PUC

ISSN 0103-9741

Monografias em Ciência da Computação
n° 11/12

Middleware Supporting Situational Awareness in Mission-Critical Scenarios with Rotorcraft

Gustavo Luiz Bastos Baptista

Markus Endler

José Viterbo Filho

Departamento de Informática

PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO DE JANEIRO

RUA MARQUÊS DE SÃO VICENTE, 225 - CEP 22451-900

RIO DE JANEIRO - BRASIL

Monografias em Ciência da Computação, No. 11/12 ISSN: 0103-9741
Editor: Prof. Carlos José Pereira de Lucena Aug, 2012

Middleware Supporting Situational Awareness in Mission-Critical Scenarios with Rotorcraft

Gustavo Luiz Bastos Baptista Markus Endler José Viterbo Filho
{gbaptista, endler}@inf.puc-rio.br viterbo@ic.uff.br

Abstract. Situational Awareness (SA) has a central role with rotorcraft operations in mission-critical scenarios, for operational performance, mission success, safety and survivability. Applications and systems that enhance SA impose requirements of real-time communication and processing of large volumes of data, with high throughput and low latency. This work presents a middleware architecture to support those applications, with capabilities such as real-time data-centric publish-subscribe communication, Quality of Service (QoS) management, Semantic Interoperability and Distributed Complex Event Processing for the detection of Situations of Interest (SoI). A realistic scenario of helicopter rescue missions for offshore drilling is presented, with a SoI to be detected by the system. A visualization tool is used to illustrate the capabilities provided to applications, and simulation results are presented that compare different event processing distribution models and abstraction levels, regarding their impact on performance and scalability.

Keywords: middleware, mission-critical applications, Situational Awareness, Complex Event Processing, CEP, Data Distribution Service, DDS

Resumo. Situational Awareness (SA) tem um papel central em operações de helicópteros em cenários de missão crítica, para desempenho operacional, sucesso de missões, segurança e sobrevivência. Aplicações e sistemas que aumentam SA impõe requisitos de comunicação e processamento de grandes volumes de dados em tempo real, com alta vazão e baixa latência. Este trabalho apresenta uma arquitetura de middleware para suportar estas aplicações, com capacidades tais como comunicação em tempo real em um modelo data-centric publish-subscribe, gerenciamento de Qualidade de Serviço (QoS), Interoperabilidade Semântica e Distributed Complex Event Processing, para a detecção de Situações de Interesse (SoIs). Um cenário realista de missões de resgate com helicópteros em explorações de petróleo é apresentado, com uma SoI a ser detectada pelo sistema. Uma ferramenta de visualização é utilizada para ilustrar as capacidades providas para aplicações, e resultados de simulações são apresentados que comparam diferentes modelos de distribuição e níveis de abstração de eventos, considerando seu impacto no desempenho e escalabilidade.

Palavras-chave: middleware, aplicações missão crítica, Situational Awareness, Complex Event Processing, CEP, Data Distribution Service, DDS

In charge of publications

Rosane Teles Lins Castilho
Assessoria de Biblioteca, Documentação e Informação
PUC-Rio Departamento de Informática
Rua Marquês de São Vicente, 225 - Gávea
22451-900 Rio de Janeiro RJ Brasil
Tel. +55 21 3527-1516 Fax: +55 21 3527-1530
E-mail: bib-di@inf.puc-rio.br
Web site: http://bib-di.inf.puc-rio.br/techreports/

Table of Contents

1 Introduction 1
- 1.1 Coordination in Mission-Critical Scenarios 1
- 1.2 Situational Awareness 1
  - 1.2.1 Situations of Interest 2
  - 1.2.2 Visualization 2
- 1.3 Goals 2

2 Base Technologies 2
- 2.1 OMG Data Distribution Service (DDS™) 3
- 2.2 Distributed Complex Event Processing 3
  - 2.2.1 Event Processing Agents and Event Processing Networks 4
  - 2.2.2 EPNs Distribution Models 4
  - 2.2.3 SoIs expressed as CEP Rules 5

3 Middleware Architecture 5
- 3.1 Quality of Service Management 5
- 3.2 Semantic Interoperability Support 6
- 3.3 Distributed Complex Event Processing Management 6
- 3.4 Net-Ready Applications With Visualization 7

4 Representative Scenario and Situation of Interest 7
- 4.1 Situation of Interest 7

5 Distributed Complex Event Processing Instantiation 8
- 5.1 Events Abstraction Levels 9

6 Performance Results 9

7 Visualization 10

8 Related Work 11

9 Conclusions 12

References 12

1 Introduction

1.1 Coordination in Mission-Critical Scenarios

Communication and coordination in mission-critical scenarios is demanding in that it involves requirements typical of critical situations, including the necessity for decision making and response under temporal and resource constraints. Different models of coordination are used by systems supporting these requirements, such as centralized command and control systems or decentralized collaboration systems.

Centralized command and control is typically present in fleet management, tactical systems, or mobile task force management systems, which are becoming essential to the coordination and optimization of people, vehicles or mobile assets to accomplish many sorts of objectives and missions. Those systems typically involve managing large groups of mobile nodes, which are equipped with portable or onboard computing devices with available wireless communication interfaces and embedded or onboard sensors (e.g. GPS, avionics sensors, etc), and access to a communication infrastructure for remote monitoring (e.g. tracking their current position). A control station typically receives information about the monitored assets in real-time, allowing rapid decisions and actions to be taken by control teams, such as modifying objectives, new targets, rerouting of task-force team members, and other major operational or tactical changes in a mission.

Another way to allow timely collaboration among the team members is to support decentralized, network centric mission communication and coordination. In this approach, each individual agent of the team performs its specific mission, but continuously shares data as needed with others so as to provide a more complete and accurate representation of the environment and better define its current role in a task force group. This type of collaboration appears suitable for emergency search-and-rescue missions (e.g. natural disasters) or tactical coordination (e.g. police or military patrol). In fact, such centralized and decentralized types of coordination can both be supported by command-and-control and collaboration systems at the same time, depending on the types of scenarios and missions. The main goal in these systems is to enhance Situational Awareness (SA) [1], both to control stations and to the mobile team participants.

1.2 Situational Awareness

Situational Awareness (SA) is a term coined in the aviation and military domains, and has many different definitions. According to [1], SA is "the perception of the elements in the environment within a volume of time and space, the comprehension of their meaning, and the projection of their status in the near future". For example, in rescue missions with rotorcraft, SA supports better coordination of rescue flights and landings, and also has a positive impact on the safety and survivability of the task force itself. SA is critical for helicopter pilots due to workload associated with conducting difficult missions with operational risks and environmental hazards [2]. In order to address SA enhancement it is necessary to have the scope of specific Situations of Interest (SoIs) that are important for the accomplishment of missions.

1.2.1 Situations of Interest

We consider as a Situation of Interest (SoI) a description of a spatio-temporal condition with regard to the context of monitored elements (e.g. their geographical position) and/or the environment state (e.g. the weather condition). The detection of a SoI usually demands a notification to interested parties (e.g. users or systems), either because it represents a desirable or undesirable/dangerous overall situation. A SoI can have a global or local focus, which can be respectively regarded as global SoI or local SoI. An example of global SoI is providing control station operators (or pilots on rotorcraft) overall situations related to a group of nodes (e.g. too many/too few helicopters in a region). An example of local SoI is showing a pilot combined local information from remaining fuel, weather data, obstacles and mission objectives.

1.2.2 Visualization

Visualization techniques play a central role in enhancing SA [3]. In the context of military and rescue missions, it generally involves the representation of geographic areas in maps or 3D environments, with the elements that are important to be monitored and considered for decisions in missions, or aiding individuals to perform specific tasks, for example, providing visual signals to a pilot during landing approaches. Appropriate visualization requires showing users the right information, at the right time, building a common picture of the operational area. Since the complexity and potential large number of elements and data sources (e.g. sensors and monitored nodes) can cause clutter to visualization, showing aggregations that combine large amounts of raw data is necessary.

1.3 Goals

In order to implement systems that support coordination and visualization enhancing SA, support is necessary from underlying middleware with mechanisms for real-time communication and processing of large volumes of data, with high throughput and low latency. In this work, we present a middleware architecture with such features, giving focus to its capabilities that allow the real time monitoring of nodes (e.g. rotorcraft) and the evaluation of general relations among mobile and stationary nodes, or virtual entities (e.g. arbitrary geographical points), in a scalable manner. After the description of base technologies in Section 2, the middleware architecture is presented in Section 3. A realistic scenario of helicopter emergency response missions in the offshore drilling domain is presented in Section 4, and 5 describes the application of this architecture to the context of the presented scenario. Performance results are then presented in Section 6 that compare simulations with different event processing distribution models and granularities for data abstraction levels, regarding their impact on scalability and performance.

2 Base Technologies

This section presents the technologies for event-based communication and processing which have been used as base for the middleware presented in this work.

2.1 OMG Data Distribution Service (DDS™)

For inter-process communication, the publish/subscribe model [9] is a well-established, robust and performance-enhancing, means of communication for open, distributed and mobile systems. Over the past few years, its capabilities have evolved from centralized publish/subscribe architectures to overlay networks of distributed brokers, leading to enhanced scalability and availability. More recently, the Data Distribution Service for Real-Time Systems (DDS™) [4] standard proposed a peer-to-peer (i.e. fully distributed and without broker nodes) data-centric publish/subscribe communication model. It provides high performance real-time communication, scalability and availability, and supports the specification of Quality of Service (QoS) contracts between data producers and consumers. It allows interoperability across different DDS™ implementations, programming languages and platforms, as well as automatic discovery of DDS™ publishers/subscribers. DDS™ is based on a Data Centric Publish/Subscribe model, where DDS Topics are logical entities defined to compose a distributed relational data model, also known as Global Shared Data Space. The Topics are the first class entities of information, which applications can publish or subscribe to, and can be regarded as distributed relational database tables. The DDS Domain, which contains all shared data, is fully distributed over the participating network nodes, without any intermediate broker or centralized management entity. Several commercial and open-source implementations of DDS™ are available, such as [5-7].

2.2 Distributed Complex Event Processing

In the same way as communication models evolved during the last years, processing models have evolved from Transactional Database Management Systems (DBMSs) that receive synchronous queries to asynchronous event processing systems. It turns out that the traditional database model is not adequate for applications with the aforementioned requirements of real-time communication and processing, high throughput and low latency. These applications are typically based on the processing of continuous streams of data, in general obtained from external sources (e.g. sensors), instead of humans submitting transactions. The main interaction model is asynchronous, e.g. the triggering of alerts when SoIs are detected. Event processing systems receive continuous streams of data (i.e. event streams), continuously process these data and send asynchronous notifications about detected situations (i.e. events representing SoIs) [12]. Actually, both asynchronous communication and processing models are complementary, in a way that typically event-processing systems use some sort of asynchronous communication (e.g. publish/subscribe).

Complex Event Processing (CEP) [8] extends the capabilities of the content-based publish subscribe model [9], with the capacity to specify relationships not only over event properties, but also relationships between different events, causality, temporality, sequencing, aggregation and composition. A complex event is thus a higher-level abstraction representing a situation derived from the occurrence of more elementary events. Causal maps can be associated to these events, allowing a complete tracking of the event causes (i.e. the sequence of events that caused an event to happen). The temporal relationships between events allow the processing of event sequences within specified time windows. It is also possible to evaluate aggregation functions over event properties observed in sets of events, such as the average, maximum or minimum property values of the observed event set. All these features allow the definition of powerful event processing rules that express application-relevant patterns of events and their relationships [10]. Many commercial and open-source implementations of CEP are available, such as [11-15].

2.2.1 Event Processing Agents and Event Processing Networks

An Event Processing Network (EPN) is a conceptual representation of an event processing system in a platform independent way. An EPN is composed of Event Processing Agents (EPAs), which receive event streams as input and process these events with different operations (e.g. filtering, aggregation, transformation, pattern detection, etc) producing events as output, as the result of this processing. In an EPN, the EPAs are conceptually connected to each other (i.e. output events from one EPA are received and further processed by other EPAs), without regard to the particular details and type of the underlying communication mechanism (e.g. push- or pull-based) used to transfer events between each other. The EPAs organized in an EPN implement the whole processing logic of situation detection through event processing [16].

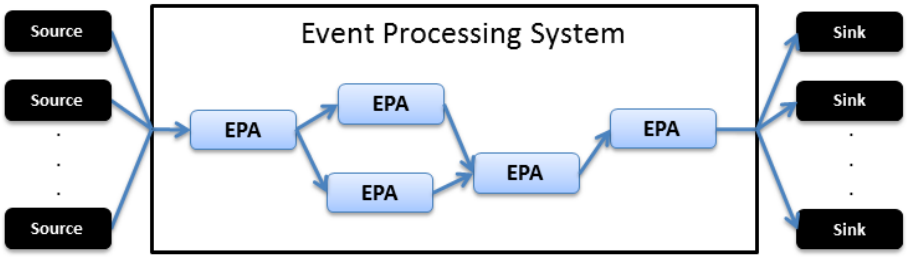

Figure 1 - Example of Event Processing System with Event Processing Agents (EPAs) organized to form an Event Processing Network (EPN) [16].

2.2.2 EPNs Distribution Models

Event processing systems (e.g. CEP systems) implement the concepts of EPNs in slightly different ways, but one important aspect that has the largest impact on scalability is the deployment model of the EPN (centralized vs. distributed) [10]. The distribution of the event processing architecture is a key aspect to allow scalability on the number of data producers and data consumers, and also scalability in the number of situations of interest to be detected from large amounts of events flowing through the system [17].

In a centralized event processing architecture, the processing of event streams is performed by only one centralized node in a computer network, which implements all processing logic or has all EPAs (i.e. the whole EPN) locally deployed. On the other hand, in a distributed architecture, the event flows are processed by EPAs deployed at different computer network nodes interconnected by a communication infrastructure.

Distributed event processing architectures are divided in two categories: clustered and networked. In the clustered model, EPAs are deployed in a cluster, where nodes that are tightly coupled by a fast and reliable network are within the same administrative domain. These clusters benefit from parallel event processing, but also require high network bandwidth between the cluster nodes and the remote producers and consumer of events. Networked architectures, on the other hand, focus on minimizing network bandwidth usage by deploying EPAs dispersed in a Wide Area Network (WAN), but closer to event producers and consumers. The placement decision for the EPAs of a particular application can balance different criteria, such as geographical location of event sources and sinks, network throughput and reliability functionality segmentation, etc [10]. Different academic and commercial distributed event-processing systems are available. The majority of them use a clustered deployment [11; 12; 15], with few implementing a networked solution[13; 18].

2.2.3 SoIs expressed as CEP Rules

The capabilities of processing event streams and the recognition of event patterns on real time provided by the CEP model make it very well suitable for the detection of SoIs, to enhance SA. Rules can be deployed on EPAs to compose EPNs, which trigger complex events representing the detected SoIs. These complex events can be captured by applications, systems or used by other rules to detect many abstraction levels of SoIs.

3 Middleware Architecture

A set of capabilities for applications were implemented, all with a network-centric approach for real-time data sharing among rotorcraft nodes [2]. Middleware APIs, encapsulating the use of DDS™ for data-centric publish/subscribe communication, were developed supporting capabilities, such as Networked Weather, Uncharted Obstacles, Own Ship Position Reporting, and Aids for Landing Operations. The API methods allow applications to interact with the DDS Global Shared Data Space synchronously or asynchronously, without requiring the application programmer to deal with DDS™-specific entities or to have deep knowledge about the DDS™ specification.

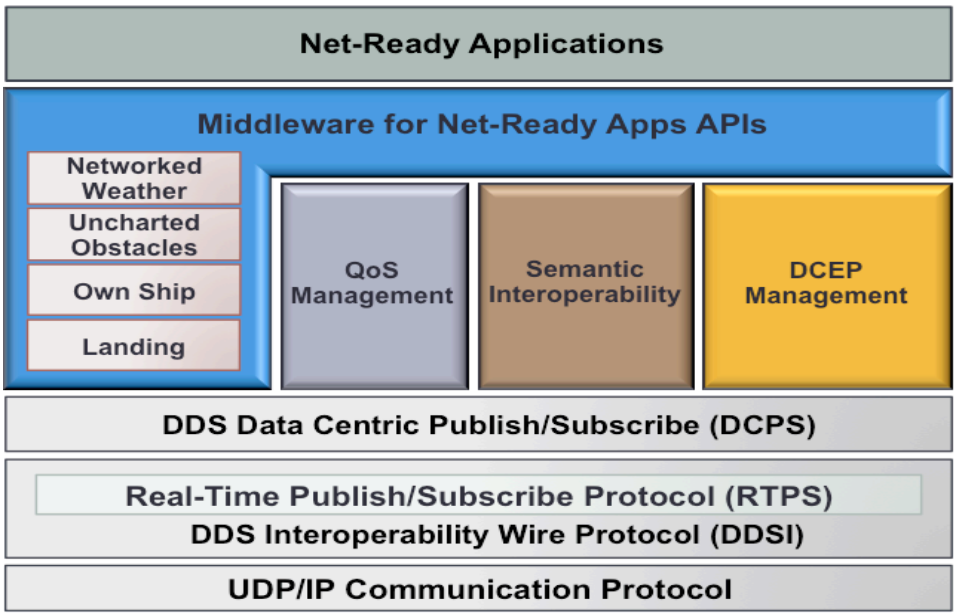

Figure 2. Middleware for Net-Ready Applications Overview

Figure 2 shows an overview of the Middleware for Net-Ready Applications, with the communication APIs for the different net-ready applications, and modules for managing QoS, Semantic Interoperability and Distributed Complex Event Processing, explained in sections 3.1, 3.2 and 3.3 respectively. We implemented the middleware for OpenSplice DDS [6], RTI Connext DDS [7] and CoreDX DDS [5] commercial DDS™ products. The DCEP module is currently implemented with instances of the Esper CEP open-source engine.

3.1 Quality of Service Management

The DDS™ specification provides a rich set of Quality of Service (QoS) parameters, features and enforcement mechanisms (e.g. communication reliability, latency, transport priority, data persistence, etc.). However, the configuration and association of these

QoS parameters with the different DDS entities (i.e. Domain participant, Publisher, Subscriber, DataReader/Writer and Topics) is usually restricted to application development time, and is work-intensive. Therefore, a QoS Management API was developed to provide direct means of defining DDS QoS parameters dynamically through configuration files, which serve as QoS templates that can be used by the other components of our middleware.

Using the QoS Management API, important QoS settings can be set, depending on the scenario of usage, for example: Defining prioritized event flows, in accordance to the level of priority required by SoIs. Setting the level reliability for data delivery to ensure delivery or the discarding of certain types of events in the case of network failures. Setting the persistence level of different Topics to allows late joiners to receive events already published into the system, or to make data volatile.

3.2 Semantic Interoperability Support

When systems operated by different groups, belonging to different organizations, need to interact and exchange information, the use of different data standards and formats may become a hurdle. Although much shared data are conceptually of same type, e.g. position, speed, cargo weight, etc., they may be represented in different units and formats. In the specific case of geographic location, it may be described by different projection, datum and coordinate systems [19].

To tackle this problem, we implemented a semantic interoperability service, where a node in the DDS Domain assumes the mediator role. Whenever a new node joins the system, it will inform the mediator which model it adopts to represent the data to be shared. This mediator is capable of querying an Ontology Manager to obtain further semantic information about how to deal with different data models adopted by the peers. The Ontology Manager stores ontologies that represent known semantic models and conversion rules between them, and delivers these rules to the mediator.

As to the data conversion process, it may be performed by two different approaches. In a centralized approach, upon receiving the conversion rules from the Ontology Manager, the mediator will be responsible for converting all data among the different models. Of course, this is only feasible if the amount/frequency of data exchange is low. In a translation-on-receiving approach, the mediator will forward the conversion rules to all nodes, and each one will produce data using its original model. When a node then receives data represented in a different model, it will itself perform the translation.

3.3 Distributed Complex Event Processing Management

The Distributed Complex Event Processing Management (DCEPM) module, as shown in Figure 2, implements the distributed event processing architecture, which allows the creation of EPNs to detect SoIs. The EPAs use DDS™ to send and receive events from/to monitored nodes, and to exchange events with each other. In the former case, the middleware APIs for Net-Ready Applications, in addition as being used by applications, are used by the EPAs to subscribe and publish to the DDS Topics that share data from monitored nodes and applications. On the latter case, the EPAs provide and use additional DDS Topics that are specific for them, which can also be used by applications to consume events.

Each EPA contains a DDS Subscriber for subscribing to events from the desired sources. For example, an EPA can subscribe to events from rotorcraft nodes or other

EPAs. A CEP engine is instantiated inside each EPA, with deployed rules that process events received from the DDS layer. Since the communication APIs are separated from the CEP Engine, EPAs can use any desired internal CEP engine for processing events (the current implementation uses the open-source Esper CEP engine). An internal DDS Publisher allows the EPA to publish events into the DDS Domain, after processing input events with the internal CEP Engine.

3.4 Net-Ready Applications With Visualization

The middleware presented in this work provides many benefits for the development applications that enhance SA in rotorcraft missions. We developed applications with capabilities for monitoring Own Ship positions, Weather Reports and Obstacles, showing a common operational picture on a map (using instances of FalconView [20] and Google EarthTM). In [2] we present details about the development of those applications, and in Section 7 we show simple functional evaluation with a visualization tool to show the detection of a SoI.

4 Representative Scenario and Situation of Interest

In Brazil, the offshore regions of the states of Rio de Janeiro, São Paulo and Espírito Santo are areas of intense oil exploration, and are responsible for up to 80% of the Brazilian production. Therefore, many offshore oil drilling and extraction platforms are deployed in this region, and can be located as far as 100km from the coast. Because of this exploration activity there are ships and helicopters transporting employees, cargo and equipment between land bases and the offshore platforms. However, both the oil extraction activities of the offshore platforms, and the traffic of ships and aircraft impose operational risks. Employees working on oil platforms are exposed to risks of fire or explosions and the inherent risks of helicopter air transportation. Several emergency incidents occurred during the last several years, such as fire and explosions on platforms. Also, helicopter crashes have been reported. The large distances these helicopters have to fly, without any close emergency landing spots, expose them to unexpected and severe weather and also make them vulnerable to mechanical failures. These risk factors and recent incidents show that the capacity of effectively performing emergency response operations in these regions is very important, to respond to incidents involving oil platforms, ships and helicopters.

4.1 Situation of Interest

In order to show how the proposed system is able to support applications that enhance SA, we describe a hypothetical SoI to be detected by the system.

The example SoI is characterized by the generation of an alert notification when a significant percentage of monitored nodes is out of range from a set of previously defined stationary set of Points of Interest (PoIs). An application of this inference could be, for example, to detect when a set of helicopters, which cover routes between support bases (e.g. land bases, oil extraction platforms, military support bases, etc.), are too far from all these points at the same time, characterizing an exposure to high operational risk. For instance, if rescue helicopters are all away from support bases at the same time, an unexpected emergency situation that requires their reallocation can be difficult to manage.

Applying this situation to the scenario presented in Section 4, we define that the helicopters for transportation between land bases and offshore platforms are the monitored nodes, and the land bases are the PoIs. Considering all the operational risk involved, it is important to have a minimum number of rescue helicopters close to support bases, so they can return and reload with the required resources (e.g. fuel, medicine or cargo). This SoI was chosen because it presents characteristics that allow us to explore its detection with multiple granularities of event abstraction levels (explained in Section 5.1).

5 Distributed Complex Event Processing Instantiation

In order to implement the detection of the SoI described in Section 4.1, an EPN with a specific hierarchical topology is used. We define three main kinds of EPAs, as shown in Figure 3. Node EPAs are deployed in the mobile monitored nodes, containing CEP rules (i.e. event correlation and patterns rules) that detect primitive events locally at the node (e.g. from local sensors), and generate events that represent abstractions of situations detected locally on that node. Network EPAs are deployed in the fixed network and receive events from a set of Node EPAs in their responsibility (the sets of Node EPAs in the responsibility of a Network EPA can be defined following any desired criteria, like geographical regions). The Network EPAs contain CEP rules that detect events related to all the nodes in their responsibility.

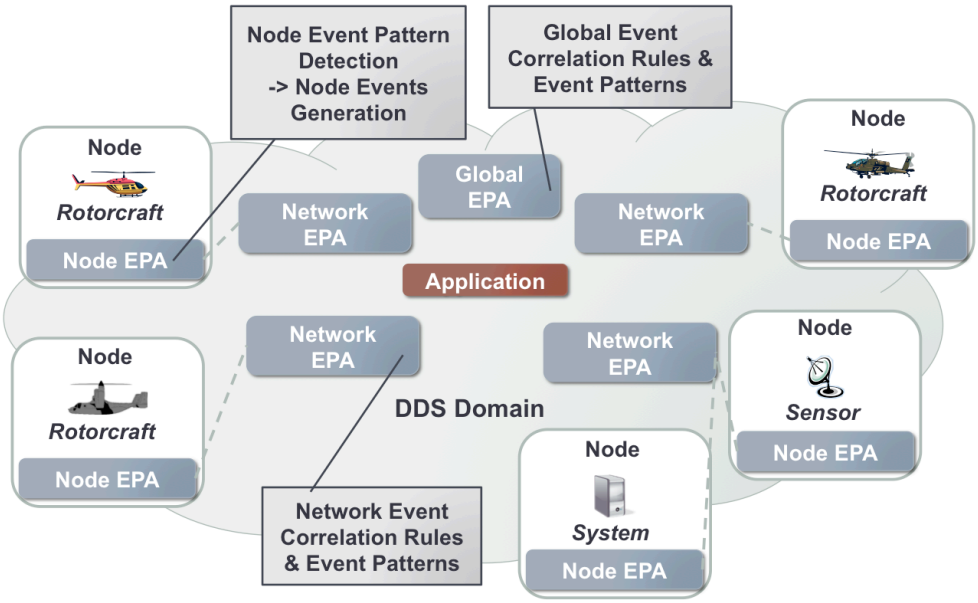

Figure 3. Example of an EPN Hierarchical Topology with Node EPAs, Network EPAs and a Global EPA

The Network EPAs process the events received from Node EPAs and generate events with a higher level of abstraction (e.g. summarizations, aggregations of primitive events). Global EPAs are also deployed in the fixed network, and receive events from a set of Network EPAs. The Global EPA contain CEP rules that detect events related to all Network EPAs, and produce events with an even higher level of abstraction/aggregation, that characterize the detected overall SoI. This hierarchical organization of agents is general enough to be used for the detection of many other similar SoIs, since it reflects an organization that promotes scalability.

5.1 Events Abstraction Levels

Different granularities and abstraction levels can be used to define CEP events, and there's subjectivity on this choice. For example, considering the SoI presented in Section 4.1, the Own Ship Position Report event sent by rotorcraft nodes is considered to be at a fine-grained abstraction level. Other events considered as having a coarser-grained abstraction level (i.e. generated by aggregation of events with lower abstraction levels) can be defined. On our example situation, a node has knowledge about existing PoIs, and calculates if it is within range of all of them. So instead of sending a fine-grained Own Ship Report, it can send a coarse-grained event representing whether the node is out of range from PoIs, called "RangeStateFromPoIs". Section 6 shows performance results of simulations with different abstraction levels of events exchanged between processing agents, and how they impact performance and scalability.

6 Performance Results

This section describes tests performed with the detection of the SoI presented in Section 4.1. The tests compare different settings of events abstraction levels and EPN distribution models, to assess how they influence performance and how they scale in the number of monitored nodes and PoIs.1

Two settings for the abstraction levels of events were used; **(A1) Fine-grained Abstraction of Events –** Rotorcraft nodes send Own Ship Report events (with their current position), and Network EPAs perform all processing to verify which nodes are out of range from PoIs. **(A2) Coarse-grained Abstraction of Events** - Each rotorcraft node has knowledge of the coordinates of all PoIs and publishes a coarse-grained abstraction event indicating whether or not it is on range from all PoIs in a given moment ("RangeStateFromPoIs"). This local processing removes the processing burden from the Network EPAs, transferring it to the nodes to promote overall system scalability.

Two settings for deployment models of EPNs were used: **(B1) Centralized Event Processing** - All monitored nodes send events to a single Network EPA. **(B2) Distributed Event Processing** - Monitored nodes send events to distributed Network EPAs. Each Network EPA processes events from nodes in its responsibility and disseminates consolidated reports (i.e. coarse-grained events) with this data. A Global EPA aggregates data from all Network EPAs, counting the overall percentage of nodes out of range from PoIs, and generating an event indicating the global situation.

The tests were performed with an infrastructure of 6 machines interconnected in a Local Area Network (LAN) by a Gigabit Ethernet switch. Up to 3 Network EPAs and 1 Global EPA, were deployed in four different machines with Quad-Core Intel i5 processors and 8GB RAM, running Fedora Linux 15, 64 bits. A load generator application, simulating rotorcraft nodes, Node EPAs, and an instance of Google EarthTM application, were respectively deployed in two laptop computers.

For each setting (A1, A2, B1 and B2), 1000, 2000 and 3000 rotorcraft nodes, and 100, 500 and 1000 PoIs were simulated. A maximum range distance from PoIs of 10 kilometers

1 Although the SoI presented in Section 4.1 does not require processing a large number of PoIs (as the number of offshore platforms and land bases is not large), other different SoIs may require the processing of a large set of virtual entities (e.g. an aircraft monitoring obstacles on real time).

was specified, and each rotorcraft node was positioned in a fixed location outside the range of all POIs. Figure 4 shows a chart comparing the throughput of the system for the different settings (A1, A2, B1 and B2).

As expected, the coarse-grained abstraction of events removed the processing effort from the Network EPAs, improving the overall system throughput. Since each monitored node sends data to a specific Network EPA, the overall system average throughput is the sum of the average throughput of all Network EPAs. This demonstrates the benefits of the distributed deployment of Network EPAs.

Regarding network bandwidth usage (not addressed in this test), it is reasonable to assume that the periodicity of events with coarser-grained abstraction levels is in general lower than the frequent sending of events with finer-grained abstraction levels. In applications where the fine-grained events are still necessary for other inferences or system functionalities (e.g. the Own Ship Position is necessary for a control station to show the location of nodes in a map), these events will be propagated through the DDS™ domain only to the interested parties, not affecting the other nodes.

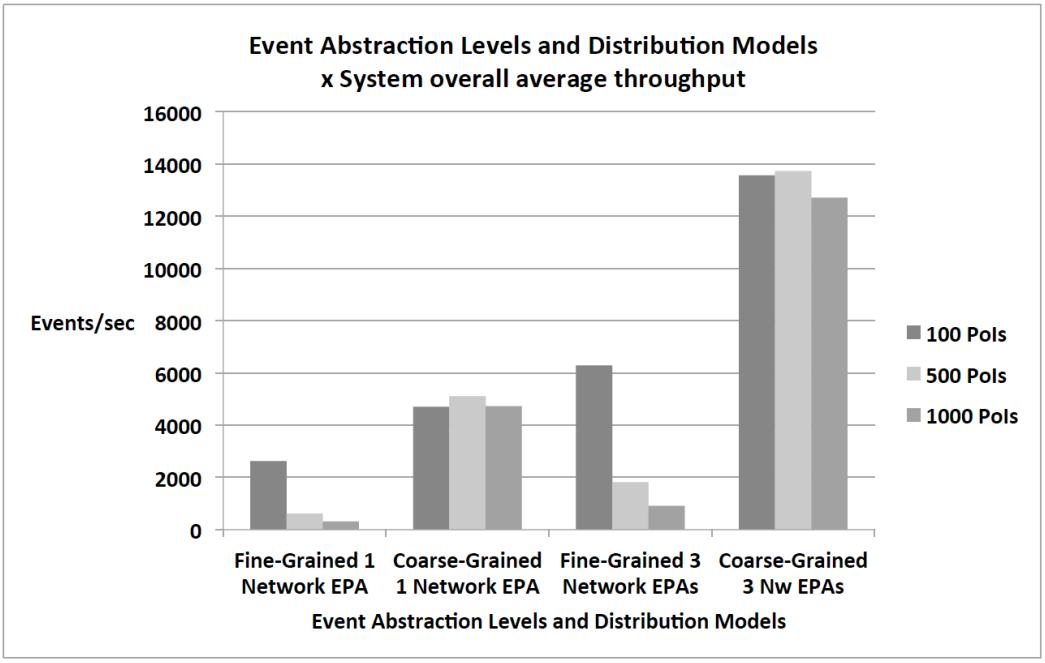

Figure 4. Overall system average throughput with different event abstraction levels and EPN distribution models

7 Visualization

This work is primarily focused on the middleware aspects to support applications and visualization techniques at a higher level and it is not the goal of this paper to evaluate the level of SA provided to users, or its actual impact on operational performance. In [2] we presented a measurement on the value of integrating the underlying communication infrastructure and the network-ready applications, on rotorcraft for improving operational performance.

A test with a visualization tool was used to serve as a functional evaluation of the capabilities provided by the underlying middleware architecture. In the context of the

representative scenario presented in Section 4, we illustrate the use of a visualization tool which benefits from the middleware architecture, and how it shows the example SoI. Figure 5 shows a screenshot of a Google EarthTM application, showing the coast of Rio de Janeiro state and rotorcraft nodes (represented in MIL-STD-2525A symbology) flying between land bases and offshore platforms. The nodes in green are inside the specified range from PoIs (i.e. land bases), and the ones in red are out of range from PoIs.

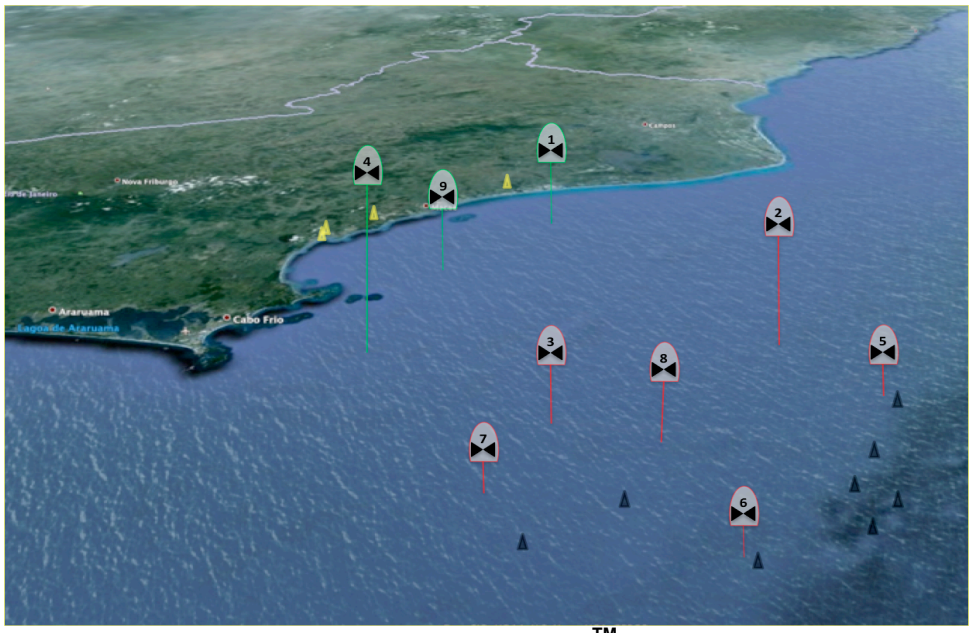

Figure 5. Real-time visualization with Google Earth™, showing rotorcraft nodes flying between land bases and oil platforms in the coast of Rio de Janeiro

8 Related Work

The use of middleware to support applications that enhance SA (e.g. visualization applications) is addressed by other work, such as [3] and [21]. The authors of [3] present a visualization tool and a middleware architecture, based on CORBA for client-server communication between monitored nodes and an infrastructure of services. Commercial solutions, like Solipsys Tactical Display Framework (TDF) [21] provide middleware and advanced visualization solutions for many types of mission-critical systems.

In general, the integration of pure visualization applications (e.g. FalconView [20]) with any type of middleware has the potential to benefit users with enhanced SA. Many different academic and commercial middleware implementations provide event-based communication and distributed event processing, such as [14-18; 21; 26; 27]. The benefits of our middleware architecture come from the combination of using real-time data-centric publish-subscribe communication model with advanced QoS features provided by the DDS™ Specification, Semantic Interoperability, and Distributed Complex Event Processing, provided by the modules and APIs presented in this work. To the best of our knowledge, none of the distributed event processing middleware solutions combines all these features. The middleware presented in this work can be integrated in any desired visualization tool or framework, and the use of Google EarthTM was due to its simplicity in providing an initial experimentation.

9 Conclusions

In this work we presented a middleware architecture to support net-ready applications that can enhance SA, with capabilities such as Real-Time Peer-to-Peer Data-Centric Publish/Subscribe Communication, QoS Management, Semantic Interoperability and Distributed Complex Event Processing for the detection of SoIs. A representative scenario and a SoI instantiation to be detected by the system were presented. Simulations with the detection of the SoI were performed, and a visualization tool was used to illustrate the capabilities of the proposed architecture, in the context of the representative scenario. Performance results show that the distribution of EPAs and the use of coarse-grained event abstractions improve the overall system scalability regarding the throughput in processing events. The use of coarse grained event abstraction levels partially removed the processing load from the infrastructure, taking advantage of the processing power present in nodes, and also potentially reducing network bandwidth usage, which is very important in resource constrained mobile networks.

References

[1] Endsley, M.R., 1995. Toward a theory of situation awareness in dynamic systems. Human Factors: The Journal of the Human Factors and Ergonomics Society 37, 1, 32-64.

[2] T.A. DuBois, B. Blanton, F. Reetz III, M. Endler, W. Kinahan, G.L.B. Baptista, and Johnson, R.L., 2012. Open Networking Technologies for the Integration of Net-Ready Applications on Rotorcraft. In Proceedings of the Annual conference of the American Helicopter Society (2012).

[3] Feibush, E., Gagvani, N., and Williams, D., 2000. Visualization for situational awareness. Computer Graphics and Applications, IEEE 20, 5, 38-45.

[4] OMG, 2006. Data-Distribution Service for Real-Time Systems (DDS).

[5] CoreDX DDS Data Distribution Service Middleware, TwinOaks Computing, Inc. http://www.twinoakscomputing.com/coredx

[6] PrismTech, PrismTech, OpenSplice DDS. www.opensplice.com

[7] RTI, RTI Connext DDS. www.rti.com

[8] Luckham, D.C., 2001. The power of events: an introduction to complex event processing in distributed enterprise systems. Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA.

[9] Eugster, P.T., Felber, P.A., Guerraoui, R., and Kermarrec, A.M., 2003. The Many Faces of Publish/Subscribe. ACM Computing Surveys 35, 2, 114-131.

[10] Cugola, G. and Margara, A., 2010. Processing flows of information: From data stream to complex event processing. Technical report, Politecnico di Milano, 2010. Submitted for Publication.

[11] SYBASE, SAP Sybase Event Stream Processor CEP. http://www.sybase.com/products/financialservicessolutions/complex-event-processing

[12] Esper, Esper. http://esper.codehaus.org/

[13] TIBCO, TIBCO Business Events. http://www.tibco.com/products/business-optimization/complex-event-processing/businessevents/default.jsp

[14] Oracle, Oracle CEP. http://www.oracle.com/technetwork/middleware/complex-event-processing

[15] StreamBase, StreamBase CEP. http://www.streambase.com

[16] Etzion, O. and Niblett, P., 2010. Event processing in action. Manning Publications Co.

[17] Papaemmanouil, O., Cetintemel, U., and Jannotti, J., 2009. Integrating Pub-Sub and Stream Processing for Internet-Scale Monitoring.

[18] Fidler, E., Jacobsen, H.A., Li, G., and Mankovski, S., 2005. The PADRES Distributed Publish/Subscribe System.

[19] Mahlinh, D., 1973. Coordinate Systems and Map Projections. George Philip and Sons Publishers, London.

[20] Georgia Tech Research Institute, FalconView. http://www.falconview.org/trac/FalconView

[21] Raytheon, Solipsys Tactical Display Framework (TDF). http://www.solipsys.com/tdf/