A framework of models for QoS-oriented adaptive deployment of multi-layer communication services in group cooperative activities

K. Guennoun, C. Chassot, K. Drira*, F. Armando, N. Van Wambeke, E. Exposito

LAAS-CNRS, University of Toulouse, INSA, 7 Av. Colonel Roche, 31077 Cedex 04, Toulouse, France

1. Introduction

Dealing with dynamically changing contexts when elaborating solutions for QoS-driven self-management is a challenging research area for the design of autonomic communication systems. Managing the adaptation which is necessary to face changes induced by dynamically discovered situations is complex and cannot be based on switching rules between predefined decisions. Policy-based, rule-oriented approaches are likely to be the most appropriate solutions for dealing with unpredictable changes and possible evolutions that may occur within the many levels of the end-to-end communication systems and their various surrounding contexts. Our work for the last few years has addressed such challenges by associating and integrating solutions from the model-based adaptability management, dynamic software architectures, and reconfigurable communication protocols. The complexity as well as the variability of context change situations and adaptation criteria required associating analytical models [1,2] and structural models [3,4] for the adaptation-related planning process. The adaptation enforcement is performed at the Transport level (TCP, UDP level) and the above group-messaging-support Middleware level. The deployment of communication services for QoS-enabled end-to-end group communication systems. The Transport level (TCP, UDP level) and the above Middleware level are considered as the two communication levels targeted by the QoS-driven adaptation process. Application to crisis management systems (CMS) is considered as a case study from the more general domain to which our results apply: cooperative activity support systems. The adaptation rules rely on graph matching and graph rewriting. The adaptation enactment is based on the dynamic composition of micro-protocols at the Transport level and on the dynamic binding of software components and services at the Middleware level. The deployment model is used as a central feature of service provisioning. The influence of the cooperation and communication contexts is expressed and maintained consistent by automated graph-based model refinement and transformation.

© 2008 Elsevier B.V. All rights reserved.
framework of models. We focus on the models that illustrate the deduction and the transformation chain which guarantees the correctness of the automation process. We describe how the deployment architecture can be deduced from the cooperation context and dynamically adapted to its changes. We show how we can model the connection dependencies at the communication level and how we can adapt this model to the changes in the network and the Middleware deployment architecture.

This paper is organized as follows. After this section, we present, in Section 2, a classified synthesis of the existing work related to adaptability management. Then, in Section 3, we discuss the adaptability problem in group communication and cooperation illustrated by the crisis management systems (CMS) scenario. In Section 4, we present a framework of models that rule the adaptation process. We focus on handling simultaneously, and in a consistent way, unpredictable changes in the communication context and predicted evolutions in the cooperation context. In Section 5, we illustrate the automated transformation and deduc tion chain by giving the main models elaborated for the CMS scenario. In Section 6, we present our concluding remarks and future work.

2. Related work

In this section, we propose a summarized synthesis and classification of context-aware adaptation solutions and dynamically configurable Transport protocols.

2.1. Classification of context adaptation solutions

Adaptation objectives, techniques and properties are among the main facets of adaptability. They are studied and classified in this section.

2.1.1. Adaptability objectives

Adaptability targets several objectives. QoS aspects such as connectivity or access bandwidth issues in roaming are considered in [8]. End-to-end QoS optimization in the best effort Internet makes heavy use of adaptation techniques [9]. Security in wireless networks, such as firewall activation and deactivation, can also benefit from adaptability [10]. Resources optimization related to device power, computation or storage capability are presented in [11].

2.1.2. Adaptation techniques

Adaptation techniques target all layers of the OSI model. Application layer. [12] addresses adaptation of video streaming applications for the best effort Internet. The proposed techniques are based on two mechanisms: an applicative congestion control (rate control, rate-adaptive video encoding) and time-aware error control with FEC. Middleware layer. Reflective architectures such as OpenORB or Xmiddle [13] are good supports for adaptation as they allow run-time modification of the architecture.

Transport layer. TCP's congestion control is a well-known adaptation example. The IETF DCCP protocol allows users to choose the congestion control. SCTP targets adaptation to network failures using multi homed associations. In [9], the authors study various types of mobile applications in wireless Internet. Adaptation consists in parameterization of congestion control mechanisms using context information. [5,14] study the architectural adaptation of Transport protocols by dynamic composition of protocol modules. Next section (2.2) is dedicated to these frameworks illustrating the modular architecture concept targeted by our work.

Network layer. [15] addresses QoS-aware routing problems within Ad hoc mobile networks. In [10], dynamic and secure provision of IP services for military wired/wireless networks is considered. In a policy-based networking management context, the need for self-adaptation is considered in [16], using a learning-based approach.

MAC layer. The solutions handle connection and access QoS problems for mobile users using different terminals and roaming. [8] provides a solution for optimizing the handover latency but the other QoS requirements are not considered.

2.1.3. Adaptation properties

The adaptation solutions suggested in the literature are defined in various ways. The adaptation is behavioral when the execution of a service can be modified without modifying its structure. TCP and specific protocols such as the ones in [9] provide behavior-based adaptation. It is easy to implement but limits the adaptability range. Indeed, the addition of new behaviors requires the component to be recompiled and the adaptation can no longer be performed during run-time. The adaptation is architectural when the structure of adapting services can be modified. The replacement of a component by another can be implemented following a plug and play approach where the new component has the same interfaces as the replaced one. Finally, adapting components can reside on a single machine or be distributed. In the first case, adaptation is vertical and changes are performed only locally. In the second case, it is horizontal and synchronization between peer adapting entities has to be managed. Table 1 summarizes the adaptability categories introduced in this section.

2.2. Dynamically configurable protocol architectures

Dynamically configurable protocol architectures are based on the protocol module concept. A protocol module is a primitive building block [14] resulting from the decomposition of the protocol's complexity into various successive elementary functions. A protocol is then viewed as the composition of various protocol modules in order to provide a global service. These architectures can be refined into two different categories depending on their internal structure: the event-based model (followed by Coyote and Cactus) and the hierarchical model (X-Kernel [14] and APPIA). ETP follows a hybrid approach combining both models [5]. These protocol architectures appear as a good choice for future communication protocol's self-adaptation as they are capable of run-time architectural adaptation, meaning that the modules composing them can change during the communication. This run-time architectural adaptation raises many problems such as: (1) synchronization of adapting peers; and (2) the choice of the best composition guided either by the user's requirements or by the modification of the context.

Table 1

<table>
<thead>
<tr>
<th>Classification</th>
<th>Objectives: QoS, security, resources optimization, cooperation, etc.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actions</td>
<td>Rate control, rate-adaptive video encoding, FEC, etc.</td>
</tr>
<tr>
<td>Middleware</td>
<td>Reflective Middleware, etc.</td>
</tr>
<tr>
<td>Transport</td>
<td>Partially reliable error control</td>
</tr>
<tr>
<td></td>
<td>Configurable congestion control</td>
</tr>
<tr>
<td></td>
<td>Protocol module composition, etc.</td>
</tr>
<tr>
<td>Network</td>
<td>Secure IP service configuration, etc.</td>
</tr>
<tr>
<td></td>
<td>QoS-oriented vertical handover, etc.</td>
</tr>
</tbody>
</table>

2.3. Need for model-driven multi-level adaptation

Adaptability management still remains a complex and unsolved problem, particularly when it is required at several abstraction levels simultaneously [17]. In such cases, the need clearly appears to ensure coherency of the adaptation choices, both within and between adaptation levels. Tackling this need by informal models may lead to specific or suboptimal solutions because of problem’s complexity. For instance, for CMS-like cooperative activities, both variations of communication/devices resources and the evolution of participants’ cooperation structure have to be managed when considering communications QoS. However, managing adaptation at various levels requires coordination without which it can lead to performances way below the targeted ones. For example, having to react to network congestion, an adaptation of the sending rate both at the application layer (e.g. by reduction of images size) and at the Transport layer (e.g. by a rate control) could result in over-reaction and as such a non-optimal solution. In front of limitations of ad hoc and informal models, we propose a model-based framework for QoS-driven adaptability management. This framework has been elaborated in the context of CMS-like activities with QoS provisioning at the Transport and Middleware levels as the final objectives.

3. Self-adaptation properties of future communication systems for group and mobile communication activities

Recent advances in computing and networking technologies enable deployment of complex group communication activities such as crisis management systems. Such activities involve structured teams of static and mobile users who cooperate within a common mission (e.g. human saving in case of tsunami). From a communication point of view, multiple time-varying requirements and constraints have to be satisfied: the evolving requirements come from several factors such as the applicative tools used by the participants, e.g. interactive audio/video applications. Depending on the codecs (e.g. H263 for a video flow) used by these applications, different QoS requirements (i.e. throughput, loss rate, delay, etc.) have to be satisfied to ensure a correct distribution of the involved media. These requirements also depend on the cooperative activity itself: for instance in a CMS-like activity, the cooperation links and their relative importance may evolve from one step of the mission to another; simultaneously, multiple time-varying constraints (e.g. bandwidth, energy, etc.) are also to be considered, depending on machine and network resources which are used by the participants’ hosts. Thanks to wireless facilities and battery support, the participants may move during their communications. Consequently, the network/machine resources they have at a given time may not be available anymore at a later moment. Tackling these two kinds of evolving requirements and constraints cannot be enforced by current communication systems. To face this limitation, new architectural self-adaptation properties are expected to be enforced coherently at several levels of the communication stack. The first goal of this paper is to illustrate the interest for such properties for end-to-end protocols, more specifically at the Middleware and Transport levels. In this section, we illustrate the interest of these two adaptation levels through a typical CMS example. The CMS example is presented in Section 3.1; some typical examples of architectural adaptation actions at both Middleware and Transport levels are illustrated in Section 3.2.

3.1. The CMS scenario

The CMS scenario, described on Fig. 1, handles structured groups of communicating entities that cooperate to achieve a common goal. These entities have various roles and unequal communication resources, CPU and energy. They are deployed on fixed and mobile machines and communicate through wirelined and wireless networks.

In our example, cooperation is based on data exchange between participants. In particular, observation data (O) and report data (R) are produced either periodically or immediately after a particular event. O data describe the environment observed by an investigator. It corresponds to a description of its immediate environment like the topography of the investigated area, temperature, wind speed, etc. R data express the analysis of the situation by an investigator or a coordinator. It may correspond, for example, to comments linked to O data (e.g. analysis of the dangerousness of the situation) or to reports related to the state of the mission.

Fig. 1. Example of CMS application.

The controller is the entity that supervises all the application. She/he receives the reports of all coordinators that synthesize the current context and inform him of mission achievements. The controller uses a fixed machine which has permanent access to energy and high communication and processing resources.

According to actions and objectives assigned by the controller, a coordinator manages a section of investigators by giving orders and assigning tasks to be performed. She/he must also collect, interpret and synthesize information received from her/his investigators and possibly forward them to the controller. The coordinators also use mobile machines.

The investigators explore the operational field, they observe, analyze and submit reports describing the situation to their assigned coordinator. They use mobile machines and have, therefore, limited resources of energy and CPU.

Two possible scenarios may be considered for each section: the exploration step and the action step. In the exploration step ("section B" of Fig. 1), mission achievement implies the following data exchanges for cooperation:

- The investigators continuously send O data to their assigned coordinator. They also periodically send data of type R.
- There is no priority difference between cooperation involving each coordinator and her/his investigators but O feedbacks have a higher priority than R feedbacks.
- The coordinators periodically send R reports to the controller describing the current state of the exploration. All coordinators in exploration step have the same priority to communicate with the controller.

The exploration step ends when a critical situation is discovered by an investigator (e.g. I1 on Fig. 1). This situation implies a reconfiguration of the architecture and the application moves to a new execution step called the "action step". This reconfiguration specifically affects the coordinator Coord1 that controls the investigator I1 as well as all the investigators of Coord1 (i.e. I2 and I3). The other coordinators and their related investigators are not affected by this reconfiguration. At the time of moving from the exploration step to the action step, the coordinator Coord1 and her/his associated investigators react in the following way:

- After having discovered the critical situation, the investigator I1 in this scenario, continues to fulfill the same functions as those performed during the exploration step (i.e. observing and reporting), but she/he also sends its observations data O to the other investigators of his section. She/he continues to send the two types of data (R and O) to the coordinator.
- The other investigators only report R data to the coordinator; those reports correspond to a complementary analysis of the O data sent by investigator I1. Additionally, new priorities are established between these cooperation links: O data exchanged between I1 and Coord1 have a high priority; R data exchanged between I1 and Coord1 have a medium priority; O data (respectively: R data) exchanged between I1 and I2/I3 (respectively: between I2/I3 and Coord1) have a low priority.

3.2. Middleware and Transport level self-adaptation for CMS-like activities

For CMS-like group and mobile cooperative activities, the first goal of this paper is to illustrate the interest in architectural self-adaptation properties in the end-to-end protocols, when the cooperation context is evolving (as illustrated in the previous section) or when the communication context is modified. Two layers of the communication stack are considered in our work, the Middleware and the Transport layers:

- The Middleware layer implements the cooperation links following different possible paradigms, such as client/server or publish/subscribe paradigms. In our work, the publish/subscribe paradigm has been chosen, as the most commonly used for distributed group applications. This communication model is also used in other contexts, for instance for SIP-based session management [18]. It has also been adopted in content-based networking (CBN) [19].
- The Transport layer implements the Middleware level links by means of protocols such as UDP, TCP, SCTP, DCCP and configurable Transport protocols such as ETP. In our work, the ETP protocol framework has been chosen to illustrate the interest in architectural adaptation properties at the Transport level. The use of different kinds of protocols together will be considered in future work.

Let’s introduce these two adaptation levels by means of a simple example applied on our CMS scenario. Let’s consider a CMS participant (e.g. I1 on Fig. 1), when she/he discovers a critical situation, and then goes from the investigation step to the action step. This new situation induces the need for I1 to maintain three cooperation links: one with her/his coordinator Coord1 (for the transfer of O and R data) and two other ones with the other investigators I2 and I3 of her/his team (the transfer of O data).

In order to simplify the following illustrations, let us focus on the O data exchanged between I1 and Coord1 (link whose priority is high) and on the O data exchanged between I1 and I2 (link whose priority is low). As a result of their priority difference, the cooperation link between I1 and Coord1 may have to be provided with a higher QoS than the one provided to the link between I1 and I2. Facing this new situation, the following section introduces typical examples of adaptation actions that can be done at the Middleware and Transport levels of the communication stack.

3.2.1. Middleware level architectural adaptation

At the Middleware level, the link between I1 and Coord1 may be implemented following different paradigms (e.g. client/server or publish/subscribe). In both cases, several possibilities may be considered regarding the deployment of the Middleware level entities, i.e. event producer (EP), channel manager (CM) and event consumer (EC) entities for the publish/subscribe mode. For instance, an appropriate adaptation action could be to deploy the CM(s) on machine(s), taking into account power, storage or energy constraints. Analogously, the choice of the pull/push communication mode between EP and CM (or between CM and EC) may also be guided by constraints-oriented considerations. These cases illustrate some typical examples of architectural adaptation at the Middleware layer (Fig. 2).

3.2.2. Transport level architectural adaptation

At the underlying Transport, network and data link levels, if the network resources are not sufficient to satisfy the QoS requirements of all I1’s cooperation links, an appropriate action would be to adapt one or many parts of the communication protocols in order to privilege (the Middleware level links that implement)
the cooperation link between \( I_1 \) and \( Coord_1 \). Particularly, at the Transport level, a suitable action could be to decrease the sending rate of \( I_1 \)'s connections which share critical network resources (such as common overloaded routers) with the connection(s) underlying the cooperation link between \( I_1 \) and \( Coord_1 \) (left part of Fig. 3). This case illustrates a typical example of behavioral adaptation at the Transport level: only parameters of the congestion control module are modified, but the internal architecture of the protocol remains unchanged.

Now, if we assume that \( Coord_1 \), whose machine is supposed to be connected to a wired access network, decides to move toward \( I_1 \)'s area, then she/he will have to use a wireless access network to communicate with her/his investigators. An appropriate adaptation action on \( Coord_1 \)'s host could be to replace the congestion control module of Transport entities, in order to take into account the new network context (right part of Fig. 3). This case illustrates a typical example of architectural adaptation.

4. The proposed framework

The adaptation decision process can be driven by different models representing the context and its evolution. The context changes
can be related to the evolution of the requirements associated to the activity (e.g. priority and QoS requirements) and to the changes in the constraints associated with the communication resources (e.g. connection performances, connection dependencies, access networks, etc.). The adaptation enactment process can be conducted by dynamically initiating and changing the deployment of the software entities handling the communication activity. This is formalized here, by modifying the deployment architecture of the Middleware and the Transport levels (i.e. Middleware event producer (EP), channel manager (CM), event consumer (EC) and Transport protocol modules). At both Middleware and Transport levels, the deployment choice issue may be formalized as a multi-criteria optimization problem. At the Transport level, starting from a proper formal description of the candidate protocol modules, we defined, in [1,6], an analytical reasoning process aimed at deciding which protocol modules to compose in a given communication context in order to optimize the composition’s efficiency with regard to the required QoS. At the Middleware level, similar work has been initiated to address the deployment choice of Middleware level entities taking into account a simple model of the energy available on the end hosts [2]. Both deployment decision models require the knowledge of the structure of interaction relationships (representing the network of producers and consumers) and associated communication modes (push/pull) for the Middleware level deployment, as well as the structure of Transport connections and associated attributes for the Transport level deployment. The functional efficiency and the performance of such a multi-level adaptation depend on the consistency of all the models driving the adaptation chain. Graph-based models associated with graph transformation techniques are provided for this purpose. Two kinds of graph transformations are considered: reconfiguration (or intra-level transformation) and refinement (or inter-levels transformation).

4.1. General overview of the framework of models

The framework is composed of three main models: the cooperation model, the connection model and the adaptive deployment model:

- The first two models are used to represent the context: the cooperation model represents the activity level cooperation context; the connection model represents the communication context.
- The adaptive deployment model is composed of two sub-models: the Middleware deployment model and the Transport deployment model; they represent the deployment choices of the Middleware and Transport entities that fit with the requirements and constraints expressed by the Cooperation model and the Connection model, respectively.

The relationships between these models (presented hereafter) are summarized on Fig. 4.

The activity requirements are derived from the cooperation context and captured by the cooperation model. The cooperation context captures the changes occurring at the activity level. Changes include modifying activity phases, role distribution, modifying priorities between roles and applications, modifying QoS parameters of applications, media and codecs, dynamic group membership, and connectivity failures. The algorithm described in Table 2, describes, in an abstract way, how to manage the changes in the cooperation context.

The adaptive deployment model (ADM) is generated from the two cooperation and communication context models. It is composed of two sub-models, the Middleware deployment model (MDM) and the Transport deployment model (TDM).

The Middleware deployment model (MDM) represents the different software components supporting the information exchange between the different actors of the cooperative activity. Such components are event producers, event consumers, and channel managers interacting following the publish/subscribe paradigm or basic clients and servers interacting through direct message exchange. The different bindings of information requesters to information providers and the different interaction modes are also elements of the MDM. These elements can change for adaptation purposes at run-time.

The Transport deployment model (TDM) represents the transport level decisions. In the case of dynamically configurable protocol architectures, the different protocol modules as well as their configuration attribute values are represented. The TDM depends on the connection model (CM) resulting from the refinement of the MDM. This constitutes a first dependency between the TDM and the CM.

Moreover, the CM may change by itself (while the MDM is unchanged), due to variation in routing information provided by monitoring, and then the TDM has to be adapted to this change, and this is a second dependency between TDM and CM.

The MDM may change for example, because a deployment node becomes highly loaded (information provided by monitoring the...
MW level) and a channel manager has then to be moved on a less loaded node. The CM is then automatically recomputed because sharing hosting nodes between communicating entities also has influence (as sharing routers) on the QoS parameters of a connection, so the CM changes and so does the TDM which directly depends on it. This is a two-steps adaptation (associated to an indirect dependency): changes in MDM lead to changes in the CM which lead to changes in the TDM.

4.2. The elaborated techniques and models

By integrating graph-based models and analytic models, we developed a chain of rule-based architecture refinement and transformation models constituting a selection framework for the adaptation decision process. Fig. 5 provides a global view of the elaborated models, their relationship, and the different techniques used for automating their implementation.

The cooperation model involves (1) characterizing all valid configurations as elements of a graph grammar, (2) describing specific valid configurations, (3) defining, as graph transformation rules, all the possible consistent structural changes at the cooperation level. This formally provides the valid reconfiguration rules and actions to be performed in reaction to changes in the cooperation context including resource-level variations and mission-level changes. The cooperation model is deduced from the cooperation context which may change (e.g. from one phase of the activity to another). Facing such changes, reconfiguration rules are used to automate model adaptation as graph transformation rules [4]. For graphs handled at the cooperation level, node labels represent actor identifiers, cooperation roles, hosting devices, and mission execution phases. Edge labels represent exchanged data, priority, and the required level of QoS. The Middleware deployment model distinguishes three types of communicating entities: data consumers (C), data producers (P), and event managers, also called channel manager, or event services (G). The deployment model also describes the distribution of these entities on the hosting devices, the communication links between the different entities and the kind of interaction (pull/push) they use to exchange information. The deployment model is described by a graph where nodes represent the communicating entities. The nodes are labelled by the type of the components (P/C/G) and their hosting devices. Edges are labelled by related attributes.

Table 2

<table>
<thead>
<tr>
<th>Managing changes in the cooperation context</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wait an event from the activity-level monitoring services</td>
</tr>
<tr>
<td>(1) Consider changes in cooperation flows, role distribution, inter-roles and inter-applications priorities, media/codec QoS, group membership and connectivity</td>
</tr>
<tr>
<td>(2) Build the updated cooperation model associated with the new cooperation context</td>
</tr>
<tr>
<td>(3) Calculate the new MW deployment model associated with the new cooperation model</td>
</tr>
<tr>
<td>(4) Select the optimized configuration with regard to cooperation role priorities and QoS</td>
</tr>
<tr>
<td>(5) Apply deployment</td>
</tr>
<tr>
<td>(6) Build the new connection model associated with the selected configuration</td>
</tr>
<tr>
<td>(7) Deduce the corresponding new deployment configurations for the Transport level; select the optimized configuration with regard to communication priorities and QoS</td>
</tr>
</tbody>
</table>

Table 3

<table>
<thead>
<tr>
<th>Managing changes in the communication context</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wait an event from the network-level monitoring services</td>
</tr>
<tr>
<td>(1) Consider changes in routers and resources load, connection performance</td>
</tr>
<tr>
<td>(2) Deduce changes in connection dependencies, in performance per connection values, in access network types</td>
</tr>
<tr>
<td>(3) Build the updated connection model associated with the new communication context</td>
</tr>
<tr>
<td>(4) Calculate the new deployment configurations for the Transport level; select the optimized configuration with regard to communication priorities and QoS</td>
</tr>
</tbody>
</table>

Fig. 4. Relationship between the context elements and the framework of models.
communication characteristics such as QoS and priorities. The kind of interaction may also be considered in edge labels. The Middleware deployment model is deduced from the cooperation model using model refinement rules expressed as graph-grammar productions. It supports two different standard architectural styles: the client/server and the publish/subscribe styles. Refinement from the cooperation model into the deployment model is implemented using extended edNCE graph grammars (see Section 5.3). The refinement system we developed allows the generation of all the consistent deployment configurations with respect to a given cooperation model. On the other hand, it also allows automated verification of the conformance of the dynamically evolving cooperation and deployment instance models. Management based on graph rewriting models acts as follows: if a given rule is applicable (a matching is found between the graph representing the rule and the graph representing the MDM), then the system will execute this rule to adapt the MDM as specified by the designer in the matching rule:

1. moving components or services from one deployment node to another (modelled by changing the attribute in a graph node: e.g. last attribute, of type “M”, of nodes in the graph depicted in Fig. 7),

2. or modifying priorities of communication links (modelled by changing the attribute in a graph label: e.g. last attribute, of type “high/low/medium”, of edges in the graph depicted in Fig. 7). The applicability of the rule means that the current situation matches one solution (among multiple others) to adapt the deployment (at the Middleware level for example of Fig. 7). When a rule is applicable, it is immediately applied and the remaining rules are no more tested. We do not separate “finding all applicable rules” and “choosing and applying a rule”. This is possible but could be costly. Our current approach does not handle considering policies such as: test the applicability of all the rules to determine which set is applicable, and then chose the most appropriate. For doing so, we can consider solutions handling priorities between rules according to statically or dynamically defined appropriateness or efficiency attributes such as “adaptation cost” in time. We can consider moving the less number of components which may be statically defined by analyzing and comparing the rules. We can also consider selecting the less loaded node to move a component. This is dynamically defined and may be associated with Graph rewriting-based model.

The algorithm presented in Table 4 describes the graph-based management of changes in the cooperation context.

The connection model is deduced from the Middleware deployment model following a set of model transformation rules expressed as graph-grammar productions. The connections are represented by graph nodes as first level elements of the connection model. Dependant connections are identified as immediate neighbours in the graph. Node labels, such as \((c1, R, QoS, ANMC, ANMC, Perf_{c})\) in Fig. 8 represent, respectively, the identifier of the connection, the type of data transported through the connection, the QoS required for the data, the priority of the connection, the access networks of the sending and the receiving machines, and the performances observed on the connection (e.g. throughput, latency, etc.).

![Diagram of model classes and relationships.](attachment:diagram.png)

**Fig. 5.** Classes of models and relationships.

<table>
<thead>
<tr>
<th>Table 4</th>
<th>Graph-based management of changes in the cooperation context</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Let</strong> ( e ) <strong>denotes the event notifying the changes</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Let</strong> ( S_e ), <strong>the set of rules handling</strong> ( e ), <strong>let</strong> ( P_t ), <strong>the associated rule execution protocol</strong></td>
<td></td>
</tr>
<tr>
<td>(1) <strong>Execute</strong> ( P_t ) <strong>to find a rule</strong> ( R ) <strong>in</strong> ( S_e ) <strong>that matches a specified pattern in the</strong> ( \text{CooperationModelGraph} )</td>
<td></td>
</tr>
<tr>
<td>(2) <strong>CooperationModelGraph</strong> := <strong>CooperationModelGraph</strong> <strong>modified following</strong> ( R )</td>
<td></td>
</tr>
<tr>
<td>(3) <strong>Calculate</strong> the new ( \text{MDMGraph} ) <strong>by model transformation from the new</strong> ( \text{CooperationModelGraph} )</td>
<td></td>
</tr>
<tr>
<td>(4) <strong>Select</strong> the optimized configuration with regard to cooperation role priorities and QoS</td>
<td></td>
</tr>
<tr>
<td>(5) <strong>Apply</strong> deployment by moving/activating/deactivating the concerned entities of type: ( G, P, C )</td>
<td></td>
</tr>
<tr>
<td>(6) <strong>Calculate</strong> the new ( \text{ConnectionModelGraph} ) <strong>by model transformation from the</strong> ( \text{MDMGraph} )</td>
<td></td>
</tr>
<tr>
<td>(7) <strong>Calculate</strong> the new deployment configurations for the Transport level, select the optimized configuration with regard to communication priorities and QoS</td>
<td></td>
</tr>
</tbody>
</table>
The sending rate of two dependent connections, still respecting TCP-friendliness property with regard to the other connections. Similarly, an architectural adaptation may consist in replacing the congestion control of a connection of low priority by a less aggressive congestion control, in order to improve the QoS of a dependent higher priority connection. We have defined and evaluated two versions of a collaborative congestion control implementing per connection groups behavioral adaptation using the NS simulator. In the first version, the sending rate of two dependent connections is increased for the connection having the highest priority, and decreased for the other connection. The policy is valid (i.e. maintains TCP-friendliness) as long as a common bottleneck is shared by the collaborating connections. In the second version of the collaborative congestion control, the sending rate of the connection having the lowest priority is decreased without explicitly increasing the others. This leads to an improvement of the QoS for the other collaborating and non-collaborating connections sharing the same bottleneck. This second policy is less efficient than the first one when there are several concurrent connections, but its implementation is simpler as it does not require coordination among the actors.

In order to illustrate the use of the previously presented models, their application to CMS management is presented in the next paragraphs.

### 5. Models for CMS adaptability management

In this section, we illustrate our framework of models applied to the CMS example. Due to the big size of these models (exhaustive study and models of the example can be found in [3]), we limit our illustration to some selected models that cover the different problems and issues introduced in Figs. 4 and 5. First, we show and comment graph-based application models from the cooperation, Middleware deployment and connection points of view. Then, we introduce, as an example, the graph grammar model characterizing the cooperation context in the action phase. Using edNCE graph grammars, we will finally give transformation models that allow ruling and managing architectural reconfigurations within and between different architectural levels.

#### 5.1. Execution description models

Execution models involve characterizing the current state of the application with respect to the different considered points of view related to the cooperation, the Middleware, the Transport and the connection levels. In this section, we introduce these models considering the CMS example.

The description of the cooperation model targets the specification of the current cooperation structure including the cooperating entities and their related dependencies. For our framework, a cooperation model instance is described by a directed graph where entities are described by graph vertices labelled by the relating attributes and where dependencies are specified using edges labelled by the corresponding characteristics. Let’s consider the cooperative model introduced in Fig. 6. This description specifies that the system is composed of one controller (named C), two coordinators (named C1 and C2), and five investigators (named I1 to I5). Corresponding graph vertices are labelled by attributes including:

- (1) the unique identifier (e.g. C1 for the first coordinator, and I1 for its first investigator),
- (2) the cooperative role (e.g. Cont for the controller role, Coord for the coordinator role and Inv for the investigator role),
- (3) The deployment machine (e.g. machine MC1 that hosts coordinator C1 and MI1 hosting investigator I1), and
- (4) for coordinator role, an attribute corresponding to the execution phase to specify whether its section is in exploration ("exp1") or in action ("act2") phase. Considered cooperation dependencies characteristics

<table>
<thead>
<tr>
<th>Table 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Graph-based management of changes in the communication context</td>
</tr>
</tbody>
</table>

- Wait the event notifying the changes
- Consider changes in routers and resources load, connection performance
- Deduce changes in connection dependencies, in performance per connection values, in access network types
- Build the updated connection graph associated with the new communication context
- Calculate the new deployment configurations for the Transport level, select the optimized configuration with regard to communication priorities and QoS

are related to the exchanged data types (\(O\) for the Observed data and \(R\) for the reported data), to the required quality of service (i.e. \(QoS_O\) and \(QoS_R\)), and to the relative priorities between different dependencies (i.e. high, med and low). The presented model instance indicates that investigation is structured in two teams, respectively composed of three investigators (i.e. \(I_1, I_2\) and \(I_3\)) coordinated by coordinator \(C_1\), and two investigators (i.e. \(I_4\) and \(I_5\)) coordinated by coordinator \(C_2\). It indicates moreover that the first team is processing a critical situation (team with asymmetric architecture and coordinator with attribute “\(\varphi_1\)”) with a primary (“\(\alpha\)”) investigator named \(I_1\) and two secondary investigators \(I_2\) and \(I_3\). The second team is described as being in exploration phase (symmetrical architecture and coordinator with attribute “\(\varphi_2\)”).

**Fig. 6.** The cooperation model for the CMS scenario.

**Fig. 7.** The Middleware deployment model graph for the CMS scenario.
channel manager binding the data producer of a “x” primary investigator to the consumer of its coordinator, and (3) the deployment machine. Middleware communication connections are specified as edges linking the concerned components. Characteristics introduced as edge labels include: (1) a unique identifier of the connection (e.g. “C”) for the connection binding the first coordinator to the controller, (2) the “O” or “R” exchanged data type, (3) the required QoS level, and (4) the relative priority of the connection. The exposed instance indicates, for example, that the machine that hosts coordinator “C1” is also hosting three channel managers. The first two are managing communication between investigator I1 and coordinator “C1” (for both O and R data types) whereas the third one manages communication of the R data type between investigators “I2” and “I3” and their coordinator. The fourth channel manager that manages communication between “I2, I3” and “C1” is deployed on the machine hosting investigator “I2.”

The connection graph model describes the current dependencies related to the connections established to support communications between the components of the Middleware level (Fig. 8). Each graph vertex corresponds to a connection. Vertices inherit the characteristics of the upper level (e.g. exchanged data type and QoS) and other architectural properties such as the involved machines (e.g. “MC1” and “MC2” for connection “C1”) and the connection performance. Two connections are neighbors (bound by an edge) if they are interdependent. The edge representing this dependency is labelled by a computed value corresponding to an edge label: (1) a unique identifier of the connection, (2) the “O” or “R” exchanged data types, or (3) the required QoS level, and (4) the relative priority of the connection. The exposed instance indicates, for example, that the machine that hosts coordinator “C1” is also hosting three channel managers. The first two are managing communication between investigator I1 and coordinator “C1” (for both O and R data types) whereas the third one manages communication of the R data type between investigators “I2” and “I3” and their coordinator. The fourth channel manager that manages communication between “I2, I3” and “C1” is deployed on the machine hosting investigator “I2.”

Fig. 8. The connection graph model for the CMS scenario.

5.2. Context description models

The graph models presented in the previous sections describe the current instance of the architecture from a given level (cooperation context, communication context, Middleware deployment, Transport deployment) at a given execution moment (investigation phase, action phase). Describing the application contexts requires specifying all instances that are considered consistent within a given level. For the cooperation level, the application context description corresponds to the description of the dynamic architecture style (i.e. a generative system that produces all cooperation architectures considered as consistent). Describing execution instances using labelled graphs leads logically to use graph grammar-based systems to characterize the context.

A graph grammar is described, using notations similar to a Chomsky’s grammar [20], by a 4-uple $GG = (AX, NT, T, P)$. Where $AX$ denotes the axiom, $NT$ denotes the set of non-terminal terms, $T$ denotes the set of terminal terms, and $P$ denotes the set of productions. We assume that instances belonging to the context are those that correspond to graphs belonging to this grammar. They are graphs containing only vertices of $T$ and obtained from $AX$ by applying a combination of productions in $P$. We use productions of type $(L, K, R)$ representing the structure of a graph grammar production of type DPO [3]. A production is applicable to a graph $G$ if it contains an occurrence of the so-called mother graph $L$. The application of this production involves transforming $G$ by deleting the subgraph $(Del = L \setminus K)$ and adding the subgraph $(Add = R \setminus K)$ while the subgraph $K$ remains unchanged.

The graph grammar $GG_{Coop}$ models the architectural style of the CMS example and constitute a part of the cooperation context when the action step is considered. It is defined by: $GG_{Coop} = (AX, NT, T, P)$, with a set of non-terminals, $NT$, composed of the unique node, “Temp”: $NT = \{\text{Temp}\}$. Terminal vertices correspond to the following set of vertices representing the different types of considered cooperating entities:
The edge of two nodes \( N(...) \) in the informal description of the grammars. In the formal presentation of grammars, they are numbered in order to be distinguished: \( N1(...) \), \( N2(...) \), etc.

The node \( N(X_{\text{Coord}}, "\text{Cont}" , "\rho_1" , M_{\text{Coord}} , "\text{R}" ) \) represents the investigator in action phase \( \rho_2 \) (node \( N2 \)), investigator that corresponds to the one who discovers the critical situation (node \( N3 \)), and a second investigator (node \( N4 \)). These components are connected by edges labelled by the type and characteristics of communication links. This production generates the only non-terminal \( \text{Temp} \) that has to be consumed at the end of the generation path. Productions \( p_4 \) and \( p_5 \) allows considering multiple coordinators for the cooperation context. Production \( p_6 \) generates one additional coordinator in action phase while \( p_6 \) generates a new coordinator whose team is in exploration phase.

Production \( p_4 \) allows generating for a coordinator in action phase, additional investigators. The added investigator, as characterized by this context description, is connected to the coordinator (node \( N1 \)) to send reported data, and to the critical investigator (node \( N2 \)) to receive its observed data.

Production \( p_5 \) eliminates the non-terminal \( \text{Temp} \) to end the generation process and obtain a graph describing the cooperation model that contains exclusively terminal vertices. Based on a given current execution model and on its related context definition (considering the same architectural point of view), consistency of a configuration can be checked. Using our framework, the consi-
tency problem is translated into a graph membership problem. The
membership of the graph describing the application’s current state
in the graph grammar characterizing its context, establishes the
consistency of the execution state.

For instance, the cooperation model described in Fig. 6 can be
proved in conformance with the cooperation context in action
phase. Indeed, a generation path can be found and proves the
membership of the related graph $G_{coop}$ to the graph grammar
$GG_{coop}$. One possible path is given in the following: (1) apply $p_1$
to generate vertices $C$, $I$, $I$, $I_2$ and the non-terminal $Temp$, (2) ap-
ply $p_2$ to generate $C_2$ and $I_5$, (3) apply $p_4$ to generate $I_3$, (4) apply $p_5$
to generate $I_4$, and finally (5) apply $p_6$ to eliminate node $Temp$.

5.3. Transformation models: reconfiguration and refinement

Monitoring services observe, at different levels, the execution
environment of the group communication system. Changes related
to the environment or to the applicative goals have to be consid-
ered and may require adaptation at one or more levels. The first
category of transformation models provided in our framework
rules the architectural adaptation. These models use graph rewriting
rules to automatically transform the architectural reconfigura-
tion. Applied to the current execution model, these rules produce a
new configuration that fits the new environment and goals. Let’s
consider the situation where the monitoring services of the coop-
eration level detect the loss of the critical investigator in a given
section. This situation may, for instance, rise when its energy
source is totally consumed. This investigator is crucial in the treat-
ment of the critical situation. For that, it is necessary to designate
another investigator that will replace it in the critical role.

In order to describe such models, we extend the DPO graph
grammars. We use productions of type $(L,K,R,C)$ where $(L,K,R)$
corresponds to the structure of a graph grammar production of
type DPO, and, where $C$ is a set of connection instructions. The
instructions belonging to $C$ are of type ednC [21]. They are speci-
ﬁed by a system $(n,p,q;d,d’)$ where $n$ corresponds to a node
belonging to the daughter graph $R$, $p$ and $q$ are two edge labels, $d$
is a node label, and $d’$ and $d’$ are elements of the set $(\text{in}, \text{out})$. For
example, a production defined by the system $(L,K,R,(n,p,q;d,d’))$
is applicable to a graph $G$ if it contains an occurrence of the
mother graph $L$. The application of this production involves trans-
forming $G$ by deleting the subgraph (Del = $L \setminus K$) and adding the
subgraph (Add = $R \setminus K$) while the subgraph $K$ remains unchanged.
All dangling edges will be removed. The execution of the con-
nection instruction implies the introduction of an edge between the
node $n$ belonging to the daughter graph $R$ and all nodes $n’$ that are
$p$-neighbours of and $d’$-neighbours. This edge is introduced fol-
lowing the direction indicated by $d’$ and labelled by $q$. Connection
instructions are especially useful for reconfiguration and refinement
models allowing, for instance, the refinement of dependencies from
a given level to another.

The graph rewriting rule presented in Fig. 10 allows trans-
foming the cooperation model by performing the reconfiguration
actions specified to implement architectural adaptability when
the previous situation is detected. The application of this rule re-
quires the existence of at least two other investigators (i.e. verti-
ces $N_2$ and $N_3$ representing two non-critical investigators)
belonging to the same section (the specification of the application
requires that the resulting section contains at least a critical
investigator and a non-critical investigator). The fact that these
three investigators belong to the same section is guaranteed by

The node $N(X_c, "Cont", M_i)$ denotes the consumer associated
with the global controller. $N(X_c, "G", M_i)$ corresponds to a given
channel manager that handles communication between a group of
producers and consumers.

For the investigation teams in the action phase, two specifc sub-
channel managers are defined: the first, $N(X_c, "G", M_i)$, handles
the group communication of the highest priority involving the criti-
cal investigator and its coordinator. The second, $N(X_c, "G", M_i)$,
handles the group communication of the lowest priority involving
non-critical investigators.

The node $N(P, C_{coop}, M_i)$ denotes a data producer of a given
coordinator. It is a Middleware component which is connected to
the controller’s data consumer via a dedicated channel manager.

The node $N(C, C_{coop}, M_i)$ denotes a data consumer of a given
coordinator. As it is required by the application specification, each
coordinator needs two or three data consumers depending on
whether the coordinated investigation team is in exploration or
action phase. In the first case, two consumers are deployed to
process the two kinds of data sent by the investigators (i.e. ob-

---

served data and reported data). In the second case, three consumers are deployed to deal with observed and reported data sent by the critical investigator and also with the data reported by the other investigators’s data producers.

\[ R_{\text{Breakdown}} = (L= \{ N1(X1, "Inv\_a", M1), N2(X2, "Inv\_b", M2), N3(X3, "Inv\_c", M3), \\
N4(X_{\text{coord}}, "Coord", "\text{\$\_2", M4}), N1_{\text{O,QoS\_high},R_{\text{QoS\_low}}}, N4, N2_{\text{R,QoS\_low}} -> N4, \\
N3_{\text{R,QoS\_low}} -> N4, \}; \\
K= \{ N4(X_{\text{coord}}, "Coord", "\text{\$\_2", M4}), N3(X3, "Inv\_c", M3), N3_{\text{R,QoS\_low}} -> N4; \\
R= \{ N5(X1, "Inv\_a", M2), N5_{\text{O,QoS\_high},R_{\text{QoS\_med}}}, N4; \}
\]

\[ i_{\text{c}}=(N5, (X1, "Inv\_a", M5), "O"/"O", \text{out, out}) \]

**Fig. 10.** Rewriting rule for managing critical investigator breakdown at the cooperation level.

\[ p_1 = (L= \{ N1(X1, "Cont", M1), N2(X2, "Cont", M2), N3(X3, "Cont", M3), N4(X4, "Cont", M4), N5(X5, "Cont", M5), N6(X6, "Cont", M6), N7(X7, "Cont", M7), N8(X8, "Cont", M8), N9(X9, "Cont", M9); \\
K= \{ \}; R= \{ \}; C= \{ \}; \]

\[ c_1 = (N2(X2, "P\_coord", M2), ("R", "X", X), \text{in, in}) \]

\[ p_2 = (L= \{ N1(X1, "P\_coord", M1), N2(X2, "P\_coord", M2), N3(X3, "P\_coord", M3), N4(X4, "P\_coord", M4), N5(X5, "P\_coord", M5), N6(X6, "P\_coord", M6), N7(X7, "P\_coord", M7), N8(X8, "P\_coord", M8), N9(X9, "P\_coord", M9); \\
K= \{ \}; R= \{ \}; C= \{ \}; \]

\[ c_1 = (N3(X3, "P\_rev", M3), ("R", "X", X), \text{in, in}) \]

**Fig. 11.** Grammar productions for the refinement model between the Middleware and the cooperation models.

\[ p_3 = (L= \{ N1(X1, "P\_coord", M1), N2(X2, "P\_coord", M2), N3(X3, "P\_coord", M3), N4(X4, "P\_coord", M4), N5(X5, "P\_coord", M5), N6(X6, "P\_coord", M6), N7(X7, "P\_coord", M7), N8(X8, "P\_coord", M8), N9(X9, "P\_coord", M9); \\
K= \{ \}; R= \{ \}; C= \{ \}; \]

\[ c_1 = (N7(X7, "P\_rev", M7), ("R", "X", X), \text{in, in}) \]

\[ p_4 = (L= \{ N1(X1, "Coord", "\text{\$_2", M1), N2(X2, "Coord", M2), N3(X3, "Coord", M3), N4(X4, "Coord", M4), N5(X5, "Coord", M5), N6(X6, "Coord", M6), N7(X7, "Coord", M7), N8(X8, "Coord", M8), N9(X9, "Coord", M9); \\
K= \{ \}; R= \{ \}; C= \{ \}; \]

\[ c_1 = (N7(X7, "P\_rev", M7), ("R", "X", X), \text{in, in}) \]

The node \( N(P_i, P_{inv}, M_i) \) denotes a data producer of a non-priority investigator. It sends via a channel manager data to the consumers of the coordinator in charge of the investigation team. The node \( N(P_i, P_{inv,x}, M_i) \) denotes the data producer associated with a critical
investigator. Three producers are deployed for each investigator of this kind to send observed and reported data to the coordinator and to send observed data to the other investigators of its team. $N(C_i, C_{inv}, M_i)$ this kind of data consumers is only deployed for non-critical investigators of a team in the action phase. They are dedicated to the processing of observed data sent by the critical investigator.

As an inter-level coherency verification example, this refinement model allows verifying that the cooperation model of Fig. 6 is in conformance with the Middleware model presented in Fig. 7. This is provided by the existence of a generation path that generates the graph of the cooperation model using the previous refinement model while considering the Middleware model as the axiom of the graph grammar. One possible generation path is obtained by applying the sequence of productions: p1 generating the cooperative role Controller C; p3 generating coordinator C2; p2 generating coordinator C1 and critical investigator I1; two times p4 generating investigators I2 and I3; and finally two times p5 generating investigators I4 and I5.

6. Conclusion and future work

The work exposed in this article deals with the need of including new QoS-oriented self-adaptation properties in the future communication systems. The goal is to tackle dynamic QoS requirements coming from complex cooperative activities (such as CMS) while still taking into account the variability of the network environment resources (typically wireless and mobile), over which they are distributed. This kind of application and network context constitutes the premises of a future ambient Internet, in which the communication resources will be accessible everywhere, authorizing the spontaneous deployment of potentially complex communicating system whose requirements and constraints will be both variables and not entirely defined a priori.

In this context, our work has targeted the need in architectural self-adaptation properties at the end-to-end protocol levels (i.e. Middleware and Transport), with the strong goal of formally ensuring the coherency of the architectural composition/adaptation choices both within each level and between the two levels. The Middleware level is involved in the management of cooperation aspects while communication issues are managed at the Transport level. It is one of our perspectives to consider the generalisation of our approach by applying it to lower protocols levels, typically network and data link levels.

Toward this challenging goal, several contributions have been provided in this article. The interest of managing architectural self-adaptation properties at Middleware and Transport levels has been motivated using a typical example of CMS scenario. For this purpose, some examples of architectural compositions and adaptations implementing the cooperation and communication services at Middleware and Transport levels have been given.

On those bases, different contextual and deployment graph-based models have been exhibited to help automating the adaptive provisioning of cooperation and communication services through their composing entities at the Middleware and Transport levels. To ensure the coherency of architectural composition/adaptation choices both within and between the two levels, those models and their dependency relationship have been identified and formalized using graph grammars, which have been applied on the CMS example scenario. The set of resulting graph grammar-based models constitutes the main part of a framework that sustains the adaptation process by providing the allowed (i.e. coherent) architectural transformations within a given level and architectural refinement from a level to another. The final decision can be supported by analytical models such as the ones initiated in [2] and [6] for the Middleware and the Transport levels, respectively.

The assessment of the approach has also been subject to theoretical studies and practical implementations and experiments, and different application contexts. Simulation models and tools have been implemented. The automated transformation rules associating the cooperation model, the Middleware deployment model and the Transport connection model have been fully developed. The automated transformation between the connection model and the Transport deployment model is under development.

References


