TIME ACCOUNTABILITY FOR LATTICE COMPUTERS

Mario R. Sanchez[1]
High Performance Database Research Center
School of Computer Science
Florida International University
Miami, Florida
msanch03@fiu.edu

Anil M. Shende
Department of Mathematics, Computer Science & Physics
Roanoke College
Salem, Virginia
shende@abdabs.roanoke.edu

ABSTRACT

Computers that emulate reality must be bound by the same often inexplicable contentious concepts that we take for granted while producing meaningful results. We present three important aspects of time that are required by such emulating computers, the lattice computer, in order to accurately and effectively simulate motion of objects in real space. We first provide a mechanical overview of the lattice computer so as to afford an understanding of our methods and the associated time related issues. We apply a universal clock theory to lattice computers so as to synchronize message passing between processors. A linearly proportional correlation between the time element of motion in the lattice computer and that of an object in real space is described. Lastly, we define the means by which the lattice computer can account for the time continuum in order to accurately emulate factors inherent to motion in real space.

1 LATTICE COMPUTERS

Lattice computers and associated algorithms present a vivid analogical representation for solving problems in the spatial and kinematic domain. The lattice computer [1][2][3] is approximately described as representing *space by space and time by time* via a mesh of interconnected computers. Each computer represents a point in space and the connections serve the means to simulate an object's motion and existence as described below.

Consider the problem of simulating the uniform motion of a 2-dimensional object, **O**, in a finite convex region, **R**, of a copy of Euclidean 2d-space (we chose a 2 dimensional object and space for simplicity of illustration.) Assume that **O** is the only object in **R**. Choosing a finite set of points in **R**, we assign identical synchronized computers to each point in this set, connecting some of these processors with bi-directional communication channels. These "black" processors (see Figure 1) will

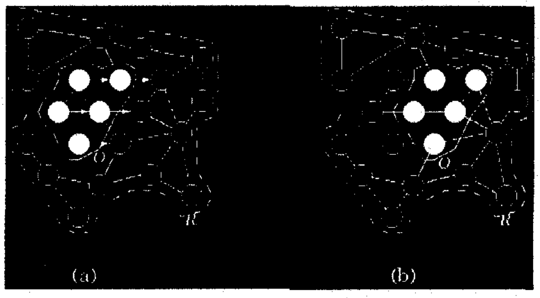

Figure 1: Analogical Representation of the Motion of **O** in **R**.

[1]This research was supported in part by NASA (under grant NAGW-4080), ARO (under BMDO grant DAAH04-0024), NATO (under grant HTECH.LG-931449), NSF (under grant CDA-9313624 for CATE Lab), and State of Florida.

emulate empty space while "white" processors will emulate particles of **O**. Motion is then imitated by the transmission of messages from the processor that represents the particle to an adjacent processor requesting that it start representing this particle. This method presents the first two issues that we will resolve in the following sections: clock synchronization and the distance/time correlation between the processors to the real distance/time traveled by **O**.

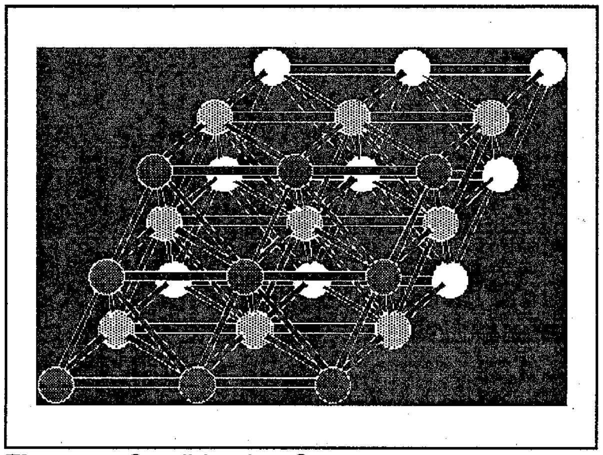
Figure 2: Small Lattice Computer

We extend this model to n-dimensional space by associating any bounded region of a copy of euclidean n-space and an n-dimensional regular lattice embedded in that copy of n-space with a meshwork computer having identical, synchronized processors on each lattice point in the region. The mesh is configured by connections being made between adjacent, i.e., adjacent processors. We define such a mesh computer as a lattice computer [4]. Figure 2 is representative of a 3-dimensional lattice computer. To enable the lattice computer to emulate the space continuum, the processors represent a region of n-space around themselves. Consider **X** as a point in an n-dimensional lattice. Further consider the set of points in n-space that are at least as close to **X** as to any other point in the lattice. Then this area (set of points) forms a convex, n-dimensional body around **X** called the Voronoi cell around **X** [5][6]. Each processor is responsible for the area (points) in its Voronoi cell. Figure 3 shows a lattice computer based on a 2-dimensional lattice and a curve **C** in 2-space traced by a moving particle starting at the point/processor **S**. Our algorithm for "moving" the object ensures that the point/processor **S** is initially active at simulation time **0**, and, afterward, that each point/processor **P** is active exactly in a duration of simulation time uniformly approximately linear in the real-time duration that the particle spends inside, or on the boundary of, the Voronoi cell around **P** [3]. While we resolve the motion accountability issue, we introduce the third area of concern where more than one processor is accounting for the same instances in time.

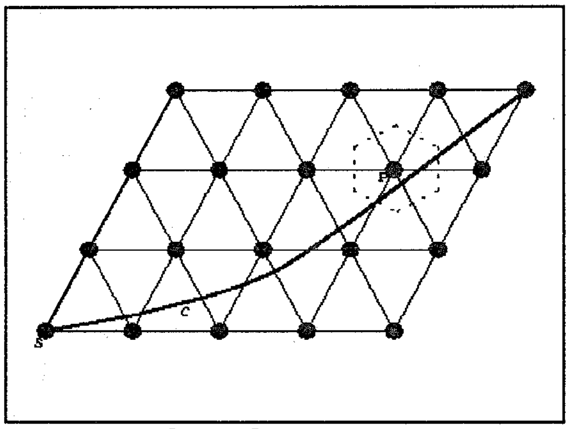

Figure 3: A Curve **C** and a Voronoi cell in a Lattice Computer

2 DOMAIN DELIMITED UNIVERSAL CLOCK

Since time is that which is measured by a clock, and clocks, over time, vary with respect to each other, then, at any particular instance, we are not able to determine what is the ***correct*** time [7]. This seeming paradox is consistently applicable to the processors of lattice computers and necessitates a means of synchronizing their clocks. Once synchronized, and consistently calibrated, the domain restricted to the lattice computer can be said to have a universal clock.

2.1 SYNCHRONIZATION

Each processor of the lattice computer possesses a clock - a logical or physical device by which to measure and account for time. The clock, initialized with a specific value, works by incrementing its value with a constant at specific intervals. The clock can be a function of a crystal and/or the system's machine's code in conjunction with some logical statements. The value inherent to the clock, at any given moment, is the time for those elements unique to that site.

To achieve the notion of real time, and to afford our solution for synchronizing messages between processors, all clocks of all processors must be the sole basis by which to calculate the same value at the same moment; i.e., directly or indirectly afford the same "time." To accomplish this task, we apply the synchronization process as first developed and proposed in [8].

The basic synchronization process derives a factor, the synchronization factor (SF), that is used to externally alter the measure accounted by a processor's clock, i.e, in humanistic terms, its time. The algorithm is thus:

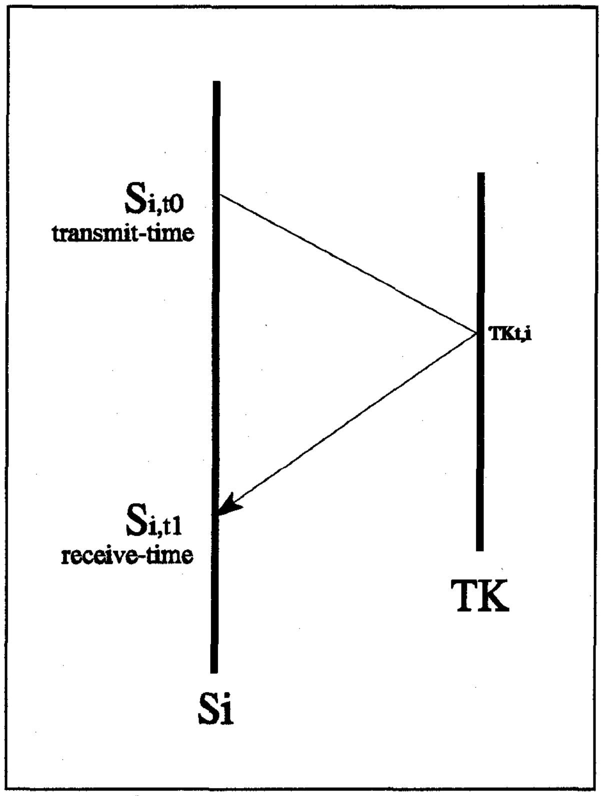

Figure 4: Synchronization Factor schema.

1. A timekeeper is designated. The timekeeper (TK) can be any of the processors.

2. Prior to any emulation, with no traffic over the communication channels, one processor, S_i, sends a message to TK at its time of $S_{i,t0}$.

3. TK immediately sends a message back that includes its clock value ($TK_{t,i}$). Although we claim immediately, time was incurred for the TK to respond. This delay will be the same for all processors hence it is disregarded.

4. S_i receives the message from TK at its time of $S_{i,t1}$ (Figure 4). The message delay (md) is computed by

$$md_i = (S_{i,t1} - S_{i,t0})$$

where

$$(\forall i)\ (S_{i,t1} \geq S_{i,t0})$$

5. The Synchronization Factor for processor i (SF_i) is computed by

$$SF_i = TK_{t,i} - (S_{i,t1} - \tfrac{1}{2}md_i)$$

6. This process is repeated for each processor.

The SF then becomes the value used by each processor to adjust its clock measure. A processor can adjust its clock measure at the moment the SF is computed, or can do so dynamically by adding the SF to its time at the moment it needs to send out its time. The former method is naturally more efficient. Hence, when a processor is required to timestamp an event, the processors's clock value is increased/reduced by the SF and this new value is what is used for the timestamp.

Synchronization at system wide start-up (perhaps the best time to designate for the synchronization process) requires 2N messages and idle time (no communication traffic) on the order of the number of processors; O(N).

Once all processors have synchronized their clocks with this method, the concept of a domain delimited (the lattice computer) universal, real and accurate time can be applied and all decisions based on "time" can be made accurately and reliably. Notice that we are not arguing that the TK's time is the correct time, if such a notion were to exist, but that it is the measurement by which all other clocks use as a basis and hence events can be uniquely ordered by the assigned values of time [9].

2.2 CALIBRATION

No sooner than clocks are calibrated that they go out of sync. Despite being affected by the same class of physical processes and hence behaving according to the same time scale [6], the fact that the processors are distinct guarantees heterogeneity in their measure of time. For a clock to provide a dependable measure, it must run at a correct rate irrespective of its deficiencies and external influences. The inherent deficiency of a standard computer crystal clock is computed as a frequency offset of $10^{-9}/600$ seconds [10]. External influences provide for a far greater and incalculable effect. For these reasons, at the very least, we must provide for a calibration process so all such effects can be ameliorated.

As each processor's clock produces a measure at a rate that is inconsistent with that of another processor's clock, the clocks begin to drift from the domain delimited absolute measure from where they started. The drifting process conceptually starts immediately, but may not be quantificationally significant after an indeterminate amount of processing.

Our calibration method is a means by which to logically eliminate the effects of the drift. With each site performing the synchronization process described above, we have implicitly eliminated the drift. Calibration can be an extension of synchronization once the initial synchronization efforts have been fulfilled (See Table 1). By assuming that all clocks will drift with respect to one another, we can institute a round of obtaining the SF at predetermined intervals or continually. The same methods described earlier for obtaining the SF can be employed. In essence, for calibration, **merely recalculate the SF.**

	< SYSTEM INITIALIZATION	><		AFTER X CYCLES	><		AFTER ROUND OF SF		>
	CLOCK VALUE	SF	D	CLOCK VALUE	SF	D	CLOCK VALUE	SF	D
P. A	13:14:58.63	+1.37	0	13:14:58.85	+1.37	+.22	13:14:58.85	+4.15	0
P. B	13:15:00.15	-0.15	0	13:15:00.15	-0.15	0	13:15:02.15	+0.85	0
P. C	13:14:58.99	+1.01	0	13:14:58.80	+1.01	-.19	13:15:02.80	+0.20	0
TK	13:15:00.00	0.00	0	13:15:00.02	0.00	+.02	13:15:03.00	0.00	0

TABLE 1. *At system initialization, the processors (P. x) obtain the synchronization factor as described. The drift (D) is then zero. After a period of X cycles*[2]*, each processor's clock has drifted from its initial value. Note that when the drift is considered, using the clock value and the SF would lead to incorrect conclusions since the measure is no longer domain delimited universally and accurately applicable (Processors A and C would incorrectly overlap in their ordering.) After a round of obtaining the SF, although the individual clocks have "drifted"; i.e., have not measured properly, the SF eliminates the drift - hence the implicit calibrator. This matrix is based on an empirical study employing the synchronization factor.*

3 ACCOUNTING FOR TIME

We noted that for computers to accurately and seamlessly represent reality, they must be bound by and account for the same forces that define our perceptive reality. For example, we can see and measure an object traversing our 3-dimensional world not yet having settled the contention of whether the traversal is accomplished discretely or analogically. Or more applicable, we can not even decide which clock to use - the observer's or the objects - since it has been proven that both generate disparate measures for the same set of instances. Then, as we emulate the object traversing space, quantifying its motion, duration and speed, we must somehow address and apply the uncertainties of the realities we are attempting to mimic with ever complex algorithms. Without doing so the model can not be deemed to be reflective of what we call reality.

3.1 CORRELATION

In section 1 we noted the need to correlate the distance/time between the processors to the real distance/time traveled by **O**. More than just correlation, the interconnectivity of the processors requires that the time delay be at least linearly proportional, and at best emulatively equal, to the length in real space that the connectivity is emulating. We accomplish this requirement as follows.

Processor P_1 computes the last time unit that it is representing the particle. During that time unit, P_1 sends a message to P_2 and relinquishes control of the object. At the beginning of the next time unit, P_2 gets the message from P_1 and obtains control of the object. In principle, processor P_1 has control over the object in its Voronoi cell and sends a message to the adjacent processor (P_2) that will subsequently control the object. The message will dictate to P_2 at what time it is to account (control) the object in addition to all of the information defining the object's trajectory. When that time *arrives*, P1 will relinquish control of the object concurrently with P_2 gaining control of the same object. P_2 then continues computing the object's trajectory and sends the updated controlling message to the adjacent ensuing processor. This cycle is repeated until the motion of the real object in real space stops.

Clearly there is a maximum speed by which two

[2]Where *cycles* is a period of time as applicable to the set number of cycles of the lattice computer.

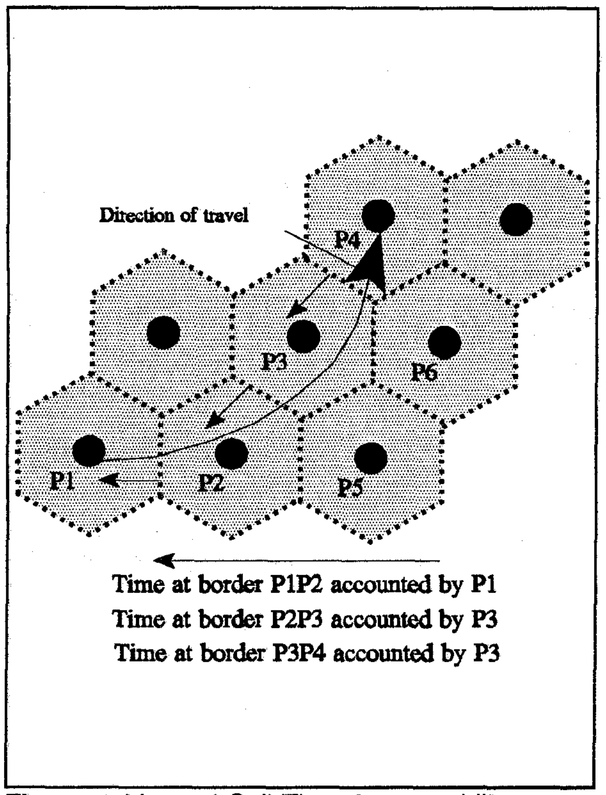
Direction of travel
P4
P3
P6
P1
P2
P5
Time at border P1P2 accounted by P1
Time at border P2P3 accounted by P3
Time at border P3P4 accounted by P3

Figure 5: Voronoi Cell Time Accountability

processors can communicate with each other. Since message passing within the lattice computer is done at this speed, the controlling message will alert the adjacent ensuing processor (say P_3) prior to the object actually entering P_3's Voronoi cell.

3.2 EXTRA TIME

As we noted in section 1, Voronoi cells share boundaries with adjacent cells. A processor whose domain is a Voronoi cell is said to be controlling if the object is completely within its Voronoi cell or **at the exiting border** (see Figure 5). In the case where an object straddles a border in the direction of travel, the conically ordered processor relative to the originating processor sharing the straddled border would be controlling. As an example, consider an object traveling from P_2 towards P_6 straddling the P_3P_5 border. P_3 would be in control. If, while along the same border, the object changes direction back to P_2 (and hence seemingly having originated from P_6), then P_5 would be in control.

Hence, with these means of accountability, and with assurances of correlation noted earlier, we ensure that the time dimension of any function of the object emulated by the lattice computer is equivalent to its real world counterpart.

Although one could suggest that "time's" accountability is negated by the use of mathematical formulas plotting the trajectory of an object, the presence of the lattice computer is to represent the analog motion of said object through real space. To represent analog motion implies traversing or emulating a continuum of both space and time. Herein we address solely the time continuum although both are resolved through the implicit mechanics of the lattice computer.

We can effectively shirk the continuum versus discrete time disputation by merely ensuring that the lattice computer correctly accounts for the time dimension throughout the duration of the object's motion through real space. Namely, if the object's motion starts at t_0 and ends at t_n, then for the lattice computer to be exactable in its measure of time, it should report the duration of travel as $t_n - t_0$.

4 FURTHER WORK AND CONCLUSION

A problem arises when individual processors have to affect the motion of the object at ever finer time intervals, thereby implicitly converting time into a discrete measure. Assuming that there is such a requirement for infinitely finer discrete measures of time, or that the theoretical implications of a time continuum are essential to any emulative process, we propose a theoretical extension to lattice computers where **an infinite number of discrete measures, absolved non-deterministically, is a continuum**. Such an extension to lattice computers can be employed to not only report time correspondingly to how same is done in reality, but can in fact be used to model time itself. The ramification is that such lattice computers may be able to simulate a continuum - including that of time.

Precise time accountability in the lattice computer is essential for the model to effectively analogically represent the motion of an object through space. In this paper we have addressed three basic concerns in this regard: more than one processor accounting for the same instance of time. For these cases we presented practical and direct solutions. These resolves continue to ensure the usefulness and feasibility of the lattice computer. A further extension to our research is the implication of the solution to the multiple accountability for the same instance in time as noted in the third area of concern presented herein.

BIBLIOGRAPHY

1. Case, J., Rajan, D.S., and Shende, A.M. *Optimally Representing Euclidean Space Discretely for Analogically Simulating Physical Phenomena*. Foundations of Software Technology & Theoretical Computer Science. vol. 472, 1991.

2. Case, J., Rajan, D.S., and Shende, A.M. *Lattice Computers for Optimally Representing Euclidean Space*. Technical Report 91-15, University of Delaware. 1991.

3. Case, J., Rajan, D.S., and Shende, A.M. *Simulating Particle Travel with Lattice Computers*. Proceedings of the ISCA on Parallel and Distributed Computing Systems. 1995.

4. Case, J., Rajan, D.S., and Shende, A.M. *Representing the Spatial/Kinematic Domain and Lattice Computers*. Journal of Experimental and Theoretical Artificial Intelligence. vol. 6, 1994.

5. Conway, J.H. and Sloane, N.J.A. *Sphere Packings, Lattices and Groups*. 2nd Ed., Springer Verlag.

6. Aurenhammer, F. *Voronoi Diagrams - A Survey of a Fundamental Geometric Data Structure*. Technical Report B-90-9, Institute of Computer Science, Dept. of Mathematics, Freie Universität Berlin.

7. Milne, Edward. *Kinematic Relativity; A Sequel to Relativity, Gravitation and World Structure*. Oxford, Claredon Press. 1948.

8. Sanchez, M.R., Orji, C.U. and Kingsley N.C. *Veritable Time Representation for Clock Synchronization in Distributed Computer Systems*. Proceedings of the ISCA on Parallel and Distributed Computing Systems. 1995.

9. Isham, Christopher. *God, Time and the Creation of the Universe*. Explorations in Science and Theology. RSA London, 1958.

10. Ellingson, C., and Kulpinski, R. *Dissemination of System Time*. IEEE Transactions on Communications, May 1973.