred_ax: **Axiom** count(wpred(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** $\neg(\text{wpred}(i)(p) \wedge \text{rpred}(i)(p))$
wpred_bridge: **Axiom**
 $\text{wvr_pred}(i)(p) \wedge \text{correct_during}(p, t_p^{i+1}, t_p^{i+2}) \supset \text{wpred}(i + 1)(p)$

Also, module `delay3` states the following axiom:

recovery_lemma: **Axiom**
 $\text{delay_pred}(i) \wedge \text{ADJ_pred}(i + 1)$
 $\wedge \text{rpred}(i)(p) \wedge \text{correct_during}(p, t_p^{i+1}, t_p^{i+2}) \wedge \text{wpred}(i + 1)(q)$
 $\supset |s_p^{i+1} - s_q^{i+1}| \leq \beta'$

There are two predicates defined, `wpred` and `rpred`. `Wpred` is used to denote a working clock, i.e. 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 which states whether or not 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 $\text{rpred}(i) = (\lambda p : \text{false})$. 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 init

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 of these can be found in module `delay`. In addition there were a few assumptions included defining 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 part (a) is captured by `good_Readclock`. This, with accuracy preservation, was sufficient to establish `bnd_delay_offset_ind_a`. Part (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 part (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 β 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 the appendix. 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 old condition 3 as `rts0_new` and `rts1_new` using the substitutions suggested on page 13 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 6 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 which are much easier to satisfy for a particular implementation. To establish that an implementation is an instance of this extended theory requires the following:

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 (new condition 7, clauses 1 and 3).
3. Solve the derived inequalities listed at the end of Chapter 2 using values determined from the implementation and properties of the convergence function.
4. Satisfy the conditions of bounded interval and nonoverlap, using the derived values.
5. Identify data structures in the implementation which 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 ← 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 β' 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 above, 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 will be established in Chapter 4; requirements 2, 3, 4, 5, and 6 will be demonstrated for an abstract design in Chapter 5; and requirement 7 will be established in Chapter 6. The inequalities used in satisfying 3 will be 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 here 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 [11]. The function consists of discarding the F largest and F smallest clock readings, 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 θ . This formulation of the convergence function is different from that used in [11]. A proof of equality between the two formulations is not needed since it is shown that this formulation satisfies the properties required by Schneider's paradigm. For this function to make sense, it is clear that 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 $3F + 1$ processes [16].

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

The properties presented in this chapter are applicable for any clock synchronization protocol which employs the fault-tolerant midpoint convergence function. All that will be required for a verified implementation is a proof that the function is correctly implemented and proofs that the other conditions have been satisfied.

4.1 Translation Invariance

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.

Old Condition 9 (translation invariance) For any function θ 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

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

upon p and q being in C , so that precondition was removed for the following weakened restatement of precision enhancement.

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

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

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 pigeon hole principle.

Lemma 4.1.1 (Pigeon Hole 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 θ , 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 pigeon hole 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 θ and γ , there exists a $p \in C$ such that,*

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

Proof: Recalling that $N \geq 3F + 1$, we can apply the pigeon hole principle twice. First to establish that $|\{p : \theta(p) \geq \theta_{(N-F)}\} \cap C| \geq F + 1$, and then to establish the conclusion. ■

A 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 θ and γ 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,

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

■

This enables us to establish the following lemma.

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

$$\begin{aligned} \theta(p) &\geq \theta_{(F+1)}, \\ \gamma(q) &\geq \gamma_{(F+1)}, \text{ and} \\ |\theta(p) - \gamma(q)| &\leq X. \end{aligned}$$

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. There are three cases to consider:

- If $\gamma(p_1) > \gamma(q_1)$, let $p = q = p_1$.
- If $\theta(q_1) > \theta(p_1)$, let $p = q = q_1$.
- Otherwise, we have satisfied the hypotheses for Lemma 4.1.4, so 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 $cfn_{MID}(p, \gamma) \geq cfn_{MID}(q, \theta)$.

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

$$\leq \left\lceil \left| \frac{\gamma_{(F+1)} + \gamma_{(N-F)} - (\theta_{(F+1)} + \theta_{(N-F)})}{2} \right| \right\rceil$$

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

$$\begin{aligned} & |\gamma_{(F+1)} + \gamma_{(N-F)} - (\theta_{(F+1)} + \theta_{(N-F)})| \\ & \leq |\gamma(q) + \gamma(p_1) - \theta(p_1) - \theta(q_1)| \\ & = |\gamma(q) + (\theta(p) - \theta(p)) + \gamma(p_1) - \theta(p_1) - \theta(q_1)| \\ & \leq |\theta(p) - \theta(q_1)| + |\gamma(q) - \theta(p)| + |\gamma(p_1) - \theta(p_1)| \\ & \leq Y + 2X \text{ (by hypotheses and Lemma 4.1.5)} \end{aligned}$$

■

4.3 Accuracy Preservation

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 a weakened form of accuracy preservation. The bound holds even if p is not in C .

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

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

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 $cfn_{MID}(p, \theta) \geq \theta(q_1)$. There are two cases to consider:

- If $\theta(q) \leq cfn_{MID}(p, \theta)$, then $|cfn_{MID}(p, \theta) - \theta(q)| \leq |\theta(p_1) - \theta(q)| \leq X$.
- 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 above. The EHDM modules used in this effort are listed in the appendix.

One underlying assumption for these proofs is that $N \geq 3F + 1$. This is a well known requirement for systems to achieve Byzantine fault-tolerance without requiring authentication [16]. 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 of this chapter. 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. 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. There is nothing to prevent the two clusters from drifting further apart. The one additional clock ensures that for any pair of good clocks, the ranges of the readings used in the convergence function 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

$$i \leq j \wedge j \leq N \supset \vartheta(\text{funsort}(\vartheta)(i)) \geq \vartheta(\text{funsort}(\vartheta)(j))$$

funsort_trans_inv: Axiom

$$k \leq N \supset (\vartheta(\text{funsort}((\lambda q : \vartheta(q) + X))(k))) = \vartheta(\text{funsort}(\vartheta)(k))$$

cnt_sort_geq: Axiom

$$k \leq N \supset \text{count}((\lambda p : \vartheta(p) \geq \vartheta(\text{funsort}(\vartheta)(k))), N) \geq k$$

cnt_sort_leq: Axiom

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

The appendix contains the proof chain analysis for the three properties stated above. 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 pigeon hole principle. In addition, Lemma `count_complement` was moved from Shankar's module `ica3` to `countmod`. Shankar's complete proof was re-run 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 in that they do not require that the process evaluating the convergence function be part of the collection of *working* clocks. The proofs that $cf_{MID}(p, \theta)$ 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 formulae.

Chapter 5 presents a hardware design of a clock synchronization system that uses the fault-tolerant midpoint convergence function. It will be shown that the design satisfies the remaining constraints of the theory.

Chapter 5

Design of a Clock

Synchronization System

This chapter describes a design of a fault-tolerant clock synchronization circuit which 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 then Section 5.2 demonstrates that the design meets requirements 2 through 6 from Section 3.4 (page 30).

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 $R + ADJ_p^i$. It is assumed that all good clocks know the index of the current interval (a simple counter is sufficient, provided that all *good* channels start the counter in the same interval). Furthermore, it is assumed 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;¹ i.e. when $LC_p^i(t) = Q$, where LC is a counter measuring elapsed time since the beginning of the current interval. Our estimate, Θ_p^{i+1} , of other clocks is

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

where t_{pq} is the time that 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, ρ , of a good clock is less than 10^{-5} ; this corresponds to the drift rate of commercially available oscillators. By selecting R to be $\leq 10^4$ ticks², 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 will always receive a signal from itself when $LC_p^i(t) = Q$. Therefore there is no error 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

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

¹This 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 $LC_p^i(t) = Q$.

²This corresponds to a synchronization interval of 1 msec for a 10MHz clock.

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

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

In this design $T^0 = 0$, so we just need to ensure that $cfn_{MID}(p, \Theta_p^{i+1}) = (i+1)R$. Using translation invariance and the definition for Θ_p^{i+1} given above, we get,

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

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

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

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

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

Assuming that 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.³ All that remains is to determine the appropriate readings to use. We know that we will observe at least $N - F$ pulses during the synchronization interval.⁴ Since Q is fixed and LC is non-decreasing during the interval, the readings $(\lambda q \cdot Q - LC_p^i(t_{pq}))$ are sorted into decreasing order by arrival time. Suppose t_{pq} is when the $(F + 1)$ th pulse is recognized; $(Q - LC_p^i(t_{pq}))$ 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 necessary information

³An arithmetic shift right of a two's complement value preserves the sign bit, while truncating the least significant bit.

⁴Remember that this chapter assumes that there are a sufficient number $(N - F)$ of synchronized nonfaulty clocks participating in the protocol.

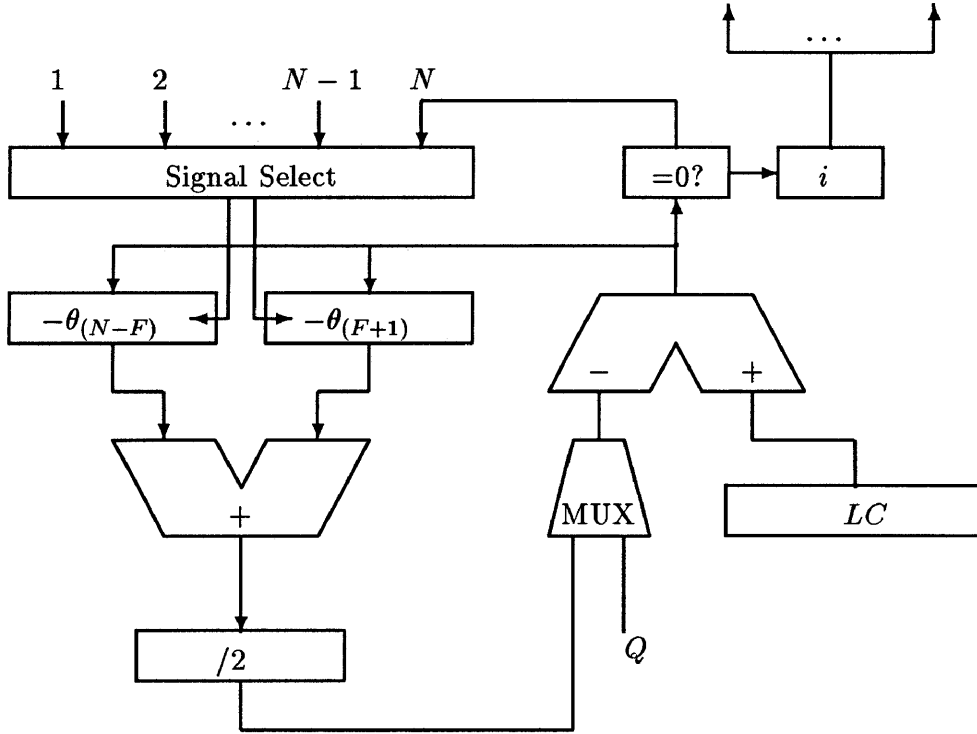


Figure 5.1: Informal Block Model

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:

- N pulse recognizers (only one pulse per clock is recognized in any given interval),
- a pulse counter (triggers events based upon pulse arrivals),
- a local counter LC (measures elapsed time since beginning of current interval),
- an 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 $-\theta_{(N-F)}$,
- an adder for computing the sum of these two registers, and
- a divide-by-2 component (arithmetic shift right).

The pulses are already sorted by arrival time, so it is natural to use a pulse counter to select the time-stamp of the $(F+1)$ th and the $(N-F)$ th pulses for the computation of the convergence function. As stated previously, all that is required is the difference between the local and remote clocks. Let $\theta = (\lambda q \cdot \Theta_p^{i+1}(q) - T_p^{i+1})$. When the $F+1$ st ($N-F$ 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 $LC_p^i(t) - (-cf_{nMID}(p, (\theta))) = R$ it is time to begin the $i+1$ st 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 on page 30.

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 above, we know that

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

$LC_p^0(t)$ corresponds 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. The definition for $VC_p(t)$ follows directly from the definition. 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 $iR + LC_p^i(t)$.

The above 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 the above, when $LC_p^i(t) = Q$, it transmits its synchronization signal to the other clocks in the system. This is clearly an instance of the general algorithm schema presented as requirement 6. 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 7 (page 16) is for Λ to equal one tick. The third clause, however, includes real time separation, and a possible value for Λ' of approximately 0.5 ticks. We will assume these values for the remainder of this thesis. A hardware realization of the above abstract design, with estimates of reading error equivalent to these is presented in [17]. These bounds have not been established formally. Preliminary research which may enable formal derivation of such bounds can be found in [18].

Using the above values for reading error, we can now solve the inequalities presented at the end of Chapter 2 (this is requirement 3). 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. $\alpha(\lfloor \beta' + 2\Lambda' \rfloor) + \Lambda + \lceil 2\rho\beta \rceil + 1 \leq \delta$.

We begin with the first; we would like to find the smallest β' which satisfies the inequality. The bound β' 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:

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

The estimate for Λ' is $\approx 0.5 < 1 - b/2$, so $\lfloor 2\Lambda' + 2 \rfloor = 3$ and $\lfloor \beta' + 2\Lambda' \rfloor = B + 1$. Using the π established for $cfm_{MID}(p, \theta)$ in Chapter 4, we get

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

The smallest B that satisfies this inequality is 7, therefore the above circuit can maintain a β' that is ≈ 7.4 ticks. By using this value in the second inequality, we see that $\delta \geq 8$. Remembering that α is the identity function for $cfm_{MID}(p, \theta)$ and that $\Lambda = 1$, 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. Using these numbers with a 10MHz clock rate, this circuit will synchronize the redundant clocks to within about one μsec . 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 the above design satisfies requirement 4. This consists of satisfying new conditions 3 and 6. We know that $\alpha(\beta' + 2\Lambda') < 9$ and that $T^0 = 0$. We can satisfy new condition 3 (page 13) 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 $R \gg (1 + \rho)\beta + \alpha(\beta' + 2\Lambda')$, new condition 6 (page 15) is easily met. There is one

remaining requirement from the list presented in Section 3.4. Requirement 7, achieving initial synchronization, will be established 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 new condition 1, bounded delay init, from page 11. It is sufficient to establish this in the absence of faults. 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 less formal than those of earlier chapters.

Finally, 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 while still allowing transient recovery. Section 6.2 also demonstrates that the design presented in Chapter 5 meets the requirements of these transient recovery conditions.

6.1 Initial Synchronization

If we can get into a state which satisfies the requirements for *precision enhancement*:

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

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

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 $X = \lfloor 2\Lambda' + 2 \rfloor$ ¹, 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 will then be required to establish a consistent interval counter.² It will be necessary to ensure that the clocks reach a state satisfying the above constraints. Clearly, we would like β_{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

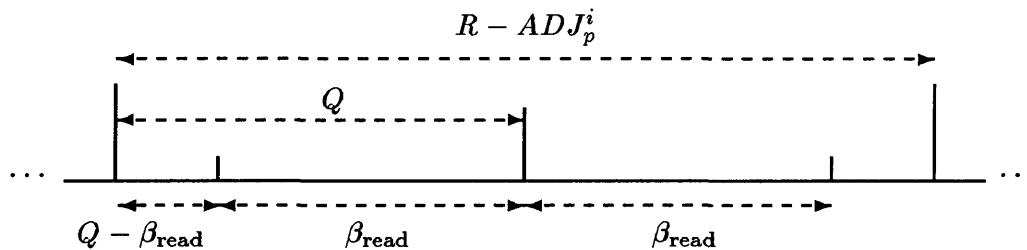


Figure 6.1: Synchronization Interval

¹This condition is satisfied when for $p, q \in C$, $|s_p^i - s_q^i| \leq \beta_{\text{read}}$. During initialization, $i = 0$.

²For 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 [19] and [20].

synchronization pulses within β_{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, there may exist more appropriate settings for Q .

6.1.1 Mechanisms for Initialization

In order to ensure that we reach a state which satisfies the above requirements, it is necessary to identify possible states which violate the above requirements. Such states would happen due to 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 that there were a sufficient number of clock readings to resolve $\theta_{(F+1)}$ and $\theta_{(N-F)}$. This may not be the case during initialization. We need to determine a course of action when we do not observe $N - F$ clocks. Two plausible options are:

Assumed Perfection — pretend all clocks are observed to be in perfect synchrony, or

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 upon 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 will cause the unobserved readings to be $(R - Q)$ which is equivalent to observing the pulse at the end of an interval of duration R .

We will discuss these two possibilities with respect to a four clock system. The arguments for the more 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 β_{read} .³

³Otherwise, the conditions enumerated above would be satisfied.

Assumed Perfection

In this case, 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 β_{read} . A real implementation needs a certain amount of time to reset for the next interval, so 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 will be able to read the other. If additional clocks are added that are synchronous with the hidden clock, they too will be hidden. This is illustrated in Figure 6.2. Clearly, this is insufficient for initial synchronization. It is also clearly unable

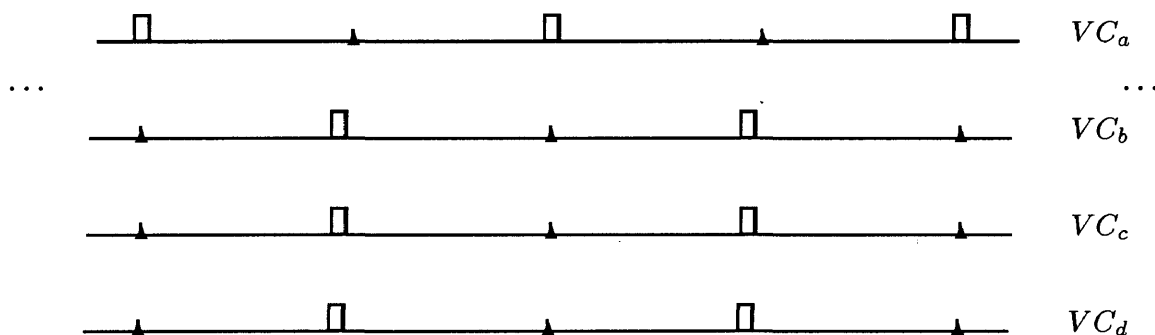


Figure 6.2: Pathological Scenario

to guarantee recovery from a transient fault. Although the illustration shows $Q = R/2$, a similar pathological scenario exists for any setting of Q .

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 a case where only two clocks are actively participating. Assume for the sake of this discussion that $Q = R/2$ (to maximize β_{read}). There are two possibilities—their pulses are either separated by more than $R/2$ or less than $R/2$. If

the former is true, then each clock computes the maximum adjustment of $R/2$, and will transmit a pulse every $3R/2$ ticks. If the latter, one clock will compute an adjustment of $R/4$, and will transmit a pulse every $5R/4$ ticks; while the other will compute an adjustment between $R/4$ and $R/2$, and will converge to a point where it transmits a pulse every $5R/4$ ticks and is synchronized with the first clock. The two cases are illustrated in Figure 6.3. If we add a third clock to the first scenario, it must be within $R/2$ of at

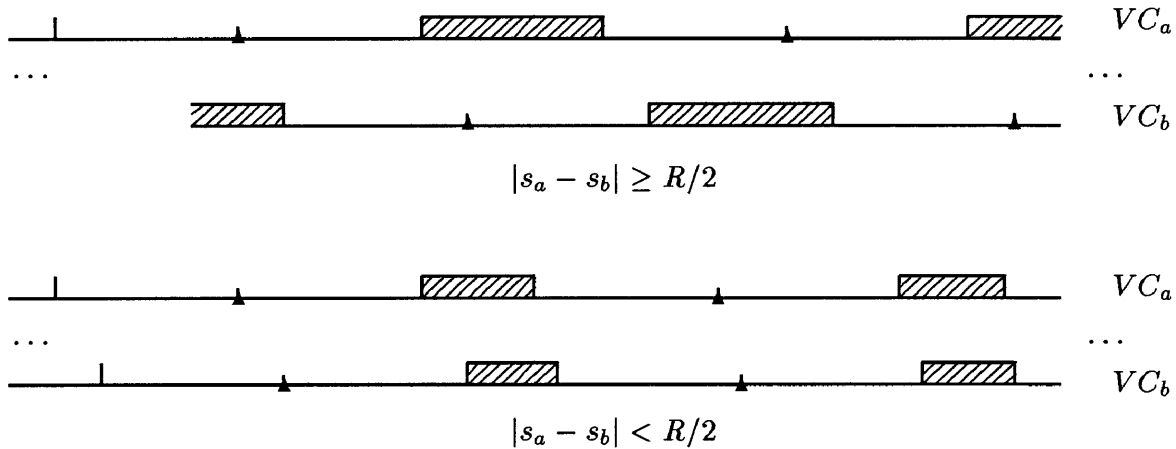


Figure 6.3: End of Interval Initialization

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 the second scenario. 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 β_{read} of each other. This argument also covers the case where we add a third clock to the second scenario. 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 is sufficient to satisfy the initialization requirement, but does not address re-initialization 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. This figure

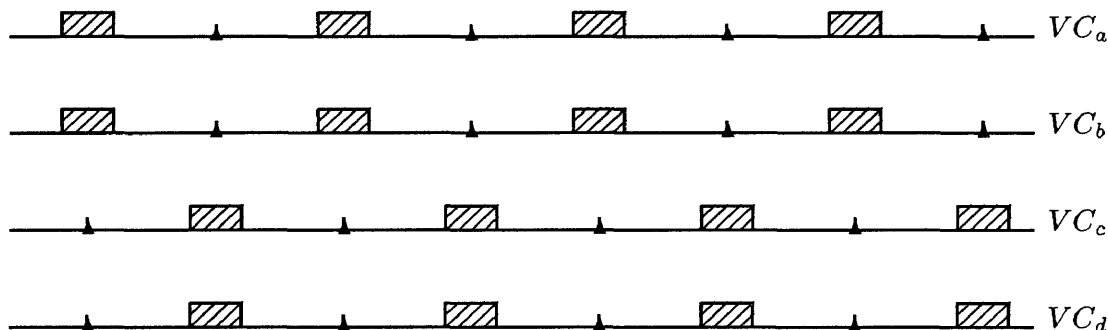


Figure 6.4: Pathological End of Interval Initialization

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.

End of Interval—Time Out

Inspection of Figure 6.4 suggests that if any of the clocks were to arbitrarily decide to not compute any adjustment, the immediately following interval would have a collection of three clocks within β_{read} of each other. This is shown in Figure 6.5. When clock b decides not to compute any adjustment, it shifts to a point where its pulse is within β_{read} of c and d . Here the algorithm takes over, and the three values converge.⁴ Clock a is also brought

⁴Figure 6.5 illustrates the fault-free case. If a were faulty, it could delay convergence by at most $\log_2(\beta_{\text{read}})$.

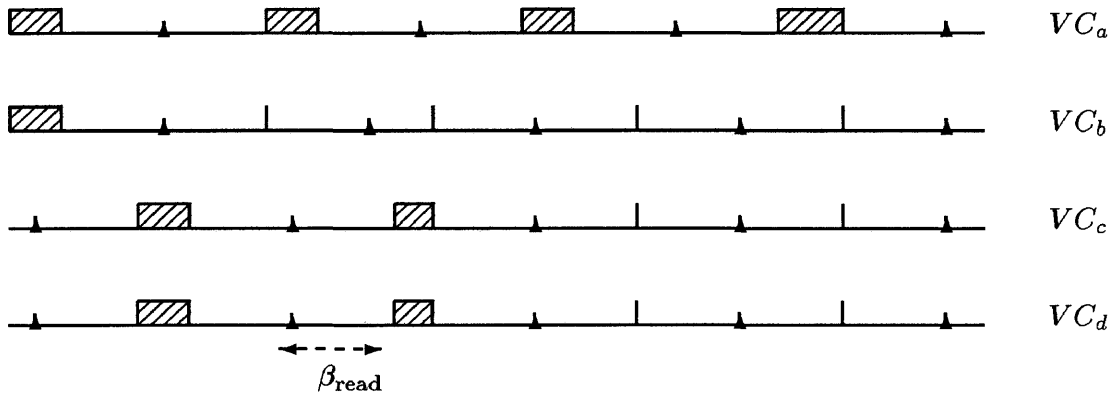


Figure 6.5: End of Interval Initialization—Time Out

into the fold because of the transient recovery process. This process will be 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 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 β_{read} , else the properties of precision enhancement force the system to synchronize. In a collection of three clocks, at least one pair must be within β_{read} ; Figure 6.3 shows that in the absence of other readings, a pair within β_{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; whether a faulty clock can prevent the time out from expiring, and if a faulty clock can 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 altering their computed corrections. Clock c is not

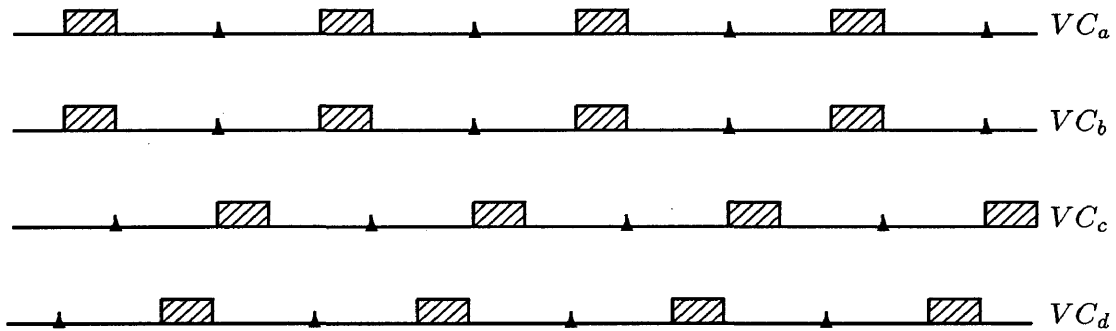


Figure 6.6: End of Interval Initialization: d Faulty—benign

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 either a or b at the end of its interval, with neither a nor b observing c . This is only possible if c and a (b) are separated by precisely $R/2$ ticks. Even then, it is more likely that a (b) will 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 (b) will see c first. If a (b) observes c , the effect will be the same as if it had timed out. Since a (b) is

synchronized with b (a), 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 on 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 β_{read} . A possible scenario is shown in Figure 6.7. Here, d prevents a from synchronizing and also causes a 's time out

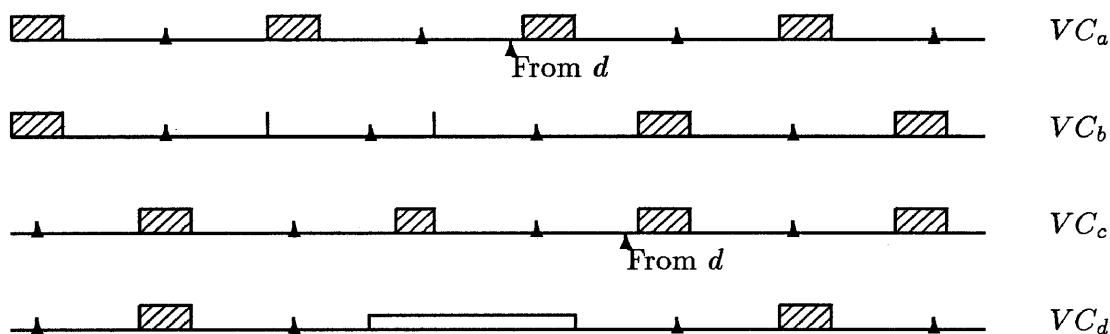


Figure 6.7: End of Interval Initialization: d Faulty—malicious

to reset. At some point, d will also need to send a pulse at the end of an interval to either b or c , ensuring that just one of them will time out. The process can then be repeated, preventing the collection of good clocks from ever becoming synchronized.

The attempt for a robust 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 to Other Approaches

The argument that the clocks converge within $\log_2(\beta_{\text{read}})$ intervals is adapted from that given by Welch and Lynch [11]. However, the approach given here for achieving initial

synchronization differs from most methods in that it first synchronizes the interval clocks, and then it decides upon an index for the current interval. Techniques in [11], [12], and [13] all depend upon 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 the agreement first approach is necessary to ensure initialization in the presence of malicious faults. The approach taken here seems similar to that mentioned in [14], however, details of their approach are not given.

6.2 Transient Recovery

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

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

Using 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 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 $\wedge \text{rpred}(i)(p) \supset |\text{ADJ}_p^i| \leq \alpha([\beta' + 2 * \Lambda'])$
4. From module `delay`—
wpred_preceding: Axiom $\text{wpred}(i + 1)(p) \supset \text{wpred}(i)(p) \vee \text{rpred}(i)(p)$
wpred_rpred_disjoint: Axiom $\neg(\text{wpred}(i)(p) \wedge \text{rpred}(i)(p))$
wpred_bridge: Axiom

$$\text{wvr_pred}(i)(p) \wedge \text{correct_during}(p, t_p^{i+1}, t_p^{i+2}) \supset \text{wpred}(i + 1)(p)$$

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 above. In Chapter 3, two options were presented for determining when to apply the adjustment. These options are:

1. $T_p^{i+1} = (i + 1)R + T^0$, or
2. $T_p^{i+1} = (i + 1)R + T^0 - \text{ADJ}_p^i$.

Since the design presented in Chapter 5 uses the second option, the arguments for transient recovery will be 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:

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

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

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

The second of the above is made possible by using the recovery version of precision enhancement. Since $\beta' \geq 4\rho r_{\text{max}} + \pi(2\Lambda' + 2, \beta' + 2\Lambda')$, all that remains is to establish that $2\rho(|S^{i+1} - S^i| + \alpha(\beta_{\text{read}} + 2\Lambda')) \leq 4\rho r_{\text{max}}$. Since $\beta_{\text{read}} < R/2$ and α is the identity function, this is easily established. Axiom `delay_recovery` is easily established for implementations using the second algorithm schema presented in Chapter 3. Since $T_p^{i+1} + ADJ_p^i = (i+1)R + T^0$ 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 above. 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 to the design required 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 β_{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 β_{read} of the collection of good clocks.

The argument for satisfying `rpred` will be given for a four clock system; the argument for the general case requires an additional timeout mechanism to avoid pathological cases. Consider the first full synchronization interval that 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 δ 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 that proposed by Welch and Lynch [11]. They propose that when a process awakens, that it observe incoming messages until it can determine which round is underway, and then wait sufficiently long to ensure that it has seen all valid messages in that round. It can then compute the necessary correction to become synchronized. Srikanth and Toueg [12] use a similar approach, modified to the context of their algorithm. Halpern et al. [13] suggest a rather complicated protocol which requires explicit cooperation of other clocks in the system. It is more appropriate when the number of clocks in the system varies greatly over time. All of these approaches have the common theme, namely, that the joining processor knows that it wants to join. This implies the presence of some diagnostic logic or timeout mechanism which 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 to restore the correct interval counter.

Chapter 7

Concluding Remarks

Clock synchronization provides the cornerstone of any fault-tolerant computer architecture. To avoid a single point failure it is imperative that each processor maintain a local clock which is periodically resynchronized with other clocks in a fault-tolerant manner. Due to subtleties involved in reasoning about interactions involving misbehaving components, it is necessary to prove that the clock synchronization function operates correctly. Shankar [7] provides a mechanical proof (using EHDm [8]) that Schneider's generalized protocol [6] achieves Byzantine fault-tolerant clock synchronization, provided that eleven constraints are satisfied. This thesis 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.¹ This thesis presents a general proof of this property from slightly revised versions of the remaining conditions. The revised conditions have also been shown to imply the original 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 synchronization system based on the fault-tolerant midpoint convergence function was shown to satisfy

¹This terminology is from Shankar's report, Schneider called this property a reliable time source.

the constraints of the revised theory.

The fault-tolerant midpoint convergence function has been proven (in EHDM) to satisfy the properties of translation invariance, precision enhancement, and accuracy preservation. These proofs are reusable in the verification of any synchronization algorithm which uses the same function. The assumed bound on the number of faults was established in the proof of precision enhancement. This proof assumes that the number of faults allowed is fewer than one-third of the number of clocks.

An informal design of a circuit to implement the clock synchronization function has been presented. This design was derived from the algebraic constraints presented in Section 2.1. Assuming the properties of bounded drift (new condition 2) and reading error (new condition 7), it was shown that this design satisfied the remaining constraints of the theory. Bounded drift is a physical property that cannot be established formally; in essence, it defines the behavior of a nonfaulty clock. Establishing reading error requires an analysis of the low-level asynchronous communication mechanism employed by the system; such an analysis is beyond the scope of this thesis.

It was hoped that the circuit could be shown to automatically initialize itself, even in the presence of faults. Two approaches for a four clock system were explored and shown to possess pathological scenarios which 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 if the failures are benign. However, malicious behavior from a failed clock can prevent good clocks from synchronizing. It appears that the standard approach of first reaching agreement, and then synchronizing, will be necessary to initialize in the presence of arbitrary failures.

In keeping with the spirit of the Reliable Computing Platform, it is desirable that the clock synchronization subsystem provide for recovery 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 [1]. It was

shown that a four clock instance of the given design will eventually satisfy the transient recovery assumptions. Extensions to accommodate the more general case require a time-out mechanism, but otherwise the argument is similar.

In summary, a mechanically checked version of Schneider's paradigm for fault-tolerant clock synchronization was extended. Use of the extended theory was illustrated in the verification of an abstract design of a fault-tolerant clock synchronization system. Some of the requirements were established via a mechanically checked formal proof using EHDM, while other constraints were demonstrated informally. Ultimately, a mechanically checked argument should be developed for all of the constraints. This will help to clarify the underlying assumptions, and in many cases can 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 will need to be revised considerably before a mechanically checked proof will be 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 would have been possible otherwise.

Appendix A

Proof of Agreement

There are two parts to this appendix. First, there is an informal proof sketch that agreement can be established using the revised constraints on δ and some of the intermediate results of Chapter 3. 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 on page 25) to

establish the second of the above constraints. The third of these is obtained by substituting the revised bounds on the array of clock readings (established in the proof of part (a) of Theorem 3.1 on page 21) into Shankar's proof¹.

We wish to 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 $\max(t_p^i, t_q^i) \leq t < \min(t_p^{i+1}, t_q^{i+1})$,*

$$|IC_p^i(t) - IC_q^i(t)| \leq \lceil (1 + \rho)\beta^i + 2\rho r_{max} \rceil$$

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 2.2 on page 12.

All that remains is to establish that $|ic_q^i(T) - ic_p^i(T)| \leq \beta^i + 2\rho r_{max}/(1 + \rho)$. On page 13, we defined r_{max} to be $(1 + \rho)(R + \alpha(\beta^i + 2\Lambda^i))$. The proof is by induction on i . For $i = 0$,

$$|ic_q^0(T) - ic_p^0(T)| \leq |t_q^0 - t_p^0| + 2\rho(\max(T_p^{1}, T_q^{1}) - T^0)$$

¹This has not been done in the mechanical proof because Shankar's proof has not yet been revised to accommodate transient recovery

$$\leq \beta' + 2\rho(R + \alpha(\beta' + 2\Lambda'))$$

For the inductive step we use Lemma 3.1.2 to establish that

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

There are two cases to consider: if $T \leq S^{i+1}$, this is clearly less than β' ; 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 $\max(T_p^{i+1}, T_q^{i+1}) - S^{i+1} \leq (R + \alpha(\beta' + 2\Lambda'))$. ■

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

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

Proof Sketch: The proof follows closely the argument given in the proof of case 2 of Theorem 2.3.2 in [7]. The proof is in two parts. First, the difference at t_q^{i+1} is bounded using accuracy preservation, and then the remainder of the interval is bounded. The difference in this presentation is that here the argument to α 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.Readererror
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_l0
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 $\text{correct}(p, t) \wedge t < t_p^i + r_{min} \supset t < t_p^{i+1}$

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

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

agreement: Lemma $\beta \leq r_{min}$

$\wedge \mu \leq \delta_S \wedge \pi([2 * \Lambda + 2 * \beta * \rho] + 1,$

$\delta_S + [2 * ((r_{max} + \beta) * \rho + \Lambda)] + 1)$

$\leq \delta_S$

$\wedge \delta_S + [2 * r_{max} * \rho] + 1 \leq \delta$

$\wedge \alpha(\delta_S + [2 * (r_{max} + \beta) * \rho + 2 * \Lambda] + 1) + \Lambda + [2 * \beta * \rho] + 1$

$\leq \delta$

$\wedge t \geq 0 \wedge \text{correct}(p, t) \wedge \text{correct}(q, t)$

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

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 ← t, y ← r_min},
synctime_multiples {i ← ⌈t/r_min⌉ + 1},
rmin_0,
div_nonnegative {x ← t, y ← r_min},
ceil_defn {x ← (t/r_min)}

```

correct_synctime_proof: Prove correct_synctime from rts1 {t ← t@CS}

```

synctime_multiples_pred: function[nat, nat, posnumber → bool] ==
  (λ i, p, t : correct(p, t) ∧ t ≥ 0 ∧ t < i * r_min ⊃ t_p^i > t)

```

synctime_multiples_step: Lemma

```

correct(p, t) ∧ t ≥ t_p^i ∧ t ≥ 0 ⊃ t_p^i ≥ i * r_min

```

**synctime_multiples_proof: Prove synctime_multiples from
synctime_multiples_step**

```

synctime_multiples_step_pred: function[nat, nat, posnumber → bool] ==
  (λ i, p, t : correct(p, t) ∧ t_p^i ≤ t ∧ t ≥ 0 ⊃ t_p^i ≥ i * r_min)

```

synctime_multiples_step_proof: Prove synctime_multiples_step from

```

induction {prop ← (λ i : synctime_multiples_step_pred(i, p, t))},
mult_l0 {x ← r_min},
synctime_0,
rts_1 {i ← j@P1},
rmin_0,
correct_closed {s ← t, t ← t_p^{j@P1+1}},
distrib {x ← j@P1, y ← 1, z ← r_min},
mult_lident {x ← 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

γ, θ : Var function[process \rightarrow Clocktime]

$\delta, \rho, r_{min}, r_{max}, \beta$: number

Λ, μ : Clocktime

$PC_{*1}(*2), VC_{*1}(*2)$: function[process, time \rightarrow Clocktime]

t_{*1}^{*2} : function[process, event \rightarrow time]

Θ_{*1}^{*2} : function[process, event \rightarrow function[process \rightarrow Clocktime]]

$IC_{*1}^{*2}(*3)$: function[process, event, time \rightarrow Clocktime]

correct: function[process, time \rightarrow bool]

cfn : function[process, function[process \rightarrow Clocktime] \rightarrow Clocktime]

π : function[Clocktime, Clocktime \rightarrow Clocktime]

α : function[Clocktime \rightarrow Clocktime]

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 $\text{correct}(p, 0) \supset PC_p(0) \geq 0 \wedge PC_p(0) \leq \mu$
correct_closed: Axiom $s \geq t \wedge \text{correct}(p, s) \supset \text{correct}(p, t)$
rate_1: Axiom $\text{correct}(p, s) \wedge s \geq t \supset PC_p(s) - PC_p(t) \leq [(s - t) \star (1 + \rho)]$
rate_2: Axiom $\text{correct}(p, s) \wedge s \geq t \supset PC_p(s) - PC_p(t) \geq [(s - t) \star (1 - \rho)]$
rts0: Axiom $\text{correct}(p, t) \wedge t \leq t_p^{i+1} \supset t - t_p^i \leq r_{max}$
rts1: Axiom $\text{correct}(p, t) \wedge t \geq t_p^{i+1} \supset t - t_p^i \geq r_{min}$
rts_0: Lemma $\text{correct}(p, t_p^{i+1}) \supset t_p^{i+1} - t_p^i \leq r_{max}$
rts_1: Lemma $\text{correct}(p, t_p^{i+1}) \supset t_p^{i+1} - t_p^i \geq r_{min}$
rts2: Axiom $\text{correct}(p, t) \wedge t \geq t_q^i + \beta \wedge \text{correct}(q, t) \supset t \geq t_p^i$
rts_2: Axiom $\text{correct}(p, t_p^i) \wedge \text{correct}(q, t_q^i) \supset t_p^i - t_q^i \leq \beta$
synctime_0: Axiom $t_p^0 = 0$
VClock_defn: Axiom
 $\text{correct}(p, t) \wedge t \geq t_p^i \wedge t < t_p^{i+1} \supset VC_p(t) = IC_p^i(t)$
adj_{x1}^{*2}: function[process, event \rightarrow Clocktime] =
 $(\lambda p, i : (\text{if } i > 0 \text{ then } cf_n(p, \Theta_p^i) - PC_p(t_p^i) \text{ else } 0 \text{ end if}))$
IClock_defn: Axiom $\text{correct}(p, t) \supset IC_p^i(t) = PC_p(t) + adj_p^i$
Readerror: Axiom $\text{correct}(p, t_p^{i+1}) \wedge \text{correct}(q, t_q^{i+1})$
 $\supset |\Theta_p^{i+1}(q) - IC_q^i(t_p^{i+1})| \leq \Lambda$
translation_invariance: Axiom
 $cf_n(p, (\lambda p_1 \rightarrow \text{Clocktime} : \gamma(p_1) + X)) = cf_n(p, \gamma) + X$
ppred: Var function[process \rightarrow bool]
F: process
okay_Readpred: function[function[process \rightarrow Clocktime], number,
function[process \rightarrow bool] \rightarrow bool] =
 $(\lambda \gamma, y, \text{ppred} : (\forall l, m : \text{ppred}(l) \wedge \text{ppred}(m) \supset |\gamma(l) - \gamma(m)| \leq y))$
okay_pairs: function[function[process \rightarrow Clocktime],
function[process \rightarrow Clocktime], number,
function[process \rightarrow bool] \rightarrow bool] =
 $(\lambda \gamma, \theta, x, \text{ppred} : (\forall p_3 : \text{ppred}(p_3) \supset |\gamma(p_3) - \theta(p_3)| \leq x))$
okay_Readpred_floor: Lemma
 $\text{okay_Readpred}(\gamma, y, \text{ppred}) \supset \text{okay_Readpred}(\gamma, \lfloor y \rfloor, \text{ppred})$
okay_pairs_floor: Lemma
 $\text{okay_pairs}(\gamma, \theta, x, \text{ppred}) \supset \text{okay_pairs}(\gamma, \theta, \lfloor x \rfloor, \text{ppred})$

N_maxfaults: Axiom $F < N$

precision_enhancement_ax: Axiom

$$\begin{aligned} & \text{count}(\text{ppred}, N) \geq N - F \\ & \quad \wedge \text{okay_Readpred}(\gamma, Y, \text{ppred}) \\ & \quad \quad \wedge \text{okay_Readpred}(\theta, Y, \text{ppred}) \\ & \quad \quad \quad \wedge \text{okay_pairs}(\gamma, \theta, X, \text{ppred}) \wedge \text{ppred}(p) \wedge \text{ppred}(q) \\ & \quad \supset |\text{cfn}(p, \gamma) - \text{cfn}(q, \theta)| \leq \pi(X, Y) \end{aligned}$$

precision_enhancement_recovery_ax: Axiom

$$\begin{aligned} & \text{count}(\text{ppred}, N) \geq N - F \\ & \quad \wedge \text{okay_Readpred}(\gamma, Y, \text{ppred}) \\ & \quad \quad \wedge \text{okay_Readpred}(\theta, Y, \text{ppred}) \wedge \text{okay_pairs}(\gamma, \theta, X, \text{ppred}) \\ & \quad \supset |\text{cfn}(p, \gamma) - \text{cfn}(q, \theta)| \leq \pi(X, Y) \end{aligned}$$

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

okay_Reading: function[function[process \rightarrow Clocktime], number, time \rightarrow bool] =

$$(\lambda \gamma, y, t : (\forall p_1, q_1 : \text{correct}(p_1, t) \wedge \text{correct}(q_1, t) \supset |\gamma(p_1) - \gamma(q_1)| \leq y))$$

okay_Readvars: function[function[process \rightarrow Clocktime], function[process \rightarrow Clocktime], number, time \rightarrow bool] =

$$(\lambda \gamma, \theta, x, t : (\forall p_3 : \text{correct}(p_3, t) \supset |\gamma(p_3) - \theta(p_3)| \leq x))$$

okay_Readpred_Reading: Lemma

$$\text{okay_Reading}(\gamma, y, t) \supset \text{okay_Readpred}(\gamma, y, (\lambda p : \text{correct}(p, t)))$$

okay_pairs_Readvars: Lemma

$$\text{okay_Readvars}(\gamma, \theta, x, t) \supset \text{okay_pairs}(\gamma, \theta, x, (\lambda p : \text{correct}(p, t)))$$

precision_enhancement: Lemma

$$\begin{aligned} & \text{okay_Reading}(\gamma, Y, t_p^{i+1}) \\ & \quad \wedge \text{okay_Reading}(\theta, Y, t_p^{i+1}) \\ & \quad \quad \wedge \text{okay_Readvars}(\gamma, \theta, X, t_p^{i+1}) \\ & \quad \quad \quad \wedge \text{correct}(p, t_p^{i+1}) \wedge \text{correct}(q, t_p^{i+1}) \\ & \quad \supset |\text{cfn}(p, \gamma) - \text{cfn}(q, \theta)| \leq \pi(X, Y) \end{aligned}$$

okay_Reading_defn_lr: Lemma

$$\begin{aligned} & \text{okay_Reading}(\gamma, y, t) \\ & \quad \supset (\forall p_1, q_1 : \text{correct}(p_1, t) \wedge \text{correct}(q_1, t) \supset |\gamma(p_1) - \gamma(q_1)| \leq y) \end{aligned}$$

okay_Reading_defn_rl: Lemma

$$\begin{aligned} & (\forall p_1, q_1 : \text{correct}(p_1, t) \wedge \text{correct}(q_1, t) \supset |\gamma(p_1) - \gamma(q_1)| \leq y) \\ & \quad \supset \text{okay_Reading}(\gamma, y, t) \end{aligned}$$

okay_Readvars_defn_lr: Lemma

$$\text{okay_Readvars}(\gamma, \theta, x, t) \supset (\forall p_3 : \text{correct}(p_3, t) \supset |\gamma(p_3) - \theta(p_3)| \leq x)$$

okay_Readvars_defn_rl: Lemma

$$(\forall p_3 : \text{correct}(p_3, t) \supset |\gamma(p_3) - \theta(p_3)| \leq x) \supset \text{okay_Readvars}(\gamma, \theta, x, t)$$

accuracy_preservation_ax: Axiom

$$\begin{aligned} &\text{okay_Readpred}(\gamma, X, \text{ppred}) \wedge \text{count}(\text{ppred}, N) \geq N - F \wedge \text{ppred}(p) \wedge \text{ppred}(q) \\ &\supset |\text{cfn}(p, \gamma) - \gamma(q)| \leq \alpha(X) \end{aligned}$$

accuracy_preservation_recovery_ax: Axiom

$$\begin{aligned} &\text{okay_Readpred}(\gamma, X, \text{ppred}) \wedge \text{count}(\text{ppred}, N) \geq N - F \wedge \text{ppred}(q) \\ &\supset |\text{cfn}(p, \gamma) - \gamma(q)| \leq \alpha(X) \end{aligned}$$

Proof

okay_Readpred_floor_pr: Prove okay_Readpred_floor from

$$\begin{aligned} &\text{okay_Readpred } \{l \leftarrow l@p2, m \leftarrow m@p2\}, \\ &\text{okay_Readpred } \{y \leftarrow \lfloor y \rfloor\}, \\ &\text{iabs_is_abs } \{X \leftarrow \gamma(l@p2) - \gamma(m@p2), x \leftarrow \gamma(l@p2) - \gamma(m@p2)\}, \\ &\text{floor_mon } \{x \leftarrow \text{iabs}(X@p3)\}, \\ &\text{floor_int } \{i \leftarrow \text{iabs}(X@p3)\} \end{aligned}$$

okay_pairs_floor_pr: Prove okay_pairs_floor from

$$\begin{aligned} &\text{okay_pairs } \{p_3 \leftarrow p_3@p2\}, \\ &\text{okay_pairs } \{x \leftarrow \lfloor x \rfloor\}, \\ &\text{iabs_is_abs } \{x \leftarrow \gamma(p_3@p2) - \theta(p_3@p2), X \leftarrow \gamma(p_3@p2) - \theta(p_3@p2)\}, \\ &\text{floor_mon } \{x \leftarrow \text{iabs}(X@p3), y \leftarrow x\}, \\ &\text{floor_int } \{i \leftarrow \text{iabs}(X@p3)\} \end{aligned}$$

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 $\{p_1 \leftarrow p_1@P1S, q_1 \leftarrow q_1@P1S\}$ **from** **okay_Reading**

okay_Reading_defn_lr_pr: Prove okay_Reading_defn_lr from

okay_Reading $\{p_1 \leftarrow p_1@CS, q_1 \leftarrow q_1@CS\}$

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

okay_Readvars

okay_Readvars_defn_lr_pr: Prove okay_Readvars_defn_lr from

okay_Readvars $\{p_3 \leftarrow p_3@CS\}$

precision_enhancement_pr: **Prove precision_enhancement from**
 precision_enhancement_ax {ppred $\leftarrow (\lambda q : \text{correct}(q, t_p^{i+1}))$ },
 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 $\leftarrow (\lambda p : \text{correct}(p, t))$ },
 okay_Reading { $p_1 \leftarrow l@P1S, q_1 \leftarrow m@P1S$ }

okay_pairs_Readvars_pr: **Prove okay_pairs_Readvars from**
 okay_pairs {ppred $\leftarrow (\lambda p : \text{correct}(p, t))$ }, 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, while `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

γ : Var function[process \rightarrow 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 \rightarrow bool]

S^0 : Clocktime

S^{*1} : function[event \rightarrow Clocktime] = $(\lambda i : i * R + S^0)$

$pc_{*1}(*2)$: function[process, Clocktime \rightarrow time]

$ic_{*1}^{*2}(*3)$: function[process, event, Clocktime \rightarrow time] =
 $(\lambda p, i, T : pc_p(T - adj_p^i))$

s_{*1}^{*2} : function[process, event \rightarrow time] = $(\lambda p, i : ic_p^i(S^i))$

T^0 : Clocktime

T_{*1}^{*2} : function[process, event \rightarrow Clocktime]

synctime_defn: Axiom $t_p^{i+1} = ic_p^i(T_p^{i+1})$

synctime0_defn: Axiom $t_p^0 = ic_p^0(T^0)$

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 \rightarrow function[process, process \rightarrow bool]]

correct_during: function[process, time, time \rightarrow 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 \rightarrow function[process \rightarrow bool]]

rpred: function[event \rightarrow function[process \rightarrow bool]]

wvr_pred: function[event \rightarrow function[process \rightarrow bool]] =

$(\lambda i : (\lambda p : \text{wpred}(i)(p) \vee \text{rpred}(i)(p)))$

working: function[process, time \rightarrow bool] =

$(\lambda p, t : (\exists i : \text{wpred}(i)(p) \wedge t_p^i \leq t \wedge t < t_p^{i+1}))$

wvr_defn: Lemma $wvr_pred(i) = (\lambda p : wpred(i)(p) \vee rpred(i)(p))$
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_{*1}^{*2}: function $[process, event \rightarrow Clocktime] = (\lambda p, i : adj_p^{i+1} - adj_p^i)$

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

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

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

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

FIXTIME_bound: Lemma

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

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

RATE_1: Axiom $\text{correct_during}(p, pc_p(T), pc_p(S)) \wedge S \geq T$

$\supset pc_p(S) - pc_p(T) \leq (S - T) * (1 + \rho)$

RATE_2: Axiom $\text{correct_during}(p, pc_p(T), pc_p(S)) \wedge S \geq T$

$\supset pc_p(S) - pc_p(T) \geq (S - T) / (1 + \rho)$

RATE_1_iclock: Lemma

$\text{correct_during}(p, ic_p^i(T), ic_p^i(S)) \wedge S \geq T$

$\supset ic_p^i(S) - ic_p^i(T) \leq (S - T) * (1 + \rho)$

RATE_2_iclock: Lemma

$\text{correct_during}(p, ic_p^i(T), ic_p^i(S)) \wedge S \geq T$

$\supset ic_p^i(S) - ic_p^i(T) \geq (S - T) / (1 + \rho)$

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

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

RATE_2_simplify: Lemma

$\text{correct_during}(p, pc_p(T), pc_p(S)) \wedge S \geq T$

$\supset pc_p(S) - pc_p(T) \geq (S - T) * (1 - \rho)$

RATE_2_simplify_iclock: Lemma

$\text{correct_during}(p, ic_p^i(T), ic_p^i(S)) \wedge S \geq T$

$\supset ic_p^i(S) - ic_p^i(T) \geq (S - T) * (1 - \rho)$

RATE_lemma1: Lemma

$\text{correct_during}(p, pc_p(T), pc_p(S))$

$\wedge \text{correct_during}(q, pc_q(T), pc_q(S)) \wedge S \geq T$

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

RATE_lemma1_iclock: Lemma

$\text{correct_during}(p, ic_p^i(T), ic_p^i(S))$

$\wedge \text{correct_during}(q, ic_q^i(T), ic_q^i(S)) \wedge S \geq T$

$\supset |ic_p^i(S) - ic_q^i(S)| \leq |ic_p^i(T) - ic_q^i(T)| + 2 * \rho * (S - T)$

RATE_lemma2: Lemma

$$\begin{aligned} & \text{correct_during}(p, pc_p(T), pc_p(S)) \wedge S \geq T \\ & \supset |(pc_p(S) - S) - (pc_p(T) - T)| \leq \rho \star (|S - T|) \end{aligned}$$

RATE_lemma2_iclock: Lemma

$$\begin{aligned} & \text{correct_during}(p, ic_p^i(T), ic_p^i(S)) \wedge S \geq T \\ & \supset |(ic_p^i(S) - S) - (ic_p^i(T) - T)| \leq \rho \star (|S - T|) \end{aligned}$$

bnd_delay_init: Axiom

$$\begin{aligned} & \text{wpred}(0)(p) \wedge \text{wpred}(0)(q) \\ & \supset |t_p^0 - t_q^0| \leq \beta' - 2 * \rho \star (S^0 - T^0) \wedge \beta' - 2 * (\rho \star (S^0 - T^0)) \leq \beta \end{aligned}$$

bnd_delay_off_init: Lemma $\text{wpred}(0)(p) \wedge \text{wpred}(0)(q) \supset |s_p^0 - s_q^0| \leq \beta'$ **good_read_pred_ax1: Axiom**

$$\begin{aligned} & \text{correct_during}(p, s_p^i, t_p^{i+1}) \\ & \quad \wedge \text{correct_during}(q, s_q^i, t_q^{i+1}) \wedge |s_p^i - s_q^i| \leq \beta_{\text{read}} \\ & \supset \text{good_read_pred}(i)(p, q) \end{aligned}$$

reading_error3: Axiom

$$\begin{aligned} & \text{good_read_pred}(i)(p, q) \\ & \supset |(\Theta_p^{i+1}(q) - IC_p^i(t_p^{i+1})) - (s_p^i - s_q^i)| \leq \Lambda' \end{aligned}$$

ADJ_lem1: Lemma $\text{correct_during}(p, s_p^i, t_p^{i+1}) \supset (ADJ_p^i = \text{cfn}(p, (\lambda p_1 : \Theta_p^{i+1}(p_1) - IC_p^i(t_p^{i+1}))))$ **ADJ_lem2: Lemma** $\text{correct_during}(p, s_p^i, t_p^{i+1}) \supset (ADJ_p^i = \text{cfn}(p, \Theta_p^{i+1}) - IC_p^i(t_p^{i+1}))$ **read_self: Axiom** $\text{wpred}(i)(p) \supset \Theta_p^{i+1}(p) = IC_p^i(t_p^{i+1})$ **fix_between_sync: Axiom**

$$\text{correct_during}(p, t_p^i, t_p^{i+1}) \supset t_p^i < s_p^i \wedge s_p^i < t_p^{i+1}$$

rts_2_lo: Lemma $\text{wpred}(i)(p) \wedge \text{wpred}(i)(q) \supset |t_p^i - t_q^i| \leq \beta$ **rts_2_hi: Axiom** $\text{wpred}(i)(p) \wedge \text{wpred}(i)(q) \supset |t_p^{i+1} - t_q^{i+1}| \leq \beta$ **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^{*1} , S^{*1} $\{i \leftarrow i + 1\}$

iclock_defn_pr: Prove iclock_defn from $ic_{x1}^{*2}(\star 3)$

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
 $(\forall p : \text{wvr_pred}(i)(p) = ((\lambda p : \text{wpred}(i)(p) \vee \text{rpred}(i)(p))p))$

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 ($\lambda p : \text{wpred}(i)(p) \vee \text{rpred}(i)(p)$)},
 wvr_defn_hack $\{p \leftarrow p@p1\}$

wvr_count_pr: Prove wvr_count from
 wpred_ax,
 count_imp
 {ppred1 \leftarrow wpred(i),
 ppred2 \leftarrow ($\lambda p : \text{wpred}(i)(p) \vee \text{rpred}(i)(p)$),
 $n \leftarrow N$ },
 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 $\{p \leftarrow q\}$,
 $s_{*1}^{*2} \{i \leftarrow 0\}$,
 $s_{*1}^{*2} \{i \leftarrow 0, p \leftarrow q\}$,
 wpred_fixtime_low $\{i \leftarrow 0\}$,
 wpred_fixtime_low $\{p \leftarrow q, i \leftarrow 0\}$,
 $S^{*1} \{i \leftarrow 0\}$

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 $\{S \leftarrow S - adj_p^i, T \leftarrow T - adj_p^i\}$,
 iclock_defn,
 iclock_defn $\{T \leftarrow S\}$

RATE_2_iclock_pr: Prove RATE_2_iclock from
 RATE_2 $\{S \leftarrow S - adj_p^i, T \leftarrow T - adj_p^i\}$,
 iclock_defn,
 iclock_defn $\{T \leftarrow S\}$

RATE_2_simplify_iclock_pr: Prove RATE_2_simplify_iclock from
 RATE_2_simplify $\{S \leftarrow S - adj_p^i, T \leftarrow T - adj_p^i\}$,
 iclock_defn,
 iclock_defn $\{T \leftarrow S\}$

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

RI1hack: Lemma $w \leq x \wedge y \leq z \wedge y \geq x \supset |y - x| \leq |z - w|$

RI1hack_pr: Prove RI1hack 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 $\{p \leftarrow q\}$,
 R11hack
 $\{x \leftarrow pc_q(S),$
 $y \leftarrow pc_p(S),$
 $w \leftarrow pc_q(T) + (S - T) \star (1 - \rho),$
 $z \leftarrow pc_p(T) + (S - T) \star (1 + \rho)\},$
 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 $\{p \leftarrow q\},$
 R11hack
 $\{x \leftarrow ic_q^i(S),$
 $y \leftarrow ic_p^i(S),$
 $w \leftarrow ic_q^i(T) + (S - T) \star (1 - \rho),$
 $z \leftarrow ic_p^i(T) + (S - T) \star (1 + \rho)\},$
 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 $\{i \leftarrow i + 1\}$, $ADJ_{\star 1}^{i+2}$

iclock_ADJ_lem_pr: Prove iclock_ADJ_lem from

iclock_defn $\{T \leftarrow T - ADJ_p^i\}$, iclock_defn $\{i \leftarrow i + 1\}$, $ADJ_{\star 1}^{i+2}$

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

$ADJ_{\star 1}^{i+2}$,
 $adj_{\star 1}^{i+2} \{i \leftarrow i + 1\}$,
 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 \rightarrow bool] =

$$(\lambda i : (\forall p, q : \text{wpred}(i)(p) \wedge \text{wpred}(i)(q) \supset |s_p^i - s_q^i| \leq \beta'))$$

ADJ_pred: function[event \rightarrow bool] =

$$(\lambda i : (\forall p : i \geq 1 \wedge \text{wpred}(i-1)(p) \supset |ADJ_p^{i-1}| \leq \alpha(\lfloor \beta' + 2 * \Lambda' \rfloor)))$$

delay_pred_lr: Lemma

$$\text{delay_pred}(i) \supset (\text{wpred}(i)(p) \wedge \text{wpred}(i)(q) \supset |s_p^i - s_q^i| \leq \beta')$$

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

$$\text{ADJ_pred}(i) \wedge \text{delay_pred}(i) \supset \text{ADJ_pred}(i+1) \wedge \text{delay_pred}(i+1)$$

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

bnd_delay_offset_ind_b: Lemma

$$\text{delay_pred}(i) \wedge \text{ADJ_pred}(i+1) \supset \text{delay_pred}(i+1)$$

good_ReadClock: Lemma

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

good_ReadClock_recover: Axiom

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

delay_prec_enh: Lemma

$$\begin{aligned} & \text{delay_pred}(i) \wedge \text{wpred}(i)(p) \wedge \text{wpred}(i)(q) \\ & \supset |(s_p^i - s_q^i) - (ADJ_p^i - ADJ_q^i)| \leq \pi(\lfloor 2 * \Lambda' + 2 \rfloor, \lfloor \beta' + 2 * \Lambda' \rfloor) \end{aligned}$$

delay_prec_enh_step1: Lemma

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

delay_prec_enh_step1_sym: Lemma

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

prec_enh_hyp1: Lemma

$$\begin{aligned} & \text{delay_pred}(i) \wedge \text{wpred}(i)(p) \wedge \text{wpred}(i)(q) \\ & \supset \text{okay_pairs}((\lambda p_1 : \Theta_p^{i+1}(p_1) - IC_p^i(t_p^{i+1}) - \lfloor s_p^i \rfloor), \\ & \quad (\lambda p_1 : \Theta_q^{i+1}(p_1) - IC_q^i(t_q^{i+1}) - \lfloor s_q^i \rfloor), \\ & \quad 2 * \Lambda' + 2, \\ & \quad \text{wpred}(i)) \end{aligned}$$

prec_enh_hyp_2: Lemma

$$\begin{aligned} & \text{delay_pred}(i) \wedge \text{wpred}(i)(p) \\ & \supset \text{okay_Readpred}((\lambda p_1 : \Theta_p^{i+1}(p_1) - IC_p^i(t_p^{i+1}) - \lfloor s_p^i \rfloor), \\ & \quad \beta' + 2 * \Lambda', \\ & \quad \text{wpred}(i)) \end{aligned}$$

prec_enh_hyp_3: Lemma

$$\begin{aligned} & \text{delay_pred}(i) \wedge \text{wpred}(i)(q) \\ & \supset \text{okay_Readpred}((\lambda p_1 : \Theta_q^{i+1}(p_1) - IC_q^i(t_q^{i+1}) - \lfloor s_q^i \rfloor), \\ & \quad \beta' + 2 * \Lambda', \\ & \quad \text{wpred}(i)) \end{aligned}$$

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),
 $Y \leftarrow \lfloor \beta' + 2 * \Lambda' \rfloor$,
 $X \leftarrow \lfloor 2 * \Lambda' + 2 \rfloor$,
 $\gamma \leftarrow (\lambda p_1 : \Theta_p^{i+1}(p_1) - IC_p^i(t_p^{i+1}) - \lfloor s_p^i \rfloor)$,
 $\theta \leftarrow (\lambda p_1 : \Theta_q^{i+1}(p_1) - IC_q^i(t_q^{i+1}) - \lfloor s_q^i \rfloor)}$,

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
 $\{\gamma \leftarrow \Theta_q^{i+1}$,
 $y \leftarrow \beta' + 2 * \Lambda'$,
 $ppred \leftarrow wpred(i)$,
 $l \leftarrow l@p2$,
 $m \leftarrow m@p2\}$

bn_dcl_off_0_pr: Prove bn_dcl_delay_offset_0 from

ADJ_pred $\{i \leftarrow 0\}$,
 delay_pred $\{i \leftarrow 0\}$,
 bn_dcl_delay_off_init $\{p \leftarrow p@p2, q \leftarrow q@p2\}$

bn_dcl_delay_offset_ind_pr: Prove bn_dcl_delay_offset_ind from

bn_dcl_delay_offset_ind_a, bn_dcl_delay_offset_ind_b

bn_dcl_delay_offset_pr: Prove bn_dcl_delay_offset from

induction $\{\text{prop} \leftarrow (\lambda i : \text{ADJ_pred}(i) \wedge \text{delay_pred}(i))\}$,
 bn_dcl_delay_offset_0,
 bn_dcl_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 $\{x \leftarrow ((a - c) - (d - e)), y \leftarrow ((d - f) - (b - c))\}$

abs_hack2: Lemma $|a| \leq b \wedge |c| \leq d \wedge |e| \leq d \supset |a| + |c| + |e| \leq b + 2 * d$

abs_hack2_pr: Prove abs_hack2

good_ReadClock_pr: Prove good_ReadClock from

okay_Readpred

$$\{\gamma \leftarrow \Theta_p^{i+1},$$

$$y \leftarrow \beta' + 2 * \Lambda',$$

$$\text{ppred} \leftarrow \text{wpred}(i)\},$$

$$\text{delay_pred} \{p \leftarrow l@p1, q \leftarrow m@p1\},$$

$$\text{delay_pred} \{q \leftarrow l@p1\},$$

$$\text{delay_pred} \{q \leftarrow m@p1\},$$

$$\text{reading_error3} \{q \leftarrow l@p1\},$$

$$\text{reading_error3} \{q \leftarrow m@p1\},$$

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

$$\{a \leftarrow e@p7 - f@p7,$$

$$b \leftarrow \beta',$$

$$c \leftarrow ((a@p7 - c@p7) - (d@p7 - e@p7)),$$

$$d \leftarrow \Lambda',$$

$$e \leftarrow ((b@p7 - c@p7) - (d@p7 - f@p7))\},$$

$$\text{good_read_pred_ax1} \{q \leftarrow l@p1\},$$

$$\text{good_read_pred_ax1} \{q \leftarrow m@p1\},$$

wpred_fixtime,

$$\text{wpred_fixtime} \{p \leftarrow l@p1\},$$

$$\text{wpred_fixtime} \{p \leftarrow m@p1\},$$

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
 $\{ppred \leftarrow wpred(i),$
 $\gamma \leftarrow \Theta_p^{i+1},$
 $p \leftarrow p@p1,$
 $q \leftarrow p@p1,$
 $X \leftarrow \lfloor \beta' + 2 * \Lambda' \rfloor\}$,

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 $wpred(i)(p)$
 $\supset ADJ_p^i - X = cfu(p, (\lambda p1 : \Theta_p^{i+1}(p1) - IC_p^i(t_p^{i+1}) - X))$

ADJ_hack_pr: Prove ADJ_hack from
 ADJ_lem1,
 translation_invariance
 $\{\gamma \leftarrow (\lambda p1 \rightarrow \text{Clocktime} : \Theta_p^{i+1}(p1) - IC_p^i(t_p^{i+1})),$
 $X \leftarrow -X\}$,
 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 { $q \leftarrow p_3 @ p_1$ },

delay_pred { $p \leftarrow q, q \leftarrow p_3 @ p_1$ },

reading_error3 { $q \leftarrow p_3 @ p_1$ },

reading_error3 { $p \leftarrow q, q \leftarrow p_3 @ p_1$ },

good_read_pred_ax1 { $q \leftarrow p_3 @ p_1$ },

good_read_pred_ax1 { $p \leftarrow q, q \leftarrow p_3 @ p_1$ },

abshack7

$$\{a \leftarrow \Theta_p^{i+1}(p_3 @ p_1),$$

$$b \leftarrow IC_p^i(t_p^{i+1}),$$

$$c \leftarrow s_p^i,$$

$$d \leftarrow s_{p_3 @ p_1}^i,$$

$$e \leftarrow \Theta_q^{i+1}(p_3 @ p_1),$$

$$f \leftarrow IC_q^i(t_q^{i+1}),$$

$$g \leftarrow s_q^i,$$

$$h \leftarrow \Lambda'\},$$

wpred_fixtime,

wpred_fixtime { $p \leftarrow q$ },

wpred_fixtime { $p \leftarrow p_3 @ p_1$ },

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 \rightarrow 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

delay_pred(i) \wedge ADJ_pred($i + 1$)
 \wedge rpred(i)(p) \wedge correct_during(p, t_p^{i+1}, t_p^{i+2}) \wedge wpred($i + 1$)(q)
 $\supset |s_p^{i+1} - s_q^{i+1}| \leq \beta'$

good_interval_lem: Lemma

wpred(i)(p) \wedge wpred($i + 1$)(p) \wedge ADJ_pred($i + 1$) \supset good_interval(p, i, S^{i+1})

betaprime_ax: Axiom

$4 * \rho * (R + \alpha(\lfloor \beta' + 2 * \Lambda' \rfloor)) + \pi(\lfloor 2 * (\Lambda' + 1) \rfloor, \lfloor \beta' + 2 * \Lambda' \rfloor) \leq \beta'$

betaprime_ind_lem: Lemma

ADJ_pred($i + 1$) \wedge wpred(i)(p)
 $\supset 2 * \rho * (R + \alpha(\lfloor \beta' + 2 * \Lambda' \rfloor)) + \pi(\lfloor 2 * (\Lambda' + 1) \rfloor, \lfloor \beta' + 2 * \Lambda' \rfloor) \leq \beta'$

betaprime_lem: Lemma

$2 * \rho * (R + \alpha(\lfloor \beta' + 2 * \Lambda' \rfloor)) + \pi(\lfloor 2 * (\Lambda' + 1) \rfloor, \lfloor \beta' + 2 * \Lambda' \rfloor) \leq \beta'$

R_0_lem: Lemma wpred(i)(p) \wedge ADJ_pred($i + 1$) $\supset R > 0$

bound_future: Lemma

delay_pred(i) \wedge ADJ_pred($i + 1$)
 \wedge wpred(i)(p)
 \wedge wpred(i)(q) \wedge good_interval(p, i, T) \wedge good_interval(q, i, T)
 $\supset |ic_p^{i+1}(T) - ic_q^{i+1}(T)|$
 $\leq 2 * \rho * (|T - S^i| + \alpha(\lfloor \beta' + 2 * \Lambda' \rfloor))$
 $+ \pi(\lfloor 2 * (\Lambda' + 1) \rfloor, \lfloor \beta' + 2 * \Lambda' \rfloor)$

bound_future1: Lemma

delay_pred(i) \wedge ADJ_pred($i + 1$) \wedge wpred(i)(p) \wedge good_interval(p, i, T)
 $\supset |(ic_p^i(T - ADJ_p^i) - s_p^i) - (T - ADJ_p^i - S^i)|$
 $\leq \rho * (|T - S^i| + \alpha(\lfloor \beta' + 2 * \Lambda' \rfloor))$

bound_future1_step: Lemma

$$\begin{aligned} & \text{delay_pred}(i) \wedge \text{ADJ_pred}(i+1) \wedge \text{wpred}(i)(p) \wedge \text{good_interval}(p, i, T) \\ & \supset |(ic_p^i(T - \text{ADJ}_p^i) - s_p^i) - (T - \text{ADJ}_p^i - S^i)| \leq \rho \star (|T - \text{ADJ}_p^i - S^i|) \end{aligned}$$

bound_FIXTIME: Lemma

$$\begin{aligned} & \text{delay_pred}(i) \wedge \text{ADJ_pred}(i+1) \\ & \quad \wedge \text{wpred}(i)(p) \\ & \quad \quad \wedge \text{wpred}(i)(q) \\ & \quad \quad \quad \wedge \text{good_interval}(p, i, S^{i+1}) \wedge \text{good_interval}(q, i, S^{i+1}) \\ & \supset |s_p^{i+1} - s_q^{i+1}| \leq \beta' \end{aligned}$$

bound_FIXTIME2: Lemma

$$\begin{aligned} & \text{delay_pred}(i) \wedge \text{ADJ_pred}(i+1) \wedge \text{wpred}(i)(p) \wedge \text{wpred}(i)(q) \\ & \supset (\text{wpred}(i+1)(p) \wedge \text{wpred}(i+1)(q) \supset |s_p^{i+1} - s_q^{i+1}| \leq \beta') \end{aligned}$$

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

$$\text{ADJ_bound: Lemma } \text{wpred}(i)(p) \supset |\text{ADJ}_p^i| \leq \alpha(\lfloor \beta' + 2 * \Lambda' \rfloor)$$

$$\text{Alpha}_0: \text{Lemma } \text{wpred}(i)(p) \supset \alpha(\lfloor \beta' + 2 * \Lambda' \rfloor) \geq 0$$

Proof

ADJ_pred_lr: Lemma

$$\text{ADJ_pred}(i+1) \supset (\text{wpred}(i)(p) \supset |\text{ADJ}_p^i| \leq \alpha(\lfloor \beta' + 2 * \Lambda' \rfloor))$$

ADJ_pred_lr_pr: Prove ADJ_pred_lr from ADJ_pred {i ← i + 1}

betaprime_ind_lem_pr: Prove betaprime_ind_lem from

$$\begin{aligned} & \text{betaprime_ax,} \\ & \text{pos_product } \{x \leftarrow \rho, y \leftarrow R + \alpha(\lfloor \beta' + 2 * \Lambda' \rfloor)\}, \\ & \text{rho}_0, \\ & \text{R_FIX_SYNC}_0, \\ & \text{FIX_SYNC,} \\ & \text{ADJ_pred_lr,} \\ & |\star 1| \{x \leftarrow \text{ADJ}_p^i\} \end{aligned}$$

betaprime_lem_pr: Prove betaprime_lem from

$$\begin{aligned} & \text{betaprime_ind_lem } \{p \leftarrow p@p4\}, \\ & \text{bnd_delay_offset } \{i \leftarrow i + 1\}, \\ & \text{wpred_ax,} \\ & \text{count_exists } \{\text{ppred} \leftarrow \text{wpred}(i@p1), n \leftarrow N\}, \\ & \text{N_maxfaults} \end{aligned}$$

delay_offset_pr: Prove delay_offset from bnd_delay_offset, delay_pred

ADJ_bound_pr: Prove ADJ_bound from

$$\text{bnd_delay_offset } \{i \leftarrow i + 1\}, \text{ADJ_pred } \{i \leftarrow i + 1\}$$

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 $\text{wpred}(i)(p) \wedge \text{ADJ_pred}(i+1) \supset S^{i+1} - S^i > 0$

R_0_hack_pr: Prove R_0_hack from

ADJ_pred $\{i \leftarrow i+1\}$,

FIXTIME_bound,

wpred_hi_lem,

abs_0 $\{a_1 \leftarrow ADJ_p^i, b_1 \leftarrow \alpha([\beta' + 2 * \Lambda'])\}$

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

$\{z \leftarrow \rho,$

$y \leftarrow |T - ADJ_p^i - S^i|,$

$x \leftarrow |T - S^i| + \alpha([\beta' + 2 * \Lambda'])\}$,

rho_0

bound_future1_step_a: Lemma

$\text{correct_during}(p, ic_p^i(T - ADJ_p^i), s_p^i) \wedge S^i \geq T - ADJ_p^i$

$\supset |(ic_p^i(T - ADJ_p^i) - s_p^i) - (T - ADJ_p^i - S^i)| \leq \rho \star (|T - ADJ_p^i - S^i|)$

bound_future1_step_b: Lemma

$\text{correct_during}(p, s_p^i, ic_p^i(T - ADJ_p^i)) \wedge T - ADJ_p^i \geq S^i$

$\supset |(ic_p^i(T - ADJ_p^i) - s_p^i) - (T - ADJ_p^i - S^i)| \leq \rho \star (|T - ADJ_p^i - S^i|)$

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_{*1}^{*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_{*1}^{*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 $\{T \leftarrow S^{i+1}\}$,

$s_{*1}^{*2} \{i \leftarrow i + 1\}$,

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^{*1} ,

$S^{*1} \{i \leftarrow i + 1\}$,

abs_ge0 $\{x \leftarrow R\}$,

R_0_lem,

$s_{*1}^{*2} \{p \leftarrow p@p1, i \leftarrow i + 1\}$,

$s_{*1}^{*2} \{p \leftarrow q@p1, i \leftarrow i + 1\}$,

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 $\{x \leftarrow s_{p@p2}^{i+1}, y \leftarrow s_{q@p2}^{i+1}\}$,
 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 $|(a - e) - (c - f - d)| \leq aa$

$$\wedge |(b - g) - (c - h - d)| \leq aa \wedge |(e - g) - (f - h)| \leq bb \\ \supset |a - b| \leq 2 * aa + bb$$

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
 $\{a \leftarrow ic_p^i(T - ADJ_p^i),$
 $b \leftarrow ic_q^i(T - ADJ_q^i),$
 $c \leftarrow T,$
 $d \leftarrow S^i,$
 $e \leftarrow s_p^i,$
 $f \leftarrow ADJ_p^i,$
 $g \leftarrow s_q^i,$
 $h \leftarrow ADJ_q^i,$
 $aa \leftarrow \rho * (|T - S^i| + \alpha(\lfloor \beta' + 2 * \Lambda' \rfloor)),$
 $bb \leftarrow \pi(\lfloor 2 * (\Lambda' + 1) \rfloor, \lfloor \beta' + 2 * \Lambda' \rfloor)\}$

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

γ : Var function[process \rightarrow Clocktime]

ppred, ppred1: Var function[process \rightarrow bool]

option1, option2: bool

option1_defn: Axiom

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

option2_defn: Axiom

$$\begin{aligned} \text{option2} \supset T_p^{i+1} &= (i + 1) * R + T^0 - ADJ_p^i \\ &\wedge (\beta = \beta' - 2 * \rho * (S^0 - T^0)) \end{aligned}$$

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

$$\begin{aligned} &(\beta = \beta' - 2 * \rho * (S^0 - T^0)) \\ &\supset 2 * \rho * ((R - (S^0 - T^0)) + \alpha([\beta' + 2 * \Lambda'])) \\ &\quad + \pi([2 * (\Lambda' + 1)], [\beta' + 2 * \Lambda']) \\ &\leq \beta \end{aligned}$$

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 $\text{option1} \vee \text{option2}$

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^{*1} ,

delay_offset,

wpred_fixtime,

wpred_fixtime $\{p \leftarrow q\}$,

synctime_defn,

synctime_defn $\{p \leftarrow q\}$,

s_{*1}^{*2} ,

$s_{*1}^{*2} \{p \leftarrow q\}$,

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^{*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] =
 $(\lambda p, q, i : (\text{if } t_p^i \geq t_q^i \text{ then } p \text{ else } q \text{ end if}))$

maxsync_correct: Lemma $\text{correct}(p, s) \wedge \text{correct}(q, s) \supset \text{correct}((p \uparrow q)[i], s)$

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

minsync_correct: Lemma $\text{correct}(p, s) \wedge \text{correct}(q, s) \supset \text{correct}((p \downarrow q)[i], s)$

minsync_maxsync: Lemma $t_{(p \downarrow q)[i]}^i \leq t_{(p \uparrow q)[i]}^i$

$t_{\star 1, \star 2}^{\star 3}$: Definition function[process, process, event \rightarrow time] =
 $(\lambda p, q, i : t_{(p \uparrow q)[i]}^i)$

delay_recovery: Axiom

$$\text{wpred}(i)(p) \wedge \text{wvr_pred}(i)(q) \supset |t_p^{i+1} - t_q^{i+1}| \leq \beta$$

rts0_new: Axiom $\text{wpred}(i)(p)$

$$\supset t_p^{i+1} - t_p^i \leq (1 + \rho) \star (R + \alpha(\lfloor \beta' + 2 \star \Lambda' \rfloor))$$

rts1_new: Axiom $\text{wpred}(i)(p)$

$$\supset ((R - \alpha(\lfloor \beta' + 2 \star \Lambda' \rfloor)) / (1 + \rho)) \leq t_p^{i+1} - t_p^i$$

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

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

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

lemma_1.2: Lemma $\text{wpred}(i)(p) \wedge \text{wpred}(i+1)(q) \supset t_p^{i+1} < t_q^{i+2}$

lemma_2.1: Lemma $\text{correct}(q, t_q^{i+1})$

$$\supset IC_q^{i+1}(t_q^{i+1}) = \text{cfn}(q, \Theta_q^{i+1})$$

rts_2_lo.i: Lemma

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

rts_2_lo.i_recover: Lemma

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

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

working_clocks_lo: Lemma

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

working_clocks_hi: Lemma

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

working_clocks_interval: Lemma

$$\begin{aligned} & i > 0 \wedge \text{wpred}(i)(p) \\ & \quad \wedge \text{wpred}(j)(q) \wedge t_p^i \leq t \wedge t < t_p^{i+1} \wedge t_q^j \leq t \wedge t < t_q^{j+1} \\ & \supset t_q^{i-1} < t_q^{j+1} \wedge t_q^j < t_q^{i+2} \end{aligned}$$

Proof

working_clocks_lo_pr: Prove working_clocks_lo from

$$\text{lemma_1_1 } \{p \leftarrow q, q \leftarrow p\}$$

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

$$\text{delay_recovery, wpred_preceding } \{p \leftarrow q\}, \text{wvr_pred } \{p \leftarrow q\}$$

rts_2_lo.i_pr: Prove rts_2_lo.i from

$$\begin{aligned} & \text{rts_2_lo_i_recover,} \\ & \text{rts_2_lo_i_recover } \{p \leftarrow q, q \leftarrow p\}, \\ & \text{abs_com } \{x \leftarrow t_p^{i+1}, y \leftarrow t_q^{i+1}\}, \\ & \text{rts_2_hi,} \\ & \text{wpred_preceding,} \\ & \text{wpred_preceding } \{p \leftarrow q\} \end{aligned}$$

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

$$\text{rts_2_hi, rts1_new, } |\star 1| \{x \leftarrow t_p^{i+1} - t_q^{i+1}\}, \text{nonoverlap}$$

lemma_2_1_proof: Prove lemma_2_1 from
 |Clock_defn $\{p \leftarrow q, i \leftarrow i + 1, t \leftarrow t_q^{i+1}\}$,
 adj_{x1}^{x2} $\{i \leftarrow i + 1, p \leftarrow q\}$

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 \rightarrow bool] =

$$(\lambda p, i : \text{wpred}(i)(p) \supset t_p^{i+1} - t_p^i \leq (1 + \rho) * (R + \alpha(\lfloor \beta' + 2 * \Lambda' \rfloor)))$$

rmin_pred: function[process, event \rightarrow bool] =

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

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

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
 $\{z \leftarrow (1 + \rho),$
 $y \leftarrow R - ADJ_p^{i@p1},$
 $x \leftarrow R - \alpha(|\beta' + 2 * \Lambda'|)\}$,
 rho_0,
 ADJ_bound $\{i \leftarrow i@p1\}$,
 |*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
 $\{z \leftarrow (1 + \rho),$
 $y \leftarrow R - ADJ_p^{i@p1},$
 $x \leftarrow R - \alpha(|\beta' + 2 * \Lambda'|)\}$,
 rho_0,
 ADJ_bound $\{i \leftarrow i@p1\}$,
 |*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 $\{i \leftarrow \text{pred}(i)\}$

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_liclock $\{i \leftarrow i@p1, S \leftarrow T_p^{i@p1+1}, T \leftarrow T^0\}$,
wpred_correct $\{i \leftarrow i@p1\}$,
mult_leq_2
 $\{z \leftarrow (1 + \rho),$
 $y \leftarrow R - ADJ_p^{i@p1},$
 $x \leftarrow R + \alpha(\lfloor \beta' + 2 * \Lambda' \rfloor)\}$,
mult_com $\{x \leftarrow (T_p^{i@p1+1} - T^0), y \leftarrow (1 + \rho)\}$,
rho_0,
ADJ_bound $\{i \leftarrow i@p1\}$,
 $\lfloor * 1 \rfloor \{x \leftarrow ADJ_p^{i@p1}\}$,
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_liclock
 $\{i \leftarrow i@p1,$
 $S \leftarrow T_p^{i@p1+1},$
 $T \leftarrow T_p^{i@p1} + ADJ_p^i\}$,
wpred_correct $\{i \leftarrow i@p1\}$,
mult_leq_2
 $\{z \leftarrow (1 + \rho),$
 $y \leftarrow R - ADJ_p^{i@p1},$
 $x \leftarrow R + \alpha(\lfloor \beta' + 2 * \Lambda' \rfloor)\}$,
mult_com $\{x \leftarrow (T_p^{i@p1+1} - (T_p^{i@p1} + ADJ_p^i)), y \leftarrow (1 + \rho)\}$,
rho_0,
ADJ_bound $\{i \leftarrow i@p1\}$,
 $\lfloor * 1 \rfloor \{x \leftarrow ADJ_p^{i@p1}\}$,
R_bound $\{i \leftarrow i@p1\}$,
wpred_hi_lem $\{i \leftarrow i@p1\}$,
Alpha_0 $\{i \leftarrow i@p1\}$,
liclock_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 $\{i \leftarrow \text{pred}(i)\}$
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_liclock $\{i \leftarrow i@p1, S \leftarrow T_p^{i@p1+1}, T \leftarrow T^0\}$,
 wpred_correct $\{i \leftarrow i@p1\}$,
 Alpha_0 $\{i \leftarrow i@p1\}$,
 div_ineq $\{z \leftarrow (1 + \rho), y \leftarrow R, x \leftarrow R - \alpha(\lfloor \beta' + 2 * \Lambda' \rfloor)\}$,
 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_liclock
 $\{i \leftarrow i@p1,$
 $S \leftarrow T_p^{i@p1+1},$
 $T \leftarrow T_p^{i@p1} + ADJ_p^i\}$,
 wpred_correct $\{i \leftarrow i@p1\}$,
 Alpha_0 $\{i \leftarrow i@p1\}$,
 div_ineq
 $\{z \leftarrow (1 + \rho),$
 $y \leftarrow R - ADJ_p^i,$
 $x \leftarrow R - \alpha(\lfloor \beta' + 2 * \Lambda' \rfloor)\}$,
 rho_0,
 R_bound $\{i \leftarrow i@p1\}$,
 wpred_hi_lem $\{i \leftarrow i@p1\}$,
 |*1| $\{x \leftarrow ADJ_p^i\}$,
 ADJ_recovery,
 ADJ_bound,
 wpred_preceding,
 icklock_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 $\{i \leftarrow \text{pred}(i)\}$

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 $\{z \leftarrow (1 + \rho), y \leftarrow R, x \leftarrow R + \alpha(\lfloor \beta' + 2 * \Lambda' \rfloor)\}$,
 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
 $\{z \leftarrow (1 + \rho),$
 $y \leftarrow R - ADJ_p^i,$
 $x \leftarrow R + \alpha(\lfloor \beta' + 2 * \Lambda' \rfloor)\}$,
 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 $\{i \leftarrow i@p1\}$,
 |*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

ϑ : Var function[process \rightarrow Clocktime]

i, j, k : Var posint

T, X, Y, Z : Var Clocktime

cfm_{MID} : function[process, function[process \rightarrow Clocktime] \rightarrow Clocktime] =
 $(\lambda p, \vartheta : [(\vartheta_{(F+1)} + \vartheta_{(N-F)})/2])$

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

Proof

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

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

ft_mid_trans_inv_pr: Prove ft_mid_trans_inv from

cfm_{MID} ,

$cfm_{MID} \{ \vartheta \leftarrow (\lambda q : \vartheta(q) + X) \}$,

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 $\{x \leftarrow (\vartheta_{(F+1)} + \vartheta_{(N-F)}), y \leftarrow 2 * X, z \leftarrow 2\}$,

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 \rightarrow Clocktime]

ppred, ppred1, ppred2: Var function[process \rightarrow bool]

good_greater.F1: Lemma

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

good_less.NF: Lemma

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

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

$\{\text{ppred} \leftarrow (\lambda p_1 : \text{ppred1}@p4(p_1) \wedge \text{ppred2}@p4(p_1)),$
 $n \leftarrow N\}$,

pigeon_hole

$\{\text{ppred1} \leftarrow \text{ppred},$
 $\text{ppred2} \leftarrow (\lambda p_1 : \vartheta(p_1) \geq \vartheta_{(F+1)}),$
 $n \leftarrow N,$
 $k \leftarrow 1\}$

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)
 $\wedge \theta(q) \leq \theta(p) \wedge \gamma(p) \leq \gamma(q) \wedge \text{okay_pairs}(\theta, \gamma, X, \text{ppred})$
 $\supset |\theta(p) - \gamma(q)| \leq X$

good_geq_F_add1: Lemma

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

okay_pair_geq_F_add1: Lemma

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

good_between: Lemma

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

ft_mid_precision_enhancement: Lemma

count(ppred, N) $\geq N - F$
 $\wedge \text{okay_pairs}(\theta, \gamma, X, \text{ppred})$
 $\wedge \text{okay_Readpred}(\theta, Z, \text{ppred}) \wedge \text{okay_Readpred}(\gamma, Z, \text{ppred})$
 $\supset |\text{cfn}_{MID}(p, \theta) - \text{cfn}_{MID}(q, \gamma)| \leq \text{ft_mid_Pi}(X, Z)$

ft_mid_prec_enh_sym: Lemma

$$\begin{aligned} & \text{count}(\text{ppred}, N) \geq N - F \\ & \quad \wedge \text{okay_pairs}(\theta, \gamma, X, \text{ppred}) \\ & \quad \quad \wedge \text{okay_Readpred}(\theta, Z, \text{ppred}) \\ & \quad \quad \quad \wedge \text{okay_Readpred}(\gamma, Z, \text{ppred}) \wedge (\text{cfn}_{MID}(p, \theta) > \text{cfn}_{MID}(q, \gamma)) \\ & \quad \supset |\text{cfn}_{MID}(p, \theta) - \text{cfn}_{MID}(q, \gamma)| \leq \text{ft_mid_Pi}(X, Z) \end{aligned}$$

ft_mid_eq: Lemma $\text{count}(\text{ppred}, N) \geq N - F$

$$\begin{aligned} & \quad \wedge \text{okay_pairs}(\theta, \gamma, X, \text{ppred}) \\ & \quad \quad \wedge \text{okay_Readpred}(\theta, Z, \text{ppred}) \\ & \quad \quad \quad \wedge \text{okay_Readpred}(\gamma, Z, \text{ppred}) \wedge (\text{cfn}_{MID}(p, \theta) = \text{cfn}_{MID}(q, \gamma)) \\ & \quad \supset |\text{cfn}_{MID}(p, \theta) - \text{cfn}_{MID}(q, \gamma)| \leq \text{ft_mid_Pi}(X, Z) \end{aligned}$$

ft_mid_prec_sym1: Lemma

$$\begin{aligned} & \text{count}(\text{ppred}, N) \geq N - F \\ & \quad \wedge \text{okay_pairs}(\theta, \gamma, X, \text{ppred}) \\ & \quad \quad \wedge \text{okay_Readpred}(\theta, Z, \text{ppred}) \\ & \quad \quad \quad \wedge \text{okay_Readpred}(\gamma, Z, \text{ppred}) \\ & \quad \quad \quad \quad \wedge ((\theta_{(F+1)} + \theta_{(N-F)}) > (\gamma_{(F+1)} + \gamma_{(N-F)})) \\ & \quad \supset |(\theta_{(F+1)} + \theta_{(N-F)}) - (\gamma_{(F+1)} + \gamma_{(N-F)})| \leq Z + 2 * X \end{aligned}$$

mid_gt_imp_sel_gt: Lemma

$$\begin{aligned} & (\text{cfn}_{MID}(p, \theta) > \text{cfn}_{MID}(q, \gamma)) \\ & \quad \supset ((\theta_{(F+1)} + \theta_{(N-F)}) > (\gamma_{(F+1)} + \gamma_{(N-F)})) \end{aligned}$$

okay_pairs_sym: Lemma

$$\text{okay_pairs}(\theta, \gamma, X, \text{ppred}) \supset \text{okay_pairs}(\gamma, \theta, X, \text{ppred})$$

Proof

ft_mid_prec_sym1-pr: Prove ft_mid_prec_sym1 from

$$\begin{aligned} & \text{good_between,} \\ & \text{okay_pair_geq_F_add1,} \\ & \text{good_less_NF } \{\vartheta \leftarrow \gamma\}, \\ & \text{abs_geq} \\ & \quad \{x \leftarrow (\gamma(q_1 @ p_2) - \gamma(p @ p_3)) + (\theta(p @ p_1) - \gamma(p @ p_1)) \\ & \quad \quad + (\theta(p_1 @ p_2) - \gamma(q_1 @ p_2)), \\ & \quad \quad y \leftarrow (\theta_{(F+1)} + \theta_{(N-F)}) - (\gamma_{(F+1)} + \gamma_{(N-F)})\}, \\ & \text{abs_plus} \\ & \quad \{x \leftarrow (\gamma(q_1 @ p_2) - \gamma(p @ p_3)) + (\theta(p @ p_1) - \gamma(p @ p_1)), \\ & \quad \quad y \leftarrow (\theta(p_1 @ p_2) - \gamma(q_1 @ p_2))\}, \\ & \text{abs_plus } \{x \leftarrow (\gamma(q_1 @ p_2) - \gamma(p @ p_3)), y \leftarrow (\theta(p @ p_1) - \gamma(p @ p_1))\}, \\ & \text{okay_pairs } \{\gamma \leftarrow \theta, \theta \leftarrow \gamma, x \leftarrow X, p_3 \leftarrow p @ p_1\}, \\ & \text{okay_Readpred } \{\gamma \leftarrow \gamma, y \leftarrow Z, l \leftarrow q_1 @ p_2, m \leftarrow p @ p_3\}, \\ & \text{distrib } \{x \leftarrow 1, y \leftarrow 1, z \leftarrow X\}, \\ & \text{mult_lident } \{x \leftarrow X\} \end{aligned}$$

mid_gt_imp_sel_gt_pr: Prove mid_gt_imp_sel_gt from

$cf_{nMID} \{\vartheta \leftarrow \theta\},$
 $cf_{nMID} \{\vartheta \leftarrow \gamma, p \leftarrow q\},$
 $mult_div \{x \leftarrow (\theta_{(F+1)} + \theta_{(N-F)}), y \leftarrow 2\},$
 $mult_div \{x \leftarrow (\gamma_{(F+1)} + \gamma_{(N-F)}), y \leftarrow 2\},$
 $mult_floor_gt \{x \leftarrow x@p3/2, y \leftarrow x@p4/2, z \leftarrow 2\}$

ft_mid_eq_pr: Prove ft_mid_eq from

$count_exists \{n \leftarrow N\},$
 $ft_mid_maxfaults,$
 $okay_pairs \{\gamma \leftarrow \theta, \theta \leftarrow \gamma, x \leftarrow X, p_3 \leftarrow p@p1\},$
 $okay_Readpred \{\gamma \leftarrow \gamma, y \leftarrow Z, l \leftarrow p@p1, m \leftarrow p@p1\},$
 $|\star 1| \{x \leftarrow cf_{nMID}(p, \theta) - cf_{nMID}(q, \gamma)\},$
 $|\star 1| \{x \leftarrow \gamma(p@p1) - \gamma(p@p1)\},$
 $|\star 1| \{x \leftarrow \theta(p@p1) - \gamma(p@p1)\},$
 $ceil_defn \{x \leftarrow Z/2 + X\},$
 $div_nonnegative \{x \leftarrow Z, y \leftarrow 2\}$

ft_mid_prec_enh_sym_pr: Prove ft_mid_prec_enh_sym from

$cf_{nMID} \{\vartheta \leftarrow \theta\},$
 $cf_{nMID} \{\vartheta \leftarrow \gamma, p \leftarrow q\},$
 $div_minus_distrib$
 $\{x \leftarrow (\theta_{(F+1)} + \theta_{(N-F)}),$
 $y \leftarrow (\gamma_{(F+1)} + \gamma_{(N-F)}),$
 $z \leftarrow 2\},$
 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 ← q@p1,
   q ← p@p1,
   θ ← γ@p1,
   γ ← θ@p1},
ft_mid_eq,
okay_pairs_sym,
abs_com {x ← cfnMID(p, θ), y ← cfnMID(q, γ)}

```

okay_pair_geq_F_add1_pr: Prove

```

okay_pair_geq_F_add1
  {p1 ← if (θ(p@p2) ≥ θ(p@p1))
    then p@p2
    elsif (γ(p@p1) ≥ γ(p@p2)) then p@p1 else p@p3
  end if,
  q1 ← if (θ(p@p2) ≥ θ(p@p1))
    then p@p2
    elsif (γ(p@p1) ≥ γ(p@p2)) then p@p1 else q@p3
  end if} from
good_geq_F_add1 {ϑ ← θ},
good_geq_F_add1 {ϑ ← γ},
exchange_order {p ← p@p1, q ← p@p2},
okay_pairs {γ ← θ, θ ← γ, x ← X, p3 ← p@p1},
okay_pairs {γ ← θ, θ ← γ, x ← X, p3 ← p@p2}

```

good_geq_F_add1_pr: Prove good_geq_F_add1 {p ← p@p1} from

```

count_exists
  {ppred ← (λ p : ((ppred1@p2)p) ∧ ((ppred2@p2)p)),
   n ← N},
pigeon_hole
  {n ← N,
   k ← 1,
   ppred1 ← ppred,
   ppred2 ← (λ p : ϑ(p) ≥ ϑ((k@p3)))},
count_geq_select {k ← F + 1},
ft_mid_maxfaults

```

good_between_pr: Prove good_between $\{p \leftarrow p@p1\}$ from

count_exists

$\{\text{ppred} \leftarrow (\lambda p : ((\text{ppred1}@p2)p) \wedge ((\text{ppred2}@p2)p)),$
 $n \leftarrow N\},$

pigeon_hole

$\{n \leftarrow N,$
 $k \leftarrow 1,$
 $\text{ppred1} \leftarrow (\lambda p : ((\text{ppred1}@p3)p) \wedge ((\text{ppred2}@p3)p)),$
 $\text{ppred2} \leftarrow (\lambda p : \theta(p) \geq \theta_{((k@p4))})\},$

pigeon_hole

$\{n \leftarrow N,$
 $k \leftarrow k@p5,$
 $\text{ppred1} \leftarrow \text{ppred},$
 $\text{ppred2} \leftarrow (\lambda p : \gamma_{((k@p5))} \geq \gamma(p))\},$

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 \rightarrow Clocktime]

ppred, ppred1, ppred2: Var function[process \rightarrow bool]

ft_mid_accuracy_preservation: Lemma

$$\text{ppred}(q) \wedge \text{count}(\text{ppred}, N) \geq N - F \wedge \text{okay_Readpred}(\vartheta, X, \text{ppred}) \\ \supset |cfn_{MID}(p, \vartheta) - \vartheta(q)| \leq X$$

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

$$\text{count}(\text{ppred}, N) \geq N - F \supset (\exists p_1 : \text{ppred}(p_1) \wedge \vartheta(p_1) \leq cfn_{MID}(p, \vartheta))$$

abs_q_greater: Lemma

$$\text{count}(\text{ppred}, N) \geq N - F \supset (\exists p_1 : \text{ppred}(p_1) \wedge \vartheta(p_1) \geq cfn_{MID}(p, \vartheta))$$

ft_mid_bnd_by_good: Lemma

$$\text{count}(\text{ppred}, N) \geq N - F \\ \supset (\exists p_1 : \text{ppred}(p_1) \wedge |cfn_{MID}(p, \vartheta) - \vartheta(q)| \leq |\vartheta(p_1) - \vartheta(q)|)$$

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
 $\{p_1 \leftarrow (\text{if } cfn_{MID}(p, \vartheta) \geq \vartheta(q) \text{ then } p_1@c_1 \text{ else } p_1@c_2 \text{ end if})\}$ 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 $\{x \leftarrow x@p_3 - y@p_3, y \leftarrow x@p_4 - y@p_4\},$
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 $\{p_1 \leftarrow p@p_1\}$ from
good_less_NF, ft_mid_greater

abs_q_greater_pr: Prove abs_q_greater $\{p_1 \leftarrow p@p_1\}$ from
good_greater_F1, ft_mid_less

mult_hack: Lemma $X + X = 2 * X$

mult_hack_pr: Prove mult_hack from $*1 * *2 \{x \leftarrow 2, y \leftarrow X\}$

ft_mid_less_pr: Prove ft_mid_less from
cf_n_{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 $\{x \leftarrow x@p_3/2\}$

ft_mid_greater_pr: Prove ft_mid_greater from
cf_n_{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 $\{x \leftarrow x@p_3/2, y \leftarrow y@p_3/2\},$
floor_int $\{i \leftarrow X@p_5\}$

ft_mid_acc_pres_pr: Prove ft_mid_accuracy_preservation from
ft_mid_bnd_by_good,
okay_Readpred $\{\gamma \leftarrow \vartheta, y \leftarrow X, l \leftarrow p_1@c_1, m \leftarrow q@c\}$

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

ϑ : Var function[process \rightarrow Clocktime]

i, j, k : Var posint

T, X, Y, Z : Var Clocktime

$\star 1_{(\star 2)}$: function[function[process \rightarrow Clocktime], posint \rightarrow 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

ϑ : Var function[process \rightarrow Clocktime]

funsort: function[function[process \rightarrow Clocktime]
 \rightarrow function[posint \rightarrow process]]

(* clock readings can be sorted *)

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

funsort_fun_1.1: Axiom

$i \leq N \wedge j \leq N \wedge \text{funsort}(\vartheta)(i) = \text{funsort}(\vartheta)(j) \supset i = j \wedge \text{funsort}(\vartheta)(i) < N$

funsort_fun_onto: Axiom $p < N \supset (\exists i : i \leq N \wedge \text{funsort}(\vartheta)(i) = p)$

funsort_trans_inv: Axiom

$k \leq N \supset (\vartheta(\text{funsort}((\lambda q : \vartheta(q) + X))(k))) = \vartheta(\text{funsort}(\vartheta)(k))$

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

cnt_sort_leq: Axiom

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

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 [7]. The induction modules are from Rushby's transient recovery verification [3]. 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.¹

¹In 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

$\star 1 \star 2$: function[number, number \rightarrow number] = ($\lambda x, y : (x \star y)$)

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 $\{x \leftarrow (\text{if } y \neq 0 \text{ then } (x/y) \text{ else } 0 \text{ end if})\}, \text{mult_div}$

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 $\{z \leftarrow (\text{if } z \neq 0 \text{ then } (1/z) \text{ else } 0 \text{ end if})\}$

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

$|\star 1| \{x \leftarrow (\text{if } y \neq 0 \text{ then } (x/y) \text{ else } 0 \text{ end if})\}$,
 $|\star 1|$,
 div_nonnegative,
 div_minus_1,
 mult_minus

mult_minus_pr: Prove mult_minus from

mult_div_1 $\{x \leftarrow -1, y \leftarrow x, z \leftarrow y\}$,
 $\star 1 \star 2 \{x \leftarrow -1, y \leftarrow x\}$,
 $\star 1 \star 2 \{x \leftarrow -1, y \leftarrow (\text{if } y \neq 0 \text{ then } (x/y) \text{ else } 1 \text{ end if})\}$

div_minus_1_pr: Prove div_minus_1 from

mult_div,
 pos_product $\{x \leftarrow (\text{if } y \neq 0 \text{ then } (x/y) \text{ else } 0 \text{ end if}), y \leftarrow y\}$

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
 $\{x \leftarrow (\text{if } z \neq 0 \text{ then } (x/z) \text{ else } 0 \text{ end if}),$
 $y \leftarrow (\text{if } z \neq 0 \text{ then } (y/z) \text{ else } 0 \text{ end if})\}$

ceil_plus_mult_div_proof: Prove ceil_plus_mult_div from

ceil_mult_div,
 distrib
 $\{x \leftarrow [(\text{if } y \neq 0 \text{ then } (x/y) \text{ else } 0 \text{ end if})],$
 $y \leftarrow 1,$
 $z \leftarrow y\}$,
 mult_lident $\{x \leftarrow y\}$

```
ceil_mult_div_proof: Prove ceil_mult_div from
  mult_div,
  mult_leq
  {x ← [( if y ≠ 0 then (x/y) else 0 end if)],
   y ← ( if y ≠ 0 then (x/y) else 0 end if),
   z ← y},
  ceil_defn {x ← ( if y ≠ 0 then (x/y) else 0 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] =
 $(\lambda x : (\text{if } x < 0 \text{ then } -x \text{ else } x \text{ end if}))$

iabs: Definition function[integer \rightarrow integer] =
 $(\lambda X : (\text{if } X < 0 \text{ then } -X \text{ else } X \text{ end if}))$

iabs_is_abs: Lemma $x = X \supset \text{iabs}(X) = |x|$

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

$[*1]$: function[number \rightarrow int]

ceil_defn: Axiom $\lceil x \rceil \geq x \wedge \lceil x \rceil - 1 < x$

$[*1]$: 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 $z > 0 \wedge \lfloor x \rfloor > \lfloor y \rfloor \supset x * z > y * z$

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 $\{x \leftarrow x + i, i \leftarrow \lfloor x \rfloor + i, j \leftarrow \lfloor x + i \rfloor\}$

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
 $|\star 1| \{x \leftarrow \lfloor x \rfloor + \lceil y \rceil\}$,
 $|\star 1| \{x \leftarrow \lfloor x + y \rfloor\}$,
 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 $(\text{prop}(0) \wedge (\forall j : \text{prop}(j) \supset \text{prop}(j + 1))) \supset \text{prop}(i)$

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 $\{p \leftarrow \text{prop}, d \leftarrow i, d_2 \leftarrow j\}$

(* Substitution for n in following could simply be $n \leftarrow n - m$
 but then the TCC would not be proveable *)

ind_m_proof: Prove induction_m $\{i \leftarrow j@p1 + m\}$ from
 induction

{proof $\leftarrow (\lambda x : p@c(x + m))$,

$i \leftarrow \text{if } n \geq m \text{ then } n - m \text{ else } 0 \text{ end if}$ }

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** (\exists measure : $a < b \supset$ measure(a) < measure(b))

Theory

p, A, B: **Var** function[dom → bool]

d, d₁, d₂: **Var** dom

general_induction: **Axiom**

$(\forall d_1 : (\forall d_2 : d_2 < d_1 \supset p(d_2)) \supset p(d_1)) \supset (\forall d : p(d))$

d₃, d₄: **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

{d₁ ← d₁@p1,

d₃ ← d₁@p1,

d₄ ← d₂} **from** general_induction {p ← (λ d : A(d) ⊃ B(d))}

End noetherian

D.7 countmod

countmod: Module

Exporting all

Theory

```

i1: Var int
posint: Type from nat with (λ i1 : i1 > 0)
l, m, n, p, q, p1, p2, q1, q2, p3, q3: 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]
ϑ, θ, γ: Var function[nat → number]
countsize: function[function[nat → bool], nat → nat] = (λ ppred, i : i)
count: Recursive function[function[nat → bool], nat → nat] =
  (λ 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((λ q : ¬ppred(q)), n) = n - count(ppred, n)

count_exists: Lemma count(ppred, n) > 0 ⊃ (∃ p : p < n ∧ ppred(p))

count_true: Lemma count((λ p : true), n) = n

count_false: Lemma count((λ p : false), n) = 0

imp_pred: function[function[nat → bool], function[nat → bool] → bool] =
  (λ ppred1, ppred2 : (∀ p : ppred1(p) ⊃ ppred2(p)))

imp_pred_lem: Lemma imp_pred(ppred1, ppred2) ⊃ (ppred1(p) ⊃ ppred2(p))

imp_pred_or: Lemma imp_pred(ppred1, (λ p : ppred1(p) ∨ ppred2(p)))

count_imp: Lemma imp_pred(ppred1, ppred2)
  ⊃ count(ppred1, n) ≤ count(ppred2, n)

count_or: Lemma count(ppred1, n) ≥ k
  ⊃ count((λ p : ppred1(p) ∨ ppred2(p)), n) ≥ k

count_bounded_imp: Lemma count((λ p : p < n ⊃ ppred(p)), n) = count(ppred, n)

```

count_bounded_and: **Lemma** $\text{count}((\lambda p : p < n \wedge \text{ppred}(p)), n) = \text{count}(\text{ppred}, n)$

pigeon_hole: **Lemma**
 $\text{count}(\text{ppred1}, n) + \text{count}(\text{ppred2}, n) \geq n + k$
 $\supset \text{count}((\lambda p : \text{ppred1}(p) \wedge \text{ppred2}(p)), n) \geq k$

pred1, pred2: **Var** function[nat \rightarrow bool]

pred_extensionality: **Axiom** $(\forall p : \text{pred1}(p) = \text{pred2}(p)) \supset \text{pred1} = \text{pred2}$

(* these are in the theory section so the tcc module won't complain *)

nk_type: **Type** = **Record** $n : \text{nat}$,
 $k : \text{nat}$
end record

nk, nk1, nk2: **Var** nk_type
nk_less: function[nk_type, nk_type \rightarrow bool] ==
 $(\lambda nk1, nk2 : nk1.n + nk1.k < nk2.n + nk2.k)$

Proof

Using natinduction, noetherian

imp_pred_lem_pr: **Prove** imp_pred_lem from imp_pred { $p \leftarrow p@c$ }

imp_pred_or_pr: **Prove** imp_pred_or from
imp_pred { $\text{ppred2} \leftarrow (\lambda p : \text{ppred1}(p) \vee \text{ppred2}(p))$ }

count_imp0: **Lemma**
imp_pred(ppred1, ppred2) $\supset \text{count}(\text{ppred1}, 0) \leq \text{count}(\text{ppred2}, 0)$

count_imp_ind: **Lemma**
 $(\text{imp_pred}(\text{ppred1}, \text{ppred2}) \supset \text{count}(\text{ppred1}, n) \leq \text{count}(\text{ppred2}, n))$
 $\supset (\text{imp_pred}(\text{ppred1}, \text{ppred2})$
 $\supset \text{count}(\text{ppred1}, n + 1) \leq \text{count}(\text{ppred2}, n + 1))$

count_imp0_pr: **Prove** count_imp0 from
count { $i \leftarrow 0$, ppred $\leftarrow \text{ppred1}$ }, count { $i \leftarrow 0$, ppred $\leftarrow \text{ppred2}$ }

count_imp_ind_pr: **Prove** count_imp_ind from
count {ppred $\leftarrow \text{ppred1}$, $i \leftarrow n + 1$ },
count {ppred $\leftarrow \text{ppred2}$, $i \leftarrow n + 1$ },
imp_pred { $p \leftarrow n$ }

count_imp_pr: Prove count_imp from induction

{prop ← (λ n :
 (imp_pred(ppred1, ppred2) ⊃ count(ppred1, n) ≤ count(ppred2, n))),
 i ← n@c},
 count_imp0,
 count_imp_ind {n ← j@p1}

count_or_pr: Prove count_or from

count_imp {ppred2 ← (λ p : ppred1(p) ∨ ppred2(p))}, imp_pred_or

count_bounded_imp0: Lemma

$k ≥ 0 ⊃ \text{count}((\lambda p : p < k \supset \text{ppred}(p)), 0) = \text{count}(\text{ppred}, 0)$

count_bounded_imp_ind: Lemma

$(k ≥ n ⊃ \text{count}((\lambda p : p < k \supset \text{ppred}(p)), n) = \text{count}(\text{ppred}, n))$
 $⊃ (k ≥ n + 1$
 $⊃ \text{count}((\lambda p : p < k \supset \text{ppred}(p)), n + 1) = \text{count}(\text{ppred}, n + 1))$

count_bounded_imp_k: Lemma

$(k ≥ n ⊃ \text{count}((\lambda p : p < k \supset \text{ppred}(p)), n) = \text{count}(\text{ppred}, n))$

count_bounded_imp0_pr: Prove count_bounded_imp0 from

count {i ← 0}, count {ppred ← (λ p : p < k ⊃ ppred(p)), i ← 0}

count_bounded_imp_ind_pr: Prove count_bounded_imp_ind from

count {i ← n + 1},
 count {ppred ← (λ p : p < k ⊃ ppred(p)), i ← n + 1}

count_bounded_imp_k_pr: Prove count_bounded_imp_k from induction

{prop ← (λ n :
 $k ≥ n ⊃ \text{count}((\lambda p : p < k \supset \text{ppred}(p)), n) = \text{count}(\text{ppred}, n)$),
 i ← n},
 count_bounded_imp0,
 count_bounded_imp_ind {n ← j@p1}

count_bounded_imp_pr: Prove count_bounded_imp from

count_bounded_imp_k {k ← n}

count_bounded_and0: Lemma

$k ≥ 0 ⊃ \text{count}((\lambda p : p < k \wedge \text{ppred}(p)), 0) = \text{count}(\text{ppred}, 0)$

count_bounded_and_ind: Lemma

$(k ≥ n ⊃ \text{count}((\lambda p : p < k \wedge \text{ppred}(p)), n) = \text{count}(\text{ppred}, n))$
 $⊃ (k ≥ n + 1$
 $⊃ \text{count}((\lambda p : p < k \wedge \text{ppred}(p)), n + 1) = \text{count}(\text{ppred}, n + 1))$

count_bounded_and_k: Lemma

$$(k \geq n \supset \text{count}((\lambda p : p < k \wedge \text{ppred}(p)), n) = \text{count}(\text{ppred}, n))$$

count_bounded_and0_pr: Prove count_bounded_and0 from

$$\text{count } \{i \leftarrow 0\}, \text{count } \{\text{ppred} \leftarrow (\lambda p : p < k \wedge \text{ppred}(p)), i \leftarrow 0\}$$

count_bounded_and_ind_pr: Prove count_bounded_and_ind from

$$\text{count } \{i \leftarrow n + 1\}, \\ \text{count } \{\text{ppred} \leftarrow (\lambda p : p < k \wedge \text{ppred}(p)), i \leftarrow n + 1\}$$

**count_bounded_and_k_pr: Prove count_bounded_and_k from
induction**

$$\{\text{prop} \leftarrow (\lambda n : \\ k \geq n \supset \text{count}((\lambda p : p < k \wedge \text{ppred}(p)), n) = \text{count}(\text{ppred}, n)), \\ i \leftarrow n\}, \\ \text{count_bounded_and0}, \\ \text{count_bounded_and_ind } \{n \leftarrow j@p1\}$$

count_bounded_and_pr: Prove count_bounded_and from

$$\text{count_bounded_and_k } \{k \leftarrow n\}$$

count_false_pr: Prove count_false from

$$\text{count_true}, \\ \text{count_complement } \{\text{ppred} \leftarrow (\lambda p : \text{true})\}, \\ \text{pred_extensionality} \\ \{\text{pred1} \leftarrow (\lambda p : \neg \text{true}), \\ \text{pred2} \leftarrow (\lambda p : \text{false})\}$$

cc0: Lemma $\text{count}((\lambda q : \neg \text{ppred}(q)), 0) = 0 - \text{count}(\text{ppred}, 0)$

cc_ind: Lemma $(\text{count}((\lambda q : \neg \text{ppred}(q)), n) = n - \text{count}(\text{ppred}, n)) \\ \supset (\text{count}((\lambda q : \neg \text{ppred}(q)), n + 1) = n + 1 - \text{count}(\text{ppred}, n + 1))$

cc0_pr: Prove cc0 from

$$\text{count } \{\text{ppred} \leftarrow (\lambda q : \neg \text{ppred}(q)), i \leftarrow 0\}, \text{count } \{i \leftarrow 0\}$$

cc_ind_pr: Prove cc_ind from

$$\text{count } \{\text{ppred} \leftarrow (\lambda q : \neg \text{ppred}(q)), i \leftarrow n + 1\}, \text{count } \{i \leftarrow n + 1\}$$

count_complement_pr: Prove count_complement from

$$\text{induction} \\ \{\text{prop} \leftarrow (\lambda n : \text{count}((\lambda q : \neg \text{ppred}(q)), n) = n - \text{count}(\text{ppred}, n)), \\ i \leftarrow n\}, \\ \text{cc0}, \\ \text{cc_ind } \{n \leftarrow j@p1\}$$

instance: Module is noetherian[nk_type, nk_less]

nk_measure: function[nk_type \rightarrow nat] == $(\lambda nk1 : nk1.n + nk1.k)$

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 ∨ 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 (λp : ppred1(p) \wedge ppred2(p)), $i \leftarrow 0$ }

ph_case0k: Lemma count((λp : ppred1(p) \wedge ppred2(p)), n) ≥ 0

ph_case0k_pr: Prove ph_case0k from

nat_invariant {nat_var \leftarrow count((λp : 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((λp : 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((λp : ppred1(p) \wedge ppred2(p)), pred(n)) $\geq k$)
 \supset (count(ppred1, n) + count(ppred2, n) $\geq n + k$)
 \supset count((λp : 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
 { $p \leftarrow$ (λnk : nk_ph_pred(ppred1, ppred2, nk)),
 $d_2 \leftarrow$ nk2@p3,
 $d \leftarrow$ nk@c},
 nk_ph_noeth_hyp {nk1 \leftarrow d1@p1},
 nk_noeth_pred {nk1 \leftarrow d1@p1}

pigeon_hole_pr: Prove pigeon_hole from

nk_ph_lem {nk ← nk with [(n) := n@c, (k) := k@c]},
nk_ph_pred {nk ← nk@p1}

exists_less: function[function[nat → bool], nat → bool] =
(λ ppred, n : (∃ p : p < n ∧ ppred(p)))

count_exists_base: Lemma count(ppred, 0) > 0 ⊃ exists_less(ppred, 0)

count_exists_base_pr: Prove count_exists_base from

count {i ← 0}, exists_less {n ← 0}

count_exists_ind: Lemma

(count(ppred, n) > 0 ⊃ exists_less(ppred, n))
⊃ (count(ppred, n + 1) > 0 ⊃ exists_less(ppred, n + 1))

count_exists_ind_pr: Prove count_exists_ind from

count {i ← n + 1},
exists_less,
exists_less {n ← n + 1, p ← (if ppred(n) then n else p@p2 end if)}

count_exists_pr: Prove count_exists {p ← p@p4} from

induction
{prop ← (λ n : count(ppred, n) > 0 ⊃ exists_less(ppred, n)),
i ← n@c},
count_exists_base,
count_exists_ind {n ← j@p1},
exists_less {n ← i@p1}

count_base: Sublemma count(ppred, 0) = 0

count_base_pr: Prove count_base from count {i ← 0}

count_true_ind: Sublemma

(count((λ p : true), n) = n) ⊃ count((λ p : true), n + 1) = n + 1

count_true_ind_pr: Prove count_true_ind from

count {ppred ← (λ p : true), i ← n + 1}

count_true_pr: Prove count_true from

induction {prop ← (λ n : count((λ p : true), n) = n), i ← n@c},
count_base {ppred ← (λ p : true)},
count_true_ind {n ← j@p1}

End countmod

Bibliography

- [1] Di Vito, Ben L.; Butler, Ricky W.; and Caldwell, James L.: *Formal Design and Verification of a Reliable Computing Platform For Real-Time Control: Phase 1 Results*. NASA, Technical Memorandum 102716, Langley Research Center, Hampton, VA, Oct. 1990.
- [2] Butler, Ricky W.; and Di Vito, Ben L.: *Formal Design and Verification of a Reliable Computing Platform For Real-Time Control: Phase 2 Results*. NASA, Technical Memorandum 104196, Langley Research Center, Hampton, VA, Jan. 1992.
- [3] Rushby, John: *Formal Specification and Verification of a Fault-Masking and Transient-Recovery Model for Digital Flight-Control Systems*. NASA, Contractor Report 4384, July 1991. Author's affiliation: SRI International, Computer Science Laboratory, Menlo Park, CA.
- [4] Federal Aviation Administration: *System Design and Analysis*. U.S. Department of Transportation, Advisory Circular AC 25.1309-1A, June 1988.
- [5] U.S. Department of Defense. *Reliability Prediction of Electronic Equipment*, Jan. 1982. MIL-HDBK-217D.
- [6] Schneider, Fred B.: *Understanding Protocols for Byzantine Clock Synchronization*. Department of Computer Science, Cornell University, Technical Report 87-859, Ithaca, NY, Aug. 1987.
- [7] Shankar, Natarajan: *Mechanical Verification of a Schematic Byzantine Clock Synchronization Algorithm*. NASA, Contractor Report 4386, July 1991. Author's affiliation: SRI International, Computer Science Laboratory, Menlo Park, CA.
- [8] Rushby, John; von Henke, Friedrich; and Owre, Sam: *An Introduction to Formal Specification and Verification Using EHDm*. Computer Science Laboratory, SRI International, Technical Report SRI-CSL-91-2, Menlo Park, CA, Feb. 1991.
- [9] Lamport, Leslie; and Melliar-Smith, P.M.: Synchronizing Clocks in the Presence of Faults. *Journal of the ACM*, vol. 21, Jan. 1985, pp. 52–78.
- [10] Rushby, John; and von Henke, Friedrich: *Formal Verification of a Fault Tolerant Clock Synchronization Algorithm*. NASA, Contractor Report 4239, June 1989. Authors' affiliation: SRI International, Computer Science Laboratory, Menlo Park, CA.

- [11] Welch, J. Lundelius; and Lynch, N.: A New Fault-Tolerant Algorithm for Clock Synchronization. *Information and Computation*, vol. 77, no. 1, Apr. 1988, pp. 1–36.
- [12] Srikanth, T.K.; and Toueg, S.: Optimal Clock Synchronization. *Journal of the ACM*, vol. 34, no. 3, July 1987, pp. 626–645.
- [13] Halpern, J.; Simons, B.; Strong, R.; and Dolev, D.: Fault-Tolerant Clock Synchronization. In *Proceedings of the 3rd ACM Symposium on Principles of Distributed Computing*. ACM, Aug. 1984, pp. 89–102.
- [14] Kieckhafer, R.M.; Walter, C.J.; Finn, A.M.; and Thambidurai, P.: The MAFT Architecture for Distributed Fault Tolerance. *IEEE Transactions on Computers*, vol. 37, no. 4, Apr. 1988, pp. 398–405.
- [15] Miner, Paul S.: *A Verified Design of a Fault-Tolerant Clock Synchronization Circuit: Preliminary Investigations*. NASA, Technical Memorandum 107568, Langley Research Center, Hampton, VA, Mar. 1992.
- [16] Dolev, Danny; Halpern, Joseph Y.; and Strong, H. Raymond: On the Possibility and Impossibility of Achieving Clock Synchronization. *Journal of Computer and System Sciences*, vol. 32, 1986, pp. 230–250.
- [17] Miner, Paul S.; Padilla, Peter A.; and Torres, Wilfredo: A Provably Correct Design of a Fault-Tolerant Clock Synchronization Circuit. To appear in the 11th Digital Avionics Systems Conference, Seattle, WA., Oct. 1992.
- [18] Moore, J Strother: *A Formal Model of Asynchronous Communication and Its Use in Mechanically Verifying a Biphase Mark Protocol*. NASA, Contractor Report 4433, June 1992. Author's affiliation: Computational Logic, Inc., Austin, TX.
- [19] Srivas, Mandayam; and Bickford, Mark: *Verification of the FtCayuga Fault-Tolerant Microprocessor System: Volume 1: A Case Study in Theorem Prover-Based Verification*. NASA, Contractor Report 4381, July 1991. Authors' affiliation: ORA Corporation, Ithaca, NY.
- [20] Bevier, William R.; and Young, William D.: *Machine Checked Proofs of the Design and Implementation of a Fault-Tolerant Circuit*. NASA, Contractor Report 182099, Nov. 1990. Authors' affiliation: Computational Logic, Inc., Austin, TX.

Vita

Paul Stevens Miner

Born in Chapel Hill, North Carolina, August 25, 1962. Graduated from Maury High School, Norfolk, Virginia, June 1980, B.S. in Computer Science, Old Dominion University, 1986. Employed, since 1989, as a Research Engineer at NASA Langley Research Center, Hampton, Virginia. In August 1992, will enter Ph.D. program in Computer Science at Indiana University.

In January 1990, the author entered the College of William and Mary as a part-time graduate student in the Department of Computer Science.