Integrating DIS V8, Challenges and Opportunities

Lance Call
CAE USA/AFRL
Dayton, OH
lance.call.ctr@us.af.mil

Dean Lewandowski
CAE USA/AFRL
Dayton, OH
dean.lewandowski.ctr@us.af.mil

ABSTRACT

Distributed Interactive Simulation (DIS) Version 8 (V8) will be the largest change to DIS in over 25 years. It will not be on-the-wire compatible with previous DIS versions, but it will be possible to translate from DIS Version 7 (V7) into DIS V8. These large changes provide challenges as well as opportunities for DIS applications. Some of the known challenges are a change in endian byte order, partial Protocol Data Unit (PDU) updates, a new dead reckoning algorithm, more variable PDU content, and, on the upside, potentially continuous ongoing yearly updates. DIS V8 provides the opportunity to increase fidelity in many areas including Electromagnetic Emission (EE) and Infrared Emission (IR) modeling. DIS V8 can improve network efficiency by updating multiple entities in a single PDU, as well as taking advantage of partial PDU updates. DIS V8 will allow a standard XML definition of the PDUs which can support automatic code generation for marshaling and unmarshalling DIS data, Wireshark dissectors, as well as potential automatic conversion to High Level Architecture (HLA) Real-Time Platform Reference (RPR) Federation Object Module (FOM) and Test and Training Enabling Architecture (TENA) RPR Object Model.

This paper covers the analysis of possible improved data bandwidth usage by up to 75% and reduction in PDU rates by up to 40 times. The paper will discuss conditions required for improved network bandwidth relative to DIS V7. A proposed approach and algorithm to integrate the multiple entity PDUs into existing applications is provided. The results of attempts to implement this approach and algorithm into the Air Force Research Laboratory (AFRL) Network Integrated Constructive Environment (NICE) threat system will be detailed.

An overview of the EE PDU modeling fidelity, including AESA radars and jamming, is provided. The new Combined Parabolic/Circular (CPC) dead reckoning algorithm will be described, and results of implementation in the NICE threat system is presented. Practical experience of dealing with the endian byte order change, new PDU format, and development of DIS V7/V8 gateway challenges are presented.

This paper will provide practical implementation information and guidance for those who are considering implementing DIS V8.

ABOUT THE AUTHORS

Lance Call is a Scientist, Systems and Software Engineer with CAE USA at the Air Force Research Laboratory (AFRL). He graduated Magna Cum Laude with a Bachelor of Science degree in Electronics Engineering Technology from Brigham Young University in 1988. He has worked on real-time threat systems, and integration of live, virtual man in the loop, and computer only simulations. He has been responsible for Cross Domain Security systems and rule set development, improving threat systems, and integrating simulators with live aircraft systems. He was the SISO Compressed DIS (C-DIS) drafting group editor. He is an IEEE member.

Dean Lewandowski is a Software Engineer with CAE USA at the Air Force Research Laboratory (AFRL). He graduated with a Bachelor of Science degree in Electronics Engineering Technology from DeVry University in 1987. He has researched, developed, and integrated physics-based, real-time, virtual man in the loop training platforms, simulated threat environment models, Computer Generated Forces (CGF) applications, and game-based Instructor/Operator controls for constructive entity tactics and behavior.

Integrating DIS V8, Challenges and Opportunities

Lance Call
CAE USA/AFRL
Dayton, OH
lance.call.ctr@us.af.mil

Dean Lewandowski
CAE USA/AFRL
Dayton, OH
Dean.lewandowski.ctr@us.af.mil

HISTORY OF DIS

Distributed Interactive Simulation (DIS) has had seven different versions with the latest one being DIS Version 7 (V7) [see IEEE (2012) and IEEE (2015)]. All of these standards have used a very similar format with changes being made by adding new Protocol Data Units (PDUs) or modifying padding fields. A PDU is a network packet with a standard specific DIS header and data format for each PDU defined by the DIS standard. This has made PDUs compatible between versions to a large extent, and has made updating from one standard to the next mostly straight-forward. However, DIS Version 8 (V8), summarized in Murray, Robert (2018), will make extensive changes to the PDU design and structure that will break the backward compatibility with all previous DIS Versions. This is being done to address issues identified in over 30+ years of use, and to make it more flexible going forward. These changes will require much more extensive changes to existing DIS interfaces and systems. This will bring both challenges and opportunities that will be explored in this paper.

CHANGE IS HARD, BUT NECESSARY

DIS is an interface standard based on sending messages called PDUs between simulations to share sufficient truth information to allow the simulations to share state and events and interact in real time. This software and approach become deeply engrained in systems and is a foundational part of a simulator system. When that foundation changes, it can cause major changes in how the entire system is architected and levy new requirements on the fidelity of the simulator. Changing this foundational part of the software can be difficult, as well as potentially expensive. From a program perspective, it is hard to justify changing the interface unless there are significant new benefits, which can be better network efficiency, new capabilities, or growth opportunities. Better network efficiency is good, but it may not be worth the expense of changing unless bandwidth is a major bottleneck to the current system or much higher entity counts are required. New interface standards must provide significant new capabilities that are not possible with the existing standard in order to justify the cost of the change. DIS V8 will be much more bandwidth efficient than DIS V7, thus supporting higher entity counts without physical network changes. DIS V8 supports new capabilities that are required to support higher fidelity training as well as providing opportunities for future growth that are not possible with DIS V7.

As with any new major interface change, there must be enough systems that adopt the new standard in order to have other systems to operate with, before the new standard is fully effective. It is likely that groups such as the United States Combat Air Force Distributed Mission Operations (CAF DMO) will need to see the benefits and adopt the new standard to encourage and pay for all of the participating platforms to move to the new standard in order to see the benefits of DIS V8. Groups that currently use DIS V7 should begin now to consider adopting DIS V8 and look for strategies for implementation among their users. This paper provides insight into the challenges and opportunities for DIS V8 integration.

INTEGRATION APPROACHES

Initial integration of DIS V8 could be done by essentially making the existing simulator interface a DIS V7/V8 gateway, and only supporting existing DIS V7 messages. This would be the lowest cost and fastest option, and provide some network efficiency gains, but it will not allow the full benefit of DIS V8. This might however be a good first step for many applications as they become familiar with DIS V8 concepts. An alternate approach would be to use a DIS V7/V8 gateway to translate between systems as they are updated. To take full advantage of DIS V8, the simulations must be improved with higher fidelity Active Electronically Scanned Array (AESA), and Infrared (IR) modeling. The simulation architecture must also be modified to take full advantage of partial updates and the

Multiple Entity capabilities using a single PDU. These will require many more changes than previous DIS version updates. These changes might be implemented as a second phase over time to reduce the initial DIS V8 adoption expense.

OVERVIEW OF THE DIS V8 FORMAT

Before discussing specific challenges and opportunities, it is necessary to have a basic understanding of the DIS V8 PDU structure. Every DIS V8 PDU will have three parts (see Figure 1):

1. First, there will be a 16-byte header that will include a new 64-bit timestamp similar to the time that is commonly used for computer system time. The PDU header contains fields that should allow more future forward compatibility including the compatibility version to indicate the oldest compatible PDU version and an explicit header length so that additional bytes could be added to the PDU header in the future.

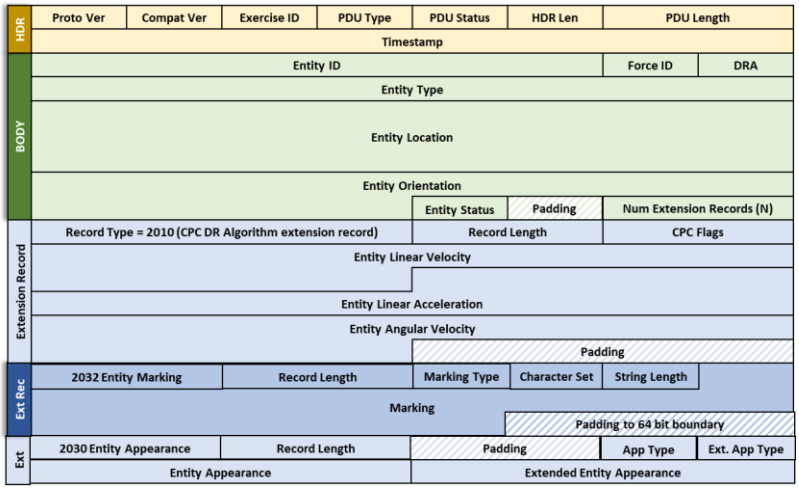

Figure 1 New DIS V8 format with header, body, and extension records

2. Second, there will be a fixed-length body that provides the information required by every PDU of that type. The body is intended to provide sufficient information to quickly and effectively filter PDUs of interest for an application. It may not provide complete state information. The number of extension records is always the last 16 bits in the body, which makes XML representation simple and consistent between PDUs.

3. Third, there will be 0 to N extension records that provide the remainder of the information. Extension records follow a common format with a 16-bit Extension Record Type (ERT) enumeration followed by a 16-bit Record Length followed by the data for that particular type of record, which is required to end on an 8-byte boundary. Receivers are required to use the 16-bit Record Length field when parsing in order to progress to the next record. If a receiver does not recognize or cannot utilize a particular ERT, it can simply skip it and go to the next extension record based on the record length. This will allow new ERTs to be added in the future without breaking compatibility with existing DIS V8 systems. In DIS V7, this would have required applications to throw all the information in the PDU away because it could not completely parse the data. In DIS V8, an application will be able to parse and use at least the information it recognizes while ignoring new additions.

Much of the data is provided in extension records, but those records may or may not be provided in every update. For items that do not change very often, it is only necessary to provide that information once during each heartbeat interval. It also allows customizing the PDUs to send only the data that is applicable rather than sending fields with zeroes that will simply be ignored. This will allow the PDUs to be smaller and more efficient.

The Entity State PDU body, for example, will contain the location and orientation of the entity, but it will not contain the velocity or acceleration because static (non-moving) entities do not require velocity or acceleration information. The velocity and acceleration will be provided in a dead reckoning extension record only for moving entities that actually require this information. This approach provides smaller PDUs for static entities leading to less network bandwidth. The Appearance and Entity Marking have been moved to extension records because these fields do not change often, and therefore only need to be sent once during each heartbeat interval allowing other updates to omit this information. This makes smaller PDUs and less bandwidth for many updates. An initial implementation of a DIS V7 to DIS V8 gateway provided a 25% reduction in bandwidth without using any Multiple Entity PDUs which promises even more efficiency.

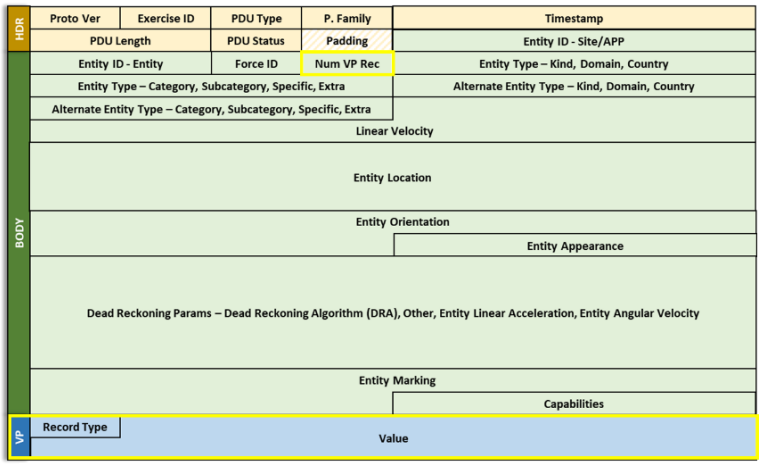

Figure 2 DIS V7 Entity State PDU with one Variable Parameter record.

Any number of extension records may be added to a PDU up to the maximum DIS V8 PDU size of 8192 (8K) octets. It is recommended, however, that PDUs should be kept below the DIS standard Maximum Transmission Unit (MTU) size of 1400 octets in order to keep PDUs from fragmenting on Ethernet networks, even if they are encrypted for security.

By contrast DIS 7 has a header that is not aligned on an 8-byte boundary (see Figure 2), followed by bodies that have different formatting schemes depending on the PDU due to the fact that they were developed by separate groups over time. Unused data is zeroed, but must still be sent. Fifty four percent of DIS V7 PDUs do not allow additional information in the PDU, making extension impossible for those PDUs. In PDUs that do allow extension, there are 16 different variable records, 9 of which are duplicate in definition and 7 which are unique. Four of those records have no length, and 3 have no type or length, which means they cannot be parsed unless they are recognized. This makes adding information difficult or impossible in DIS V7. The count of the variable records (shown by the first yellow outline in Figure 2) could be located anywhere in the DIS V7 PDU, making it difficult to represent in XML and support auto code generation. DIS V8 standardizes the extension record count as the last item in the body with a single format type enumeration and length that allows all PDUs to add extension records and be parsed even if they are not recognized and used by an application.

IMPLICATIONS OF BREAKING COMPATIBILITY

The Simulation Interoperability Standards Organization (SISO) DIS Product Support Group (PSG) chose to break compatibility with DIS V7 in order to provide benefits that would not be possible within the DIS V7 paradigm. This means that simulations that are currently using DIS V7 will require significant software changes in order to send and receive DIS V8 messages. DIS V7 equivalent capabilities can be translated using a gateway. Applications can take advantage of the XML schema and PDU definitions for DIS V8 to auto generate marshalling and unmarshalling code to ease some of the changes. The XML definition may also allow for the automatic generation of HLA RPR FOM and TENA RPR Object Models. There will be additional information to track internally within the simulation such as AESA radar states, IR signatures, laser weapons, weather, and other information that are not part of the DIS V7 standard.

Seven major changes from the previous DIS versions are:

1. Little-endian byte order
2. Partial PDU updates
3. Combined Parabolic/Circular (CPC) dead reckoning algorithm
4. Ability to send multiple entities in a single PDU
5. Support for higher fidelity modeling of AESA radars and jamming
6. Support for higher fidelity modeling of IR
7. Continuous periodic updates based on SISO-REF-030

By making these changes, new opportunities for advancement will be available to DIS simulations, but they will also provide challenges. This paper provides a short overview of how these changes provide challenges as well as opportunities.

CHALLENGE 1: LITTLE ENDIAN BYTE ORDER

In the 1990's when DIS was created, the most common computers were big-endian (most to least significant byte order for multi-byte numeric values). Today, most computers used in simulation are little-endian (least to most significant byte order). This means that most DIS V7 applications today must swap the current big-endian network

byte order into the local little-endian byte order when marshalling or unmarshalling data from the network. DIS V8 will change to little-endian byte order in order to make marshalling and unmarshalling data simpler as well as faster and more efficient for most applications.

This means that software developers will need to change all of the code that sends and receives data. DIS V7/V8 gateways will be required to handle both little-endian and big-endian network data, which may require multiple libraries or some other way to control the byte swapping on the fly. While this code is not difficult conceptually, it does affect every read and write to the network, and it can easily be done incorrectly leading to challenging software bugs. We discovered this when creating our DIS V7/V8 gateway for verification of the DIS V8 concepts. We created two separate low-level libraries to deal with the proper endian orientation. This requires care when invoking calls so that the correct library is being used. This low-level software also exists in many mature libraries that have been used for decades without modification. This may mean that the personnel familiar with that software are no longer available to make the modifications.

CHALLENGE 2: PARTIAL UPDATES

DIS historically has provided all of the information for stateful items such as entities and emitters in a single PDU. This makes it easy to simply read in a PDU and have a complete definition of the current state of the entity. This makes it relatively easy to copy in the new information and use that same format or very similar format of the PDU as the object definition internally in the simulation software.

Because of the new PDU format, simulations may now provide only information for extension records that have changed in the PDU or may send some extension records with heartbeats that are slower than the default PDU update rate. For example, information about the IR signature of an entity may be sent with a heartbeat of 30 seconds while the position updates are provided with five second heartbeats. This minimizes sending redundant information and makes DIS V8 more efficient. The partial updates also allow one simulation system to update one set of extension records, while a second separate simulation system updates a different second set of extension records. Receivers must combine all of the updates into a single internal object state. Figure 3 illustrates a system that is using two subsets, SS1 and SS2, that are sent independently of each other to provide information that updates a single state such as an Entity State.

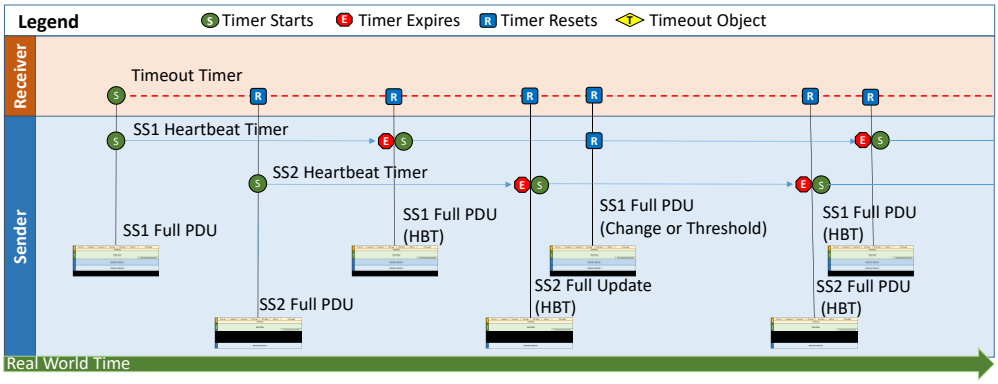

Figure 3 Illustration of the Partial update issuance method using two subsets (SS1 and SS2)

What this means for DIS V8 applications is that they will be required to keep a local state that represents a complete state of information they need to process, which can be updated piece by piece rather than completely replacing the existing state. The local internal state will need to be a complete copy of all possible information that might be received or be flexible enough to add pieces as they are received. Applications may not have full state information for up to two heartbeat periods, and information may be received via multiple PDUs. Knowing what kind of an object to allocate may be difficult with only partial information.

If the individual piece approach is used then when processing the object internally, it will be necessary to loop or switch and constantly check for which pieces are available or being used rather than being able to access the information directly, making the process less efficient, as well as adding complexity. If the parts are added in the order they are received, the order of the parts will also vary, making internal processing more difficult.

The DIS V8 approach is well suited for efficient network transmission, but less suited for internal object state usage. The DIS V8 approach will provide the opportunity for higher fidelity data, and for the ability to add more information over time to improve the simulation. Having this flexibility with internal simulation objects complicates the simulation software.

CHALLENGE 3: CPC DEAD RECKONING ALGORITHM

Dead reckoning has been an enabling technology for distributed simulation since the beginning. DIS V7 references eight dead reckoning algorithms. The algorithms would more accurately be called formulas. There are two categories of these formulas, the so-called World and Body formulas, named after the coordinate systems where the extrapolation is performed. These algorithm categories have been renamed parabolic and circular. The parabolic formula (formerly World algorithm) extrapolates in parabolas (with acceleration) or a straight-line path (no acceleration). The circular formula (formerly Body algorithm) extrapolates in a circular or spiral path.

DIS V8 establishes a single algorithm for selecting the best formula to use for dead reckoning, described in Murray (2025). The algorithm uses the entity type, current motion values, and input guidance from the user to select the best mathematical formula to match the likely path of the entity. All simulations are expected to implement it. The mathematics of the circular formula are fairly complex and difficult to implement. Once libraries to perform the required functions are created however, they are quite straight-forward to use. It requires that implementors have someone who has a good mathematics background to implement the libraries. Because this has historically been shown to be a challenge, Murray has created an open source C based software library that implements the full CPC algorithm at https://sourceforge.net/projects/dsu. [see Murray (2025) and Call (2024) for additional details].

Applications will either need to implement their own libraries or integrate a third-party library such as the referenced open source library into their software. This will change how the existing network code operates, but the internal movement of the entities will also need to use the same algorithm in order to maintain positional accuracy. For example, if the CPC dead reckoning algorithm (DRA) selects the circular formulas for an aircraft then receivers of that information will need to use the circular formulas internally to keep an accurate location for that entity. If the parabolic formulas are used internally rather than the circular formulas, then the internal position will deviate from the true position by much more than the threshold value. The longer update time of the circular formula will lead to large positional errors which will be seen at the next network update resulting in large jumps or large errors that need to be smoothed. This CPC change will therefore affect both the network and internal update processes which is a much more significant change than previous DIS updates.

CHALLENGE 4: ABILITY TO SEND MULTIPLE ENTITIES IN A SINGLE PDU

DIS V8 adds new Multiple Entity extension records that allow for the definition of multiple entities in a single Entity State PDU. There are specific records for static, moving, and accelerating entities.

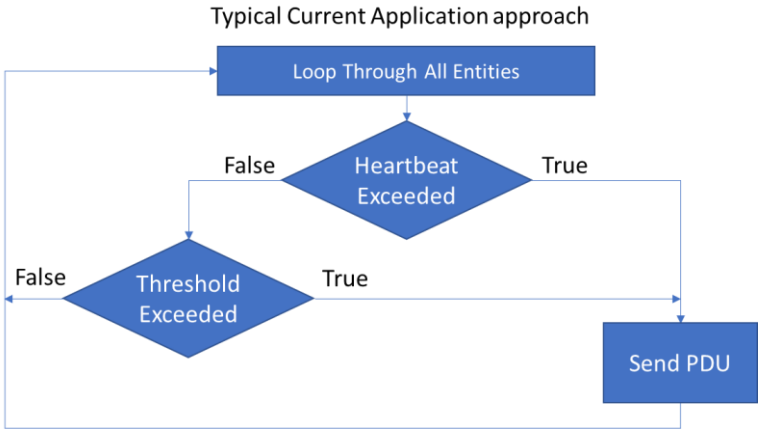

Figure 4 Typical Current approach to updating DIS data

This will require receivers to process a single PDU and update the internal state of many entity states. This type of processing has not been previously required. It may require some refactoring and restructuring of existing network approaches in order to support this capability. It may require a change in how a simulation loops through and updates all of its entities.

A larger challenge is for simulations that produce multiple entities to populate the PDUs. It is expected that the most common usage will be for Computer Generated Force (CGF) systems that want to generate many thousands of entities. How does a CGF meet this challenge?

Figure 4 illustrates the typical current network update approach for CGF's. The CGF loops through all entities, and checks each one to see if the heartbeat time has been exceeded or a threshold has been exceeded. If either of these is true then it sends an updated PDU to the network. We propose that applications use an algorithm similar to the one shown in Figure 5. The application would loop through all of the entities it is updating as usual, and perform the heartbeat and threshold checks. But rather than sending that data to the DIS network it would need to filter the entities that are of the same Kind and Domain and queue them into temporary PDU buffers. This is necessary because entities within a single Entity State PDU must have the same Kind and Domain. Each entity type (F-22, F-35, SA-20, etc.) would be added to a unique Multiple Entity extension record in the PDU buffer. This is illustrated in Figure 5 as Type A, Type B, etc. Data could actually be stored in a temporary PDU buffer or a pointer to the objects to be added to the list could be saved. As items are added the algorithm must estimate the size of the final PDU using the equations shown for PDU size in Figure 5. When the PDU size exceeds a maximum size chosen by the user, such as the DIS V8 Maximum MTU size of 1400 bytes, then the software uses the PDU buffer or lists to populate the PDU and sends it. Once the PDU is sent, the buffer or lists are cleared and the process continues until all entities have been processed. Note that the algorithm and size calculations are different for static, moving and accelerating entities. Static, moving, and accelerating entities would need to be handled in separate PDU buffers or lists and use Multiple Static, Multiple Moving, or Multiple Accelerating Entity extension records as appropriate. Each entity class has a different size (32/48/72) bytes for static/moving/accelerating entities respectively.

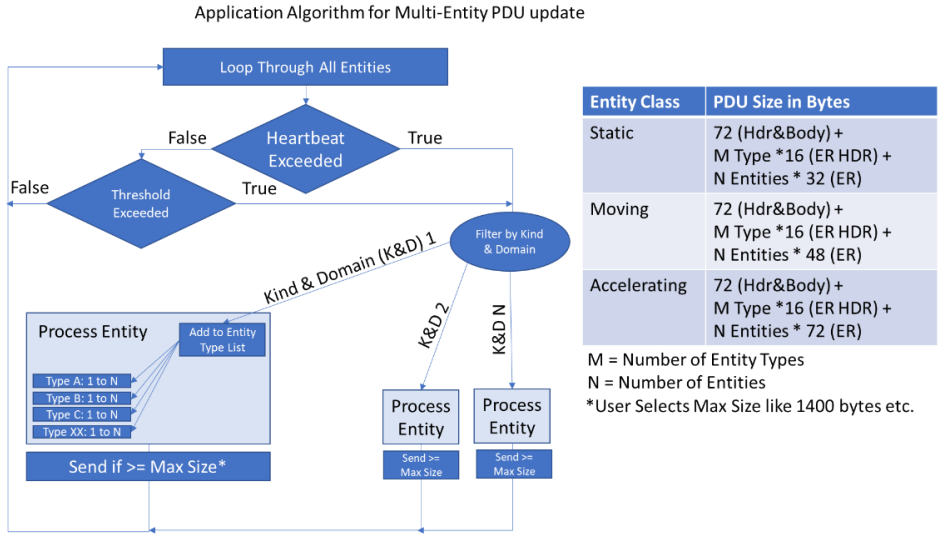

Figure 5 Recommended Algorithm for Multiple Entity PDU

Depending on the number and types of entities, not all Multiple-Entity Entity State PDU buffers will reach the maximum size. When the last entity is processed then the data that exists in the PDU buffers or lists must be sent and cleared in preparation for the next simulation cycle. If entities use a single marking or can use a simple numeric identifier appended to a single marking, such as Viper 1, Viper 2, etc., they will require no further PDUs. If the entities require unique marking fields such as Viper 1, Eagle 12, Dark Star, etc., then the application could use the algorithm in Figure 6 to build a PDU using multiple Marking with Entity Number extension records to define them as necessary. The loop sorts through all entities if the heartbeats are the same, or it may need to sort through the entities based on Kind and Domain or other needed criteria so that entities with the same heartbeats are processed together as a set. Modifying an existing CGF to essentially group entities together and

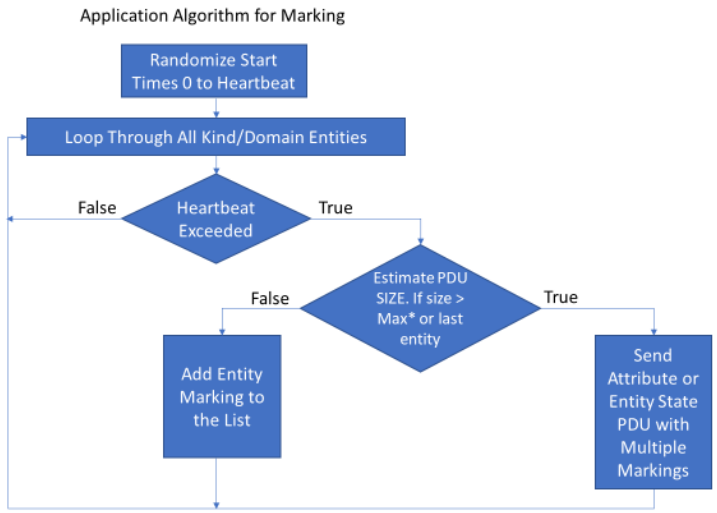

Figure 6 Algorithm for updating Entity Marking for entities sent with Multiple Entity extension records

marshal them into lists until the list is full and then sending the list rather than sending each individual entity will take some modifications but it is feasible. This might mean that in the future, the most common PDU for Entities is not the individual entity in each Entity State PDU, but rather Entity State PDUs with multiple entities in each Entity State PDU, and an associated Attribute PDU where Marking and Appearance are provided on a heartbeat basis.

CHALLENGE 5: SUPPORT FOR HIGHER FIDELITY MODELING OF AESA RADAR AND JAMMING

DIS V8 provides several new extension records that allow for higher fidelity modeling of AESA radars, as well as jamming [see Call (2022) and Call (2023)]. In order to provide that additional level of information, it will mean the simulations that provide that data will also need to improve their fidelity, which will be a challenge. The internal radar models will need to have the ability to model complex beam scans as well as dwell times and signal changes for the various purposes the beam is being used for such as getting a weapons quality track or simply keeping a rough idea of where a target is, while using a minimum amount of energy in order to minimize the chances that the scan will be detected by the targets electronic countermeasures systems.

Many CGF's do not have the ability to model this level of radar behavior. It is most likely that the manned virtual simulators are the ones that will have this type of sophisticated modeling initially, with CGF's adding this capability as time goes on. Jamming including onboard reactive jammers as well as standoff jammers will need to improve to describe the specific jamming methods being used and intended results. Simple noise jamming will not be sufficient. The effects of jamming on entities that are in proximity to reactive jammers can now be modeled leading to more complexity and realistic jamming effects.

CHALLENGE 6: SUPPORT FOR HIGHER FIDELITY MODELING OF IR

Current modeling of IR is generally limited to the usage of Image Generators (IGs) providing images for targeting pods and other visual based IR systems. This is good and needed. However, the amount of information available to the IGs to create accurate imagery is limited to things like is the engine on or off, and is the afterburner engaged or not. There is not any information about the actual temperatures of surfaces, current engine temperatures, inlet or exhaust temperatures, wing leading and trailing edge temperatures, how long an engine has been running, etc. There is no concept of warming up when an engine starts up or when an engine is being pushed hard for an extended period of time, or cooling gradually after being turned off. There is also little to no atmospheric information including clouds that would cause attenuation for a sensor. There is almost no modeling of background temperatures or environmental emitters such as the sun.

There is virtually no support for non-imagery-based IR sensors such as Infrared Search and Track (IRST) systems that track hot points from long ranges rather than using visual contrast in a 3D model like imagery-based systems. DIS V8 is adding more support for weather and other atmospheric modeling as well as more detailed entity information to support both IRST and imagery-based IR capabilities. This will require simulations to begin to track IR information for entities in much more detail and to use the atmospheric information from DIS V8 to affect their IR modeling. This IR status information will add to the amount of DIS traffic for each entity. This will be a very difficult challenge, but one that is critical in order to model modern sensor systems and attack methodologies.

CHALLENGE 7: CONSTANT UPDATES

DIS V7 has been static since 2012. This means that DIS applications have had little requirement to change for over a decade, and contracts have not needed to be modified. DIS V8 will allow new extension records to be defined in the SISO-REF-030 periodically. It is expected for this to be done on an annual basis. This may require DIS applications to be updated on an approximately yearly basis in order to support new capabilities. Contracts will need to consider this constant change. Compatibility among different systems, with different contracts, will need to be managed for compatibility as capabilities are added.

OPPORTUNITY 1: LITTLE-ENDIAN BYTE ORDER

Changing to a little-endian byte order will speed up the processing of DIS PDUs slightly for all little-endian systems because they can skip the current byte swapping step. This is expected to be only a minor improvement, but will simplify coding and processing for most modern hardware and software going forward.

OPPORTUNITY 2: PARTIAL UPDATES

Changing to partial updates opens up many possible new paradigms for updating information possibly resulting in a significant reduction in network bandwidth.

For example, all DIS Entity State updates currently require at least 144 bytes regardless of whether the entity is moving or not. In DIS V8, static entities require 72 bytes (50% saving) without Marking and Appearance, or 104 bytes (28% saving) with Marking and Appearance included every time. Users may choose to not use marking, and they may use the Entity Status bits in the body to provide Deactivated, Frozen, Power Plant On, Mobility Killed, Damage, Is Smoke Emanating, and Flaming information that historically was provided in Appearance. Appearance is only necessary for entities that need additional appearance bits. Marking and Appearance can also be assigned different heartbeats than the entity state PDU allowing them to be updated at less frequent intervals.

Moving entities without Marking require 96 bytes (33% saving) and Accelerating entities without Marking require 120 bytes (17% saving). Markings could be added at heartbeat intervals of 5 seconds, lowering the savings slightly depending on the number of threshold updates probably from around 1 in 3 to 1 in 10 updates for Marking. In DIS V8, marking is variable length so the exact size difference will also depend on the length of the marking string.

The partial updates allow for separate simulation systems to provide information for the same entity without the need to coordinate using an alternate communication channel. For example, one system could be responsible for modeling the motion of an entity and providing Entity State positional updates based on heartbeat and threshold. A second independent system might model the IR characteristics of the entity and provide that information separately using the Attribute PDU without providing any location information or coordinating updates with the system updating the entities position.

Overall, senders can take advantage of the partial update paradigm to send items that are not changing only once every heartbeat interval and send only the data that is actually changing for other updates. This will save bandwidth and allow more entities to be sent over the same network bandwidth.

OPPORTUNITY 3: CPC DEAD RECKONING ALGORITHM

Implementing the CPC DR algorithm will reduce the bandwidth for many entities such as aircraft, ships and ground vehicles that travel in a relatively circular path when turning. The authors tested the new CPC algorithm using the Air Force Research Laboratory (AFRL) Network Integrated Constructive Environment (NICE) Computer Generated Forces (CGF) system. We created twelve different use cases including both ground vehicles and aircraft to compare how the CPC algorithm performs relative to the DIS V7 DRA 4 which uses Linear Velocity, Linear Acceleration and Rotational Velocity. On average the CPC algorithm reduced PDU updates by 18%. The more time an entity spends turning, the more the PDU count will be reduced. If an entity is constantly turning the PDU count will be reduced by 60%. Almost all entities, other than lifeforms, turn in circular paths and will therefore see significant bandwidth reductions. [See Call (2024) for more details].

In DIS exercises, the PDU rates increase significantly as entities begin to interact and maneuver. The CPC algorithm will reduce the amount that the PDU rate increases in these scenarios, leading to the ability to push entity counts higher and closer to the limits of the network bandwidth without risking network overload.

OPPORTUNITY 4: ABILITY TO SEND MULTIPLE ENTITIES IN A SINGLE PDU

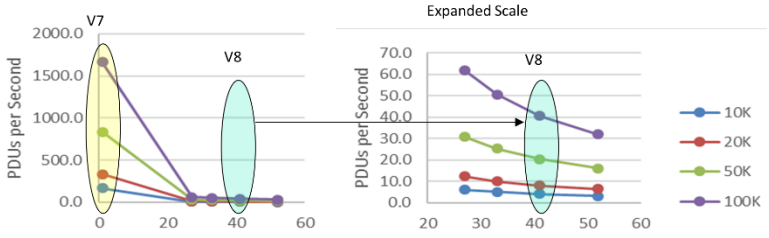

Figure 7 PDUs Per second based on four entity counts (10/20/50/100K)

Sending multiple entities in a single PDU will reduce the number of PDUs needed by the number of entities per PDU. To illustrate this point, to represent 100K static entities in DIS V7 with a 60 second heartbeat requires 1666 PDUs per second (see Figure 7). To send those same entities in DIS V8

using the Entity State PDU with the Multiple Static Entity extension record and ~1452 bytes per PDU, it takes only 40 PDUs per second and the number of bits per second is reduced by 75%. This smaller reduction in bits per second is because while the number of the PDUs decreases dramatically, the size of each PDU increases so the reduction in bits per second is less than the ratio of the number of entities per PDU. The Multiple Entity extension records do have some limitations. The main limitation is that all entities in an extension record must be of the same type (F-16, M1A1, etc.). Entity State PDUs can contain several multiple entity extension records, each with different entity types, but they must have the same Kind and Domain.

The primary intent of the Multiple Entity extension records is to add large numbers of entities efficiently. The algorithm described in paragraph CHALLENGE 4 above can be used to take advantage of this capability to the fullest possible extent. The amount of actual improvement using the multiple entity extension records depends on how many entities are actually added to a single PDU when they are sent. For Static entities, the analysis is fairly straight-forward, as previously described. The only requirement is that there are enough entities in the scenario to populate the PDUs completely. For example, assume a CGF has a 20Hz update rate. Assuming it sends one PDU each update cycle, it can send 20 PDUs per second. If each PDU can hold 40 static entities (4 different entity types case, see) then it can update 40 entities per PDU x 20 PDUs per second = 800 entity updates per second. In order to reach maximum efficiency then there must be at least 800 static entities available to be sent. If there are only 400 entities, then the PDUs would only be half filled. If there were 820 entities then it would require additional PDUs, perhaps up to 2 each frame for the additional single entity. The most efficient updates would all occur with entity counts that are multiples of 800 (800, 1600, 2400, etc.).

Of course, scenarios are not designed for the maximum efficient entity count. Scenario only model what is necessary for the simulation, so the efficiency will vary but will be much better in PDU count than with DIS V7. The PDU reduction factor will be up to the number of entities in each PDU. See Table 1 for a list of how many entities can be packed into a single PDU based on the different class of entities (static, moving or accelerating). The situation for moving and accelerating entities gets even more complicated to do a simple math analysis because the entities will be required to be updated when they break their dead reckoning threshold, which is in turn dependent on how much they are maneuvering at any given time.

These update rates will be different than DIS V7 due to the new CPC DR algorithm. It also depends on how many different types of entities are being modeled. shows just a few examples of the numbers of entities that can be added to a PDU based on the number of different types of entities in that particular update. If there are 4 different types of accelerating entities in the PDU, then it can support up to 18 entities in a single PDU before the PDU reaches the Ethernet MTU size of 1500 bytes total. This is the maximum desired size so that the Ethernet packets do not fragment. For maximum efficiency, this would mean that if there were 10 different types of entities in every Entity State PDU with accelerating entities, there could be 16 entities. Using a 20Hz CGF example, that would mean 16 entities per PDU x 20 PDUs per second = 320 entities if the entities broke threshold once a second and they were all exactly evenly distributed in every simulation frame. That is clearly not going to be the case. Historical DIS V7 data suggests that on average moving entities update at an average Entity State PDU rate of 0.83 PDUs/sec, with a typical maximum PDU rate reaching 2.83 PDUs/sec for relatively short periods of time. This suggests that the for CGF's that are representing lower numbers of entities, they will have the least benefit from implementing the Multiple Entity extension records. For 20Hz CGF's, the maximum reduction would probably be around 320/0.83 = 386 entities to see the most efficiency. If the CGF is 60Hz then it would be three times higher i.e., 386 x 3 = 1158 entities. While these counts or multiples of them would be the highest efficiency, smaller entity counts would still see significant efficiency gains relative to DIS V7. Due the varying nature in time of the maneuvering and threshold breaking, some frames would require more than one, and some frames would not completely fill a PDU. We currently do not have any practical experience with entity counts and percentage of time that PDUs would be completely filled. It is clear, however, that this approach is much more efficient from a PDU count perspective for DIS V7 vs DIS V8 by up to a maximum factor of 42/28/18 for static/moving/accelerating entities respectively.

Table 1 Table of Entity Counts per PDU

Extension Record Type	1 Entity Type	4 Entity Types	10 Entity Types	20 Entity Types
Max Static Entities Count	42	40	37	32
Max Moving Entities Count	28	27	25	21
Max Accelerating Entities Count	18	18	16	14

From a bits per second bandwidth perspective for DIS V7 vs DIS V8, a maximum factor is 4.2/2.8/1.9 for static/moving/accelerating entities, respectively. These are the maximum factors. What the typical improvement factors are requires more practical experience using DIS V8 in practice and will vary depending on the number of types of entities and how the entities are maneuvering.

OPPORTUNITY 5: SUPPORT FOR HIGHER FIDELITY MODELING OF AESA RADAR AND JAMMING

The original emission PDUs were created to model the mechanically scanned radars that existed in the 1990's. Today, however, most new radars are AESA radars. Both CAF DMO and Battlefield Simulation Incorporated (BSI) created custom but conflicting datums to represent AESA radars in DIS V7. Using these messages as a starting point the Advanced Radar and Jammer Tiger Team created a common set of extension records that allow for much more extensive modeling of AESA radars as well as jammers that are standardized and not conflicting. The Tiger Team also added new capabilities [see Call (2022), Call (2023) and Call (2024) for more details].

Having the ability to model AESA radars and complex jamming is critical and necessary to accurately model the current and future battlefield.

OPPORTUNITY 6: SUPPORT FOR HIGHER FIDELITY MODELING OF IR

Internally modeling more IR detail for entities and providing more IR information over the DIS V8 network will allow users to model modern IRST sensors. Visual IR based systems will have much more data to accurately represent entity IR signatures in their 3D models leading to more accurate results and better training.

This is critical in order to be able to operate in a modern combat environment where stealth is at a premium, and passive IR systems including IRSTs are becoming more critical to combat operations. Without IR modeling, the modern battlefield cannot be recreated, resulting in training that does not reflect the expected wartime environment.

OPPORTUNITY 7: CONSTANT UPDATES

Having a standard method available to provide updates to the DIS standard via SISO-REF-030 allows DIS to meet the changing needs of simulations on a timely basis. It also allows for multiple systems to improve in a standard interoperable way rather than implementing custom proprietary extensions to meet user needs. This allows DIS simulators to incrementally improve rather than waiting for a large leap in capability each decade.

CONCLUSIONS

It is clear that DIS V8 will bring many challenges and opportunities for DIS Simulations. Some of the challenges will be relatively minor and easy to overcome, such as the little-endian byte order change. Other changes, such as the multiple entity extension records in Entity State PDUs will require more changes to existing software but will provide significant bandwidth reductions leading to the ability to support larger exercises with many thousands of entities. The increase in entity counts reflects a training requirement.

Probably the most significant challenges and opportunities, however, will be the ability to improve the fidelity of the simulations, including the modeling of the AESA radars, jamming, and IR, as well as providing future opportunities for continued fidelity improvements. Systems can now continuously improve by leveraging the periodic SISO-REF-030 updates rather than waiting for DIS updates every decade or more. These capabilities together will allow us to get closer to training like we fight.

REFERENCES

IEEE. (2012). IEEE Std 1278.1™-2012 *IEEE Standard for Distributed Interactive Simulation – Application Protocols*. The Institute of Electrical and Electronics Engineers, Inc.

SISO. (2023). SISO-REF-010-2023 *Reference for Enumerations for Simulation Interoperability*. Simulation Interoperability Standards Organization.

IEEE. (2015). IEEE Std 1278.2™-2015 *IEEE Standard for Distributed Interactive Simulation – Communication Services and Profiles*. The Institute of Electrical and Electronics Engineers, Inc.

I/ITSEC. (2021). Interservice/Industry Training, Simulation and Education Conference 2021 Paper 21274 *SISO C-DIS Techniques and Performance*. National Defense Industrial Association.

SISO. (2024). SISO-STD-023-2024 *Standard for Compressed-Distributed Interactive Simulation (C-DIS)*. Simulation Interoperability Standards Organization.

SISO. (2014). SISO-STD-013-2014 *Standard for Common Image Generator Interface (CIGI) 4.0*. Simulation Interoperability Standards Organization.

Call, Lance. (2022). *AESA Radar and Advanced Jamming – A CAF DMO Approach*. Simulation Interoperability Standards Organization paper 2022-SIW-006.

Call, Lance. (2023). *Advanced Radar and Jamming in DIS V8*. Simulation Interoperability Standards Organization paper 2023-SIW-001.

Call, Lance. (2024). *Top 10 DIS V8 Improvements.* Interservice/Industry Training, Simulation and Education Conference (I/ITSEC) paper 24439.

Murray, Robert. (2018). *Gen3: What's Up with DIS Version 8*. Simulation Interoperability Standards Organization paper 18W-SIW-028.

Murray, Robert. (2025). *A Simpler Dead Reckoning Algorithm with Better Performance*. Simulation Interoperability Standards Organization paper 2025-SIW-027.

The views expressed are those of the authors and do not reflect the official guidance or position of the United States Government, the Department of Defense or of the United States Air Force.