Designing for palpability: Workshop at Pervasive 2007, 13-16 May 2007, Toronto, Canada. http://www.ist-palcom.org/palpable-pervasive-2007

A Seamless Hybrid Communication System for Transient Locations

Roberto Ghizzioli, Giovanni Rimassa, and Dominic Greenwood

Whitestein Technologies AG,
Pestalozzistrasse 24, 8032 Zürich, Switzerland
{rgh,gri,dgr}@whitestein.com
http://www.whitestein.com

Abstract. This paper presents the RASCAL System, a middleware component able to palpably ensure that user-level services communicating across infrastructure or ad-hoc networks continue to work even when deployed in disruptive environments such as major incidents sites. By palpable we mean that users should be able to notice, comprehend communication actions and - where necessary - negotiate different levels of user control. This work highlights the features of the system, the end-user interaction and an evaluation scenario.

Key words: pervasive computing, hybrid networks, autonomic communication

1 Introduction

Mobility is now a central aspect of everyday life with mobile users expecting to be always-best-connected in whatever location they are and in whatever task they are performing. In the majority of cases this implies two things: (1) they expect anywhere and anytime access with the maximum capacity on offer, and (2) they expect their information to get through (both sent and received) at all times. Under normal circumstances both of these requirements are relatively easy to meet, but each becomes more difficult (especially the second aspect) when users move through, or are located in, environments that are disruptive, resulting in rapidly changing coverage of different network technologies. This is particularly problematic in emergency scenarios when reliable communication between emergency workers at major incident sites is vital.

The *Resilience and Adaptivity System for Connectivity over Ad-hoc Links* (RASCAL) was designed to deal with such disruption by maximizing the opportunity for a message to reach its target. RASCAL is developed as a contribution to the Palpable Computing (PalCom) initiative and forms a layer of the PalCom Communication stack [3]. PalCom is particularly concerned with the composition of devices, software services and other resources into connected assemblies that may flux over time, be non-localized and heterogeneous. Devices in these assemblies are typically loosely connected and thus require the network independent, flexible, and dependable connectivity offered by RASCAL.

As people can only trust and creatively employ technologies they can understand, RASCAL exhibits *palpability*, that is, people can notice and easily comprehend provided services and make them "plain or obvious"[1] or "palpable". RASCAL exploits the support for palpability that the PalCom open architecture provides through an easy to use GUI that allows users to notice and inspect the internal state of the system at multiple levels.

The remainder of this paper is organized as follows: Section 2 presents the design features of RASCAL. Section 3 describes the RASCAL architecture. Section 4 presents the evaluation of a prototype before concluding in Section 5.

2 Design Features of RASCAL

RASCAL is a middleware software that resides between a user application (such as email, instant messengers, Web browsers, GPS and map services, etc.) and the underlying network communication interfaces acting to enhance the user experience when these applications are used in disruptive environments. We define a *disruptive environment* as a location where one or more disruptive behaviours can affect the status of a system. Examples include natural disasters (e.g., tsunamis, hurricanes, floods, bush fires, etc.), major incidents (e.g., plane crash, traffic accident, building fires, etc.), and everyday events (e.g., power failure, server breakdown, etc.), etc.

Enhancing the user experience in disruptive situations implies making *autonomic decisions* when sending/receiving application messages to/from a target node. To make autonomic decisions, RASCAL must be aware of the network resources available in the environment and optionally of the usage context of applications. However, to make computing palpable, autonomic decisions should be taken in ways that users can notice and make sense of, and users should be supported in negotiating a different level of user control if they so wish.

With *connection-aware communication* we provide a set of policies enacted by the RASCAL prototype based on the status of the network when sending/receiving user application messages. These policies can for example be based on available network technologies or on some QoS parameters such as the load of the nodes/devices involved in the communication or with contingency situations such as failovers. A connection-aware policy could consider using an alternative interface for sending messages to a particular node if the currently used interface fails.

With *usage-aware communication* we provide a set of policies enacted by RASCAL based on deployed user-level services. Examples of these are: contingency management decisions, content adaptation decisions, deferred service provisioning decisions, role management decisions, etc. An example of a contingency policy particularly useful in disruptive environments consists of sending a message using two or more different routes simultaneously to increase the likelihood that the message reaches the target node.

[1] Oxford English Dictionary

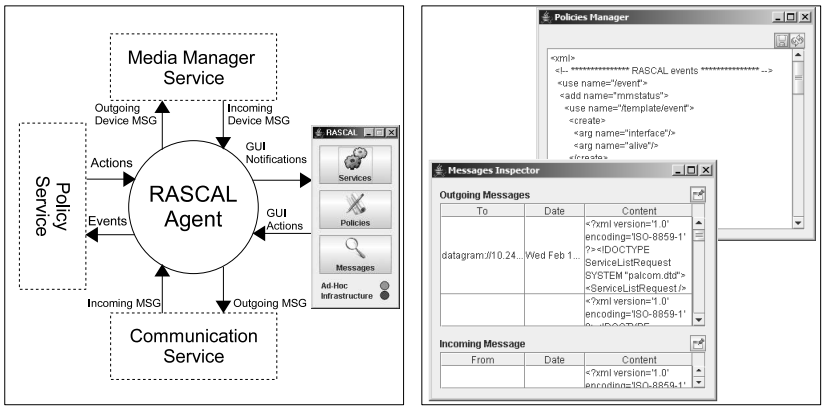
All these network and usage aware policies can be combined differently. However, the ability to combine in itself does not alone make for useful and powerful RASCAL functionality. It is here that RASCAL benefits from, and contributes to, the development of the PalCom open architecture. While people should not be inundated with system information, they do need to be able to notice and make sense of actual and potential functionality. Therefore, most of the time when disruption of communication occurs, RASCAL acts autonomically and automatically, while recording all actions taken through the user GUI.

3 The RASCAL System Architecture

A high level description of how RASCAL operates is as follows: a message to be sent by a PalCom device (e.g., user application) is intercepted by RASCAL which decides, based on a set of user-configurable policies, the most appropriate actions to take on that message. RASCAL can easily be plugged into the PalCom communication stack used by all PalCom devices, that in turn constitute parts of PalCom assemblies (see section sec:evaluation for an example). When RASCAL is integrated into a PalCom node the node becomes "RASCALized".

The RASCAL prototype is built using JADE [1], an open source software agent middleware. The operational logic of RASCAL is thus managed by a software agent which treats messages by applying policies before sending the message into the network.

To sense and negotiate with the external world the agent implements several JADE kernel services such as services able to communicate with underlying network layers, the routing or function layers and with the policy engine. Moreover, the RASCAL agent updates, and gets commands from, the end user through a GUI. Figure 1(a) shows the described architecture.



(a) Internal architecture. (b) Graphical user interface.

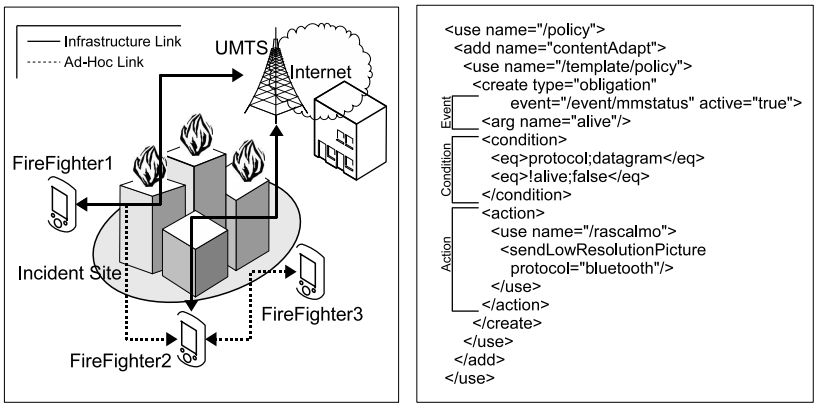
Fig. 1. The RASCAL System

The RASCAL user interface is an important part of the prototype providing support for palpability. It is able to receive all the events triggered by environmental changes. Figure 1(b) shows the outgoing and the incoming messages received by the current device and the policy editor. Its features are under a continuous refinement process based on feedback from end users.

Decisions on how to treat received messages can be applied using *policies*, that is, as a rule governing choices in the behaviour of systems. The used policy server is Ponder2 [2] which provides both a policy engine and an XML human readable policy description language.

4 Evaluation

RASCAL has recently been integrated into the iterative, participatory design process practiced in PalCom. We are currently carrying out experiments with end users in major incident emergency response scenarios. This section presents one of the recently conducted mock-up situations of a real-world major incident.



(a) Scenario. (b) Policy.

Fig. 2. Major Incident evaluation.

Within this experiment a building fire depicted in Figure 2(a) is considered. Here three firefighters are moving relatively close to one another to evacuate people from the building on fire. They are using special RASCALized digital maps (in our examples instantiated on tablet PCs, each with a built in camera and also running a map service, which shows the FireFighter where they are on the map). Their duty is to notify the command center and the other members of the team of findings related to the visited building(s), e.g., the positions of injured people. To do this, they make special marks on the map displayed on their tablet PC. Firefighters can also take pictures to assist the command center with gaining a visible overview of the overall incident status. Furthermore, in

this example, FF1 and FF2 are connected via both ad-hoc and infrastructure networks and FF3 only via an ad-hoc connection. In this situation, through the multi-hop capabilities of RASCAL, seamlessly all four actors (the three firefighters and the command center) are able to communicate with one another. For example FF3 communicates with the command center via the ad-hoc connection with FF2.

Figure 2(b) shows the definition of one of the policies defined in the firefighter's RASCALized devices. This policy sends low resolution pictures to the command center in the case when a device is not infrastructure connected.

In our experiment FF1 moves into an area where the infrastructure connection fails. The RASCAL agent running on the device is notified by the policy engine and hands over all communication with FF2 to the available ad-hoc connection. Given the importance of sending images to the command center and giving the low nominal bandwidth of the Bluetooth technology, pictures are first automatically reduced in quality (i.e., resolution) before transmission. Later, when FF1 returns to an area with infrastructure network coverage, communications with the command center are automatically returned to the infrastructure connection with images once again sent in normal, high resolution.

To achieve a balance between automation and user control RASCAL policies can be set to either automatically adjust quality according to connection constraints, or to ask the device user (i.e., the FireFighter) to explicitly select either higher quality/slower transmission or lower quality/faster transmission. Automatic decisions can also provide user notifications offering the option to override if necessary or preferred.

5 Conclusions

This paper has briefly presented the RASCAL prototype which helps users to palpably maximize the chances of messages reaching their target, even when moving in or through disruptive environments. In particular, this paper gives an overview of how RASCAL features can be useful in a major incident scenario. Perhaps the most innovative and powerful feature of RASCAL is how it supports people in making its actual and potential functioning palpable by defining resilience and adaptivity rules able to deal with disruptive environments such as major incidents.

The RASCAL prototype is currently under development. The outcome of the PalCom open architecture and the feedback of the end user are all inputs used to improve the prototype itself. Future work will concentrate on autonomic aspects, in particular on the definition of policies used in major incidents and improvement of the graphical user interface.

Acknowledgments. The authors acknowledge the EU Palpable Computing (IST-002057) FET project, which funded a proportion of the RASCAL project, and the many PalCom consortium members that contributed toward to work.

References

1. Bellifemine F. L., Caire G., and Greenwood D. *Developing Multi-Agent Systems with JADE.* John Wiley & Sons, 2007.
2. Damianou N., Dulay N., Lupu E., and Sloman M. The ponder policy specification language. In *POLICY '01: Proceedings of the International Workshop on Policies for Distributed Systems and Networks*, pages 18–38, London, UK, 2001. Springer-Verlag.
3. Andersen P., Bardram J. E., Christensen H. B., Corry A.A. V., Greenwood D., Hansen K. M., and Schmid R. An open architecture for palpable computing - some thoughts on object technology, palpable computing, and architectures for ambient computing. In *Proceedings of the Workshop on Object Technology for Ambient Intelligence*, 2005.