Communication and Coordination Support in Ad Hoc Networks for Emergency Management Scenarios

[Position Paper]

José Mocito
INESC-ID / FCUL
Portugal
jmocito@gsd.inesc-id.pt

Luís Rodrigues
INESC-ID / IST
Portugal
ler@ist.utl.pt

Hugo Miranda
University of Lisbon
Portugal
hmiranda@di.fc.ul.pt

ABSTRACT

In recent years the world has witnessed many catastrophic events where the intervention of first responders was required to manage massive disaster scenarios. To be effective, emergency management teams must be able to coordinate and communicate efficiently. To the best of our knowledge no solution has been proposed to support the communication and coordination of these teams in an integrated fashion. In this paper we propose MUSTUS, an architecture that provides communication mechanisms and coordination primitives, based on the interaction through a tuple space, whose semantics are specifically designed to support the needs of emergency management applications.

Categories and Subject Descriptors

C.2.1 [**Network Architecture & Design**]: Wireless communication; C.2.2 [**Network Protocols**]: Protocol architecture

General Terms

Algorithms, Design, Reliability

Keywords

Tuple space, mobility, coordination, emergency management

1. INTRODUCTION

In recent years the world has witnessed many catastrophic events where the intervention of first responders was required to manage massive disaster scenarios. Examples such as the tsunami in the Indian Ocean (Dec. 26, 2004) or the hurricane Katrina (Aug. 29, 2005), to name a few, have shown that disaster recovery teams need to be able to communicate and coordinate effectively, in order to provide efficient emergency assistance. This has highlighted the relevance of research in technologies for the support of emergency management activities.

Recent research has addressed the problem of energy efficient routing [17], modeling the mobility of rescue agents [2], or characterizing the kind of traffic produced in such scenarios [3]. However, to the best of our knowledge, no effective solution has yet been proposed that addresses the coordination requirements of teams acting in disaster recovery situations.

The way rescue teams are organized in a disaster location is usually defined by tactical decisions. Typically some form of command center is established to supervise the operations on the field. All the teams must report to their respective *leaders*, which must coordinate the placement and activities of each team, in order to maximize the effectiveness of their operations.

Communication and *coordination* support is therefore of the utmost importance to the success of rescue missions. In this paper we propose MUSTUS, an architecture that provides communication mechanisms and coordination primitives whose operation semantics are specifically designed to support the needs of emergency management personnel. This is achieved through the use of a specialized tuple space that provides implicit communication through a shared data storage and coordination support through operations that change the state of the tuple space.

The rest of the paper is structured as follows. Section 2 surveys the related work. The characteristics of the disaster recovery scenario considered are described in Section 3. The architecture is presented in Section 4. A description of the tuple space service is provided in Section 5. The two strategies supported by the data dissemination layer are presented in Section 6. Section 7 describes a set of example tuple properties and how they reflect in the behavior of the underlying mechanisms. Finally, Section 8 provides the concluding remarks.

2. RELATED WORK

Research on communication support for emergency management in disaster area networks is prolific, addressing a wide range of topics, from requirements definition to the service deployment.

Jang et al. [10] propose a set of requirements for a MANET based rescue information system targeting earthquake dis-

*This work was partially supported by FCT project REDICO – Dynamic Reconfiguration of Communication Protocols – (PTDC/EIA/71752/2006) through POSI and FEDER.

asters and propose a solution where seismic related information is collected by notebook PCs and fed to a command center. However, very little detail is given on the supported coordination capabilities.

Graaf et al. [4] define a set of requirements for emergency networks and propose an architecture focused on routing and multicast support.

Guo et al. [8] study the dynamic placement of relay nodes as a solution to overcome network partitioning of the ad hoc network formed by first responders.

Alves et al. [1] propose a solution for the deployment of a backup network that helps mitigate the effects of partial or complete disruption of the primary communication infrastructure used by rescue personnel. The idea is to use off-the-shelf equipment along with specialized software that provides useful services, like data dissemination and storage or location awareness. Our proposed architecture is less general, but more simple and focused on the efficient coordination support.

Tuple spaces for ad hoc networks have also been studied for scenarios other than emergency management, and several systems have been proposed in the literature.

LIME [14] is an extension of the Linda model specifically tailored to support applications for mobile hosts. Every host has access to a local tuple space that is transiently shared with other hosts in the immediate vicinity, forming a federated tuple space. This is accomplished by merging both tuple spaces whenever two hosts become in radio contact with one another. Since multi-hop sharing of a tuple space is not supported this system is not adequate for remote coordination support.

EgoSpaces [11] is a coordination model and middleware for ad hoc mobile environments. It introduces the notion of a view, that encompasses the subset of tuples available at other hosts, selected according to some criteria, like number of hops, tuple patterns or geographic context. Although a complete specification of a view is provided, few details are provided on how to efficiently implement the implicit filtering during routing.

TOTA [12], as opposed to the previous solutions, proposes a tuple-centric approach, where tuples are spread hop-by-hop among nodes according to some associated propagation rules. These can include the scope, i.e. the distance and/or direction of propagation, and dependency relations regarding other tuples, i.e. the presence or absence of other tuples can affect the propagation. Furthermore, the propagation rules can also contain instructions on how the tupleâĂŹs content should change during propagation. Like LIME, tuples are only accessible when stored locally or by immediate neighbors.

All the above mentioned solutions provide a set of operations that support implicit communication and coordination by manipulating a shared tuple space. However, LIME only supports transient sharing of tuple spaces between immediate neighbors, while EgoSpaces and TOTA do not provide optimized communication mechanisms for MANETs in emergency management scenarios.

3. EMERGENCY MANAGEMENT SCENARIOS

In emergency management scenarios teams of first responders operate together in order to extinguish fires, rescue people from debris, maintain public order and safety, repair infrastructural damages, among other activities. These teams are usually composed by firefighters, police officers, paramedics, civil protection officers and construction agents. Depending on the specificity of the emergency situation, other structures are deployed, such as emergency transportation sites (ambulances and helicopters), temporary field hospitals and temporary housing for affected civilians. The dynamics and responsibilities of each team and functional unit imply a high degree of coordination activity, usually performed by a command center deployed near the incident site.

In this paper we assume no pre-existing communication infrastructure and that every agent involved in the disaster recovery operation holds an electronic device, capable of wireless ad hoc communication. Examples of such devices are PDAs, tablets or specialized artifacts developed specifically for emergency situations that possess standard wireless communication capabilities (e.g. 802.11b).

Moreover, we also consider that mobility is unrestricted and non-uniform, with patterns ranging from quasi-stable (e.g. agents in the command center) to highly mobile (e.g. firefighters). Furthermore, we assume that when the amount and positioning of agents across the operational space is not enough to ensure connectivity of the ad hoc network, controlled dedicated relay disposition can be used to avoid partitioning [8].

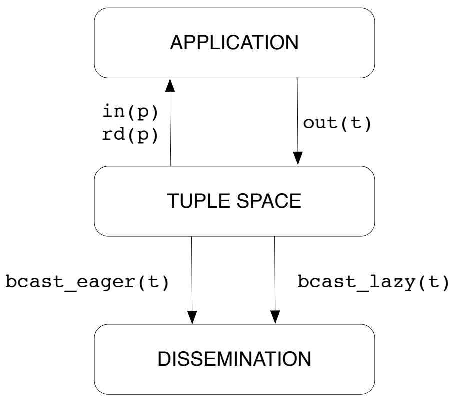

Figure 1: Architecture.

4. THE MUSTUS ARCHITECTURE

Given the lack of solutions for providing an integrated support for communication and coordination in emergency management scenarios, we propose a modular MUlti-Strategy TUple Space architecture (MUSTUS) to address this problem, as illustrated in Figure 1. The architecture is based on three layers: the application, the tuple space service, and the dissemination service.

The application layer is responsible for providing the user interface and functionality required to operate an information system capable of aiding rescue personnel, and its implementation is out of the scope of this paper. The tuple space service is responsible for providing the coordination primitives provided to the application. Such primitives encapsulate the communication activities performed by the bottom-most layer, the dissemination service. In the next sections

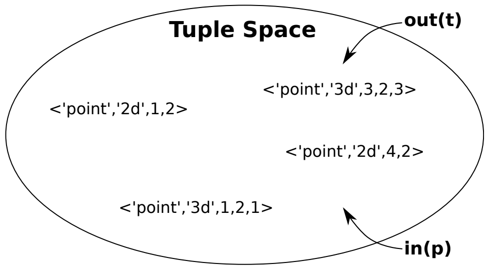

Figure 2: Example of a tuple space holding 2D and 3D coordinates.

we describe in detail the characteristics and competences of both layers.

5. TUPLE SPACE

The *tuple space* is a programming paradigm popularized by the Linda coordination language [7]. It provides a communication and coordination model where a group of processes is allowed to store and retrieve objects from a shared, virtual, associative memory (cf. Figure 2).

A tuple consists of an ordered, unlimited sized, set of values. For instance, a point in a Cartesian space can be represented as `<'point',3,5>`.

Usually three operations are supported by all tuple space implementations:

`out(t)` stores the tuple `t` in the tuple space

`in(p)` retrieves a tuple that matches pattern `p`

`rd(p)` reads (but not erases) a tuple that matches `p`

Tuple matching is usually performed through wildcards and literals. For instance the pattern `<'point',?,?>` would match any point of a Cartesian space.

The last two operations are typically blocking, only returning when a tuple that matches the pattern exists or is inserted in the tuple space. To provide more flexibility to the developer most systems also provide non-blocking variants of these primitives, that are allowed to return null values when no tuple matches the pattern.

Coordination can be achieved through a combination of (blocking) operations that modify the state of the space (`out`, and `in`) in order to achieve a synchronized behavior. A typical example in emergency management scenarios is the distribution of rescue teams to incidents. For instance a paramedic team can issue an `in` operation on a tuple of the form `<'medical',?,?>` which will only return when an `out` operation inserts a tuple patching the pattern, for instance, `<'medical','38.73°','-9.30°'>`. No other team will be directed to the same spot, since the tuple is effectively retrieved from the space.

Three challenges arise when developing a tuple space service for a MANET network in an emergency management scenario: i) data access patterns, ii) data availability, iii) data timeliness.

5.1 Data Access Patterns

Storing data in ad hoc networks is a complex task that, depending on the level of availability required, may consume substantial resources (bandwidth and power). Therefore, the way data is accessed in a tuple space should affect how tuples are stored in the ad hoc network. For tuples that are seldom removed and read by many nodes it is worth to spend resources in the `out` operation, by creating multiple copies of the tuple in the network, such that the `rd` operation can be implemented efficiently. On the contrary, tuples that are removed frequently, but read only sporadically, should be placed close to the source of the updates, such that updates are cheap at the expense of slower reads.

There are also tuples that do not need to be removed by an `in` operation. In this case it is possible to improve the access times of `rd` operations, by creating multiples copies of the tuple and by spreading these copies in the network such that they can be closer to the readers.

In MUSTUS, each tuple may be labeled with one or more *property* tags. One of these tags characterizes the expected access patterns to the tuple. This tag is used by the tuple space implementation to decide on the number and location of the tuple replicas to be placed in the ad hoc network.

The tags supported by MUSTUS regarding data access patterns are as follows:

ILRH `in` low, `rd` high. A tuple for which both *in* and *rd* operations are supported, that is seldom removed and read frequently.

IHRL `in` high, `rd` low. A tuple for which both *in* and *rd* operations are supported, that is frequently removed and read sporadically.

ROH `rd`-only high. A tuple for which the `in` operation is not supported, that is read frequently.

ROL `rd`-only low. A tuple for which the `in` operation is not supported, that is read sporadically.

Note however, that the tags above are not the only factor that affects the number and placement of tuple copies. Data availability needs to be taken into account as well, as explained below.

5.2 Data Availability

Another challenge in developing a persistent and reliable storage service, like a tuple space, is the data availability guarantees. Since in emergency management scenarios nodes are likely to become disconnected (due to terrain features, battery exhaustion, physical damage, etc), one needs to use data replication to ensure that data remains available to all the nodes in the network.

Also, since it is impossible to maintain an up-to-date view of the status of each node in the system, update operations (i.e., `out` and `in` operations) need to be performed on a quorum of replicas. As in any system of this nature, the use of a higher number of replicas provides more fault-tolerance but also induces higher communication costs, since larger update quorums need to be used (update quorums must intersect).

Read-only tuples are a special case where quorums are not required, since `out` is the only update operation and therefore replicas never get inconsistent.

Therefore, in MUSTUS, each tuple is associated with a *replication tag.* Data with greater availability requirements is stored using a higher replication factor (tagged with HRF), while less relevant information can be stored with a lower cost and be available as long as the small number of replicas exists in the network (tagged with LRF). Each replication

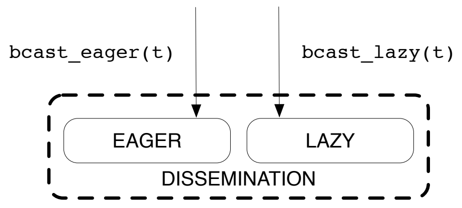

Figure 3: Data dissemination layer.

tag has an associated replication degree interval, that defines the minimum and maximum number of replicas that should store the data. A replication tag HRF corresponds to a higher interval while the LRF tag corresponds to a lower interval. The exact number of replicas is a number in this interval that is determined by weighting the characteristics of the associated data access pattern.

Also, for a given replication degree, the quorum of read and write operations is adjusted according to the declared access patterns for that tuple.

5.3 Data Timeliness

In critical scenarios it is common for data to have associated timeliness properties. To start with, different data may have different time constraints on how fast it needs to be disseminated in the system. Very urgent notifications may be required to be added/retrieved in an eager manner (tagged FAST), while other information may be added/retrieved in a lazy fashion (tagged LAZY). For instance, a remote request for medical assistance must be made available to the paramedic teams with minimum delay. On the other hand, logistic information, such as the expected time for the arrival of supplies may be propagated at a slower rate, in order to save the scarce communication resources. Typically, lazy techniques are less demanding on network resources. In MUSTUS, the timeliness requirements of data are used to select the communication mechanisms employed to execute the operation on the tuple space (discussed in detail in the next section).

Furthermore, many tuples will have a limited lifespan, as their contents become obsolete after some time. The use of tuple lifespans is employed in MUSTUS as a means to garbage collect information that is no longer relevant, to ensure that enough resources exist in the system to propagate and store new tuples.

6. DATA DISSEMINATION

As discussed in the previous section, tuple operations are implemented by creating a number of replicas for the tuple and then updating a quorum of these replicas. Furthermore, MUSTUS attempts to place these replicas on the ad hoc network in a manner that optimizes the accesses according to the declared access pattern for each tuple.

There are two basic strategies to perform these operations in an ad hoc network, namely, *eager* and *lazy* dissemination strategies (cf. Figure 3).

In the eager data dissemination scheme, an update or query is immediately propagated in the network, such that the desired number of replicas is created or contacted as soon as possible. This is the solution that provides lower latencies, and should be used for tuples with stringent timeliness requirements.

However, the effective use of communication resources is one of the hardest challenges faced when designing middleware for MANET environments, due to the low bandwidth and high packet drop ratio that usually characterizes the wireless medium. Moreover, contention and packet collisions can be easily triggered by an uncontrolled use of the communication resources, also affecting the provided quality of service. Therefore, it is interesting to use alternatives to eager data propagation that allow for a more efficient use of network bandwidth. Such alternative is lazy data dissemination. In this strategy the update or the query is not disseminated immediately. Instead, it is combined with other updates or queries, and propagated in the background.

In the next paragraphs, we discuss how eager and lazy dissemination strategies are materialized in MUSTUS.

6.1 Eager Dissemination

Eager dissemination is accomplished through an improved flooding scheme that ensures immediate data propagation while limiting the number of nodes that forward a packet. Many heuristics can be used to decide on whether to propagate a packet or not, and the literature is rich in such proposals, including probabilistic [9], counter-based [16], distance-based [13], and location-based [5].

Location-aware solutions provide the best trade-off between the communication cost and the delivery ratio, since they can effectively pinpoint the best forwarders. However, obtaining accurate position readings is not always possible, and the alternative distance-based schemes provide better performance in such situations, when compared to the probabilistic and counter-based approaches.

Therefore, to be able to cover all the emergency management scenarios MUSTUS uses the PAMPA [13] distance-based algorithm as the eager dissemination scheme.

6.2 Lazy Dissemination

Lazy dissemination relies on a gossip propagation process inspired by solutions like SharedState [6] or EraMobile [15].

Using background gossip for data dissemination is not a frequent choice, given the high latency of the approach. However, in challenging scenarios where the available communication resources are scarce and the mobility of nodes favors opportunistic communication, the trade-off between not being able to send messages due to contention or high packet drop rate, or sacrificing latency, can favor background gossip.

MUSTUS integrates a lazy dissemination scheme inspired by SharedState [6] where every node has a local cache where it deposits the produced messages along with the data received from neighboring nodes. The dissemination process evolves in rounds, where a random subset of the stored messages is selected for retransmission. Upon reception, this set of messages is collected in the local storage and a discard policy ensures that storage space remains available.

7. TUPLE PROPERTIES EXAMPLES

To illustrate our approach, we now provide a set of examples of how the properties associated with a tuple can drive the behavior of the MUSTUS components. The examples consider several combinations of the different tags supported in the system, namely, access pattern, availability and timeliness tags.

ILRH,LRF,SLOW.

An example of a tuple with this combination of tags is a tuple that marks a useful landmark found by a team (for instance, a water source or an obstacle). Given that the tuple is seldom removed but frequently read (ILRH), it is copied to as much replicas as possible. This number is limited by the associated low replication degree (LRF) therefore it will be the maximum allowed value in the lower replication interval. Since the data is not urgent (SLOW) the lazy scheme is used to disseminate the tuple to the respective replicas.

IHRL,HRF,FAST.

Typical information with such properties are orders from the command center to field personnel, for instance to provide assistance to victims on specific locations. Given that the tuple is frequently removed but seldom read (IHRL), it is copied to as fewer replicas as possible, in order to improve the response time of the `in` operation. Since the associated replication degree is high, the actual number of replicas used corresponds to the minimum value in the higher replication interval. The urgency of the data (FAST) results in the eager dissemination being used to transmit the tuple to the respective replicas.

ROH,HRF,FAST.

An example of a tuple with this combination of tags is a tuple that marks a location to be avoided by teams without proper protection (for instance, a contaminated building). Given that tuple is read-only and frequently read (ROH), it can be spread to as much replicas as possible in order to improve the speed of the `rd` operation. Since the replication degree is high (HRF), as long as storage space is available every node in the network will store the tuple locally. Moreover, due to the urgency of the data (FAST) the eager dissemination scheme is used to spread the tuple throughout the network.

8. CONCLUSIONS

Research in communication and coordination support in emergency management scenarios using ad hoc networks is gaining relevance, due to the proliferation of powerful mobile devices easily carried by first responders, and the flexibility offered by not requiring a preexisting communication infrastructure.

In this paper we introduce the MUSTUS architecture, that provides efficient communication and coordination support through the use of a tuple space service specifically designed for disaster area networks, where communication resource usage, data availability and information usefulness are critical to the performance of the emergency management teams.

The simplicity of the operations provided allows application developers to produce effective services to support the coordination activities in this type of scenario.

9. REFERENCES

[1] S. Alves, B. Koldehofe, H. Miranda, and F. Taiani. Design of a backup network for catastrophe scenarios. In *IWCMC '09: Proceedings of the 2009 International Conference on Wireless Communications and Mobile Computing*, pages 613–617, New York, NY, USA, 2009. ACM.

[2] N. Aschenbruck, E. Gerhards-Padilla, M. Gerharz, M. Frank, and P. Martini. Modelling mobility in disaster area scenarios. In *MSWiM '07: Proceedings of the 10th ACM Symposium on Modeling, analysis, and simulation of wireless and mobile systems*, pages 4–12, New York, NY, USA, 2007. ACM.

[3] N. Aschenbruck, M. Gerharz, M. Frank, and P. Martini. Modelling voice communication in disaster area scenarios. In *Local Computer Networks, Proceedings 2006 31st IEEE Conference on*, pages 211–220, Nov. 2006.

[4] M. de Graaf, J. L. van den Berg, R. J. Boucherie, F. Brouwer, I. de Bruin, H. Elfrink, I. Fernandez-Diaz, S. M. H. de Groot, R. de Haan, J. de Jongh, S. Nunez, J. C. W. van Ommeren, F. Roijers, J. Stemerdink, and E. Tromp. Easy wireless: broadband ad-hoc networking for emergency services. In *The Sixth Annual Mediterranean Ad Hoc Networking Workshop, Corfu, Greece*, Corfu, Greece, March 2007. Ionian University.

[5] B. Garbinato, A. Holzer, and F. Vessaz. Six-shot broadcast: A context-aware algorithm for efficient message diffusion in manets. In R. Meersman and Z. Tari, editors, *OTM Conferences (1)*, volume 5331 of *LNCS*, pages 625–638. Springer, 2008.

[6] D. Gavidia and M. van Steen. A probabilistic replication and storage scheme for large wireless networks of small devices. In *5th IEEE International Conference on Mobile Ad Hoc and Sensor Systems (MASS)*, pages 469–476, 29 2008-Oct. 2 2008.

[7] D. Gelernter. Generative communication in linda. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 7(1):80–112, 1985.

[8] W. Guo and X. Huang. Mobility model and relay management for disaster area wireless networks. In *Wireless Algorithms, Systems, and Applications*, volume 5258 of *Lecture Notes in Computer Science*, pages 274–285. Springer Berlin / Heidelberg, 2008.

[9] Z. J. Haas, J. Y. Halpern, and L. Li. Gossip-based ad hoc routing. *IEEE/ACM Transactions on Networking*, 14(3):479–491, 2006.

[10] H.-C. Jang, Y.-N. Lien, and T.-C. Tsai. Rescue information system for earthquake disasters based on manet emergency communication platform. In *IWCMC '09: Proceedings of the 2009 International Conference on Wireless Communications and Mobile Computing*, pages 623–627, New York, NY, USA, 2009. ACM.

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

[12] M. Mamei and F. Zambonelli. Programming pervasive and mobile computing applications with the tota middleware. In *Proceedings of the Second IEEE International Conference on Pervasive Computing and Communications (PerCom)*, page 263, Washington, DC, USA, 2004. IEEE Computer Society.

[13] H. Miranda, S. Leggio, L. Rodrigues, and K. Raatikainen. A power-aware broadcasting algorithm. In *Personal, Indoor and Mobile Radio Communications, 2006 IEEE 17th International Symposium on*, pages 1–5, Sept. 2006.

[14] A. L. Murphy, G. P. Picco, and G.-C. Roman. Lime: A coordination model and middleware supporting mobility of hosts and agents. *ACM Transactions on Software Engineering and Methodology*, 15(3):279–328, 2006.

[15] O. Ozkasap, Z. Genc, and E. Atsan. Epidemic-based reliable and adaptive multicast for mobile ad hoc networks. *Comput. Netw.*, 53(9):1409–1430, 2009.

[16] Y.-C. Tseng, S.-Y. Ni, Y.-S. Chen, and J.-P. Sheu. The broadcast storm problem in a mobile ad hoc network. *Wireless Networking*, 8(2/3):153–167, 2002.

[17] G. Zussman and A. Segall. Energy efficient routing in ad hoc disaster recovery networks. In *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies*, volume 1, pages 682–691 vol.1, March-3 April 2003.