JOURNAL OF AIRCRAFT
Vol. 32, No. 3, May–June 1995

Event Correlation for Networked Simulators

Amnon Katz*
University of Alabama, Tuscaloosa, Alabama 35487

The timing and synchronization issues of networked simulation over large distances are studied. It is shown that the absolute time-stamp is effective in removing inconsistencies between the world pictures presented by the networked simulators. The tolerance that can be maintained on correlation errors is dominated by the precision of the dead-reckoning process over a time span that is the sum of propagation delay and clock error. Dead-reckoning statistics are invoked to determine the level of correlation that can be achieved for global networking. The clock accuracy required for this purpose is assessed.

Nomenclature

t_n = frame time, nth frame, the epoch at which a frame display is complete
Δt_c = clock error
Δt_{DR} = dead-reckoning interval, an interval following an update that dead-reckoning can maintain prescribed error limits
Δt^+_{DR} = upward adjusted dead-reckoning interval, consisting of the smallest number of simulation frames that exceeds Δt_{DR}
Δt_f = frame interval
Δt_p = propagation delay
Δt_{ra} = reception advance, how long in advance of frame-time information must be received to be included in the frame display
Δt_s = synchronization delay, interval between beginning of blackout period and sender's previous frame time
Δt_{ta} = transmission advance, how long before frame-time information is transmitted (if not withheld)

I. Introduction

THE subject of this article is long-haul networked simulation. We address the accuracy with which remotely located simulators can represent close quarter interaction between the entities they simulate. Formation flying is a prime example for such an interaction.

The analysis links the resulting accuracy with the precision of the "dead-reckoning" process used to predict the future states of the players. Quantitative bounds on the accuracy possible with currently known methods of dead-reckoning are provided. The absolute time-stamp plays a key role in the analysis.

The concept of large-scale, remotely placed distributed interactive simulation (DIS) has been pursued by the U.S. military for several years. The purpose is to create a synthetic battlefield in which units operating simulators located at their home bases can participate and interact. The same concept is applicable to civilian scenarios, air traffic control being one. The many simulators communicate over an internet (realized by use of local and long-haul physical links). The simulators broadcast information packets, known as PDUs (protocol data units) in which information computed in one simulator is shared with all others.

Received May 9, 1994; revision received Sept. 1, 1994; accepted for publication Sept. 3, 1994.

*Professor, Department of Aerospace Engineering, P.O. Box 870280. Member AIAA.

In order to allow realistic interactions between all players, the worlds they observe must be closely correlated in space and time. The importance of timing was pointed out early.[1] The requirement for a time-stamp is included in the DIS data exchange standard from its first version.[2] Optionally, this is an absolute time-stamp, derived from synchronized clocks.

At the time of this writing, the absolute time-stamp has never been employed. This is partly due to the lack of a commonly available source for sufficiently accurate synchronized time. The situation is changing. The global positioning system (GPS) now offers this functionality. The University of Alabama has developed and demonstrated a GPS-based absolute clock for use in simulation.[3]

Quite apart from the lack of equipment, the function of the time-stamp is not generally understood within the DIS community. Correlation accuracy is largely a long-haul issue, that is critical only where precise close quarter interaction is required between entities that are simulated remotely. As such, it can be easily overlooked. This article highlights the role of the time-stamp in the consistent simulation of close interactions.

In Sec. II we lay the groundwork and point out the inconsistencies that arise from the use of raw positions reported by remote, or even local, simulators. Section III shows how an absolute time-stamp eliminates the systematic errors. The discrepancy between what different simulators display is reduced to a question of the accuracy of dead-reckoning. Equation (7) relates the allowed combined value of propagation delay and clock error to Δt_{DR}, the interval over which dead-reckoning[4] can maintain the prescribed precision.

In Sec. IV we draw on recent work on dead-reckoning[5–7] to establish a quantitative relationship between the precision required and Δt_{DR}. Section V spells out the precision limits within which close-quarter interactions can be simulated over a global network. A recommended specification for the accuracy of the absolute clock follows.

II. Event Correlation

In discussing correlations in space and time, we adopt the terminology that has long been prevalent in relativity physics:

1) *Space-time*, the four-dimensional space parametrized by three space coordinates and time, e.g., (x, y, z, t).

2) *Event*, a point in space-time. Such occurrences as detonations, launching of projectiles and missiles, or collisions are discrete events.

3) *World line*, a curve in space-time that allows a unique event with each time (epoch). Enduring objects trace world lines. The object (or rather its center or representative point) is at some unique space location at any given time.

The simulation exercise unfolds in terms of the world lines of the participating entities, and the events where such entities initiate action, or sustain damage. Other related information

about the entity, including its orientation, is also maintained along the world line.

Simulators break time into discrete simulation frames that typically last anywhere from 5 to 100 ms, depending on the level of fidelity of the device. We will accept the rate at which the information in visual displays is updated as the frame rate. We will refer to the times when a complete display is shown as frame times. The events at the intersection of frame times with world lines will be called frame events. The difference between consecutive frame times will be called frame interval and denoted Δt_f.

Simulators represent world lines by sequences of discrete frame events. At each frame time, the simulator's display reflects a set of frame events for the different players all corresponding to its own frame time. We will call this set of events a frame display.

Dynamic computations for the ownership and collection and processing of data regarding other players must be repeated for each frame. These procedures may be accomplished during the frame. More often it takes longer than a frame interval, and computations for successive frame times are staggered. It is typical that the display generator prepares the frame display for the nth frame at the same time as the host computer performs dynamic computations for the $(n + 1)$st frame, and peripheral devices collect data for the $(n + 2)$nd frame. In any event, PDUs to be used in preparing a frame display must be received in advance of the frame time. By the same token, ownship information must be available in advance of frame time and, therefore, can be transmitted in advance of frame time.

We introduce the following terminology: 1) reception advance Δt_{ra} and 2) transmission advance Δt_{ta} (see Nomenclature).

Consider several simulators that share the same sequence of frame times, are subject to the same values of Δt_{ra} and Δt_{ra}, and, further, satisfy

$$\Delta t_{ta} > \Delta t_{ra} + \Delta t_p \tag{1}$$

where Δt_p is the propagation delay over the net. In this case the positions transmitted can be picked up and used by the other simulators without adjustment and without attention to times and time-stamps.

The situation described above is atypical. Even when the simulators are synchronized, condition (1) is likely not to be satisfied. In general, however, the relationship between remotely located simulators is asynchronous; their frames agree neither in length nor in phase. There is a great deal to be said for this type of architecture. It allows simulators to be independent of one another. Common failure points are eliminated. The failure of any unit deprives the total system of no more than the representation of the entities computed there. Simulators of different type and complexity can play together.

In this general case, however, it is not permissible to use the positions transmitted by other simulators without adjustment. The error involved in the use of raw position, measured as a distance in the receiving simulator's display is given by

$$\Delta x = V\Delta t \tag{2}$$

where Δt is the discrepancy between the original event time and the frame time at which the position is displayed, and V is the speed of the entity in question.

The case of formation flying, wingtip-to-wingtip, may serve to illustrate the pitfall of using broadcast positions uncorrected. Suppose the two simulators are synchronized. They share the same sequence of frame times

$$t_n = t_0 + n\Delta t_f \tag{3}$$

and common values of Δt_{ra} and Δt_{ta}. Let us further assume that the simulators are close to each other so that the propagation delay Δt_p can be neglected. Still, let Eq. (1) be violated because the reception advance is slightly longer than the transmission advance:

$$\Delta t_{ra} = \Delta t_{ta} + \varepsilon \tag{4}$$

L, the simulator representing the lead aircraft, transmits the position X_n for its nth frame at time $t_n - \Delta t_{ta}$. This is too late for simulator W representing the wingman, to include in its nth frame display. It is included in W's $(n + 1)$st frame. The wingman then places himself alongside X_n and transmits this position. The earliest that L might be able to display this information is at its $(n + 2)$nd frame. The lead pilot observes his wingman aligned with where he was two frame intervals earlier (Fig. 1), at the same time that the wing pilot sees the wingtips perfectly aligned.

Table 1 Typical speeds of simulated entities and related position errors

Vehicle	Speed, km/hr	Distance per 16-ms frame, m
Ground vehicle	100	0.44
Helicopter	250	1.11
Transport	700	3.11
Mach 1	1225	5.44
Mach 3	3675	16.33

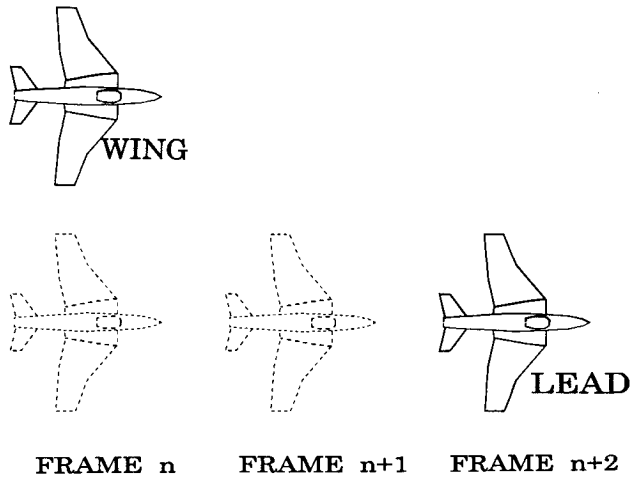

Fig. 1 Formation flying with raw positions.

Obviously, fast movers are more sensitive to this problem. Some examples are assembled in Table 1. The errors shown can be tolerated uncorrected for displays of ground vehicles and for even Mach 3 fighters interacting at a distance. For the Mach 3 fighters flying formation they are clearly unacceptable.

The effect is further aggravated by propagation delays. A propagation delay equal to one frame interval increases the observed discrepancy by two frame intervals. Similar effects apply to simulators that are not synchronized. In every case, propagation delay increases the discrepancy between the observations of the lead pilot and his wingman.

Data packages exchanged between the networked simulators travel at or close to the speed of light of $c = 2.998 \times 10^8$ m/s (about 1 ft/ns). This is much higher than any of the speeds listed in Table 1. Table 2 offers a sampling of time-distance correlation using the speed of light. It is seen that the influence of propagation delay on the position error is a long-haul effect. Over global distances, or using a geostationary satellite as a relay station, the effects are very significant. Reduction of the propagation rate in cables and additional delays in switching equipment aggravate the problem.

Table 2 Speed-of-light delays

Time interval	Distance	Note
1 ns	0.3 m	1 ft
1 μs	300 m	1,000 ft
→100 μs	30 km	←Target time-stamp accuracy[a]
1 ms	300 km	——
16 ms	480 km	Typical simulation frame
67 ms	20,000 km	Halfway around the globe
240 ms	70,000 km	To geostationary satellite and back

[a]Value recommended in Sec. V.

III. Absolute Time-Stamp

The problems described in the previous section arise from the use of raw information that is wrong for the frame time. They are eliminated by use of the absolute time-stamp. With the time-stamp included, the PDU conveys not a position, but rather an event. Given a sequence of events corresponding to the world line of entity B, simulator A must reconstruct the world line and derive the frame events it requires. This is done by a process of estimation known, in the DIS context, as dead-reckoning. A more detailed consideration of DIS dead-reckoning is deferred to Sec. IV. For a general survey of dead-reckoning algorithms, see Ref. 4. For recent advances, see Refs. 5–7.

Figure 2 shows the world lines of entities A and B simulated in devices designated similarly. Simulator B represents the world line of its ownship by the frame events shown as circles. This information is broadcast on the net and becomes available to simulator A, which in turn derives the frame events it needs, shown as diamonds. Assuming a capacity for perfect dead-reckoning, the positions of A and B in A's frame displays are in precise correlation.

All simulators on the net receive the same PDUs and in them the identical digital encoding of individual events, including those sequences of events that represent world lines. As a consequence, they all agree on the presentation of ground truth for all players. Neither differences in frame time, nor propagation delays of packets, nor even inaccuracy of the absolute clock, can create a conflict as to where every player is when.

On a closer look, however, even though all simulators receive the same PDUs, they do not all receive them in time for a particular frame display. Figure 3 addresses this point for simulator A in its task of preparing for the nth frame display at t_{An}. The frame events computed by both simulators, each for its respective entity, are shown as circles. A's frame events that are already computed and transmitted are shown solid. They run to t_{An}. The ones that are still not done are shown in outline. Similarly, those frame events that have been received from B in time for use in preparing A's nth frame display are shown solid. These are the ones received no later than $t_{An} - \Delta t_{Ara}$. Later frame events for B (that are not available for the nth frame) are only outlined. What A needs for its nth frame is B's frame event at its own frame time, which is shown as a solid diamond. A must estimate the diamond event based on B's solid circles.

Not only is there no circle on B's line at the diamond, several of the circles earlier than the diamond are not solid. They fall into a "blackout period" (Fig. 3) that is due to the cumulative effect of the following:

1) The excess of Δt_{Ara} over Δt_{Bta}. The transmissions of B may be advanced by less than the reception by A must be.

2) A discrepancy between the absolute clocks of A and B. If its clock is late by Δt_c, B believes that time $t_{An} - \Delta t_{Ara}$ has not arrived yet. In consequence information pertaining to B in the interval $(t_{An} - \Delta t_c - \Delta t_{Bta}, t_{An} - \Delta t_{Bta})$ will not have been transmitted.

3) Propagation delay. Frame events that have been broadcast by B in the interval $(t_{An} - \Delta t_{Ara} - \Delta t_p, t_{An} - \Delta t_{Ara})$ have not been received by A in time.

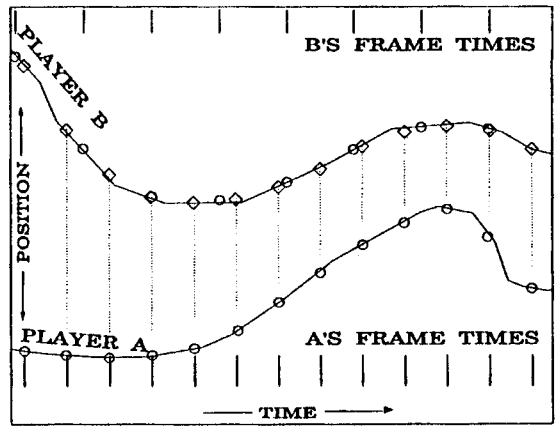

Fig. 2 A's estimation of frame events on B's world line.

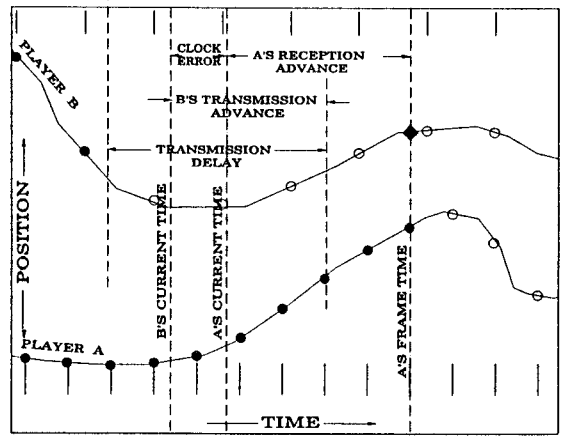

Fig. 3 Blackout period.

The blackout period consists of A's reception advance to which the clock error (B's clock relative delay) must be added, from which B's transmission advance is subtracted, and to which the propagation delay is added. Figure 3 illustrates the blackout period.

The interval δt between A's frame time and the frame time of B's most recent event that can be included in A's frame display includes in addition to the blackout period also the synchronization delay—the interval between the beginning of the blackout period and B's actual preceding frame time. δt is the interval over which A must "dead-reckon" B's state (Fig. 4). This interval is

$$\delta t = (\Delta t_{Ara} - \Delta t_{Bta}) + \Delta t_c + \Delta t_p + \Delta t_s \qquad (5)$$

The longer δt, the more shaky the estimate of the solid diamond becomes. Nevertheless, no systematic error is involved. In cases where future positions can be perfectly predicted (e.g., when B maintains uniform motion) the diamond event can be computed without error, regardless of how outdated the available data.

Note that we have set out to identify the worst case. If B's clock were early instead of late, it would have made A's task

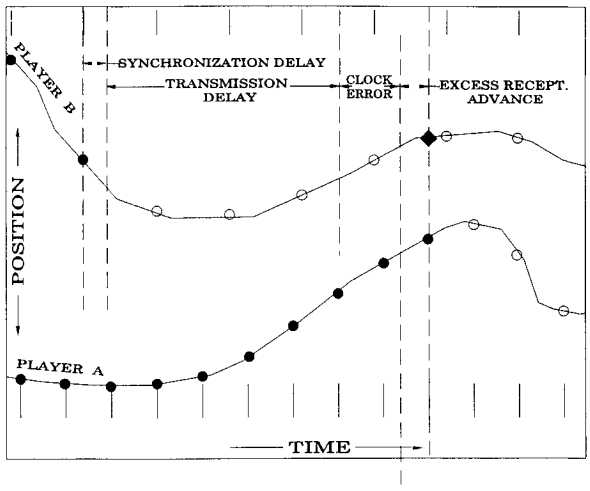

Fig. 4 Interval requiring dead-reckoning.

easier. But B's task would then be more difficult. Whenever the two clocks are not in sync, one simulator's task (and dead-reckoning accuracy) is hurt.

In an extreme case of a fast clock, B might have broadcast information for later than t_{An}. This causes no problem for A. The information about the future is not revealed to the pilot. It merely turns the simulator's task of estimating B's position at t_{An} easier, being an interpolation rather than a prediction.

Similarly A's reception advance might have been less than B's transmission advance. This would have made things easier for A, but, in all likelihood, then A's transmission advance would be shorter too, making it more difficult for B. In general, a higher fidelity simulator running shorter frame intervals will require less reception advance and offer a shorter transmission advance. A simulator will find its dead-reckoning task more difficult when it interacts with a device of higher fidelity than itself.

Propagation delays hurt both sides of the communication link. The dead-reckoning task of all simulators becomes more difficult as propagation delays increase. And propagation delays as a function of distance cannot be below the values shown in Table 2.

When simulators are unsynchronized, the worst case synchronization delay equals the frame time of the sending simulator.

Permissible network delays and clock errors are limited by the interval Δt_{DR} over which dead-reckoning can maintain the desired accuracy. The condition may be stated as

$$\delta t \leq \Delta t_{DR} \tag{6}$$

Using Eq. (5), with the clock error taken positive for worst case, this becomes

$$\Delta t_p + |\Delta t_c| \leq \Delta t_{DR} - (\Delta t_{Rra} - \Delta t_{Sta}) - \Delta t_{Sf} \tag{7}$$

In the last expression the subscripts A and B have been replaced by R and S, standing for receiver and sender.

The main ingredient in the upper bound Eq. (7) is Δt_{DR}. The next section is concerned with estimates of this parameter.

IV. Dead-Reckoning

In the last section, dead-reckoning—the extrapolation of position from outdated information—was introduced as a necessity imposed by timing details. Up-to-date information is not available and therefore one must extrapolate. The DIS standard, drawing upon the practice of the earlier SIMNET protocol,[8] introduces dead-reckoning as a process of choice for the purpose of reducing network traffic. Every simulator withholds information as long as same can be dead-reckoned with sufficient accuracy. The originating simulator verifies this last point by dead-reckoning the position and orientation of its ownship and comparing them to its computed ground-truth. Only when the difference exceeds a predetermined tolerance is an update broadcast.

The current DIS standard does not guarantee the selected threshold at the receiver's simulator. That end requires a stricter standard based on Eq. (7). It is not our purpose here to define this stricter standard. Nor do we concern ourselves with the very real problem of how the sending simulator might determine the frame in which an update is required to meet Eq. (7). The purpose of this discussion of the standard is merely as background for the data from which we estimate Δt_{DR}.

In Refs. 5–7 this author, together with K. Graham, derived improved dead-reckoning algorithms for airplanes in coordinated flight. These algorithms were evaluated by computing the number of updates required by DIS rules for a given flight history. (Note that Refs. 5–7 as well as this article use coordinate invariant tolerances.) Conventional algorithms consisting of second-order extrapolation of position and first order for orientation were also treated for comparison. The data used described an F-16 maneuvering in the vertical plane. Reference 7 includes analysis of additional data (X-31, and helicopter). However, the F-16 data proved most conservative. In the following we call on it as representing the worst case.

In the references, an arbitrary threshold of 10 ft in position and 10 deg in orientation was set, and the number of updates tabulated for the different dead-reckoning algorithms. For our present purpose we need the update interval Δt_{DR} as a function of threshold. The information was produced with the software and methods of Ref. 5 and is presented in Table 3 for conventional dead-reckoning, and in Table 4 for the improved dead-reckoning of Katz and Graham (K&G).

Note that the data in Tables 3 and 4 are not the true dead-reckoning limit Δt_{DR}, but rather intervals over which information is withheld under the DIS standard. We denote this interval by Δt^+_{DR}. It is obtained from Δt_{DR} by extending to an integral number of frames.

In Refs. 5–7 the driving consideration was network traffic, which is determined by the average Δt^+_{DR}. (The average is defined as the overall flight time divided by the number of updates. It may be obtained as the arithmetic average of the individual Δt^+_{DR} intervals.) In the current context, the worst case arises in conjunction with the smallest permissible delay. Minimum values of Δt^+_{DR} are tabulated here for the first time. The F-16 data was generated in 25-ms frames. As a result, all values of minimum Δt^+_{DR} in the tables are multiples of 25 ms. The minimum cannot be less than one frame (25 ms).

Our next task is to estimate the minimum value of Δt_{DR} as a function of threshold, based on the data in the tables. Clearly

$$\Delta t^+_{DR} - \Delta t_f \leq \Delta t_{DR} \leq \Delta t^+_{DR} \tag{8}$$

where Δt_f is the simulation frame. So long as the intervals are long compared to the frame, one would expect a uniform

Table 3 Dead-reckoning update interval Δt^+_{DR}, second-order position and first-order orientation

Threshold		Dead-reckoning interval Δt^+_{DR}, ms	
Position, m	Orientation, deg	Average	Minimum
0.1	1	57	25
0.3	1	59	25
1.0	3	149	50
3.0	10	435	150
10.0	30	1067	475
30.0	30	1215	475
100.0	30	1305	475
300.0	30	1355	475

Table 4 Dead-reckoning update interval Δt^{+}_{DR}, with the phugoid method of K&G

Threshold		Dead-reckoning interval Δt^{+}_{DR}, ms	
Position, m	Orientation, deg	Average	Minimum
0.1	1	130	25
0.3	1	173	25
1.0	3	476	100
3.0	10	1305	300
10.0	30	2348	1675
30.0	30	3202	1375
100.0	30	4403	1450
300.0	30	5871	3225

Table 5 Interaction fidelity vs Δt_{DR}

Threshold		Conservative Δt_{DR}, ms	
Position, m	Orientation, deg	Second-order[a]	K&G[b]
1.0	3	25	75
3.0	10	125	275
10.0	30	450	1650

[a]Second-order position, first-order orientation. [b]K&G phugoid method.

distribution of threshold exceedance over the frame, in which case the lower bound in Eq. (8) would prevail:

$$\Delta t_{DR} \approx \Delta t^{+}_{DR} - \Delta t_f \tag{9}$$

As the true Δt_{DR} becomes shorter than a frame, Eq. (9) becomes overly conservative, since threshold violations concentrate in the early part of each frame. In either case $\Delta t^{+}_{DR} - \Delta t_f$ is a lower bound on Δt_{DR}. We are going to use it as a conservative estimate. Table 5 correlates fidelity thresholds with this estimate of Δt_{DR}.

V. Conclusions

We are now in a position to draw some conclusions about what is feasible in the way of global interactive simulation and what is required of the absolute clock in order to realize it.

Table 2 shows that speed-of-light global communications entails net delays of 67 ms. If K&G dead-reckoning is employed, the tolerances of line 1 of Table 5—1 m and 3 deg—can be achieved, leaving a cushion of 8 ms for the other terms in Eq. (7). This is a tight reserve. Without clock error, Δt_f must be no more than 8 ms (a frame rate of 125 Hz). It would not make sense to let clock error use up more than a fraction of 1 ms. By the same token, there is no urgency to make clock errors a very small fraction of a millisecond.

Staying with more traditional dead-reckoning algorithms, one must settle for a tolerance close to that of line 2 of Table 5—3 m and 10 deg. The error budget is not quite as tight; again the clock error should not be more than one millisecond, but does not need to be much less.

With the K&G improved dead-reckoning, the error thresholds of line 2 can be maintained even with a geostationary satellite as a relay station. Note that the K&G methods are predicated on coordinated flying. They apply to loose formations, but not to rigid formations where the wingman must deviate from coordinations. Falling back to the more traditional methods rules out the use of a geostationary satellite for rigid formations.

Traditional methods coupled with transmission by geostationary satellite and the corresponding line 3 tolerances are adequate for many interactions. The timing requirements and the timing error budget are less severe.

In conclusion, a clock accuracy of 1 ms is marginally adequate. This level of accuracy can be approached or achieved by use of radio signals broadcast by the National Institute of Standards and Technology (NIST) from its radio stations WWV, WWVB, and WWVH.[9] At one time this author advocated such a system for DIS and held (on a more heuristic basis) that 1-ms accuracy would suffice.[1] The GPS clock maintains an accuracy about three orders of magnitude better, at no added cost. For this reason the GPS approach should be preferred.

With the GPS system, the accuracy can easily be tightened to say, 100 μs. This helps the timing error budget of networked systems at no extra cost. The other factor of 50–100 in accuracy may be best used by permitting the timing signal from a single GPS clock to be distributed to a number of simulators. Distribution delays of about 100 μs can be tolerated without correction. Based on speed-of-light propagation (Table 2), one clock can serve simulators within a radius of 30 km. Near speed-of-light distribution of a timing signal could be accomplished by a dedicated radio link. Distribution in dedicated wires or optical fibers can reduce the speed of propagation, and with it the permissible size of the site, by a factor of 2 to 3. Distribution by a local net is dominated by processing delays at nodes, which further restricts site dimensions. These considerations as they apply to the architecture of a simulation site are addressed in Ref. 10.

The reader may be wondering what happened to the old-fashioned "transport delay." Does it not affect formation flying over the net? Of course it does. Delayed response makes all control tasks more difficult, and formation flying is no exception. However, transport delay does not enter the question of the consistency between the pictures of reality observed by the two players. This was the only issue we addressed.

Acknowledgments

This work was supported by U.S. Army STRICOM under Contract N61339-93-K-0003. I would like to thank Cuo-Chi Lin of the University of Central Florida, Institute for Simulation and Training, for providing data of F-16 flight that was used in deriving the data, and K. Graham for the use of the software tools developed in our joint work on dead-reckoning.

References

[1]Katz, A., "Absolute Time Stamp in Networking of Simulators," Second Conf. on Interoperability of Defense Simulations, Orlando, FL, Jan. 1990.

[2]"IEEE Standard for Information Technology—Protocols for Distributed Interactive Simulation Applications," *IEEE Std 1278-1993*, Inst. of Electrical and Electronics Engineers, New York, May 12, 1993.

[3]Katz, A., Dudgeon, J., Schamlé, M., and Evans, M., "An Absolute Clock for Distributive Interactive Simulation (DIS) Based on the Global Positioning System (GPS)," prepared for U.S. Army STRICOM, Univ. of Alabama Flight Dynamics Lab. Rept. 94S01B, Contract N61339-93-K-0003, Oct. 31, 1994.

[4]Lin, C., "The Performance Assessment of the Dead-Reckoning Algorithms in DIS," *10th Workshop on Standards for the Interoperability of Defense Simulators*, edited by D. Haworth, Vol. 4, Inst. for Simulation and Training, Orlando, FL, 1994, pp. 239–258.

[5]Katz, A., and Graham, K., "Extrapolation of Airplane Flight," prepared for Loral Western Development Labs, UA FDL Rept. 93S03, Subcontract SO-242825-A, Item 04, San Jose, CA, Dec. 1993.

[6]Katz, A., and Graham, K., "Dead-Reckoning for Airplanes in Coordinated Flight," *10th Workshop on Standards for the Interoperability of Defense Simulators*, edited by D. Haworth, Vol. 2, Inst. for Simulation and Training, Orlando, FL, March 1994, pp. 5–13.

[7]Katz, A., and Graham, K., "Prediction of Airplane States," *Journal of Aircraft* (to be published).

[8]Pope, A. R., "The SIMNET Network and Protocols," BBN Systems and Technologies, Rept. 7627, Cambridge, MA, June 1991.

[9]Howe, S. L., "NBS Time and Frequency Dissemination Service," National Bureau of Standards, NBS Special Publication 432, Washington, DC, Sept. 1979.

[10]Katz, A., "The Absolute Clock in the DIS Scheme," *10th Workshop on Standards for the Interoperability of Defense Simulators*, edited by D. Haworth, Vol. 2, Inst. for Simulation and Training, Orlando, FL, March 1994, pp. 1–4.