Mobile Agent-Based Distributed Fusion (MADFUSION) System

Joel Landis and Li Bai*,
Department of Electrical and Computer Engineering
Temple University, Philadelphia, PA 19122
*Email: lbai@temple.edu

John Salerno and Michael Hinman and Douglas Boulware
Information Directorate
Air Force Research Laboratories
Rome, New York 13441
† Email:hinmanm@rl.af.mil

***Abstract*— This paper considers a system architecture referred to as the Mobile Agent-Based Distributed Fusion (MADFUSION) system. The system environment consists of a peer-to-peer ad-hoc network in which information may be dynamically distributed and collected via publish/subscribe functionality implemented at each node of the network to facilitate data sharing and decision making in Level 2 Fusion. The Level 2 decision making process implemented in the system consists of the Enhanced Doctrinal Template Matching (EDTM) algorithm which is shown to be an improvement over the pre-existing Doctrinal Template Matching algorithm. This algorithm is developed to operates on information obtained from lower layer fusion processes in order to identify aggregated groups of entities. The template matching algorithm is shown to be an improvement over a previously existing algorithm. The MADFUSION system is proposed to extend the client/server architecture of various publish/subscribe applications to an architecture providing decentralization, reconfigurability, mobility, attainability and prevention of single points of failure. The system is implemented in a wireless ad-hoc network (802.11b) and performs the publish/subscribe functionality through the implementation of a mobile agent based framework. The software agents travel deterministically from node-to-node carrying a data payload consisting of information which may be subscribed to by users within the network. Within this system, situation awareness (Level 2 fusion) can be sought by using these multi-domain information sources (GMTI, Video, or SAR) for evaluation at each node with different distributed information fusion algorithms.**

I. INTRODUCTION

The modern intelligence and electronic warfare (IEW) [1] depends on tactical situation awareness for mission success in order to dominate across the operational spectrum [2]. The situation awareness with rapid response must be backboned by a reliable and real-time supported information managements system operated in a secure collaborative networked environment [3]. An example of such an information management system is the Joint Battlespace Infosphere (JBI) [4]. The advantage of these systems is in their ability to create and maintain a common operating picture for decision support in situation awareness at multiple heterogonous information sources. It is essential that higher level fusion applications must rely upon an integrated information management system to ensure rapid responses from multiple heterogeneous information sources, or sensor payloads. These sensors can range from communication intelligence, electronic intelligence, imagery intelligence and measurement and signature intelligence (MASINT) in electro-optic (EO), infrared (IR), synthetic aperture radar (SAR), or ground moving target indicator (GMTI). These multiple arrays of sensor information place many constraints on the information fusion capability. Salerno *et. al.* [3] advocated a new approach to combining two information models: Joint Directors of Laboratories (JDL)'s fusion model and Endsley's situation awareness model. The lower layer of information extraction is base upon the information management systems such as [4]. The work presented in this paper seeks to establish an information fusion process in the level 2 situation awareness using multiple arrays of sensory data. Two different convoy aggregate identification algorithms are compared in classifying the military convoys. We assume that the individual entities have been identified with corresponding confidence levels by lower level Fusion processes prior to application of the convoy aggregate identification algorithms.

Previous research has concentrated on the identification of vehicles that are members of the same group using GMTI sensors [5]. If other sensors can more accurately report the vehicle type (e.g. SAR), we can then design an algorithm to match the observed groups with the correct convoy templates. The challenge is to establish these algorithms in a distributed environment using the multiple types of sensor reports. This requires mobility of the system and a template matching algorithm. The focus of the work presented in this paper consists of the following:

1. develop a miniature mobile information management sensor report system
2. implement a convoy template matching algorithm through combining sensor reports

In addition, mobile information management sensor reports can accurately report distributed sensor information. For example, two information sources, sensor A and sensor B, can construct different hypotheses about the same observed event due to the fact that the event information captured by sensor A differs from the event information captured by sensor B. This can result when sensor A possesses a different view of the event than sensor B. Information source A must then deliver its observations to the other sensor (sensor B) and/or vice versa. Once B receives the observations from A the new sensor report based upon the combination of the individual sensor reports can be evaluated to determine if the sensor reports agree or

0-7803-9286-8/05/$20.00 © 2005 IEEE

disagree with each other.

The system is implemented on a peer-to-peer wireless ad hoc network where the information management functional tasks of *publish* and *subscribe* are carried out through the use of software agents. The use of software agents in such an environment has received a great deal of research and has been shown by such works as [6] to provide a more robust system architecture than the traditional client/server architecture. Such an environment presents a number of metrics that need to be considered when comparing agent-based architectures with client/server architectures (network bandwidth, message and agent size, power management, security, etc.). Studies on the comparison between agent-based and client/server architectures with regard to these metrics have been done by [7], [8]

The rest of the paper is organized as follows. Section II gives the detailed description of the system implementation, and Section III discusses the template matching algorithm used to identify the vehicle convoys. Section IV presents simulation results in the comparison to the two defined template matching algorithms. The conclusion and future work are given in Section V.

II. System Implementation

The purpose of the MADFUSION system is to distribute information observations across the Mobile Ad-Hoc Network (MANET) using a mobile agent-based framework for implementing publish/subscribe functionality. The agent-based framework provides a simplified design for the distributed systems such as portable devices or cellular devices. The network socket layer protocol and system concurrent consistence was shielded inside the agent-based implementation. This system is implemented in an 802.11b (802.11g) ad-hoc wireless network which consists of three Dell laptops and three LinkSys wireless adapter cards. Among these three laptops, two are running Windows XP operation system and one is running in Linux Redhat 9.0 operating system.

The information sources are Voice, GMTI, Video and SAR images. An information exchange session can often begin with a spoken request with the resulting voice stream delivered by mobile agents to the recipients. The agent roams around the route carrying an information payload and decides who should receive the information using the payload's meta-data. Any computer is capable of dispatching a mobile agent to deliver information data or make a decision based on the information it observes.

The mobile agent-based framework is based upon Lockheed Martin's Extensive Mobile Agent Architecture (EMAA) [9]. EMAA provides three different ways of building mobile agents:

1. Identifiable Agent, an agent has a state identification number and can perform one or more tasks in different hosts.
2. Composable Agent, an agent has a state with tasks and an agenda of where it should travel inside the network.
3. Standard Agent, an agent does not have a state but rather an ability to roam into another host

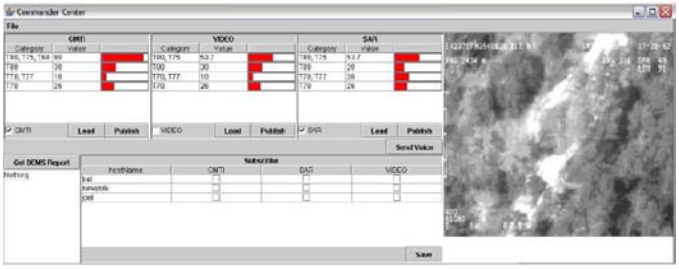


Fig. 1. Distributed System Interface

The composable agent is simplest way of creating an agent because it is programmed with two components, 1) agent tasks and 2) agent itinerary. This approach allows the main program to concentrate on what the agent should perform in the agent tasks. In this project, the mobile agents act like a miniaturized JBI system. Two tasks are specified for the mobile agents 1) publish task and 2) subscribe task which are shown in Figure 1.

The mobile agent will visit all N hosts, and begin by getting the published data from the first host. Any of the N-1 hosts may call the mobile agent by performing the subscribe task in order to extract the data published from the first host. In the end, the agent returns back to the first host to show what data has been subscribed to by the other hosts. The agent then goes to "Hault" and dies in the network.

In this approach, we can see many similarities to the JBI architecture of a publish/subscribe system. However, the centralized JBI system will present the problem of single point failure. The failure of the JBI system can cause a hault of all information flow to JBI clients. The mobile agent, however, checks the next host's availability. If the next host fails to respond in time, the mobile agent will roam to the next available host. Even if some of the systems fail, the mobile agent can still extract information from other systems.

During the publish task, the agent wraps the data object with meta-description and payload binary data. In the subscribe task, the agent can check whether the visited host is subscribed to the data object by extracting and comparing the meta-description.

Figure 1 illustrates one interface on one of the three Dell laptops with three sources of information which can be published by this computing node. The right hand side of the panel displays the information delivered to the computing node by the agents from other distributed fusion nodes. The dispatched display has a tank which moves through the forest.

As the "Get DEMS Report" button is clicked in this interface shown in Figure 1, the ranking algorithm based upon the belief measures assigned to each vehicle are used to calculate new decisions based upon the selection of the system and data received from other computation nodes. The clustered fusion algorithm (template matching algorithm) is a modified version of an information fusion project developed at Carnegie Mellon University known as doctrine template matching [10]. The fusion results are combined using this algorithm written in

1: Given a template $T_i = \{\{V_1, N_1\}, \ldots, \{V_p, N_p\}\}$ and a cluster of vehicles $CL = \{\{V_1', m_1\}, \ldots, \{V_q', m_q\}\}$, for any vehicle type V_i, N_i is the number of the type N_i in the template; m_i is the belief function for V_i' in the cluster CL. Initially $conflict = 0$.
2: **for** $V_i' \in CL$ **do**
3: Select a matched type of vehicle V_j in the template
4: **if** (V_j is found in template T_i) and ($N_j > 0$) **then**
5: $CL = CL - \{V_i', m_i\}$; $N_j = N_j - 1$
6: conflict = conflict + $m_i(\{V_i'\})$
7: **else**
8: $CL = CL - \{V_i', m_i\}$
9: conflict = conflict + 1
10: **end if**
11: **end for**
12: **if** $|CL| = 0$ **then**
13: return conflict
14: **else**
15: conflict = conflict + $|CL|$
16: return conflict
17: **end if**

Fig. 2. Doctrinal Template Matching Algorithm

1: Given a template $T_i = \{V_1, V_2, \ldots, V_p\}$ and a cluster of vehicles $CL = \{\{V_1', m_1\}, \{V_2', m_2\}, \ldots, \{V_q', m_q\}\}$, m_j is the confidence measure corresponding to vehicle V_j'.
2: score = 0
3: **for** $V_j' \in CL$ **do**
4: Select a matched type of vehicle V_j in the template.
5: **if** (V_j' is found in template T_i) **then**
6: score = score + $m_j(\{V_j'\})$
7: **end if**
8: **end for**
9: score = $\dfrac{score}{\max(p, q)}$

Fig. 3. Enhanced Doctrinal Template Matching Algorithm

Java and displayed in the window below the button. Each data source can give a matched result. In the case of conflict, the computing node can request for more data or decisions from other computing nodes in the distributed wireless network. As more data is collected, a better overall decision can be made.

III. Template Matching Algorithms

In this study we assume that an aggregated cluster of observed vehicles have been detected and are represented as a cluster report. Each cluster report consists of the number of vehicles that have been detected, along with the types of vehicle each observed vehicle is believed to be. The confidence given for the type of each vehicle is represented by a number in the range 0-1, where the value 0 represents no confidence and the value 1 represents complete confidence. We define this number as the *belief measure* of an observed vehicle. In addition to cluster reports, there also exists a database of predefined convoy templates. Each convoy template defines a single type of vehicle convoy by listing the type and number of vehicles present in a given convoy.

With this information it is necessary to decide which convoy templates best matches the current situation. This decision is made automated by implementing an algorithm that matches the observed group of vehicles with predetermined vehicle convoy templates. This work can be added upon previous work on convoy cluster identification algorithms [5]. Here we implemented a convoy template matching algorithm to demonstrate the distributed and automated decision making process. Our algorithm is based on the Doctrinal Template Matching algorithm developed by a research group from Carnegie Mellon [10]. This algorithm, given in Figure 2, is based on assigning conflict values when elements of the templates do not match the given cluster. This algorithm then chooses the template with the lowest conflict value as the matching template. There are, however, two shortcomings to this algorithm. The first shortcoming is given by the fact that a cluster element with a small belief measure contributes less to the conflict score than a cluster element with a larger belief measure. This characteristic penalizes cluster elements who's identity is more certain with a larger contribution to the overall conflict. Another shortcoming to the algorithm is the fact that the algorithm cannot distinguish between two templates when one template is a subset of the other template and the vehicle cluster matches the subset. We seek to develop an improved template matching algorithm that addresses these shortcomings.

The Enhanced Doctrinal Template Matching (EDTM) algorithm offers improved performance by better distinguishing between templates which may be similar, or in templates which may be subsets of larger templates. This new algorithm allows for more specific templates to be implemented which, in turn, provides a more specific and detailed representations of the current situation to be defined than what would have been possible using the Doctrinal Template Matching algorithm. These improvements are shown to be significant by the simulation results presented in the following section. The Enhanced Doctrinal Template Matching Algorithm, is illustrated in Figure 3 and defined further in the following.

Given a list of templates and an observation cluster of vehicles, the Enhanced Doctrinal Template Matching algorithm generates a matching score for each of the templates. The matching template is determined to be the template which possesses the highest score. A convoy template, T_i, can be described as $T_i = \{V_1, V_2, \ldots V_p\}$ where V_j is the type of the j^{th} vehicle present in the template. For example, an Army division convoy division T_{X1} has two T80 tanks and four trucks. Using the above notation, the template for this convoy division is represented as T_{X1} = {T80, T80, Truck, Truck, Truck, Truck}. An observed cluster of vehicles, C, is denoted by C = $\{\{V_1', m_1\}, \{V_2', m_2\}, \ldots, \{V_q', m_q\}\}$ where V_i' is the vehicle type and m_i is the corresponding belief measure where $0 \leq m_i \leq 1$. Take as an example a SAR sensor report with four observed vehicles; T80 with 70% belief, T80 with 60%, T80 with 55% and one truck with 90% belief. Using the above notation we can represent the cluster report as C_{SAR} = {{T80, 0.7},{T80, 0.6},{T80,0.55},{Truck,0.9}}. The template matching algorithm gives a score to describe how close the cluster data matches the template. The algorithm is given in Figure 3. According to the above algorithm, the matching score is (0.7 + 0.6 + 0.9)/6 = 0.367. The

highest score will represent the highest correlation between the vehicle cluster and the individual template. As is evident, the Enhanced Doctrinal Template Matching Algorithm addresses the shortcomings identified in the Doctrinal Template Matching Algorithm. The next section provides a performance comparison between the two template matching algorithms.

IV. Simulation Results

In this section we present a comparison of the two template matching algorithms. The algorithms were compared in their performance in identifying randomly generated clusters from randomly generated templates of prescribed similarity to each other. One simulation trial consists of the following steps:

1. Generate one template by choosing a random size and populating it by randomly choosing one item at a time out of ten possible items.
2. Randomly generate five more templates, each with prescribed similarity to the other generated templates.
3. Randomly pick one of the five templates and remember which template was picked.
4. From the chosen template, generate a cluster of random size and prescribed similarity to the chosen template using only items from the chosen template.
5. Perform both matching algorithms and determine if they correctly choose the matching template to be the template used for deriving the cluster.

We define the following expression for representing similarity between clusters and templates as values in the range [0-1], where 0 corresponds to no similarity and 1 corresponds to templates and clusters which are the same:

$$Similarity = \frac{Ti \cap Tj}{max(Size(Ti), Size(Tj))} \quad (1)$$

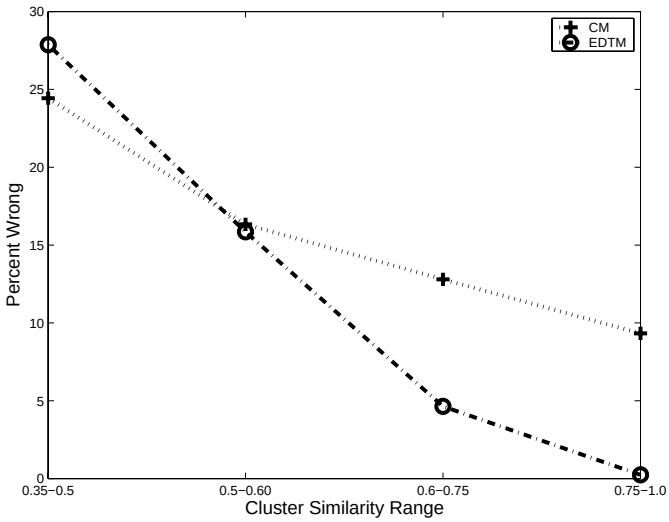
Figure 4 illustrates the performance of each algorithm to a number of different cluster and template similarity ranges.

The results of Figure 4 illustrate that both algorithms perform most optimal when the templates are less similar to each other and the similarity between the cluster and chosen template is high. Within this optimal range the Enhanced Doctrinal Template Matching Algorithm demonstrates significantly improved performance on the order of 10% in 4(a), 5% in 4(b) and 15% in 4(c) over the Doctrinal Template Matching Algorithm.

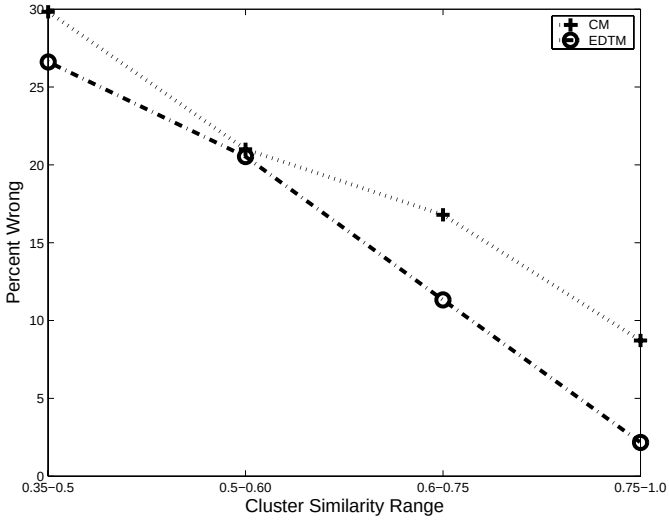
V. Conclusion and Future Work

In this paper it is demonstrated that the use of mobile agents extended from the information management concept can have an impact on distributed decision making. The fusion result has a rapid response and better decision. Most importantly, we can see that the distributed system does not have to rely on one system to disseminate information. The system provides a decision making functionality rather than a simple data source provider.

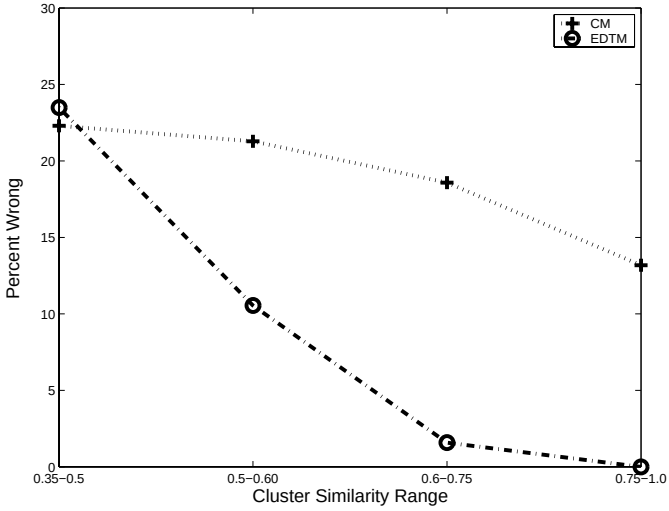
Additional work needs to be done for establishing publish and subscribe capabilities. The current system considers all computing nodes and information as publishable while all



(a) Broad template similarity range



(b) High template similarity range



(c) Low template similarity range

Fig. 4. Comparison of Matching Algorithms: Range of Template and Cluster Sizes: 5-20 (a) Template Belief Range: 0.4-0.7 (b) Template Belief Range: 0.55-0.8 (c) Template belief Range: 0.25-0.55

distributed systems can subscribe to any of the published information by retrieving it from the mobile agents' payload. In the future, a robust and dynamic subscribe functionality will be built into the system. The publish agent can compare its payload meta data with meta data from all distributed node subscriber's meta data. If the meta data matches the subscribers' request, the publish agent will drop off the payload information to the subscribing node. In addition, more fusion algorithms are sought for developing different fusion measure values in order to compare the effectiveness of the distributed decision making.

ACKNOWLEDGMENT

This research work has been support by National Research Council (NRC) and Air Force Research Laboratory (AFRL) Information Institute.

REFERENCES

[1] *Army Magazine, Intelligence and Electronic Warfare System Modernization*, Apr. 2002.

[2] E. P. Blasch and P. Hanselman, "Information fusion for information superiority," in *National Aerospace and Electronics Conference*, Dayton, OH, October 2000, pp. 290–297.

[3] J. Salerno, M. Hinman, and D. Boulware, "The many faces of situation awareness," presented at the Information Fusion Conf., Sweden, July 2004.

[4] *IEEE Aerosp. Electron. Syst. Mag, Programs in Higher Levels of Information Fusion*, vol. 18-11, Nov. 2003.

[5] L. Bai and M. L. Hinman, "Object aggregation using neyman-pearson analysis," in *Proceedings of the SPIE*, vol. 5099, Apr. 2003, pp. 201–210.

[6] W. Caripe, G. Cybenko, K. Moizumi, and R. S. Gray, "Network awareness and mobile-agent systems," *IEEE Communications Magazine*, vol. 36, no. 7, pp. 44–49, July 1998.

[7] Y. Jiao and A. R. Hurson, "Mobile agents in mobile heterogeneous database environment - performance and power consumption analysis," in *Proceedings of the Advanced Simulation Technologies Conference (ASTC)*, Arlington, Virginia, April 2004, pp. 185–190.

[8] E. Sultanik, D. Artz, G. Anderson, M. Kam, W. Regli, M. Peysakhov, J. Sevy, N. Belov, N. Morizio, and A. Mroczkowski, "Secure mobile agents on ad hoc wireless networks," in *The Fifteenth Innovative Applications of Artificial Intelligence Conference*. American Association for Artificial Intelligence, Aug 2003, pp. 129–36.

[9] R. P. Lentini, G. P. Rao, J. N. Thies, and J. Kay, "Emaa: An extendable mobile agent architecture," in *15th National Conference on Artificial Intelligence (AAAI'98)*, Madison, Wisconsin, 1998.

[10] B. Yu, K. Sycara, J. Giampapa, and S. Owens, "Uncertain information fusion for force aggregation and classification in airborne sensor networks," *American Association for Artificial Intelligence*, 2003.