DISTRIBUTED SIMULATION USING DDS AND CLOUD COMPUTING

Michael M. Madden
Simulation Development and Analysis Branch
NASA Langley Research Center
Mail Stop 125B
Hampton, VA, USA
Michael.M.Madden@nasa.gov

Patricia C. Glaab
Aerospace Vehicle Design and
Mission Analysis Branch
NASA Langley Research Center
Mail Stop 442, Hampton, VA, USA
Patricia.C.Glaab@nasa.gov

ABSTRACT

NASA Langley Research Center identified a need for a distributed simulation architecture that enables collaboration of live, virtual, and constructive nodes across its local area network with extensibility to other NASA Centers and external partners. One architecture that was prototyped and evaluated employed Data Distribution Service (DDS) middleware and the GovCloud cloud computing service. The nodes used DDS to exchange data. GovCloud was added as a potential solution to enable other Centers and external partners to join the distributed simulation through an existing trusted network, removing the need to establish case-by-case interconnection security agreements. The prototype architecture was applied to an airspace simulation of manned and unmanned vehicles exchanging Auto-Dependent Surveillance Broadcast (ADS-B) messages. Various configurations of nodes were run and evaluated to assess the architecture with respect to upfront investment to augment a node for the architecture, integration and interoperability of nodes, performance, and connectivity and security.

Keywords: distributed simulation, data distribution service, cloud computing.

1 INTRODUCTION

In 2013, NASA Langley Research Center (LaRC) launched a strategic initiative called the Comprehensive Digital Transformation (CDT) "to develop a comprehensive vision for our aerospace disciplines, the future digital work environment and their combined ability to impact NASA mission needs" (LaRC 2016). One issue relevant to the CDT mission was enabling the transition of experimental labs at LaRC from self-contained entities to a paradigm allowing connected partners to conduct research in a distributed but unified environment. While connected and distributed IT systems were status quo in commercial applications, the NASA experimental environment was lagging. Part of the problem was simply inertia. An isolated computer environment is much easier to secure than a connected environment. However, in most cases the uncertainty of the performance of these technologies in an environment such as that required for LaRC scientific research was the primary concern. Even if technically feasible, changes to operational environments require investment in hardware, software, and training. The CDT wanted a design to demonstrate technical feasibility of a new system in a real facility operating within the current LaRC process and security environment to assess the cost of such a new paradigm against the potential research benefit.

To better understand the issues of applying new digital paradigms to research and development at LaRC, the CDT funded a series of deep dive efforts. These deep dives were small, targeted initiatives that could be accomplished with minimal funding to help inform larger questions. One of those deep dives was the Distributed Simulation Prototype (DSP) whose focus was networked modeling and simulation and system-of-systems integration (Glaab and Madden, 2015). LaRC already had experience running distributed

simulations using Distributed Interactive Simulation (DIS) and High-Level Architecture (HLA) implementations (IEEE 2010, IEEE 2012). Therefore, their strengths, limits, and costs to support an architecture for a broad array of networked modeling and simulation products over local-area network (LAN) and wide-area network (WAN) topologies were known. DSP would investigate a distributed simulation architecture using the Distributed Data Service (DDS) specification from the Object Management Group (OMG 2007) and the GovCloud cloud computing service provided by Amazon Web Services (Amazon 2016). The result would be a comprehensive comparison of technology costs and requirements to inform the CDT's assessment of benefits for the larger LaRC scientific community.

DDS is data-centric, publish-subscribe middleware that DSP used for data exchange between the simulation nodes. A number of factors generated interest in evaluating DDS. First, DDS is an open specification for data exchange between real-time applications, and it includes configurable quality of service (QoS). Thus, DDS was expected to handle the low-latency information needs of live nodes, virtual nodes, and real-time constructive nodes while not precluding use by nodes running fast-time or asynchronously. Second, the DDS specification includes an interoperability wire protocol to accomplish interoperation between different vendor implementations. Thus, an investment in DDS should not require vendor lock-in. Third, there exists more than one no-cost implementation of DDS. Therefore, any team, internal or external, would not be required to incur upfront software licensing costs to join the distributed simulation. Finally, DDS includes peer discovery that permits multiple nodes to form a distributed application at runtime without prior knowledge of each other embedded in the software. As will be discussed later in this paper, DDS peer discovery tends to be configuration-free only with simple network topologies, however.

LaRC was also interested in GovCloud as a potential solution to a couple of problems related to distributed simulation. The first problem was giving external partners, including other NASA Centers, access to the distributed simulation's network. Under LaRC's security rules at the start of the deep dive, each project with external partners had to establish one or more interconnection security agreements to allow the bidirectional flow of the distributed simulation traffic through the LaRC firewall. The distributed simulation therefore had to be configurable to communicate through a firewall and, often, over a private WAN or the open Internet. GovCloud, on the other hand, is a trusted network that appears as an extension of the LaRC LAN (LaRCNet). Moreover, having a standard technology for connecting with external partners could streamline the process for the interconnection security agreements and possibly allow fixed-duration renewable agreements for external partners who regularly collaborate with LaRC on a variety of projects. The second problem was providing every team the ability to increase system-of-systems simulations to real-world scales without increasing contention for NASA's high-performance computing (HPC) resources. With GovCloud, teams can increase the number of virtual machines running simulation nodes, paying only for the time those additional virtual machines are needed and used.

To assess the DSP architecture, LaRC developed 27 goals that described LaRC's ideal architecture (Glaab and Madden 2015). Those goals fell into five broad categories:

- upfront investment including direct costs and staff training to add distributed capability
- development or augmentation of nodes and node maintainability
- integration and interoperability of nodes
- performance
- connectivity and security

A few examples of these goals are listed below:

- Allows nodes to be written in any ubiquitous computing language at use in NASA
- Scalable to any number of nodes, configurable at run-time
- Allows a node to join at any time
- Can operate and isolate multiple distributed simulations simultaneously
- Publishers not coupled to subscribers and vice-versa

The assessment results were then used to generate findings to inform future planning and decisions. This paper highlights the distributed simulation configurations in which DSP was exercised and selected findings of the project regarding that application of DDS and cloud computing to distributed simulation at LaRC.

2 THE DISTRIBUTED SIMULATION PROTOTYPE

The Distributed Simulation Prototype was initially conceived as a pair of executables, an aircraft simulation and a simulation manager. The aircraft simulation was pre-existing software from the Langley Standard Real-time Simulation in C++ (Leslie et al. 1998) with augmentation to incorporate DDS for distributed communications. The aircraft simulation published Auto-Dependent Surveillance-Broadcast (ADS-B) reports (RTCA 2002) and execution-control responses, and it subscribed to both ADS-B reports and execution-control commands. The execution-control topics were used to remotely control and synchronize mode transitions among the participating simulation nodes. The control topics included transitions to reset, trim (dynamic equilibrium calculations), hold (freeze), and operate modes. Multiple copies of the aircraft simulation would be run in DSP scenarios with one copy per simulation node. The simulation manager was new software with a simple terminal interface that allowed a user command mode transitions in the distributed simulation and was also capable of dumping the DDS traffic to screen. Thus, the simulation manager published execution-control commands, and it subscribed to execution-control responses and ADS-B reports. Only one instance of the simulation manager ran in DSP scenarios. Both executables utilized OpenSplice DDS Community Edition Version 6.4. OpenSplice was chosen because it is has a license-free version, and, in prior developer experience, it appeared to be a mature, full-featured, and low-latency implementation.

Figure 1 depicts the runtime environment for the DSP scenarios. The computing platforms for running simulation nodes were the real-time servers in the Simulation Development and Analysis Branch (SDAB), the development desktops in SDAB, and the NASA Cloud (i.e., NASA's account within GovCloud). The SDAB desktops were utilized in DSP scenarios because they reside in a separate subnet of the LaRC LAN (LaRCNet) from the real-time servers and could therefore emulate the issues that different labs at the Center

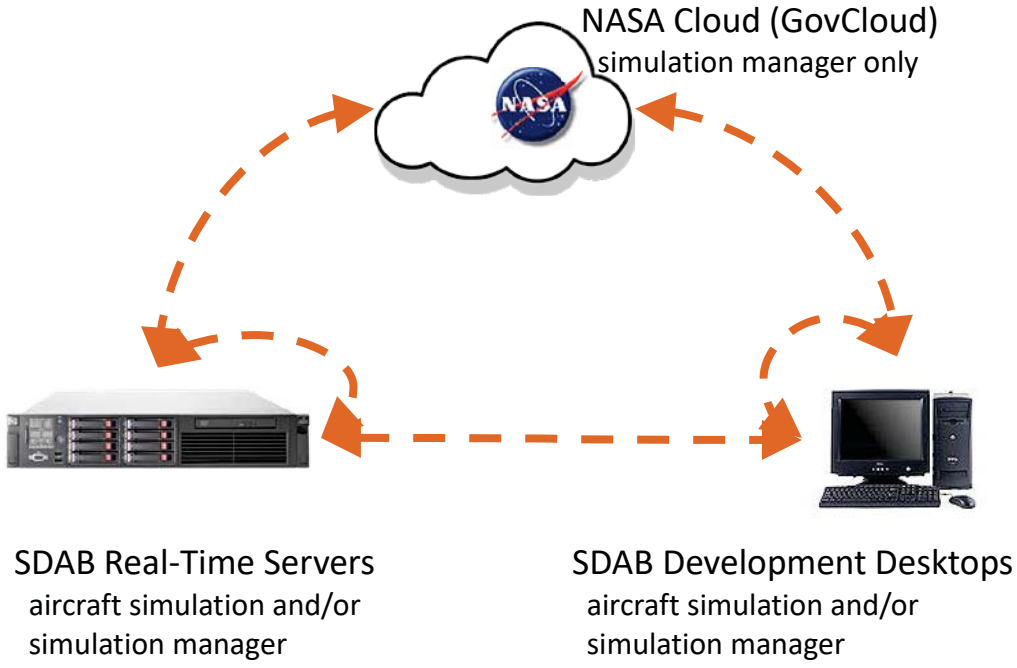

Figure 1: DSP Runtime Environment.

may encounter in trying to establish and maintain communication. Within this environment, the following configurations were run:

- Two or more aircraft simulations exchange ADS-B messages under control of the simulation manager with each executable running on a different real-time server. In this configuration, all nodes are on the same subnet.
- Two or more aircraft simulations exchange ADS-B messages across real-time servers and are controlled by a simulation manager on a developer desktop. In this configuration, the simulation manager is on a different subnet from the aircraft simulations.
- One aircraft simulation on a real-time server exchanges ADS-B messages with a second aircraft simulation on a developer desktop. The simulation manager, running on a cloud instance, controls both aircraft simulations. In this configuration, each aircraft simulation runs on a different subnet and both communicate with the simulation manager running in GovCloud.

Partway through the DSP project, additional Langley projects chose to implement DDS, and one of those projects also implemented a scalable, national-airspace simulation in GovCloud called Shadow Mode Assessment using Realistic Technologies for the National Airspace System (SMART-NAS) (Palopo et al. 2015). This created an opportunity to demonstrate DDS and cloud computing with an expanded set of assets including live and virtual nodes. Figure 2 displays the runtime environment for this expanded demonstration. In this environment, the aircraft simulation from DSP operated on one or two of LaRC's high fidelity simulators; the simulation manager was not used in this environment. The other participants in the demonstration were SMART-NAS running in GovCloud, the Cirrus SR-22 Surrogate Unmanned Aerial System (UAS) in flight, and the ground control station for the Airborne Subscale Transport Aircraft Research (AirSTAR) UAS being driven by a simulation of AirSTAR. (The lead time for approval to fly AirSTAR prevented use of the live vehicle for the demonstration.) The Data Generator shown in the diagram supported development of the demonstration by mimicking the Cirrus SR-22 data. The Cirrus SR-22 ground station, SMART-NAS, and the SDAB simulators all communicated directly over DDS. The

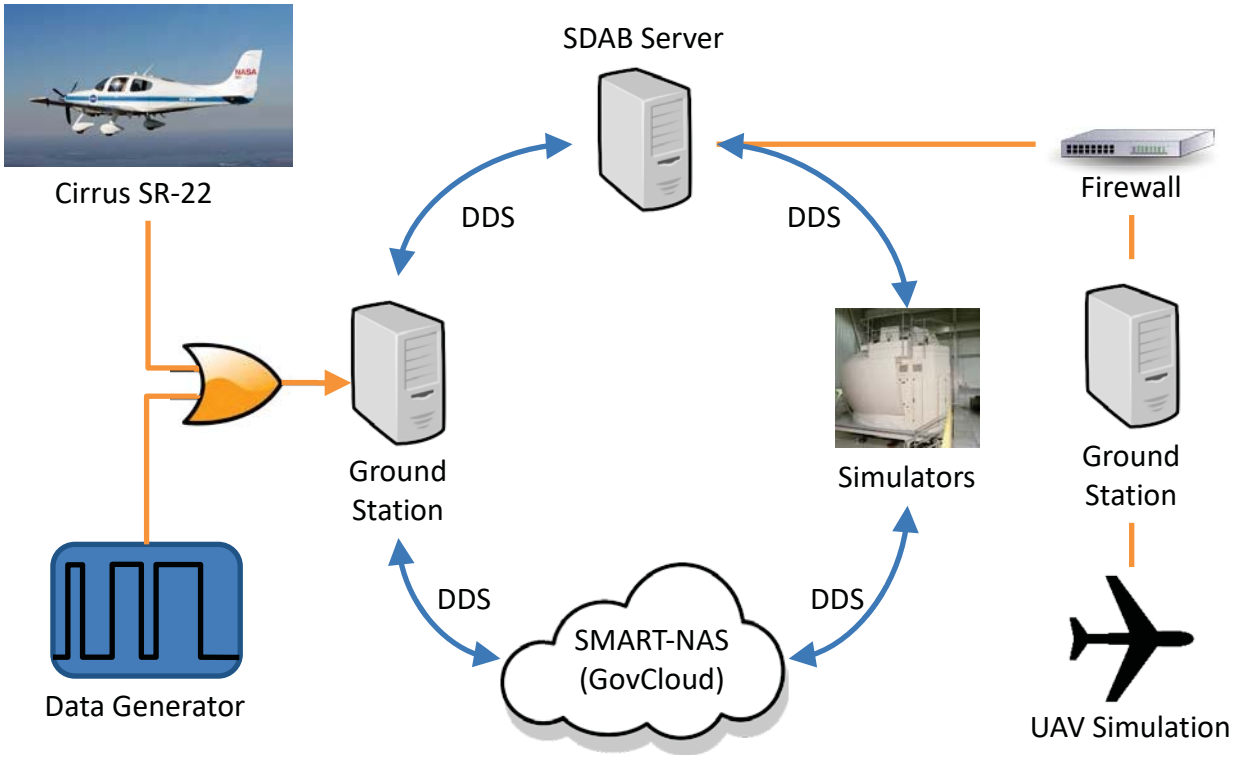

Figure 2: SMART NAS Demonstration Environment.

AirSTAR ground station, on the other hand, sat behind a firewall whose configuration could not be changed, due to security restrictions, to support DDS communication with multiple peers. Therefore, the AirSTAR ground station communicated with an SDAB real-time server that acted as a bridge for DDS publications and subscriptions. In this demonstration environment, all participants also used OpenSplice DDS 6.4.

3 RESULTS

3.1 Upfront Investment

The sponsors of this research effort, the CDT, wanted an estimate of the investment required to transition an autonomous Langley lab environment to one that could operate as part of a distributed research environment. This cost was to include not only required procurement, but also the implementation time to inform developer difficulty and LaRC process hurdles. The DSP development team had prior experience in the implementation of wholly new technologies which may have facilitated their task, but they also had to build their own expertise without prior similar Langley initiatives to leverage to offset any unfair advantage. The delay and expense incurred by the Langley processes (primarily GovCloud access and firewall changes) should be similar for any LaRC lab, though.

Four developers working part-time on DSP completed the run-time environment in four months for a total effort of 1073 staff hours. The cloud computing expenses were the only direct costs incurred outside of charged hours. An abundance of DDS overviews and tutorials including the OpenSplice Tutorial were effective in providing a short learning curve for the staff to adopt DDS (OMG 2016b, PrismTech 2014b). One advantage that the DSP run-time environment had at the start was that the existing aircraft simulation already had a distributed simulation capability utilizing a data model developed for HLA that was transmitted via a TCP/IP stream. DDS utilizes Interface Definition Language (IDL) to define the application's data model. Because IDL has syntax similar to the C++ language utilized by the simulation, it was simple to translate the existing data model into IDL. OpenSplice then provides a program, idlpp, that generates the C++ source code that implements the DDS support for the data model. To complete the DDS augmentation of the aircraft simulation, the development team simply replaced the code that managed the TCP/IP connection with code that managed a DDS connection. The data going into or out of this connection component remained the same to the rest of the simulation whether the connection was TCP/IP or DDS. The connection code was also developed so that it could be reused in the simulation manager, reducing its development effort. Initial testing on the SDAB real-time servers also went very smoothly using the default OpenSplice configuration file. Additional research, debugging, and configuration did not become necessary until attempts were made to expand communications outside of the subnet containing the SDAB real-time servers. The next section addresses this in more detail.

3.2 Connectivity

Results related to connectivity issues illustrate the importance of in-house prototype testing to inform the larger investment decisions. Configuration choices for the Langley network impacted the ability to use one of the nice features of DDS – peer-to-peer discovery – which was not evident from the literature study. Security restrictions on the LaRC firewall also necessitated work-around solutions that increased the complexity of the final system as well as the implementation time. If this type of system is to be used across the Center by many labs, the benefits and costs of changes to the LaRC network will have to be weighed against the time and effort required by each lab environment to implement the same type of work-around.

In the initial testing among the SDAB real-time servers, any number of nodes could join the distributed simulation at any time during its execution without any user configuration. This configuration-free, peer-to-peer discovery is designed into DDS. However, the first attempt to add one of the developer desktops as a node resulted in the failure of that node to make a connection. That failure led the developers to research how DDS accomplishes peer discovery and also how LaRC configures LaRCNet.

DDS peer discovery relies on Simple Participant Discovery Protocol (SPDP) messages that are sent using a multicast address (PrismTech 2014a). Thus, two peers will discover each other only if they succeed in exchanging SPDP messages. The routers on LaRCNet are configured to contain standard multicast addresses within a subnet and to add all machines in that subnet to the associated multicast group. Thus, only machines within the same subnet will accomplish configuration-free discovery. Routers could be configured to bind multiple subnets into a multicast group. However, there are network security and maintenance concerns associated with such an approach, and changing the network configuration for a pathfinder project like DSP would not be realistic. Fortunately, OpenSplice DDS does support unicast delivery of SPDP messages. In the OpenSplice DDS configuration file, the developer can list the machines to contact directly during peer discover. Discovery succeeds between two machines if each machine lists the other as a unicast peer. With OpenSplice DDS 6.4, the project was unable to reliably connect nodes when enabling both multicast and unicast discovery. Reliable connectivity occurred only when the nodes were configured for only multicast or only unicast. Therefore, when including nodes outside the local subnet, the configuration file was configured for unicast discovery only and listed all of the potential participating nodes. OpenSplice DDS does not require that all the nodes on the list connect to begin exchanging data. Data exchange begins once two nodes on the list connect, and a distributed simulation can operate with any subset of nodes on the list. However, a node cannot connect and participate in the distributed simulation unless it appears on the list. Managing and deploying node lists does incur some overhead that increases with the number of participating nodes. Moreover, it is another element of pre-coordination among partners that hampers agility in adding new nodes. Nevertheless, it retains the convenience of having run-time configuration of participants for the distributed simulation.

When the run-time environment was further expanded to include a node in GovCloud, further research and configuration became necessary because, at the time, LaRCNet put the GovCloud addresses on the other side of the LaRCNet firewall. Therefore, to establish connectivity between LaRCNet and GovCloud peers required firewall rules that would allow the DDS traffic to traverse the firewall. The DDS Interoperability (DDSI) wire protocol defaults to sending DDS traffic using User Datagram Protocol (UDP) over a range of ports whose size is, in part, dependent on the number of participant nodes. Establishing connectivity to GovCloud was a simple matter of opening enough UDP ports between LaRCNet and GovCloud. This range did differ from the DDSI default that starts with a base port number of 7400 but OpenSplice does provide the option of setting the base port in its configuration file. All participants, of course, were required to use this same base port setting. However, once the ports were open in the firewall and all participants had the new base port set, connections between nodes were established without problems.

Another firewall presented a challenge during preparations for connecting to the SMART-NAS demonstration environment. The AirSTAR ground station sat behind a firewall whose security plan did not permit the opening of UDP ports. Only Transmission Control Protocol (TCP) connections were permitted through Network Address Translation (NAT). OpenSplice does provide an option for using TCP instead of UDP for DDS traffic. However, in practice, only a single node-to-node connection could be reliably established using TCP over this firewall configuration, and the TCP option was exclusive to the UDP option. Therefore, a direct DDS connection could not be established between the AirSTAR ground station and all participants in the SMART-NAS demonstration. To get around this, the project developed a program to act as a DDS bridge for the AirSTAR ground station. This program ran on an SDAB real-time server and would establish the DDS peer connections with the other participating nodes. The program would also establish a non-DDS TCP connection with the AirSTAR ground station. The program would read data from the AirSTAR ground station and publish the data as DDS topic samples; likewise, the program would extract traffic data from DDS subscriptions and forward it to the AirSTAR ground station.

3.3 Latency

OpenSplice provides numerous source code examples to demonstrate how developers can interface with the DDS application programming interface (API). One of these examples generates a pair of executables called ping and pong, and they are conveniently written to report round-trip latency. The ping executable

sends a topic sample, and the pong executable echoes the sample back. The ping executable measures round-trip latency as the difference in the system times recorded when it sent the sample and when it received the echo. The ping executable generates statistics for 20 blocks of 100 published samples by default. The project used ping-pong to measure latency of node-to-node connections for the three types of connections in the DSP run-time environment. The results are shown in Table 1.

Table 1: Data Transfer Latency Using DDS.

Connection Type	**Latency (µs)**
Same subnet	200 to 300
Across subnets	~700
LaRCNet to GovCloud	~115,000

The results show that the contribution of the DDS middleware to latency appears secondary to the network's contribution. This is especially true for the LaRCNet-to-GovCloud communication which represents data being exchanged between LaRC's East Coast computers and GovCloud's West Coast data center. The latency for the same-subnet and across-subnet exchanges are good enough to support nodes with low-latency coupling such as human-machine interaction. These would be more challenging for LaRCNet to GovCloud exchanges; however, this latency remains good for loosely coupled system-of-systems simulation such as the asynchronous exchange of ADS-B messages used for the DSP and SMART-NAS scenarios. Additionally, the round-trip latency also affects the time it takes for nodes to discover peers. There was no noticeable delay in peer discovery between nodes within LaRCNet. However, peer discovery between a LaRCNet node and a GovCloud node was both noticeable and variable; observed times to establish a connection range from a few seconds to as long as two minutes.

Measuring round-trip latency of the CDT programs is more challenging. Unlike ping-pong which responds to receipt of new data, the two CDT simulations perform read and publish operations periodically within a cyclic execution frame of 20 milliseconds; furthermore, the execution frames are not synchronized between the CDT simulations. Variability and uncertainty in publication time, read time, and time differences in the system clocks among the nodes prevents precise computation of end-to-end round trip latency. However, DDS generates publication and reception timestamps for each sample. If these timestamps were echoed back to the publisher using a new topic, then the DDS timestamps of the "echo" topic samples can be used with the echoed timestamps published in those samples to estimate round trip latency. This estimate only covers the round-trip latency within the DDS middleware and omits the time to transfer data between the application and DDS. However, profiling of the CDT simulations indicates that this omission may be as small as 2 microseconds/sample. The DDS timestamp estimation method was incorporated into a variant of the CDT simulation and run with the same-subnet and across-subnet scenarios. (The CDT simulations were not built to run in GovCloud, precluding that scenario.) Additionally, the scenarios were run with different traffic loads of 2, 8, and 68 vehicles split evenly between each simulation execution. In each of these cases, each simulation bursts its vehicle traffic in a single frame twice per second. In the last case, each simulation publishes 34 samples with a total size of 29,376 bytes each burst. Table 2 shows the resulting mean and standard deviation of the estimated latency from a run of each scenario; each run recorded latency estimates from at least 7200 publication bursts. The differences in mean round-trip latency between same-subnet scenarios and across-subnet are less dramatic than for the ping-pong program. However, the standard deviation of the round trip latency does become substantially higher when one crosses subnets. The data also shows that round-trip latency increases with traffic load but the increases shown are proportional to the expected increase in placement of the data on the network. For example, an ideal gigabit connection would take approximately 300 µs to place the 34 vehicle data burst of the 68 vehicle scenario onto the network (assuming a UDP packet for each sample and 180 bytes for DDS overhead). In either case, the latency remains low enough to support low-latency coupling in a distributed application.

Table 2: Latency (μs) Estimated in CDT Simulations with Different Traffic Loads.

	2 Vehicles		8 Vehicles		68 Vehicles	
Connection Type	**Mean**	**Std. Dev.**	**Mean**	**Std. Dev.**	**Mean**	**Std. Dev.**
Same subnet	505	32	516	26	803	116
Across subnets	574	101	610	112	1006	192

3.4 Project Isolation and Data Security

Since DSP was a pathfinder for technologies to enable a general distributed system-of-systems modeling capability within LaRC, both the ability to concurrently support multiple projects and to control access to published data were important goals. DDS provides domains that enable different groups to communicate using DDS over the same network without interference. A domain is a simple numeric identifier for the group that a node intends to join as a participant. A node joining one domain sees no traffic from nodes that have joined other domains; domains offer complete isolation from each other. Thus, different distributed simulations can run simultaneously without interference if each simulation is associated with a unique domain ID. The DDS specification assumes that an application can join only one domain and this limitation exists in most license-free implementations of DDS. However, some of the commercial implementations do provide multi-domain connectivity features so that a node can bridge multiple domains. Within a domain, a group can further compartmentalize into subgroups using partitions. Partitions differ from domains in a number of ways. First, a node can publish or subscribe to more than one partition. Second, each node within the domain will continue to receive traffic for all the partitions in the domain; if a node does not participate in a partition, then all the data received from that partition is simply unsubscribed data and dropped. Third, partitions cannot define topics with the same name since topic definitions are global to the domain. Finally, a partition has a string name rather than a numeric identifier. The SMART-NAS runtime environment did use a partition to differentiate its traffic from other traffic that participating nodes might publish that were not part of the SMART-NAS data model.

Domains and partitions provide application-level isolation and compartmentalization. However, neither of these features provide data security. In fact, DDS vendors provide tools that allow a user to join any domain and view all of its traffic. The OMG has a working group that is drafting a security specification for DDS that maintains interoperability for supporting implementations. The current draft specification addresses authentication, access control, encryption, and digital signatures (OMG 2016A). However, built-in security can currently be found only as proprietary extensions in paid implementations of DDS. Nevertheless, projects can also establish security using networking technologies like IP Security (IPSec) or DDS can simply be used as a conduit to exchange encrypted data payloads. Neither DSP nor SMART-NAS explored data security for DDS traffic.

3.5 Platform Support and Vendor Interoperability

Among the DSP evaluation goals was support for programming languages and operating systems ubiquitous within LaRC. For programming languages, this included C, C++, Java, and MATLAB. For operating systems, this included Linux, Windows, and Mac OS X. There does exist DDS implementations that cover all of these languages and operating systems (RTI 2016). But not all implementations support all languages and operating systems. For example, OpenSplice DDS doesn't support the MATLAB language or Mac OS X. Thus, if the DSP architecture were to be deployed, then vendor interoperability would be a necessity. DDS includes an interoperability specification (DDSI) to assure interoperability among those vendors who support it. This specification provides for run-time interoperability. However, it does require that topics be identically defined by nodes when using different vendors. DDS relies on the use of the OMG Interface Description Language (IDL) specification to define topics and relies on associated specifications for mapping IDL to programming languages. However, the IDL specifications do not address all the needs for

generating DDS topic source code, and there exists some ambiguity into how IDL maps to DDS topics. Thus, each vendor has added proprietary extensions to IDL in support of DDS topic generation. Consequently, one or more IDL files must be written and maintained to support multiple vendors, and experimentation is necessary to verify that the auto-generated source code defines identical topics. This is less than ideal but workable.

The DSP project team performed some ad-hoc verification of interoperability between different languages (C++ and Java), operating systems (Linux and Windows), and different vendors (OpenSplice DDS and RTI Connext DDS). However, incorporating these assessments into the DSP runtime environment remains future work.

3.6 Time Management

Time management is the one set of goals were the DSP architecture lacks support. DDS does not provide time management services. Therefore, the runtime environments had to develop their own time management or rely on other services like Network Time Protocol (NTP). The simulation manager in the DSP run-time environment could command a simulated coordinated universal time (UTC) for the participating nodes to set upon entry to OPERATE mode. This ensured that that the 'time of applicability' reported in ADS-B messages would be based on the same simulated UTC start time. The DSP run-time environment could also rely on the local system clock as the source for the initial simulated UTC time for each node. This, however, relied on the nodes being in agreement on system time to better than ± 15 seconds since validity criteria for ADS-B messages required that the time of applicability in the message be within 30 seconds of current time on the recipient. The LaRCNet nodes in the DSP runtime environment are configured to use NTP with a common time server to set their system clocks. Therefore, these nodes all agreed to well within a second. System time in GovCloud also matched the LaRCNet time server to within a second. Therefore, system time among DSP nodes did match well enough for the ADS-B simulation. Similarly, SMART-NAS publications included timestamps derived from the local system clocks of each node. It too had to rely on the LaRCNet nodes setting their system clocks using NTP and on agreement among the LaRCNet node time servers and GovCloud system time.

In addition to lack of clock synchronization, the DSP architecture provides no time step synchronization among nodes. In other words, the DSP architecture has no mechanism to synchronize the simulation start among the nodes and the advancement of time in agreed upon steps among the nodes. To work around this, the simulation manager in DSP included a rudimentary mechanism to synchronize simulation start by commanding a system time at which all nodes should enter OPERATE mode. This mechanism relied on the local system time of the nodes being in agreement to within sub-second accuracy, which was accomplished using NTP with a common time server for the LaRCNet nodes. However, this could not necessarily be guaranteed between the LaRCNet nodes and GovCloud. This is one of the reasons why GovCloud hosted only the simulation manager and not an aircraft simulation. Furthermore, DSP did not attempt to command or monitor time advancement after simulation start. It relied on each node accurately advancing time. In other words, neither node would experience local clock drift or a local frame overrun that would cause a substantial lead or lag relative to the other nodes for the duration of a simulation run. For DSP, the typical simulation run was 30 minutes or less. The SMART-NAS runtime environment also utilized no time step synchronization. Because the SMART-NAS demonstration used wall-clock time to tag state data, nodes could join and begin operation at any time. Furthermore, like DSP, the SMART-NAS demonstration still relied on the ability of constructive and virtual nodes to locally maintain accurate time advancement after they entered operation.

As another consequence of lacking time management, DDS is also unable to guarantee deadlines for synchronous communications or provide time-triggered communications. For example, DDS does include a LATENCY_BUDGET quality of service (QoS) policy setting that allows the application to specify "the maximum acceptable delay from the time the data is written until the data is inserted in the receiver's application-cache and the receiving application is notified of the fact" (OMG 2014). However, the DDS

specification then states, "This policy is a hint to the Service, not something that must be monitored or enforced. The Service is not required to track or alert the user of any violation" (OMG 2014). In fact, the specification only requires that an implementation compares the policy setting between the publisher and its subscribers to assure the values are compatible. It is up to each implementation whether the policy setting has other effects such as influencing internal prioritization of traffic. Whether the system can reliably meet the latency budget is often left to the designers of the node applications and the network. In general, DDS, at best, provides real-time compatible publication and subscription services whose actual end-to-end performance often relies on the design and configuration of the underlying network. There is no requirement in the DDS specification for implementations to manipulate network parameters, like the differentiated services field in the IP header, to improve real-time performance or manipulate network QoS.

Without time management, the DSP architecture cannot support distributed simulations with low-latency, cause-and-effect coupling between nodes, which is a common trait among many traditional distributed simulations. Time management services could be built on top of DDS. However, to involve diverse partners from government, academia, and industry would require development of a standard for time management over DDS.

4 CONCLUSIONS

LaRC funded the Distributed Simulation Prototype (DSP) project to perform a deep dive in applying Distributed Data Service (DDS) middleware and cloud computing to the problem of networked modeling and simulation and of system-of-systems integration. This deep dive evaluated the resulting distributed computing architecture against a set of goals covering upfront investment, integration, interoperability, connectivity, security, and performance. Many aspects of these goals were demonstrated in a runtime environment that included aircraft simulation nodes communicating via ADS-B messages and a simulation manager to synchronize simulated UTC clocks and to synchronize mode transitions among the participating nodes. Additional assessment with a collection of live aircraft, virtual aircraft simulators, and constructive air traffic was possible through collaboration with NASA's SMART-NAS project.

The DSP project established its demonstration prototype in four months with a modest investment of 1073 staff hours. No software licensing costs were incurred because the project used the community edition of OpenSplice DDS. Using the demonstration prototype, the team succeeded in establishing connections among simulation nodes in the same LaRCNet subnet, across LaRCNet subnets, and between LaRCNet and GovCloud. Connectivity within a subnet was configuration-free, allowing any number of nodes to connect without prior coordination. However, because DDS relies on multicast messages for peer discovery and LaRCNet contains standard multicast addresses within a subnet, connectivity across subnets or between LaRCNet and GovCloud required configuration files that limited OpenSplice DDS to unicast discovery using a list of all participating nodes. Connecting to GovCloud also required new LaRCNet firewall rules that allowed UDP packets to pass through a range of ports used by DDS. Measurement of latency for DDS communication between nodes indicated that the network is the primary contributor to latency; the contribution of the DDS middleware to latency is secondary. Roundtrip latency between nodes within LaRCNet was within a millisecond, which is low enough to support low-latency interactions between nodes such as man-machine interfaces. Because LaRCNet and GovCloud are hosted on opposite coasts of the U.S., round-trip latency between nodes hosted in each exceeded 100 milliseconds. Thus, GovCloud is best used for loosely-coupled, system-of-systems simulation such as the asynchronous exchange of ADS-B messages among aircraft in a next-generation airspace system.

To support LaRC needs for a future campus-wide distributed simulation capability, a DSP-like architecture would also need to support isolation of simultaneous projects, data access control and security, multiple programming languages, multiple operating systems, and time management. DDS provides numbered domains as an isolation mechanism. All DDS traffic within a domain is only exchanged between the participants in that domain. However, data access control and security remains a specification extension under development. Currently, access control and security features appear only in paid implementations of

DDS as proprietary extensions and may not be interoperable. DDS implementations do exist for many popular computing languages and operating systems including those at LaRC. However, not every implementation supports the same set of computing languages and operating systems. Thus, vendor interoperability becomes important to realize a general capability to integrate distributed simulation nodes. DDS does specify an interoperability wire protocol (DDSI) that enables run-time interoperability among vendors. However, it relies on assuring that topics are identically defined for each DDS implementation. Although the DDS specification relies on OMG IDL to define DDS topics that can be translated to various computing languages, ambiguities in the specification has led to differing vendor extensions to IDL for auto-generation of DDS topic code. Thus, a project must craft one or more IDL files for multi-vendor support and often needs to run experiments to assure that the generated topic definitions are equivalent.

In addition to lack of security features, DDS also lacks time management features for synchronizing clocks (system and simulated) and progression of simulated time (i.e., synchronizing time steps). DSP developers were left to construct their own time management features or operate their nodes using wall-clock time on systems that synchronize their system clocks to compatible time servers using NTP. In the latter case, the distributed simulation also trusts that nodes will not exhibit a lead or lag in time progression due to clock drift or missed deadlines. Lack of time management also hampers DDS from monitoring or guaranteeing communication deadlines defined by maximum latency or by periodicity. Such detection or guarantees are left to the designers of the network and application. As a result, the DSP architecture has incomplete support for traditional distributed simulations that exhibit low-latency, cause-and-effect coupling. This deficiency would be best solved by development of a standard for time management services over DDS.

Partway through the DSP project, NASA's SMART-NAS project also decided to employ DDS middleware and cloud computing to create a national airspace simulation. The project conducted a demonstration that successfully extended this technology to a distributed combination of live, virtual, and constructive simulation nodes with DSP providing the virtual nodes. However, this demonstration fit well within the limitations of this architecture. It operated using wall-clock time with the assumption that all nodes had previous synchronized their system clocks to compatible time servers. It's reliance on asynchronous exchange of aircraft state data with timestamps also permitted nodes to join at any time without synchronizing time steps, but also assumed that nodes would maintain the pace of simulated time with respect to wall-clock time for the duration of the run.

ACKNOWLEDGMENTS

The authors want to acknowledge the support of Edward McLarney, Joseph Morrison, and the Langley CDT project, which funded the DSP prototype. The authors also want to acknowledge the support of Michael Guminsky and the SMART-NAS project which provided the opportunity and funding to incorporate DSP into the SMART-NAS demonstration environment. Lastly, Figure 2 contains the following third-party clip art used under the Creative Commons Attribution-Share Alike 3.0 Unported license: Server2 by mimooh.svg.

REFERENCES

Amazon 2016. "AWS GovCloud (US)". http://aws.amazon.com/govcloud-us. Accessed November 16, 2016.

Glaab, P. C., and Madden, M. M. 2015. "Description and Requirements for the Comprehensive Digital Transformation (CDT) Distributed Simulation Prototype". Internal Document. NASA Langley Research Center, Hampton, VA.

IEEE 2010. *IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA) - Framework and Rules*. IEEE Std 1516-2010. New York, NY. The Institute of Electrical and Electronics Engineers, Inc.

IEEE 2012. *IEEE Standard for Distributed Interactive Simulation - Application Protocols*. IEEE Std 1278.1-2012. New York, NY. The Institute of Electrical and Electronics Engineers, Inc.

LaRC 2016. "Comprehensive Digital Transformation". https://sites.larc.nasa.gov/strategy/comprehensive-digital-transformation. Accessed Nov. 16, 2016.

Leslie, R., D. Geyer, K. Cunningham, M. Madden, P. Kenney, and P. Glaab, "LaSRS++: An Object-Oriented Framework for Real-Time Simulation of Aircraft," AIAA-98-4529, AIAA Modeling and Simulation Technologies Conference and Exhibit, August 1998, Boston, Massachusetts, American Institute of Aeronautics and Astronautics.

McLarney, Edward L., Glaessgen, Edward H., Ambur, Manjula Y., Hammond, Dana P., Bey, Kim S., Blattnig, Steve R., Cerro, Jeff, Copeland, Wendy A., Del Corso, Joseph A., Fremaux, Charles M., Freidt, Karen L.., Jensen, Brian J., Jones, Kennie H., Thomas, James L., Reith, William, Rinsland, Pamela L., Silva, Walter A., and Wallace, Terryl A. 2014. *Comprehensive Digital Transformation Strategy and Roadmap 2014-2035 Version 1.0*. NASA/TM-2014- 218531, National Aeronautics and Space Administration, Washington, DC.

OMG 2007. *Data Distribution Service for Real-time Systems Version 1.2*. Object Management Group. Available via http://www.omg.org/spec/DDS. Accessed Nov. 16, 2016.

OMG 2016a. *DDS Security*. Version 1.0 – Beta 2, Object Management Group. Available via http://www.omg.org/spec/DDS-SECURITY/1.0/Beta2/. Accessed Nov. 16, 2016.

OMG 2016b. "DDS Resources." http://portals.omg.org/dds/dds-resources. Accessed Nov. 16 2016.

Palopo, K., Gano, B. C., Guminksy, M. D., and Glaab, P. C.. "Shadow Mode Assessment using Realistic Technologies for the National Airspace System (SMART NAS) Test Bed Development". AIAA 2015-2794, AIAA Modeling and Simulation Technologies Conference, June 2015, Dallas, TX, American Institute of Aeronautics and Astronautics.

PrismTech 2014a. *OpenSplice DDS Version 6.x Deployment Guide*. Doc Issue 60, January 31, 2014, Woburn, MA, PrismTech, Limited.

PrismTech 2014b. *OpenSplice DDS Version 6.x C Tutorial Guide*. Doc Issue 25, January 16, 2014, Woburn, MA, PrismTech, Limited.

PrismTech 2016. "DDS Community Edition". http://www.prismtech.com/dds-community. Accessed Nov. 16, 2016.

RTCA, Inc, "Minimum Aviation System Performance Standards for Automatic Dependent Surveillance Broadcast (ADS-B)", DO-242A, Washington, D.C., June 25, 2002, RTCA, Inc.

RTI 2016. " RTI Connext DDS Professional Datasheet." Real-Time Innovations. Available via https://info.rti.com/hubfs/docs/RTI_Pro.pdf. Accessed Nov. 16, 2016.

AUTHOR BIOGRAPHIES

MICHAEL M. MADDEN is the Chief Scientist for the Simulation Development and Analysis Branch at NASA Langley Research Center. He holds a B.S. and M.S. degree in Aerospace Engineering from Virginia Tech. His areas of interests include physical modeling of vehicles and their operating environments, simulation and real-time software architectures, and avionics software for both aircraft and spacecraft. His email address is Michael.M.Madden@nasa.gov.

PATRICIA C. GLAABis the Technical Lead for MDAO under NASA's Transformative Tools and Technologies (TTT) Project, and a Research Aerospace Engineer in the Aeronautics Systems Analysis Branch at NASA Langley Research Center. She holds a B.S. degree in Aerospace Engineering from the University of Virginia. Her areas of interest include aircraft conceptual design to support MDAO tools development and systems-level simulation for arrival and departure routing to support airspace research. Her email address is Patricia.C.Glaab@nasa.gov.