Software architecture for self-adapting sub-sea sensor networks

Work in progress

Svein Hallsteinsen and Richard Torbjørn Sanders
SINTEF
Trondheim, Norway
e-mail: {svein.hallsteinsen; richard.sanders}@sintef.no

Abstract— **Monitoring of the sub-sea environment requires advanced sensor networks with both stationary and mobile nodes on the surface and underwater, each node playing a different role. The combination of mobile and stationary nodes, the loss of nodes due to harsh conditions, the difficult conditions for wireless sub-sea communication and the need for nodes to collaborate are all factors that require new approaches to software architectures for such nodes and systems. This articles presents work in progress that sets out to address these issues by adapting approaches from software engineering solutions to ubiquitous computing.**

Keywords- software architecture;underwater sensor networks, ubiquitous computing

I. Introduction

We witness an increasing understanding of the importance of the oceans for the well-being of our society, in terms of food production, transport, off-shore petroleum production and other sub-sea natural resources, as well as the ocean's capability to bind CO_2 and other climate gasses, and as an important element in climate influence. This has resulted in increasing interest in research and development concerned with monitoring of the oceans and the development and deployment of underwater sensor.

In particular the vulnerable areas in the north are important, with rich maritime life being influenced by shipping, petroleum activities, global climate change and other environmental threats. The Norwegian government has recently launched an initiative involving key players within research and industry to tackle the issues at hand [1].

One challenge is to design and deploy systems of nodes that are capable of collaborating to monitor environment data [2], and to adapt to varying conditions and collaboration structures. A promising approach seems to be to adapt software engineering techniques originally created for mobile systems and ubiquitous computing, while taking into account the special requirements posed by the sub-sea environment.

This article presents work in progress. The ProSUS project at SINTEF will research on software solutions on the application level for underwater sensor networks in the sea that enable adaptive and collaborative behavior.

Such systems must cope with nodes that appear and/or disappear, either because they move in or out of communication range, or due to nodes being destroyed or new nodes being deployed. Furthermore there will be limitations and variation in available battery power, as well as poor and often variable communication capacity between nodes [2]. Access to nodes to maintain or upgrade hardware and software is difficult and/or costly, and the large number of nodes involved adds to the challenge. Single nodes or groups of nodes must be able to work for sustained periods of time without contact with other systems, and exploit periods with external communication contact to exchange data efficiently. The adaptation must enable ad-hoc service collaboration between nodes, where some nodes are stationary while others are mobile, and where nodes take over tasks from nodes that disappear or fail – what we can call the transfer of roles.

II. Approach

There exist solutions to ad-hoc communication and adaptation in sensor networks [3] [4], but these do not support collaborative services and transfer of roles as explained above. However, similar challenges have been addressed within other application areas, such as ubiquitous computing, where mobile terminals roam in environments embedding numerous networked computing elements, and industrial robotics, where nodes collaborate to solve complex tasks.

SINTEF has been involved in the development of technologies for mobile services and ubiquitous computing in several international projects [5], [6], [7]. These solutions are based on self-adaptive nodes involved in dynamic collaboration relationships through dynamic service discovery and negotiation of quality of service. Nodes may adapt their internal configurations to suit the environment, based on reflection. This contributes to more robust systems. The task at hand is to investigate if such solutions are applicable to underwater sensor networks.

In terms of scientific approach, the project is based on design science [8]. It involves case studies where we apply solutions for collaborative and self-adapting systems from earlier work within ubiquitous computing on typical scenarios for future underwater sensor networks, and investigate how they can be adapted for use in this setting. Since wireless underwater communication is still in its infancy, and the resources of the project are limited, we will rely on developing software prototypes which we will test in simulated and controlled environments, with variation of parameters concerning e.g. communication and power in line with the scenarios. Through systematic observation and analysis we will evaluate the applicability of the suggested solutions, and suggest further adaptation.

We plan to exploit ns-2 [9] to predict network parameters in accordance to the chosen scenarios. As a communication framework we plan to use and extend the open source middleware platform ActorFrame [10]. ActorFrame was used in SIMS [7], and supports addressing between nodes, dynamic role binding and role negotiation. This choice facilitates the development of prototypes, in particular for supporting role transfer.

III. SELF-ADAPTATION MIDDLEWARE

As already mentioned, a main goal of our work is to investigate the applicability of the MUSIC self-adaptation technology to underwater sensor networks. MUSIC is an open source technology including methods, modeling notations, tools and middleware supporting the development of self-adapting distributed systems. It is being developed by the MUSIC project [5]; prototype tools and middleware and several trial application are available. MUSIC is based on an externalized approach to self-adaptation where the adaptation logic is delegated to generic middleware. The middleware exploits knowledge about the composition and quality of service (QoS) characteristics of its constituting components and services used [11], [12], [13]. The middleware monitors relevant context parameters, including 3rd party services available in the environment, and adapts the active applications by reconfiguring internal components and binding to 3rd party services when changes occur. An overview of the MUSIC platform is shown in Fig. 1.

The knowledge about the system being adapted is provided by the developer in the form of a *QoS-aware model* (cf. Fig 2), which describes the abstract composition, the relevant QoS dimensions and how they are affected when varying the actual component configuration. This model is exploited at runtime by the adaptation middleware to select, connect, and deploy a configuration of Component Realizations or external services providing the *best utility*. The utility measures the degree of fulfillment of user preferences while optimizing device resource utilization.

The model describes the abstract composition as a set of Roles collaborating through Ports, which represent services either provided to or required from collaborating components. Properties and property predictor functions associated with the ports define how the QoS properties and resource needs of components are influenced by the QoS properties of the components they depend on. A port has a Type defining the service represented by the port in terms of interfaces and protocol. Component realizations and external services implement ports and a component realization can be used in a role if the ports match (same type). Component realizations are Atomic or Composite. A Composite Realization is itself an abstract composition and thus allows for recursive decomposition. Constraints are predicates over the properties of the constituting components of a composition which restrict the possible combinations of component realizations (*e.g.*, configuration consistencies).

The model is represented at runtime as *plans* within the middleware. A plan reflects a component realization or an available external service and describes its ports and associated property predictors as well as *implicit dependencies* on the hosting platform (*e.g.*, platform type and version). In the case of an atomic component realization, it contains a reference to the class which realizes the component. In the case of an external service it contains a reference to the service description. In the case of a composite realization, the plan describes the internal structure in terms of roles and ports and the connections between them. Variation is obtained by describing or discovering possible alternative realizations of the roles.

Planning refers to the process of selecting the realizations which make up the application configuration providing the best possible utility to the end-user. This process is triggered at start-up of the application and at run-time when the middleware discovers relevant context changes. The planning middleware iterates over all possible bindings of the roles, computes the Predicted Properties and the utility [13]. The utility function of an application is provided by the developer and is typically expressed as a weighted sum of dimensional utilities where the weights express user preferences (*i.e.*, relative importance of a dimension to the user). A dimensional utility measures user satisfaction in one property dimension based on the predicted property in the given context.

The middleware then uses the selected set of plans to reconfigure the application. This requires the collaboration of the realizations, which must implement a reconfiguration interface allowing the middleware to bring them to a state where they can be safely replaced and transfer their state to an alternative realization. When binding to an external service, a SLA (service level agreement) is established with the provider, expressed in terms of agreed QoS properties. It is the responsibility of the provider to alert the consumer if the provided service level deviates significantly from the agreed one. This will trigger re-planning in the consumer, which may

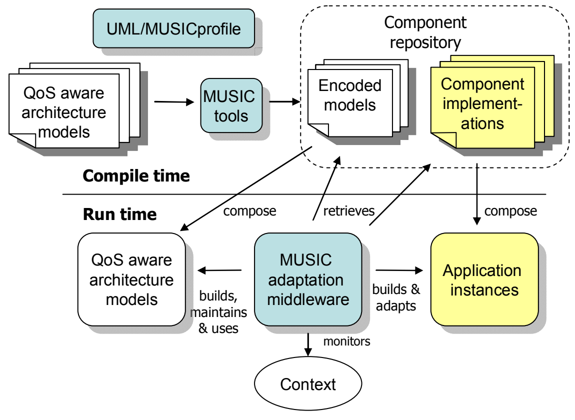

Figure 1. MUSIC platform overview

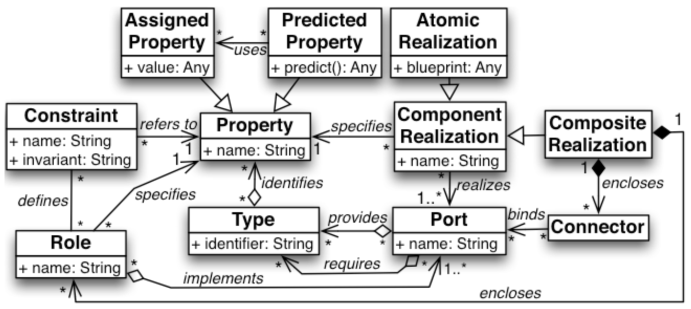

Figure 2. Metamodel for MUSIC QoS aware architecture models

result in rebinding to an alternative provider or internal reconfiguration in the consumer to adapt to the change in provided QoS. The middleware manages a collection of active applications and seeks to maximize the overall utility which is computed as a weighted sum of individual application utilities. The weights in this case express application priorities of the user.

As an illustration of how the MUSIC technology works we present an example application and a walk-through of a scenario explaining how the middleware acts in response to various context changes. The scenario is about a metro trip in a large city assisted by a MUSIC-enabled mobile device with support for WiFi, UPnP, and GPS. The MUSIC based travel assistant application assists with route planning and navigation outside and inside stations, notifies about train changes, and detects delays and notifies the traveler if he is affected.

The QoS aware architecture model for the travel assistant is shown in Fig. 3. It is described as a composition with the roles GUI, Main and GPS. GUI presents a graphical user interface on the device, Main embeds the application logic and binds the different functionalities together. Main depends on external services route to find the shortest route and the estimated travel time, map to get localized maps and loc to get the current location. Alternatively the location service can be provided by a local component using the onboard GPS unit. The QoS properties used in the model are specified in Table 1. Property predictors for the application, specified as functions of the properties of the components it consists of, are associated with the composition in Fig 3. The utility function assumes that the user always prefers high accuracy, low battery consumption and low cost, while the relative weighting (w_acc, w_bat, w_cost) is extracted from the user profile by the middleware. Service providers and associated property predictors assumed in the scenario are shown in Table II.

Scene 1. The traveler enters the metro station and launches the travel assistant application to plan the trip. In the station the metro company operates a WLAN and offers a location service, a route service for public transportation and a map service of the city at two QoS levels: *basic* and *premium* quality. Via UMTS, there is also access to a commercial service of high quality, though at a higher monetary price. In this situation the configuration using the RATP loc, detailed map, and high quality route services predicts the highest utility and the adaptation middleware instantiates this configuration.

Scene 2. During the trip, there is an incident in the metro, blocking the planned itinerary. The travel assistant, which monitors traffic information through the route service, discovers this, notifies the traveler and proposes an alternative itinerary with a different final station. Meanwhile RATP reserves a large share of its bandwidth and computing infrastructure to assist the emergency personnel and declines to offer the highest QoS level for the map service. This is detected as an SLA violation by the adaptation middleware, which thus triggers a re-planning. Now using the commercial Map service instead yields the highest utility and the middleware reconfigures the application's service binding accordingly.

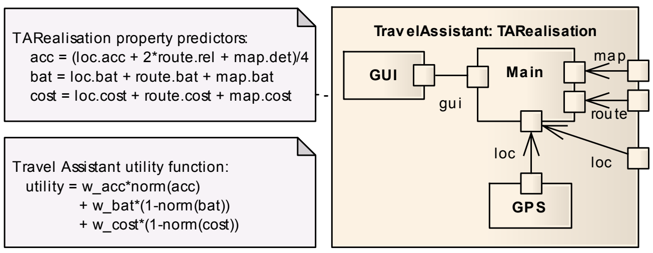
TARealisation property predictors:
acc = (loc.acc + 2*route.rel + map.det)/4
bat = loc.bat + route.bat + map.bat
cost = loc.cost + route.cost + map.cost
Travel Assistant utility function:
utility = w_acc*norm(acc)
+ w_bat*(1-norm(bat))
+ w_cost*(1-norm(cost))
TravelAssistant: TARealisation
GUI
Main
gui
map
route
loc
loc
GPS

Figure 3. Architecture model for the travel assistant

Scene 3. The GPS signals do not reach the metro stations and tunnels. Therefore the GPS component has predicted 0 accuracy and has not been chosen. However, after leaving the metro at the destination station, the GPS module starts working. This is detected by the middleware and triggers re-planning. As this service provided by a local component is free and accurate, the adaptation middleware predicts its use to have the highest utility and reconfigures the application accordingly.

IV. APPLICATION IN SENSOR NETWORKS

We believe that the technology described above will be well suited to deal with the challenges typical of under water sensor networks which have to survive for long periods of time without manual intervention in a highly dynamic and harsh environment. Obviously sub-sea sensor networks represent a

TABLE I. RELEVANT QOS PROPERTIES FOR THE *TRAVEL ASSISTANT* APPLICATION

Property	Description	Value range
acc	Accuracy	1-10
det	Level of detail of map	1-10
rel	Reliability of estimated travel time	1-10
cost	Monetary cost of using a service	0-∞
bat	Power consumption of a component or link	0-∞

TABLE II. SERVICE LANDSCAPE

Service	Description	Provider	Level	Property predictors
loc	Locates the device geographically	RATP		cost=0, acc=5, bat=1
		Local component using the builtin GPS		cost=0, acc=7 if GPS signal 0 otherwise, bat=3
map	Provides a map of a limited area	RATP	basic	cost=0, det=1, bat=2
		RATP	detailed	cost=5, det=9, bat=4
		3rd party		cost=9, det=9, bat=4
route	Computes best route and estimated travel time	RATP	basic	cost=0, rel=1, bat=1
		RATP	reliable	cost=5, rel=7, bat=1

resource constrained computing environment, which require attention both to limitations of the computing power and constraints on energy consumption. However, handheld devices which has been the target platform for the MUSIC middleware, also have this characteristic. Therefore we think that this technology is a good starting point for our project.

The context monitoring and service discovery mechanism of the MUSIC middleware enables each node to be aware of other nodes within communication reach, to know the services they provide and the offered QoS as well as the capacities and resource demands of the available communication links. In addition to the service and communication landscape, the internal computing and energy resources can also be monitored by the context monitoring mechanism and taken into account in the adaptation. In particular adapting to variation in the availability of energy we expect to be very important.

The bandwidth and latency characteristics of underwater communication based on acoustic signals are several decades away from what we are used to in radio based communication in air. Therefore the discovery and communication protocols supported by MUSIC (SLP, UPnP and Http) may not be applicable. However, the MUSIC architecture is designed to allow easy plug-in of different technologies in this area.

By providing models of the software in the nodes, as required by the self-adaptation mechanism of the middleware, we will enable the dynamic and intelligent selection of nodes to collaborate with, the adaptation of the internal configuration of the software and optimization of the use of internal resources. In sensor networks, the utility function will not express the satisfaction of a human user, as was the case in the travel assistant example, but rather policies for the behavior of the system.

In the example, we focused on optimizing the behavior of one node. In sub-sea sensor networks the emphasis will be more on optimizing the collective behavior of a dynamically varying collection of nodes. This requires that the utility function takes into account goals both at the level of the individual nodes and at the level of the entire network. At the individual node level we will use the middleware more or less as explained in the example in the previous section, where the utility function expresses the goals of the node. At the inter node level we will exploit service discovery and service level awareness and extend the utility function to take into account higher level goals for the behavior of the entire network. Parameters conveying policies at the inter-node level can be distributed either as special property dimensions of services or through context distribution.

There is also the possibility to create more tightly coupled groups of nodes managed by one adaptation authority. In this case the adaptation middleware also selects a suitable deployment of the components on the nodes in the group. This can be relevant when a group of resource poor sensors are supported by a more resource rich mother node.

With these approaches we can achieve the coordinated adaptation of a collection of autonomous nodes or node groups. This approach to achieving the coordinated goal driven behavior of a collection of autonomous nodes has already been investigated in the area of multi-user mobile applications [15].

V. Conclusion and Outlook

In this paper we have outlined an experiment with applying service oriented adaptation middleware initially developed for ubiquitous computing to underwater sensor networks. The experiment will be based on developing and testing prototype software on typical scenarios in simulated environments.

At this early stage of the work, we cannot provide any conclusions. However, the initial analysis of the capabilities of the selected technologies compared with the challenges in underwater sensor networks looks promising. The next steps are to develop scenarios in cooperation with domain experts responsible for deployment and operation of such networks, implement the necessary sensor network software, simulate the environment to support them, and analyze the resulting behavior.

References

[1] Jens M. Hovem, “NNN- New Nerve Network for Northern Waters”, SENSORCOMM 2008

[2] I. F. Akyildiz, D. Pompili, T. Melodia, "Underwater Acoustic Sensor Networks: Research Challenges," Elsevier's Journal of Ad Hoc Networks, March 2005, Vol. 3, Issue 3, pp. 257-279

[3] Grace, P., Coulson, G., Blair, G., Porter, B., Hughes, D., "Dynamic Reconfiguration in Sensor Middleware", In Proceedings of the 1st International Workshop on Middleware for Sensor Networks (MidSens '06), co-located with Middleware 2006, Melbourne, Australia, November 2006

[4] D. McIntyre et al., The Low Power Enegy Aware Processing (LEAP) Embedded Networked Sensro System, In Proceedings of the 5th international conference on Information processing in sensor networks, ACM New York 2006, pp. 449-457

[5] IST FP6 035166 MUSIC – Self-Adapting Applications for Mobile Users in Ubiquitous Environments, www.ist-music.eu

[6] IST FP6 027055 MIDAS – Mobile Platform for Developing and Deploying Advanced Mobile Services, www.ist-midas.org

[7] IST FP6 027610 SIMS – Semantic interfaces for mobile services, www.ist-sims.org

[8] A. H. Hevner et al., Design Science in Information Systems Research, MIS Qarterly Vol. 28 No. 1, pp. 75-105/March 2004

[9] The Network Simulator – ns-2, www.isi.edu/nsnam/ns/

[10] ActorFrame, www.tellu.no

[11] R. Rouvoy, et al. *Composing Components and Services using a Planning-based Adaptation Middleware*. In 7th Int. Symp. on Software Composition (SC), LNCS 4954, Springer, 2008.

[12] G. Brataas, et al. *Scalability of Decision Models for Dynamic Product Lines*. In Int. Work. on Dynamic Software Product Line (DSPL). 2007.

[13] J. Floch, et al. Using Architecture Models for Runtime Adaptability. IEEE Software, 23(2), 2006.

[14] M. U. Khan, R. Reichle, and K. Geihs. Architectural Constraints in the Model-Driven Development of Self-Adaptive Applications. IEEE Distributed Systems Online 9(7), 2008.

[15] L. Fraga, S. Hallsteinsen, and U. Scholz. InstantSocial – Implementing a Distributed Mobile Multi-user Application with Adaptation Middleware. EASST Communications 11, 2008.