

CanCoastWatch Dynamic Configuration Manager^{*}

Roozbeh Farahbod¹, Uwe Glässer¹, and Hans Wehn²

¹ Software Technology Lab, Simon Fraser University, Burnaby, B.C., Canada
{[rfarahbo](mailto:rfarahbo@cs.sfu.ca), [glaesser](mailto:glaesser@cs.sfu.ca)}@cs.sfu.ca

² Research and Development, MDA Corp., Richmond, B.C., Canada
hw@mdacorporation.com

Abstract. We propose an abstract operational model of a highly adaptive *distributed resource management architecture* (DRMA) for network enabled operations in large-area surveillance. The architectural design of DRMA is described in terms of a distributed *abstract state machine*. As part of this work, we present the specification and design of a Dynamic Configuration Manager (DCM) that actively maintains the configuration of a network of resources by observing mission requirements, communication links, and measuring various parameter values.

1 Introduction

As part of the CanCoastWatch system [1], we propose an abstract operational model of a highly adaptive *distributed resource management architecture* (DRMA) for network enabled operations in large-area surveillance. CanCoastWatch is a simulator for testing algorithms needed for large-area surveillance missions that require efficient deployment of multiple search resources and rapid extraction and distribution of key information [1, 2]. A key requirement for such a system is the need for dynamic configuration and efficient management of distributed resources under spontaneously changing and essentially unpredictable conditions: rapidly varying workload, multiple resource failures, and unreliable communication mechanisms. To facilitate dynamic reorganization and to avoid the bottleneck of centralized systems, our design embodies a multi-agent based approach to robust and efficient fusion of heterogeneous data and information. In that respect, our approach resembles the concept of Distributed Perception Networks introduced in [3, 4], although the later one is less general and only loosely defined.

The main focus of this paper is on the specification and design of a Dynamic Configuration Manager (DCM), as part of DRMA, that actively maintains the configuration of a network of resources by observing mission requirements, communication links, and measuring various parameters. The architectural design of the DRMA is described in terms of a distributed *abstract state machine* (ASM). The resulting model abstractly characterizes the dynamic properties of the resource management architecture and serves as a framework for requirements specification and design analysis of the key DRMA attributes prior to actually

^{*} This research project is funded by Precarn under the Intelligent Systems Program.

building the system. Overall, the DRMA is characterized by its concurrent and reactive nature, making it difficult to predict the resulting system behavior with sufficient detail and precision under all circumstances. Hence, there is a need for a formal framework. It will allow us to not only reason about specification and design issues, but also uncover deficiencies that would otherwise go unnoticed.

2 CCW System Requirements

This section gives a brief overview of the key CCW system requirements.

2.1 Terminology

With regard to the CanCoastWatch System Concept [1], a *resource* can be a sensor, a platform, or a communication device. A platform usually groups a set of resources that are physically bonded to each other. A boat or a helicopter with various sensors are two examples of a platform. Each resource identifies a nonempty set of *capabilities* specifying the types of services that the resource provides. Capability of platforms is the aggregated capability of its resources. A group of resources that are logically suitable to perform a specific task may form a *cluster*. These structures are formally defined in the following:

universe RESOURCE \equiv SENSOR \cup PLATFORM \cup DEVICE

platformResources : PLATFORM \rightarrow RESOURCE-SET

cluster : CLUSTER \rightarrow RESOURCE-SET

capabilities : RESOURCE \rightarrow CAPABILITY-SET

$\forall p \in \text{PLATFORM}, \text{capabilities}(p) \equiv \bigcup_{r \in \text{platformResources}(p)} \text{capabilities}(r)$

A set of resources that can communicate with each other form a network graph. We refer to the set of all these network graphs as *the network*. The network has a dynamic organization which we call *network configuration*. A Dynamic Configuration Manager (DCM) maintains the configuration of the network as it dynamically changes over time.

2.2 DRMA Requirements

The abstract requirements of DRMA can be summarized as follows:

1. The application context calls for a *decentralized* architecture and protocol for dynamic network configuration management that is robust, supports future development and improvements, and can represent a mix of peer-to-peer and hierarchical networks.
2. The network can reconfigure itself, and the resources can be rescheduled dynamically to best achieve a given mission goal.

3. It should be possible to command a group of resources in addition to individual resources.
4. For military applications, it is always clear who commands whom. The military command hierarchy must be distinct from the communication topology, i.e. who can talk to whom [5].
5. The Dynamic Configuration Manager has two roles:
 - (a) It suggests possible task-to-node assignments based on mission plans and, if required, offers a set of possible (feasible) network configurations;
 - (b) It actively maintains the configuration by observing the communication links and measuring various parameters.

3 Abstract Architecture of DRMA

Here we sketch a high-level model of the proposed DRMA in terms of its characteristic attributes and features.

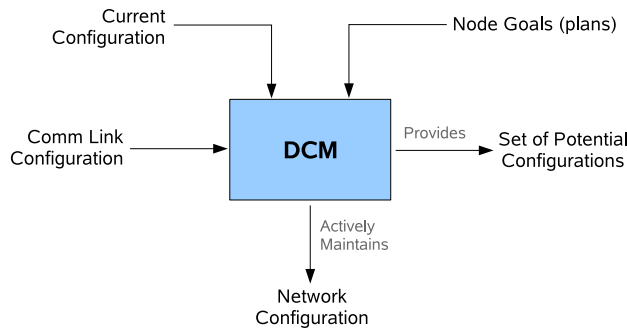


Fig. 1. DCM Inputs and Outputs

Nodes form the basic elements of the CCW architecture [5]. They either represent individual resources or resource clusters (see Section 2.1). Every node has a DCM component that is responsible for maintaining the configuration of the node and, if exists, the subtree below the node.

DCM relies on the following *input* information: *node goals* (which are received from another node with higher authority level), *platform configuration*, and *configuration of communication links* that connect this node to other nodes.

As *output*, DCM *a)* provides a set of *potential configurations* and task-to-node assignments for planning and scheduling, and *b)* continuously updates the *current configuration* of the node.

3.1 Network Configuration

We classify the nodes of the network into two basically different types: *real* and *virtual* (see Figure 3.1). Real nodes map onto real platforms (i.e., resources) and receive complete schedules (tasks and time of operation) from virtual nodes. Virtual nodes represent clusters of nodes and receive mission goals based on which they compute schedules for the nodes under their control.

Every DCM module functions as part of a specific node as stated by a function $node : \text{DCMAGENT} \rightarrow \text{NODE}$ which refers to the node on which a given DCM agent is running. The network configuration is realized by the following functions defined on nodes:

- $isVirtual : \text{NODE} \rightarrow \text{BOOLEAN}$, $isReal : \text{NODE} \rightarrow \text{BOOLEAN}$
holds if $node$ is a virtual or a real node respectively.
- $childNodes : \text{VIRTUALNODE} \rightarrow \text{NODE-SET}$
yields the set of nodes (real or virtual) under direct authority of a given virtual node.
- $parentNode : \text{NODE} \rightarrow \text{VIRTUALNODE}$
points to the parent node of a node.
- $nodeCapabilities : \text{NODE} \rightarrow \text{CAPABILITY-SET}$
holds the capability set of nodes. On a virtual node, this would be the aggregated capabilities of all its child nodes.

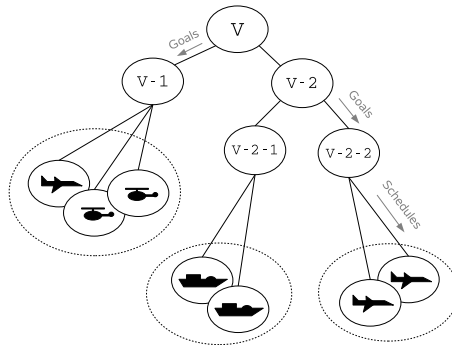


Fig. 2. Real nodes represent actual platforms and virtual nodes represent clusters.

3.2 Maintaining the Network Configuration

The following events can trigger a change in network configuration:

1. new or modified node goals;

2. changes in the platform configuration or availability of resources (platforms), e.g. a platform may disappear or become unavailable or a sensor may stop functioning;
3. problems with communication links.

The network configuration can be changed through the following canonical *transformation patterns*:

1. New clusters can be created, for instance, in response to a new high priority mission goal.
2. Resources can be taken from one cluster and assigned to others, e.g., in order to better serve a higher priority mission or to balance resource load.
3. Resources can form their own solitary clusters if they get disconnected from other resources due to a communication fault.
4. Clusters can be merged into a new larger cluster to satisfy a new/changed goal. This can be considered as a special case of 1.

3.3 DCM Interface

The main purpose of DCM is to maintain a *good* configuration of the network of nodes in the system. To achieve this goal, DCM requires frequent interaction with its environment³. In this section we look at the interface of the DCM module.

The DCM component should have access to the following input information to function properly:

1. Goals, in the form of *Capability Plans* with required capabilities, mission area, etc.,
2. Availability of communication links,
3. Capability of nodes.

Suggesting Mode Plans Capability plans are mission plans with a list of tasks and a list of required capabilities for each task. When a new goal (in the form of a capability plan) is received by a node, the DCM module of that node will compute and offer a sequence of *Mode Plans* that can be employed to satisfy the goal. Mode plans are derived from capability plans and in addition include an assignment of nodes to tasks. The following function abstractly captures this service:

$$\textit{suggestedModePlans} : \text{CAPABILITYPLAN} \rightarrow \text{MODEPLAN-seq}$$

The sequence is ordered so that the first mode plan is relatively the best candidate (i.e., with the lowest cost) that DCM has to offer. By offering these potential mode plans, DCM may also suggest possible changes to the configuration of the nodes below it. The part of DCM that is responsible for suggesting potential modes for plans is called *Dynamic Configuration Advisor* or DCA.

³ Here we exclude any interaction between DCM modules.

Network Reconfiguration When one of the suggested mode plans is accepted by the node, DCM will be notified to reconfigure the network if a change is necessary.

3.4 Dynamic Configuration Advisor

This section presents a high level operational model of DCM agents and focuses on the behavior of DCM as a dynamic configuration advisor. The behavior of a DCM agent can be decomposed into four parallel rules:

```

DCMProgram  $\equiv$ 
  MonitorGoals
  MonitorResources
  MonitorComLinks
  RespondToDCMMessages

```

MonitorGoals monitors changes to node plans that would require a reconfiguration of the network. Here, we only focus on dynamic configuration advisory behavior of DCM and refine the *suggestedModePlans(plan)* function (see Sect. 3.3) that suggests new mode plans for every new capability plan received by the node.

The idea of suggesting new mode plans is to find all subsets of the children of the node that together have the capabilities required by the capability plan. Starting with the subset with the minimum cost of use and reconfiguration (first candidate), if the candidate can perform the tasks of the plan (see *matches(c, t)*), no reconfiguration is needed and a new mode plan can be created based on the given capability plan; otherwise, a reconfiguration of the network is required. The derived *suggestedModes* function is refined by the following rule:

```

SuggestedModePlan(plan)  $\equiv$ 
  let  $nodes_m = \{s \mid s \subset childNodes(node(self)) \text{ with } satisfies(s, capabilities(plan))\}$  in
  while ( $\neg visitedAll(nodes_m)$ )
  choose  $candid \in nodes_m$  with
    nextBestCandidate(candids, nodes_m, capabilities(plan)) do
  if  $matches(candid, planTasks(plan))$  then
    append result with CreateModePlan(candids, plan)
  else
    append result with SuggestNewConfiguration(candids, plan)

```

where

```

nextBestCandidate(candid, set, caps)  $\equiv \neg visited(candid) \wedge$ 
   $\forall s_i \in set, \neg visited(s_i) \rightarrow cost(candid, caps) \leq cost(s_i, caps)$ 

matches(candid, tasks)  $\equiv \exists f : candid \rightarrow tasks$  such that
   $\forall n \in candid, \forall t \in tasks, f(n, t) \rightarrow requiredCapabilities(t) \subset nodeCapabilities(n)$ 
   $\wedge \forall n \in candid, \exists! t \in tasks, f(n, t)$ 
   $\wedge \forall t \in tasks, \exists! n \in candid, f(n, t)$ 

```

The heart of this rule is the *matches(candid, tasks)* function. A set of nodes *candid* matches a set of tasks if and only if there exists a mapping function that

assigns one and only one task to every node and one and only one node to every task such that a node n is assigned to a task t only if node n has the required capabilities of the task t .

We assume *nodeCapabilities*(*node*) to be the capabilities of *node*. For virtual nodes, this is recursively defined as the union of the capabilities of its child nodes.

MonitorResources monitors resources to observe dead nodes (disappeared nodes), new nodes (nodes appeared in the group), or changes in child node capabilities. Loss of a child node or changes in the capabilities of a child node will change the aggregated capability of the parent node and can potentially trigger a replanning or reconfiguration.

MonitorComLinks monitors the quality of communication links and maintains the information needed to satisfy communication requirements and to choose communication channels for message transmission. A reachability table that keeps a list of reachable nodes is also maintained by this rule.

RespondtoDCMMessages is the routine that responds to messages received from other DCM modules. DCM nodes communicate with each other over a communication framework that supports one or more of the following basic communication mechanisms: *peer-to-peer*, *multicast*, and *broadcast*.

4 Conclusion

We propose an abstract operational model of a distributed resource management architecture (DRMA) for network enabled operations in large-area surveillance. Our main focus here is on the high level specification and design of a Dynamic Configuration Manager as part of DRMA. Specifically, we address the refinement of the Dynamic Configuration Advisory (DCA) behavior of the DCM component. The DCA subcomponent responds to new or changed mission plans and suggests various assignments of plan tasks to resources which could potentially trigger reconfiguration of the resource network.

The CanCoastWatch project started as a novel attempt to create intelligent solutions to the generic problem of distributed large-area surveillance using mobile sensors. Although the ultimate goal and high-level requirements were established from the beginning, the more detailed requirements on the behavior of the system and the underlying resource network architecture were not clear at first. Hence, a high-level approach towards modeling using abstract state machines and precisely documenting the specifications was very helpful in communicating and understanding the requirements and analyzing various approaches in modeling the system and its underlying network of resources. The specification and design of DRMA is the result of a series of technical interchange meetings with Defence Research and Development Canada (DRDC)⁴, MacDonald, Detwiler and Associates Ltd. (MDA)⁵, and our other project partners. The abstract

⁴ www.drdc-rddc.gc.ca

⁵ www.mdacorporation.com

specification and design of DRMA, and, in particular those of DCA, have gone through many revisions which gradually helped us in better understanding the fairly complex CanCoastWatch system requirements.

Acknowledgments. We sincerely thank Adel Guitouni, DRDC and Richard Yates, MDA for the many stimulating discussions and their valuable input to this project. We would also like to thank the anonymous reviewers for their precious comments and suggestions for improvements.

References

1. CCW Team: CanCoastWatch System Concept. Technical report, MacDonald, Dettwiler and Associates Ltd. (2006)
2. Wehn, H., et al.: A testbed simulator to evaluate the efficiency of net-enabled surveillance. In: UVS Canada. (2006)
3. Pavlin, G., Maris, M., Nunnink, J.: An agent-based approach to distributed data and information fusion. In: IAT. (2004) 466–470
4. de Oude, P., Ottens, B., Pavlin, G.: Information fusion with distributed probabilistic networks. In: Artificial Intelligence and Applications. (2005) 195–201
5. CCW Team: CanCoastWatch Build 2 Specification. Software Specification RX-SP-52-4285 R1.2, MacDonald, Dettwiler and Associates Ltd. (2006)