TIME MECHANICS AS APPLIED TO EVENT ORDERING[1]

Mario R. Sanchez, Cyril U. Orji, Naphtali D. Rishe and Kingsley C. Nwosu[2]
High-Performance Database Research Center
School of Computer Science
Florida International University
University Park
Miami, Florida 33199
{*msanch03, orji, rishen*}@fiu.edu

ABSTRACT

The notion of time is interpreted and quantified as a means to ensure consistent and accurate evaluations by time-relevant functions. As an integral part of a study encompassing all functions of automata, we formulate the process as it applies to event ordering. In developing this process, logical clocks are first examined to aid in defining the inter-relation between explicitly and implicitly related events. Understanding the extent to which logical clocks can be representative of system wide ordering, an ubiquitous measure is considered. We define "real time" as a means by which all measurement devices account for the same measure at the same moment, and whereby they remain consistent with respect to each other for the life of the system. "Real time" as an absolute and universally applicable system measure is analyzed and an algorithm for its implementation is proposed. With each event assigned a time value that is respected system wide, the application of mutual exclusion algorithms are presented and their behavior assessed when ordering certainty is ensured. Further, a means for maintaining consistent and accurate system wide time is developed.

1. INTRODUCTION

Mutual Exclusion (ME) algorithms are concerned with granting critical section access to only one process at a time. The algorithms must do so efficiently, fairly, without creating deadlock or causing starvation. To be efficient and fair, ME algorithms in a distributed system need to know which process requested the critical section first.

In determining the order, ME algorithms have relied on logical clocks for partially ordering events, or an extension of logical clocks by introducing a "click" condition, or on a timestamp. In the case of logical clocks for partially or totally ordering events, with or without the click condition, doubts arise when the implied assumptions are questioned. As an example, what can be said of two distributed events that requested access to a critical section when the ordering messages have yet to take place? In the case of the timestamp, the ME algorithms rely heavily on these assignments of measure, despite that their makeup and determinant factors are unknown. In essence, there is no certainty that the *time* that constitutes the timestamp is accurate and universally coherent; that is, exactly applicable to all events from all processes at all sites across the entire network.

In essence, what we are concerned with is the precise and meaningful application of time when such is needed as a means to understand the relationship between events. Through this effort, we note that the duration of inter-event/inter-process messaging can be discounted without casting a doubt over the integrity of the ME algorithms.

2. LOGICAL CLOCKS

Logical clocks, by definition [1], order events of variant processes. However, logical ordering may not be decidable in all perspective meanings of causality. There is evidence that when a relation is defined asymmetrically, the cause may create the effect, (as in a process sending a message to another process causing an event to be created for the purpose of answering that message), but the implication from such causality may not be what can be inferred [5]. Therefore, the Space-Time diagrams of [1] are nothing more than a linear depictions of events that have no relation to "real time." Although our assignment of a conceptual time affords us the means by which to examine the transitive states of the processes through the promulgation of events within, time is not inherent in the events, we can not infer any ordering from their mere existence and hence require a means to assign an empirical position in our frame of reference. For after all, if we take away our sense of sensibility, time vanishes, and then we will not be able to order anything [2]. And this is in fact the case in a system of logical clocks, for if no events were to occur in the system, or their occurrence are of a frequency beyond applicable usefulness, any numeric or otherwise symbolic assignment would not be created, or created

[1]This research was supported in part by NASA (under grant NAGW-4080), ARO (under BMO grant DAAH04-0024) and NSF (under grants IRI-9409661 and CDA-9313624 for CATE Lab). [2]Kingsley C. Nwosu is with AT&T Bell Labs.

beyond usefulness [3].

Clock Clicks

For the "space-time" diagrams to be reflective of the order by which events occurred, the "click" (dashed horizontal) lines were introduced in other Space-Time diagrams of various processes. However, these lines must be in sync - one click spans all processes in the same deterministic manner. And if this was the case, we would not need to consider the "logical" ordering of events, but merely consider the occurrence of the events relative to the click lines.

In applying these clicks, we have, in essence, quantified absolute time for disparate processes without using a clock, i.e., without calling a clock a clock. In an isolated physical system (the one that binds the clicks from independent processors into an albeit imaginary horizontal line) when the same succession of events or states are repeated, those series of events, in such a system, is called a clock [4]. This being the case, we can extend this notion of clicks as a manifestation of a clock to order ***all*** events by deduction as opposed to order ***some*** events by implication. If "clicks" are universally relative to all events, albeit one click is the same for all events or one click of one process can be derived from another click of another process, then we can determine the order of ***all*** events quite easily. Hence, if clicks are introduced in a system of ordering where there are no devices to measure that what we perceive to be the passage of time (i.e., real clocks), then the "clicks" are either inconsequential, or when depicted to distinguish the relative occurrences of variant processes, they become an *implicit* representation of time.

But again, this is based on the notion that the clicks are the sequences that make up a synchronized clock applicable for all events. However, this "clock" is an object that *explicitly* does not exist. The "clicks" of one process are not atomically the same as the "clicks" of another process. Nor have we established that any specific "click" is congruent to, or derivable from, any "click" of other processes. Therefore, a closed system devoid of an explicitly stated universal clock establishes that nothing can be said of the events that are not deterministically related. Antecedence is relative in connection with pairs of casually unconnected events (events of different processes), and absolute with regard to events that are casually connected and genetically linked to one another (events of the same process) [5].

The ordering for those events not deterministically related remains indeterminate. No matter how all events of concurrent processes are depicted, when a unifying element in their definition is not universally applicable, the ordering of a certain number of events is indescribable. What we require is the absolute criteria of time measurement independent of the casual or deterministic postulate [6].

3. REAL TIME

Each event occurs in a process and each process is performed in a site. Each site possesses a clock - a logical or physical device by which to measure and account for time. The clock is set with a specific value then works by augmenting that value with a constant at specific intervals. The value inherent to the clock, at any given moment, is the time for those elements unique to that site. To consecrate the notion of real time, and to afford our solution for properly ordering events, all clocks of all sites 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 introduce the synchronization process.

3.1 Ideal Conditions

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

1. A timekeeper is designated. The timekeeper (TK) can be the server/host or a site.

2. At system/network initialization, with no traffic over the communication channels (we will relax this restriction momentarily), one site, S_i, sends a message to TK at its time of S_{it0}.

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

4. S_i receives the message from TK at its time of S_{it1}. The message delay (md) is computed by

$$md = (S_{it1} - S_{it0})$$

5. The Synchronization Factor (SF) is computed by

$$SF = TK_t - (S_{it1} - \tfrac{1}{2}md)$$

6. This process is repeated for each site.

And, $\forall_{t1t0} : t1,t0 > 0 :: (S_{it1} \geq S_{it0})$

The SF then becomes the value used by each site to adjust its clock measure. A site 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 process is required to timestamp an event,

the site's clock value is increased/reduced by the SF and this new value is what is used for the timestamp.

Once all sites have synchronized their clocks with this method, the concept of universal, real and accurate time can be applied and all ordering and all decisions based on "time" can be made fairly and reliably. Notice that we are not arguing that 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.

3.2 Real World

For a network to be idle while the synchronization process ensues can become an expensive proposition. Additionally, there are networks that have no "start-up" for they are running constantly. But a site sending a synchronization message to a timekeeper over a busy network will yield a *md* that we can not use since we do not know how long the message took to get to the timekeeper. This scenario would render our algorithm inaccurate and unusable. Therefore, we propose three synchronization factor supplier methodologies that can serve as an extension to our algorithm thereby enabling clocks to become synchronized under real (i.e, traffic in the network) conditions.

A node signing on to an already active network (i.e., network with traffic) can obtain the SF from any node. The new site can use the SF as its own, adjusting the SF by the time required for a message to travel or not travel the distance differential between itself and the supplying node

If the timekeeper is a server or host, then it knows when there is no network traffic in its realm of the distributed system. If the timekeeper is not a server or host, then it can be so informed by the server or host on an optimally determined periodic basis. Then, the timekeeper, knowing when the network is idle, can institute a round of the synchronization process. The TK can send a message to one site directing it to send the synchronization message. The site responds by sending the message and the TK receives the message and sends it back immediately as previously specified. If after a determinate period of time there is still no network traffic, the TK sends a message to the site asserting that the recent synchronization process is valid. If during the determinate period of time network traffic appeared, the TK sends a message to the site that the recent synchronization process is invalid since the md may include uncountable delays from the unexpected traffic. In this latter case, the TK can start with that site the next time a synchronization process is instituted. The dynamically instituted synchronization process can be extended to include those sites that borrowed the SF (as in 1 and 2 above).

Applying the rationale of one node/site supplying the SF to other node/sites as a topological criteria, then our original premise that **all** sites perform the basic synchronization process need not be a stringent requirement. In fact, using any combination of the three procedures suggested, only one node/site needs to perform the basic synchronization process and all other nodes/sites merely inherit the SF. The synchronization factor supplier extension then becomes a chain of perpetually lending/supplying the SF throughout the network. In this approach, if only one site be needed to supply the SF to all other sites, then the synchronization process at system start-up time requires 2 messages and idle time of O(1).

3.3 Continuation

We have applied the notion of real time to accurately account for the ordering of events and implicitly demonstrated that "real time" does not exist. Kant assigns the notion as a necessary representation of on which all ***intuition*** depends [2]. And this quantitation of perception applies to the application of our algorithm. Notice that the "time" of the timekeeper need not be real nor in accordance with an absolute measure of time - if one were to exist. Time is merely a measure by which ***all*** processes adhere to in a universally determinate means. The consideration then remains that this measure, as used by all sites, must be consistent. If clocks, the creators of the measure, were to execute their computations too fast or too slow with respect to ***each other*** by a factor that will make the measure grossly inconsistent over the perennial life of the system, then the SF would be rendered useless. To resolve this problem, we must then consider calibration.

4. CALIBRATION

Crystal based clocks that are the mainstay of powerful desktop computers are not always consistent in the measurement of the cycles that represent the passage of time. Add on the effect of interrupts and a heavy network o/s, and time representation will not consistently represent the same measure. In essence, different machines will account for time differently. Although improvements have been made on the ill effects of heavy network o/s, BIOS functions and interrupts (i.e, external influences) [7], there remains factors that influence how each machine accounts for the measure which has been designated as 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 deficiencies of standard computer crystal clocks is computed as a frequency offset of $10^{-9}/600$ seconds [8]. 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 both effects can be ameliorated.

As the clocks of each site produce their measure at a rate

that is inconsistent with each other, they begin to drift from the 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. By 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 frequently when the network (local and distributed network access) is idle, or with low traffic. The same methods described earlier for obtaining the SF can be employed. In essence, for calibration, merely perform on a routine basis a blend of the three noted procedures.

The results from our empirical studies in obtaining the SF and performing the calibration are summarized in Table 1.

At system initialization, the sites obtain the synchronization factor as described. The drift (D) is then zero. After a period of exactly 6 days, each site's clock drifted from its initial value. Note then that when the drift is considered, using the clock value and SF would lead to incorrect conclusions since the measure is no longer universally accurately applicable (Sites A and C would incorrectly overlap in their ordering.) After a round of obtaining the SF, although the individual clocks have "drifted" - have not measured properly, the SF eliminates the drift - hence the implicit calibrator.

5. CONCLUSION

This study has shown that the notion of real time can be introduced into a system for correctly and doubtlessly ordering events. The process is relatively straightforward to implement and maintain and derives an absolute ordering/chronology as demonstrated in our findings in Table 1. The benefit to ME algorithms and other constructs depending on event ordering is that they can then produce ensure fair and accurate decisions. Additionally, external factors that previously influenced such decisions, e.g., communication delays, have been voided.

	< SYSTEM INITIALIZATION			>< AFTER PRECISELY 6 DAYS			>< AFTER ROUND OF SF		>
	CLOCK VALUE	SF	D	CLOCK VALUE	SF	D	CLOCK VALUE	SF	D
SITE A	13:14:58.63	+1.37	0	13:14:58.85	+1.37	+.22	13:14:58.85	+4.15	0
SITE B	13:15:00.15	-0.15	0	13:15:00.15	-0.15	0	13:15:02.15	+0.85	0
SITE 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.

6. REFERENCES

1. Lamport, Leslie. *Time, Clocks, and the Ordering of Events in a Distributed System.* Communications of the ACM, 1978 Vol. 21 No. 7.

2. Kant, Immanuel. *Critique of Pure Reason.* (1781). Doubleday 1966.

3. Mattern, F. *Virtual Time and Global States of Distributed Systems.* Parallel and Distributed Algorithms, Elsevier Science, 1989.

4. Weyl, Hermann. *Space-Time-Matter.* (1921). Dover, 1952.

5. Bunge, Mario. *Causality and Modern Science.* Dover, 1979.

6. Nevanlinna, Rolf. *Space Time and Relativity.* Addison-Wesley Publishing, 1968.

7. Sanchez, Mario. *Resolving an Errant Clock in Servicing a Novell Network.* RAM Systems, Inc. white paper, 1992.

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