

Designing FOMs For Performance

Bob Murray

Adam Faier

The Boeing Company

P. O. Box 516

Saint Louis, MO 63166

314-234-0178, 314-232-0316

bob.murray@boeing.com, adam.j.faier@boeing.com

Keywords:

HLA, RTI Performance, Scalability, RPR FOM

ABSTRACT: *Conventional wisdom has held that there is a significant RTI run-time cost per attribute sent, that one large attribute is more efficient than several smaller ones. Therefore, the design of a FOM can directly affect federation performance. But, how significant is the cost? Is it high enough to consider redesigning existing standards such as the RPR FOM?*

This paper gives the results of a series of tests that reveal the affect of FOM attribute design on RTI performance. Specifically, the tests compared the processing overhead and network bandwidth used for a varying number of attributes. The five RPR FOM spatial attributes (location, velocity, etc.) were used as an example for testing. Tests were run on both the DMSO NG and MaK RTIs.

Results showed that sending five separate spatial attributes takes 25% more processing time and 10% more bandwidth than a single attribute for both the DMSO and MaK RTIs.

Multiple attributes have some benefit for bandwidth efficiency because they easily allow the minimum amount of data to be sent. When combining multiple attributes, use of a variant record in the merged attribute is often required to maintain this bandwidth efficiency. However, variant records greatly increase software coding complexity. The paper concludes with a recommendation for merging the five RPR FOM spatial attributes in a way that uses variant records to minimize bandwidth and also keeps the variant data structures similar to reduce coding complexity.

1. Introduction

Two limits to the scalability of HLA (High Level Architecture) federations are RTI (Run-Time Infrastructure) processing time and network bandwidth. It is a well-known fact that networked data is transferred more efficiently (i.e. using less processing time and bandwidth) in fewer large packets than it is in many small packets. The amount of per-packet software processing greatly hinders performance if data is broken up into small units. This philosophy has been applied to improve performance when using the RTI by putting as many attributes as possible into a single set to be updated together. Each set generally results in a single network packet and thus is handled more efficiently.

However, it has been discovered more recently that the RTI has a significant per-attribute processing cost

independent of whether the attributes are associated in a set or not [1]. There is nothing the user software implementation can do to eliminate this overhead; it is strictly a matter of how the attributes are designed in the FOM (Federation Object Model).

A case in point is the RPR (Real-time Platform Reference) FOM [2]. It was originally designed more toward the principle of using many small attributes rather than fewer larger attributes. The thinking was that small attributes could take advantage of the RTI's ability to send only the attributes that have changed and thus save bandwidth. The goal is, of course, to improve scalability with some fixed set of resources (bandwidth, CPU processing power).

Another way to save bandwidth is to use attributes with variant records, that is, a variable data format. However, a variable format is difficult to deal with, both in getting

the format defined exactly in the FOM and in the user software that implements the FOM. The RPR FOM designers chose to avoid variant records wherever possible for this reason, and because they are not explicitly permitted by the 1.3 RTI specification. IEEE 1516 does provide explicit support for variant records.

Real-world use of the RPR FOM has shown that the attribute breakup went too far. It has already been decided to combine several radio communication attributes into a single larger attribute because they are always sent together and the processing overhead is greatly reduced. The Tasmanian Devil project has recommend combining some emissions attributes for the same reason.

This paper proposes that the spatial attributes should be combined in a similar manner. It will discuss three solutions for doing so, then narrow the field down to two practical solutions (one with variant records and one without), and then give the results of tests that were run to measure the performance improvement. The results show that the current solution of using separate attributes takes as much as 25 to 30% more processing time than a single larger spatial attribute. A 10% saving in bandwidth can also be achieved by reorganizing the spatial attribute.

2. FOM Design Considerations

There are three primary tradeoffs to be considered when designing a FOM for performance: CPU processing time, bandwidth, and ease of implementation and use.

2.1 CPU Processing Time

The amount of CPU time spent executing code per byte of data sent or received is the most important factor to be addressed.

Research has shown that sending or receiving one large attribute is more efficient than several smaller ones, due to the necessary per-attribute computation. However, using fewer, large attributes removes the fine-grained distribution of attributes and resulting potential bandwidth savings.

2.2 Bandwidth

Each attribute sent in an update has some amount of overhead associated with it, needed (at least) to indicate what attribute is being updated. By reducing the number of attributes needed to transmit a block of related data, there can be a potentially significant bandwidth savings.

Additionally, every byte less of overhead is one byte less that needs to be processed by the sender and receiver.

2.3 Ease of Implementation and Use

In addition to the performance and bandwidth advantages provided by a given FOM design, the ease of implementation and use from the developer's perspective must also be considered. Placing more data into fewer attributes can lead to very complicated complex datatypes that require many lines of code to manipulate. Variant records especially can require many similar lines of code, which can be error prone.

3. FOM Attribute Formats Tested

Values that are frequently updated and almost always updated at the same time are good candidates for performance optimization. Once a set of values has been defined, the next step is to determine the best way to represent them in the FOM.

The "spatial attributes" of the `BaseEntity` class in the RPR FOM 1.0 were chosen for our experiments because of their central role in a simulation and their high update rate. We defined the set of spatial attributes to include the five that contain spatial information:

- `AccelerationVector`
- `AngularVelocityVector`
- `Orientation`
- `WorldLocation`
- `VelocityVector`

Additionally `DeadReckoningAlgorithm` and `IsFrozen` were included in the set, as these values are vitally important to the interpretation of the other five. We examined four different arrangements of this spatial data.

3.1 Multiple Attributes

The Multiple Attributes format, which is used in RPR FOM 1.0, specifies spatial data using seven attributes, as shown in Table 3.1. This format is used as the baseline for comparison.

Attribute / Data Type	Cardinality
AccelerationVector AccelerationVectorStruct	1
AngularVelocityVector AngularVelocityVectorStruct	1
DeadReckoningAlgorithm DeadReckoningAlgorithmEnum8	1
IsFrozen boolean	1
Orientation OrientationStruct	1
WorldLocation WorldLocationStruct	1
VelocityVector VelocityVectorStruct	1

Table 3.1 RPR FOM's Multiple Attributes Format

3.2 Fixed-Form Attribute

The Fixed-Form Attribute format uses a single attribute with a complex datatype. The fields of this complex datatype are all of cardinality 1, hence fixed-form. For some Dead Reckoning modes, this means that unnecessary data is transmitted. This format trades some potentially wasted bandwidth for very simple processing on the sending/receiving sides.

In Dead Reckoning modes DRM_FPW and DRM_FPB (2 and 6, respectively), both the acceleration vector and angular velocity information is unneeded, resulting in 24 bytes of wasted bandwidth. In modes DRM_RPW, DRM_FVW, DRM_RPB, and DRM_FVB (3, 5, 7, and 9 respectively), either the acceleration vector or the angular velocity is unused, resulting in 12 bytes of wasted bandwidth. In DRM_RVW (4) and DRM_RVB (8), all of the data is used. In the case of DRM_STATIC (1), where only location and orientation are used, the overhead is 36 bytes, but this is offset by the fact that static entities are not moving and these values are not updated often.

Table 3.2 shows the fields of the complex datatype FixedSpatialStruct, which was used to evaluate this format. Orientation was put before WorldLocation to preserve alignment and eliminate the need for extra padding.

Field Name / Data Type	Cardinality
DeadReckoningAlgorithm DeadReckoningAlgorithmEnum8	1
IsFrozen Boolean	1
pad1 Char	1
pad2 Char	1
Orientation OrientationStruct	1
WorldLocation WorldLocationStruct	1
VelocityVector VelocityVectorStruct	1
AngularVelocity AngularVelocityVectorStruct	1
AccelerationVector AccelerationVectorStruct	1

Table 3.2 FixedSpatialStruct complex datatype

3.3 Variable-Length, Fixed-Form Attribute

The Variable-Length, Fixed-Form Attribute format uses a single attribute with a complex datatype. The fields of this complex datatype are fixed-form, as in the Fixed-Form Attribute case, but the length of the data sent can vary depending on the Dead Reckoning algorithm in use. This is done by representing the fields not used by every mode (VelocityVector, AccelerationVector, and AngularVelocity) as variant records, with a twist. Variant records in RPR FOM are typically assumed to have mutually exclusive discriminates. Our processing code, however, permitted non-mutually exclusive conditions for determining the format of the received data.

The approach requires knowledge of only one in-memory representation of the variant data. Additionally, wasted bandwidth occurs only in Dead Reckoning modes DRM_FVW and DRM_FVB (5 and 9, respectively), which use AccelerationVector but not AngularVelocity. In this case, there are 12 wasted bytes.

This format is included only for completeness, as the mechanism it uses violates both the spirit of the RPR FOM definition of a variant record and the behavior for variant records as specified by IEEE 1516. This format was not included in the performance tests. The FixedFormVarLenSpatialStruct used for evaluation is shown in Table 3.3.

RPR FOM denotes variant records by specifying their cardinality to be “0-1”.

Field Name / Data Type	Cardinality
DeadReckoningAlgorithm DeadReckoningAlgorithmEnum8	1
IsFrozen Boolean	1
pad1 Char	1
pad2 Char	1
Orientation OrientationStruct	1
WorldLocation WorldLocationStruct	1
VelocityVector VelocityVectorStruct	0-1
AngularVelocity AngularVelocityVectorStruct	0-1
AccelerationVector AccelerationVectorStruct	0-1

Table 3.3 FixedFormVarLenSpatialStruct complex datatype

3.4 Variant Attribute

The Variant Attribute format uses a single attribute with a complex datatype. Some portion of the fields of this datatype are variant records, meaning that depending on the value of an always-present field, one of the variants will be present.

Table 3.4 shows the complex datatype used for evaluation. Table 3.5 through Table 3.9 contain the definitions of the variant record datatypes used by VarSpatialStruct. Note that only five different datatypes are required for all nine modes. Additionally, due to the way these datatypes have been laid out, there are only two in-memory formats for the data. This simplifies the implementation software by eliminating the need for a large case statements typically required when processing variable data formats and instead using a simpler if-then-else for the two formats.

Field Name / Data Type	Cardinality
IsFrozen Boolean	1
DeadReckoningAlgorithm DeadReckoningAlgorithmEnum8	1
pad1 char	1
pad2 char	1
SpatialStatic SpatialStaticStruct	0-1
SpatialFPW SpatialFPStruct	0-1
SpatialRPW SpatialRPStruct	0-1
SpatialRVW SpatialRVStruct	0-1
SpatialFVW SpatialFVStruct	0-1
SpatialFPB SpatialFPStruct	0-1
SpatialRPB SpatialRPStruct	0-1
SpatialRVB SpatialRVStruct	0-1
SpatialFVB SpatialFVStruct	0-1

Table 3.4 VarSpatialStruct complex datatype

Field Name / Data Type	Cardinality
Orientation OrientationStruct	1
WorldLocation WorldLocationStruct	1

Table 3.5 SpatialStaticStruct

Field Name / Data Type	Cardinality
Orientation OrientationStruct	1
WorldLocation WorldLocationStruct	1
VelocityVector VelocityVectorStruct	1

Table 3.6 SpatialFPStruct

Field Name / Data Type	Cardinality
Orientation OrientationStruct	1
WorldLocation WorldLocationStruct	1
VelocityVector VelocityVectorStruct	1
AngularVelocity AngularVelocityVectorStruct	1

Table 3.7 SpatialRPStruct

Field Name / Data Type	Cardinality
Orientation OrientationStruct	1
WorldLocation WorldLocationStruct	1
VelocityVector VelocityVectorStruct	1
AngularVelocity AngularVelocityVectorStruct	1
AccelerationVector AccelerationVectorStruct	1

Table 3.8 SpatialRVStruct

Field Name / Data Type	Cardinality
Orientation OrientationStruct	1
WorldLocation WorldLocationStruct	1
VelocityVector VelocityVectorStruct	1
AccelerationVector AccelerationVectorStruct	1

Table 3.9 SpatialFVStruct

A final note on the attribute names used here: the names `FixedSpatialStruct`, `VarSpatialStruct`, and `FixedFormVarLenSpatialStruct` were chosen only to allow inclusion of all solutions in a single FOM for testing purposes. They are not the proposed names for use in the RPR FOM. The real name should be something simple, such as an attribute name of `Spatial` with type `SpatialStruct`.

4. Test Setup - Hardware, Operating System

All tests were run between a pair of Motorola VME Single Board Computers (MVME2700) with 266 MHz PowerPC processors. The operating system on the boards was VxWorks using C++ compilers from Green Hills.

All RTI updates used best effort (UDP/IP) transport. The multiple attribute tests sent the attributes as a single set. The DMSO RTI used was 1.3NG version 2. The default RID file was used, which means bundling is on and set to 5 ms. However, a wait of more than 5 ms was put between updates to ensure no updates were bundled together. The MaK RTI was Version 1.3.

The RPR FOM spatial attributes were used as the test data. The original RPR FOM use of separate attributes and the two new solutions of attribute format described in the previous sections (Combined with Fixed Format and Combined with Variant Records) were tested. The number of attributes was varied from one to five, starting with position only, then adding orientation, velocity, angular velocity, and acceleration. Each of the last four attributes adds 12 bytes of user data.

The most consistent processor timing measure was found to be the time it takes to make an update call to the RTI. A high-resolution timer on the Power PC board was used to tag the time immediately before an RTI update call was and again immediately after it returned. The “Time per Update” is the difference between these two times. Each test condition was run in a loop of 5000 repetitions with sufficient time between each repetition to prevent overrunning any resource (CPU or network bandwidth). The results shown here are the average of the 5000 repetitions.

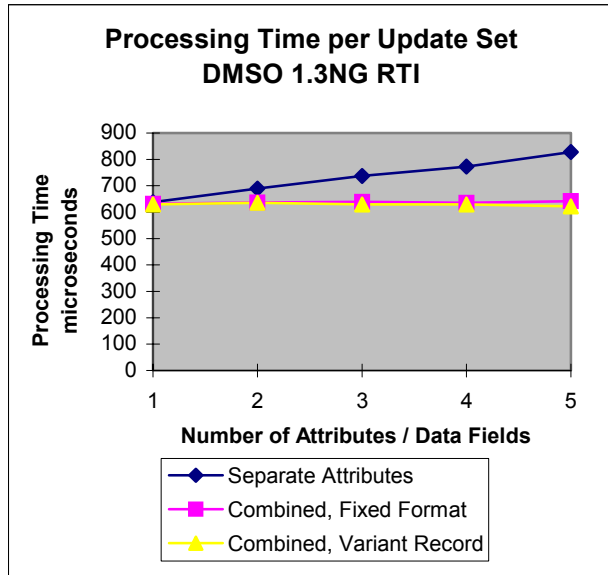
An attempt was made to measure performance on the receiving side by timing the RTI tick calls that resulted in an attribute reflection callback. The times varied too widely to get usable results.

Network utilization was measured simply by capturing the Ethernet packets that were sent as a result of each update call and recording their size. The size of the Ethernet and UDP/IP header is not included in the results, just the “payload”, *i.e.* the data generated by the RTI.

No testing was done for interaction parameters, only for object attributes.

5. Test results

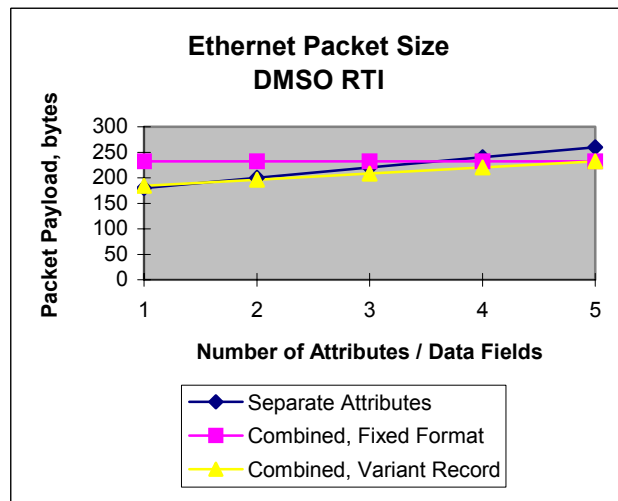
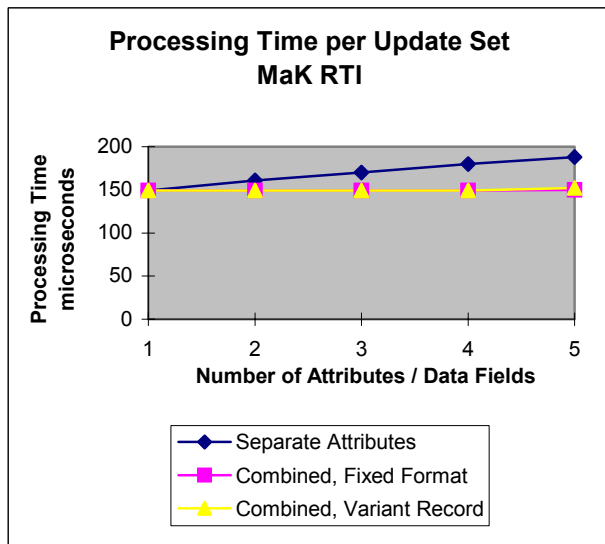
The following two charts show the results for processor time per update, the first chart for the DMSO RTI and the second for the MaK. It can be clearly seen how much processing overhead is incurred for each separate attribute compared to the combined attribute methods. Each attribute adds about 7.5% of processing time for the DMSO RTI and about 6.5% for the MaK RTI.



- 22.5% for DR algorithms 3, 5, 7, and 9 (RPW, FVW, RPB, and FVB, adds rotational velocity or acceleration)
- A full 30% penalty for DR algorithm 4 (RVW, all five attributes).

Clearly, a significant increase in performance can be gained by changing the RPR FOM to use a single spatial attribute, either fixed format or variant record solution.

The next two charts show the amount of network bandwidth utilization per update. This is simply the number of bytes in the Ethernet packet generated as a result of each update. The UDP/IP and Ethernet headers are not counted, just the user data and RTI overhead.

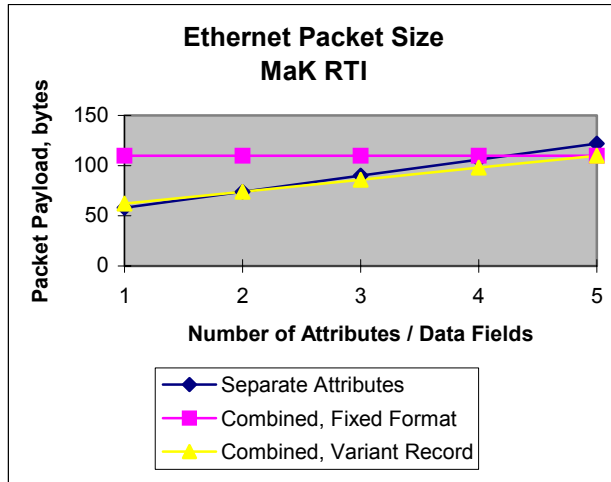


Results for the DMSO RTI show eight bytes of additional overhead per attribute. These eight bytes are thus the amount of savings per attribute if they are combined. The significance of this extra overhead depends on the amount of user data in the attributes. Here, eight bytes are quite significant compared to the 12 bytes of user data. However, eight bytes may be considered insignificant for larger attributes. The MaK RTI shows 4 bytes of additional overhead per attribute.

The size of the data has no effect on processing time in this example. Note that for the fixed format attribute, which always sends all 76 bytes of user data, there is no distinguishable difference in processing time than for the variable length attribute, which varies from 28 to 76. Thus, the simple fixed format solution has no processing time penalty, only a network bandwidth usage penalty.

For the RPR FOM, the impact of its current method of using separate attributes for spatial data is seen to be:

- 7.5% increase in processing time for Dead Reckoning algorithm 1 (static, uses two attributes: location and orientation)
- 15% for DR algorithm 2 and 6 (FPW and FPB, adds velocity)



Minor note: The charts show that for one attribute, the combined packet size is four bytes larger than the "separate" attribute. This is because when attributes are combined, one extra data field must be added to indicate which variant is in use. In the case of the RPR FOM, this is the Dead Reckoning Algorithm. Padding for alignment makes it four extra bytes.

6. Conclusions & Recommendations

The results clearly show that fewer large attributes are significantly more efficient than many small attributes. Each additional attribute beyond the first adds roughly 6 to 7% to the RTI update processing time for the attribute set for both the DMSO and MaK RTIs. There is also a bandwidth savings, about eight bytes per attribute for the DMSO RTI and four bytes per attribute for the MaK RTI. It makes sense to combine pieces of FOM data that normally get sent together into larger complex attributes, especially if the data is sent often. Processor overhead is greatly reduced and bandwidth utilization is somewhat reduced.

The size of the attribute has much less effect on processor utilization than the number of attributes. In fact, the difference in processing time for size differences of a few tens of bytes is insignificant. If network bandwidth usage is not critical, the all-around easiest solution is to combine attributes that may be sent together into a single fixed format complex attribute.

If bandwidth is limited, variant records may be used to combine attributes and send only the pieces that are necessary. This can cause the FOM and the implementation software to get complicated, making coding and debugging more difficult. The software complexity can be mitigated by keeping the data format constant and setting up the variant records in the FOM so that only the data length is varied.

Since spatial attributes make up a significant number of updates for federations using the RPR FOM, we propose that the RPR FOM be changed to use just one spatial attribute. While the fixed-format solution is simpler, there are some bandwidth-sensitive users who require the variant record solution. The FOM complexity for the variant records need be dealt with only once. Using the data format described in this paper would minimize the implementation software complexity because the code needs to use only two different data structure formats.

References

- [1] Nemeth, David: "Benchmarking the RTI for Use in a Simulated Radio Environment", 1999 Spring Simulation Interoperability Workshop Proceedings, 99S-SIW-046, Orlando, FL.
- [2] RPR FOM Standard Development Group, SISO-STD-001.1-1999: Real-time Platform Reference Federation Object Model

Author Biographies

BOB MURRAY is an Associate Technical Fellow in the Research and Development group of Boeing Training and Support Systems in St. Louis with 15 years of experience in the flight simulation technology. He has worked on nearly all software and hardware aspects of applying commercial network products to meet real time I/O and distributed simulation requirements. Mr. Murray was responsible for the HLA implementation in the C-5 DMT program and continues to develop and support distributed computing products for training systems, engineering flight simulators, and other programs in Boeing. Mr. Murray received an MSEE from Washington University in St. Louis in 1993 and a BSEE from the University of Cincinnati in 1983.

ADAM FAIER is an Embedded Software Engineer in the Research and Development group of Boeing Training and Support Systems in St. Louis, focusing on simulation networking. He has worked on a variety of problems relating to real-time distributed simulation and provides DIS and HLA products and technical support for TSS programs. Mr. Faier received an MSCS from Washington University in St. Louis in 1998, as well as BSCS and BSCoE in the same year.