Spotcast – A Communication Abstraction for Proximity-Based Mobile Applications

Behnaz Bostanipour Benoît Garbinato Adrian Holzer

Distributed Object Programming Laboratory
University of Lausanne
CH-1015 Lausanne, Switzerland
Email: {behnaz.bostanipour,benoit.garbinato,adrian.holzer}@unil.ch

***Abstract*—We introduce *spotcast*, a new communication abstraction specifically aimed at the development of mobile proximity-based applications running in mobile ad hoc networks (MANETs). Our motivation lies in the fact that traditional communication abstractions, typically broadcast primitives with strong consistency guarantees, do not adequately capture the intrinsic *here-and-now* nature of such applications. Rather, developers need a communication abstraction offering the notion of *proximity-based diffusion* and some level of *message durability*, which is precisely what spotcast provides. We illustrate how spotcast can be used to implement mobile applications and we shortly discuss the correctness and the implementability of the spotcast abstraction in MANETs.**

***Keywords*-Proximity-Based Broadcast; MANET; Distributed Systems;**

I. Distributed Systems: Evolution or Revolution?

The importance of distributed systems has grown dramatically in the past 20 years, due to the exponential increase in the number of computers of all sizes interconnected via the Internet. Driven by this growth, distributed applications broke free from intranets where they tended to be strictly confined until the mid-nineties. As a natural consequence, they became ubiquitous, as testified by today's omnipresence of communication-oriented applications in our daily lives, typically running on mobile devices such as smartphones.

A. First Evolution

Examples of this evolution are the strong interests for peer-to-peer file sharing in the late nineties and the growing interest for cloud computing today. To address the evolving challenges posed by such open and potentially large-scale distributed systems, the research community has been constantly adapting the same communication abstractions it originally devised for intranet-oriented applications (database applications, n-tier services, etc). Such communication abstractions typically include various flavors of reliable broadcast and multicast [14], as well as different variants of consensus [23], [10], [5].

B. Then Paradigm Shift

When looking at how computer systems are being used today, particularly in the *mobile arena*, it is becoming more and more obvious that the evolution of distributed system is now reaching a breaking point. That is, deep changes are no longer occurring along the line of *how we do things* in distributed systems (client/server, peer-to-peer, cloud computing, etc.) but rather along the line of *what we do* with distributed systems (mobile online gaming, location-based services, mobile multimedia and interactive entertainment, etc.).

A good example of this ongoing paradigm shift can be found in the exploding number of social networking applications available on mobile devices: almost 10,000 such applications can be found in Apple's AppStore today,[1] and probably as many in the corresponding Android market. When put in perspective, mobile social networking is a particular case of an even larger set of mobile distributed applications known as *proximity-based mobile applications*. Contrary to typical distributed applications that flourished in the past two decades, this new blend of mobile applications tend to exhibit an intrinsic and somehow elastic *here-and-now* nature, as illustrated hereafter with concrete examples.

1) On-the-spot Survey: The notion of *on-the-spot survey*, proposed by mobile applications such as SpotMe (www.spotme.com), iSurvey (www.isurveysoft.com), or Voxco Mobile (www.voxco.com), is a good example of mobile proximity-based applications. Here the idea is for a user to ask questions to other nearby users, e.g., a teacher surveying students in the classroom, a speaker surveying customers at a marketing event, etc. This application requires a rather strict *here-and-now* semantics, since it is important for the surveyor to get feedback from the audience in a timely manner.

2) Social Radar: The concept of *social radar*, proposed by various mobile applications such as FourSquare (www.foursquare.com), FriendThem (www.friendthem.com) or

[1]March 2012.

Blendr (www.blendr.com), is another example of proximity-based mobile application. Intuitively, a social radar provides mobile device users with the ability to spot other users around them. This application can live with a somehow more elastic *here-and-now* semantics, since it is sufficient that users staying close enough for long enough eventually detect each other.

3) Mobile Photo Sharing: The idea behind *mobile photo sharing*, proposed by applications such as Instagram (www.instagram.com), Streamzoo (www.streamzoo.com), or Picplz (www.picplz.com), is to allow users participating in some social event to take photos with their mobile devices and to share them with other users present at that event. This application exhibits a rather loose *here-and-now* semantics: even if it might not be possible for two users to share all their photos while at the event, e.g., because one has to leave earlier, as soon as they get together again (possibly long after the event), the sharing can resume.

C. Spotcast: A New Communication Abstraction

Traditional communication abstractions, such as atomic broadcast or consensus, were devised long before mobile proximity-based applications became ubiquitous. For this reason, we advocate that this new blend of distributed applications demands a new communication paradigm, i.e., one that appropriately captures their here-and-now nature. To address this need, this paper introduces *spotcast*, a new communication abstraction that enables a mobile entity to disseminate a message in a defined range around it (here) for a defined duration (now). To be more precise, we actually present three variants of the spotcast abstraction, with slightly different delivery guarantees, in order to support various communication semantics required by mobile proximity-based applications.

D. Roadmap

This paper is organized as follows. In Section II, we describe our system model. In Section III, we introduce our novel spotcast communication abstraction. In Section IV, we present the implementation of spotcast, together with a discussion on its implementability. In Section V, we discuss the implementability of the underlying scoped broadcast service. Finally, we discuss related work in Section VI before concluding in Section VII with a perspective on future work.

II. System Model

We consider a mobile ad-hoc network (MANET) consisting of a finite set of n processes $P = \{p_1, p_2, ..., p_n\}$. We use the terms *process* and *node* interchangeably. Processes are in a two-dimensional plane. Each process has a unique identifier. Processes are *mobile* i.e., their geographic location can change unpredictably over time. Processes can experience *crash* failures. A crash faulty process stops prematurely. Prior to stopping, it behaves correctly.[2]

Processes exchange messages over a wireless radio network and all processes have the same radio transmission range. We assume the existence of a discrete global clock, i.e., the range T of the clock's ticks is the set of non-negative integers. We also assume the existence of a known bound on the relative speed of processes in the system. Finally, let p_i, p_j be two processes in P, we introduce the definitions given hereafter in order to capture proximity-based semantics.

- $loc_i(t)$ denotes the geographic location of mobile process p_i at time $t \in T$.
- $Z_i(\Delta_r, t)$ denotes all the points inside or on the circle centered at $loc_i(t)$ with radius Δ_r. We call $Z_i(\Delta_r, t)$, the *mobile circular zone* of radius Δ_r around p_i. Thus, we use the term *mobile circular zone* to capture the notion of *neighborhood*.
- We say p_j is *in* $Z_i(\Delta_r, t)$ if $loc_j(t) \in Z_i(\Delta_r, t)$.
- We say p_j *follows* p_i *within a radius* Δ_r *during a time interval* $[t_1, t_2]$, with $t_1, t_2 \in T$ *and* $t_1 \leq t_2$, if p_j remains permanently in the mobile circular zone of radius Δ_r around p_i during the time interval. This can be expressed as follows:

$$\forall t \in [t_1, t_2] : loc_j(t) \in Z_i(\Delta_r, t) \quad (1)$$

  If $t_1 = t_2$, stating that p_j *follows* p_i *within a radius* Δ_r *during a time interval* $[t_1, t_2]$ is equivalent to say that $loc_j(t) \in Z_i(\Delta_r, t)$ at time $t = t_1 = t_2$.
- We say p_j *follows* p_i *within a radius* Δ_r *after time* t, if p_j remains permanently in the mobile circular zone of radius Δ_r around p_i after time $t \in T$. This can be expressed as follows:

$$\forall t' \in T : t' \geq t \Rightarrow loc_j(t') \in Z_i(\Delta_r, t') \quad (2)$$

- We say p_j *eventually follows* p_i *within a radius* Δ_r, if there exists a time after which p_j remains permanently in the mobile circular zone of radius Δ_r around p_i. This can be expressed as follows:

$$\exists t \in T : \forall t' \in T : t' \geq t \Rightarrow loc_j(t') \in Z_i(\Delta_r, t') \quad (3)$$

III. The Spotcast Abstraction

In this section, we introduce the *spotcast* communication abstraction in three variants. Intuitively, by using the spotcast service a mobile process can disseminate a message for a given time to all mobile processes located in its proximity. The name "spotcast" is chosen by analogy with a spotlight

[2]Since we do not consider Byzantine behaviors, issues related to information security and privacy are beyond the scope of this paper.

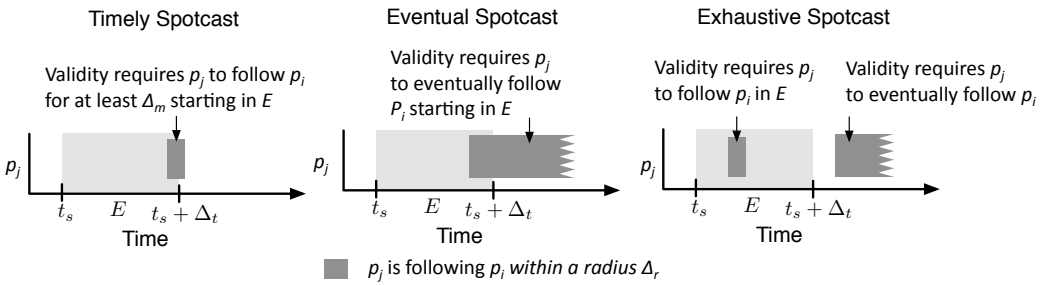

Figure 1: Validity Properties of Spotcast Variants.

following the source of the message (the spotcaster). Formally, the spotcast service supports the following functional interface:

- SPOTCAST(m, Δ_r, Δ_t): disseminates a message m in $Z_i(\Delta_r, t)$ for all t $\in E = [t_s, t_s + \Delta_t]$, where p_i is the process that invokes the spotcast and t_s is the time when the spotcast is invoked.
- DELIVER(m, p_i): callback delivering a message m spotcast by process p_i.

The time interval E is called the *spotcast epoch*. All spotcast variants share a set of properties called the *core spotcast properties* listed hereafter.

A. Core Spotcast Properties

Non-triviality. If some process p_j delivers a message m with sender p_i, then there exists a time t_E in E such that $loc_j(t_E) \in Z_i(\Delta_r, t_E)$.

No Duplication. No message is delivered more than once.

No Creation. If some process p_j delivers a message m with sender p_i, then m was previously spotcast by process p_i.

B. Spotcast Variants and their Validity Properties

In addition to the core properties, each spotcast variant satisfies a specific *validity* property. Hereafter, we present each variant and its *validity* property, together with an example of its usage (see Figure 1).

1) Timely Spotcast: This variant provides the strictest *here-and-now* semantics among the three spotcast variants. Intuitively, it guarantees that there exists a time limit after which a message is delivered to all processes in the neighborhood of the process who spotcast that message. The corresponding formal validity property is given hereafter.

Validity. If a correct process p_i spotcasts a message m, there exists a bounded time duration Δ_m such that every correct process p_j delivers m in E'=$[t_s, t_s+\Delta_t+\Delta_m]$, if p_j follows p_i within radius Δ_r during $[t_E, t_E+\Delta_m]$ with t_E in E.

The time interval E'=$[t_s, t_s + \Delta_t + \Delta_m]$ is called *the delivery epoch*. The on-the-spot survey application discussed in Section I is an example of Timely Spotcast usage: the person launching such a survey is interested in disseminating questions to the audience in a timely manner.

2) Eventual Spotcast: This variant provides a somehow more elastic *here-and-now* semantics. Intuitively, it guarantees that any process staying long enough in the neighborhood of some process who is spotcasting a message, will eventually receive that message. The corresponding formal validity property is given hereafter.

Validity. If a correct process p_i spotcasts a message m, every correct process p_j eventually delivers m, if there exists a time t_E in E after which p_j follows p_i within the radius Δ_r.

The social radar application discussed in Section I is an example of Eventual Spotcast usage: each user regularly spotcasts her location, so that users in her neighborhood will eventually learn about her nearby presence.

3) Exhaustive Spotcast: Finally, this variant provides the most elastic *here-and-now* semantics. Intuitively, it guarantees that any process who was in the neighborhood of some other process when the latter is spotcasting a message, will eventually receive that message, provided both processes eventually get the chance to stay close enough for long enough. The corresponding formal validity property is given hereafter.

Validity. If a correct process p_i spotcasts a message m, every correct process p_j eventually delivers m, if there exists a time t_E in E when $loc_j(t_E) \in Z_i(\Delta_r, t_E)$ and also p_j eventually follows p_i within the radius Δ_r.

The mobile photo sharing application discussed in Section I is an example of Exhaustive Spotcast usage: consider Alice, who is spotcasting a URL pointing to the photos she is taking while at a party. The exhaustive delivery property allows another participant in that party, say Bob, to receive the URL, even if Alice has to leave before Bob can actually receive the URL. The delivery will occur as soon as Alice and Bob get together again, possibly after the party, and stay in the neighborhood of each other long enough.

IV. A SPOTCAST ALGORITHM

We provide an architecture overview of our algorithm for the spotcast communication abstraction in Figure 2. At the top, a mobile proximity-based application uses the spotcast service, which itself relies on two lower-level services for its implementation, namely a *global positioning service* and a *scoped broadcast service*.

A. Global Positioning Service

This service allows each mobile process p_i to know its position in space and time, via the following functions:[3]

- GETTIME: returns the current global time. Formally, this implies that each process p_i has access to the global clock modeled in Section II.
- GETLOCATIONS(t_1, t_2): returns the set of locations occupied by p_i during time interval $[t_1, t_2]$. If $t_1 <$ GETTIME and $t_2 >$ GETTIME, only locations occupied during time interval $[t_1, \text{GETTIME}]$ are returned. If $t_1 >$ GETTIME or if $t_1 > t_2$, no location is returned.

B. Scoped Broadcast Service

This communication service allows processes to send messages to all processes located within a given radius. Formally, the scoped broadcast service exposes the following primitives:

- BROADCAST(m, Δ_r): broadcasts a message m *in* $Z_i(\Delta_r, t_b)$, where p_i is the sender and t_b is the time when the broadcast is invoked.
- RECEIVE(m, p_i): callback delivering a message m broadcast by process p_i.

In terms of safety, the scoped broadcast service satisfies the *no duplication* property and the *no creation* property given hereafter.

> ***No Duplication.*** No message is delivered more than once.
>
> ***No Creation.*** If some process p_j delivers a message m with sender p_i, then m was previously broadcast by process p_i.

[3]In practice, such a service would typically be implemented using NASA's GPS or ESA's Galileo space-based satellite navigation technologies.

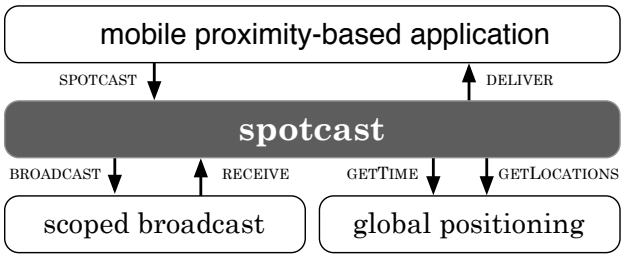

Figure 2: Spotcast – Architecture Overview.

When it comes to liveness, we consider two alternatives of the delivery property, namely *fair-loss delivery* and *timely delivery*; these two alternatives are presented below. As discussed in Section IV-D, the implementability of the various flavors of spotcast closely depends on which delivery property is being considered.

> ***Fair-Loss Delivery.*** If a correct process p_i broadcasts a message m an infinite number of times, any correct process p_j eventually delivers m, if p_j eventually follows p_i within the radius Δ_r.
>
> ***Timely Delivery.*** If a correct process p_i broadcasts a message *m*, there exists a bounded time duration Δ_{bcast} such that every correct process p_j delivers m in interval $[t_b, t_b + \Delta_{bcast}]$, if p_j follows p_i within the radius Δ_r during $[t_b, t_b + \Delta_{bcast}]$.

Note that it is beyond the scope of this paper to provide an implementation of scoped broadcast service. However, we discuss its implementability in Section V.

C. Generic Spotcast Algorithm

Algorithm 1 presents our implementation of the spostcast communication abstraction. It is *generic* in the sense that it serves as basis for implementing any of the three variants introduced in Section III. Its genericity is captured by function ISBOUNDEDBUFFER and by procedure BROADCAST. That is, depending on how they are defined, Algorithm 1 implements a specific variant of spotcast. Specifically, BROADCAST relies on an implementation of the Scoped Broadcast Service ensuring either *fair-loss delivery* or the *timely delivery*, while ISBOUNDEDBUFFER returns $true$ if an implementation with bounded buffer is possible, $false$ otherwise. Algorithm 1 can be divided in three parts, namely *initialization*, *spotcasting* and *delivering*, which are discussed hereafter.

1) Initialization: (lines 3-5). The algorithm relies on two message sets: *msgset* for storing messages to be broadcast and *delivered* for storing the delivered spotcast messages.

2) Spotcasting: (lines 6-17). Spotcasting entails a setup and a running phase. The setup phase (lines 6-13) starts when SPOTCAST primitive is called with following parameters: a message m to be spotcast in a radius Δ_r during a time duration Δ_t. This call creates a *msg* that encapsulates m and some other parameters. Among these parameters the hash map *locs* is used to store the location of the sender at each time t in the spotcast epoch. The *msg* is then added to the set *msgset*. The running phase (lines 14-17) periodically goes through each *msg* in *msgset*. Thus, all the locations of the process during the spotcast epoch are first assigned to the *msg*'s *locs*. Then, *msg* is broadcast within the *msg*'s radius using the scoped broadcast service's BROADCAST primitive. After the broadcast, if it is possible (line 17), *msg* is removed from *msgset*.

Algorithm 1 Generic Spotcast Algorithm at Process p_i

```
1:  implements: SPOTCAST
2:  uses: BROADCAST, ISBOUNDEDBUFFER                                   {generic procedure/function}

3:  initialisation:
4:      msgset ← ∅
5:      delivered ← ∅

6:  upon SPOTCAST(m, Δ_r, Δ_t) do
7:      msg ← ⊥                                  {creates msg to encapsulate m plus some additional parameters}
8:      msg.m ← m                                                                          {assigns m}
9:      msg.Δ_r ← Δ_r                                                                      {assigns the radius}
10:     msg.t_start ← GETTIME                                                              {assigns the start of the spotcast epoch}
11:     msg.t_end ← msg.t_start + Δ_t                                                      {assigns the end of the spotcast epoch}
12:     msg.locs ← [ ]                  {locs is a hash map to store (t, loc_i(t)) for all t in the spotcast epoch}
13:     msgset ← msgset ∪ {msg}                                                            {adds msg to msgset}

14: do every Δ_period for each msg in msgset                                               {broadcasts periodically}
15:     msg.locs ← GETLOCATIONS(msg.t_start, msg.t_end)
16:     trigger BROADCAST(msg, msg.Δ_r)
17:     if ISBOUNDEDBUFFER ∧ GETTIME > msg.t_end then
18:         msgset ← msgset \ {msg}                                                        {removes msg from the msgset}

19: upon RECEIVE(msg, p_j) do
20:     if LOCATIONMATCH(msg) ∧ msg.m ∉ delivered then
21:         trigger DELIVER(msg.m, p_j)
22:         delivered ← delivered ∪ {msg.m}                                                {adds m to delivered messages}

23: function LOCATIONMATCH(msg) returns boolean is
24:     for all t ∈ [msg.t_start, msg.t_end] do
25:         if DISTANCE(GETLOCATIONS(t, t), msg.locs.get(t)) ≤ msg.Δ_r then
26:             return true                                  {returns true if a location match occurred during the epoch}
27:     return false
```

3) Delivering: (lines 19-22). When a broadcast *msg* is received, RECEIVE callback of the underlying scoped broadcast is triggered. Then m of *msg* is delivered by the sptocast service via the DELIVER callback if it has not been already delivered and if the receiving process meets the requirement of a location match. This is done by calling the LOCATIONMATCH function. For a location match to occur (lines 23-27), the distance between the sender and the receiver must have been less than or equal to the $msg.\Delta_r$ at least for some time during the spotcast epoch.

D. Correctness and Implementability

The *no creation* property of spotcast follows directly from the *no creation* property of the underlying scoped broadcast service. As for the *no duplication* property of spotcast, it follows from the *no duplication* property of scoped broadcast service and from the management of the *delivered* set (lines 19 to 22). The *non-triviality* property of spotcast is ensured by calling the function LOCATIONMATCH (lines 23 to 27) and delivering the message if and only if a location match occurs. Besides these three core properties, each spotcast variant comes with a distinct *validity* property. We separately discuss their respective correctness and implementability hereafter. Table I summarizes this discussion.

1) Validity of Timely Spotcast: Algorithm 1 implements Timely Spotcast if generic procedure BROADCAST guarantees *timely delivery*. Indeed, recall that a timely spotcast message m must be delivered in the *delivery epoch* E'=$[t_s, t_s + \Delta_t + \Delta_m]$, with Δ_m bounded. Note also that we can express $\Delta_m = \Delta_{period} + \Delta_{exec} + \Delta_{bcast}$, where Δ_{exec} is the execution time of (lines 14-17), and Δ_{bcast} is the communication delay introduced by the underlying scoped broadcast. Δ_{period} is a constant. Since there exists a known upper bound on the relative processing speed in the system, we already know that Δ_{exec} is bounded. So for

Table I: Spotcast – Correctness and Implementability.

Spotcast	**Scoped Broadcast**	
	fair-loss delivery	*timely delivery*
timely	not implementable	implementable with bounded buffer
eventual	implementable with unbounded buffer	implementable with bounded buffer
exhaustive	implementable with unbounded buffer	implementable with unbounded buffer

Δ_m to be bounded, we only need to be sure that Δ_{bcast} is also bounded. This is precisely what the timely delivery property guarantees. Now, since delivery is required only during bounded delivery epoch E', a bounded message buffer is sufficient, i.e., function ISBOUNDEDBUFFER can return $true$, allowing *msgset* to be purged (line 18). On the other hand, Algorithm 1 can not implement Timely Spotcast if the procedure BROADCAST guarantees *fair-loss delivery*. The reason is that the fair-loss delivery property only guarantees an eventual message delivery.

2) Validity of Eventual Spotcast: Algorithm 1 implements Eventual Spotcast if the generic procedure BROADCAST guarantees *fair-loss delivery* and if ISBOUNDEDBUFFER returns *false* (hence requires unbounded buffer). Indeed, Eventual Spotcast guarantees an eventual message delivery. This is offered by the fair-loss delivery property of the underlying scoped broadcast. Moreover, since fair-loss delivery requires the message to be broadcast infinitely often, an unbounded message buffer is necessary.

Algorithm 1 also implements Eventual Spotcast in the case that the generic procedure BROADCAST guarantees *timely delivery*. In this case however, a bounded message buffer is sufficient. Indeed, recall that the *validity* of Eventual Spotcast guarantees the message delivery to a correct p_j which follows the spotcaster p_i within the radius Δ_r after t_E. This means that the latest possible time for p_j to start to follow p_i is $t_E = t_s + \Delta_t$ (end of the spotcast epoch). According to the algorithm, if function ISBOUNDEDBUFFER returns *true*, the message *msg* containing the spotcast message m is broadcast for the last time after the termination of the *spotcast epoch* (lines 14 to 17). The timely delivery property guarantees the delivery of this last broadcast of *msg* to any p_j following p_i after t_E even for $t_E = t_s + \Delta_t$. This means that *msg* can be safely removed from *msgset* after its last broadcast which implies using a bounded buffer.

3) Validity of Exhaustive Spotcast: Algorithm 1 implements Exhaustive Spotcast if the generic procedure BROADCAST guarantees *fair-loss delivery* or *timely delivery* and if ISBOUNDEDBUFFER returns *false* (hence requires unbounded buffer). Indeed, recall that the *validity* property of Exhaustive Spotcast guarantees an eventual message delivery to all correct processes which were within the radius Δ_r for some time t_E in the *spotcast epoch*, and which eventually follow the spotcaster process p_i within the radius Δ_r. Since the time when a process starts to eventually follow p_i is unknown, p_i should continue to broadcast the message infinitely often. This means that regardless of the underlying scoped broadcast service, Algorithm 1 can only implement Exhaustive Spotcast using an unbounded message buffer.

V. IMPLEMENTABILITY OF SCOPED BROADCAST SERVICE

The correctness of our spotcast algorithm relies on the existence of the scoped broadcast service, whose implementability in turn depends on the properties of the underlying MAC layer. So, in order to discuss the implementability of the scoped broadcast service over wireless ad hoc networks, we rely on the MAC layer models described in [8] and [18], which have been shown to realistically model the 802.11 MAC layer [1]. Since both these MAC layer models guarantee that no message is created or duplicated, the *no duplication* and the *no creation* properties of our scoped broadcast service are trivially ensured. So in the following, we focus on the implementability of the *delivery* property of each scoped broadcast variant. That is, we use the model described in [8] to show the implementability of the *fair-loss delivery* property, while the model described in [18] is used to show the implementability of the *timely delivery* property. For each delivery property, the discussion distinguishes two cases: (1) the *single-hop* case, where the radio transmission range is greater than or equal to the Δ_r radius of the scoped broadcast, and (2) the *multi-hop* case, where the radio transmission range is smaller than Δ_r.

A. Implementability of the Fair-Loss Delivery Property

In [8], the MAC Layer model assumes that the communication medium is prone to collisions but guarantees *an eventual collision freedom* property. In addition, node clock skews and inter-node communication delays are assumed to be bounded by known constants. Processing is then conceptually divided into synchronous rounds and at each round each node broadcasts at most one message. Nodes can fail by crashing but cannot crash while broadcasting. A node that does not crash throughout an entire execution is said to be *correct*.

1) Single-hop Case: The *fair-loss delivery* property guarantees an *eventual* delivery of a message if it is broadcast infinitely often. In the single-hop case, one way to achieve the *fair-loss delivery* property is to assume that eventually the communication medium becomes collision-free.[4] This is precisely the property ensured by the *eventual collision freedom* property of the considered MAC layer. According to this property, there exists a positive integer b, such that in each execution, there exists a round r_{ecf}, so that for every round $r \geq r_{ecf}$, if at most b nodes broadcast in r, then every message broadcast in r is reliably delivered by all correct nodes. To eventually reach a round in which at most b nodes broadcast, a *wake-up service* is used. This service reduces contention by determining which nodes should broadcast in a given round. The wake-up service eventually stabilizes, i.e., it eventually recommends that at least one, and no more than b, correct nodes can broadcast in a round. Thus, in executions which satisfy the *eventual collision freedom* property, this allows the messages to be delivered without collision. The wake-up service can be implemented by using

[4]We assume that collisions are the major source of message losses.

a simple approximation of a well-known back-off strategy [12], [32], [31], [35].

2) Multi-hop Case: When the radio transmission range is smaller than the radius Δ_r of the scoped broadcast, a scoped broadcast protocol can guarantee the *fair-loss delivery* property if the two following conditions are satisfied: a) the underlying MAC layer satisfies the *fair-loss delivery* property and b) a route is established between each correct node p_i and any other correct node p_j which is in radius Δ_r around p_i. As just discussed, the MAC layer modeled in [8] satisfies the *fair-loss delivery* property, so Condition a) can be fulfilled by using such a MAC layer. For Condition b) to hold, the scoped broadcast can use for instance a *proactive* routing protocol (e.g., [33]) as sub-protocol, provided of course that there exists a path between p_i and p_j where the distance between any two direct neighbors along the path does not exceed the transmission range.

B. Implementability of the Timely Delivery Property

To discuss the implementability of the *timely delivery* property, we rely on the MAC layer modeled in [18]. Intuitively, this MAC layer provides the ability to reliably broadcast messages and to receive acknowledgments when those messages have been successfully delivered to all nearby nodes, *with timing guarantees*. In practice, such a MAC layer is implementable with very high probability[5], by using contention-management mechanisms such as carrier sensing and back-off [1], receiver-side collision detection with negative acknowledgment [9], or network coding methods [13].

1) Single-hop Case: An implementation of the *timely delivery* property in the single-hop case must guarantee a finite upper bound Δ_{bcast} on the duration between the broadcast and the delivery of a message, provided the receiver remains in the transmission range of the sender during Δ_{bcast}. This can be ensured thanks to the *guaranteed communication* property of the discussed MAC layer. This property can be stated as follows: let process p_i be the sender of a message m, then any process p_j which remains in the communication range of p_i for all time between the broadcast and the acknowledgment of m, receives m before the acknowledgment of m at p_i. In addition, the *guaranteed communication* property comes with a timing guarantee on m's reception, defined as function F^+_{ack}. This function expresses an upper bound on the elapsed time between the broadcast of m and the reception of its acknowledgment at sender p_i in the worst case, which corresponds to the maximum contention level. The contention level is defined as the number of distinct senders in the neighborhood of m's receivers and m's sender, and whose broadcast overlaps the broadcast and acknowledgment of m. Thus, assuming $\Delta_{bcast} = F^+_{ack}$, the *timely delivery* property is trivially ensured by the considered MAC layer for the single-hop case.

2) Multi-hop Case: When the radio transmission range is smaller than the radius Δ_r of the scoped broadcast, a scoped broadcast protocol can guarantee the *timely delivery* property if the two following conditions are satisfied: a) the underlying MAC layer satisfies the *timely delivery* property and b) a route is established between each correct node p_i and any other correct node p_j which is in radius Δ_r around p_i. As just discussed, the MAC layer modeled in [18] satisfies the *timely delivery* property for the single-hop case with $\Delta_{bcast} = F^+_{ack}$, so Condition a) is fulfilled. For Condition b), we can follow the same reasoning we did in Section V-A2 for the *fair-loss delivery* property: to find a route between any pair of correct nodes p_i and p_j, provided there exists a path between p_i and p_j. In addition, we must show that the routing of any message is time bounded. For this, let $d(i, j)$ be the length of the shortest path between any two pairs of correct nodes p_i and p_j, and let k be the maximum of such $d(i, j)$ lengths. By observing that Δ_{bcast} = $O(k \times F^+_{ack})$ for the considered MAC layer, we can say that the *timely delivery* property can be ensured for the multi-hop case.

VI. Related Work

Among location-aware or time-aware communication services proposed for mobile ad hoc networks, *geocast* and *mobicast* are the closest to our work. In this section, we discuss these services and compare them to spotcast. We then compare spotcast to higher level services such as *location-based publish/subscribe*.

A. Geocast

In a geocast routing protocol, a message is disseminated to all nodes which are within a given geographic area called the *geocast region*. Thus, geocast is a type of multicast in which group membership is defined with respect to a geographic area. Geocast was initially proposed for the Internet [19]. Then, various geocast routing protocols were proposed for ad hoc networks [21], [22], [4], [25], [24], [29], [28], [3] and in particular, for vehicular ad hoc networks (VANETs) [2], [20], [34], [26], [27]. The classical geocast routing is semantically time-oblivious i.e., the geocast message is assumed to be delivered as soon as possible. In *abiding geocast* [29], the geocast message is disseminated in the geocast region for a time duration. Some papers use abiding geocast to disseminate traffic warning messages or commercial advertisement to a group of vehicles in a given zone for a given time [34], [26]. There exist several differences between geocast and spotcast, depending on the geocast variant. With spotcast, the zone in which the message is disseminated is a disk centered at the source of the message and that zone moves with the source of the message. With geocast on the contrary, the dissemination region is generally not associated with the

[5]In practice, synchronous assumptions are always probabilistic.

source of the message and remains stationary. Furthermore, the delivery guarantees of our different spotcast variants require the receiver to follow the source within some range for some amount of time, whereas such a requirement is absent from the geocast specification.

B. Mobicast

Mobicast is a class of multicast which is both location-aware and time-aware. In mobicast, a message is disseminated in an area called the *delivery zone* for a time duration T. Delivery zone can move during T and is denoted as $Z[t]$, where t is in T. As the delivery zone moves, some nodes enter the zone and some nodes leave the zone. The ultimate goal of mobicast is to achieve *just-in-time* message delivery, i.e., $Z[t]$ represents the area where the mobicast message should be delivered at time t [16]. Some mobicast protocols were proposed for wireless sensor networks [16], [6], [17] and VANETs [7]. In some of these protocols, the delivery zone is not associated to the source of the message e.g., in [16]. In [7], the delivery zone is an elliptic area around the source of the message (a moving car). Contrary to spotcast, mobicast does not require the nodes to follow the message source in order to guarantee the message delivery. Instead, mobicast protocols usually assume the existence of a *forwarding zone* to ensure the implementation of the strong *just-in-time* message delivery property.

C. Location-Based Publish/Subscribe

Some authors proposed high level communication abstractions derived from the publish/subscribe paradigm, which include some type of constraint on messages in time and/or space [15], [30], [11]. In STEAM [30], messages are constrained to a location around the sender. However, there is no support for persistence and the specifications and underlying communication abstractions are not detailed. In [11], messages are persisted and can be localized. However, they are assigned to a fixed geographical zone and do not move around with the sender. In [15], messages are persisted in a geographical zone around the publisher for a certain duration. The communication abstraction used in [15] to propagate messages is the closest to spotcast, however it does not provide any guarantees. This lack of guarantees is one of the main motivations for the present work.

VII. Conclusion

In this paper, we presented spotcast, a communication abstraction specifically devised for proximity-based mobile applications. Spotcast allows processes to send messages to peers located within a defined range in a defined time frame. We presented three variants of spotcast offering different levels of delivery guarantees. By proposing an interface and formal properties for a novel communication abstraction in MANETs, this paper aims at filling a current gap in communication support for emerging proximity-based mobile applications. However, several issues remain open, which we intend to address in future work. For instance, we will consider Byzantine processes in order to model security and privacy issues, which are inherent to proximity-based mobile applications.

Moreover, the partial requirement for unbounded buffers needs to be further addressed from a realistic, application-driven perspective. Another issue which could be investigated is the feasibility of spotcast variants, particularly exhaustive spotcast, based on real applications, the network size and the number of broadcasting nodes.

References

[1] IEEE 802.11: Wireless LAN MAC and Physical Layer Specifications. June (1999).

[2] Bachir, A., Benslimane, A.: A Multicast Protocol in Ad hoc Networks Inter-vehicle Geocast. In: Proceedings of IEEE Semiannual Vehicular Technology Conference, VTC'03, 4:2456–2460, IEEE (2003).

[3] Baldoni, R., Ioannidou, K., Milani, A.: Mobility Versus the Cost of Geocasting in Mobile Ad-hoc Networks. DISC, Lecture Notes in Computer Science, 4731:48-62 (2007).

[4] Boleng, J., Camp, T., Tolety, V.: Mesh-Based Geocast Routing Protocols in an Ad hoc Network. In: Proceedings of 15th International Parallel and Distributed Processing Symposium, IPDPS '01, pp. 1924–1933, IEEE Comput. Soc. (2001).

[5] Chandra, T. D., Toueg, S.: Unreliable Failure Detectors for Reliable Distributed Systems. JACM., 43:225–267 (1996).

[6] Chen, Y.S., Ann, S.Y., Lin, Y.W.: VE-Mobicast: A Variant-Egg-Based Mobicast Routing Protocol for Sensornet. ACM Wireless Networks., 14:199–218 (2008).

[7] Chen, Y.S., Lin, Y.W., Lee, S.L.: A Mobicast Routing Protocol in Vehicular Ad-hoc Networks. Mob. Netw. Appl., 15:20–35 (2010).

[8] Chockler, G., Demirbas, M., Gilbert, S., Newport, C., Nolte, T.: Consensus and Collision Detectors in Wireless Ad hoc Networks. In: Proceedings of the Annual ACM Symposium on Principles of Distributed Computing, PODC'05, pp. 197–206, ACM (2005).

[9] Chockler, G., Demirbas, M., Gilbert, S., Lynch, N., Newport, C., Nolte, T.: Consensus and Collision Detectors in Radio Networks. Distrib. Comput., 21, 55–84 (2008).

[10] Dwork, C., Lynch, N., Stockmeyer, L.: Consensus in the Presence of Partial Synchrony. JACM., 35:288–323 (1988).

[11] Frey, D., Roman, G.: Context-Aware Publish Subscribe in Mobile Ad hoc Networks. In: Proceedings of the 9th International Conference on Coordination Models and Languages, Coordination'07, pp. 37–55, Springer (2007).

[12] Gallager, R.: A perspective on Multiaccess Channels. IEEE Transactions on Information Theory, 31(2):124–142 (1985).

[13] Gollakota, S., Katabi, D.: ZigZag Decoding: Combating Hidden Terminals in Wireless Networks. In: Proceedings of the ACM SIGCOMM Conference (2008).

[14] Hadzilacos, V., Toueg, S.: Fault-Tolerant Broadcasts and Related Problems. pp. 97–145, ACM Press/Addison-Wesley Publishing Co., New York (1993).

[15] Holzer, A., Eugster, P., Garbinato, B.: Evaluating Implementation Strategies for Location-based Multicast Addressing. IEEE TMC (to appear) (2012).

[16] Huang, Q., Lu, C., Roman, G.: Mobicast : Just-in-Time Multicast for Sensor Networks under Spatiotemporal Constraints. In: Proceedings of the International Conference on Information Processing in Sensor Networks, IPSN'03, pp. 442–457 (2003).

[17] Huang, Q., Lu, C., Roman, G.: Reliable Mobicast via Face-Aware Routing. In: Proceedings of IEEE International Conference on Computer Communications, INFOCOM'04, pp. 205–217 (2004).

[18] Kuhn, F., Lynch, N., Newport, C.: The Abstract Mac Layer. Distributed Computing, 24(3-4):187–206 (2011).

[19] Imielinski, T., Navas, J.C.: Gps-Based Geographic Addressing, Routing, and Resource Discovery. Commun. ACM., 42:86–92 (1999).

[20] Joshi, H.P., Sichitiu, M., Kihl, M.: Distributed Robust Geocast Multicast Routing for Inter-vehicle Communication. In: Proceedings of Weird Workshop on WiMAX, Wireless and Mobility, Weird'07, pp. 9–21 (2007).

[21] Ko, Y.B., Vaidya, N.H.: Geocasting in Mobile Ad hoc Networks: Location-Based Multicast Algorithms. In: Proceedings of Second IEEE Workshop on Mobile Computing Systems and Applications, WMCSA'99, pp. 101–110, IEEE (1999).

[22] Ko, Y.B., Vaidya, N.H.: Geotora: a Protocol for Geocasting in Mobile Ad hoc Networks. In: Proceedings of International Conference on Network Protocols, ICNP'00, (00-010):240–250, IEEE Comput. Soc. (2000).

[23] Lamport, L., Shostak, R., Pease, M.: The Byzantine Generals Problem. ACM. TOPLAS., 4:382–401 (1982).

[24] Lee, S.H., Ko, Y.B.: Geometry-Driven Scheme for Geocast Routing in Mobile Ad Hoc Networks. In: Proceedings of the IEEE Vehicular Technology Conference, VTC'06, pp. 638–642, IEEE (2006).

[25] Liao, W. H., Tseng, Y.C., Lo, K.L. , Sheu, J. P.: GeoGRID: A Geocasting Protocol for Mobile Ad Hoc Networks Based on GRID. JIT., 1–2:23–32 (2000).

[26] Lim, Y., Ahn, S., Cho, K.H.: Abiding Geocast for Commercial Ad Dissemination in the Vehicular Ad hoc Network. In: Proceedings of IEEE International Conference on Consumer Electronics, ICCE'11, pp. 115–116 (2011).

[27] Lin, Y., Chen, Y., Lee, S.: Routing Protocols in Vehicular Ad hoc Networks: A Survey and Future Perspectives. J. Inf. Sci. Eng., 26(3):913–932 (2010).

[28] Maihöfer, C.: A Survey of Geocast Routing Protocols. IEEE Communications Surveys and Tutorials, 6(1-4):32–42 (2004).

[29] Maihöfer, C., Leinmüller, T., Schoch, E.: Abiding Geocast: Time–Stable Geocast for Ad hoc Networks. In: Proceedings of the 2nd ACM International Workshop on Vehicular Ad hoc Networks, VANET '05, pp. 20–29, ACM (2005).

[30] Meier, R., Cahill, V.: On Event-Based Middleware for Location-Aware Mobile Applications. IEEE TSE, 36(3):409–430 (2010).

[31] Nakano, K., Olariu, S.: Uniform Leader Election Protocols in Radio Networks. In: Proceedings of the International Conference on Parallel Processing, ICPP'01, pp. 240–250. IEEE Computer Soc. (2001).

[32] Olariu, S., Nakano, K.: A survey on Leader Election Protocols for Radio Networks. In: Proceedings of the International Symposium on Parallel Architectures, Algorithms and Networks, ISPAN'02, pp. 71–79, IEEE Computer Soc. (2002).

[33] Perkins, C.E., Bhagwat, P.: Highly dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers. In: Proceedings of ACM Conference of the Special Interest Group on Data Communication (SIGCOMM'94), vol. 24, no.4, pp.234-244, (1994).

[34] Yu, Q., Heijenk, G.: Abiding Geocast for Warning Message Dissemination in Vehicular Ad hoc Networks. In: Proceedings of International Conference On Communications, ICC'08, pp. 400–404 (2008).

[35] Willard, D. E.: Log-logarithmic Selection Resolution Protocols in a Multiple Access Channel. SIAM Journal of Computing, 15(2):468–477 (1986).