Distributed Control of Resources in the Force Protection Joint Experiment (FPJE)

C.M. Barngrover, R.T. Laird, T.A. Kramer, J.R. Cruickshanks, S.H. Cutler, A.F. Nans

Space and Naval Warfare Systems Center, San Diego (SSC SD)

53560 Hull Street, San Diego, CA 92152

Office: 619-553-9905

Fax: 619-553-6188

barngrov@spawar.navy.mil

http://www.spawar.navy.mil/robots

ABSTRACT

The FPJE was tasked to create a system of systems (SoS) within the realm of Force Protection. It was sponsored by Physical Security Equipment Action Group (PSEAG) and Joint Program Manager – Guardian (JPM-G), and was managed by the Product Manager - Force Protection Systems (PM-FPS). The experiment attempted to understand the challenges of integrating disparate systems into a cohesive, operationally effective SoS, and the resulting additional challenges of managing data flow into the system and its dispersion to all subscribed Command and Control (C2) nodes.

AUVSI Unmanned Systems Europe, La Spezia, Italy, 20-22 April, 2009

To handle this data flow we created the Data Fusion Engine (DFE), which receives and publishes data via the Security Equipment Integration Working Group (SEIWG) ICD-0100 protocol. With the DFE distributing information to each C2 Node, adding distributed control of all the resources was the logical next step. Providing control to all C2 Nodes via the ICD-0100 protocol was relatively trivial compared to the problems that arise from this shared control. This paper discusses the problems associated with distributed control between multiple operators and control inputs generated by automated processes. It also discusses the solutions that have since been designed and implemented, as well as the problems remaining and possible solutions for the future.

1. BACKGROUND

The primary objective of the FPJE over the one year period was to evaluate the impact of data fusion and automation on system performance and in turn to evaluate the overall system performance. The experiment began with the sensor selection and the subsequent integration of each sensor into the system of systems (SoS). The integration of these sensors was done in tiers during the first and second Integration Assessment events. Through most of the experiment, the system consisted of dual command and control interfaces through JBC2S and TASS. The integration and testing of all the sensors into the system culminated in the third Integration Assessment (IA-III). During IA-III there were multiple JBC2S stations and a TASS station that shared data in a star topology. It was during this event that it became clear that the shear amount of data coming into the system was a hindrance to the operator. Furthermore, it was clear that having two different C2 applications would increase operator training and decrease operator effectiveness. The focus of this paper is the work that was done in preparation for IA-IV as well as the continuing work that has been done since the conclusion of the experiment. The main efforts for the final IA included work on the DFE and work

on a module to replace the functionality of TASS that was not handled by JBC2S, specifically sensor and camera integration. The DFE was built on the infrastructure of JBC2S and its main roles were data distribution, data fusion, networked control, and automation. The module to replace TASS integration capabilities is called the Tactical Device Manager (TDM) and specifically communicates with TASS Sensors, Wide-Area Surveillance Thermal Imager (WSTI) and Long Range Thermal Imager (LRTI) cameras, and Battlefield Anti-Intrusion System (BAIS) Sensors.

2. JBC2S OVERVIEW

2.1 Background

The Joint Battlespace Command and Control System (JBC2S) has a long history beginning with the Multi-Robot Operator Control Unit (MOCU), which was originally developed as a tactical operator control unit for a small man-portable UGV called the URBOT. Since its start MOCU has evolved to support many other unmanned systems including the SPARTAN unmanned surface vehicle (USV) and the iRobot Packbot. MOCU is designed to support new protocols, video CODECS, mapping engines, and other components without modifying the core software through the use of modules with predefined interfaces.

JBC2S branched off of MOCU as a result of a need for integrated control of unmanned systems as well as other sensors including camera, radars, ground sensors, etc. As JBC2S evolved it also incorporated a new protocol based on the Multiple Resource Host Architecture (MRHA), which was developed at SSC Pacific as the command-and-control system for the Army's Mobile Detection Assessment Response System (MDARS) program. MDARS is a fixed-site robotic security system that consists of multiple semi-autonomous unmanned ground vehicles (UGVs) performing

automated patrols, as well as intrusion sensors and active barriers. This new protocol allowed for most of the MRHA functionality to exist in JBC2S.

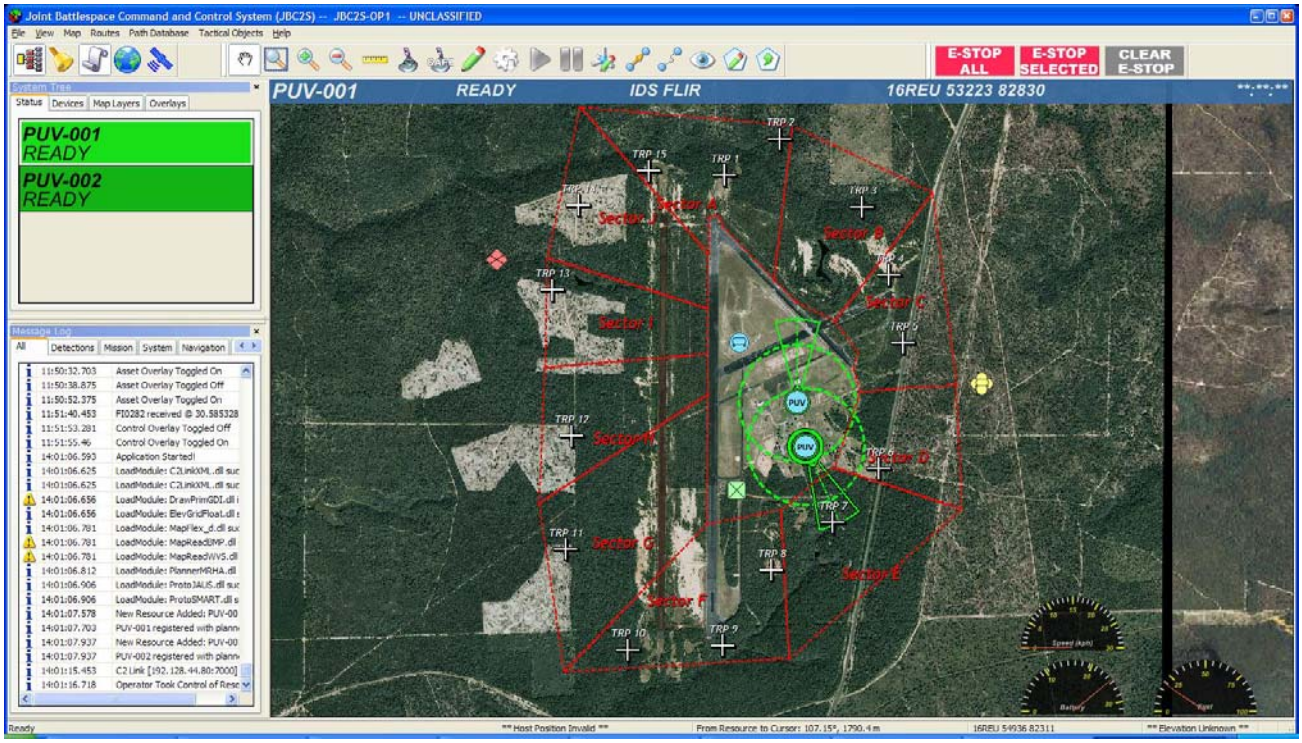

Figure 1: JBC2S software with two MDARS UGVs. This screenshot shows the ability to have sectors, points of interests, and targets. Each UGV also has a camera field of view and a radar coverage arc.

2.2 C2 Link Module

The Command and Control (C2) Link Module is a TCP/IP based framework for communicating protocols between control software. The protocol used for C2 communication during this experiment was the SEIWG ICD-0100 protocol. This XML based protocol was first adopted in 2006 to demonstrate integration with the US Army's Counter Rocket Artillery Mortar (C-RAM) program. In the C-RAM demonstration, JBC2S would deliver and receive basic XML messages to and from the Enhanced Tactical Automated Security System (eTASS), which would then interface with other Army C2 Systems.

The C2 Link Module creates and manages a series of client and server sockets, as well as all persistent TCP sockets from current links. Each persistent TCP socket has its own reference to a protocol parser; in this case all sockets used the ICD-0100 protocol parser. There are two main data flows through the C2 Link Module. First, when data changes within the core of JBC2S (i.e. robot location), the C2 Link Module is notified. Then the C2 Link Module will create the appropriate XML message for each socket and push the message through. Conversely, if a full XML message is received across one of the sockets, the message is parsed and the relevant information is passed through to the core of JBC2S. Entering into the FPJE, this module had been thoroughly tested with regards to connectivity, data integrity, data throughput, and stability.

2.3 Control Methodology

In order to simplify the user interface, JBC2S operates under a very simple rule set. The system is either in Monitor Mode, in which the operator observes and monitors the status of the unmanned vehicles and sensors (known collectively as resources), or the operator is in Direct Control of a single resource, such as a single MDARS robot. In addition, the robot itself carries a number of systems, each of which must also be controlled individually. For example, MDARS may have eight cameras, two radars, and four lights, but only a single camera, radar, and light is controllable at any given time. Multiple cameras can be viewed simultaneously, but the operator cannot pan two of them at the same time. A joystick button or dialog button can be configured to cycle between the various payloads. The control of the resource and each of its selected sub-resources is integrated in such a way that a single joystick can be used to perform all relevant joystick-requiring tasks. For example, the operator can pan the selected camera with the joystick, and then press a button to put

the robot in teleoperation mode and drive the whole robot using the same joystick. The concept of modes is used to provide rich control features for each resource. This paradigm reduces the complexity of the software and simplifies the user interface presented, while still giving the operator ample control of each asset.

3. Distributed Control

3.1 Framework

Following the third IA it was clear that we needed a way to remove clutter from the screen to remove burden from the operators. Given our time and funding constraints, we decided to solve this problem in-house, rather than through a contractor, as a proof of concept for data fusion benefits. We decided to build upon the framework of JBC2S and its C2 Link Module. The DFE would consist of an in-memory database of the state of the system, a Data Fusion Module, and a C2 Link Module. The first step was pulling out the in-memory database and C2 Link Module of JBC2S. Next, we focused on implementing the Fusion Engine, Response Engine, and data distribution. The Fusion Engine combined related data so it could be provided to the operator in a more concise format. The Response Engine allowed for automation of the system based on incoming data and the general state of the system. Data distribution was the method by which multiple operator stations could simultaneously receive pertinent fused data. The data distribution through a central server also provided the possibility for distributed control of assets among multiple operator stations.

In order to allow for distributed control, we implemented the Command Message paradigm of the ICD-0100 protocol, including many extended commands where enumerated commands did not suffice. During the early events of the experiment, we only had shared ability to send the simple

"GoTo" command, which caused a camera device to look at the location and caused a UGV to physically move to the location. Later in the event we implemented other major controls for assets through ICD-0100, including pan, tilt, zoom, secure, access, and day/night. With this version of JBC2S, any operator could take control of an asset and utilize its capabilities, which improved work load distribution and response times. We found that it still made sense for the operator to have primary assets to begin with, but if one operator was busy with something in a sector, another operator could control a nearby asset for over-watch and redundant visual of the target.

3.2 Lessons Learned from FPJE

The problem that quickly arose due to the shared control of assets was conflict of control. The underlying problem was that assets did not have a sense of who was in control. Because of this characteristic of the system, any resource would accept and act upon commands from multiple JBC2S Operators at the same time. The most common symptom of this problem was a camera jumping from one location to another, never really settling because it was constantly receiving conflicting commands. It became evident that there needed to be some sort of mechanism for locking a resource when an operator takes control and therefore some management of this state at the resource level.

3.3 Joint Force Protection Advanced Security System (JFPASS)

Following the FPJE, work began on the JFPASS project utilizing the same JBC2S software. The JFPASS project is a Joint Capability Technology Demonstration (JCTD) with a two year timeline. It has many of the same requirements as the FPJE including distributed control of resources.

3.3.1 Resource Locking Paradigm

One of the first efforts was to remedy the conflicting control problems created by distributed control during the final FPJE event. So we began to implement a protocol for requesting and releasing control as well as providing control state in status messages. In the ICD-0100 protocol, we created extended commands called RequestControl and ReleaseControl for this purpose. An operator can request control of a given resource, which sends an ICD-0100 message to the resource itself. The resource then provides a response to the command indicating whether or not control is granted. If a response is not received before a timeout threshold, then a failure to take control is shown to the operator. There are multiple, disparate reasons why a response may time out. One reason is that the resource software or sensor may have a software or hardware problem. Another reason is that the system is operational, yet network or software latency causes the response to be outside of the threshold.

Similar to the RequestControl paradigm, a ReleaseControl message is sent via ICD-0100 to the resource, which then responds with a confirmation that control has been released. However, we found that if no response was received, it did not make sense to lock up JBC2S with a non-responsive resource. So if there is no response, JBC2S still releases control in the GUI so that the operator can begin another process. In this problematic scenario, the resource may still believe that it is being controlled, and therefore may not allow other JBC2S operators to take control. This problem and its solution will be discussed in a later section.

In addition to the locking command messages, we also created a status field to provide the name of the system that is currently in control of a resource. This is useful from a GUI standpoint for

showing an operator which resources are available for control and also for viewing who currently has control. JBC2S currently has one main list of all assets in the system, and this feature allows us to show a brown lock icon next to any asset that is currently locked, with a blue lock for the resource currently being controlled by the operator. A further mouse-over will show the operator who has control, in addition to other status information.

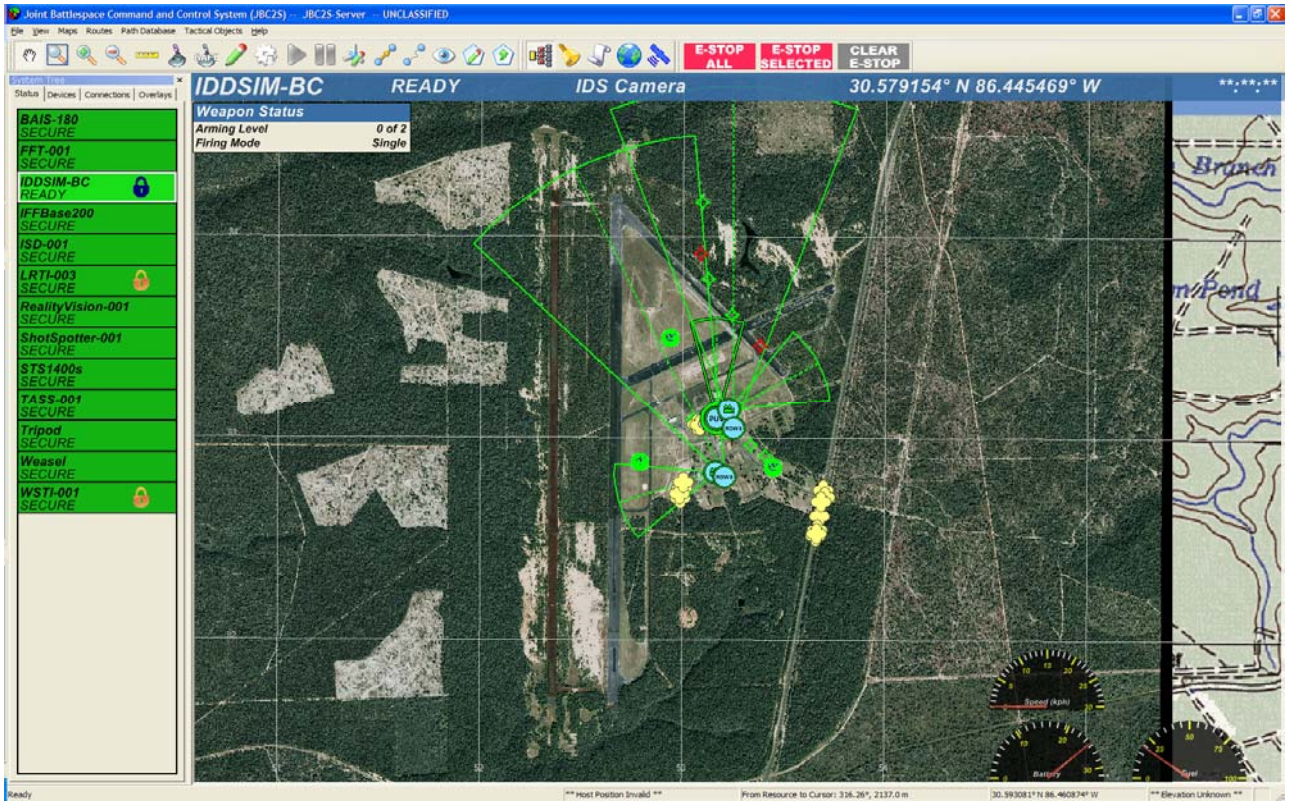

Figure 2: JBC2S Software with various resources as shown in the green list at the left. IDDSIM-BC is currently being controlled by this operator station so it has a blue lock. LRTI-003 and WSTI-001 are also being controlled, but since they are controlled by other stations, they have the brown lock. These indicators allow the operator to quickly assess which assets are available.

3.3.2 New Distributed Controls Implemented in JFPASS

We implemented many more commands in ICD-0100 and revamped the existing extended commands for optimal messaging. One important addition is teleoperation of Unmanned Ground Vehicles (UGVs) through the ICD-0100 protocol. Previously, teleoperation of the robot could only be accomplished at the operator station connected directly to the robot, along with the ability to change the robot mode (for example, from patrol mode to sentry mode). Now any operator at any station can perform these tasks, which is especially useful during the more intense moments of an operation where a single operator cannot possibly do everything required to cover his sector.

Other commands were added to support the control and remote firing of Networked Remotely Operated Weapon Systems (NROWS). Any operator at any station can now enable, arm, and fire a remote weapon. In addition, the weapon firing mode can be changed to single, burst shot (with an operator-configurable number of rounds per burst), and fully automatic modes. The operator can also set soft limits for the weapon based on the battlespace awareness provided by JBC2S (including information from blue force trackers). That way, he can keep the weapon from pointing at any friendly buildings, vehicles or people.

Because control is shared across the entire JFPASS system, any computer running JBC2S with sufficient permissions can control any asset. For example, the quick reaction force (QRF) team can carry with them a laptop that connects to the system via wireless access points. A QRF member could take control of a camera with a better viewing angle than his and gain a decided advantage over the enemy.

3.3.3 Latency Issues

One difficulty with the current locking paradigm is that it can severely hinder operations on a slightly latent network. Due to the enormous amounts of data flowing during an operation, it is possible for a command message to take up to 20 seconds to reach the targeted resource. The following example illustrates this problem: an operator requests control of a camera to view new radar contacts that have appeared at the edge of the base. Other operators are frantically trying to train their cameras on the intruders and this slows down the system. The control request arrives at the resource a full ten seconds after the operator sent it. The resource, unaware of the delay, responds positively to the request and locks the resource. Unfortunately, the timeout period has passed and JBC2S is no longer expecting a control request response, and when it arrives, the user is not granted control. However, the device is now locked and no one can use it. Our current solution is to allow a single operator station to retake control of devices it already supposedly has control over, and thus the operator can take control when there is less lag and release the resource for others to use. This clearly has the major disadvantage that it completely eliminates the use of a resource in the heat of battle (when there is heavy network traffic). Other solutions are being looked into, including the use of time synchronization between the resource and JBC2S to determine whether a device should be locked.

3.4 Future Efforts

The distributed control through JBC2S as part of JFPASS is nearly complete, allowing full control of all the assets in the system and including a locking paradigm to avoid conflicts. One feature that could improve the distributed control is the concept of priority. Operators with higher priority should be able to force another operator to release control. This will also help alleviate some of the

symptoms of latency issues causing a resource not to be released properly. In addition, there could be a timeout for control of an asset. Not that the system will automatically release control at the timeout, but that if a request is received and the timeout has been passed, then the release is forced, or the operator is notified of his long period of control and given the option to release to another operator. Also, if a lower priority operator requests control of a locked asset, a message can be sent to the controlling operator in case the lock is no longer needed. As you can see, all of the remaining efforts to consider involve more robust locking and unlocking, because the basic locking functionality is already complete.

3. Conclusion

From the FPJE to JFPASS, the abilities for distributed control have advanced to a more stable and useful state. This ability has already been demonstrated at multiple events with great success. The additional robustness for the locking paradigm will make the system more flexible as a whole, but the current configuration is functional. Furthermore, it has been clear from our assessments of the events that this distributed control better utilizes the operators and helps to spread workload among them. As JFPASS continues we will stress the system and its distributed control abilities even further so that the culmination of the JCTD will produce a tested and viable command and control system with full distributed control of its assets among all operators.