DAIDS: a Distributed, Agent-based Information Dissemination System

Pete HAGLICH, Mike KOPACK and David VAN BRACKLE
{peter.haglich, michael.a.kopack, david.van.brackle}@lmco.com
Lockheed Martin Advanced Technology Laboratories, 1800 Parkway Place, Suite 900
Marietta, GA 30067, USA

ABSTRACT

The Distributed Agent-Based Information Dissemination System (DAIDS) concept was motivated by the need to share information among the members of a military tactical team in an atmosphere of extremely limited or intermittent bandwidth. The DAIDS approach recognizes that in many cases communications limitations will preclude the complete sharing of all tactical information between the members of the tactical team. Communications may be limited by obstructions to the line of sight between platforms; electronic warfare; or environmental conditions, or just contention from other users of that bandwidth. Since it may not be possible to achieve a complete information exchange, it is important to prioritize transmissions so the most critical information *from the standpoint of the recipient* is disseminated first. The challenge is to be able to determine which elements of information are the most important to each teammate. The key innovation of the DAIDS concept is the use of software proxy agents to represent the information needs of the recipient of the information. The DAIDS approach uses these proxy agents to evaluate the content of a message in accordance with the context and information needs of the recipient platform (the agent's principal) and prioritize the message for dissemination. In our research we implemented this approach and demonstrated that it provides nearly a reduction in transmission times for critical tactical reports by up to a factor of 30 under severe bandwidth limitations.

DAIDS was developed as the result of a Phase I and Phase II Small Business Innovative Research (SBIR) project for the Army's Aviation Applied Technology Directorate (AATD). It was developed by ISX Corporation, with support from the Lockheed Martin Advanced Technology Laboratories (LM ATL), and implements LM ATL's Grapevine concept. (LM ATL acquired the ISX Corporation in July 2007, after this was written. All references to ISX are now LM ATL.)

Keywords: Agents, Agent Architectures, Interoperability, Data Access, Fusion, Information Dissemination, Situation Awareness.

1. DAIDS TECHNICAL DESCRIPTION

DAIDS is an agent-based communications prioritization system. It maximizes the use of available bandwidth by applying a "**First-In-Best-Out"** scheme. The determination of "Best" is policy-based, using operational knowledge. These policies are very easy to modify, and can even be determined in the field. Thus, DAIDS provides a communications prioritization mechanism that is based upon operational elements, rather than simply technical elements such as message size.

Each platform in a DAIDS-enabled scenario has a Proxy for each teammate and other friendly platform in the scenario. These Proxies represent their principals in vying for information this platform has to offer.

Each DAIDS platform maintains a Context—a collection of attribute/value pairs that are useful in determining importance of information to that platform. These contexts may contain static information, such as the platform type (e.g., Helicopter, Ground unit, UAV), and various immutable attributes (e.g., size, weight, max speed). Contexts may also contain dynamic information, such as position, altitude, speed, current mission, current operating mode, etc. Contexts can also contain fine-grained specifications of platform information needs, such as areas of interest or specific enemy units that are of particular interest. Every platform maintains its own Context. It keeps track of its own internal state, and updates its context accordingly.

Every Proxy maintains a Context for its principal, which it uses to help determine the importance of new information to its principal. Platforms keep their Proxies up-to-date via Entity State Messages (ESMs). These messages are sent to all other platforms whenever a platform significantly deviates from the information in its context. Entity State Messages contain ONLY changes to Context information. They are prioritized and placed on the Priority Queue, along with all other elements of new information. They initially have a very low priority, so other information may go out first. However, as an ESM remains in the queue, its priority increases so that it will, eventually, be sent. There is never more than one ESM in the queue. If a new one is generated while an old one is still on the queue, the two are merged.

Each platform and each proxy maintains a Deduced Reckoning (DR) model for the physical platform's movement, in order to reduce the number of ESMs issued. The DR model is linear. As long as a platform maintains a constant speed and bearing, it remains true to its DR model, and no new ESM needs to be sent. This is consistent with many simulation paradigms.

Contexts, both at the platform level and on the proxies, are extremely configurable. Clearly, different platform types require different information, so they have different Context templates. These are specified in XML by platform type, and are easy to modify. At runtime, new elements may be added to a Context for even more flexibility.

Information is packaged for distribution in Messages. Any new piece of information is passed to an appropriate Parser, which extracts a set of attribute/value (A/V) pairs describing that information. There are different Parsers for different types of messages. The A/V pairs are passed to the proxies for analysis. The information itself is packaged in a Message and placed on the Priority Queue with the assigned priority. These message parsers are separate, and easily implemented, so new types of information may be easily added for prioritization and distribution.

After a new piece of information is parsed into A/V pairs, these pairs are given to the Proxies, which then run their own Prioritization Scripts. These scripts take A/V pairs from the Message Parser, and from the Proxy's Context, and compute a priority and an expiration time. There is a different Prioritization Script for each type of platform, and more can be added easily, without having to recompile the system. The scripts are currently coded in Python, a simple scripting language, and are easily modifiable or even written from scratch, using only the knowledge of a user familiar with spreadsheet formulas.

Once a priority is computed, each message is placed on the Priority Queue, and is dispatched in priority order. There are some additional algorithmic enhancements to increase the efficiency of this mechanism. Messages waiting in the queue past their expiration times are sent back to the proxies to be reprioritized, as are messages which are selected for dispatch but can't be sent due to a network collision. The retry rate also varies based on priority, so that important messages have a higher probability of getting network resources, even across platforms.

DAIDS supports two modes of operation: Broadcast, and Peer-to-peer. In Broadcast mode, it is assumed that there is a single communications channel, so there is no point-to-point communications between platforms, and information must be broadcast to all Platforms at the same time. In this mode, the Priority responses from the Proxies are averaged, and the information is sent once. Other aggregation techniques could also be used. In Peer-to-peer Mode, it is assumed there are multiple communications channels, and information can be sent directly between specific platforms. In this mode, a piece of information will be placed on the Priority Queue multiple times, once for each other platform it must be sent to. For example, if there are five platforms in the scenario, then each platform will have proxies for the other four platforms. Each new piece of information will be put on the queue four times, possibly with four different priorities.

Architecture

The DAIDS system is an agent-based architecture, centers around a cluster of agents that are referred to as a Platform. These java-based agents can run on virtually any agent architecture system through the use of an ISX abstraction layer known as ISAF. By using ISAF it is simple to change the underlying agent system (provided there is an ISAF implementation available to support the new agent architecture.) As a result, while the DAIDS program was initially based on top of the CoABS Grid agent system, it now runs on top of the Lockheed Martin EMAA system.

In addition to the base configuration, ISX has built a testbed and supporting user interface components to assist in development, testing, and demonstration of the core DAIDS system. These supporting modules provide simulation capabilities and allow us to create multiple DAIDS nodes running on one physical machine, while at the same time simulating bandwidth constraints and network topologies.

Components

Each physical system that will use DAIDS for communications control will contain its own DAIDS Platform constellation. This group of agents contains a primary control agent called a Platform, a Disseminator that is responsible for controlling prioritization of messages, a group of PlatformProxy agents that represent the other platform nodes in the network, and a CommIO agent that is responsible for all communications processing between nodes. These are the minimum required components to support the DAIDS functionality.

Platform Agent: The platform agent provides a virtualized representation of the system on which DAIDS has been deployed. Information such as the context of the local system (location, operation mode, direction and speed of travel, etc.) is held in the platform and used for updating remote DAIDS nodes about the status of the local platform. The platform agent is responsible for accepting communications from other remote DAIDS nodes, and interfacing with applications on the local platform. The platform agent maintains a list of all incoming message traffic from other nodes and relays it to local applications. Conversely, any message traffic coming from local applications are prioritized and queued for delivery to the other remote DAIDS nodes. The process of prioritization is controlled through the Disseminator agent.

Disseminator Agent: The disseminator agent is responsible for taking any messages that come into the platform and determining a delivery priority score. It passes the message to each of the Platform Proxy agents and waits for their evaluation of importance. It applies an algorithm to combine the scores from each of the remote nodes to arrive at a final priority score. This algorithm is encoded within a user-modifiable script and can be changed at any time to fit the needs of the system on which DAIDS is being deployed.

PlatformProxy Agents: Each local DAIDS node will contain one PlatformProxy Agent per remote DAIDS node on the network. These Proxy agents represent the last communicated state of the remote node, except for position, which is projected via the dead reckoning model. This state, or context, is used by the proxy in determining the importance of messages to be delivered to that node. The determination of this priority is controlled through a user-editable script file that correlates information between the message and the proxy's context and arrives at a numerical score. This score is returned to the Disseminator.

PlatformCommIO Agent: Each platform suite also contains one PlatformCommIO agent. This agent is responsible for handling all communications between DAIDS nodes. It is expected that every deployment of DAIDS in a real system will require a new implementation of the CommIO agents. In our testbed environment the implementation is responsible for handling flow control and simulating bandwidth constraints. These requirements are not necessary when deploying in an actual operational environment.

Application Interface Agents: In our testbed environment the main application supported by the DAIDS network is the sensor sharing system that feeds into a common relevant operational picture (CROP) display. This is not the only type of application that could be supported through the DAIDS infrastructure. To support other applications, Agents that interface the application with the DAIDS platform are needed. The Platform agent must be modified to understand these agents, and handlers for any application specific message traffic types must be created. While this does require a code recompile operation to integrate new applications, the procedure involved is simple and straightforward, with template interface specifications provided for new message formats, and clearly

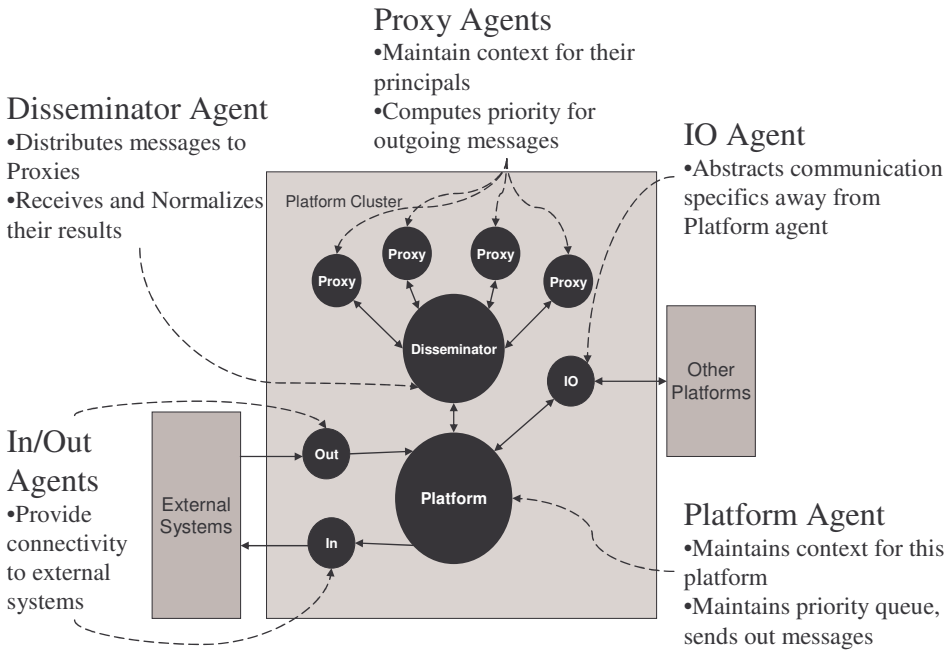

Figure 1: Platform-Level DAIDS Architecture

demarcated locations in the code for placing implementation specific code.

Assessment Scripts: The real power of DAIDS is in the assessment scripts. These scripts are a user-configurable piece of Python code that combine the data found in the message to be prioritized, and the perceived context held in the Proxy of the platform as inputs to the assessment process. The user can use these inputs to determine a formula for a prioritization score. This prioritization score should fall within a rank of 0 (not interested at all, don't send it ever) to 1000.0 (VERY important, send immediately). It is important for the assessment script to adequately spread scores across this range to reflect varying importance levels of messages. Average importance messages should fall somewhere in the middle of this range, near 500.0.

In addition, assessments have a timeout value associated with them. This time-out indicates to the platform how long the assessment score is valid. This helps ensure that the propriety score of a message waiting on the queue for a period of time doesn't become invalid as the situation changes.

Devising and testing assessment scripts that prioritize information correctly will always be the biggest challenge in deploying a DAIDS system. However, with a well-crafted assessment script, exemplary performance can be achieved.

Contexts: The DAIDS system uses the concept of a Context to represent internally the state of the platform it is representing. Contexts can contain any type of information that can be represented in an XML format. An example context for a helicopter might contain location, speed, mode of operation, status, flight plan, and weapons. The information template can be uniquely specified for each platform type, will each instance having its own data set. The platform communicates this context to all other nodes on the DAIDS network, so that the proxies on those other nodes can know the state of their principle, and use it for prioritization. In addition, the DAIDS system periodically sends updated context information to the other nodes to synchronize them with the current state of the principle. In these situations, only changes are sent, not a full picture of the context. This reduces network impact to the minimum possible while maintaining DAIDS capabilities.

Testbed Environment: The testbed environment was built in order to debug and demonstrate DAIDS. It provides visibility into the operation of DAIDS, as well as the ability to manipulate the scenario. Among the features the testbed environment provides:

- A view of the agents
- A view of the message traffic, both textually and graphically
- A view of each platform's CROP
- A view of each platform's Outgoing Message Queue
- A view of each platform's Context
- The ability to create new platforms
- The ability to manipulate the apparent bandwidth and simulate pairwise communications blackouts
- A Scripting capability to run designed scenarios

With the testbed, a complex scenario can be executed, observed, and metrics measured.

Protocols

In the DAIDS development testbed we have developed two different communication protocol examples. The first is a broadcast mechanism that simulates half-duplex shared communications similar to multiple users talking on a single radio frequency. When one node is sending, all remote nodes receive the message (even if it's not meant for them), and all other nodes are blocked from sending during that time. When running in broadcast mode, there is a 1:1 relationship between incoming messages and outgoing messages in the Platform. For example, if the local platform needs to deliver a message to five

remote nodes, there will be a single copy of the message queued with all five remote nodes addresses encoded.

The second protocol is a Peer-to-Peer mode that is full duplex and operates similarly to each pair of nodes having two channels on which to communicate, independent of all other nodes. If the local platform needs to deliver a message to five remote nodes, for example, it will need to queue up the message five times, each copy having its own distinct destination node address.

All messages in the DAIDS system encoded as java String data. This does preclude handling binary data. To support binary or serialized data we require the information be UUEncoded (Base64) so it can be represented as a String.

Multiple message types and formats can be supported, and it is simple to support additional types. For each message type a custom parser class must be implemented according to an interface. This parser should be written to pull important data elements out of the message and store them in a structured, delimited manner, as well as encoding the names of individual data element fields. This information is passed to the PlatformProxies that take this data and inject it into a Jython interpreter for the assessment scripts. For example, in our testbed, images are encoded in Base 64. The ImageReport parser reads this data, and determines the file size, specifies a type (GIF or JPEG) and assigns each of those data elements to a field name. When the message is injected into the Jython Assessment interpreter, the assessment script can use the type and file size to determine the criticality score for this message (for example, larger images should have a lower priority because they'll tie up bandwidth longer.)

All communications between DAIDS nodes is provided by the underlying Agent architecture (which must rely upon the communications infrastructure of Java, the OS and the network socket driver implementation.) As a result, DAIDS itself is not in control of the network. DAIDS simply prioritizes and orders the data so it is sent in a most-important-first manner.

Advanced Features: Built into the DAIDS system are several features that were implemented to maximize the benefits of DAIDS and allow for a balanced control of multiple DAIDS nodes without additional communications overhead. These features are:

Periodic Reprioritization – A thread examines all queued outbound messages to look for any messages that have timed out. These messages are sent back for re-assessment so they can be re-prioritized with new scores that more accurately reflect the importance to the destination platform at this time.

Reprioritization on Block – Any time a platform attempts to send a message, if the bandwidth is blocked causing an inability to send (for instance, in a broadcast environment where somebody else transmitting) the outbound message is sent back for reprioritization. This can possibly cause it to be given a lower prioritization score. This lower score might force it to no longer be the most important thing to send at this time and would allow the new highest priority element to bubble to the top of the queue.

Dynamic Retry Rates – DAIDS will dynamically adjust the rate at which it tries to send a message based on the priority score of that message. The higher the priority, the less time it will wait between attempts. This lowers the latency between attempts for high priority items, while at the same time providing a basic means of network balancing between nodes. For example, if node A is transmitting and nodes B and C both have data waiting to be sent, when A finishes and clears the channel whichever node (B or C) has a higher priority message will retry faster and will be more likely to be the one to get control of the channel next. While this is not guaranteed, it is more likely to occur, and without the need for transmitting sync messages to control this behavior.

Bandwidth Awareness – The platform measures the time it takes between attempts to send a message and when it receives acknowledgement (from the PlatformCommIO agent) that the messages was successfully sent. This measurement is used to determine the effective bandwidth at the time of transmission. This measurement is fed back into the Disseminator agent and can be used inside the Disseminator's Assessor script to modify the prioritization scores coming into it from the Proxies. This allows DAIDS to perform any tweaking that is desired (for example, if in a low bandwidth situation, we might want to distribute low, medium and high priority scores more than normal.)

2. DISSEMINTATION POLICIES IN DAIDS

Prioritized dissemination of critical information in DAIDS is predicated on the ability to express information needs and pass them to proxy agents hosted by the platform providing the information. These information needs are expressed as dissemination policies. The technical details for the implementation of the dissemination policies by the proxy agents were described in Section 1. In this section, we describe the various types of dissemination policies used in DAIDS, the factors that are considered by the specific dissemination policies encoded in the Phase II DAIDS Prototype, and how messages to different platforms are arbitrated.

Types of Dissemination Policies

The DAIDS implementation of dissemination is flexible and allows for a variety of prioritization policies. In the Phase I and Phase II development we focused on dissemination policies needed for an application of DAIDS to a problem of interest for the sponsor. This scenario involves the exchange of information between military units cooperating as a team in tracking potentially hostile targets. We categorize these policies in several ways. The first categorization axis relates to the type of message being prioritized. The next categorization axis relates to whether the prioritization policy is target-based or location based. The final categorization axis describes the intended recipient of the message.

Message Types: The current implementation of DAIDS includes prioritization policies for three specific types of messages. These are sensor reports, image reports, and entity state messages. Our DAIDS architecture includes abstraction of the message concept. This allows the future addition of new types of messages to be sent using the tactical data link and the development of dissemination policies for them.

The primary focus of DAIDS was to enhance the data fusion process through the prioritized transmission of critical target **Sensor Reports**. The contractor team spent the most effort in developing dissemination policies for sensor reports. The prioritization of sensor reports is discussed at length in Section 3 of this chapter.

While prioritization of sensor reports for the fusion process was the initial focus of DAIDS, we were prompted by the project sponsor to consider situations where the tactical data link is used

to share image data. These **Image Reports** compete for bandwidth with sensor reports. Because a typical image has more content than a sensor report, it has the potential to tie up the data link for a long time. These considerations led us to a prioritization criteria where most images received a lower priority than most sensor reports. Our initial dissemination policy for image reports is simply based on the size of the image. In Section 4 below we suggest some more advanced dissemination policies for image reports.

The current prioritization formula for image reports is:

$$Priority = ImageSize/100$$

In order for the proxy agents to be able to assign a meaningful priority to messages, the proxy agents must maintain the current context of their principals. For this and other reasons Blue team members who are not in radio silence send periodic reports of their position and context to other members of the tactical team. These **Entity State Messages** share the same data link and compete for bandwidth with sensor reports. Therefore, the priority of these e messages needs to be reconciled with the priority of sensor reports.

In our implementation, we assume that most entity state reports have a lower priority than most sensor reports. We also assume that the urgency of reporting own context to teammates should increase with the time since the previous report. The formula for the priority of an entity state message is:

$$Priority = 550 + Increment*TransmissionAttempts$$

The time between transmission attempts and the amount of priority increase per transmission attempt can be configured in an XML file that is provided with the DAIDS software distribution. Presently DAIDS uses a 50 second interval between transmission attempts for entity state messages and an increment of 25 per transmission attempt.

Message-Based vs. Location-Based Policies: Another aspect of dissemination policies examines whether they are message-based or location-based. By this, we mean whether the dissemination priority is based on the content of the message or strictly on the location of the underlying data.

In DAIDS we developed one location-based policy suitable for any platform that has a geographic area of interest. This policy treats that area as an information filter, assigning the maximum priority of 1000 to reports from that area and a priority of 0 to reports outside that area.

This type of dissemination policy was developed to implement a suggestion by the sponsor concerning implementation of "Commander's Information Pull." This type of dissemination policy achieves "virtual pull" but directing the top priority dissemination of information meeting a particular criterion determined by the commander.

Platform Policies: The third method of classifying dissemination policies is by the platform is receiving the message. For example, the proxy agent for an Apache Longbow helicopter will have different types and weights for information than a dismounted infantry unit. In DAIDS we developed dissemination policies for helicopters, dismounted infantry units, and artillery units. The dissemination policies for helicopters and infantry units were of the same form but with different parameters reflecting differing needs. The dissemination policy for the artillery unit is an implementation of the location-based policy described earlier.

Sensor Report Dissemination Factors

In this section we use sensor reports as an illustrative example of message prioritization in DAIDS. The dissemination policies for sensor reports received the most attention during the DAIDS research because sensor report messages are most relevant to the initial goal of DAIDS to make the sharing of important parts of the CROP more effective.

At the most abstract level, the priority score for a sensor report is:

$$Priority = BaseScore + ContextScore + ContentScore$$

The **Base Score** is adjusted to ensure all but the least important sensor reports receive a priority greater than the initial priority of entity state messages. Currently DAIDS uses a base score of 400 for sensor reports.

The **Context Score** reflects unique information priorities for the receiving platform. The operational context of the receiving platform plays an important role in evaluating the priority of messages for the platform. In DAIDS our context model for receiving platform includes position and velocity data and the operational mode of the platform. Factors considered include whether the recipient is operating its own sensors and if the recipient and the sender have a relationship of close cooperation known as a "wingman" relationship.

The **Content Score** is computed from the sensor report data. The sensor report includes data on the target's kinematics parameters (position and velocity), identity (type, affiliation), and associated positional uncertainty. The DAIDS dissemination policies currently disregard the positional uncertainty but use the other factors in determining the content score, which is modeled:

$$ContentScore = AffiliationScore + TargetTypeScore + RangeScore + SpeedScore$$

The affiliation score reflects whether the sensor report is about a friendly, neutral, or hostile target. The target type score captures a classification of the target as air, ground, etc. and combines this with the type of the receiving platform to evaluate level of threat, and therefore, level of importance. The range score is based on the current distance of the target from the recipient, and also its projected closest point of approach. The speed score considers the speed of the target, including"closing speed, which is the speed of approach to the receiving platform.

Arbitration

Each proxy agent uses the dissemination policies to compute the importance of the message to its principal platform. However, a sending platform may be sending messages to multiple recipients. The sending platform requires an arbitration policy to reconcile the priorities computed by the proxy agents. As currently implemented, these arbitration policies depend on whether the data link uses a broadcast protocol or a peer-to-peer protocol.

When the network uses a broadcast protocol, each report is sent to all recipients. In this case, the arbitration policy takes the average of the priorities computed by the proxy agents to determine the place of the message on the DAIDS priority queue.

When the network uses a peer-to-peer protocol messages are sent individually to each recipient. In this case the messages are placed on the priority queue in accordance with their priority scores as computed by the proxies. If one recipient is privileged

to hold "wingman" status with the sender, messages to that recipient are placed higher on the queue and sent earlier than messages to other recipients.

Future Enhancements to Dissemination Policies

As presently implemented, the dissemination policies in DAIDS prove the DAIDS hypothesis that operational benefits can be obtained from intelligent prioritization of information in bandwidth-limited environments. These policies were heavily tailored toward use in the DAIDS simulation environment and the AMUST-D sensor report format. The adaptation of DAIDS for use in other command and control systems gives an opportunity to develop enhanced dissemination policies. In this section we describe some possible enhancements.

Intervisibility Modeling: The present DAIDS dissemination policies do not address line-of-sight issues between the principal and the target. This is because the DAIDS simulation did not include an intervisibility model. If line-of-sight computations are available to DAIDS these results can be used to improve the dissemination policies. The principle to be implemented would be "Targets obscured from the principal should receive a higher priority than targets the principal can detect with his own sensors." For this enhancement to be useful DAIDS would need to be embedded in a higher-fidelity simulation or a command and control system.

Target Posture: The posture of the target reflects the likelihood the target will attempt to engage Blue units. Indicators that the target is adopting a hostile posture include weapons employment, starting up weapons systems, using fire control sensors, and aggressively pursuing Blue units. This information could enhance the dissemination policy, using the principle that "targets with a hostile posture should receive increased priority."

Target Aging: Sensor reports are currently prioritized based on their snapshot information. However, it seems that sensor reports concerning a target which hasn't been seen or shared lately would have a higher priority than reports on a target for which reports are being sent regularly. An additional score for target aging would address this aspect.

Information Quality: The data associated with a sensor report includes an estimate of the associated area of uncertainty, expressed as an ellipse. It is plausible that reports with a lower degree of uncertainty have a higher dissemination priority than reports with a higher degree of uncertainty. One candidate model for the information quality score is:

$$InfoQualityScore = K * NominalAreaOfUncertainty / AreaOfUncertainty$$

where K is a normalizing constant.

Image Report Location Metadata: The present dissemination policy for image reports is simple, considering only the size of the image report. A better dissemination policy could consider metadata associated with the image itself. One type of metadata concerns the location of the image. A candidate policy would assign priority to the image report based on the degree of overlap with an area of interest or based on the distance from the receiving platform.

Entity State Deviation From Plan: Presently, DAIDS uses the principle that "The longer it has been since I reported in to my teammates, the more important it is that I do so." However, there is an opportunity to reduce the number of entity state messages being sent. Each DAIDS platform has the capability to publish a plan describing its intended position and posture over time. It is possible to compute the platform's deviation from its plan and use this to determine the priority of the entity state message. The corresponding principle would be "My entity state reports have a lower priority if I am operating in accordance with my published plan."

More Sophisticated Arbitration Policies: The current arbitration policy in broadcast mode has the drawback that the priority of important messages can be diluted if there are other platforms where the message is not important. For example, there are three recipients for a sensor report with the priority for recipient A being 900, the priority for recipient B being 700 and the priority for recipient C being 500. The average priority of the message will be 700, which may fall quite low in the priority queue. To address this problem, alternative methods of combining priorities from different proxies can be studied and implemented in DAIDS. Some alternative arbitration policies are:

- Place the message on the priority queue in accordance with the maximum priority computed by any of the proxies.
- Compute a weighted average of the priority scores that emphasizes the platforms where the message is the most important. Taking the example above, we assign a weight of nine for where the most important message, four for the second most important, and one for the least important. This would result in a composite priority of 814, placing it higher on the priority queue than a score of 700 would place it. The actual weights to be assigned need to be determined through experimentation.
- Determine a criticality threshold (priority score of 900, for example). If any proxy assigns a priority score about that threshold then use that priority score for arbitration. Otherwise, use the average score. This will have the advantage in percolating the most critical reports to the top of the queue.

3. DAIDS TESTING

Summary

We conducted experiments in order to determine whether DAIDS prioritization provides a measurable improvement in the time to disseminate critical information under a variety of conditions typically found in a simple military scenario. We found that DAIDS provides significant reduction in the time needed to share critical information when compared with "First In, First Out" dissemination whenever the available bandwidth is limited. When bandwidth is not limited DAIDS performs as well as FIFO dissemination. We also found that the performance of DAIDS is strongly tuned to the information needs of the recipient.

Test Procedure

Hypothesis: Using DAIDS intelligent prioritization, critical information is delivered significantly faster than a "first-in, first-out" (FIFO) dissemination protocol.

Test Metric: The test metric was the effective latency of the dissemination process for critical reports. We define the effective latency as the mean value of the time delay between

the queuing of the report by the sender and the receipt of the report.

Scenario Assumptions: In order to control the number of variables in the experiment, we made some simplifying assumptions:

- The scenario was set on flat terrain in daytime with good visibility conditions. There was no modeling of intervisibility.
- The receiving platforms remained essentially stationary during the scenario.
- Information being shared was limited to sensor reports and entity state messages. No image reports were transmitted during the experiment.
- The DAIDS platforms ran on a single computer. This was necessary to ensure accurate timestamps for queuing and receipt events.

Controllable Variables

Communications Bandwidth: We ran the scenario using bandwidths of 4800 baud and 1200 baud. 4800 baud was chosen to approximate the nominal bandwidth of a representative tactical communications network currently in use by the Army. 1200 baud was chosen to examine network performance when that bandwidth is degraded.

Communications Mode: We simulated broadcast and point-to-point (P2P) protocols.

Team Size: We ran the experiments with team sizes of seven platforms (one UAV, three helicopters, and three infantry platoons) and four platforms (one UAV, two helicopters, and one infantry platoon.)

Uncontrollable Variables

The primary uncontrollable variable in the experiment was processor performance. To ensure accurate time stamps for the collection of the latency metric we had to run all DAIDS agents on a single computer. This is an unnatural and artificial condition. Early experiments have shown that the efficiency of the processor depends on external factors like the ambient temperature of the room in which the computer is housed. We assume that the performance of the processor is essentially constant during a run but may vary between runs. We performed a small pilot experiment to examine the change in performance from run to run. We observed a high degree of variation between runs, but the relative values of the test metrics tracked each other.

Scenario Description

Our scenario featured a Blue tactical team consisting of a UAV sending sensor reports to helicopters and dismounted infantry units. There were several hostile and neutral units to represent possible combinations of target type (aircraft or ground vehicle), affiliation (hostile or neutral), and offensive capabilities (sensors and weapons carried).

The hostile and neutral units maneuvered through the scenario area during a 40-minute period. The movement of the targets was designed to explore several combinations of proximity and closing speed between the targets and the Blue platforms.

Analysis Methodology: Before running the test we obtained assessments for the importance of the sensor reports from a retired Army Aviation Major General who was a consultant for the DAIDS contractor team and from a representative of the sponsoring organization. The consultant and the sponsor's representative were each presented with the scenario and asked to determine the time intervals during which the targets were of critical importance, major importance, or minor importance for each member of the tactical team. The guidelines provided by the consultant were also used in the development of the information dissemination policies used by DAIDS. For each run we determined the average latency for sensor reports of critical, normal, and minor importance. These statistics are not independent, as an increase in the number of critical reports results in an increased time to transmit sensor reports of lower importance. Therefore statistical techniques such as Analysis of Variance (ANOVA) have questionable applicability. However, we will still be able to get a strong comparison of the average latency for each level of importance.

Pilot Experiment: Run-to-Run Variation: The pilot experiment examined whether multiple runs of the experiment under the same conditions (team size seven platforms, communications bandwidth 1200 baud, peer-to-peer communications protocol) exhibited the same DAIDS performance. We ran this same scenario six times to test for variation in latency values across the runs.

Critical reports had an overall mean of 242.3 seconds with a standard deviation of 76.1. (High value 334, low value 157). Normal criticality reports had an overall mean of 2910.3 seconds with a standard deviation of 695.3. (High value 3770, low value 2031). Minor criticality reports had an overall mean of 7187.2 seconds with a standard deviation of 767.6. (High value 8178, low value 6046).

The average latencies of the three criticality categories had a very high degree of correlation. For example, Critical and Normal reports had a correlation coefficient of 0.974 across the six runs. This means that the external factors affected the entire experiment independently of the criticality categories. This indicates that comparisons of results for a given set of conditions are a valid indicator of relative benefits of DAIDS prioritization. However, comparisons of results across differing sets of conditions are less valid, as external factors may be a greater source of differences in results.

Results Using Contractor Team Importance Assessments: We performed initial test runs using report criticalities determined by the DAIDS contractor team. These runs showed that DAIDS prioritization shows great benefits in bandwidth-limited conditions and under peer-to-peer communications protocols. Under high bandwidth, broadcast-protocol conditions, DAIDS performs as well as a "first in, first out" dissemination protocol.

We present the results for a 1200-baud bandwidth using a peer-to-peer protocol and a large tactical team, the most challenging bandwidth conditions. We divided the average latency by the overall average transmission time to normalize each set of results. This was necessary because external factors render the comparison of absolute magnitudes useless for comparison purposes, as we discovered in our pilot experiment. In Table 1, the number shown is the factor by which the dissemination time is improved, equal to the average time for all reports divided bythe average time for reports of the given level of criticality. As the bandwidth conditions became less restrictive the

performance improvements shown by DAIDS were less marked. However, at no point did DAIDS perform worse than FIFO.

	DAIDS	FIFO
Critical	27.139	0.895
Normal	2.286	0.973
Minor	0.768	1.023

Table 1: Timeliness Improvement Factor, Peer-to-Peer, 1200 Baud, Small Team

Comparison of Results Using Both Importance Assessments: Having observed the disparity between the results using importance assessments generated by the Contractor Team and by the sponsor, we decided to perform one more experiment with both sets of criticality encodings. We chose conditions of 1200 baud, peer-to-peer communications protocol and a small team size enhance the magnitude of the latency differences. The table below gives the number of reports and the mean latency for reports that have importance assessments corresponding to the row and column headings. For example, there were 27 reports that were assessed as being critical using both Contractor team and sponsor assessments. The average latency for these reports was 334.6 seconds. The average latency for all reports was 5762.9 seconds, which was used as the normalizing factor in Table 2.

Number of Reports Improvement Factor	Sponsor Critical	Sponsor Normal	Sponsor Minor	Row Aggregates
Contractor Critical	27 17.241	52 27.027	265 31.250	344 28.249
Contractor Normal	16 3.448	476 2.232	792 1.536	1284 1.751
Contractor Minor	165 0.873	658 1.245	2841 0.742	3664 0.806
Column Aggregates	208 1.066	1186 1.595	3898 0.895	5292 1.000

Table 2: Comparison of Results Using Sponsor and Contractor Assessments

4. SUMMARY

Conclusions

The performance results using the Contractor assessment indicate that DAIDS provides measurably, significantly faster dissemination of critical sensor reports in situations where the available bandwidth limits the rate at which reports can be transmitted. In our experiment, these conditions occurred when

the baud rate was low (1200 baud and below) or a peer-to-peer communications protocol was used.

Where the Contractor and sponsor assessments agreed, DAIDS provided the expected benefits in reduction of latency for higher priority reports.

In any column, DAIDS provided the expected benefits in reduction of latency for higher priority reports.

Less than 8% of the reports assessed by the contractor team as critical were assessed as critical by the sponsor's evaluator. Less than 13% of the reports assessed by the sponsor's evaluator as critical were assessed as critical by the Contractor team. This indicates a divergence of information prioritization needs between the two evaluators. The DAIDS prioritization scripts were based on the same parameters and heuristics as the contractor assessment, which means that reports assessed as critical by the sponsor's assessor did not receive high dissemination priority scores.

We conclude that DAIDS performance is strongly tuned to the information needs of the recipient. When the dissemination prioritization scripts match the information needs closely then DAIDS is effective at disseminating critical information much more quickly

than non-critical information. When the prioritization policies are not matched to the information needs of the recipient DAIDS does not provide any performance benefits.

Ongoing Work: In our Phase II research we have implemented the DAIDS concept. Our testing proves that DAIDS provides measurable, significant benefits with the faster transmission of critical information in bandwidth-limited environments. This proven technology is available to the U.S. Army for transition toward operational use.

The DAIDS technology is be the backbone for two follow-on LM ATL research initiatives, described here.

GUAVA: In the General Unmanned Aerial Vehicle Associate (GUAVA) system, DAIDS was used as a communication control layer between a UAV and a command and control PDA system. DAIDS was responsible for prioritizing control commands, context changes and system failure messages between the processing engines on both systems. The UAV was considered one platform, the PDA another.

UAMS: In the UAV Airspace Management System (UAMS), we proposed a hybrid approach to airspace management that dynamically adjusts the balance of centralized and distributed control strategies among multiple distributed airspace managers (AMs). The distribution of intelligence among these multiple AMs within the airspace management system provides a greater number of resources available for the computation of the problem of collision detection and deconfliction. Dynamically adjusting the control of airspace management among these airspace managers permits exploitation by decomposing both the computational load and the bandwidth requirements of the problem, and can benefit from the topographical/geometric decomposition of the airspace. The DAIDS architecture provides a head start in the necessary intelligent dissemination and communications awareness aspects of UAMS.