Spatio-Temporal Communication Primitives for Delay Tolerant Systems

Mirco Musolesi and Cecilia Mascolo
Department of Computer Science, University College London
Gower Street, London, WC1E 6BT, United Kingdom
{m.musolesi|c.mascolo}@cs.ucl.ac.uk

Abstract

Computing and communication devices pervasively surround our daily life and the presence of embedded systems, including tiny sensors, is increasing exponentially. However, the software and communication mechanisms used to network these devices are still the ones that we have been devised 30 years ago for standard computer systems. Different communication and coordination patterns are emerging for these environments, ranging from those related to delay tolerant systems *[3], where communication happens asynchronously between devices, to location based communication, where hosts receive information only when they are in a specific location. In these environments, several concepts, not captured by the semantics of the programming interfaces of traditional systems, such as location or temporal validity of the disseminated and replicated information, are fundamental.*

In this paper we propose a novel set of communication primitives for this kind of systems that would allow developers to better exploit the potential of these environments. These primitives combine spatial and temporal concerns to cope with the dynamics and mobility of pervasive systems. We also discuss a middleware framework that implements the proposed programming interface.

1 Introduction

The number of mobile phones in Europe is higher than the number of personal computers. Computing and communication devices pervasively surround our daily life and the presence of embedded systems, including tiny sensors, is increasing exponentially. Thanks to the progress in digital technologies, mobile computers have processing power and memory comparable to advanced and extremely expensive workstations of twenty years ago.

New possible scenarios are enabled by the availability of such technologies and their internetworking. One of the more promising is that of *delay tolerant networks* [3]. Delay Tolerant Networks are characterised by long delay paths and frequent (in some cases also unpredictable) disconnections and network partitioning. Possible examples may be intermittently connected mobile ad hoc networks [7], interplanetary and satellite communications [5] and mobile systems to provide transitive connectivity to isolated villages in rural areas [16, 2, 11]. In the Data Mules Project [15], for instance, the data of the sensor nodes are collected by a device (the "mule") that travels among them. Another existing solution is DakNet [11], which aims to provide intermittent connectivity to the global Internet to rural areas of India and Cambodia. People in villages access services such as email in e-kiosks: messages are collected and transported to (and from) an Internet gateway in the nearest town by buses. These are equipped with wireless technologies so that they can download and upload messages from and to the e-kiosks and the Internet gateways. Another example is the design of software systems for space exploration [10, 5]: delays in transmitting information due to the very long distances are common in this setting. Moreover, satellites, space probes and rovers may not be directly reachable from the Earth due to their position since they may be on the non visible side of a planet or covered by other celestial bodies.

The concept of delay tolerant networks represents a very general abstraction that includes existing systems relying on fixed networks, mobile ad hoc networks and hybrid networks composed of fixed and mobile nodes. For this reason, it provides a very general scenario for the design of primitives for a very large number of systems and deployment settings. On the other hand, communication mechanisms and algorithms currently used to network modern devices are still the ones that we have devised 30 years ago for traditional computer systems. These primitives may be still effective for a vast class of application scenarios, especially for business automation and scientific computing, however, they do not exploit the full potential of the pervasive computing scenarios, such as the one just described. In fact, classic programming paradigms and, in particular, programming interfaces defined for traditional middleware systems do not capture the natural communication paradigms typical of intermittently and location based networks; some examples include the ability to send a message to a number of hosts in a location or send it to hosts currently not connected to the same portion of the network (also with the possibility of specifying their location).

The contribution of this paper is the design of novel primitives for communication in delay tolerant mobile systems that fully exploits the potential of the environments considered, by combining spatial aspects with temporal ones. These primitives allow, for example, to specify that a message has to be sent (in a synchronous or asynchronous ways) to a recipient, only when this is in a specific location; or to all the devices that are in a specific location (i.e., *geocasting*). They also allow to express the fact that the delivery has to happen only when the sender is located in a specific point of the geographical space. Simmetrically, similar constraints and requirements in terms of space and time can be also specified by the recipients, for instance, by saying that the recipient will only want to receive a message from a certain sender, when reaching a specific location. Existing work in the area has focussed more on the pure networking aspects [3], rather than on the analysis and the design of programming paradigm and interfaces. To our knowledge, there are no works on these issues. We have also designed a general architecture and developed a prototype that implements the semantics of the proposed middleware interface. To summarise, the contribution of this paper is twofold:

- we propose a set of primitives that can be used to specify various spatio-temporal aspects of communication in settings where these issues are fundamental, such as in delay tolerant networked systems;
- we present a middleware architecture that gives semantics to this programming interface and briefly discuss the implementation of a prototype.

The paper is organized as follows. In Section 2 we analyse the challenges and the requirements of the design of communication primitives in delay tolerant mobile networks. In Section 3 we describe the communication primitives and we outline an architecture of the middleware offering them as an API. The proposed model is then discussed in Section 4, where we also outline our current research directions.

2 Communication in Delay Tolerant Mobile Networks: A Scenario

In order to define the design requirements of the primitives for delay tolerant mobile systems, we start from a realistic example, considering the case of providing communication to a village in a poor area, distant from the nearest main town and therefore connected to the global Internet by means of a bus that acts as message carrier (as in the DakNet project [11]). We suppose that the village is composed of three main locations covered by a local network that is disconnected from the Internet[1]: the residential area, the local administrative offices and the rural area near the village. We also assume that all nodes in the village are connected by a LAN attached to a WiMAX network that covers the rural area outside the village, providing connectivity to all the farmers. In a sense, the village can be seen as a connectivity island. When the bus is in proximity of the main town, it is connected to the Internet by a wireless gateway, whereas, when it is in the village, it is connected in a similar way to the local network.

In this setting, the ability to send messages to a location, to exploit the asynchronous communication through the use of the bus as a *store and forward* engine for messages, or to deliver messages to all hosts in a location, are examples of communication patterns difficult to express by using the synchronous and asynchronous primitives exploited by most middleware for traditional systems. The argument of this paper is that middleware should provide support for, for instance, sending a message from a host in the global Internet to a recipient in the isolated (or better, *intermittently connected*) village and viceversa. A host in the global Internet should also be able to send a message to all the hosts that are in a specific geographical region. Messages may have an expiration time, in order to express the validity of the information. In fact, a message containing weather forecast about a certain day, for instance, would not be so useful if received on the same day (or after).

Let us envisage some more specific scenarios: for example, the central meteorological office in the near big city should be able to send a weather alert to all farmers in the rural area. Let us further assume that the computer system of the offices is managed by technicians remotely from the near city: a system administrator should be able to specify his/her interest in receiving only requests from clerks of the offices in the villages that he/she manages and supervises. What is needed in this case is a **send**() primitive that allows developers to specify that a message has to be delivered to a certain recipient, in a certain area of the village, or to all the recipients that are in that area, or to a certain recipient only if he/she is in that particular area and so on. A symmetric semantics should be made available to indicate the recipients and/or their locations in the **receive**() primitive.

As far as the message reliability is concerned, a critical weather alert should be sent with high reliability, whereas an ordinary hourly update of the wheather forecast may be characterised by a lower one. Let us consider again our scenario. Let us suppose that a sensor network is deployed in the fields in the rural area outside the village, in order to measure environmental indicators such as humidity, pollution and so on. We suppose that the data are collected by

[1]As discussed in [11], considering the user requirements in these scenarios, the connectivity to the Internet by means of a satellite link or a wired line is not convenient in most of the cases.

means of motorbikes and helicopters. We would also like to be able to express the fact that the collection of the data will be performed only in certain locations (i.e., receival should only happen in these locations). In other words, we would like to be able to specify not only the locations of the senders and the receivers of the message, but also where the **send**() and the **receive**() should be fired.

To summarise, there is a need to provide primitives that enable and combine:

- *synchronous/asynchronous* communication (i.e., the sender and/or the receiver are or are not blocked awaiting for the successful execution of the primitives);
- *delay tolerant/non delay tolerant* communication (i.e., it is admissible or not that messages will be delivered with a delay that may not be negligible);
- *spatial/non spatial* communication (i.e., there is the support for geo-casting and location-awareness or not).

In the following section we will present a set of primitives that allows developers to specify these dimensions and we will discuss a middleware architecture that supports them.

3 Middleware Support for Spatio-Temporal Delay Tolerant Communication

3.1 Definition of the Primitives

We now present a detailed definition of the communication primitives. The primitives incorporate spatio-temporal concepts which, for instance, allow the description of the operations sketched in Section 2. The novelty of these primitives resides in their expressiveness and their flexibility, since, by using them, the software engineer can specify a wide range of requirements and constraints for the communication process. More specifically, we define a new set of primitives for sending and receiving messages. In order to meet the requirements defined in Section 2, the **send**() primitive has the following signature:

send (*m*, *recipient*, *recipientLocation*, *senderLocation*, *tExp*, *tBlock*, *reliability*)

By using this primitive, developers are not only able to define the recipients of the message *m* (in *recipient*), but also to express spatial concepts, such as the location where the message has to be delivered to (in *recipientLocation*) and the location where the effective sending has to be performed (in *senderLocation*) (i.e., the sending is performed only when the host is in the location expressed in *senderLocation*).

Moreover, the *recipient* field can assume two values, the identifier of the receiver, a list of receivers or $*$, to indicate that the message is sent to every host. Similarly, developers can specify one recipient location or a generic location (using the same symbol $*$)[2].

In order to clarify these concepts, let us consider some examples:

- **send** (*m*, *, *,...) has to be used to send a message to all the hosts, independently of their position;
- **send** (*m*, 32, *,...) indicates that the messages is to be sent to host 32; host 32 can receive it independently of its position;
- **send** (*m*, 32, *ruralArea*,...) indicates that the message is to be sent to host 32; host 32 can receive the message only when it is in location *ruralArea*;
- **send** (*m*, *, *ruralArea*,...) indicates that the message is to be sent to all hosts in location *ruralArea*.

By using the *tExp* field, developers are able to set the expiration time of the message, whereas *tBlock* defines the interval of time during which the application is blocked waiting for the correct delivery of the message. The expiration time indicates the validity of the message. The corresponding acknowledgment message will have the same expiration time.

Through these timers it is possible to specify synchronicity and asynchronicity on top of the possible location based operations specified above; for example, **send** (*m*, 32, *,.., 20, 20,...) indicates that the message is to be sent to host 32, independently of its position, but that the message expiration time is set to 20 time units and that the application is blocked for 20 time units while waiting for this to be delivered and acknowledged. On the other hand, **send** (*m*, 32, *,.., 20, 0,...) will be used for the asynchronous delivery of a message with an expiration time equal to 20 time units.

Finally, the desired reliability of the delivery process can be specified as a percentage in the *reliability* field. Since, in many cases, the evolution of the deployment scenarios cannot be predicted with accuracy (i.e., it is not deterministic), the reliability value specified in the sending primitives is evaluated in probabilistic terms. The middleware driven delivery process associated to the primitives depends on the *dissemination strategy* used to replicate or forward the messages to the other hosts. If messages cannot be delivered immediately (i.e., when the recipients are not in the same connected portion of the network), they are stored in intermediate buffers that we call *Message Buffers*. The dissemination strategy is composed by a set of algorithms and protocols used to transfer and disseminate the information in the system in order to deliver the information as close as

[2]It is clearly possible to extend the syntax and the semantics of these primitives in order to specify a list of senders and receivers in different locations, by using a list of tuples with the format *(hostId,LocationId)*.

possible to the recipient(s) and/or to the the locations specified in the **send**() primitive.

Messages are transferred from the Message Buffer of a host H_A to that of a host H_B using *forwarding mechanisms* (i.e., the message is transferred from H_A to H_B then the message is deleted from the Message Buffer of H_A) or *replication mechanisms* (messages are copied from H_A to H_B and the copy on H_A is maintained).

We define, in a symmetric way, the **receive**() primitive as follows:

m=**receive** (*sender*, *senderLocation*, *receiverLocation*, *tBlock*)

Similarly to the **send()** primitive, developers can specify the sender (in *sender*), its location (in *senderLocation*), the location where the receiving needs to happen (in *receiverLocation*) and the time interval during which the receiving application is blocked waiting for a message (in *tBlock*).

3.2 Implementing the Middleware Interface

The primitives have to be embedded in a middleware framework that supports the dissemination and the persistence of the information as shown in Figure 1. These aspects are key in delay tolerant networking. In fact, if a message cannot be delivered immediately as the recipient is not in the same connected portion of the network, the message might have to be stored in intermediate *Message Buffers*. For this purpose, intelligent and reliable replication and forwarding strategies must be devised. Clearly, the reliability of the system is strictly dependent on the use of the available resources, especially in terms of memory, bandwidth and computational power. These issues are extremely relevant in the case of mobile devices. Therefore, there is a trade-off between the reliability of the system and the resource consumption. The reliability value will be used by the underlying delivery protocols in order to make decisions about the exploitation of the available resources. In this sense, the reliability value is interpreted as a sort of message priority.

We envisage a middleware that deliver messages by using different dissemination and message forwarding strategies that form a *Protocols Library*. These will be selected and tuned to ensure the expected reliability by the *Protocol Selector* component. There are many possible ad hoc solutions that suit best depending on the application scenarios. We believe that the middleware should be able to select the best suitable protocol according to the values assigned to the parameters of the primitives. It is outside the scope of the paper to describe the different dissemination strategies that can be used to implement specific delivery probabilities. Possible solutions which could be plugged into the framework can be gossip-style dissemination algorithms, based on epidemic techniques [9] or intelligent forwarding protocols based on the history of the system [7]. Let us briefly consider the case of the availability of three dissemination and forwarding protocols, a synchronous routing protocol (like DSDV [12]), a pure epidemic protocol [17] and a generic geo-routing protocol based on geographical information. A reasonable selection policy might consist in using the synchronous protocol only if the recipient is in the same connected portion of the network. In our example, this is the case of messages that are exchanged by two hosts that are inside the village. If the recipient is not directly reachable and if the developer has specified the location, the geo-routing algorithm might be chosen. If this information is not available, the epidemic protocol has to be selected to try to reach the recipient by spreading replicas of the message across the system (in our case study, for example, by disseminating a copy in every device in the rural area).

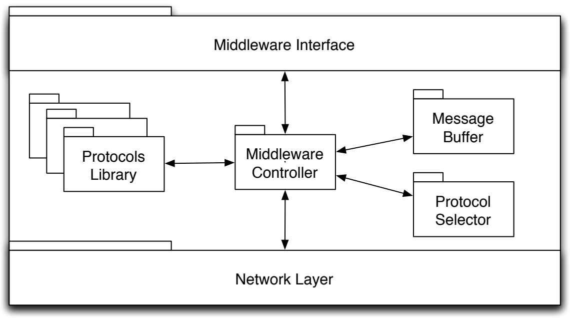

Figure 1. Middleware Architecture.

With respect to the reliability issues, as far as the geo-routing protocol is concerned, the number of forwarding paths towards the location can be proportional to the value expressed by the developer. In the case of the epidemic protocol, if for example the reliability value is 100, the message might be replicated on every host, whereas if it is lower, it might be copied only on a subset of nodes. In general, the epidemic-style replication mechanism in terms of number of copies can be tuned according to the desired reliability with good accuracy. The number of replicas will be proportional to the desired reliability.

We are developing a first prototype of a middleware that implements the communication primitives presented in Section 3.1. The middleware will rely on the Context-aware Adaptive Routing (CAR) protocol [7] and a tunable epidemic protocol based on models of epidemic spreading in complex networks [8], to support asynchronous communication in case of disconnection. Location information is used in this case only to check the geographical locations of senders and receivers and not to drive the delivery process. In fact, it is important to note that the underlying protocols may only partially support the general semantics of the primitives [3].

[3]In general, this is the case of the protocols that do not support geo-

4 Discussion and Future Work

In the recent years, the research community has proposed new paradigms and architectures for communication and coordination in the general area of pervasive computing. The proposed solutions have been founded on several different mechanisms and abstractions such as the sharing of tuple spaces [13] and events [6], which go beyond the traditional synchronous communication mechanisms imposed by standard middleware systems. Other works have been focussed on other aspects of the communication in pervasive environments, such as context-awareness [14] and location-awareness [1]. In general, the pervasive computing scenario seems to offer more opportunities than those exploited for more complex communication primitives.

This is even more true if we extend to study delay tolerant networked systems. The challenges posed by this scenario were firstly discussed by Fall in [3]. Some examples of existing prototypes of delay tolerant systems have been presented in Section 1, such as DakNet [11] and Data Mules [15]. However, in these works, the authors do not discuss the potential of these systems in terms of primitives and the set of operations which could be performed on such systems. To our knowledge, our work represents the first attempt to the design and the specification of communication primitives in delay tolerant networks.

In general, the current work in delay tolerant networking propose ad hoc solutions targeting particular problems in specific deployment scenarios. Moreover, they are more focussed on networking issues rather than on the definition of application-level abstractions and programming paradigms. The contribution and the novelty of our work resides in the definition of a flexible and expressive set of primitives and high-level abstractions for this challenging class of networks and in general for systems for which spatio-temporal aspects are important. We plan to investigate possible integration with existing delay tolerant network technologies [3] in order to provide a common programming interface for systems that rely on different delivery mechanisms.

In this paper, we have presented a novel set of primitives for communication in delay tolerant networks. We consider this work as the foundation and the starting point of our investigation on the design of middleware for delay tolerant systems and, in general, for systems where spatial and temporal aspects are relevant. In fact, the primitives presented in this paper may be used to design more complex distributed systems. Namely, the middleware architecture could be extended with the addition of a layer for publish/subscribe systems for delay tolerant networks. Subscriptions and notifications may be delivered by means of mechanisms similar to those described in this paper. In fact, by considering and learning from past experiences in designing building publish/subscribe systems for delay tolerant mobile ad hoc networking [9], we really believe that the existing programming interfaces (such as the JMS interface [4]) are not able to capture the essential aspects of the interaction in these settings.

casting. On the other hand, there may be situations where geographical information are not available (also temporarily) so that location-aware routing is not achievable.

References

[1] M. Bauer, C. Becker, and K. Rothermel. Location Models from the Perspective of Context-Aware Applications and Mobile Ad Hoc Networks. *Journal of Personal and Ubiquitous Computing*, 6(5/6), December 2002.

[2] A. Doria, M. Uden, and D. P. Pandey. Providing Connectivity to the Saami nomadic community. In *Proceedings of the Second International Conference on Open Collaborative Design for Sustainable Innovation*, December 2002.

[3] K. Fall. A delay-tolerant network architecture for challenged internets. In *Proceedings of SIGCOMM'03*, August 2004.

[4] M. Hapner, R. Burridge, R. Sharma, J. Fialli, and K. Stout. *Java Message Service Specification Version 1.1.* Sun Microsystems, Inc., April 2002. http://java.sun.com/products/jms/.

[5] Internet Society. Interplanetary Internet Project. http://www.ipnsig.org.

[6] R. Meier and V. Cahill. STEAM: Event-Based Middleware for Wireless Ad Hoc Networks. In *22nd International Conference on Distributed Computing Systems Workshops (ICDCSW '02)*, pages 639–644, June 2002.

[7] M. Musolesi, S. Hailes, and C. Mascolo. Adaptive routing for intermittently connected mobile ad hoc networks. In *Proceedings of the 6th International Symposium on a World of Wireless, Mobile, and Multimedia Networks (WoWMoM 2005). Taormina, Italy.* IEEE press, June 2005.

[8] M. Musolesi and C. Mascolo. Controlled Epidemic-style Dissemination Middleware for Mobile Ad Hoc Networks. Technical report, Dept. of Computer Science, University College London, November 2005.

[9] M. Musolesi, C. Mascolo, and S. Hailes. EMMA: Epidemic Messaging Middleware for Ad hoc networks. *Journal of Personal and Ubiquitous Computing*, 2005. To appear.

[10] NASA. Mars Exploration Program Website. http://marsweb.jpl.nasa.gov/.

[11] A. S. Pentland, R. Fletcher, and H. Hasson. Daknet: Rethinking connectivity in developing nations. *IEEE Computer*, 37(1):78–83, January 2004.

[12] C. E. Perkins and P. Bhagwat. Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers. In *Proceedings of SIGCOMM 94*, pages 234–244, August 1994.

[13] G. P. Picco, A. L. Murphy, and G.-C. Roman. LIME: Linda Meets Mobility. In D. Garlan, editor, *Proceedings of the 21st International Conference on Software Engineering (ICSE'99)*, pages 368–377, Los Angeles, CA, USA, May 1999. ACM Press.

[14] M. Roman, C. Hess, R. Cerqueira, A. Ranganathan, R. H. Campbell, and K. Nahrstedt. Gaia: a Middleware Platform for Active Spaces. *ACM SIGMOBILE Mobile Computing and Communications Review*, 6(4):65–67, 2002.

[15] R. C. Shah, S. Roy, S. Jain, and W. Brunette. Data MULEs: Modelling a Three-Tier Architecture for Sparse Sensor Networks. Technical Report IRS-TR-03-001, Intel Corporation, January 2003.

[16] I. T. Union. Connecting remote communities. *Documents of the World Summit on Information Society*, 2003. http://www.itu.int/osg/spu/wsis-themes.

[17] A. Vahdat and D. Becker. Epidemic routing for partially connected ad hoc networks. Technical Report CS-2000-06, Department of Computer Science, Duke University, 2000.