SISO C-DIS Techniques and Performance

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

Jesse Vater
Cubic Corporation
San Diego, CA
Jesse.vater@cubic.com

ABSTRACT

Live Virtual Constructive (LVC) training systems will need to use Radio Frequency (RF) networks that have significantly less bandwidth than existing wired Local Area Networks. One way to pass more simulation data over such networks or buses with limited bandwidth is to incorporate compression and state management techniques. The Air Force Research Lab (AFRL) developed a compressed DIS (C-DIS) data standard for the Secure Live Virtual Constructive Advanced Training Environment (SLATE). In Feb 2019 AFRL proposed to the Simulation Interoperability Organization (SISO) to develop a more general C-DIS Standard. SISO formed a Product Development Group which spent a year reviewing the C-DIS V1.2 standard, making updates and improvements, and has completed this work.

The SISO Compressed DIS Standard will be balloted for approval in 2021. This paper will review the compression techniques used by the new SISO C-DIS standard (SISO_STD_023-2021) and differences from the original C-DIS V1.2 standard, including a technique adopted from Google. It will also review the compression performance of this version compared to C-DIS V1.2 on several unclassified data sets as well as the SLATE demonstration data.

This paper will provide a solid technical basis for evaluation of C-DIS compression and the applicability of C-DIS to potential use cases. It will show how compression ratios of over 6:1 are possible while using less than 1% of the CPU to perform the encoding and decoding.

ABOUT THE AUTHORS

Lance Call is a Principal 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 C-DIS drafting group editor. He is an IEEE member.

Jesse Vater is a Principal Software Engineer with Cubic Corporation. He graduated with a Bachelor and Master's degree in Computer Science from Binghamton University. His early career involved managing development teams for flight and mission control systems. He now focuses on architecting Live, Virtual, and Constructive training systems. Jesse was the lead software engineer for the SLATE ATD effort and personally implemented the Compressed DIS engine used during the ATD. He is a participating member of the SISO C-DIS product development group.

SISO C-DIS Techniques and Performance

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

Jesse Vater
Cubic Corporation
San Diego, CA
Jesse.vater@cubic.com

History of Compressed DIS (C-DIS)

The military desires to stimulate live aircraft with simulation data coming from other Live, Virtual and Constructive (LVC) systems. This enhances live training by creating a more robust environment than we can afford to create using live aggressor aircraft and hardware based Surface-to-Air Missile (SAM) emulation systems (See ACC, 2021). The required simulation data must be passed between live aircraft and the ground via a long range Radio Frequency (RF) network. This type of RF network has a constrained bandwidth when compared to a ground-based network. We began looking at techniques that could reduce the RF bandwidth requirement in order to maximize the amount of Distributed Interactive Simulation (DIS) data that we could exchange to create the most robust live training possible. We considered several generic compression techniques such as Limpel-Ziv-Welch (LZW) but found that they were not effective for small data sets such as DIS Protocol Data Units (PDUs). We therefore created the initial Compressed DIS (C-DIS) V1.0 specification in late 2013, this progressed to C-DIS V1.2 in 2017. The C-DIS V1.2 specification was implemented as part of the Secure LVC Advanced Training Environment (SLATE) Advanced Technology Demonstration (ATD). The 2018 SLATE ATD demonstration had 16 live aircraft, four virtual simulations, and constructive assets that verified the technology.

The original paper on C-DIS was presented at I/ITSEC 2017 (see Call, L. 2017). After the successful SLATE ATD demonstration in 2018, C-DIS V1.2 was proposed as a Simulation Interoperability Standards Organization (SISO) standard. The SISO C-DIS Product Development Group (PDG) was established at the 2020 Simulation Innovation Workshop (SIW). The SISO PDG has been reviewing C-DIS V1.2 and incorporating updates and new ideas into the standard. This was done to make the standard applicable for more users and use cases and to increase the compression rate. It is expected to become a formal SISO standard (SISO_STD_023-2021) in 2021. SISO C-DIS describes how to compress IEEE1278.1-2012 DIS data (DIS V7). Work is currently in progress on the next generation of DIS (DIS V8). It is expected that the concepts developed for SISO C-DIS will be used to create a new C-DIS version applicable to DIS V8 messages. C-DIS will be the recommended approach for live players using DIS V8. The Live Entity (LE) Information/Interaction protocol will be removed from DIS V8, because C-DIS will better address that use case. It was decided that the C-DIS standards will be managed under SISO rather than becoming part of the IEEE standards in order to be more flexible in making changes and additions to the standard. The SISO PDG will become a Product Support Group (PSG) and continue to support the current C-DIS version, make changes and updates as needed and pursue a C-DIS version for DIS V8.

Concept of Execution

C-DIS uses encoder/decoders to provide compressed communications between standard DIS applications or networks (Figure 1). C-DIS relies on DIS simulation applications to define the PDU update rates based on dead reckoning thresholds, heartbeats, and timeout values. The DIS standard uses these timing mechanisms to minimize network load. C-DIS does not attempt to redefine the DIS timing mechanisms but rather reduces message size.

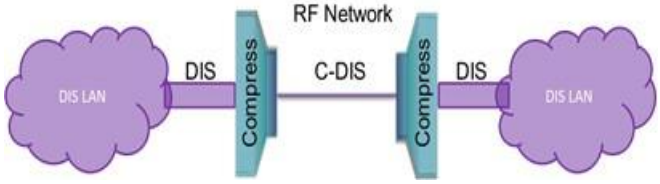

Figure 1. C-DIS encoder/decoder compression concept

C-DIS Compression Techniques

The primary compression technique used in C-DIS is bit packing. Bit-oriented representations of each DIS field are generated in a form that is as small as possible while still accurately representing required values.

C-DIS V1.2 Compression Techniques

All of the techniques used in C-DIS V1.2 were adopted for SISO C-DIS. These techniques are:

1. Eliminate all padding fields, including those required for byte alignment.
2. Reduce enumerations to just the number of bits required for SISO-REF-010 enumerations.
3. Reduce integer value fields to only support the required values.
4. Change floating point fields to scaled integer.
5. Create custom floating point numbers by defining specific mantissa and exponent sizes.
6. Use "Field Present" bit flags at the beginning of PDU's to indicate the presence or absence of fields.
7. Use "Units" enumeration if values require a large span of values.
8. Use 5/6 bit characters in Entity Marking and a length field to indicate the number of provided characters.
9. Lengths refer to bits rather than bytes. Limit lengths to allow for defining a packet up to 1500 bytes.
10. Omit fields that may be easily recreated at the receiver. For example PDU Family.
11. Use Latitude, Longitude, Altitude rather than Earth Centered Earth Fixed coordinates for world position.

While all of these techniques were used, some were applied differently to fields in the SISO version compared to C-DIS V1.2.

Techniques Added in SISO C-DIS

SISO C-DIS more aggressively compresses the Emission PDU. The organization of the PDU was modified to make single lists of unique Fundamental Parameters (Frequency, Power, and Pulse information), Beam Data (Beam Azimuth, Elevation, and Sweep), and Site/Application pairs. This allows these items to appear only once in the PDU but be referenced via index by multiple beams and tracks eliminating redundancy and size in complex emissions. The effect is that smaller basic emitters are slightly larger due to the overhead of the lists, but larger complex emissions can be compressed more than with the C-DIS V1.2 approach.

SISO C-DIS applied the C-DIS principles to the variable records such as the articulated and attached parts that were not covered in the C-DIS V1.2 standard. This provides compression for a wider variety of use cases.

The largest change was the adoption of variable integers referred to as VARINTs. The VARINT concept comes from Google's Protocol Buffers. Nearly 40% of the SISO C-DIS fields are VARINTs. A VARINT is made up of two parts. It has a fixed number of initial flag bits that indicate how many bits of data will follow in the bit stream for the value. If two flag bits are used then it is possible to specify four different sizes of data bits that will follow as shown in Table 1. If one flag bit is used then only two different data sizes are possible. This approach was used for signed and unsigned float and integer values.

Table 1. Unsigned VARINT16 (UVARINT16) Flag and Data Values

UVARINT16 2 Bit Flag	Value Bit Sizes	Max Value
0	8	255
1	11	2047
2	14	16383
3	16	65535

When most of the values in a field are small then a maximum amount of compression can be achieved. If most of the values are large, VARINTs are not a good approach since the flag bits will increase the bits required to the base size of the field; increasing the size rather than compressing it. This approach can support a full range of values while providing compression for smaller values. C-DIS uses Unsigned VARINT8/16/32 and Signed VARINT12/13/14/16/24, where the number indicates the number of data bits. The PDG selects the sizes for each VARINT that would provide the most compression for a DIS field based on analysis of historical DIS recordings and judgments of expected data values. This makes SISO C-DIS applicable to a much wider range of values and use cases, making the standard appropriate for a wider range of users. It also increases compression rates.

Techniques Rejected for SISO C-DIS

Some techniques were considered and rejected by the PDG because the PDG felt they were not a good match for C-DIS at this time. Some of the techniques could be considered for future addition as more users implement C-DIS encoder/decoders, the user community expands, and experience shows what changes would be most beneficial.

The PDG considered implementing Dead Reckoning Algorithm 10 (DRA10) (See Oliver, W. 2018) in C-DIS Encoder/Decoders. DRA10 looks at the movement of the entity and determines which of the existing DR algorithms would be best/most efficient to send. The main advantage is that DRA10 would implement the body based algorithms that project movement in a circular arc rather than the world based algorithms that project a parabola. Many entities, including aircraft travel in approximately a circular arc when turning. This would reduce the number of updates required while aircraft are turning, reducing the required bandwidth. This was the primary motivation for considering DRA 10 in C-DIS encoder/decoders. This breaks the simple one-to-one C-DIS PDU for each DIS PDU relationship. It would require the encoder to generate rotation matrices and do matrix multiplications in order to transform world to body (encoder) and body to world (decoder) coordinates. It would also require that the encoder perform threshold checks and potentially provide heartbeat updates, making the C-DIS data sent essentially asynchronous to received DIS updates. This would make the encoders and decoders significantly more complex. A C-DIS Encoder only has network updates to work with which occur only when thresholds are exceeded, whereas a source application has full frame rates available to perform DRA10 calculations. It's not clear how much compression could be achieved. This concept was therefore rejected for the current SISO C-DIS but may be explored in the future. SISO C-DIS supports passing world and body DR algorithms if implemented by simulator applications. Simulator implementation is believed to be a better solution than implementing DRA10 in the C-DIS encoder/decoder.

The PDG considered applying more efficient audio codecs to audio/radio transmissions. This would involve decoding the audio stream and then re-encoding with a modern codec like OPUS, or using a smaller bit sample or frequency sampling rate to reduce the data rate. Encoding audio twice is not an ideal solution, and there are limitations to changes that can be made to a single sample of audio data. It was decided that audio encoding is better done by a simulator application rather than the C-DIS encoder/decoder. In SLATE and many LVC applications, audio is not passed over the link that would use C-DIS. SLATE voice is transmitted over actual voice radios, and therefore compression of voice for SLATE and likely future LVC applications where C-DIS is a usage candidate is not applicable.

High Level Architecture (HLA) uses a publish-and-subscribe model to limit data. The HLA Run Time Infrastructures (RTIs) manage the state of objects, and only send information that has changed, reducing the amount of data that must be transmitted. When federates initially join the network (or if they leave and rejoin) then they must request from senders full updates for all objects over the network. If data is missed they must also request updates. This leads to high network data spikes as every object state is pushed to the network as quickly as possible when initializing systems, or if a federate drops out of the network and rejoins. In an RF network with aircraft where network joining and dropouts and missed data are likely, this behavior could lead to broadcast storms. This approach is more suitable to a network with few nodes that are relatively easy to coordinate initialization and share a reliable network. LVC will have a network with many nodes that are hard to coordinate and have a network that will likely have some dropouts. There are several possible ways to try to mitigate this effect that are beyond the scope of this paper, but all add complexity and require additional coordination overhead, which requires bandwidth and therefore takes away from the benefits of only sending changes. After much discussion and debate the PDG decided to use an approach based on the HBT_CDIS_FULL_UPDATE_MPLIER (described below) and DIS heartbeat which is less complex, doesn't add administrative network overhead for requests, and avoids network spikes for initialization, late joiners, and missed object updates.

C-DIS V1.2 had the concept of sending Full C-DIS updates periodically, and in between sending partial updates with only things that have changed, known as partial update mode. The update rate was fixed as the DIS HBT_TIMEOUT_MPLIER constant. The partial update mode was not implemented for the SLATE ATD demonstration. Partial Update mode was implemented and used in standalone engineering tests and evaluations of C-DIS V1.2.

SISO C-DIS changed from using the DIS HBT_TIMEOUT_MPLIER to a configurable HBT_CDIS_FULL_UPDATE_MPLIER. This parameter is required for all C-DIS Encoder/Decoders. By default its value is 2.4, the same as HBT_TIMEOUT_MPLIER.

A C-DIS encoder sends a full update on the first PDU, last PDU and once every HBT_CDIS_FULL_UPDATE_MPLIER * DIS Heartbeat seconds. If an entity has a five second heartbeat then by default the C-DIS full update must occur at least once every 2.4 * 5 = 12 seconds. This parameter must be modifiable and the value used is based on a federation agreement. This allows the update rate to be customized for the particular application. If, for example, there are two nodes exchanging data and the network is reliable (like an aircraft bus) then a federation might choose to set the HBT_CDIS_FULL_UPDATE_MPLIER to a very large value, effectively making it like HLA where the full update is sent once and only changes are sent after that. As long as the initialization order is known and controlled, and the network is reliable, this will work without adding the complexity of publish-subscribe messaging, and update requests for missed objects are not necessary since the network is reliable.

The downside of using the C-DIS approach is that late joiners may wait for up to HBT_CDIS_FULL_UPDATE_MPLIER rounded up to the next full number * DIS Heartbeat seconds before receiving the Full Update required to initialize an object. For example, if HBT_CDIS_FULL_UPDATE_MPLIER = 2.4 then round up to 3. If DIS Heartbeat is 5 seconds then 3*5 = 15 seconds may be required before receiving that object when it is first created. For a ground entity with a Heartbeat of 60 seconds up to 3*60 = 180 seconds might go by before receiving that object. This is the maximum time; updates may be received at any time before the maximum. Each Federation can decide the most appropriate value for HBT_CDIS_FULL_UPDATE_MPLIER. The higher the HBT_CDIS_FULL_UPDATE_MPLIER value the more compression, while lower values will require less time to receive the first full updates. DIS Heartbeat values can also be adjusted as needed.

This approach scatters full updates across time rather than requiring many full updates simultaneously. This distributes the network load over time and reduces network spikes, which can be critical for networks with limited bandwidth.

C-DIS Limitations

The following list outlines known C-DIS limitations:

1. C-DIS supports 64 simultaneous DIS exercises (0-63) compared to 255.
2. Scaled Integer values have minimum and maximum values as well as less precision than floating points.
3. In Emission PDUs the Emitter Systems number of unique fundamental parameters within a PDU is limited to 31, beam parameters 31, and up to 63 Site/App pairs. There is no direct equivalent limitation in DIS.
4. C-DIS will not compress the audio portion of Signal PDUs.
5. C-DIS will not compress user DATUMs. (Users could apply C-DIS techniques to custom Datums).
6. C-DIS doesn't currently support the Logistics, Minefield, Entity Management, and Simulation Management with Reliability or Information Operations PDU Families.
7. Entity Markings only support capital letters. Unsupported characters will be replaced with an asterisk.

Table 2 lists the precision and maximum values that are able to be represented in C-DIS

Table 2. C-DIS Approximate Precision and Maximum Values

Item	Approximate Precision	Maximum Value
Latitude/Longitude	8.3819×10^{-8} deg (0.93 cm)	+-90/+-180 deg
Altitude	1 cm/Dekameter	+8388607,-8388608 (-8388608 special case to indicate ECEF x=0,y=0,z=0)
Velocity	0.1 m/sec	+-3276.7 m/sec (6369 Knots)
Acceleration	0.1 m/sec/sec	+-819.1 m/sec/sec (83.5G's)
Angle	0.0439 deg	+-180 deg
Angular Velocity	0.35 deg/sec	+-720 deg/sec
Emission Frequency/Emission Frequency Range/Transmit Frequency Bandwidth	5 significant decimal digits	131071×10^{15}Hz
Transmitter Frequency	7 significant decimal digits	16777215×10^{15}Hz
PRF	100Hz	6553.5 KHz
Pulse Width	4 significant decimal digits	16383×10^{3}µsec
Power	1 dBm	255 dBm
Time	107 µsec	1 hour (timestamp)

C-DIS Strengths

C-DIS will compress DIS stateful PDUs (Entity State, Emission, Transmitter, Designator, and IFF) well. These PDUs will make up the vast majority of network bandwidth unless a network contains many voice radios. We refer to event PDUs (Fire, Detonate, etc.) and these stateful PDUs without radio and Link16 as the "LVC" PDUs since they will be passed up to live aircraft for LVC operations. C-DIS compresses these LVC PDUs with compression ratios between 2.5:1 and 6:1. The new SISO C-DIS improves the compression for a much wider array of data values than C-DIS V1.2 which makes it applicable to a wider variety of applications and usages rather than being more tailored to the expected SLATE data values.

Implementing C-DIS encoder/decoders is relatively straightforward and can be accomplished by each interested party independently. There is no middleware or proprietary software required which avoids security issues, especially for foreign governments.

C-DIS Performance

Compression results for C-DIS V1.2 (shown as V1.2) and the SISO C-DIS (shown as SISO or V1.4 SISO) are shown in Figure 2, Figure 3 and Figure 4. A "Combined Scenario" was created with 5 minute samples of six different unclassified scenarios with different characteristics for usage in evaluation of C-DIS compression. It has portions with large numbers of static entities as well as scenarios with only aircraft, weapons, and emitters. "All" refers to all LVC PDUs (No voice or Link16) while "ES" is the abbreviation for Entity State PDUs. "Emit" means Emission PDUs, and "No Data" means Data PDUs were filtered out for that test. "Full" indicates that C-DIS was operating in the Full update mode where every update contains all data. "Partial" indicates that C-DIS was operating in the Partial Update mode where only data that changed would be sent, except for periodic Full Updates based on the HBT_CDIS_FULL_UPDATE_MPLIER (set to 2.4 for all of these tests). Figure 2 and Figure 3 show the compression ratios observed on the left part of the figure and then the sizes of the PDUs (in bits) on the right side of the figure. Both the compression rates and message sizes are important in the case of SLATE where the Fifth Generation Advanced Training Waveform (5G-ATW) used has a 768 bit message size. Messages that are larger than 768 bits would require fragmentation, effectively doubling the bandwidth usage, so tracking message size is as important as the compression ratio for this use case. Figure 2 shows that SISO C-DIS outperforms C-DIS V1.2 especially when operating in the partial update mode where compression increased from 2.6:1 to 5.5:1 overall. Both C-DIS V1.2 and SISO keep nearly 100% of the messages less than the 768 bit 5G-ATW message size so that messages are not fragmented. With SISO operating in Partial mode just over 90% of the messages are only 384bits, which would allow for packing multiple messages into a single 5G-ATW packet, or allow creating a 5G-ATW message that uses the same Forward Error Correction (FEC) code rate as the current message, but uses half of the data pulses. This would allow 320 msgs/sec rather than 200 msgs/sec from a single SLATE 5G-ATW transmitter.

Figure 3 shows the summary of the data compression from the SLATE demonstration. Note that the increase in compression rate from Full to Partial Update mode is less than in the Combined Scenario. This is mostly because the SLATE scenario had few static entities. Also, the C-DIS V1.2 approach to emitters performed better than SISO in the full update mode, but had essentially the same compression rate in Partial update mode while SISO C-DIS improved from 2.5:1 to 4.2:1 beating C-DIS V1.2 in Partial mode. This is also reflected in the packet size graph which shows a large number of PDUs around the 500 bit size for all modes except for the SISO Partial update mode, which is closer to 300 bits. Another observed change is that the number of different message sizes increases from around 150 for C-DIS V1.2 to over 800 when using SISO C-DIS. This is because the VARINTS take advantage of a number of small improvements to each data item that result in a wider variety of message sizes. Compression is being applied at a finer grain level in SISO, leading to better compression overall.

Figure 4 shows the PDU sizes for two PDUs that typically make up a large percentage of the DIS traffic. It's critical that C-DIS compress Entity State and Emission PDUs well in order for it to perform well in most applications. For SLATE these two PDUs make up 73% of the PDUs and 64% of the data bytes. For the Combined scenario they make up 87% of the PDUs and 94% of the data bytes. SISO C-DIS out performs C-DIS V1.2 in compressing both Entity State and Emission PDUs when used in the Partial update mode.

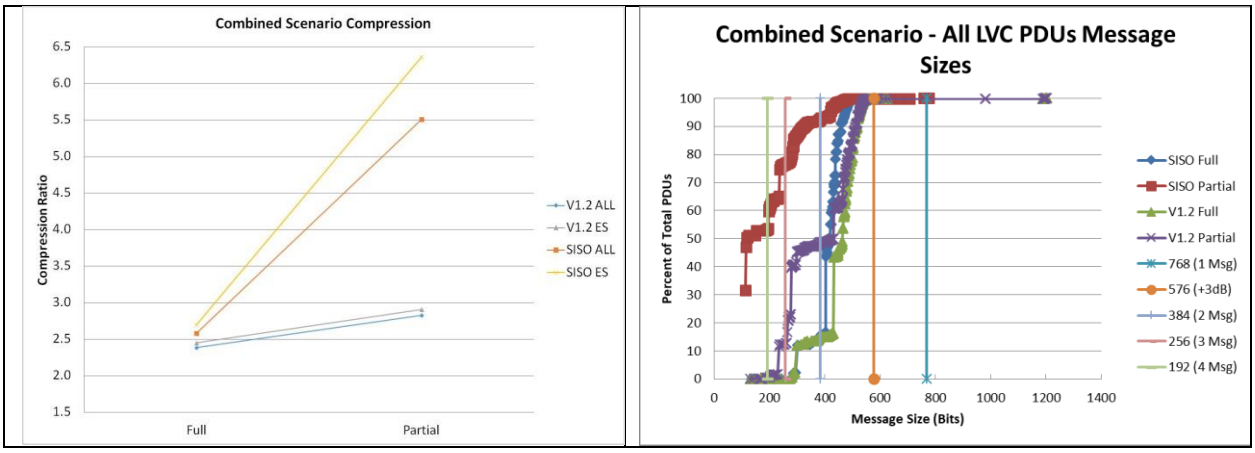

Figure 2. C-DIS Compression Rates and PDU Sizes for the Combined Scenario

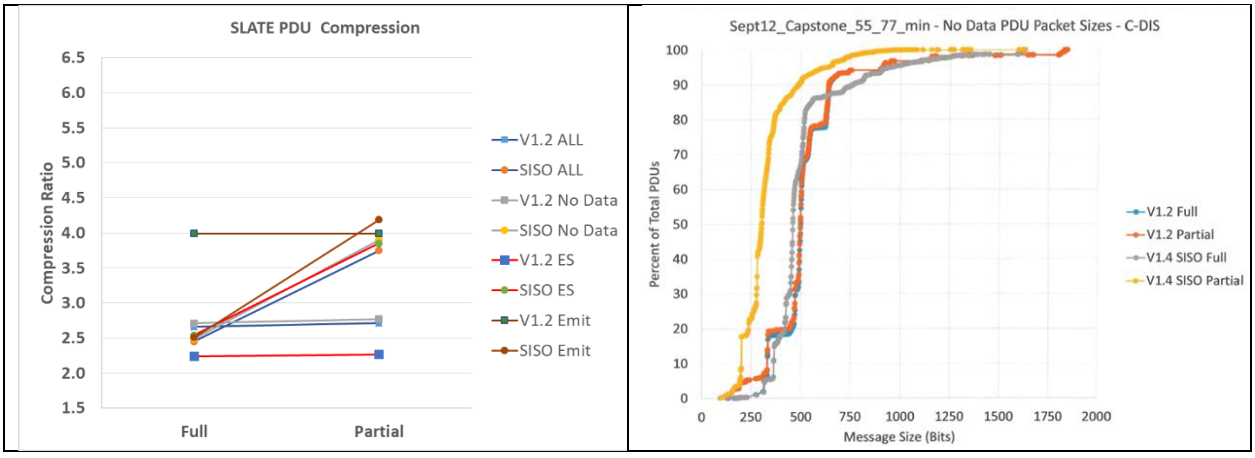

Figure 3. Compression Rates and Summary of All LVC PDU Sizes for the SLATE Capstone Demonstration

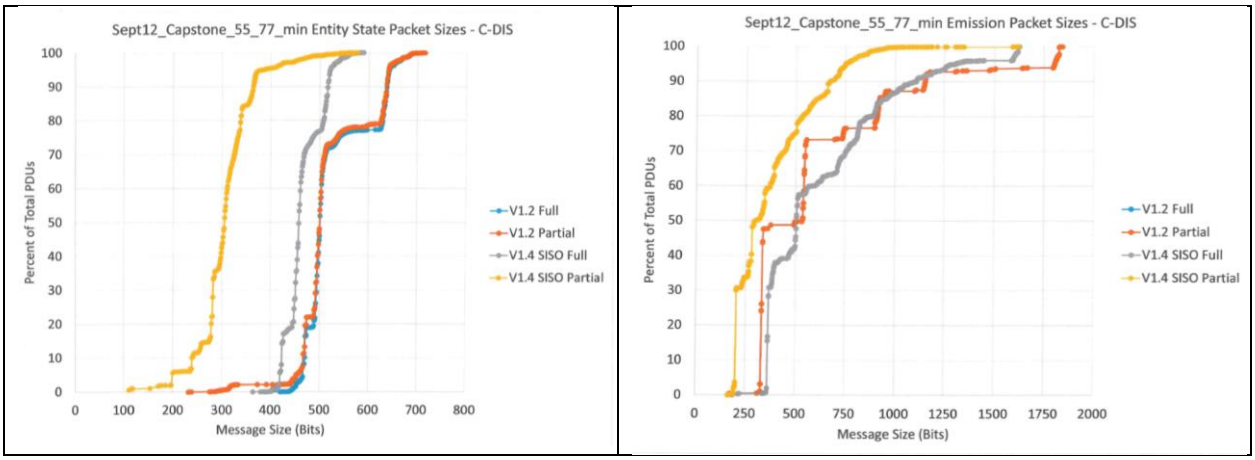

Figure 4. C-DIS PDU sizes for Entity State and Emission PDUs

An analysis was performed for two virtual fighter aircraft during a 20 minute engagement with up to 7.7 G's of maneuvering to determine if there were any differences in the positional errors observed when using DIS or C-DIS. Figure 5 shows that there is little difference in positional errors. Positional error was determined by comparing each DIS network update to the dead reckoned position using velocity and acceleration (Dead Reckoning Algorithm 4) provided in the previous network update.

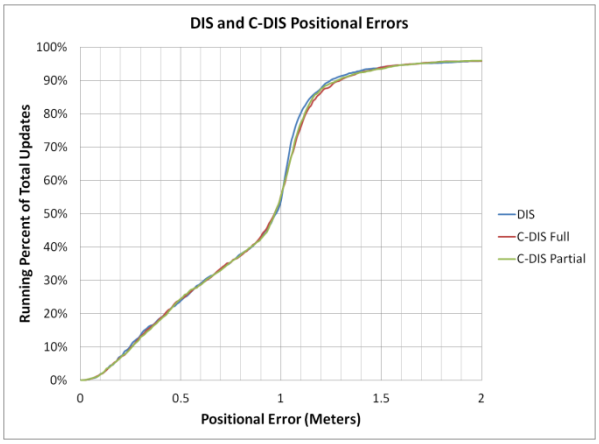

Figure 5. Histogram of Positional Errors Observed for DIS and C-DIS from a Virtual Fighter During an Engagement

C-DIS shows slightly more errors at 1 meter and slightly less at 1.1 meters. This analysis accounts for all C-DIS precision limitations related to position including time, position, velocity, and accelerations.

Implementation Tasks

C-DIS is a bit-oriented standard that requires turning DIS PDUs into a buffer of bits. This requires more bit manipulations than are typical in modern software. We created a class to handle these manipulations so that the programmer could focus on higher level tasks. This is highly recommended. On the sending side, the DIS data must be encoded into the proper C-DIS size number of bits for that field. On the receiving side, the C-DIS data must be decoded and turned back into a standard DIS packet that is then sent to the receiving network. There are four different cases for encoding. The simplest is when encoding an unsigned integer value such as an enumeration where a fixed number of bits can simply be added into the C-DIS stream buffer. The second requires that a data value be scaled first and then the result can be added into the C-DIS stream buffer. The third requires that the value first be normalized (example 380 degrees changed to 20 degrees) between the minimum and maximum values before scaling and adding bits into the C-DIS stream buffer. The fourth is to encode a VARINT which requires data first be normalized, then the value must be checked against the number of sizes allowed for that VARINT (two or four) including the sign bit so that the VARINT flag bits can be set which indicates how many data bits will follow, and finally the data can be scaled and added to the C-DIS stream buffer. The only data that requires more complex operations is the Earth Centered Earth Fixed (ECEF) world coordinates that must be converted to Latitude, Longitude, and Altitude before encoding occurs for each individual field.

There are four common decoding cases. The first is reading the bits from the C-DIS bit stream into a standard size DIS Field like a 16 or 32 bit integer. The second is to read in the C-DIS bits and applying a scale factor when assigning to the DIS field. The third case is to read in the C-DIS bits, extend the sign bit, then apply a scale factor and assign to the DIS field. The fourth is to read in a VARINT by reading the flag bits which indicate how many additional bits should be read, reading the data bits and extending the sign if necessary, and applying scaling when assigning to the DIS field. The only data that requires more complex manipulation is reading the individual Latitude, Longitude and Altitude fields and then using those values to convert from Latitude, Longitude, Altitude to ECEF to assign to the DIS XYZ fields.

Implementation Experience and Lessons Learned

Our team developed two different C-DIS Encoder/Decoders. One for evaluation and analysis purposes and one for the SLATE pod and ground station. For evaluation and analysis the team has several DIS applications and analysis tools that use JAVA, so we implemented a C-DIS Encoder/Decoder using JAVA leveraging our existing DIS libraries. The code consists of roughly 15K lines of code for DIS representation, 15K lines of code for C-DIS Encoding/Decoding, and 10K lines of code for a small GUI and miscellaneous utilities, for a total of approximately 40K lines of code. Lines of Code in this case include all lines in the .java file including comments, so actual lines of code would be smaller. For the Encoder with 121 entities our I7-5930 @3.5HGz Intel workstation CPU shows 0.1-0.3% usage and 50MB Memory before garbage collection. With 3500 entities, the CPU is 1 to 2% with 75MB memory before garbage collection and 15MB after garbage collection to encode data. Measurements were made with the NetBeans profiler. The Encoder/Decoder uses relatively few computer resources, but impact will increase as the number of messages encoded/decoded increase.

Java has no unsigned data representations, so care must be taken to handle signed and unsigned values. We use 32 bit integer values to handle unsigned short values to ensure that sign bits don't get extended, and 64 bit longs to deal with 32 bit unsigned values. Sometimes masking is required. We created a utility class that handles adding bits into the C-DIS stream buffer and reading bits from the C-DIS buffer. This code deals with the ENDIAN representation, bit order and values that cross over byte boundaries, and marshalling or parsing the bit buffer. Once this utility is written it

makes adding or reading bits from the C-DIS stream easy. A VARINT utility was created to make encoding or decoding VARINTs a single method call for the appropriate size VARINT.

Debugging during development can be challenging since data are bit values. One approach used was to run the encoder on known data, recording the data, then playing that recorded data back through the decoder in the debugger. Following along as each field is decoded allows you to identify exactly how far you are able to successfully encode/decode data. Test values that cover min/max cases were used to verify proper operation.

Original DIS recordings were compared to recorded data that had passed through and encode/decode cycle in order to verify that the original data and translated data were within the precision of C-DIS. We used an in house tool called Packet Diff to compare these two recordings. This is challenging since floating point values will be slightly different due to the precision limitations of C-DIS. This requires that fields be within a specific range of values that are within the precision of C-DIS, but each particular field has different precision making this difficult. Angles such as -10.0 and 350.0 are equivalent even though the values are very different which makes comparisons difficult as well. Packet Diff highlights changes making it easier for a user to go through and verify that fields are equivalent. Some threshold adjustments were added to the tool to only flag data outside of a range as being different, so that PDUs could be evaluated more quickly for changes.

The SLATE ATD effort provided valuable insight into the application of C-DIS in an operational environment. There is a delicate balance between loose and tight coupling of the application-layer simulation protocol (i.e. C-DIS) and the underlying transport-hardware layers that is worth recognizing. The SLATE ATD utilizes the government-owned 5G-ATW. The communication of messages over the 5G-ATW is purely implemented per the C-DIS V1.2 standard for the SLATE ATD. DIS/C-DIS updates occur whenever a threshold is broken rather than at fixed regular time intervals. This makes DIS/C-DIS data asynchronous, therefore DIS/C-DIS is most effectively used with a waveform that complements the ad hoc nature of DIS/C-DIS updates. Preferably, payload frame size can be adapted to package a single, nominally sized C-DIS PDU.

For the SLATE ATD, the pod and ground station C-DIS implementation was designed as a portable C++11 library that integrated with open-source DIS libraries. This C-DIS engine operates in both a Windows and embedded Linux environment. Early design decisions were made that favored a software implementation over firmware primarily because floating-point math is a requirement of C-DIS and not native to firmware constructs. The entire C-DIS V1.2 library is under 5000 source lines of code (SLOC).

One of the benefits of having an open-source standard such as DIS as the system's core simulation protocol is the opportunity to use accompanying analysis tools such as Wireshark DIS dissectors. Since at the time of SLATE ATD C-DIS was in its infancy, it was logical to develop a Wireshark C-DIS dissector that decodes C-DIS messages for real-time analysis.

There are several programming key lessons learned from the SLATE ATD experience developing and executing the C-DIS protocol:

a) Typecasting is plentiful. Be mindful of type limitations for your platform. Developers must be cognizant of compiler limitations regarding floating-point casts to integer types.
b) Invest in bitstream constructs. A preferable approach is to implement abstract layers including an underlayment to handle bitstream manipulation and byte-ordering. C-DIS is a bit-oriented protocol that requires significant bit manipulation through masking and logical operators. Be mindful of unaligned memory accesses on processors. Not all processors support such access.
c) Design twice, Copy once. Determine the best approach to minimize memory copies, and memory accesses in general. When decoding and encoding new messages, performance degradation is more likely to come from memory transactions than any other aspect of C-DIS. Consider a memory access laddering approach to perform the largest access possible and step down to smaller accesses if required.
d) Standard application of scaling factors. To avoid common programming errors, it is recommended to define C-DIS scaling factors such that values are multiplied with the scale factor when performing encoding and divided when performing decoding.

The overall CPU and memory utilization of the airborne subsystem's processor module during the ATD live flight events was captured. The C-DIS function was estimated to contribute less than 1% of the CPU utilization in the embedded application.

Non-Proprietary

The SISO C-DIS standard is an open non-proprietary international standard that requires no middleware or other software. It can be implemented within a short time frame by personnel familiar with the DIS protocol and use few computer resources. Different vendors can implement compatible encoder/decoders like different vendors can implement compatible DIS products. This is especially attractive for foreign governments in classified settings that would wish to compress data and exchange it with other nations without having to buy middleware from the foreign country. This reduces the risk of backdoors or other malware in software obtained from a foreign source.

Encoder Interoperability

AFRL and Cubic exchanged recorded C-DIS data to verify compatibility between their encoder/decoders. The AFRL decoder was able to decode all Cubic Encoded DIS Entity State, Detonation, Action Request, Emission and Signal PDUs with no errors on the first try. An AFRL coding error was identified that fixed the Fire PDU. One error was found in the Transmitter PDU that identified -270db which C-DIS doesn't support as 255 db. Overall the two implementations were very compatible with small bugs that were able to be quickly fixed. No errors related to flaws in the standard were identified. This demonstrates that multiple vendors working independently are able to create compatible C-DIS encoder/decoders. Interoperability testing between implementations is recommended in order to ensure proper operations. This would be a similar effort as connecting two different DIS applications for the first time.

Compression Techniques Compatibility with Physical Transport

Compression of messages is only one piece of a larger system. The system performance depends on the combination of all of the pieces. For example the 5G-ATW Radio Frequency (RF) waveform uses a 768 bit user data message. Compressing data from 768 bits to 500 bits changes the compression ratio, but still takes up a single user data message in the RF resulting in no benefit to the overall system. However, if the data can be compressed to 384 bits = ½ of the current size, this opens up some opportunities to pack multiple messages into a single 5G-ATW packet or to define a shorter 5G-ATW message. A shorter message uses half the number of words at the same FEC rate, leading to an increase in message per second rate (320msg/sec from 200msg/sec). That's up to a 60% increase in the data that can be sent leading to the ability to model more aircraft, more radars, more weapons, and more complex and higher fidelity interactions. The compression algorithm and the physical transport mechanism and characteristics must be taken into account together to improve or analyze performance. Compression techniques and physical transport need to match up well in order for the overall system to be efficient and effective.

SLATE ATD 2018

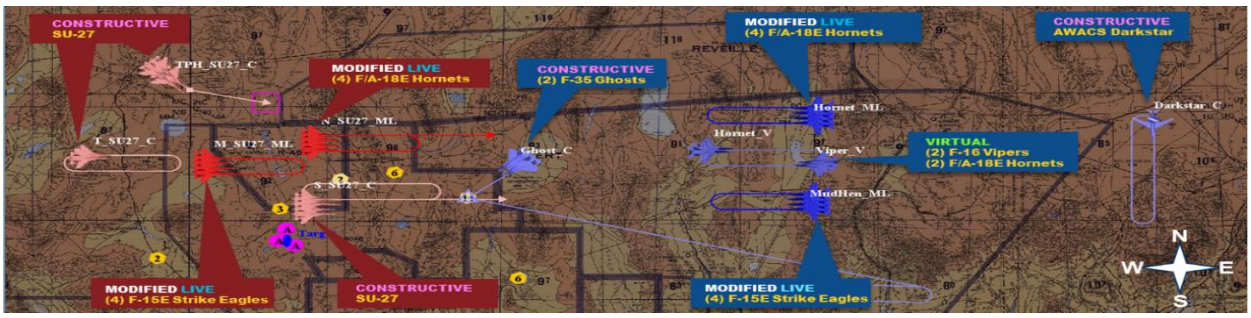

Figure 6. SLATE 2018 Capstone LVC Demonstration

The largest usage of the C-DIS standard to date was the 2018 SLATE ATD at Nellis AFB shown in Figure 6.

This demonstration featured eight live F-15E aircraft and eight live F/A-18 aircraft with modified Operational Flight Programs that allowed them to process, display, sense, target, shoot, and kill Live, Virtual and Constructive assets. There were four manned Virtual cockpits, 13 constructive air, eight SAMS, three AAA, and four ground vehicles. Air-to-Air, Air-to-Ground, and Surface-to-Air weapons and sensors were all modeled. All data other than Voice or Link16 were sent over the 5G-ATW data link in C-DIS V1.2 format in Full Update mode using the pod/ground station C++ C-DIS encoder/decoder. As shown in Figure 3, about 94% of the analyzed DIS messages would be compressed to less than 768 bits, preventing them from being fragmented into two 5G-ATW messages. If standard DIS messages

were used then only 25% of the messages would fit in one message and 75% would be fragmented, overloading the network much sooner than with C-DIS. Using compression allowed more entities to participate and usage of more weapons and sensors as well. In short, C-DIS made the complexity of the demonstration possible.

NAVAIR SLATE Maturation 2021

Naval Air Systems Command (NAVAIR) is currently conducting a SLATE technical maturation effort culminating in a flight demonstration in late 2021. C-DIS is identified as a Core Enabling Technology (CET) required to meet the objectives of several program focus areas. The Navy is modifying their pod and ground station encoder/decoder to support the new SISO C-DIS standard, to include support for Partial Update mode for Entity State and Emission PDUs. This will be the first opportunity to verify that the SISO C-DIS Partial Update mode works properly in an operational environment. The demonstration will include live F/A-18 aircraft, virtual 4th and 5th generation air platforms, virtual surface platforms, and a variety of constructive threats as shown in Figure 7. The system will leverage the Navy Continuous Training Environment (NCTE) to perform distributed LVC and demonstrate advanced tactics, techniques, and procedures (TTPs). The Navy's Next Generation Threat System (NGTS) will manage the constructive participants in both the tethered and untethered environment and use SISO C-DIS to communicate their simulation data across the 5G-ATW Link.

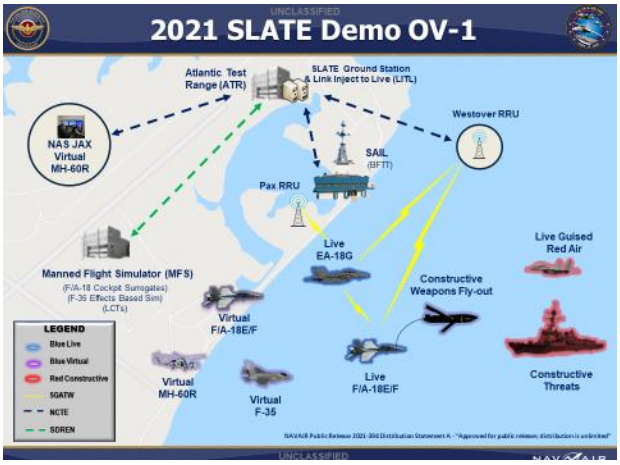

Figure 7. NAVAIR SLATE Maturation Demonstration

Conclusions

The SISO C-DIS implementation introduces some novel techniques and incorporates best practices from similar industry-relevant data protocols. SISO C-DIS provides a significant improvement over C-DIS V1.2 used for SLATE. It is both more effective and applicable to a wider range of use cases, making it more appropriate as a DIS compression approach for a wide range of users. The amount of compression achieved will depend on the actual data. Partial Update operation can significantly increase the compression amount, but requires more complex stateful encoder/decoders. SISO C-DIS is a relatively straightforward compression of data for fast execution, low computer requirements and simple implementation. It is flexible and can be adjusted to user specific applications by adjusting the HBT_CDIS_FULL_UPDATE_MPLIER to meet the needs of the specific federation. It is an open Non-Proprietary International standard that requires no middleware, eliminating many potential security concerns. SISO C-DIS is an effective and standard way to compress DIS data for networks or busses that have low bandwidth.

REFERENCES

IEEE. (2012). IEEE 1278.1-2012: IEEE Standard for Distributed Interactive Simulation: Application Protocols. New York, NY: Institute of Electrical and Electronics Engineers (IEEE).

SISO. (2019). Enumerations for Simulation Interoperability SISO-REF-010.1-2019. Orlando, FL: SISO

Systems, N. G. (2018). Combat Air Force Distributed Mission Operations, Operations, and Integration (CAF DMO O&I). Dayton: Northrop Grumman Information Systems

Call, L. (2017). Compressed DIS. Orlando: IITSEC.

Oliver, W (2018). Performance of Dead Reckoning Algorithms Across Technology Eras: 18F-SIW-011

ACC. (2021). Draft ADARI CDPv10 May 21: ACC