Original Research Paper

SocialMANET: A Publish/Subscribe Events Dissemination Protocol for Mobile Ad-Hoc Networks

[1,2]Maurice Tchoupé Tchendji, [1]Martin Xavier Tchembé and [1]Innocent Tialo Necheu

[1]Department of Mathematics and Computer Science, University of Dschang, Dschang, Cameroun
[2]Lirima - Fuchsia team lirima.inria.fr/project.inria.fr/fuchsia

Article history
Received:13-04-2019
Revised: 18-08-2019
Accepted: 30-08-2019

Corresponding Author:
Maurice Tchoupé Tchendji
Department of Mathematics and Computer Science,
University of Dschang,
Dschang, Cameroun
Tel: +237699663341
Email: ttchoupe@yahoo.fr

Abstract: The popularization of mobile equipment capable of communicating autonomously nowadays offers the possibility of thinking more seriously about a mode of communication in which stakeholders communicate in an ad-hoc manner and use mobility to convey information. However, due to their mobile nature, such equipments are limited by their low energy and storage capacity. Algorithms designed for them must therefore be able to deal with these limitations. In this paper, we present SocialMANET, a distributed publish/subscribe events dissemination protocol for mobile ad-hoc networks. It implements the hierarchical subjects subscription model which, in addition to allowing to formulate refined subscriptions requests close to the content-based subscription model, also permits to minimize the number of subscriptions required to receive events in several topics: to this end, we have proposed a strategy for generating subject identifiers based on the *dynamic level numbering* technique. Through the use of a dedicated message format that we have defined, SocialMANET minimizes the number of exchanges required between terminals during (fugitive) contacts to detect and respond to their needs by making the best use of the information extracted from these exchanges. This message format, coupled with the concept of *lazily altruistic nodes* that we propose, allows SocialMANET to guarantee a high rate of events receipt, even in situations where the interests of the network nodes tend to diverge: the results of the simulations conducted show that, compared to other dissemination protocols of the literature, SocialMANET guarantees a better rate of receiving event while minimizing the power consumption of network nodes. We also present in this paper an application prototype for the dissemination of academic informations within a community of students; it's based on an implementation of SocialMANET.

Keywords: Publish/Subscribe, Mobile Ad-hoc Network, Lazy Altruistic Sharing, Energy, Events Dissemination, Structured Message

Introduction

The ever-increasing number and power of the types of interconnected terminals (mobile phones, PDAs, computers, etc.) used today for various needs (communication, work, leisure, etc.), strongly influences the lifestyle of users of these tools and especially the way they use them to communicate. They nevertheless have a permanent concern: that the exchanged data (files, images, videos, etc.) are always available (availability constraint) without any restrictions on their freedom of movement (mobility constraint).

Mobile ad-hoc networks (MANET) offer a response to the mobility constraint because by their very nature, they eliminate the need for fixed infrastructures to communicate. However, the mobile nature of the stations in such networks requires that they have limited capacities (computing power, energy, storage, etc.); therefore, the problem of data availability cannot be appropriately addressed by implementing a traditional client-server communication protocol. It is indeed inappropriate to ask one of the stations to play the role of a server without running the risk that it will quickly be inaccessible either because it has exhausted its energy or because of mobility. Hence it is no longer within the reach of the other stations.

Broadcast communications coupled with the publish/subscribe events dissemination paradigm seem to be better suited in this context of mobility, limited resource capacity and information sharing. Indeed, the dissemination of events by publish/subscribe offers a high

© 2019 Maurice Tchoupé Tchendji, Martin Xavier Tchembé and Innocent Tialo Necheu. This open access article is distributed under a Creative Commons Attribution (CC-BY) 3.0 license.

level of decoupling between the entities interacting in space, time and synchronisation: all explicit dependencies between the entities are removed (Eugster *et al.*, 2003).

Several works with goal to design a publish/subscribe events dissemination protocol for ad-hoc mobile networks have been conducted (Pongthawornkamol *et al.* (2007); Costa *et al.* (2008); Huang and Garcia-Molina (2003); Baehni *et al.* (2005); Haillot and Guidec (2010); Paridel *et al.* (2010)). All the proposed protocols have the objective of ensuring a high ratio of reception of events by the subscribers while minimizing the energy consumption of the nodes; however, this double challenge is not yet optimally achieved. Indeed, protocols providing a good ratio of event dissemination (Baehni *et al.* (2005); Haillot and Guidec (2010)), exchange generally a great number of messages; this has a negative impact on energy consumption. In other side, those who have emphasis on reducing the number of messages exchanged as that of (Paridel *et al.*, 2010) have an unsatisfactory dissemination ratio. In general, the existing protocols do not yet make it possible to reconcile perfectly energy saving and high ratio of receipt of events by subscribers.

In this study, we propose within the context ofmobile ad-hoc networks, a distributed publish/subscribe protocol for disseminating events (texts, images, files, etc.) relating to hierarchical topics. In this protocol, (1) Events are disseminated in the network, either by an action known as *publish* which consists for a publisher to inject information into the system, or at the request of a peer (a station) wishing to be updated; to this end, stations periodically broadcast *announcement messages* allowing not only the station issuing it to signal its presence to its neighbors, but also and specially to request new events on its subjects of interest. (2) A peer involved in the dissemination of an event is either the producer of the event, or a subscriber to the topic to which the event belongs and who has previously received a copy of the event, or an *altruistic peer* (see sec. 2.2).

The objectives of the proposed protocol are to ensure a high rate of events reception by subscribers (utility objective) while minimizing the number of messages exchanged (usability objective: energy savings). These objectives must be achieved while considering the intrinsic constraints characterizing MANET (mobility - the ”contacts” between stations are potentially ”fugitive”-, the low power and limited range of the terminals that constitute it (Corson and Macker, 1999), etc.) in order to enable the applications that will implement SocialMANET to offer quality services. SocialMANET implements the hierarchical subjects subscription model which, in addition to minimize the number of subscriptions required to receive events in several topics, also permits to formulate refined subscriptions requests close to the content-based subscription model without inheriting its drawbacks (higher computational load on nodes, complex to implement, etc).

To achieve the above presented objectives, we have (the major contributions of the paper):

1) Developed a technique to significantly reduce the number of messages exchanged during contacts between nodes to discover and receive new events in the subjects to which they subscribe. In fact, a single message sent from a node written in a dedicated format that we have defined is enough to detect its needs (the identifiers of events that it does not have from the point of view of the message receiver), before the events transfer phase; note that it takes several exchanges in other protocols (like in the ones proposed by Baehni *et al.* (2005); Haillot and Guidec (2010)) to obtain the same result. It should also be noted that SocialMAN*ET* also significantly reduces the number of times a node can be sent the same event of the same topic by his neighbors. Indeed, by borrowing a technique presented by Baehni *et al.* (2005), when the neighbors of a node detect that it do not have some events in a topic in which it have subscribed, each calculates a date (different a priori) on which it will send it to him; and before that date, if by listening the network it realize that another peer has already sent it to him, it will no longer do so.
2) Introduces the concept of *lazily altruistic nodes*. Such nodes are able during exchanges to detect topics of exclusive interest to their neighbors in order to search (by necessity), store and transfer to them the events they do not yet have in these topics at the appropriate time. SocialMANET thus also adapts to situations in which the interests of the network nodes tend to diverge by allowing altruistic nodes to lazily ensure the availability of events in topics to which they are not subscribers.
3) Construct topics identifiers from the *dynamic level numbering* technique proposed by (Böhme and Rahm, 2004); this technique is inspired from the *Dewey Decimal System* (Mitra, 2009). The strategy of generatingtopics's identifiers allows, among other things, to determine just from a peer's subscription list whether or not an event related to a topic interests it, even if it has only subscribed to a parent topic of the one of that event.
4) Produced with great ease *FacInfo*: a prototype system implementing SocialMANET and allowing the dissemination of academic informations within a community of students in a higher education institution.

The rest of this manuscript is organized as follows: section 2 presents a brief state of the art on publish/subscribe events dissemination protocols. A running example is introduced in the section 3 followed by a presentation of our strategy for generating subject identifiers. Section 4 is dedicated to the detailed and illustrated presentation of the SocialMANET protocol. A critical evaluation of its performance is made in section 5. Finally, we conclude this paper in section 6.

Overview on Publish/Subscribe and State of the Art

Overview on Publish/Subscribe

The functioning of a system in which the peers communicate according to the publish/subscribe paradigm (in the following, we will call them *publish/subscribe-based systems*) can be summarized as follows: the publisher (a peer) introduces an event into the system (publish), then this event is notified to all subscribers; these are those who have previously declared to the system their interest in this event by defining constraints on the content of the events for which they wish to be notified (this is called content-based subscription (Liu and Özsu, 2018)) or by subscribing to the class of this event (this is called topic-based subscription (Liu and Özsu, 2018)).

An important variant of the topic-based subscription is the one in which subjects are hierarchized (Baehni *et al.*, 2005); here, any subscription to a topic leads to an implicit subscription to the topics that precede it in the hierarchy. Topic hierarchization allows to formulate refined subscriptions similar to those of a content-based publish/subscribe system by moving down the topic hierarchy (simply subscribe to the topics at the bottom of the hierarchy); moreover, it permit to minimize the number of subscriptions required to receive events in multiple topics (simply subscribe to the topics at the top of the hierarchy).

In some publish/subscribe systems, a validity period is assigned to each event and, when it expires, the event is no longer disseminated in the network. This saves resources (memory, bandwidth and energy) by avoiding storing and/or transferring obsolete information.

Events produced in a publish/subscribe system are assimilable to documents; this generic term borrowed from Haillot and Guidec (2010) avoids limiting the information exchanged to simple messages as do most publish/subscribe events dissemination protocols (Baldoni *et al.* (2007); Costa *et al.* (2008); Baehni *et al.* (2005), but also to disseminate files by taking advantage of the power of the publish/subscribe mechanism. However, for our purposes, in SocialMANET we will limit ourselves on files of sufficiently small sizes (in the order of megabyte), knowing that this model can be improved by splitting large files into several small files as most peer-to-peer file transfer protocols such as BitTorrent (Pouwelse *et al.*, 2005) does. In the following, as long as there is no ambiguity, we will use the term *event* to refer to both an event and the corresponding document.

State of the Art

Over the past two decades, many publish/subscribe-based systems have been proposed for both essentially static environments (e.g. CEA (Hayton *et al.*, 1996), SCRIBE (Rowstron and Druschel, 2001) and SIENA (Carzaniga *et al.*, 2000)) as well as for extremely dynamic mobile environments such as ad-hoc mobile networks (Pongthawornkamol *et al.*, 2007; Costa *et al.* 2008; Huang and Garcia-Molina, 2003; Baehni *et al.* 2005; Haillot and Guidec, 2010; Paridel *et al.*, 2010) which are of particular interest in this paper. As mentioned in the introduction, we distinguish those who propose solutions for the topic-based variant (Costa *et al.*, 2008; Baehni *et al.*, 2005) and those who propose solutions for the content-based variant (Pongthawornkamol *et al.*, 2007; Haillot and Guidec, 2010).

Based on the observation that people with a common social bond tend to group together most often, Costa *et al.* (2008) build what they calls *SocialCast*: it is a publish/subscribe-based system that uses people's social interaction measures to choose the most prominent one to carry information and share it with others. The choice of "carrier" in SocialCast is based on the notion of *utility*, which for an interest (subscription), represents for a given node, how eligible it is to be the carrier of the related events.

SocialCast requires that it exists at a given time, only β carriers for a given event. This considerably reduces the system's dissemination power because only carriers have the right to distribute events in the network. Baehni *et al.* (2005) on the other hand propose a hierarchical topic-based publish/subscribe protocol in which each node of the system can participate in the distribution of events. Moreover, each event has a validity period and each node shares events on a topic only with the nodes that subscribe to it in order to save node energy. It should be noted immediately that such systems only work well in an environment where the proportion of nodes sharing common interests is high; however, they show their limitations (poor dissemination rate) if nodes tend to subscribe to different topics. To remedy this type of behavior, Haillot and Guidec (2010) introduced the notion of *altruistic carriers* to designate the nodes that can distribute publications on subjects to which their neighbors have subscribed but not them. The protocol described by Haillot and Guidec (2010) provides that an altruistic nodes can disable its altruistic behavior.

The Running Example and the Topics Identifiers Generation Strategy

The Running Example

In this subsection, we describe an example that will be used throughout this paper to illustrate the various concepts and protocols proposed.

Consider the case of the implementation of an application for disseminating informations on academic activities within a community of students in a faculty.

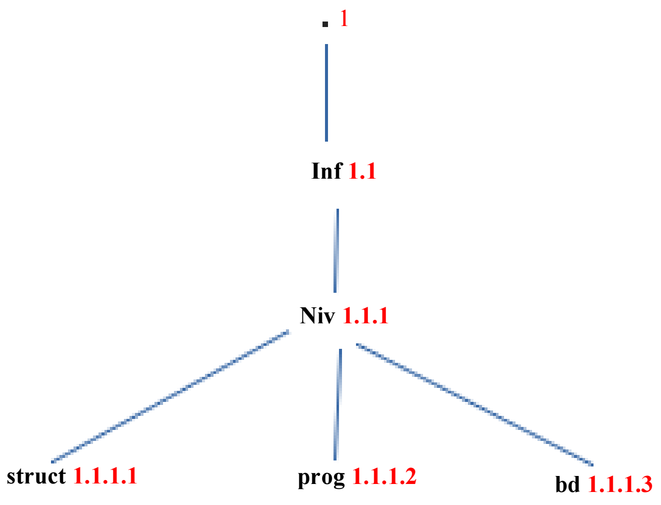


Fig. 1: Hierarchical structure of the topics of the running example

Table 1: List of topics and associated identifiers

Subjects	Identifiers
.	1
./inf	1.1
./inf/niv	1.1.1
./inf/niv/struct	1.1.1.1
./inf/niv/prog	1.1.1.2
./inf/niv/bd	1.1.1.3

This application should enable students who have installed it on their mobile terminals (tablets, smartphones, etc.) to be informed of the scheduling of lectures, tutorials (TD/TP), the availability of hand-out for lectures and TD/TP's materials, the scheduling of exam, etc. for each level of study.

By structuring this information into categories so that each category designates a "*topic*" according to the terminology used in the publish/subscribe systems, the topics *Field of study, Level of study, Class (UV)* can be obtained; they can be structured as follows: each field of study has as sub-topics all the levels of study existing in this field of study, each of them also having as sub-topics the constitutive elements (UV) of the teaching units provided at this level of study.

An instance of this hierarchy for the case of a (imaginary) faculty with a single field of study, the IT field (*Inf*), a single level of study (*Niv 1*) and three UV: Data Structure (*struc*), Programming (*prog*) and Databases (*bd*) is illustrated in the Fig. 1. The dot "." is the root subject and a node that subscribes to it indicates its interest in all informations related to the whole faculty.

Topics Identifiers Generation Strategy

In SocialMANET, subject identifiers (hierarchical) are generated according to the *dynamic level numbering* technique (Böhme and Rahm, 2004) which is inspired from the *Dewey Decimal System* (Mitra, 2009).

Let's recall that, in the *Dewey Decimal System* an identifier is a sequence of numerical values generally separated by dots.

For a given hierarchical structure, the *Dewey* identifiers of its nodes can be generated as follows:

- The root node is identified by 1 and its sons (run from left to right) are identified by 1.1, 1.2, 1.3, etc
- At each subsequent level, the sons of a *x* node are identified by *identifier*(*x*).*1*, *identifier*(*x*).*2*, *identifier*(*x*).*3*, etc., where the function *identifier*(*x*) returns the identifier of the node *x*.

As far as we're concerned, since the nodes represent topics, the topics of identifier *id.1*, . . . , *id.m* are all sub-topics of the topic whose identifier is *id*. This generation strategy of topics identifiers makes it possible not only to avoid regenerating some of the topic identifiers when new topics are added, but also and above all, to easily determine whether there is a relationship of kinship *(ancestor-descendant)* between two given topics. It will therefore be easy to know just from the list of subscriptions of a peer, whether or not an event relating to a topic interests it, even if it has only subscribed to a parent topic of the one of that event. Indeed, an event published in a topic of identifier *id1* interests a peer *p* only if *p* is a subscriber to a topic of identifier *id2* such that *id2* is a prefix to *id1*.

Table 1 shows for each of the topics selected for our running example (Fig. 1) the Dewey identifier associated with it. Note that the topic *bd* having identifier *1.1.1.3* is a sub-topic of the one with identifier *1.1* because, the string *1.1* is a prefix of the string *1.1.1.3*.

The SocialMANET Protocol

This section is dedicated to the presentation and the illustration of SocialMANET.

Overview

SocialMANET is a distributed publish/subscribe protocol that is collaboratively executed by the nodes of a mobile ad-hoc network (all nodes execute the same protocol).

As with any publish/subscribe protocol, SocialMANET distinguishes three main sub-protocols governing the subscription, publication and dissemination phases respectively. The dissemination phase that is of more interest to us in this paper is broken down into two sub-phases in SocialMANET: the *needs detection* sub-phase, during which a node by signaling itself in a neighborhood allows other nodes in that neighborhood to detect which of its needs can be met, and the *events transfer* sub-phase, during which nodes broadcast the events they own according to the detected needs of their neighbors.

In practice, SocialMANET is run on a virtual network layer within an existing physical network (MANET). Any node of the network participating in the execution of the SocialMANET protocol is autonomous and is able to communicate with any other node in its neighborhood (the neighborhood of a node consists of the nodes that are within its radio range). Each node can be both publisher and subscriber. All nodes ensure the dissemination of the events they own to interested subscribers. Communications are made by broadcast using Wi-Fi (alias IEEE 802.11). There is no message forwarding (Costa *et al*., 2008); in other words, a node cannot send a message to another node outside its vicinity. Therefore, during simulations, the horizon of each message (defined as the number of nodes it can cross) will be set to zero (0).

Data Structures

On a node, the events are stored (cache) in a directory named *EventsRepository*. It is organized in order to group all the publications related to a topic and its subtopics in a single tree structure in accordance with the hierarchy chosen for the topics. Thus, the *EventsRepository* directory corresponds to the root topic and each of his subdirectories correspond to one of his direct subtopics and so on. Figure 2 shows an illustration of the cache for our running example.

Each event contains meta-informations: its identifier, the identifier of the topic to which it belongs, its publishing date and its validity. These meta-informations are stored in a memory data structure named *EventsTable*.

When a node subscribes to a topic, its id is added (without duplication) to a table called *LocalSubscriptionsTable*. Each node also has a table named *NeighborhoodSubscriptionsTable* that allows it in case of altruistic behavior (see section 4.1), to detect and store its neighbors' subscriptions to topics that are not part of his own subscriptions. In the following, as long as there is no ambiguity, we will designate both *LocalSubscriptionsTable* and *NeighborhoodSubscriptionsTable* by *SubscriptionsTable* when we have to talk about the subscriptions of a node.

Information about the possessions of nodes located in the vicinity of a given node is stored in the data structure named *NeighborhoodTable*. It contains for each neighbor: its identifier, the identifiers of the topics to which it subscribes and for each of them the identifiers of the associated events (these are the events already in the possession of this neighbor). Figure 3 shows an example of the *EventsTable* and *NeighborhoodTable* tables. In this figure, the first entry in the table *EventsTable* states for example that, in the topic identified by *1.1.1*, an event identified by *112ACF45O* was published on *12/02/2018* at 02:00 PM and has a validity period of 72h. The second entry in the table *NeighborhoodTable* states that, the neighbor identified by *FE00145* has subscribed to the topic identified by *1.1.1* and already has the event identified by *45HHT0LP* done in this topic.

The information about the events to be served in the network are stored in a table called *EventsToSend* (an example is shown in the Fig. 6). Each entry in this table is associated with an event and has the following information: the identifier of the topic to which the event belongs, the identifier of the concerned event, its publishing date, its validity and the list of identifiers of the recipient nodes (the nodes that the sender knows that they are on need of this event).

Due to the limited storage capacity of the nodes, the tables *EventsRepository*, *EventsTable*, *NeighborhoodTable*, etc. may at some point be saturated. The memory management policy adopted is the one proposed by Baehni *et al*. (2005).

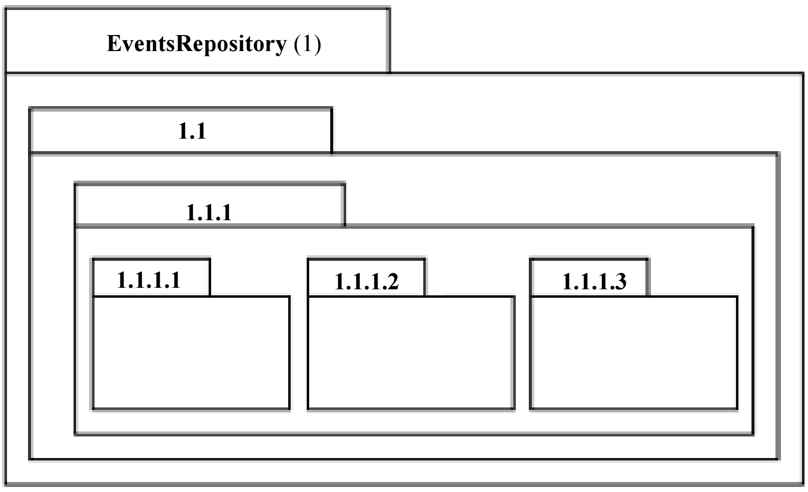


Fig. 2: Storing documents in the file system

1.1.1	112ACF450	12/02/2018 14:00:12	72H
1.1.1	PG2457PK	17/02/2018 09:34:55	7J
1.1.1.2	7777HB56	17/02/2018 17:03:20	2H

EventsTable

FE00145	1.1	12ACF450
FE00145	1.1.1	45HHT0LP
FEP0D44	1.1.1.2	7777HB56

NeighborhoodTable

Fig. 3: Data Structures EventsTable and NeighborhoodTable

Nevertheless, note that when an event is no longer valid, the corresponding entry is removed from the *EventsTable* table. The corresponding document remains, however, stored in the *EventsRepository* directory until it is explicitly deleted by the user.

The Protocol

As mentioned above, SocialMANET is broken down into three sub-protocols, the most important of which is the one relating to dissemination. We describe each of them below.

The Subscription Sub-Protocol

A subscription is a declaration of interest made by a given node according to a specific subject. It can be done at any time.

In practice, the node invokes the subscription routine by providing as parameter the identifier of the topic to which it wishes to subscribe (*subscribe(subjectID)*); this identifier is then inserted in the *LocalSubscriptionsTable* and the dissemination phase is triggered immediately. Since it is a declaration of interest for a topic, it is imperative to verify without delay whether related events exist in the neighborhood; we take the opportunity to update the events related to other topics of interest in the *SubscriptionsTable*.

The Publication Sub-Protocol

A node *n* can at any time publish in a given topic by invoking the publishing routine *publish (subject ID, Doc, Validity)*; it takes as parameters the identifier *subject ID* of the topic in which it wishes to publish, the document *Doc* to publish as well as the validity of the event. In response to this solicitation, the system stores the document to be published in the appropriate subdirectory (that corresponding to the topic of the event) in the *EventsRepository* directory, then generates the identifier of the event and inserts it into the table *EventsTable*. Then, at the same time as the event, it produces and distributes a *dissemination message*, with the value "*" in the recipient list field, to signify that it is a new event: any subscriber of the topic to which the event belongs is therefore recipient.

The Dissemination Sub-Protocol

As mentioned previously, the dissemination phase consists of two sub-phases: the needs detection and the transfer of events.

The Needs Detection Sub-Phase

It is triggered by a node just after he has subscribed to a topic, or periodically after a fixed delay called *RESEARCH-DELAY* has elapsed since the last needs detection sub-phase. This is done by broadcasting an *announcement message* (called *message QUERY*) into the network by any node as soon as one of the previously stated conditions is satisfied.

A *QUERY* message encapsulates the list of topic identifiers contained in the *SubscriptionsTable*; each of these identifiers is accompanied by the identifiers of the events made in the topic that the node already possesses. It is formatted as in the Fig. 4. In this format, *Id* is the identifier of the node that emits the message, *subject-i* is the identifier of a topic and, *key-i-j* is the identifier of the *j-th* event published in the subject identified by *subject-i*.

Id [subject-1 (key-1-1, key-1-2,…, key-1-n1),
subject-2 (key-2-1, key-2-2,…, key-2-n2)
…
subject-m (key-m-1, key-m-2,…, key-m-nm)]

Fig. 4: *QUERY* message format

For example, going back to our running example and considering the case of a student named Fabien from our imaginary faculty, who has an tablet identifier by *FE754P0* and has subscribed to the topics *./inf/niv/bd* and *./inf/niv/struct*: he must have in his *SubscriptionsTable* the identifiers of the topics *1.1.1.1* and *1.1.1.3*. If we suppose moreover that he already has the events identified by *11GT158P2* and *BTA14P159* (resp. *4TR854L4Z*) already carried out in the topic identified by *1.1.1.1* (resp. *1.1.1.3*), the message *QUERY* that he will broadcast will be formatted as follows: *FE754P0[1.1.1.1(11GT158P2, BTA14P159), 1.1.1.3(4TR854L4Z)].*

Node processing on receipt of a *QUERY* message depends on whether it has adopted altruistic behavior or not:

- If it is non-altruistic, for each of the subjects contained in the request, it calculates the list *pubToSend* (resp. *pubToAsk*) containing the difference between the identifiers of the events which it possesses (resp. that he has received) in the topic and those received (resp. it possesses). These lists make it possible to know if there are needs of the transmitter which it can satisfy (in case *pubToSend* is not empty: it then updates its *NeighborhoodTable*) or then, if this one can satisfy some of its needs (in case *pubToAsk* is not empty: it precipitates the execution of its next phase of sending the message *QUERY* in order to increase its chances to receive events. In fact, because of the mobility of the nodes, the transmitter can very quickly be out of reach.
- If it is (lazily) altruistic, in addition to the treatments described above for the non-altruistic case, it gleans information about topics it does not share with the sender and updates its *NeighborhoodTable* and *NeighborhoodSubscriptionsTable* tables. Specifically, for each of these topics, it inserts in its *NeighborhoodTable* the identifier of the issuer, the identifiers of all its subscriptions and the identifiers of event that it already has; its *NeighborhoodSubscriptionsTable* table is also updated by including the identifiers of these topics.

The different types of informations that can be synthesized from the information encapsulated in a *QUERY* message received and the identifiers of the events present at a given moment on a node are highlighted in the Fig. 5.

When a node has finished broadcasting a *QUERY* message, it schedules the next broadcast within a time frame determined by the *RESEARCH-DELAY* parameter; its value is suggested but updatable by the user and must be chosen wisely because, as we will see in the section 5.1, a very low value of *RESEARCH-DELAY* generates a substantial multiplication of the number of messages sent to the system, while a very large value can lead to the degradation of the system's performance as a result of the significant reduction in the events reception rate.

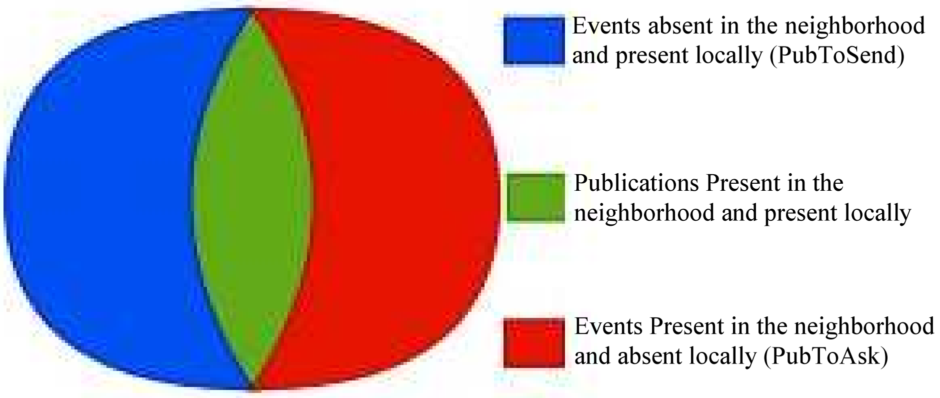

Events absent in the neighborhood and present locally (PubToSend)
Publications Present in the neighborhood and present locally
Events Present in the neighborhood and absent locally (PubToAsk)

Fig. 5: Information synthesized from a *QUERY* message received by a node

1.1.1	788OX15UI	03/03/2019 11:23:02	2H	FE120S2 FE451PI, JE47OP9
1.1.1.2	17475HY95P	25/02/2019 06:30:45	48H	PX12452
1.1.1.2	AS4V055UI	01/03/2019 13:43:12	2J	TI1452P

Fig. 6: Table *EventsToSend*

Node_Id [Topic_Id,
Event_Id,
Publication date,
Validity,
(Neighbor-1, Neighbor-2,…, Neighbor-n | *)]
(recipient identifiers or the * symbol to indicate a new event)

Fig. 7: Format of dissemination messages

Events Transfer Sub-Phase

Following receipt of a *QUERY* message and after updating the different tables as described above, the receiver node builds the *EventsToSend* table containing informations on the list of events to be broadcast in the network, then, it calculates a *Back-off* delay whose value is inversely proportional to the size of the *EventsToSend* table and goes on hold: it is in *broadcast waiting mode*. At the end of this delay, the transfer sub-phase is triggered by the update of the *NeighborhoodTable* table. Then, for each entry in the *EventsToSend* table (which also contains the list of recipients of the message), a *dissemination message* whose format is shown in Fig. 7 is sent at the same time as the associated event. The presence of the list of recipients in this message allows the receiver of the message to know who has potentially already received what event so that, it can dynamically update its table *EventsToSend*: we try as much as possible to reduce the number of *redundant broadcasts*. An example of the *EventsToSend* table is shown in the Fig. 6.

While waiting for the *Back-off* delay, the node continues to listen to the network; it can receive other *QUERY* messages and update its *EventsToSend* table. The *Back-off* delay allows to collect a set of requests so that the broadcast of the events is done for a large number of recipients. This ingenious technique borrowed from Baehni *et al.* (2005) avoids the broadcast of the same event for each request received, because obviously, since it is a broadcast, it is useless to make one to satisfy each neighbor individually: it reduces the number of messages issued in the network. Moreover, the fact that this delay is inversely proportional to the size of the *EventsToSend* table of each node, makes it possible to hope that the distribution dates of the different nodes are shifted in time, so that to minimize the number of redundant broadcasts. Indeed, while a node is waiting to switch to broadcast mode, it can receive dissemination messages and realize that a need of one of its neighbors that it was about to satisfy has just been met (another node has just sent him an event that he was about to broadcast); he updates his *EventsToSend* table accordingly by ejecting this neighbor from the list of recipients for this event.

Upon receipt of a *dissemination message* (and associated event) by a node, if this message concerns him (case of an event belongs to a subject to which he subscribed and not appearing in his *EventsTable* table), then, it saves it in the *EventsRepository* directory, consequently update the *EventsTable* table and possibly the *EventsToSend* table if it is altruistic and turns out that the event received may be of interest to one of his neighbors.

We are not interested here in the order of reception of the informations on an event (the dissemination message and the corresponding document), nor in the way in which they are reconstituted at the reception. However, we hypothesize that the receipt of an event implies that both the dissemination message and the corresponding document are received.

Illustrations

We present in this subsection an example of protocol execution, illustrating it with the running example introduced in section 3.1; we also present an application prototype implementing SocialMANET.

An Example of Protocol Execution

Let's take the running example from the 3.1 section and assume that students John (J), Thomas (T) and Cyril (C) are enrolled in our imaginary faculty. Suppose also that J, T and C are the nodes of our ad-hoc mobile network (see Fig. 8) representing the terminals used by these respective students; in these terminals (smart-phone or laptop) runs an application implementing the SocialMANET protocol. In the Fig. 8, the circular neighborhood highlighted in dotted around each node represents its communication range. Thus, the nodes J and T (respectively T and C) can communicate with each other, which is not the case for the nodes J and C.

The initials states of the different nodes (given for each of them by the initial values of its different data structures) are presented in the Fig. 9. In order to simplify the presentation by focusing on the essentials, we have kept for the different tables only entries with a proven interest as far as the example is concerned. So, by going through the different cells of the tables from left to right, we have:

- For the *EventsTable* table: the identifier of the topic to which the event belongs, the identifier of the event
- For the *NeighborhoodTable* table: the identifier of the neighbor node, the identifier of the topic to which the neighbor node is subscribed, the identifiers of the events associated with each of these topics
- For the *EventsToSend* table: the identifier of the topic to which the event belongs, the identifier of the event and the identifiers of the neighboring nodes who are supposed to receive the event

For this illustration, we considered that only the node T is lazily altruistic. By carefully observing the Fig. 9, it is easy to deduce that J is subscribed to the topic *./inf* while T and C subscribe to the topics *./inf/niv/struct* and *./inf/niv/prog*. We can also notice that, following a previous interaction, T knows about the presence of C in its neighborhood. Moreover, the altruistic character of T has given him the possibility, at a given moment in the past, to record in his *NeighborhoodSubscriptionsTable* the identifier of the topic *./inf/niv/bd* to which he is not subscribed.

In what follows, we will show how the system evolves over time by considering primarily the dissemination phase. Thus, starting from a global state of the system given by the Fig. 9, we first explore the scenario in which the node T broadcasts a new event then, the one in which it initiates the needs detection phase.

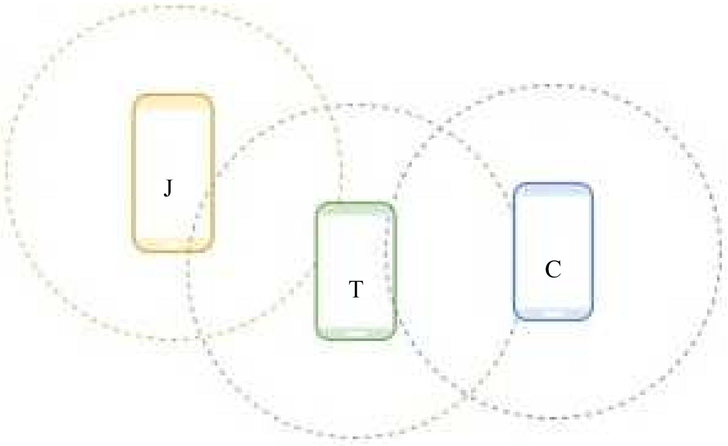


Fig. 8: Network nodes: nodes J and T (respectively T and C) are within radio range of each other and can communicate

Data structures		Node		
		J		
Subscriptions Table	Local	11		
	Neighborhood			
EventsTable		11 1112	mysq1.pdf td1.txt	
NeighborhoodTable				
EventsToSend				

Data structures		Node		
		T (a)		
Subscriptions Table	Local	1111, 1112		
	Neighborhood	1113		
EventsTable		1111 1112	jsx.js ex1.png	
NeighborhoodTable		C	1112	ex1.png
EventsToSend		1111	jsx.js	

Data structures		Node		
		C		
Subscriptions Table	Local	1111, 1112		
	Neighborhood			
EventsTable		1112	ex1.png	
NeighborhoodTable				
EventsToSend				

Fig. 9: Initial state of the nodes

Data structures		Node		
		J		
Subscriptions Table	Local	11		
	Neighborhood			
EventsTable		11 1112 1111	myq1.pdf td1.txt jsx.js	
NeighborhoodTable		T	1111	jsx.js
EventsToSend				

Data structures		Node		
		T (a)		
Subscriptions Table	Local	1111, 1112		
	Neighborhood	1113		
EventsTable		1111 1112	jsx.js ex1.png	
NeighborhoodTable		C	1112	ex1.png
EventsToSend				

Data structures		Node		
		C		
Subscriptions Table	Local	1111, 1112		
	Neighborhood			
EventsTable		1112 1111	ex1.png jsx.js	
NeighborhoodTable		T	1111	jsx.js
EventsToSend				

Fig. 10: Update of tables after receipt of T publication

[*First scenario:*] On the Fig. 9, we can see that, the node T has filled its table *EventsToSend* with information about the *jsx.js* document belonging to the topic *1.1.1.1* (./inf/niv/struct) that it is about to publish. When this document is published by T (broadcast of the dissemination message "*T [1.1.1.1, jsx.js, (*)]*"), it is received by nodes J and C. *C* has subscribed to the topic *1.1.1.1* to which this event belongs, as it does not have it yet, it stores it and updates its tables (Fig. 10). On the other hand, the node *J* not having explicitly subscribed to the topic *1.1.1.1*, but rather about *1.1* which is parent of *1.1.1.1*, is implicitly interested in the new event; It stores it and updates its tables (Fig. 10).

[*Second scenario:*] the node T broadcasts the following *QUERY* message: *T [1.1.1.1 (jsx.js), 1.1.1.2 (exl.png), 1.1. 1.3 ()]* declaring that he is interested in events belonging to the topics. */inf/niv/struct,./ inf/niv/prog* and *./inf/niv/bd* and that it owns the events identified by *jsx.js* and *ex1.png* (they are respectively Javascript and an image files) respectively belonging to the topics *./inf/niv/struct* and *./inf/niv/prog*. This message will be received by C and J and they will update their tables immediately as shown in Fig. 11. We can notice that the node C did not save in its table *NeighborhoodTable* an entry corresponding to the topic *1.1.1.3* as J did because, it is neither a subscriber of this topic nor an altruistic node. Node J did this because it is subscribed to the subject *1.1* which is a parent of *1.1.1.3*.

This message allowed J to know that he has to publish the *td1.txt* document, which he stores locally and of which T is not yet aware. So he built his *EventsToSend* table accordingly and waits for a pre-calculated *Back-off* delay. At the end of this delay, it broadcasts the unique entry of its table *EventsToSend* and the associated document (td1.txt). This post is received by T who updates his tables *EventsTable* and *NeighborHoodTable*, and then inserts it into his *EventsToSend* table because he realized that his neighbor C is interested. The system's state at this stage is represented by Fig. 12.

When the *Back-off* delay calculated by node T expires, it broadcasts the unique entry of its *EventsToSend* table and the associated document *(td1.txt)*. This event will be received by J and C who will update their tables as shown in Fig. 13.

The message *QUERY* that the node J received from T allowed him to detect the presence of the event *ex1.png* of the topic*./inf/niv/struct* at T. It will broadcast its message *QUERY* sooner than expected so that T knows that it needs this event and transfers it to him as soon as possible.

Data structures		Node		
		J		
Subscriptions Table	Local	11		
	Neighborhood			
EventsTable		11 1112 1111	mysq1.pdf td1. txt jsx js	
NeighborhoodTable		T T T	1111 1112 1113	jsx.js ex1.png
EventsToSend		1112	td1. tsx	T

Data structures		Node		
		T(a)		
Subscriptions Table		1111, 1112		
	Neighborhood	1113		
EventsTable		1111 1112	jsx.js ex1.png	
NeighborhoodTable		C	1112	ex1.png
EventsToSend				

Data structures		Node		
		C		
Subscriptions Table	Local	1111, 1112		
	Neighborhood			
EventsTable		1112 1111	ex1.png jsx js	
NeighborhoodTable		T T	1111 1112	jsx js ex1.png
EventsToSend				

Fig. 11: Update tables after receipt of the message *QUERY* issued by T

Data structures		Node		
		J		
Subscriptions Table	Local	11		
	Neighborhood			
EventsTable		11 1112 1111	mysq1.pdf td1.txt jsx.js	
NeighborhoodTable		T T T	1111 1112 1113	jsx.js exl.png, tdl.txt
EventsToSend				

Data structures		Node		
		T (a)		
Subscriptions Table	Local	1111, 1112		
	Neighborhood	1113		
EventsTable		1111 1112 1112	jsx js ex1.png td1.txt	
NeighborhoodTable		C J	11112 11112	ex1.png td1.txt
EventsToSend			td1.txt	C

Data structures		Node		
		C		
Subscriptions Table	Local	1111, 1112		
	Neighborhood			
EventsTable		1112 1111	ex1.png jsx.js	
NeighborhoodTable		T	1111 1112	jsx.js ex1.png
EventsToSend		T		

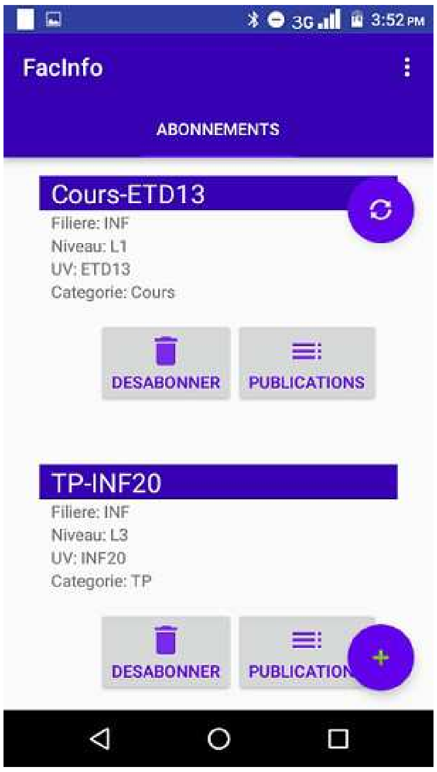
Fig. 12: System status after receiving td1.txt from T

Data structures		Node		
		J		
Subscriptions Table	Local	11		
	Neighborhood			
EventsTable		11 1112 1111	mysq1. pdf td1. txt jsx. js	
NeighborhoodTable		T T T	1111 1112 1113	jsx. js ex1.png td1.txt
EventsToSend				

Data structures		Node		
		T (a)		
Subscriptions Table	Local	1111, 1112		
	Neighborhood	1113		
EventsTable		1111 1112 1112	jsx.js ex1.png td1.txt	
NeighborhoodTable		C J	1112 1112	ex1.png td1.txt td1.txt
EventsToSend				

Data structures		Node		
		C		
Subscriptions Table	Local	1111, 1112		
	Neighborhood			
EventsTable		1112 1111 1112	ex1.png jsx.js td1.txt	
NeighborhoodTable		T T	1111 1112	jsx. js ex1.png td1.txt
EventsToSend				

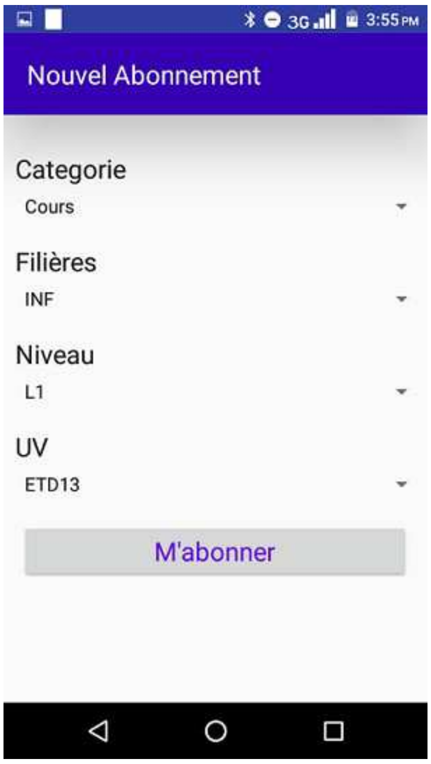
Fig. 13: Status of the system after receiving the latest publications issued by T



Cours-ETD13
Filiere: INF
Niveau: L1
UV: ETD13
Categorie: Cours

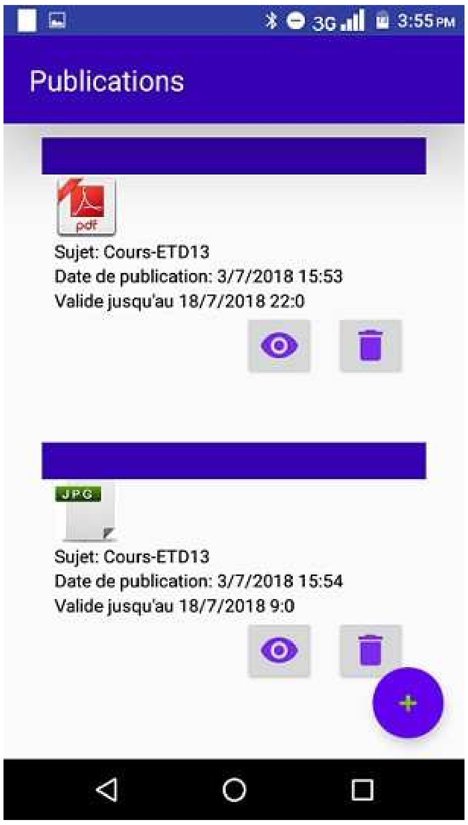
TP-INF20
Filiere: INF
Niveau: L3
UV: INF20
Categorie: TP

Fig. 14: A view of the subjects a user is subscribed to in FacInfo



Categorie
Cours
Filières
INF
Niveau
L1
UV
ETD13

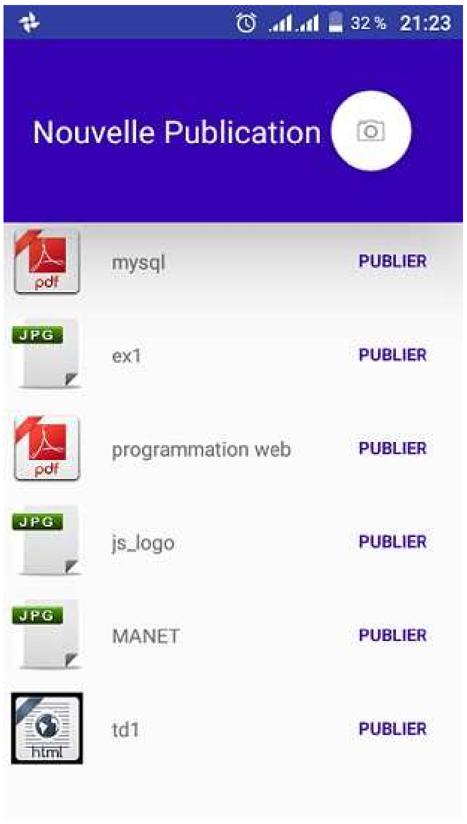
Fig. 15: A view of the subject subscription form in FacInfo



Sujet: Cours-ETD13
Date de publication: 3/7/2018 15:53
Valide jusqu'au 18/7/2018 22:0

Sujet: Cours-ETD13
Date de publication: 3/7/2018 15:54
Valide jusqu'au 18/7/2018 9:0

Fig. 16: A view of publications stored locally in FacInfo



pdf	mysql	PUBLIER
JPG	ex1	PUBLIER
pdf	programmation web	PUBLIER
JPG	js_logo	PUBLIER
JPG	MANET	PUBLIER
html	td1	PUBLIER

Fig. 17: A view of the screen for publishing a document in FacInfo

An Example of SocialMANET Implementation: The FacInfo Platform

In order to concretely experiment with the SocialMANET protocol, we developed a prototype (an Android application) of an information disseminating system called *FacInfo*. *FacInfo* implements the SocialMANET protocol and allows the students of a Faculty to share academic informations organized in hierarchical topics. Each topic is characterized by the *Field of study*, the *Level of study*, the Class (UV) concerned and the category (course, TD, TP, Exam, Notes). The student provides these informations to subscribe to a subject.

Figures 14-17 are some *FacInfo* screen-shots. They show respectively the list of subscriptions of the current user (Fig. 14), the different informations to be filled in (Category, Field of study, Level, UV) to subscribe to a given topic (Fig. 15), a view of events received in a topic (Fig. 16) and finally, a view of the selection screen of a document for publishing (Fig. 17).

Performances and Discussion

Performances

In this section, we present the results of the performance tests obtained by simulating our protocol using the NS2 simulator (*"NS-2 Network Simulator" http://www.isi.edu/nsnam/ns/*).

We conducted tests to assess the events reception rate based on parameters such as the *RESEARCH-DELAY* or the proportion of altruistic nodes. We also simulated under the same conditions the protocol proposed by Baehni *et al.* (2005) (it is conceptually close to ours) and compared the results obtained with those of our protocol.

Simulation Parameters

The simulations were carried out with a set of 100 nodes deployed over an area of 950m×800m, moving according to the Random Waypoint Mobility model (RWM) (Kraaier and Killat, 2005); this model makes it possible to randomly move the nodes in the network.

Throughout the simulation, each node randomly chooses a destination, moves there at a random speed lower than a fixed value, remains motionless for a so-called pause time, then chooses a new destination and the cycle starts again. We set the maximum speed of the nodes at 4 m/s and the pause time at 2 sec. The global parameters of the simulation are summarized in the Table 2.

Table 2: Simulation parameters

Parameters	Values
Mac	802.11b
Routing protocol	None
Bandwidth	1Mb/s
Detection range	50m
Communication range	50m
Simulation time	900s

Results and Discussion

Simulation 1*: Events reception rate according to the number of lazily altruistic nodes*

We varied the proportion of (lazily) altruistic nodes in the network, as well as the seed of the random number generator (*it is a number that is used by the simulator to generate the random numbers usually obtained using the rand () function*) and measured the events reception rate. We defined a set of 5 topics for which each node could randomly decide to subscribe before the 300th second of the simulation time. Each node is activated at a random time during the simulation; the activation here consists in launching the first iteration of the needs detection phase (the other iterations follow one another later).

During this simulation, only one event is published by a randomly selected node, for each of the 5 existing topics; this event is published at a random time before the 400th second from the beginning of the simulation.

The results obtained during this experiment are summarized in the Fig. 18. It shows that the results are essentially the same for different seed values. Overall, the events reception rate increases as the number of lazily altruistic nodes increases, and stabilizes as we approach and exceed 20 lazily altruistic nodes.

Altruistic behavior is costly for the node that adopts it (rapid decrease in its residual energy, rapid filling of its tables, etc.). It is all the more expensive for the network if there are a large number of them. However, as just noted, there is a threshold above which increasing the number of lazily altruistic nodes does not increase the events reception rate. Therefore, if the number of lazily altruistic nodes can be kept close to or equal to this threshold at all times, a high events reception rate will be ensured, while minimizing the cost induice by their altruistic behavior for the network.

Simulation 2*: events reception rate according to the value of RESEARCH-DELAY.*

In the second experiment, we maintained the seed value at 0 and varied the *RESEARCH-DELAY*, the delay that separates two successive runs from the needs detection phase. We also set the proportion of lazily altruistic nodes to zero. The other simulation parameters are identical to those of simulation 1.

Not surprisingly, Fig. 19 shows that the events reception rate decreases as the *RESEARCH-DELAY* increases. Indeed, when it is small (values between 0 and 150), nodes communicate more often, increasing their chances of discovering new events. On the other hand, when it is large (value greater than 350) communications between nodes are less frequent and this leads to a low rate of events reception. However, it should be noted that when nodes communicate a lot, this leads to high resource consumption (energy, bandwidth, etc.). It would therefore be important to find an appropriate value for this delay, that would minimize resource consumption, while ensuring a high rate of events reception.

Simulation 3*: SocialMANET Vs* Baehni *et al.* (2005*) protocol*.

By keeping the same simulation parameters, we measured the network load in terms of the number of messages exchanged during the simulation and the average size of the messages respectively. We measured these values for our protocol as well as for that of Baehni *et al.* (2005); as mentioned above, the choice of this protocol among those in the literature is motivated by the fact that it shares many concepts (publish/subscribe with topic-based subscriptions, *RESEARCH-DELAY*, etc.) with our protocol. However, they do not assimilate events to files and do not take into account the altruistic nodes as we do.

Figures 20 and 21 present the ratios of the data collected for our protocol by those of the protocol of (Baehni *et al.*, 2005) by applying the following respective formulas:

$$\frac{\textit{Number of messages for SocialMANET}}{\textit{Number of messages for the protocol of Sebastien et al.}}$$

and:

$$\frac{\textit{Average message size for SocialMANET}}{\textit{Average message size for the protocol of Sebastien et al.}}$$

Not surprisingly, it can be seen in Fig. 20, that SocialMANET considerably reduces the number of messages exchanged in the network. Indeed, with SocialMANET, a single message (*QUERY*) broadcast by a node allows its neighbors to detect its needs; which is done in more than one step (several messages) in many other protocols of the literature as the ones described by Haillot and Guidec (2010) and by Baehni *et al.* (2005) (particularyly in which two exchanges are required). In the worst case (when the *RESEARCH-DELAY* is very long), SocialMANET generates at least half as many messages as the protocol of Baehni *et al.* (2005).

This significant reduction in the number of messages is an asset for SocialMANET because it allows nodes to save resources that should be allocated to processing these messages, but also to save time that can be used to carry out other tasks. However, it is not without consequences because, as shown in Fig. 21, the average message size in SocialMANET is higher. However, as can be seen in Fig. 22 which shows the impact of the number of messages exchanged on node energy consumption, although SocialMANET generates larger messages, it consumes less energy than Baehni *et al.* (2005) protocol. As could also be expected, Fig. 22 also shows that energy consumption is closely related to the number of messages exchanged. Indeed, since the number of messages exchanged decreases as the *RESEARCH-DELAY* increases, it can be seen in Fig. 22 that the amount of energy consumed decreases as the *RESEARCH-DELAY* increases.

Simulation 4*: Events reception rate according to their validity*.

For the last experiment, we considered a single subject to which all nodes subscribed at 200s from the beginning of the simulation at the latest. Each node is activated at a random time during the simulation and only one event is published by a randomly selected node before the 200th second of the simulation start. The seed value was set to 0 and the *RESEARCH-DELAY* to 50s. We varied the validity of the event and measured the events reception rate, both for our protocol and for that of Baehni *et al.* (2005). Figure 23 shows the results obtained during this experiment.

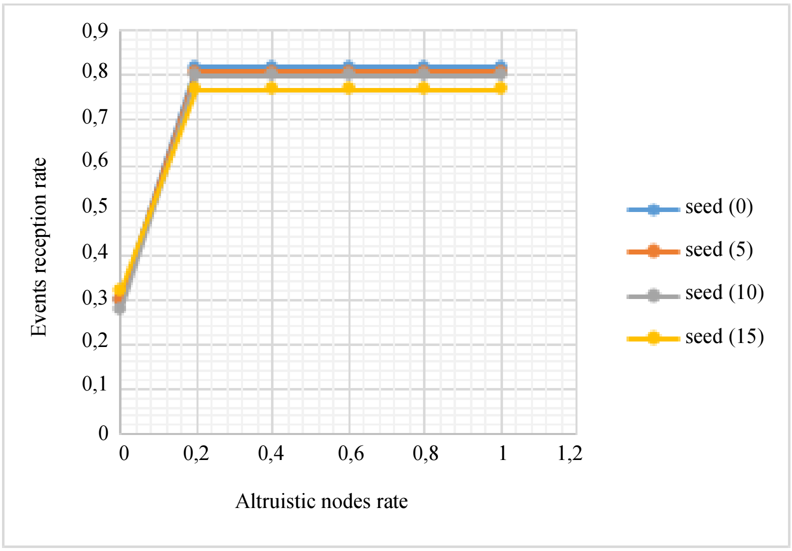


Fig. 18: Events reception rate according to the number of lazily altruistic nodes

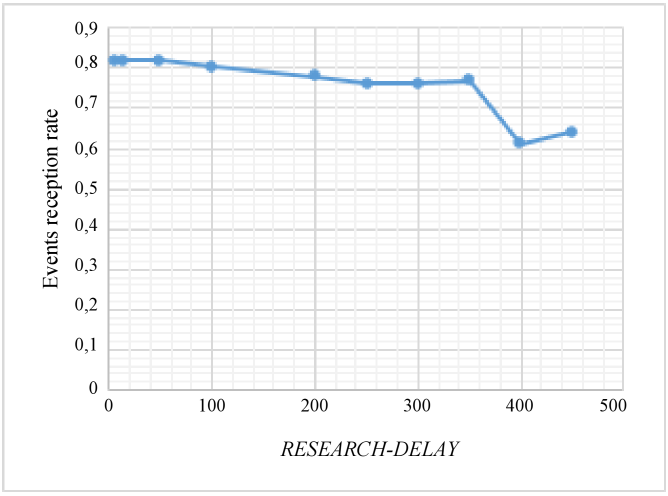


Fig. 19: Events reception rate according to the value of *RESEARCH-DELAY*

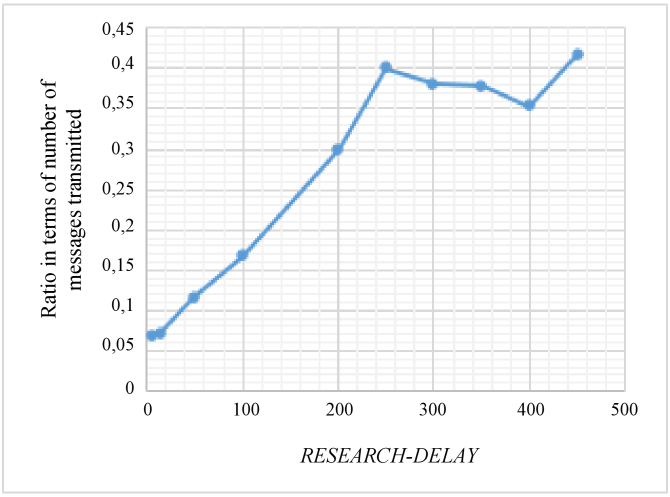


Fig. 20: Ratio in terms of number of messages between the SocialMANET protocol and the Baehni *et al*.(2005) protocol

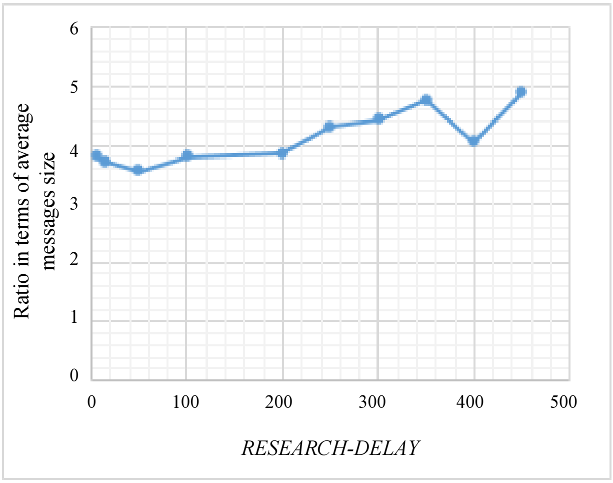


Fig. 21: Ratio in terms of average message size between the SocialMANET protocol and the Baehni *et al*.(2005)

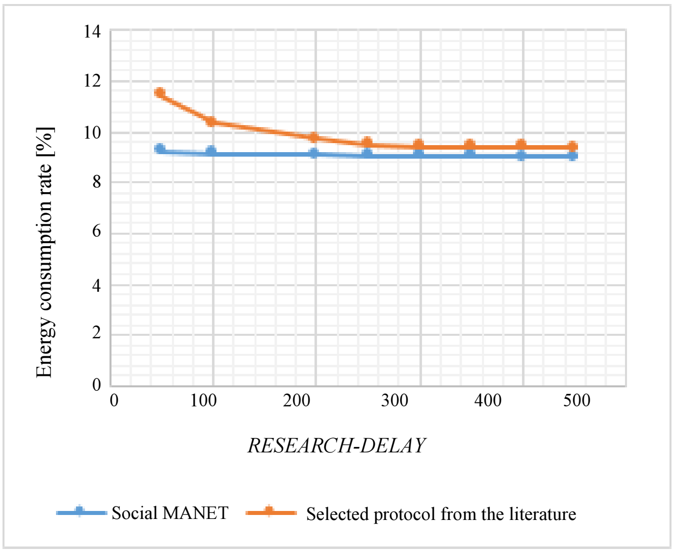


Fig. 22: Amount of energy consumed by the nodes

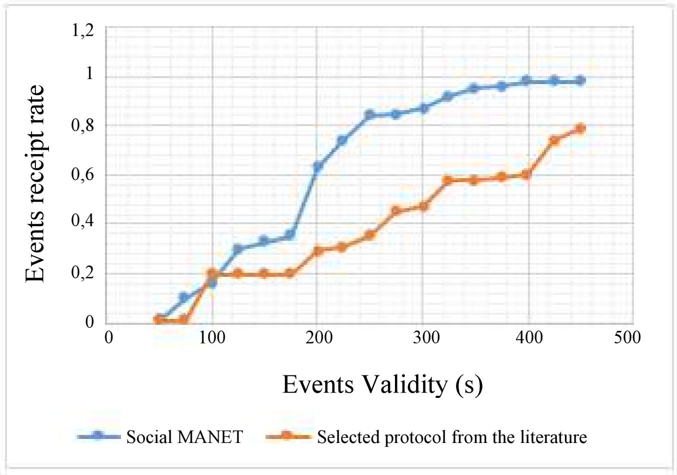


Fig. 23: Events reception rate according to their validity

As might be expected, it can be seen that the higher the validity of the event, the higher the rate of receipt. Indeed, validity represents the duration that a event can make in the network while being subject to different exchanges. The longer this time, the more the event is shared, which increases the reception rate. We can also note that, in the configuration where all nodes subscribe to the same subjects, SocialMAN*ET al*ways guarantees a higher reception rate than the protocol of Baehni *et al*. (2005).

Conclusion

In this article, we presented SocialMANET, a distributed publish/subscribe events dissemination protocol for mobile ad-hoc networks. It makes the best use of informations resulting from exchanges between nodes to quickly detect their needs and satisfy them in order to ensure a high rate of event reception. In fact, the proposed (new) strategy for generating subject identifiers based on the dynamic level numbering technique, allows nodes to detect with a lower cost if any node has subscribed (even implicitly: subscription to a parent subject) to a given subject. Moreover, the introduction of the concept of *lazily altruistic nodes* in SocialMANET has resulted in a protocol that also adapts perfectly to situations in which the interests of the network nodes tend to diverge.

SocialMANET has been validated both through simulations - which have provided very satisfactory results - and experimentation: the production of an application's prototype for the dissemination of academic information based on an implementation of SocialMANET (FacInfo). This experiment has confirmed that SocialMANET can be implemented easily and efficiently.

In contrast to other protocols in the literature that perform needs detection in two phases consisting respectively on the exchange of identifiers of topics of interest and the exchange of identifiers of events owned on these topics, SocialMANET via a compact QUERY message format does so in a single phase. This drastically reduces the number of messages exchanged and consequently the amount of energy consumed by the nodes. We therefore have a protocol that meets the objectives of utility and usability set out in the introduction. However, there is still a lot of work to be done to secure SocialMANET.

Acknowledgement

Authors warmly thank doctor Kengne T. Vianney for enriching discussions they had, during both laboratory seminars and the defense of Master's thesis of Mr. Tchembé Martin Xavier. They also thank Mr. Foko S. Miguel for the invaluable help provided during the simulation phase.

Author's Contributions

Maurice Tchoupé Tchendji: Designed the research proposal, organized and directed the study, contributed to the literature review in the field and the writing of the final manuscript.

Martin Xavier Tchembé: Contributed to the literature review in the field, conduct the simulations and data analysis, the design, implementation and testing of the prototype and to the writing of the final manuscript.

Innocent Tialo Necheu: Contributed to the literature review in the field, participated to the design, implementation and testing of the prototype and in drafting the article.

Ethics

The authors declare no conflicts of interest regarding the publication of this paper who is original and not published elsewhere. Authors also confirm that they have read and approved the manuscript and that, there are no ethical issues involved.

References

Baehni, S., C.S. Chhabra and R. Guerraoui, 2005. Frugal event dissemination in a mobile environment. Proceedings of the, 6th International Middleware Conference ACM/IFIP/USENIX, Grenoble, France, Nov. 28-Dec. 2, Springer, Berlin, Heidelberg, pp: 205-224. DOI: 10.1007/11587552_11

Baldoni, R., R. Beraldi, V. Quéma, L. Querzoni and S. Tucci Piergiovanni, 2007. TERA: Topic-based event routing for peer-to-peer architectures. Proceedings of the International Conference on Inaugural Distributed Event-Based Systems, June 20-22, Toronto, Ontario, Canada, pp: 2-13. DOI: 10.1145/1266894.1266898

Böhme, T. and E. Rahm, 2004. Supporting efficient streaming and insertion of xml data in RDBMS. University of Leipzig, Germany. pp: 70-81.

Carzaniga, A., D.S. Rosenblum and A.L. Wolf, 2000. Achieving scalability and expressiveness in an internet-scale event notification service. Proceedings of the 19th Annual ACM Symposium on Principles of Distributed Computing, July 16-19, Portland, Oregon, USA, pp: 219-227. DOI: 10.1145/343477.343622

Corson, M.S. and J.P. Macker, 1999. Mobile ad hoc networking (MANET): Routing protocol performance issues and evaluation considerations. RFC, 2501: 1-12. DOI: 10.17487/rfc2501

Costa, P., C. Mascolo, M. Musolesi and G.P. Picco, 2008. Socially-aware routing for publish-subscribe in delay-tolerant mobile ad hoc networks. IEEE J. Selected Areas Commun., 26: 748-760. DOI: 10.17487/rfc2501

Eugster, P., P. Felber, R. Guerraoui and A. Kermarrec, 2003. The many faces of publish/subscribe. ACM Comp. Surveys, 35: 114-131. DOI: 10.1145/857076.857078

Haillot, J. and F. Guidec, 2010. A protocol for content-based communication in disconnected mobile ad hoc networks. Mobile Information Syst., 6: 123-154. DOI: 10.1155/2010/839457

Hayton, R., J. Bacon, J. Bates and K. Moody, 1996. Using events to build large scale distributed applications. Proceedings of the 7th ACM SIGOPS European Workshop: Systems Support for Worldwide Applications, Connemara, Ireland, September 9-11, pp: 9-16. DOI: 10.1145/504450.504453

Huang, Y. and H. Garcia-Molina, 2003. Publish/Subscribe Tree Construction in Wireless Ad-Hoc Networks. In: Mobile Data Management, Chen, M.S., P.K. Chrysanthis, M. Sloman and A. Zaslavsky (Eds.), Lecture Notes in Computer Science, Springer, Berlin, Heidelberg.

Kraaier, J. and U. Killat, 2005. The random waypoint city model -: user distribution in a street-based mobility model for wireless network simulations. Proceedings of the 3rd ACM International Workshop on Wireless Mobile Applications and Services on WLAN Hotspots, Sep. 02 - 02, ACM, Cologne, Germany. DOI: 10.1145/1080730.1080749

Liu, L. and M.T. Ozsu Editors, 2018. Encyclopedia of Database Systems, Second Edition. Springer. DOI: 10.1007/978-981-10-0152-9

Mitra, P. 2009. Dewey Decimal System, pp: 808-809. Springer US, Boston, MA.

Paridel, K., Y. Vanrompay and Y. Berbers, 2010. Fadip: Lightweight publish/subscribe for mobile ad hoc networks. 6427: 798-810. DOI: 10.1007/978-3-642-16949-6_9

Pongthawornkamol, T., K. Nahrstedt and G. Wang, 2007. The analysis of publish/subscribe systems over mobile wireless ad hoc networks. Proceedings of the 4th Annual International Conference on Mobile and Ubiquitous Systems (MobiQuitous 2007), Aug. 6-10, IEEE Explore Press, Philadelphia, PA, USA, pp: 1-8.

Pouwelse, J., P. Garbacki, D. Epema and H. Sips, 2005. The bittorrent p2p file-sharing: Measurements and analysis. Proceedings of the international conference on Peer-to-Peer Systems, Feb. 24-25, Ithaca, NY, pp: 205-216. DOI: 10.1007/11558989_19

Rowstron, A.I.T. and P. Druschel, 2001. Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. Proceedings of the IFIP/ACM International Conference on Distributed Systems Platforms Heidelberg, Nov. 12-16, Germany, pp: 329-350. DOI: 10.1007/3-540-45518-3_18