Supporting Mobility using Context-Based Reasoning

Aline Senart, Mélanie Bouroche, Neil O'Connor,
Barbara Hughes, Kulpreet Singh and Vinny Cahill

Distributed Systems Group, Department of Computer Science,
Trinity College Dublin, Ireland,
`first.last@cs.tcd.ie`

Abstract. Recent research in mobile computing has given rise to a new class of ubiquitous applications that are deployed in dynamic networks where communication is not reliable. In this paper, we describe how context-based reasoning can be used to overcome this limitation of (ad hoc) wireless communication. We present a small set of abstractions that facilitate the development of mobile ubiquitous applications centred on the definition of contexts of interest. Mobile intelligent agents, called *sentient objects*, extract, interpret and use context information to drive their behaviour. Communication between sentient objects is supported by an event-based middleware that provides feedback when communication cannot be maintained, allowing sentient objects to dynamically adapt their behaviour. We show how our approach has been successfully applied to an application from the transportation domain.

1 Introduction

Mobile ubiquitous applications, such as music and file exchange or autonomous mobile cars, are challenged by the dynamism of today's wireless networks, especially with the advent of protocols supporting ad hoc communication. Ad hoc wireless networks comprise sets of mobile nodes connected by wireless links that form arbitrary wireless network topologies without the use of any infrastructure. As nodes move in and out of range of each other, the connectivity and network topology changes dynamically. The rate of link failure due to mobility and variations in signal strength often result in network partitions [GC04]. Providing support for mobile ubiquitous applications that can operate reliably in this environment is therefore very difficult.

Previous attempts at providing such support [CK02,RC03] defined abstractions for the collection, aggregation and dissemination of higher-level context data by a set of central access points. These approaches do no map well to large-scale applications in mobile ad hoc networks. Tuple-based systems that have been developed for ad hoc networks [MRar,JR06] do not provide any abstractions for handling link failures or network partitions. Finally, mobile computing middleware exploiting the principle of reflection to enhance the construction of adaptive and context-aware mobile applications [CEM03] do not address reliability constraints in mobile settings.

In this paper, we present a novel approach to addressing the impediments to building ubiquitous applications arising from the use of (ad hoc) wireless networks through the definition of contexts of interests. We define a small set of commonly-required programming abstractions that facilitate the development of ubiquitous applications for deployment in (ad hoc) mobile environments. These abstractions build on ***sentient objects***, mobile intelligent software agents that extract, interpret and use context information obtained from sensors, other sentient objects and their underlying infrastructure to drive their behaviour. Communication between sentient objects is supported by an event-based communication middleware that has been designed for mobile applications in wireless ad hoc networks. Depending on the resources available from the underlying network, communication may not be reliable. Therefore, our approach is to provide contextual feedback to sentient objects about the current state of communication, so that they can adapt their behaviour accordingly while respecting time bounds.

To demonstrate the use of the programming abstractions and the feasability of our approach, we have developed an application scenario from the Intelligent Transportation System (ITS) area, that enables autonomous mobile cars and pedestrian traffic lights to coordinate their context-driven behaviour through wireless communication. This example shows how the feedback from the environment can be used within contexts of interest.

The remainder of this paper is structured as follows. In Section 2, we describe how mobility is supported in ubiquitous applications using context-based reasoning. Then, in Section 3, we describe and discuss the use of the programming abstractions in the scenario. Finally, Section 4 presents our conclusions.

2 Integrating Mobility into Contexts

In this section, we describe how we overcome the impact of the characteristics of wireless networks by taking them into account as contextual information, and describe the programming abstractions that support this approach.

2.1 Supporting Context-based Reasoning

A sentient object may receive input from a variety of sources that needs to be integrated before being used in determining its context. The fusion process relies on a set of pipelines, comprising a combination of components that extract and combine ***readings*** (i.e., raw values) from the received events, resulting in higher-level information, in the form of one or more ***facts*** (c.f. Figure 1).

The contexts in which a sentient object can be during its lifetime are organised as a context graph, to which only a subset of contexts can be transitioned from the current context. Context filtering is used to reduce the complexity of the system by restricting the set of abstractions to be considered (c.f. Figure 2).

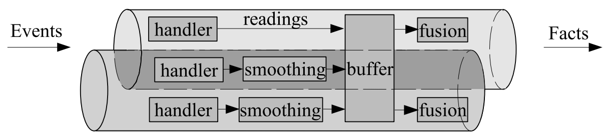

Fig. 1. Example of two pipelines

When in a context, only one pipeline is active thereby constraining the set of events being processed and the facts to be asserted.

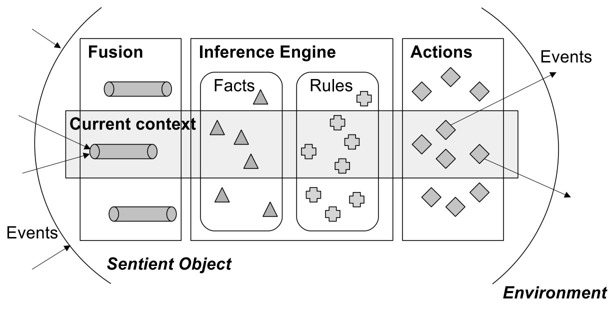

Fig. 2. The MoCoA architecture

The action selection decision within a context is based on ***rules***. Rules may be predefined as applicable within specific contexts. They take the form of `condition/actions`, where a condition is expressed in terms of facts. As the number of facts is limited within a context, only a subset of the rules needs to be evaluated in the current context, restricting the number of actions potentially available.

Since the set of actions available is typically fixed (e.g., constrained by the available actuators), rules may also be inferred. Depending on the feedback returned from the environment, the long term reward of choosing these actions again given the current context can be learned [DCCC05]. As a result, optimal rules are learned over time.

2.2 Supporting Mobility

In the kinds of complex and large-scale mobile ubiquitous systems that we are considering, where changes in the quality of communication are frequent, new programming abstractions are required.

Filters have typically been applied to the subject or content of events to ensure that they are only propagated to consumers that have expressed an interest in them [MFB02]. In addition to these filters, we support proximity-based filtering which is used to define geographical areas, ***proximities***, within which events are valid. Such filtering confines propagation of events to a geographical area surrounding a sentient object since the closer event consumers are located to a producer the more likely they are to be interested in the events that it produces. This proximity can be stationary (e.g., defined absolutely with GPS coordinates) or mobile, i.e., relative to a mobile sentient object.

To tolerate link failures and network partitions that occur in (ad hoc) networks, we provide feedback to sentient objects about the state of communication. The proximity in which a sentient object wants its events to be propagated is termed ***desired coverage (DC)***. Depending on the network connectivity and the available resources, it might not be possible over some period of time to deliver events in the DC. The proximity in which delivery of events is provided is called the ***actual coverage (AC)*** and its size changes over time depending on the available connectivity.

Therefore, when the AC becomes smaller than the DC, because there is a change, like a disconnection or a significant decrease in event-delivery latency that would prevent required communication to be achieved, the underlying infrastructure notifies the sentient object by means of an adaptation event (an example of an event raised by the infrastructure). This event is processed through some pipeline(s) as any other event and participates in context recognition, potentially triggering a transition to a new context. The sentient object, informed about the revised proximity in which communication is guaranteed, is then free to dynamically adapt its behaviour (e.g., by adopting a fail-safe behaviour) in order to respect the safety requirements of the application[1].

By reconciliating mobility and context-based reasoning, event-based communication in adaptable proximities with feedback represents a powerful abstraction for large-scale ubiquitous applications deployed in (ad hoc) wireless networks that provides well-defined guarantees allowing stringent safety constraints to be enforced.

2.3 Implementation

We offer, through a high-level API in Java, the possibility for software architects to easily design sentient objects by assembling basic components. For each of the previously defined abstractions, we have a set of implementations available in a library. We also provide a tool that is able to generate the executable C++ code of the sentient object, by selecting the appropriate components from the design specifications. To further assist the construction of sentient objects, we are currently extending the tool with a graphical interface.

In our library of components, different instantiations of the STEAM event service [MC03] are available: STEAM for use in wireless ad hoc networks, and RT-STEAM for soft and hard real-time in ad hoc environments. RT-STEAM relies on further components providing real-time resource reservation and routing in ad hoc networks, as well as time-bounded media access control [HC03,CC02].

[1] The adaptation notification can be timely.

Developers can also find in the library different implementations of fusion (e.g., sum, average and Bayesian Network) and different inference engines (e.g., based on first order logic rules or collaborative reinforcement learning [DCCC05]).

3 Modeling the Application Scenario

We have designed mobile sentient cars that navigate autonomously towards some destination, while obeying pedestrian traffic lights they encounter en route. Periodically, the traffic lights send events to cars containing the current time, their position and colour, and the time of their next planned colour change. Such events allow sentient cars to discover traffic lights dynamically and decide whether to brake depending on what colour the light will be when they will pass it. In this scenario, we focus on traffic lights and more specially on what happens when communication with mobile cars is not ensured.

Pedestrians traffic lights are modeled as sentient objects that receive two kind of events: `button` events emitted by their button sensor and `adaptation` events raised by their underlying infrastructure when their AC is smaller than their DC. These events are processed in two different pipelines. The first pipeline can assert `buttonPressed` fact when appropriate, and derive the next planned change. As for the second pipeline, it extracts the value of the AC from `adaptation` events.

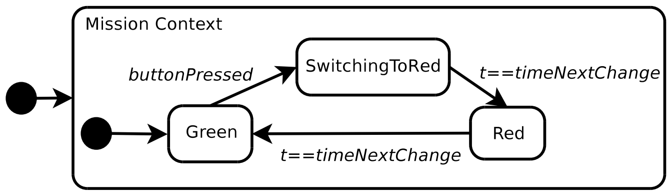

Fig. 3. Context graph of traffic lights

To allow mobile cars to stop in time at a red light, traffic lights need to warn incoming cars early enough - we call the minimum time required to warn cars before switching to red Δ. Therefore, traffic lights need to communicate with cars in a proximity that is big enough so that cars will have time to receive events and stop. As the cars approach traffic lights without being announced, traffic lights assume the worse case, i.e., the presence of cars, and will adapt their behaviour each time the AC is smaller than the size of this proximity. When communication is down, the fail-safe behaviour for traffic lights is to stay green since incoming cars are not aware of the presence of the traffic lights and will not stop. In [BHC06], we have shown that the size of this proximity depends on the time to become present in the proximity, the period of traffic lights' events, the braking distance of cars, and finally the maximum time required to notify traffic lights of the AC and the maximum time required by traffic lights to react.

Figure 3 presents the context graph of traffic lights in UML notation. Traffic lights are either green or red, or switching from green to red.

To allow traffic lights to periodically send events to cars, the following rule has been defined within the mission context (which is inherited by the three subcontexts):

```
every T / raise trafficLight(time, location, status, timeNextChange);
assert messageSent(time)
```

MessageSent is a fact kept in the knowledge base of the traffic lights to remember at what time the last event was sent to cars. In the `Green` context, the transition to SwitchingToRed is triggered by the following rule:

```
buttonPressed / assert timeNextChange(time + T + Δ)); switchTo(Red)
```

If the communication is not good enough, i.e., that the AC is under some critical threshold (called CC), traffic lights need to adopt a fail-safe behaviour by switching to green if the light was red, or postponing their change otherwise. Therefore, in the `Red` context, we have the following rule:

```
(∃ messageSent ms, time == ms.time + stabilityPeriod) && (AC < CC) /
assert timeNextChange(time)
```

and in the `SwitchingToRed` context:

```
(∃ messageSent ms, time == ms.time + stabilityPeriod) && (AC < CC) /
assert timeNextChange(time + T + Δ))
```

where the stability period includes the events' latency and the worst case adaptation notification time.

This scenario demonstrates the ease of use and applicability of the abstractions that we have defined to support mobile ubiquitous applications in wireless networks. Context-based reasoning has been successfully employed to overcome mobility concerns, such as network partitions and link failures.

4 Conclusion

This paper presents some simple programming abstractions that facilitate ubiquitous application development in (ad hoc) mobile networks. When communication within a desired proximity cannot be maintained, sentient objects are notified. They are then free to dynamically adapt their behaviour to respect application-specific safety constraints if the actual coverage reaches a critical threshold. Our approach has been successfully applied to an ITS application.

References

[BHC06] M. Bouroche, B. Hughes, and V. Cahill. Building reliable mobile applications with space-elastic adaptation. In *International Workshop on Mobile Distributed Computing*, Niagara Falls, New York, USA, June 2006.

[CC02] R. Cunningham and V. Cahill. Time bounded medium access control for ad hoc networks. In *Workshop on Principles of Mobile Computing*, Toulouse, France, October 2002.

[CEM03] L. Capra, W. Emmerich, and C. Mascolo. Carisma: Context-aware reflective middleware system for mobile applications. In *IEEE Transactions on Software Engineering*, number 29(10), pages 929–945, October 2003.

[CK02] G. Chen and D. Kotz. Solar: An open platform for context-aware mobile applications. In *International Conference on Pervasive Computing*, pages 41–47, Zurich, Switzerland, June 2002.

[DCCC05] J. Dowling, E. Curran, R. Cunningham, and V. Cahill. Using feedback in collaborative reinforcement learning to adaptively optimise manet routing. *IEEE Transactions on Systems, Man and Cybernetics*, 35(3):360–372, 2005.

[GC04] G. Gaertner and V. Cahill. Understanding link quality in 802.11 mobile ad hoc networks. *IEEE Internet Computing*, 8(1):55–60, 2004.

[HC03] B. Hughes and V. Cahill. Achieving real-time guarantees in mobile wireless ad hoc networks. In *Real-Time Systems Symposium*, pages 37–40, Cancun, Mexico, 2003.

[JR06] C. Julien and G.-C. Roman. Egospaces: Facilitating rapid development of context-aware mobile applications. *IEEE Transactions on Software Engineering*, 32(5):281–298, May 2006.

[MC03] R. Meier and V. Cahill. Exploiting proximity in event-based middleware for collaborative mobile applications. In *4th IFIP International Conference on Distributed Applications and Interoperable Systems*, pages 285–296, Paris, France, November 2003. Springer-Verlag.

[MFB02] G. Mühl, L. Fiege, and A. Buchmann. Filter similarities in content-based publish/subscribe systems. In *International Conference on Architecture of Computing Systems*, volume 2299, pages 224–238, 2002.

[MRar] A. L. Murphy and G.-C. Roman. Lime: A coordination middleware supporting mobility of hosts and agents. *ACM Transactions on Software Engineering and Methodology*, To appear.

[RC03] M. Roman and R. H. Campbell. A middleware-based application framework for active space applications. In *International Middleware Conference*, Rio de Janeiro, Brazil, June 2003.