Ontology-based support for reconfigurable adaptive group communication architecture

Ismael Bouassida Rodriguez, Aymen Kamoun, Sakkaravarthi Ramanathan, Khalil Drira

Ontology-based support for reconfigurable adaptive group communication architecture

Ismael Bouassida Rodriguez[1,2], Aymen Kamoun[1,2], Sakkaravarthi Ramanathan[1,2] and Khalil Drira[1,2]
{bouassida,akamoun,sakkaravarthi,khalil}@laas.fr

[1] CNRS, LAAS, 7 avenue du colonel Roche, F-31400 Toulouse, France
[2] Univ de Toulouse, LAAS, F-31400 Toulouse, France

Abstract. In collaborative communication, adaptation is required to maintain the reliable connection. Within the framework of a wireless environment, it is very challenging for the host entities to handle a sudden/unexpected change in communication and available resources. This issue is addressed in the context of save and rescue missions carried out during natural disasters such as floods and forest fires by human and voluntary operators aided by ground mobile devices and autonomous robots. This paper proposes a multi level architecture that supports reconfigurable adaptive group communications using semantic model. Modeling follows the well-known SWRL (Semantic Web Rule Language) instructions which establish the degree of importance of each communication link or component. Adaptation is then achieved through initializing between various configurations and for a given entity, the latter is modified as need be. Providing generic and scalable solutions for automated self-reconfiguration is driven by rule-based reconfiguration policies using ontologies. Finally, we illustrate constraints capable of meeting evolving requirements and the solutions followed by implementation of this scenario.

1 Introduction

In autonomic communication, the design of self-configurable systems that can adapt dynamically is a major research topic. With respect to adaptive systems, the issue at hand is that of finding a way of enabling us to concentrate on the important service request and also on a specific aspect relevant to a particular situation only. An architecture that analyzes and manages the request in order to get the best possible decision in real-time needs to be designed. Many existing dynamic architectures provide just enough functionality to support simple distributed communications but in case of distributed software systems, more work has to focus on managing the group membership and the nodes deployment. In a group - wide communication under different situations, one has to ensure the tractability of the elaborated solutions when migrating from several users, node or components.

In our case study, we study issues related to Crisis Management Systems (CMS). In this kind of system, there exist different nodes communicating through

various types of flows. For example, a set of smart devices communicate with robots to achieve a given mission and to self-adapt in a dynamic environment. Each device is embedded with software entities with properties, particularly in terms of reliability, self-healability, etc. The system relies on autonomous entities that manage the resources available in order to provide the communication. In this paper, we use semantic modeling approach to solve the different aspects related to communication challenges. Our goal is to provide a communication architecture that can manage the resources to ensure uninterrupted connectivity between mission participants according to the goals.

Section 2 presents related work and section 3 outlines the ROSACE project scenario. Section 4 details our multi-level architecture followed by context representation. Section 6 illustrates some constraints and the solutions while section 7 talks about the implementation. Conclusion and open issues are presented in section 8.

2 Related Work

After developing an information system, there might be many reasons for adaptation, e.g corrective, evolutional or perspective [1]. If we consider the corrective adaptation, application does not behave as expected. The solution is to identify the module that causes the problem and replace it by a new module providing the same functionality. In case of evolutional adaptation, changing user needs is supported by adding new modules or modifying the existing functionality of the application architecture. Finally in perspective adaptation, the idea is to improve performance. For example, if a module receives numerous requests that result in performance degradation, it can be duplicated to share the workload.

Several studies focus on a model-based approach with automated management techniques for adaptation, i.e, introducing changes to a program or maintaining functionalities and, whenever possible, improving performance. In [2], an adaptive framework supporting multiple classes of multimedia services with different quality requirements in wireless cellular networks is proposed. This work focused on communication level but the migration of adaptive policies to higher level is not detailed. The work in [3] envisions a middleware architecture for service adaptation based on network awareness to manage resources in an adaptive context. Though it highlights the necessary adaptation, there is no solution for context-aware problem such as resource constraint. Further research [4] provides frameworks for designing transport protocols whose internal structure can be modified according to the application requirements and network constraints. In [5], a schema is described for dynamically managing distributed computing resources by continuously computing and assessing quality. Here, resource utilization metrics are determined *a posteriori* and an adaptive distributed system reference architecture is equally put forward but the authors didn't explain the adaptive policies.

Recently, ontology has been the subject of much attention as it allows semantics of the managed systems to be represented. As described in [6], ontology

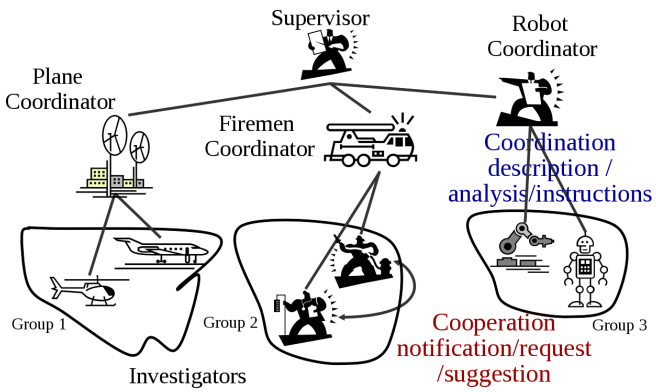

Fig. 1. The CMS Activity Description

supports heterogeneous management entities by capturing semantics from the organizational, environmental and operational viewpoints of overall distributed environment. In [7], the combined use of ontologies and policies is proposed to enforce, in a context-aware approach, the adaptive behavior of autonomous management entities.We also found that in the case of architecture adaptation, model-based strategies are widely used for system transformations that changes according to the environment and requirement. To cope with evolving constraints, behavioral and architectural adaptability is required at several levels. This entails coordination management without which performance would drop much below target.

Motivated by the above discussion, we present a novel ontology-based support for reconfigurable adaptive group communication architecture. Our framework is divided into many levels and the adaptive decision at higher level is migrated to lower for execution. Similarly, decision could be also at lower level thus overcoming the bottleneck associated with waiting for the response from the upper ones.

3 ROSACE Project Scenario

This scenario involves various types of mobile actors using a heterogeneous communication devices. Ground and aerial communicating robots along with human actors utilizing mobile devices operate within a wireless communication context. A distinction is made between the professionals with specific communication devices or occasional actors that carry a mobile device (e.g. PDAs, Phones). Likewise, robot actors such as Autonomous Aerial Vehicles (AAV) are differentiated from Autonomous Ground Vehicle (AGV). However in all cases, the communication system must deal with the evolution of user needs and with the changes due to device constraints. The idea is to exchange information between mobile participants to achieve a common mission. The three major roles assigned to this

scenario are: mission supervisor, coordinators, and field investigators (Figure 1). Each participant plays his own role and is associated with an identifier and the devices he uses. The performed functions are as follows:

- The supervisor's function is to monitor, manage and authorize actions to coordinators and investigators. This entity supervises the whole mission. Once the data (information) has been received from the coordinators, the supervisor reviews the current situation. When he makes a decision, he then passes to the appropriate coordinators. He is equipped with permanent energy resources, high communication and CPU Capabilities;
- Coordinator's job is to report to the supervisor. He manages the group of investigators during the mission and assign tasks to each one of them. The coordinator has to collect, interpret, summarize and diffuse information from and towards investigators. The coordinator has high software and hardware capabilities;
- The investigator's role is to explore the operational field, observe, analyze, and report about the situation. They also take care of helping, rescuing and repairing.

To support this schema, network-oriented services should be dynamically activated to respond actor requests. They should provide ubiquitous access to peers and be technically transparent, taking into account different context requirements depending on the targeted activity, users' mobility, exchanged data flows (e.g. audio, video), and constraints such as variable communication and device resources. Moreover, changes in the cooperation structure need to be focussed in response to coordinator's decision or information acquired by the participants.

There exist two major steps, i.e., "Exploration step" (localizing and identifying the crisis situation) and "Action step" (decision making and allocating tasks). To support this, there are two types of communication flows i.e. coordination and cooperation flows.

Coordination flows take place between investigators and their coordinator and between the coordinators and the supervisor. The investigators send D (Descriptive) data and P (Produced) data describing the situation to the coordinator. The following steps explain data exchanges for the coordination in the exploration step:

- Investigators continuously send type D data to their assigned coordinator and, periodically, type P data.
- Coordinators periodically send P data to the supervisor describing the current state of exploration. All coordinators under exploration step have the same priority to communicate with the supervisor.
- When a critical situation is discovered by an investigator, exploration step ends. The architecture is then reconfigured and the mission moves to a new step called the "action step".

Cooperation flows occur between the investigators within the same group (A2A type: fireman2fireman, robot2robot, etc.) or between the investigators of

different groups (A2B type: robot2fireman, AAV2fireman, etc.). In case of A2A, a distinction is made between the flows such as cooperation notifications, cooperation requests and cooperation suggestions. In the case of A2B cooperation flows, the flows are: cooperation notifications and cooperation requests.

Challenges are here to analyze and distribute requests equitably among a group of investigators and to actively monitor, control and allow the actors to make decision on their own. Although centralized control is simpler to design and implement, it introduces a single point of failure, which can impede service reliability and scalability. By distribution, we can retain the picture of current scenario thus influencing the results of future situation (a prior knowledge database is used). The following describes the messages between the different actors:

1. An investigator makes a request which is transparently intercepted by the coordinator.
2. The coordinator gets a decision from the knowledge database (supervisor) and a solution is provided.
3. At times, decision has to be made by the coordinator itself. In this case, it is stored in the local database, then notified to the supervisor.

4 Multi-level architecture for CMS Scenario

In order to clearly separate different concerns in our approach, a multi-level architecture is proposed. Each level encapsulates issues into a specific model, thus abstracting complexity to a higher level. The modules at the higher level are abstract system representations, that tend to resemble human activities, while lower ones are much closer to real implementations of abstractions supporting these activities.

Relevant levels are identified and adaptation at the highest levels should be governed by changes in development of activity requirements. Adaptation at the lowest levels should be driven by execution context constraint changes. The levels retained are depicted in Figure 2 and detailed in the following paragraphs.

Here, we have chosen an ontology-based model because it constitutes a standard knowledge representation system, allowing reasoning and inference. Moreover, ontologies facilitate knowledge reuse and sharing through formal and real world semantics. Therefore, ontologies are high-level representations of business concepts and relations. These representations are close to developers' minds and therefore well suited to depict application level models. We have chosen to describe these models in OWL [8], the Semantic Web standard for metadata and ontologies.

4.1 Application Level

The retained model in this level is a domain specific ontology that represents concepts and relations of the context of the application provided. It also represents

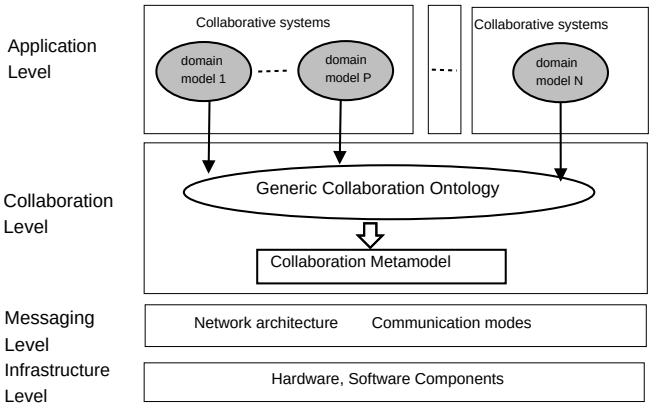

Fig. 2. Abstraction Levels

the applications that need collaboration inside the group of users and/or devices (generically called entities). It includes software elements implementing the activity's business, as well as user interfaces, security modules, etc. Among these elements, (at least) those relevant to collaboration are represented in the architectural model corresponding to this level of abstraction. Only collaboration-related elements of the activity level model will be taken into account in the refinement process. Nevertheless, other business elements (non collaboration-related) are also included and can be used in order to represent the whole activity. The system designer will define the possible roles that can be playes by the defined entities.

In our previous work, we show an ontology that models the business concepts and its relations to the CMS activity. The main concept is the `Participant` featuring several properties. The different types of participant (`Supervisor`, `Coordinator`, `Investigator`) are modeled as sub-concepts, each one with his own additional properties. A `Participant` belongs to a `Group` that is operating under the authority of a `Manager`. The latter can be a `Supervisor` who manages a `CoordinatorGroup` or a `Coordinator` who handles an `InvestigatorGroup`. The other important idea is the `Entity` concept. Thus a `Participant` is also an `Entity` with two sub-concepts: `Artificial` and `Human`. Various human participants are represented as concepts e.g. `Fireman`, `Pilot`, or `Walker` or artificial entities e.g. robots or vehicles, for instance. The different types of robots (`AmphibiousRobot`, `Drone`, `GroundRobot`) are modeled as sub-concepts of `Robot`, each one with its own additional properties. This is related to the Generic Collaboration Ontology (`GCO`) developed by Sancho et al. [9], because `GCO:CommunicationFlow` is defined as a sub-concept of `Flow` and `Entity` as a sub-concept of `GCO:Node`. In other words the participants' roles is in agreement with the definition given in the collaboration ontology, and thus they inherit

all their properties. For example, they have a related `GCO:Node` deployed on a `GCO:Device`, etc.

4.2 Collaboration Level

Here, session level abstraction is provided and the model details the structure of one or more sessions. It describes the way members in a group are organized within sessions, where they can send and receive data flows. The main issue is that of determining a high-level collaboration schema that meets the needs of application's collaboration. Hence, it supports collaborative sessions and can determine those elements needed to implement these sessions. Further explanations about this ontology and the associated choices can be found in [9]. In [9], an ontology model, containing generic collaboration knowledge as well as domain-specific knowledge, is proposed in order to enable architecture adaptation and to support spontaneous and implicit sessions inside groups of humans and devices.

4.3 Messaging Level

This level provides a communication model that masks low-level details (like TCP sockets, UDP datagrams, IP addresses, multicast, etc.) in order to simplify the representation of communication channels. It furnishes an abstract view of distributed systems, so that they become transparent for upper levels. For example, this model may be based on abstractions like Event-based Communications, Peer-to-Peer, Remote Procedure Calls or Remote Method Invocation. In this work, we have retained the Event-Based Communication (EBC)[10]. It represents a well established paradigm for interconnecting loosely coupled components and it provides one-to-many or many-to-many communication pattern. This model is a detailed graph containing a set of event producers (EP), event consumers (EC) and channel managers (CM) connected with push and pull links. Multiple producers and consumers may be associated through the same CM. Since this model represents a graph, it can also be expressed in the GraphML language.

4.4 Infrastructure Level

It contains all needed software components and hardware that enable to run the collaborative system and ensures communication.

5 Context Representation

In our work, adaptation is achieved through various context changes. Context-awareness does not depict the location/environment but represents resource constraints (e.g., connectivity, energy level, available memory, etc.) or changes in the activity and environment where participants can arrive/leave, change roles, etc.

The clear distinction of distributed adaptation at specific levels is the primary advantage of our architecture compared to previous existing ones. Policies are executed according to the context requirements, i.e, if there is a change in the mission, activity level policies are activated whereas in case of low energy level at the device, policies at low level are initiated without affecting the higher levels.

5.1 Reconfiguration Rules

This section presents our approach for modeling and implementation of the CMS activity. Also provided are examples of reconfiguration policies for activity evolution and resource context change using SWRL rules.

To make this system adaptable, reconfiguration rules are needed to adapt the ontology instance to the current situation. As events play a major role in our application, the transformation of entities needs to be triggered. Here, events could occur at activity level, i.e, addition of new participants, changing an action, transfer of a participant from one group to another, new connection between investigators from different groups, etc. The events could also take the form of resource context changes, e.g, parameters like the energy of a device, bandwidth range, CPU processing capacity, RAM availability etc. We use SWRL[11] rules to define our adaptation policy. The application designer defines these rules according to context changes he wants to handle. As we explained earlier, each level executes their own rule when there is an appropriate need. Thanks to the decision model at every level, not all requests are passed to the supervisor. If there is no solution for an event at a particular level, then it triggers the higher level. In SWRL, the head points to the adaptation transformations whereas the body indicates the context of ontology elements. This reconfiguration rules are really useful for adapting the scenario dynamically.

In Figure 1, the supervisor controls the mission and manages the coordinator group, in our case, AAV and fireman coordinators. The flows are exchanged between these operators and each coordinator further manages his own group. Changes in the environment trigger adaptation events. When they occur, the supervisor executes the appropriate SWRL rules to the current instance of the CMS ontology. In our case, adaptation events are classified into 2 types:

High Level Event Reconfiguration Table 1 shows an example in the activity level evolution. Here, the supervisor is managing two coordinators (one for AAVs and other for firemen) and each coordinator has their own investigator group. To make it simpler for the reader, we presented here four investigators, two AAVs and two firemen. As we explained previously, actors communicated through coordination flows. If there is a need to establish a cooperation flow between the investigators among the different groups, the following rule is applied. The conditions for establishing the flow is explained in the scenario illustration.

Low Level Event Reconfiguration Table 2 lists an example of changes in the communication level constraints. if the energy level of a participant's device become low (e.g, below 20 %), a transformation is performed to move the channel

$$Supervisor(?S) \wedge CoordinatorGroup(?CG) \wedge managesGroup(S, CG) \wedge$$
$$Coordinator(?CO1) \wedge Coordinator(?CO2) \wedge hasMember(CG, CO1) \wedge$$
$$hasMember(CG, CO2) \wedge InvestigatorGroup(?IG1) \wedge InvestigatorGroup(?IG2) \wedge$$
$$managesGroup(CO1, IG1) \wedge managesGroup(CO2, IG2) \wedge$$
$$Investigator(?I1) \wedge Investigator(?I2) \wedge hasMember(IG1, I1) \wedge$$
$$hasMember(IG2, I2) \wedge CoordinationFlow(?CF1) \wedge hasReceiver(CF1, CO1) \wedge$$
$$hasSender(CF1, S) \wedge CoordinationFlow(?CF4) \wedge hasReceiver(CF4, CO2) \wedge$$
$$hasSender(CF4, S) \wedge CoordinationFlow(?CF3) \wedge hasReceiver(CF3, I1) \wedge$$
$$hasSender(CF3, CO1) \wedge CoordinationFlow(?CF5) \wedge hasReceiver(CF5, I3) \wedge$$
$$hasSender(CF5, CO2) \rightarrow$$
$$SWRLb : createOWLThing\ (CooperationFlow(?CPF1)) \wedge$$
$$hasSender(CPF1, I2) \wedge hasReceiver(CPF1, I1)$$

Table 1. Rule1–High Level Event Reconfiguration

$$Supervisor(?S) \wedge CoordinatorGroup(?CG) \wedge managesGroup(S, CG) \wedge$$
$$Coordinator(?CO1) \wedge hasMember(CG, CO1) \wedge InvestigatorGroup(IG1) \wedge$$
$$managesGroup(CO1, IG1) \wedge Investigator(?I1) \wedge Investigator(?I2) \wedge$$
$$hasMember(IG1, I1) \wedge hasMember(IG1, I2) \wedge CoordinationFlow(?CF1) \wedge$$
$$hasReceiver(CF1, CO1) \wedge hasSender(CF1, I1) \wedge CooperationFlow(?CPF1) \wedge$$
$$hasReceiver(CPF1, I1) \wedge hasSender(CPF1, I2) \wedge Device(?device) \wedge$$
$$hasID(?device, ?idDevice) \wedge hasHardwareProfile(?device, ?hardware) \wedge$$
$$hasParameter(?hardware, ?param) \wedge sameAs(?param, powerlevel) \wedge$$
$$hasValue(?param, ?value) \wedge swrlb : lessThan(?value, 20)$$
$$\rightarrow adapt : dischargeParticipantDevice(?idDevice)$$

Table 2. Rule2–Low Level Event Reconfiguration

manager from this device. In this case, the decision is made by the communication level of the coordinator that manages the participant having an energy problem by triggering the SWRL rule. More details can be found in the next section about the *dischargeParticipantDevice*.

6 Scenario Illustration

This section presents a top-down approach with respect to the architecture models used for the activity and communication levels. It also illustrates the refinement processes that exist between levels in the adaptation process. The architectural configuration of the activity level is captured and represented in the CMS ontology instance as shown in the upper part of Figure 3. Here, concepts and relations from the activity-specific and from the generic collaboration ontologies are instantiated. To refine this activity model, rules are processed over these ontological instances. Firstly, activity-specific rules are processed and then, generic collaboration rules are applied. Then, rules are processed for each instance of `GCO:CommunicationFlow` found, thus creating the corresponding channel man-

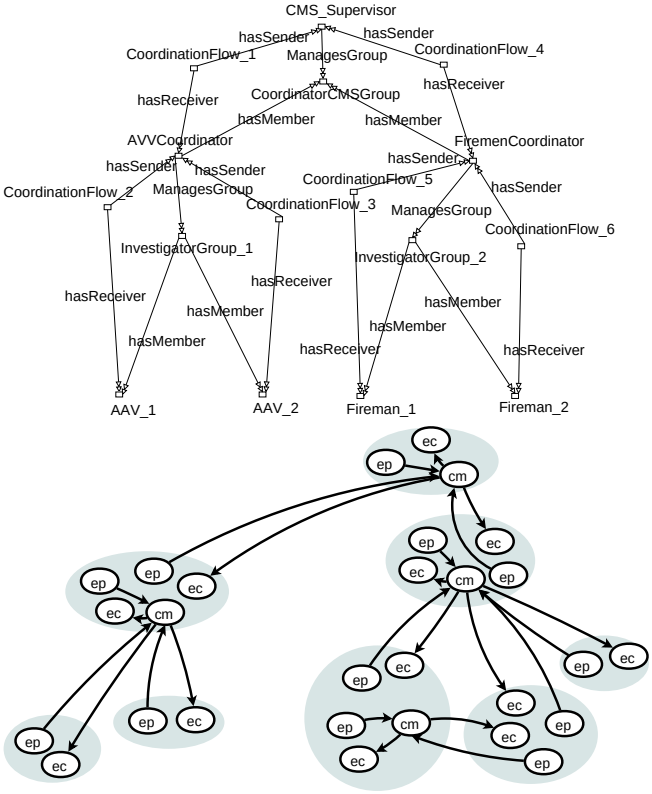

Fig. 3. EBC deployment: ontology and graph after initial refinement

agers, event consumers and event producers. The resulting set of ontological instances form a collaboration level graph. It is translated into GraphML language by means of XSLT transformation. In order to refine this collaboration level, a detailed graph grammar is used. This produces a valid configuration that contains terminal nodes only (i.e. nodes belonging to the EBC level). It is obtained by application of the sequence graph grammar production. This refinement creates a detailed deployment descriptor graph used by a deployment service in order to utilize the indicated components on each device, thus implementing the required activity level session.

Here two situations are considered. The first one deals with an activity-based event and second with resource change event.

Firstly, consider a situation where an investigator (AAV) intends to drop water in an area where another investigator (denoted as Fireman2) is already busy. The simplified version of this situation at activity and communication levels is shown in Figure 3.

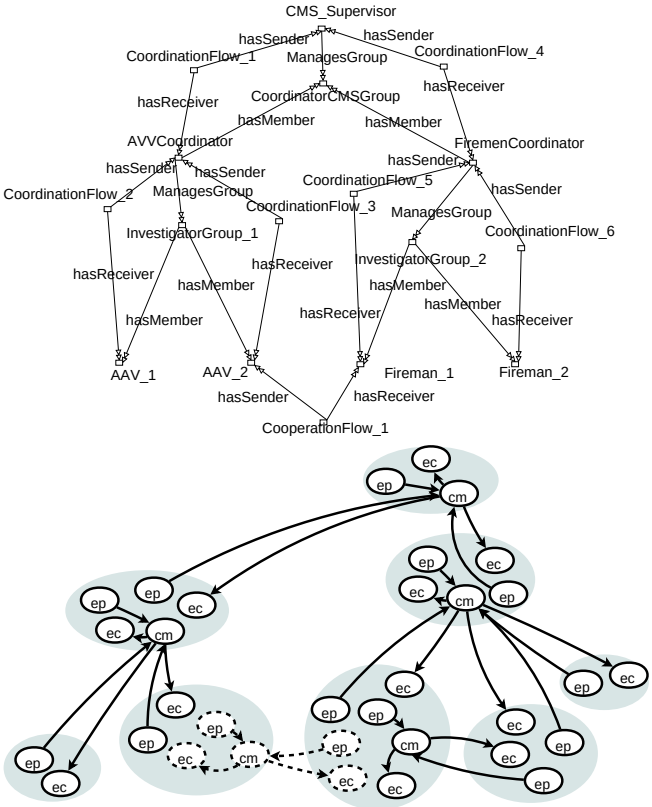

Fig. 4. EBC Deployment: ontology and graph after Reconfiguration Event

In this case, the AAV has to be notified as soon as possible, not to drop water. Another investigator (denoted as Fireman1), aware of the situation, establishes a coordination flow to its coordinator and subsequently the coordinator reports to the supervisor. The latter already knows the position of the approaching AAV and notifies the AAV not to drop any water in that area via the AAV coordinator.

As there is no connection between the AAV and the Fireman2, the latter has to obtain the supervisor's decision. The other simpler solution could be to establish a connection between the AAV and Fireman1 through use of a new cooperation flow obtained by running the SWRL rule (Table 1). Thus, Fireman1 can warn the AAV not to drop any water. After processing the rule, the activity level and the communication level have been changed as shown in Figure 4. With the new cooperation flow between Fireman1 and the AAV, a channel manager is needed in Fireman1's device and the corresponding event producer and event consumer must be deployed in the AAV (represented by dotted line on Figure 4).

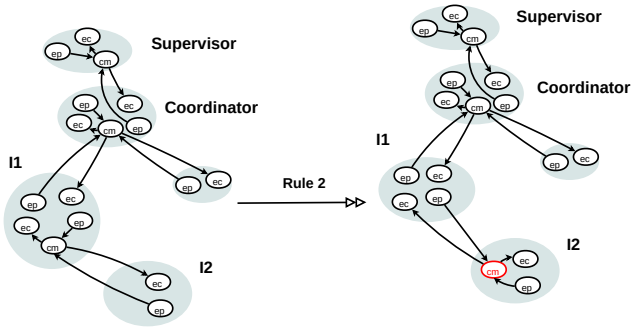

Fig. 5. EBC Redeployment after Power Diminution Event

An example for context-awareness is illustrated in the following example. Consider an investigator (denoted as I2) sending messages to his coordinator via another investigator (denoted as I1). As each one is entrusted with his own task, may arise a situation where I1 could not handle this communication as he needs to move to another location or his power runs out. So he triggers the coordinator to make a decision because communication between I2 and coordinator is very important. In case of I1 moving to somewhere else, a solution could be another investigator establishing a new connection with I2.

In case of I1's power deficiency, a solution is to use the decision that is explained in Table 2. In the first part of the rule, the current situation is identified and due to the power deficiency, the procedure `dischargeParticipantDevice` is triggered. In our case, the channel manager moves from I1 either to coordinator or to I2. By doing so, I2 can send this prioritized flow for a sufficient time through I1 to coordinator. Clearly, we act only at the communication level, i.e, on EBC. Nothing has been modified in the activity level and the new EBC descriptor graph is shown in Figure 5.

7 Implementation

We implemented our work using FACUS (Framework for Adaptive Collaborative Ubiquitous Systems), an architecture that supports semantic adaptation enabling the awareness of the presence/absence, roles and tasks of collaborators. This framework is based on a generic multi-level modeling approach that ensures multi-level adaptation. A generic collaboration model, based on Semantic Web technologies is proposed in order to support real-time collaboration between groups of participants working together in different tasks. The framework defines common interfaces for collaborative systems to enable the management of cooperative actions.

In this framework, a node represents a communicating entity which takes part in a collaborative activity. Nodes may represent human users (i.e. human-controlled software components) but also autonomous software components,

agents, etc. Whether a node is an autonomous software component or it is a human-controlled component, it has to be executed on a physical machine. Such machines are represented by the concept Device (Node is linked to Device by the property hasHostingDevice). The execution context of the node will depend on the resources of the device that hosts it. At the present time, a minimal set of device properties is considered, containing IP addresses (hasIpAddress), operating system (hasOS), available memory (hasAvailableMemory), CPU load (hasAvailableMem) and battery level (hasBatteryLevel).

The concept Flow represents a communication link between two entities. Therefore, Flow is linked to Node by two properties: hasSource and hasDestination. In this ontology, flows are considered as being unidirectional, and thus if a bidirectional communication between two nodes is required, it will be represented by two instances of Flow with two opposite directions.

In order to handle data flows, nodes use external software components that are deployed on the same device as them. These external components are represented by the Tool concept. Tools are composed of several components, e.g., a sender component and a receiver component. Therefore the Tool concept is related to a concept called Component through the property hasComponent. Since components handle flows, a property called managesFlow links Component and Flow. Components have a data type (the same as the data type of the flow that they manage) and are deployed on a single device (isDeployedOn property which links Component and Device). SenderComponent and ReceiverComponent are linked to Flow by two sub-relations of managesFlow: sendsFlow and receivesFlow, respectively.

Finally, the Session concept represents a set of flows belonging to the same collaborative activity. The hasFlow property relates a session to a flow. The inverse property, belongsToSession, is functional, i.e., a flow belongs to a single session. Since flows are related to nodes, nodes are indirectly related to one or more sessions depending on the flows that connect them to other entities.

In FACUS, we have chosen one group (fireman)to show the adaptation. Initially, the fireman1 and fireman2 are connected to firemancoordinator at WiFi infrastructure mode. This initial stage of this situation at application, collaboration and massaging levels are shown in Figure 6, Figure 7 and Figure 8.

Consider fireman1 lost the connection with its coordinator while searching for a victim. Once the connection is lost, the coordinator aware this situation and thanks to our policies, the coordinator and the other investigator will shift to adhoc mode. Also, the local decision of the lost investigator changes to adhoc mode automatically such that communication is established. Figure 9 and Figure 10 show the adaptive collaboration and middleware graph in the implementation.

8 Conclusion and Open Issues

In this paper, a multi-level modeling approach designed to support group communications has been detailed. For such a complex system, the whole scenario has been divided into different levels. Ontology has been used at the top two

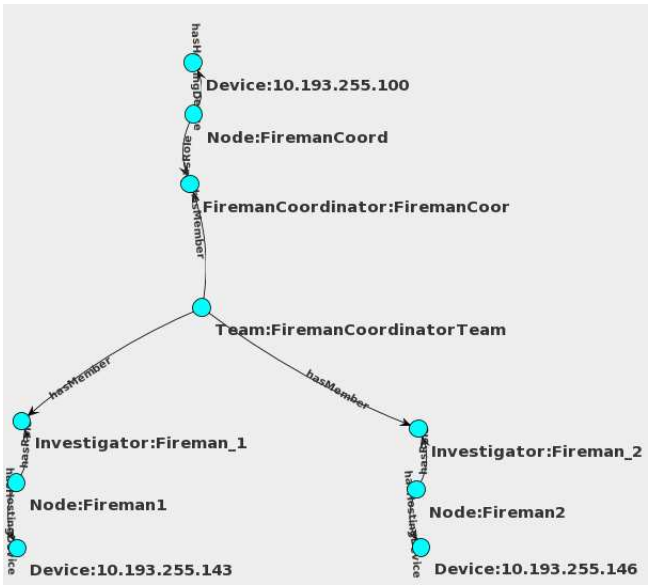

Fig. 6. Application instance at Initial phase

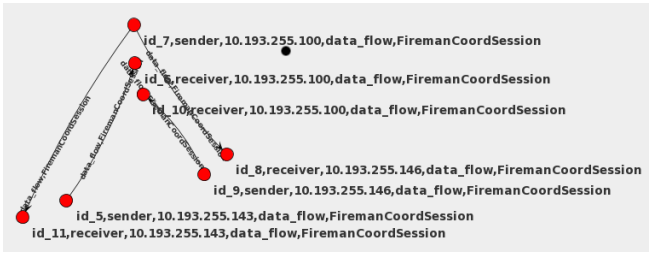

Fig. 7. Initial Collaboration instance

levels while we retained event-based communication at the third level to establish the flows between devices. The relations and transformations from top level to lower levels are presented. If a change arises in the environment, reconfiguration can be achieved by using SWRL rules. This is key for an appropriate management in case of changing resources in the environment. By using this approach, we could allow the architectural reconfigurations at run-time to handle the activity's evolving conditions. Unlike previous approaches, distributed decision model allows our architecture to make adaptive mechanism at lower levels and not influencing the higher ones.

Many open issues need to be discussed in our scenario. In case of environment change, triggers play an essential role to notify the decision components for

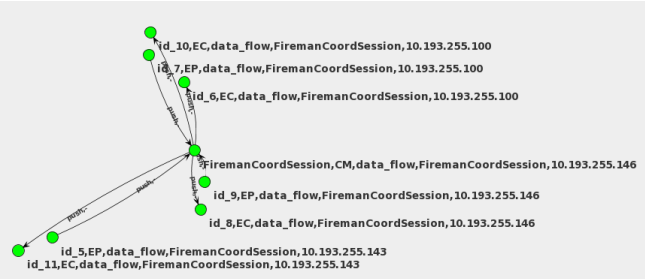

Fig. 8. Initial middleware graph

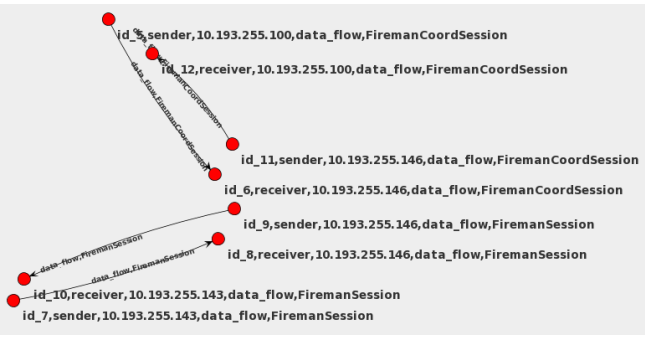

Fig. 9. collaboration Instance after adaptation

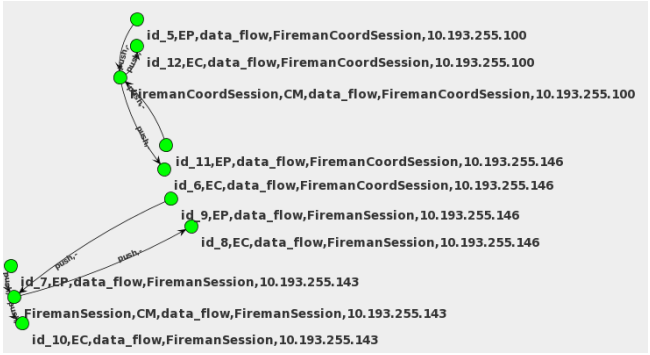

Fig. 10. Middleware graph after adaptation

initiating the adaptive policies. Even though the triggers are asynchronous and synchronous messages, a generic model for failure cases need to be analyzed. Also, assigning priorities to flows as well as monitoring resource deficiencies are

our main challenges. Non-cooperative situations like low QoS and performance degradation are the topics worth for future work.

Acknowledgments

This research is supported by the French project Rosace (RObots et Systémes Auto-adaptatifs Communiquants Embarquès).

References

1. Ketfi, A., Belkhatir, N., Cunin, P.Y.: Adaptation dynamique, concepts et experimentations. In: Proceedings of ICSSEA. (2002) In French.
2. Nasser, N., Hassanein, H.: Adaptive bandwidth framework for provisioning connection-level qos for next-generation wireless cellular networks. Canadian Journal of Electrical and Computer Engineering **29**(1) (2004) 101–108
3. Sun, J.Z., Tenhunen, J., Sauvola, J.: Cme: a middleware architecture for network-aware adaptive applications. In: Proc. 14th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications. Volume 3., Beijing, China (2003) 839–843
4. Exposito, E., Senac, P., Diaz, M.: FPTP: the XQoS aware and fully programmable transport protocol. In: Proc. The 11th IEEE International Conference on Networks (ICON'2003), Sydney, Australia (2003)
5. Welch, L.R., Masters, M.W., Madden, L.A., Marlow, D.T., Irey, IV, P.M., Werme, P.V., Shirazi, B.: A distributed system reference architecture for adaptive qos and resource management. In: Proceedings of the 11 IPPS/SPDP'99 Workshops Held in Conjunction with the 13th International Parallel Processing Symposium and 10th Symposium on Parallel and Distributed Processing, London, UK, Springer-Verlag (1999) 1316–1326
6. Lavinal, E., Desprats, T., Raynaud, Y.: A multi-agent self-adaptative management framework. International Journal of Network Management **19**(3) (2009) 217–235
7. Strassner, J., Meer, S., O'Sullivan, D., Dobson, S.: The use of context-aware policies and ontologies to facilitate business-aware network management. J. Netw. Syst. Manage. **17**(3) (2009) 255–284
8. Smith, M.K., Welty, C., McGuinness, D.L.: OWL Web ontology Langage Guide. W3C Recommendation (2004) Url : http://www.w3.org/TR/owl-guide/.
9. Sancho, G., Tazi, S., Villemur, T.: A Semantic-driven Auto-adaptive Architecture for Collaborative Ubiquitous Systems. In: 5th International Conference on Soft Computing as Transdisciplinary Science and Technology (CSTST'2008), Cergy Pontoise (France) (2008) 650–655
10. Meier, R., Cahill, V.: Taxonomy of distributed event-based programming systems. In: ICDCSW '02: Proceedings of the 22nd International Conference on Distributed Computing Systems, Washington, DC, USA, IEEE Computer Society (2002) 585–588
11. Horrocks, I., Patel-Schneider, P.F., Boley, H., Tabet, S., Grosof, B., Dean, M.: SWRL: A Semantic Web Rule Language Combining OWL and RuleML. W3C Member Submission 21 May 2004 (2004)