Dead Reckoning in a Mixed HLA/DIS Environment

Dr. Tony Valle,
SPARTA, Inc.
Orlando, FL
Tony.Valle@sparta.com

Benjamin Leppard, Christopher Santora
Northrop Grumman
Orlando, FL
Benjamin.Leppard@ngc.com, Chris.Santora@ngc.com

ABSTRACT

The DMO Portal serves as a mechanism to bring simulators using different simulation protocols into a common battlespace. Integration testing of the first HLA simulators with the legacy DIS components has some implications for dead reckoning implementation that arise from the nature of HLA state updates. They do not occur in homogenous DIS or HLA environments. The authors describe the observed anomalies and the implementation developed for the DMON to address them, as well as test cases and procedures that help isolate the discrepancies

ABOUT THE AUTHORS

Dr. Tony Valle is both a military and commercial simulation designer. Dr. Valle was the Lead for the OneSAF Objective System Architecture Advisory Board, and maintains the CAF DMO Master Conceptual Model. He served as the Chief Architect of both the Joint Simulation System (JSIMS) and Advanced Distributed Simulation Technology (ADST) programs and worked for LORAL and Lockheed Martin before taking on the job of Division Manager for the Orlando, FL office of SPARTA, Inc. Dr. Valle's work on commercial air combat modeling includes contributions to a variety of flight and air combat simulations.

Benjamin Leppard is a Software Engineer for the DMO Operations and Integration Program at Northrop Grumman Corporation. He received his Bachelor in Computer Engineering at the University of Central Florida.

Christopher Santora is a Software Engineer for the DMO Operations and Integration Program at Northrop Grumman Corporation. He received his Bachelor in Computer Science at the University of Central Florida.

Dead Reckoning in a Mixed HLA/DIS Environment

Dr. Tony Valle,
SPARTA, Inc.
Orlando, FL
Tony.Valle@sparta.com

Benjamin Leppard, Christopher Santora
Northrop Grumman
Orlando, FL
Benjamin.Leppard@ngc.com, Chris.Santora@ngc.com

Be

DMO PORTAL INTEGRATION

The Distributed Mission Operations (DMO) Operations and Integration (O&I) contract provides a wide area network that links high fidelity virtual aircrew trainers in support of regular inter-team training. The boundary device between the all the simulation assets contained in the local Mission Training Center (MTC) and the wide area DMO Network (DMON) is known as the DMO Portal. The Portal provides several services including simulation protocol translation, network routing, simulation flow control, and bandwidth control. A sample exercise configuration is shown in Figure 1.

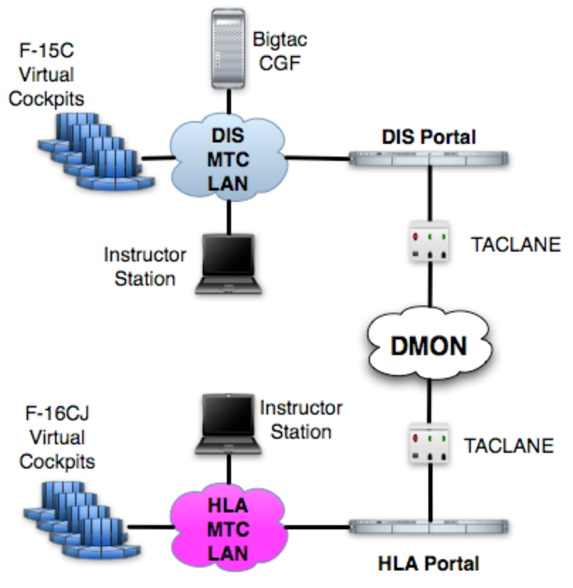

Figure 1. Sample DMO Exercise Configuration

The bandwidth control functions include not only a variety of filters and means to partition traffic, but also a mechanism for reducing the update rate of entity state information over the DMON when that data is not essential. This latter function interacts differently with HLA and DIS simulations as discovered during testing of the latest Portal release.

PROBLEM DEFINITION

The DIS and HLA protocols (the latter through a chosen FOM) use "dead reckoning" (or DR) techniques to reduce the amount of network traffic required to maintain consistent distributed state for a simulated entity. The basic idea behind DR is that entities publish a *projected future state* using a known algorithm with parameters. The owner of the entity maintains both the true behavior and the projected state and updates both when they differ by more than a pre-negotiated *threshold* value. There are 11 DR algorithms discussed in the IEEE 1278.1-1995 standard, but we primarily encounter only 2 of them in common use for dynamic entities: algorithms 3 and 4. DR3 is a "constant velocity" projection with a state equation of

$$\begin{aligned}\mathbf{x}(t) &= \mathbf{x}(t_0) + \mathbf{v}\cdot(t - t_0)\\ \mathbf{v}(t) &= \mathbf{v}\end{aligned} \qquad (1)$$

where t_0 is the time of the last state update and **v** is a constant vector. DR4 is a "constant acceleration" projection with a state equation of:

$$\begin{aligned}\mathbf{x}(t) &= \mathbf{x}(t_0) + \mathbf{v}(t_0)\cdot(t - t_0) + \tfrac{1}{2}\mathbf{a}\cdot(t - t_0)^2\\ \mathbf{v}(t) &= \mathbf{v}(t_0) + \mathbf{a}\cdot(t - t_0)\end{aligned} \qquad (2)$$

Where **a** is an additional constant vector parameter. In both cases, a state update must be issued when the actual state of the object and the project position differ by more than 1 meter in any coordinate.

Portal Interactions

The Portal uses two techniques that interact with normal DIS dead reckoning algorithms, which are also used in the RPR FOM based HLA implementations found on the DMON. First, the Portal uses independent dead reckoning on its own to reduce the state update rate across the wide area network. This DR could in principle be distinct from the locally generated DR of the entity owner, so long as the Portal understands and maintains projected future motion correctly for the local entity using the owner's specified algorithm. For example, early versions of the Portal used DR4 at all times, assuming a zero for the **a** vector when the owner specified DR3.

In addition, the Portal uses *thresholding* to ensure that entity state updates are not too frequent. Because the DIS standard does not prohibit entity owners from updating state as frequently as they wish, and because passing all such state updates over the WAN could saturate the available bandwidth, the Portal ensures that updates are propagated only when they actually exceed the DR threshold. This ensures that too-frequent state updates are confined to the local MTC network.

Second, the Portal employs *heartbeating* mechanisms to allow HLA and DIS sites to interoperate. HLA and DIS differ fundamentally in that HLA only sends state updates when the state changes while DIS sends and expects regular updates for all entities. The Portal fills the gap by sending entity heartbeats to the DIS sites on behalf of the HLA site. Dead reckoning algorithms must be used to ensure that the positional data in the heartbeats accurately reflect the current state of the HLA entities.

These interactions between the Portal and local simulator dead reckoning led to issues that arose in integration testing. In particular, they became evident only when virtual simulators using different simulation protocols interacted in close visual range.

OBSERVED ANOMALIES

Three distinct types of behaviors arose in integration testing that had to be diagnosed and resolved, known as *rubber banding*, *ratcheting*, and *jitter* based on their visual appearance. These are described below.

Rubber Banding

Rubber banding is a visual anomaly in which an aircraft in close formation seems to jump back to a previous position and then jump forward to the correct position some time later (on the order of seconds). The displacement seen could be large – thousands of feet or more. This problem arises solely because of properties of the HLA protocol. It was not observed to occur between two visual simulators that used only the DIS protocol, but only when one site was an HLA MTC.

Ratcheting

Ratcheting is an anomaly observed in smooth, medium-rate formation turns. The aircraft in formation seems to slide to the outside of the turn for several seconds and then snap back into the proper location, only to repeat the motion again from this position. It was observed only in certain cases when a constructive simulation generated one of the objects. Eventually this was determined to be associated with two separate issues: dead reckoning algorithm mismatches, and the event-driven nature of the DMO Portal.

Jitter

Jitter is a set of small, apparently random, motions of an aircraft in formation around its expected position. The distances are on the order of a few feet and the time scale is widely variable.

All of these anomalies were eventually traced to interactions between different dead reckoning algorithm implementations on different devices. In the case of rubber banding and ratcheting, the anomalies were sufficiently large to impact the fidelity of inter-team training and would have resulted in unacceptable training limitations. Jitter is a much milder phenomenon, which is caused by the fact that any DR algorithm will deviate from smooth motion by an amount on the order of the threshold value. Eliminating jitter entirely would require very tight threshold values, or much more sophisticated DR algorithms than currently defined in the DIS and HLA standards.

It was observed that some simulations in the DMO environment produced entity updates at a rapid rate. Thresholding was introduced to transmit updates only when the threshold is broken. This caused a serious unintended side effect due to interactions with DR.

A naïve implementation of thresholding would simply compare the current packet's position with the last packet's position dead reckoned to the current time. This turns out to be too restrictive, because it doesn't take into account changes in the DR parameters. An entity update that has no deviation from the projected track but has substantially different velocity or acceleration can break threshold in a matter of tens of milliseconds. Or, a more common case, an entity begins a gradual turn and sends out a state update. The Portal will drop this update because the position will not *yet* have broken the position threshold. Because the turn corresponds closely to the sending simulation's DR contract path, it will not send out another position update until the heartbeat period has elapsed. By this time a significant deviation has occurred between the state of the sending and receiving simulation. This is a problem specific to implementing thresholding in a gateway.

Normally simulators will check for threshold violations at frame rate (e.g. 50 times per second), but the Portal is event driven and checks the threshold on a packet-by-packet basis. To reconcile this, the Portal implements an algorithm that solves for the time at which the two different DR tracks will differ by the threshold. An update is scheduled for that time. If a new update is received before the update is scheduled to be sent, the update schedule is recalculated. If an update is not received the last received update is sent dead reckoned to the current time. This solution was not without problems as discussed below.

ANALYSIS & RESULTS

One the major challenges in understanding the DR-related anomalies and diagnosing the causes was the nature of the test environment. The anomalies were easily observable in the visual systems associated with a virtual cockpit but not in typical plan-view display systems. Consequently, testing required that simulator assets be committed to the test, taking away from their availability as training assets. While this was mitigated to some extent by using the developmental sites as much as possible, the differences between the development and fielded systems also present difficulties. Furthermore, the O&I team does not have access to a visual system that can be used to reproduce the observed anomalies, and had to rely on verbal descriptions of them and log file analysis to understand the nature of the problem. This in itself presented a challenge as in most instances the log file does not contain evidence of the problem! It is the rapidly updated projected state of the entity as done by the visual system that produces the observed behaviors.

Delay Limitation

The simplest problem to diagnose and correct, at least partially, is jitter. Because of the thresholding done by the Portal, state updates are not always propagated across the wide area network. In formation flying, particularly, pilots are making continuous small corrections to the stick, rudder, and throttle. These produce small, discontinuous changes in velocity, which in turn generate state updates.

But the future projected position of the state matches the previous project position very closely. This small update may not be sent over the wide area network, producing a discrepancy in the projected future state of the entity as computed by the local Portal (at the MTC which owns the entity) and any remote Portal located at some other MTC on the DMON. As a result, the Portal on the entity owner side would project a time in the future at which the difference between the remote and local projected states would differ by a threshold amount, and schedule an event at that time to send the update over the wide area network.

The net result is that differences in projected and actual states are virtually guaranteed to build up to the level of the DR threshold value, and when state updates are finally issued a visual system produces a jump on the order of a meter. These are especially apparent in manned simulators flying in close formation.

The solution is to ensure that updated state information held by the local Portal is eventually distributed over the WAN. The maximum allowed delay should be set to compromise between the size of visual jitter anomalies and the need to control network bandwidth. Because the small changes in state induced by formation flying do not build up quickly, this time can be set on the order of a few tenths of a second and still produce a marked reduction in jitter magnitude and frequency.

Algorithm Mismatch

Different simulation assets at a given MTC may use different dead reckoning algorithms. Because all DIS- or HLA-compatible simulators are supposed to be able to use any of the DR algorithms defined in the respective standard, the owner of an entity is free to select the one that best suits the motion of that entity. In order to speed development and make best use of the most sophisticated DR algorithm, the Portal development team settled on the use of DR4 across the wide area network, regardless of the algorithms selected by the entity owner. The mismatch between the owner and Portal DR algorithms resulted in the ratcheting seen during integration testing. The chain of state updates is shown in Figure 2.

The figure shows a simulated aircraft in a steady-state constant velocity turn to the right. In this case, the constructive simulation that owns the entity produces a state update that is of type DR3, but nevertheless has a valid non-zero acceleration value. The local Portal acquires this state and translates it to DR4, preserving the original positional data. The DIS standard recommends that unused DR fields should be set to zero so the Portal assumed that this would always be the case for acceleration in DR3 updates. As such, the DR3 entities would behave properly even when treated like DR4. However, this assumption does not hold in all cases and some simulations provide non-zero acceleration values even for DR3 updates. As a result, both the local and remote Portal propagate the entity as if it is moving with constant acceleration, along a parabolic path that closely matches the actual circular path for a long interval. But the original DR “contract” is preserved over the network – the remote Portal publishes the same DR3 algorithm values with the non-zero acceleration value.

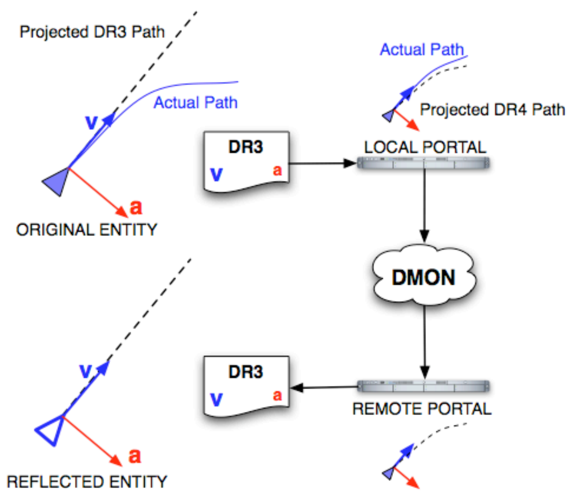

Figure 2. DR Algorithm Mismatch

The owning simulation produces frequent state updates as the entity diverges from the project straight line path, but the local Portal sees the entity maintaining good agreement with the DR4 projected state, and so does not send the state update across the DMON. At the remote site, the local visual system continues to propagate the entity in a straight line until finally the circular motion deviates from the parabolic motion and a state update is sent over the WAN. This results in a noticeable discontinuous update in position, and the process then repeats.

Interestingly, this problem only arose during moderate-G turns. Analysis revealed that if the aircraft were turning at a low rate, the entity heartbeat timer would fire before the deviation from straight-line motion became visually large. And when the turn rate was high, the deviation from the projected path occurred very quickly, resulting in more frequent state updates so once again the deviation could not build up. A dynamic situation with changing turn rates had the same effect. Only moderate-rate, constant G turns produced pronounced ratcheting, which resulted in the observation being infrequent and spurious, and test and diagnosis more difficult.

One possible resolution to this problem is to “zero-out” the acceleration value for any DR3 contract received. This would cause the DR4 algorithm used by the Portal to correctly reflect straight-line motion. But there are other reasons to include current acceleration values besides support for DR4. For example, current acceleration might be used to help resolve the probability of hit for an air-to-air missile. For this reason, it was decided to preserve the acceleration value.

Another possible approach would be to publish the entity with a DR4 contract at the remote site. That would make the remote visual system agree with the projected motion seen by the local Portal. But it implies that the motion model is of a higher order than it actually is, and complications may arise from such a misrepresentation that are best avoided.

The final solution was to implement both DR3 and DR4 state propagation algorithms in the Portal and reorganize the internal state to associate entities with their native DR technique.

Predictive Thresholding

Even after correcting the DR algorithm mismatch issues, ratcheting was still observed during testing. This was due to the fact that the Portal is a gateway as opposed to a standard simulator. Normally simulators check for threshold violations at frame rate (e.g. 50 times per second), but the Portal is event driven and checks the threshold on a packet-by-packet basis.

The original implementation of thresholding simply compared the current packet’s position with the expected position based on the last packet’s state and the elapsed time. This turned out to be too restrictive, because it didn’t take into account changes in the DR parameters. An entity update that has no deviation from the projected track would be dropped even though it has substantially different velocity or acceleration vectors. This track can break threshold in a matter of tens of milliseconds. But since the Portal drops this update, it doesn’t update the remote Portal with the new path until the next heartbeat. This causes ratcheting.

For example, an entity begins a gradual turn and sends out a state update. The Portal drops this update because the position has not *yet* broken the position threshold. The receiving simulation’s visual systems show the entity drifting away from the turn. When the heartbeat period has elapsed the local Portal finally sends an update. By this time a significant deviation has occurred between the state of the sending and receiving simulation and visual system shows the entity snap back to the real position.

To reconcile this, the Portal now implements an algorithm called *predictive thresholding* that determines the time at which the two DR tracks will differ by the threshold. A WAN update is scheduled for that time. If the local Portal receives a new LAN update before the next scheduled WAN update is sent, then the schedule is recalculated. Otherwise, the local Portal dead reckons the last received update to the current time and sends it to the remote Portal. This algorithm was not without problems and resulted in minor jitter as previously discussed.

Partial State Updates
One of the advantages of the HLA protocol is that state attributes can be updated individually rather than all at once, as is the case for network-based protocols. In particular, an entity state update in DIS always contains the compete current state information for the entity, even if the only change is due to a change in appearance. In HLA, an appearance change may result in a state update call from the RTI that only contains the changed appearance attribute and nothing else.

A Portal at an HLA MTC is maintaining the state of entities owned by the local simulation assets. When state updates are received, it must pass those updates over the DMON to remote sites if they break the thresholding criteria. Because HLA uses partial state updates, the local Portal caches all the state information required to fill in a DIS Entity State PDU. This allows the Portal to fill in information such as appearance attributes when a partial state update is received from the RTI.

In testing, the RTI used was shown to update position, velocity, and acceleration in their entirety along with any other DR parameters whenever a state update was issued. What was not initially noticed was that when an appearance update occurred, no other entity state data was received. This led to a situation where the prior DR parameter data in the cache was retransmitted to the remote site when a local entity made a change in an appearance attribute alone.

The effect is most pronounced when aircraft were in formation at constant velocity. In this situation, the update rate is at an absolute minimum, occurring at the DIS heartbeat rate. If a virtual simulator in an HLA MTC makes an appearance change by selecting afterburner power, or changing the state of running lights, for example, the state update will contain only the effected attribute. This causes the local Portal to issue a state update using cached information, which can be several seconds old because of the low update rate.

When a remote site receives the update, the effect is to jump the reflected entity back in time by the elapsed interval since the last state update. When the next complete state update comes in, the entity jumps forward to the correct location. With aircraft traveling at 1000 feet per second, and a heartbeat interval on the order of 5 seconds, the length of the jump can be on the order of a mile.

The solution is to ensure that state updates passed out of an HLA portal are always dead-reckoned up the time of transmission. This ensures that no matter what causes a state update, it is always the current "best known" position and velocity for the represented entity.

OTHER CONSIDERATIONS

HLA also presents some other challenges in the DMO environment, some of them drivers in the choices made for the Portal implementation.

HLA entities do not have a "heartbeat" as DIS entities do, so whenever both types of MTC are present in an event, the Portals must conspire to issue entity state updates at DIS sites within the heartbeat interval or the entity will be removed by the DIS simulations. Since the Portals share a common set of projected paths, there is no reason to carry this redundant information over the WAN as it can be generated locally. But DIS Portals cannot continue to update the HLA-owned entity indefinitely because they might then never remove entities in the event of a network or Portal failure. So another heartbeat timer must be used, though it can be set to a much longer interval than the standard DIS heartbeat. As the DIS standard changes and a variety of different heartbeat timers become part of the standard, this will introduce further interaction complexity that will have to be tested and issues resolved.

Similarly, because of the performance of the RTI, the HLA MTCs in use on the DMON are not able to tolerate very high state update rates from DIS simulators, and would prefer to receive only changed data. The original thresholding algorithm was put in place in the Portal to address this issue and ensure that only necessary updates were delivered to the HLA simulations. At present, state updates issued to HLA MTCs are cached, and only the attributes that have changed since the last update are published.

SUMMARY

Operating in a mixed HLA and DIS simulation environment requires some device that translates between the two disparate simulation protocols, in our case the DMO Portal. That device will necessarily be forced to implement adjustments in dead reckoning to account for the differences between HLA and DIS, and perhaps other changes if required for performance reasons. These adjustments can have both obvious and subtle visual effects for simulated entities, especially for fast-moving objects such as those encountered in air combat simulation. Partial state updates as used by HLA introduce complications, as does the lack of a heartbeat update. These complications can be resolved through comprehensive integration testing and careful implementation of the algorithms in the edge device,

ACKNOWLEDGEMENTS

The authors gratefully acknowledge the support of the USAF SMSG/TO. Also, special thanks go to the members of the O&I team who were important in the analysis of the testing results: Shane Marcus, Melissa Fish, Jessica Crane, Seth Lytle, Jennifer Iding, and Valerie Valle.

REFERENCES

IEEE 1278.1-1995. IEEE Standard for Distributed Interactive Simulation— Application Protocols

IEEE 1278.1a-1998. IEEE Standard for Distributed Interactive Simulation— Application Protocols

IEEE 1516 Standard for Modeling and Simulation (M&S) High Level Architecture (HLA)—Framework and Rules