A Protocol for Content-Based Communication in Disconnected Mobile Ad Hoc Networks

Julien Haillot, Frédéric Guidec

A Protocol for Content-Based Communication in Disconnected Mobile Ad Hoc Networks

Julien Haillot and Frédéric Guidec

Laboratoire VALORIA, Université de Bretagne Sud / Université Européenne de Bretagne, France
E-mail: {Julien.Haillot|Frederic.Guidec}@univ-ubs.fr

Abstract

In content-based communication, information flows towards interested hosts rather than towards specifically set destinations. This new style of communication perfectly fits the needs of applications dedicated to information sharing, news distribution, service advertisement and discovery, etc. In this paper we address the problem of supporting content-based communication in partially or intermittently connected mobile ad hoc networks (MANETs). The protocol we designed leverages on the concepts of opportunistic networking and delay-tolerant networking in order to account for the absence of end-to-end connectivity in disconnected MANETs. The paper provides an overview of the protocol, as well as simulation results that show how this protocol can perform in realistic conditions.

Keywords: Mobile ad hoc network, delay/disruption tolerant networking, opportunistic networking, content-based networking.

Acknowledgements

This work is supported by the French *Agence Nationale de la Recherche* under contract ANR-05-SSIA-0002-01. It is also supported by the French Armament Procurement Agency (DGA) by means of a Ph.D. grant.

1 Introduction

Applications dedicated to information sharing, news distribution, or service advertisement and discovery, all share a common characteristic: they require a communication model where information can flow towards any interested receiver rather than towards set destinations.

Content-based communication is a style of communication that fits perfectly the needs of such applications. In content-based communication, the flow of information is interest-driven rather than destination-driven [5]. Receivers specify the kind of information they

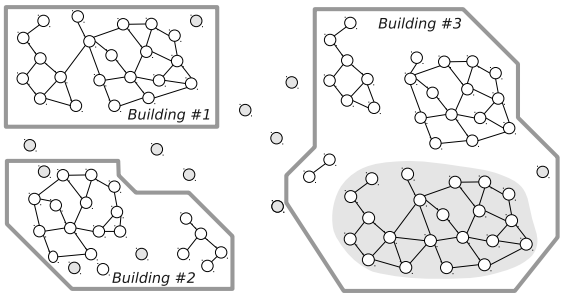


Fig. 1: Example of a disconnected MANET

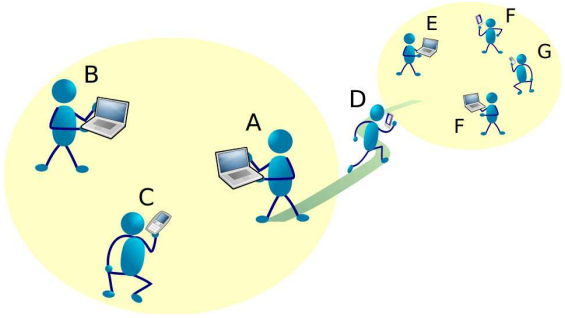


Fig. 2: Detail of Fig. 1, showing how users moving in the campus can carry information between connectivity islands

are interested in, without regard to any specific source. Senders simply send information in the network without addressing it to any specific destination.

In this paper we address the problem of supporting content-based communication in a disconnected mobile ad hoc network (MANET). A MANET can become disconnected when, for example, the mobile hosts that compose the network are very sparsely or irregularly distributed. The whole network then appears as a collection of distinct ``islands''. Communication between hosts that belong to the same island is possible, but no temporaneous communication is possible between hosts that reside on distinct islands. Figure 1 shows a disconnected MANET, which is typical of the kind of network we consider in our work. This MANET is composed of a number of laptops carried by users, which can move in and between buildings (for example, the buildings of a campus). In this example, some laptops are temporarily isolated (either because there is no other laptop within their transmission range, or more simply because they have been put in suspend mode for a while), while other laptops have a number of neighbours, with which they can try to communicate using either single-hop or multi-hop transmissions.

In fully connected wired networks, content-based communication can be achieved by constructing an underlying communication system, whose role is to forward each piece of information from its sender to all interested hosts [5]. This system is usually organised as a logical, content-driven routing infrastructure, which itself can be implemented as an overlay network that covers the whole physical point-to-point network. This approach can hardly be applied in a disconnected MANET, since in such a network the absence of end-to-end connectivity between distinct islands precludes building any network-wide overlay.

In such conditions, a method must be devised in order to bridge the gap between non-connected parts of the network. Delay-tolerant networking is an approach that can help with that respect [12]. In a delay-tolerant network, a message can be *stored* temporarily on a host, in order to be *forwarded* later by this host when circumstances permit. If the network includes mobile hosts---or if all hosts are mobile---, then mobility becomes an advantage, as it makes it possible for messages to propagate network-wide, using mobile hosts as *carriers* that can move between remote---and possibly non-connected---fragments of the network. In a disconnected MANET such as that shown in Fig. 1, people moving

between buildings (or between different parts of a building) can thus contribute to disseminate information between non-connected fragments of the MANET. Figure 2 shows a typical example, where the laptop of a user moving between two groups of users can contribute to carry information between these two groups.

In the remainder of this paper we present the main features of a middleware platform we designed along these lines. An overview of this platform is given in Section 2, and details about this platform's components are provided in later sections. The way information differentiation is performed in this platform is notably described in Section 3. The platform actually supports the opportunistic, content-driven dissemination in a disconnected MANET of structured pieces of information we refer to as *documents* (Section 3.1). Application services running on a mobile host can subscribe for receiving particular kinds of documents by specifying a *pattern* characterizing each category of desired documents. The patterns defined by all the subscribers running on the same host are combined and constitute the host's *interest profile* (Section 3.2). The dissemination of documents in the network relies on a *gossip-based protocol*: transient contacts between mobile hosts are exploited to exchange documents between these hosts, according to their respective interest profiles (Section 4). A host that subcribes for receiving a particular kind of documents is expected to serve as a mobile carrier for these documents. However it can also serve as an altruistic carrier for documents it is not especially interested in, provided this behaviour does not compromise its chances of collecting and carrying interesting documents.

The protocol implemented in the platform is actually defined as a two-layer stack. The *upper layer* (Section 4.1) defines how neighbour hosts can discover each other and exchange documents. It also provides means for storing documents in a local cache, so each host can serve as a mobile carrier while roaming the network. The *lower layer* of the protocol (Section 4.2) makes it possible for hosts that reside in a connected fragment of the network---such as the greyish area in Fig. 1---to interact with n-hop neighbours using temporaneous multi-hop forwarding. This approach helps disseminate documents faster and more efficiently in the network.

The whole protocol was designed so as to be very frugal regarding the resources it consumes, and yet efficient at disseminating documents. Simulation results presented in Section 5 confirm this claim, and show how our platform can perform for disseminating documents in an environment such as that shown in Fig. 1.

Related works---and especially works we took inspiration from---are discussed in Section 6. In Section 7 we conclude this paper, listing a number of directions we contemplate investigating in future work.

2 Overview of the system

Figure 3 provides an overview of DoDWAN (stands for Document Dissemination in disconnected Wireless Ad hoc Networks), the middleware platform we designed in order to support content-based information dissemination in disconnected mobile ad hoc networks. This platform is not just simulation code: it has been fully implemented in Java, and is now being used for developping effective application-level software.

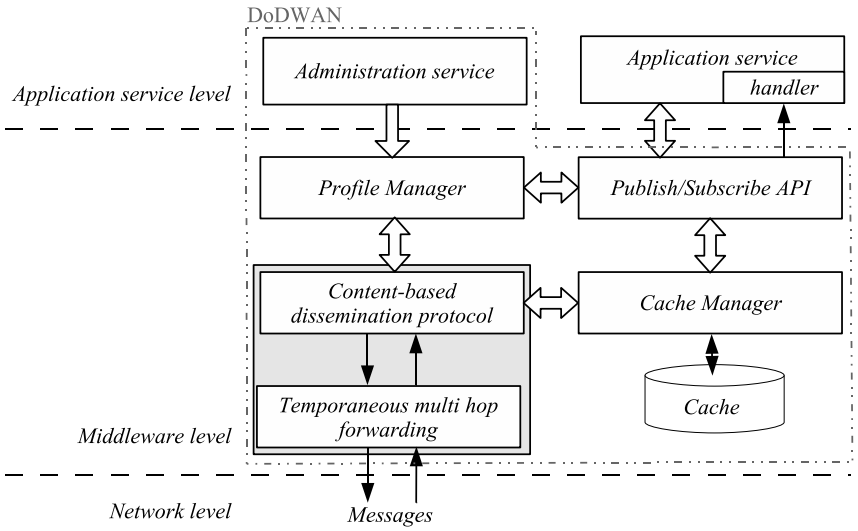


Fig. 3: Architecture of the communication platform

DoDWAN is distributed under the terms of the GNU General Public License[1]. It provides high-level application services with means to publish and subscribe for structured pieces of information we refer to as *documents*, using a dedicated API. Each host is expected to allocate a certain amount of storage space for implementing a local cache, in which documents can be stored for a while. Although we make no assumption about the storage capacity on each mobile host, it is assumed that this capacity is bounded, and that it can actually be different on different hosts. Each host's cache is therefore under the responsability of a local cache manager that can decide what documents can be put in the cache, and that can remove them from the cache whenever needed. Note that in this paper our objective is not to evaluate the merits of different strategies for cache management, as this kind of work has been done (e.g. in [14]).

When a document is published by a local application service through the Pub/Sub API, it is simply put in the host's local cache. Afterwards this document can disseminate from host to host in the network, using as carriers and forwarders those hosts that are willing to help in this dissemination.

Conversely, in order to subscribe for a particular kind of document, an application service must use the Pub/Sub API to specify a pattern that characterizes this kind of document. This pattern is then passed to a profile manager, that keeps track of all local subscription patterns and combines them to define the host's so-called ``interest profile'' (which somehow characterizes the whole set of documents the host is interested in). This interest profile can also be directly altered by a dedicated administration service. Thus, the user or administrator of a mobile host can if needed bypass the standard Pub/Sub API in order to indicate that this host should serve as a mobile carrier for specific categories of documents, even if these documents are of no direct interest to local application services.

The two remaining blocks (that are highlighted by a grey background in Fig. 3) are the key elements to content-based networking in our system. They implement a two-layer protocol we designed to support opportunistic, content-driven interactions between mobile

[1] http://www-valoria.univ-ubs.fr/CASA/DoDWAN

hosts. This protocol can be perceived as a particular implementation of the---somewhat abstract---Autonomous Gossipping (A/G) algorithm described in [10]. It allows mobile hosts to exploit transient contacts for exchanging information according to their respective interest profiles. Interaction between mobile hosts relies on a simple model, whereby each host periodically informs other hosts located in its neighbourhood about its own interest profile and about the documents that are currently available in its local cache. When a host discovers that one of its neighbours can provide a document it is interested in (that is, a document that matches its own interest profile and that is not already available in its own cache), it can request a copy of this document from this neighbour. Transient contacts between mobile hosts are thus exploited opportunistically for exchanging documents between these hosts, based on their respective interest profiles, and based on the documents they can provide each other on demand.

In the remainder of this paper we provide a detailed description of this two-layer protocol we implemented in the DoDWAN platform, motivating the role of each layer, and showing how they together support content-driven gossiping between mobile hosts. In Section 3 we specify how content-based information differentiation is realized in our system and in Section 4 we detail how content-based information dissemination is carried out in our protocol.

3 Information differentiation: the key to content-driven dissemination

Keys to content-driven information dissemination are:

1. the ability to differentiate pieces of information based on their content
2. the ability to specify the kinds of information each subscriber is interested in.

In our system, the ``pieces of information'' we consider are actually referred to as ``documents''. Selection (or filtering) patterns are used to describe what kinds of documents a subscriber is interested in, and interest profiles aggregate the patterns of all subscribers running on a single host, and therefore define the whole set of documents this host is interested in.

3.1 Documents, descriptors and identifiers

A document is actually composed of two parts (see Fig. 4): its descriptor, and its content (or payload). In this paper we use an XML dialect to illustrate most of the data structures we deal with, including messages and message parts. This is actually for the sake of clarity only. In practice, the system we designed uses more effective (i.e. binary) formats for data structures and messages.

The descriptor can be perceived as a collection of attributes, which can provide any kind of information about the corresponding document, such as its identifier, its origin, its topic, a list of keywords, the type of its content, etc. In our system the identifier is the only mandatory attribute in a descriptor. Each document must be assigned a unique identifier

```
<descriptor>
 ...
</descriptor>
<payload>
 ...
</payload>
```

Fig. 4: General structure of a document

```
<descriptor>   // (D1)
 id="254d3g64z36cd"
 service="filesharing"
 type="application/pdf"
 date="Fri Jul 11 09:52:11 CEST 2008"
 deadline="Sat Jul 12 14:00:00 CEST 2008"
 publisher="Fred"
 keywords="mobile,ad hoc,delay-tolerant,\
       opportunistic"
</descriptor>
```

```
<descriptor>   // (D2)
 id="3ab7285ef6548"
 service="news"
 group="comp.networking"
 type="text/rfc850"
 date="Thu Jul 10 10:52:11 CEST 2008"
 publisher="Julien"
 keywords="mobile,ad hoc"
</descriptor>
```

Fig. 5: Examples of document descriptors

value, as this value is used as a means to differentiate documents and to identify duplicate copies of a document. Besides, as a general rule we assume that the size of a document is far greater than that of its descriptor, which is itself significantly greater than that of its identifier (typical orders of magnitude are *O(10 kB)* for a document, *O(100 B)* for its descriptor, and *O(10 B)* for its identifier). Our protocol leverages on this contrast between the size of a document and that of its descriptor or identifier whenever possible in order to minimise the amounts of data exchanged by neighbouring hosts.

Examples of document descriptors are shown in Fig. 5. The first descriptor, labelled *D1* in the figure, is that of a document published within the context of a (hypothetical) filesharing service. It specifies what is the type of the document (in that case it is a PDF document), and it includes keywords characterizing this document. Note that the identity of the publisher is specified in the descriptor (although that is not a requirement), as well as indications about when the document was published and when it should be considered as being obsolete.

The second descriptor *(D2)* is that of a document published within the context of a peer-to-peer newsgroup-based discussion service (see [OppDNA] for the description of a DoDWAN-based application service we actually designed along that line). In that case the newsgroup the document is published in is specified by attribute *group*, and the document's payload type is plain text with a standard RFC 850 (NNTP) header.

According to the principle of content-based networking, any piece of information contained in a document's descriptor can bring useful indication about how this document should be managed by a mobile host. By matching a document's descriptor against its own interest profile, a host can decide if this document should be put in its local cache. Yet indications provided in the descriptor can also help the local cache manager decide what documents should be removed from the cache, and arbitrate between conflicting docu-

```
<pattern>   // (P1)
 service="filesharing"
</pattern>
```

```
<pattern>   // (P2)
 service="filesharing"
 type="application/(gif|jpg|png)"
</pattern>
```

Fig. 6: Examples of selection patterns

ments. For example descriptor *D1* specifies explicitly the deadline of the corresponding document, whereas descriptor *D2* contains no such information. The information provided in *D1* is thus a clear indication that any host that maintains a copy of the document described by *D1* in its local cache can discard this copy after the set deadline. In contrast a host that carries a copy of the document described by *D2* must decide freely when this copy should be removed from its cache.

Document descriptors can also include attributes specifying priority levels, so as to help mobile hosts decide which documents they should preferably maintain in their local cache. Indeed, in this work our objective is to design a system such that, whenever a host is proposed a new document by another host, it is able to decide whether it is worth receiving and storing this document, considering what is already present in its local cache. Again document differentiation is the key to this approach, as it makes it possible for each host to classify documents based on their descriptors, and decide that some documents are actually more important than others.

3.2 Selection patterns and interest profiles

Selection patterns. Document differentiation is performed based on the information available in document descriptors, and this differentiation relies on selection patterns. For practical reasons, we define a selection pattern as a collection of attributes whose values take the form of regular expressions. A pattern is said to *match* a document's descriptor if, for each attribute defined in the pattern, the same attribute exists in the descriptor, and if the value of the descriptor's attribute matches the regular expression of the pattern's attribute. Figure 6 shows two simple patterns. Let us examine how these patterns patterns in Fig. 6 match against the descriptors in Fig. 5. Obviously pattern *P1* matches descriptor *D1*, since attribute *service* is defined and has exactly the same value in *P1* and in *D1*. In contrast *P1* does not match descriptor *D2*, for the value of attribute service in *D2* does not match that defined in *P1*. Pattern *P2* does not match descriptor *D1*. Although *P2* and *D1* both carry the same *service* attribute, the regular expression defined for attribute *type* in *P2* does not match the value of this attribute in *D1* (*P2* actually allows to select only documents that contain images in either JPEG, GIF, or PNG format, whereas *D1* is the descriptor of a PDF document). Finally *P2* does not match *D2*, since the values of the *service* attribute differ in both structures.

Interest profiles. The interest profile of a host determines the different kinds of documents it is interested in, and thus the kinds of documents for which it is willing to serve as a mobile carrier. It is defined as an aggregate of all selection patterns defined by local subscribers (plus possibly additional patterns defined through the platform's administration

```
<profile>   // (host B)
 <pattern>
  service="news"
  group="comp.networking|
       alt.fan.science-fiction"
 </pattern>
 <pattern>
  service="filesharing"
  keywords="mobile|ad hoc"
 </pattern>
</profile>
```

```
<profile>   // (host C)
 <pattern>
  publisher="Fred"
 </pattern>
</profile>
```

Fig. 7: Interest profile of hosts *B* and *C*

service). The interest profile is therefore notably updated whenever a local application service subscribes for a new kind of documents (or cancels a former subscription) through the platform's Pub/Sub API.

Whenever the host is offered a document by one of its neighbours, it must decide whether this document is an interesting one or not. A document's descriptor is said to *match* a host's profile if it matches at least one of the patterns defined in this profile.

Examples of possible interest profiles are shown in Fig. 7. Let us respectively call *B* and C the hosts that present these two profiles. In that case *B*'s profile consists of two basic selection patterns, which indicate that *B* is interested:

-- in documents---or articles, for that matter---published within the context of the newsgroup service and pertaining to any of the two specified newsgroups;

-- in documents published within the context of the filesharing service and characterized by any combination of the specified keywords.

Likewise, *C*'s profile indicates that it is interested in any document that has been published by user Fred.

4 Communication protocol

As explained in Section 2 the gossip-based protocol implemented in our platform is actually defined as a two-layer stack. The upper layer supports the content-driven, delay-tolerant dissemination of documents in the network. It notably provides support for storing documents in a host's local cache, so this host can serve as a mobile carrier for these documents while moving in the network. It also defines how neighbour hosts can interact in order to exchange documents according to their respective interest profiles.

Neighbour hosts are hosts that temporarily reside on the same connected fragment of the network. Interaction between such hosts requires that they be able to communicate using either single-hop or multi-hop transmissions depending on their location in the network. The lower layer of the protocol provides mechanisms for temporaneous, multi-hop forwarding, which is required in the latter case.

Frugal use of the wireless medium. While designing this protocol, we strived to make it as frugal as possible regarding the resources it consumes, and especially regarding its consumption of wireless bandwidth. Both the number and the size of the messages required for disseminating documents are systematically kept at a minimum.

Moreover, in this work we assume that mobile hosts communicate using the Wi-Fi technology, which supports two distinct transmission modes, depending on whether data frames are sent in unicast mode or in broadcast mode. We therefore chosed to rely (as suggested in [1, 30]) on broadcast transmissions rather than on unicast transmissions whenever possible. This point actually needs further explanation, as it is not common to favour broadcast transmission over unicast transmission when dealing with wireless ad hoc communication.

With the Wi-Fi (a.k.a. IEEE 802.11) technology, broadcast transmissions are admittedly less reliable than unicast transmissions. This is mostly because an ARQ (Automatic Repeat-Query) mechanism is implemented at MAC level, and this mechanism is used only when a frame is sent in unicast mode. Sending a unicast data frame is therefore a fairly---though not totally---reliable operation, as the sender keeps re-sending the frame until it receives an ACK frame from the destination host, or until the maximum number of retransmissions has been reached. In contrast sending a data frame in broadcast mode implies no such mechanism. The frame is sent once and once only on the medium. If interferences occur during this transmission, the frame can be lost without the sender and/or receiver(s) being aware of this loss. Sending a data frame in broadcast mode is thus more risky than sending it in unicast mode, but it is also significantly less costly, since it implies no ACK frames, and no retransmission. It also makes it possible to send a message to all direct neighbours of the sender using a single broadcast frame, rather than by using a round of unicast frames addressed to each neighbour successively.

While designing our protocol we decided to favour a frugal consumption of resources, to the detriment of the reliability of transmissions. We therefore use broadcast transmissions whenever possible, while ensuring that the protocol is totally resilient to transmission failures. As a general rule, interactions between neighbour hosts rely on an opportunistic exchange scheme rather than on a strict transactional scheme. Each host only maintains soft-state information about its neighbourhood, and no communication session is ever established between neighbours.

4.1 Support for content-driven, delay-tolerant document dissemination (upper layer)

In our system, interaction between mobile hosts relies on a simple model, whereby each host periodically informs other hosts located in its neighbourhood about its own interest profile and about the documents that are currently available in its local cache. When a host discovers that one of its neighbours can provide a document it is interested in (that is, a document that matches its own interest profile and that is not already available in its own cache), it can request a copy of this document from this neighbour. Transient contacts between mobile hosts are thus exploited opportunistically for exchanging documents between these hosts, based on their respective interest profiles, and based on the documents they can provide each other on demand.

```
<announce>   // (Comprehensive form)
 from="host_id"
 key="24f6g4dq6"
 <profile>
  // As shown in Fig. 7
 </profile>
 <catalog>
  // As shown in Fig. 9
 </catalog>
</announce>
```

```
<announce>   // (Short form)
 from="host_id"
 key="24f6g4dq6"
</announce>
```

Fig. 8: Examples of the two forms of announces (one complete, one short) a host can broadcast periodically

The system is mostly event-based: each host simply reacts to internal events (such as a timer triggering periodic tasks) and external events (such as the receipt of a message). The protocol is presented in pseudo-code in Section A. Details about the main parts of this code are provided below.

Announcing one's catalog and personal interest profile. Each host n_i periodically broadcasts an announce that combines all or part of the following elements:

- the host's identity n_i
- a hash-key k
- a description of its own interest profile *prof*(n_i)
- a catalog *cat*(n_i), which contains the descriptors of locally cached documents that can be of interest to its neighbours

This announce is broadcast as a single control message, whose propagation scope can be set explicitly by the sender (this is explained further in Section 4.2).

An example of an announce is shown in Fig. 8. Note that each announce can actually be broadcast either in a comprehensive (meaning long) form, or in a short form. Let us first consider the elements contained in a comprehensive announce.

By broadcasting an announce that contains its identity, a host allows its neighbours to discover or confirm that it is itself one of their current neighbours. By also inserting its own interest profile in this announce, it lets them know what kinds of documents it is interested in. Conversely, by receiving similar announces from its neighbours, each host can maintain an accurate vision of its neighbourhood, and most importantly about what kinds of documents each neighbour is interested in.

Since the neighbourhood of each host can change continuously, the information it maintains about this neighbourhood must also be updated accordingly. In practice, every time a host constructs a new announce, it forgets everything about its neighbours (see line 8 in the pseudo-code) and starts collecting ``fresh'' information about them. Thus, whenever it must re-construct its catalog (line 3), the profiles used to build this catalog are those of current neighbours, or more precisely those of neighbours it has heard about since the

```
<catalog>  // (built by A based on B's
           // and C's profiles)
  <descriptor>    // (excerpt from D1)
   id="254d3g64z36cd"
   service="filesharing"
   keywords="mobile,ad hoc,\
           delay-tolerant,opportunistic"
   publisher="Fred"
  </descriptor>
  <descriptor>    // (excerpt from D2)
   id="3ab7285ef6548"
   service="news"
   group="comp.networking"
  </descriptor>
</catalog>
```

Fig. 9: Catalog built by *A* according to *B*'s and *C*'s profiles

last time it updated its catalog. With this approach the catalog a host broadcast is continuously adjusted so as to fit specifically the interest profiles of its current neighbours. The cost of broadcasting an announce that contains this catalog is thus kept at a minimum: a host that maintains many documents in its local cache can avoid broadcasting blindly a large catalog on the wireless medium. Instead the catalog only pertains to documents that match its neighbours' interest profiles.

Let us consider a simple scenario for the sake of illustration. Consider the neighbour hosts *A*, *B*, and *C* in Fig. 2. Assume *A* has already received the last round of *B*'s and *C*'s periodic announces. *A* therefore knows that *B* and *C* are (currently) its neighbours. It also knows that *B*'s interest profile and that C's profile shown in Fig. 7. Now assume it is time for *A* to broadcast its own announce again. *A* must thus construct a catalog based on the descriptors of the documents contained in its local cache, while trying to build as small a catalog as possible. Indeed, if *A* maintains in its cache several hundreds or thousands of documents, it does not make sense to construct a large catalog containing all these documents' descriptors, if only a few of these descriptors actually match either *B*'s or *C*'s interest profile. *A* therefore parses the descriptors of the documents contained in its cache, in order to select only those descriptors that match at least one of its neighbours' profiles. Moreover, while parsing these descriptors *A* strives to select only those attributes that are distinctive selection criteria for its neighbours.

Assume *A*'s cache contains the documents whose descriptors are shown in Fig. 5. When parsing its cache looking for documents that might interest its neighbours, *A* can observe that descriptor *D2* matches the first pattern in *B*'s profile (see Fig. 7), and that in this descriptor only the attributes *service* and *group* are distinctive selection criteria for host *B*. Likewise descriptor *D1* matches the second pattern in *B*'s profile, and the attributes *service* and *keyword* are the only distinctive selection criteria for host *B*.

While considering host C's profile, *A* can similarly observe that the only pattern contained in C's profile (see Fig. 7) is matched by descriptor D1, although in that case the distinctive attribute is *publisher*.

Based on these observations *A* can build a catalog such as that shown in Fig. 9 (for the sake of brevity, we assume that host *A* has no other neighbours than *B* and *C*, and

that its cache contains no other descriptor that matches either *B* or *C*'s interested profiles). This catalog is actually composed of excerpts of the selected documents' descriptors, since besides the *id* attribute---which must be included in each descriptor in any case---the only attributes that appear in this catalog are those that can help hosts *B* and *C* decide if they wish to receive the corresponding documents. This is consistent with our objective that the size of messages (including the periodic announce that contains a catalog) should always be kept at a minimum.

Now there are circumstances when the size of an announce can be reduced even further. As mentioned above an announce can be broadcast either in a comprehensive form or in a short form (see Fig. 8). This makes it possible for a host to avoid broadcasting a comprehensive announce, if it considers there is no point in doing so. In that case the host simply broadcasts a short-form announce, using in this announce the same hash-key as in the last comprehensive announce (note that both forms of announces include a *key* attribute, whose value is calculated based on the host's current catalog and profile, and which therefore changes only when a new comprehensive announce is constructed). By doing so it confirms its neighbours that it is still in their neighbourhood, while informing them that, from its viewpoint, the last comprehensive announce it broadcast is still valid. In practise a host can avoid building and broadcasting a new catalog when the following conditions are all verified simultaneously:

-- there has been no significant change in its neighbourhood since it last broadcast a comprehensive announce (more precisely: former neighbours may have disappeared, but no new neighbour has appeared);

-- the interest profiles of all known neighbours have not changed during the same interval;

-- the interest profile of the announcer itself has not changed either during that interval;

-- no new document has been put in the local cache during that interval.

When these conditions are all verified, a host can legitimately assume that the last comprehensive announce it has broadcast has been received and processed by its neighbours, so it can prevent from broadcasting this announce again.

Receiving a neighbour's announce. Upon receiving an announce from one of its neighbours, the receiver behaves differently depending on whether this announce is in comprehensive form or in short form.

Indeed, since all hosts in the network are assumed to move frequently (if not continuously), a host may occasionally receive a short announce from an as-yet-unknown neighbour. Moreover, because of radio interferences a host may fail to receive a comprehensive announce from one of its neighbours, and receive only a subsequent short announce from this neighbour. Our system provides for such situations. Whenever a host constructs a new comprehensive announce, this announce is also put in its local cache (line 6 in the pseudo-code), while the former announce is removed from the cache (line 2). Thus, when a host receives a short announce and realizes that it has missed the corresponding comprehensive announce, it can request that this comprehensive announce be broadcast again (line 16).

```
<request>
 from="host_id"
 to="host_id"
 docIds="254d3g64z36cd ... 3ab7285ef6548"
</request>
```

Fig. 10: Structure of a request for missing documents

If the announce it has received is in comprehensive form, then the receiver first updates its vision of its neighbourhood based on the information (neighbour's identity and interest profile) it has just received (line 13). It then parses the catalog contained in the announce in order to identify documents whose characteristics match its own interest profile, and that are not already in its local cache (line 14). If there exists such documents, then it must actually decide which of these documents it wishes to request from the announcer. The strategy applied for managing the local cache is here of major importance, since it can influence the way the host selects these documents. Admittedly, the host may follow a greedy strategy, systematically attempting to obtain all the documents it is missing from any announcer. Yet, if the local cache is already saturated (or close to saturation), then the host must balance between the documents it is being offered, and those that are already present in its cache. Obviously it would not make sense to request many documents from a neighbour, and realize once these documents have been received that they cannot be stored in the local cache.

As mentioned in Section 2 our current objective is not to compare several cache management strategies. Instead we simply assume that each host implements a function with which it can somehow sort documents based on their sole descriptors. By applying this function to the combined set of all document descriptors it knows about (that is, those present in its local cache, plus those it is being offered by one of its neighbours), a host can decide which of these documents it wishes to maintain in its cache and, most importantly, which of these documents it must request from its neighbour.

Once this list has been defined, the host prepares a request that simply contains the identifiers of the desired documents. This request is then sent to the announcer in a unicast control message (line 15), as shown in Fig. 10.

Processing requests. After broadcasting an announce, a host may receive requests from several of its neighbours. These requests- are processed sequentially (lines 18-22 in the pseudo-code): for each requested document, the host retrieves this document from the local cache, and broadcasts it in the network as the payload of a data message. Notice that this document is broadcast rather than being sent only to the requester in unicast mode. This is because, after broadcasting its catalog, a host may receive a series of requests for the same document (because several neighbours are interested by this single document). In such a case, all the neighbours requesting a single document from the same host can be satisfied with a single broadcast of this document. In order to avoid that consecutive requests for the same document yield a succession of re-transmissions of this document, each host maintains a history of the documents it has broadcast recently. This history

is reset every time the host broadcasts a new announce. With this approach, when several neighbours ask for the same document, this document is broadcast only once in the network.

If a host receives no request after sending an announce, then it means either that it currently has no neighbour at all, or that none of its neighbours is interested in any document it can provide. Another reason might be either that the original catalog broadcast was lost, or that subsequent requests were lost, because of transient radio interferences. Such a failure is non-critical in our system, since a mobile host that misses an opportunity to exchange documents with some of its neighbours will find many other opportunities to do so in the future (and possibly with other newly found neighbours). In any case, if a host receives no request after broadcasting its catalog, then no document will be broadcast unnecessarily. This is consistent with our objective that unnecessary transmissions should be avoided, and especially transmissions of documents, which are assumed to be far larger than their descriptors.

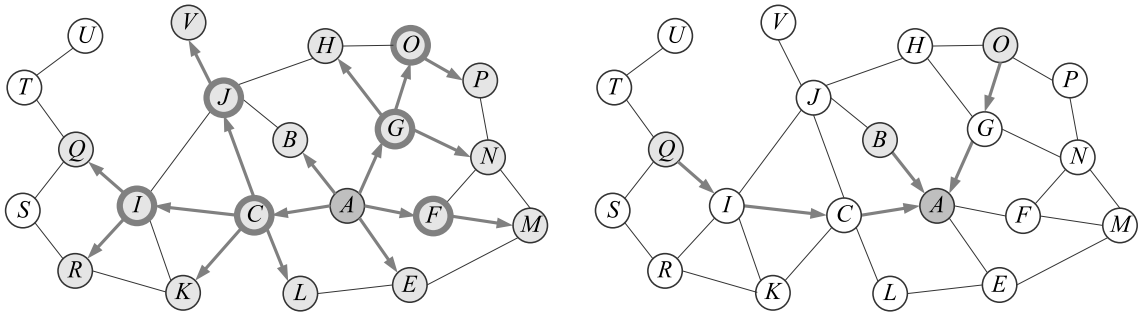
Receiving documents. When a document is broadcast, it can be received by any neighbour of the sender. Any host that receives a document verifies if it is interested in this document by checking whether the document's descriptor matches its own interest profile (line 23). If so, then the receiver attempts to put the document in its cache, while ensuring that this is consistent with its own cache management policy.

Remember that if the host's cache is already saturated, then the host may decide that the newly received document---however interesting it might seem---is however less important than the documents present in the cache. In such a case the newly received document can be passed (if needed) to the local subscriber services, but it is not put in the local cache.

Conversely, if the host finally succeeds in storing the new document in its local cache, then it will thereafter serve as a mobile carrier for this document, thus contributing to help disseminate it further in the network.

An interesting consequence of our preferring broadcast transmissions to unicast transmissions is that it makes it possible for mobile hosts to collect documents just by *overhearing* transmissions initiated by other hosts in their neighbourhood. A host can therefore receive a document ``just by chance'' (that is, just because this document has been recently requested by another host). If this document is indeed an interesting one, then it can be put in the receiver's cache. The receiver has therefore obtained a document without even requesting it, and most importantly it will later refrain from requesting this document from another neighbour.

This possibility for mobile hosts to receive documents without requesting them can actually be exploited further, by allowing them to behave as *altruistic carriers* for some of these documents. The cache management policy of a host can be implemented in such a way that a host accepts to receive and store documents it is not especially interested in, provided this behaviour is obtained at low cost and does not jeopardize its prime objective, which is to collect and help disseminate interesting documents. In practice, an *altruistic* host that receives a non-interesting document can put this document in its cache only if this cache is not saturated (line 24). It will thus help disseminate this document in the network, until the cache becomes saturated and non-interesting documents must be discarded in favour to interesting ones. Note that when a host receives a non-interesting document and



(a) Host A broadcasts a message (via multi-point relays) up to its 3-hop neighbours

(b) Hosts B, O, and Q send unicast messages to host A (using source-routing as a forwarding mode)

Fig. 11: Illustration of the two kinds of temporaneous message forwarding supported by the lower layer of our protocol

decides to store it in its cache for a while, there is absolutely no transmission overhead since the host simply receives ``by chance'' a document that has been broadcast in reply to another host's request. On the other hand, a host that carries non-interesting documents in its cache may have to send these documents to interested neighbours every now and then, which will contribute to deplete its battery. Behaving as an altruistic carrier therefore has an impact on a host's power budget. This is the reason why our system allows mobile hosts to behave as altruistic carriers, while permitting that this option be enabled or disabled on each host depending on the strategy enforced on this host.

4.2 Support for temporaneous message forwarding among neighbour hosts (lower layer)

As explained in Section 4.1, the upper layer of the protocol requires that a host be able to send messages (containing either an announce, a request, or a document) to its current neighbours. Temporaneous message forwarding---as opposed to delay-tolerant forwarding---is thus required in order to exploit transient connectivity between hosts that happen to reside in the same connected fragment of the network for a while. Consider for example the network shown in Fig. 1, and let us focus on the connected fragment (or island) that is marked in grey in this figure. Hosts that temporarily belong to this island can attempt to exploit the connectivity in this island in order to exchange documents through multi-hop transmissions, rather than interacting only with their direct neighbours.

Since the upper layer of our protocol requires that messages be sent either in broadcast mode or in unicast mode, the lower layer of the protocol provides support for temporaneous forwarding of unicast and broadcast messages in a connected fragment of the network.

Broadcast message forwarding --- Multi-hop broadcasting in a MANET is known to be a bandwidth-consuming activity, which can occasionally lead to the so-called ``broadcast storm'' problem. In order to limit the overhead due to message broadcasting, the lower layer of our protocol implements a mechanism that is inspired from that used in the Optimized Link State Routing (OLSR) protocol for diffusing link-level information in the network [6, 26, 18]. Basically, each node regularly selects a subset of its direct neighbours

```
<broadcast_parameters>
 from="host_A"
 nbOfHops="2"
 mpr_set="host_I host_J"
 history="host_A host_C"
</broadcast_parameters>
```

Fig. 12: Information required in each broadcast message in order to limit its propagation scope and have it forwarded by selected MPRs only

```
<announce>   // (Comprehensive or short form)
 ...
 1hop_neighbours="host_id1 host_id2 ..."
 ...
</announce>
```

Fig. 13: Information required for MPR selection (namely the list of 1hop neighbours) is piggy-backed in periodic announces

```
<unicast_parameters>
 from="host_Q"
 to="host_A"
 path="host_I host_C host_A"
</unicast_parameters>
```

Fig. 14: Information required in each unicast message so it can be source-routed toward its destination

as multi-point relays (MPR), and it then relies exclusively on these MPRs for forwarding broadcast messages beyond its own radio coverage. The scope of a broadcast can be controlled by specifying how many hops a message is allowed to perform while being relayed by MPRs. Figure 11-a shows how a message can be broadcast within the greyish island shown in Fig. 1. In this example host *A* needs to broadcast a message, which could be for example an announce containing its profile and catalog. This message is allowed to propagate up to its 3-hop neighbours, but not further. Figure 12 shows the parameters inserted in each broadcast message so it can be processed and forwarded only by selected MPRs. In that case we assume that the message has already reached host *C* (one of the MPRs selected by *A*), which must now forward this message via its own MPRs *I* and *J*.

The algorithm used by each host to construct its MPR set is not detailed in this paper for the sake of brevity, and because this algorithm is very similar to that described in [26]. Basically, each host must periodically broadcast a control message in order to inform its direct (one-hop) neighbours about its presence in the network, while informing these neighbours about its own current vision of its 1-hop neighbourhood. By receiving such control messages, each host can identify its one-hop and two-hop neighbours, and use this information to calculate its MPR set. With the approach described in [26], specific control messages are broadcast periodically, that contain the information needed for calculating MPR sets. In our implementation, this information is piggy-backed in the announces the upper layer of the protocol must also broadcast periodically (see Fig. 13). Thus the calculation of MPR sets does not imply sending any additional message in the network: both kinds of control information (required by both layers of the protocol) are broadcast together on the wireless medium.

Unicast message forwarding --- The upper layer of the protocol requires that mobile hosts be able to send requests as replies to an announce they have just received. Unicast

messages must thus be forwarded towards the sender of a broadcast message. Source-routing is used as a means to perform this forwarding. Each broadcast message that propagates in the network encapsulates a history of the hosts by which it has been forwarded so far (see Fig. 12). Thus, whenever the receiver of a broadcast message decides to reply to this message, the path for sending this reply to its source is simply deduced from the path the former broadcast message has followed before reaching the receiver. Note that, in order to be effective, this approach requires that when a host decides to reply to a broadcast message, this reply is sent immediately after the broadcast message has been received. In such conditions, the path the broadcast message has followed downwards to reach the receiver is still valid in the network, so it can be followed upwards to the sender of the broadcast message.

Consider again the example shown in Fig. 11-a, and assume that hosts *B*, *Q*, and *O* decide to reply to the message broadcast by *A*. Figure 11-b shows how their replies can propagate upwards along the path the broadcast message has just followed downwards, each reply containing a specification of the path it must follow before reaching host *A*. Figure 14 shows the parameters that must be inserted in a request sent by host *Q* so it can be forwarded upwards to host *A*.

5 Evaluation

Our protocol for content-driven, delay-tolerant communication has been fully implemented in Java, and embedded within the DoDWAN middleware platform (as explained in Section 2). DoDWAN makes it possible to implement and experiment with different kinds of applications (such as filesharing, news distribution, messaging, etc.) in disconnected MANETs. To date it has been deployed and used extensively on up to thirty laptops with Wi-Fi capability. Yet, since it is quite difficult to run experiments with dozens or hundreds of mobile devices, DoDWAN was designed so it can also be interfaced with the MADHOC simulator [15]. Based on this combination we run a number of simulations in order to observe how the protocol can perform in different conditions, using the experience we acquired previously during real-conditions experiments to define the simulation parameters. In this section we present some of the results we obtained by performing series of 14.000 second simulation runs, with the parameters and communication scenario described below.

5.1 Simulation conditions

Simulation parameters. We consider a simulation scenario in which a population of 120 users move in an environment that resembles that shown in Fig. 1. In that particular scenario, we actually consider a set of 5 buildings which are located within a 1 km 1 km area. Each building has a rectangular shape, with edges between 100 and 150 meters long. Each user is assumed to carry a laptop equipped with an IEEE 802.11 (Wi-Fi) interface.

The mobility of users---and therefore that of the mobile hosts they are carrying---is simulated using a variant of the random waypoint model: a user can remain motionless for a while, afterwards he/she begins to walk towards a set destination, which is selected randomly in any one of the buildings in the simulation area.

```
<profile>
  <pattern> topic="Ti|Tj" </pattern>
</profile>
```

Fig. 15: Profile of a host interested in documents pertaining to topics *Ti* and *Tj*

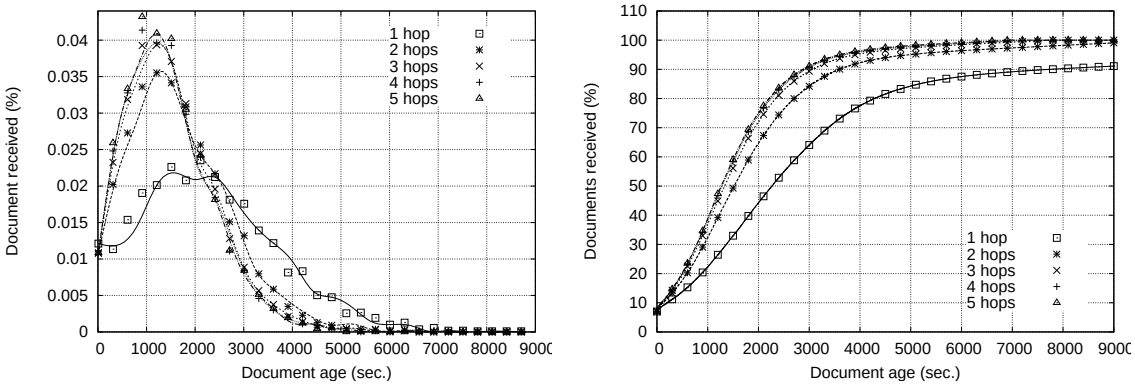
In the simulation runs whose results are discussed below, we used the following mobility parameters: users are assumed to walk at speeds varying between 0.5 m/s and 2 m/s (that is, typical pedestrian speed); a stay between two consecutive moves can last between 30 seconds and 3 minutes; and the amount of intra-building mobility is set to 40 % against 60 % for inter-building mobility. Wi-Fi interfaces are assumed to have an omni-directional transmission range of 40 meters when used indoor, and 100 meters when used outdoor. All these parameters are consistent with observations we made while experimenting with DoDWAN in a real campus environment and with real users.

Communication scenario. We consider a communication scenario whereby all mobile hosts continuously produce new documents and publish these documents in the network. Each document weighs 50 kB, and each host publishes one new document every 5 minutes. As a whole, documents are thus published in the network at an average global rate of one new document every 2.5 seconds. Topic-labelling is used as a simple means to differentiate documents: there are 16 different topics labelled *T0* to *T15*, but each document is tagged as pertaining to only one topic.

Each mobile host is assumed to be interested in documents pertaining to only two distinct topics (hence $1/8$ of the global amount of documents published in the network). The interest profile of a host is thus defined as shown in Fig. 15. No two different hosts in the network have exactly the same interest profile.

Protocol parameters. Our protocol can be adjusted by setting two main parameters. The first parameter is the period with which a host broadcasts an announce (in either comprehensive or short form, depending on circumstances). We set this period at 15 seconds, for experience with DoDWAN in real conditions proves that this value is generally adequate in a MANET where hosts move at pedestrian speeds. Of course a shorter (resp. longer) period could be used if the hosts moved faster (resp. slower) and were expected to experience shorter (resp. longer) contacts with each other.

Another parameter is the maximum number of hops used in temporaneous message forwarding, and most notably when a host broadcasts an announce. By adjusting this parameter, we can somehow extend the ``sphere of communication'' of each host, controlling the scope of the announces it broadcasts periodically, and therefore the number of neighbours with which it is liable to exchange documents before moving to another part of the network.



(a) Distribution of the age of documents at delivery-time

(b) Smoothed cumulative distribution of the age of documents at delivery-time

Fig. 16: Distribution and cumulative distribution of the age of documents at delivery time

5.2 Simulation results

Our first objective is to show how the scope of temporaneous message forwarding can influence the gloabl performance of our protocol. The expected result is that, when a mobile host is allowed to use multi-hop forwarding in order to interact with a large number of neighbouring hosts, documents can disseminate faster than when each host can only exchange documents with direct (one-hop) neighbours.

Speed of document dissemination. We first consider a---somewhat unrealistic---scenario where documents can propagate eternally in the network. We notably assume that the cache capacity on each host is unlimited, and that no document is given a set lifetime by its publisher. Moreover we assume, for the time being, that the option for ``altruistic behaviour'' (as described in Section 4) is disabled on each host, so that it is only willing to collect, carry, and forward documents that match its own interest profile.

The mobility model used during the simulation ensures that each host eventually gets close to any other host in the network. In such conditions, a document that can propagate forever in the network is guaranteed to eventually reach any interested receiver. Yet the time before this document is delivered to an interested receiver can be influenced by the protocol parameters, and notably by the scope of temporaneous message forwarding.

In Fig. 16 we observe how long it takes for documents to reach interested receivers. More precisely, Fig. 16-a shows the normalized distribution of the age of these documents at delivery time, and Fig. 16-b shows the corresponding cumulative distribution.

Let us first consider the case where the hosts can only use 1-hop transmissions. In such circumstances it can be observed that about 40 % of the documents are delivered in less than 30 minutes. After an hour, about 75 % of the documents have reached their receivers, and after two hours about 90 % have been delivered.

Let us now observe how multi-hop forwarding can influence the performance of document dissemination. Fig 16-a shows that, when temporaneous 2-hop forwarding is used (that is, when each host is allowed to interact with its 1-hop and 2-hop neighbours), most documents are received after about 20 minutes (against 30 minutes when only 1-hop for-

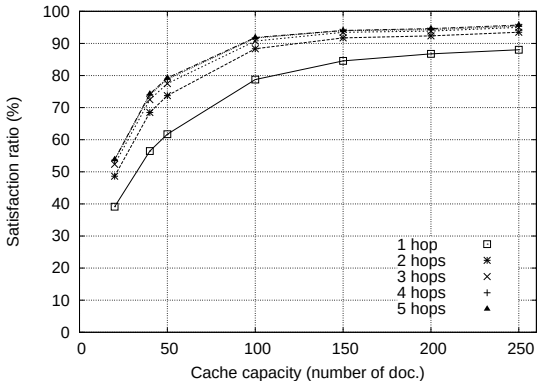


Fig. 17: Satisfaction ratio (of document delivery) vs. cache capacity

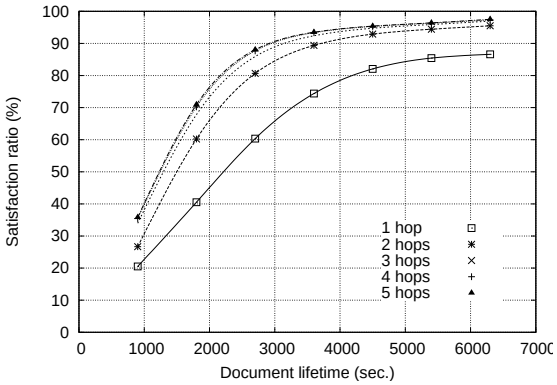


Fig. 18: Satisfaction ratio (of document delivery) vs. document lifetime

warding is used). In such conditions about 98 % of the documents are actually received in less than two hours, about 90 % in less than an hour, and about 60 % in less than 30 minutes.

A similar---though comparatively minor---improvement can be observed when multi-hop forwarding is pushed further, so that each host is allowed to extend its sphere of communication up to its 3-hop, 4-hop, and 5-hop neighbours respectively. Indeed, with the simulation parameters used during this experiment, the islands (or connected fragments of the network) that can form in the buildings have a limited extension. Their elongation varies between 0 (isolated hosts) and 7 hops, with an average value of 4.2 hops. This explains why extending the sphere of communication of each host beyond a couple of hops does not bring much improvement. Another reason is that the propagation of documents between different buildings (or between non-connected parts of a building) depends primarily on how fast document carriers---that is, pedestrians in the scenario considered---actually move in the simulation area.

In any case, this first experiment confirms that by extending the sphere of communication of each mobile host our protocol allows documents to disseminate better and faster in each island, thus increasing the number of hosts that can then serve as carriers between non-connected parts of the network.

Cache capacity. In the simulation runs whose results were discussed above, we assumed that documents could propagate forever in the network. As mentioned above this is not very realistic, since most resources in a MANET are usually severely constrained. For example the cache where mobile hosts can store documents is of limited capacity. An adequate policy must thus be devised---and then enforced on each host---in order to deal with saturation conditions.

Figure 17 shows how the capacity of each host's cache can influence the performance of document dissemination. To obtain these results we run a series of simulations, considering cache capacities ranging between 50 and 200 documents. During each simulation the cache policy enforced was such that, when a cache reached saturation, the oldest document in this cache was discarded in order to make room for a new document. In the figure

we plot the satisfaction ratio (that is, the percentage of documents that are eventually delivered to interested receivers) against the capacity of the cache. First, Fig. 17 confirms the natural expectation that a host with a larger cache is liable to carry documents further and longer in the network.

More interesting is the influence of temporaneous multi-hop forwarding on the performance of document dissemination. In Fig. 17 it can be observed that the satisfaction ratio of document delivery increases significantly when the scope of message forwarding is extended to a couple of hops around each host. Consider for example the case where each host can only maintain 100 documents in its cache. In such conditions, the documents sent in the network are received (on average) by only 78 % of the interested receivers if each host is only allowed to interact with direct neighbours. Yet this figure is increased by 10 % when the scope of temporaneous forwarding is extended to 2-hop neighbours, and again by 2 % when it is extended to 3-hop neighbours.

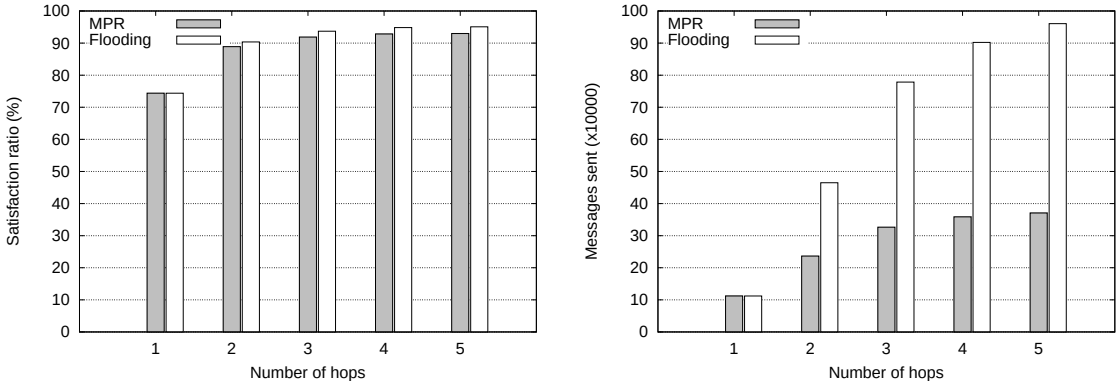
Document lifetime. Another way to prevent documents from remaining eternally in the hosts' caches is to give each document a set lifetime, so that whenever a document gets obsolete it is automatically removed from any cache it might have been stored in. This method can be used either as a substitute or as a complement to the method that limits the capacity of each cache.

Figure 18 shows how different values of document lifetime influence the performance of document dissemination. These results were obtained with unbounded cache capacity, so that the two types of constraints do not interfere during the simulation. In the figure we plot the satisfaction ratio (percentage of documents that are delivered to interested receivers) against the set lifetime of documents. Not surprisingly, the satisfaction ratio increases as documents are given a longer lifetime. Yet it can again be observed that temporaneous multi-hop forwarding gives significant improvement in document dissemination. For example, when documents are given a 30-minute lifetime, they are eventually received (on average) by only 40 % of the interested receivers if each host is only allowed to interact with direct neighbours. Yet this figure is increased by 20 % when the scope of temporaneous forwarding is extended to 2-hop neighbours, and again by 7 % when it is extended to 3-hop neighbours.

Communication overhead. The above results confirm that by resorting to temporaneous multi-hop forwarding, the dissemination of documents in the network can be made faster, and thus more efficient. They also show that even a slight extension of the sphere of communication of each host (by only two or three hops in the scenario considered) can bring a significant improvement over a situation where a host can only interact with direct neighbours.

The drawback of multi-hop forwarding is that it yields an important overhead in terms of the resources it mobilises on each host. Indeed, whenever a host forwards a message, this transmission drains the battery of this host, while occupying the shared wireless medium around this host.

While designing our protocol we decided to rely on MPR-based forwarding for broadcasting messages around each host. Obviously it would have been a lot easier for us to use plain flooding for broadcasting these messages. Since above-mentioned results show that



(a) Satisfaction ratio. (b) Transmission cost.

Fig. 19: Comparison of the MPR-based and flooding-based versions of the protocol

messages need only be forwarded on a limited scope (typically, two or three hops) it is worth wondering whether MPR-based forwarding brings any benefit over plain flooding in such conditions.

In order to evaluate the difference between our approach relying on multi-point relays and an alternate one relying on plain flooding, we implemented a variant of our protocol that uses plain flooding as a means to broadcast messages around each sender. The results are presented in Fig. 19. They were obtained when running our communication scenario during four hours (in simulation time), with unlimited cache capacity and 1-hour document lifetime.

It can be observed (Fig. 19-a) that the MPR-based and flooding-based versions of the protocol do not give exactly the same satisfaction ratio. This is because the MPR-based version is slightly slower at disseminating documents in the network. Indeed, with this version a host whose neighbourhood changes needs to wait a while (precisely, two consecutive announce cycles) because it can effectively interact with its new neighbours. In contrast, with the flooding-based version of the protocol a host whose neighbourhood changes can immediately reach its new neighbours.

The satisfaction ratio observed with the MPR-based version of the protocol is therefore slightly lower than with the flooding-based protocol. Yet this difference remains under 3 %, while the cost of using one or the other way of broadcasting messages is very different. Figure 19-b shows how the cost of transmissions compares with both versions of the protocol. Obviously our decision to rely on multi-hop relays for forwarding broadcast messages is fully justified, as the global number of messages sent when using multi-hop relays is far below that observed when flooding messages in the network.

Adaptive catalog. In Section 4 we have claimed that our protocol has been designed so as to consume as little resources as possible. We have notably described how the catalog each host inserts in its periodic announces is constructed so as to match exactly the interest profiles of its current neighbours. Figure 20 shows how the size of the catalog broadcast by a particular host evolves over time, depending on whether this host actually has neighbours, depending on these neighbours' interest profiles, and depending of course on the documents it already maintains in its cache. The results presented in Fig. 20 were

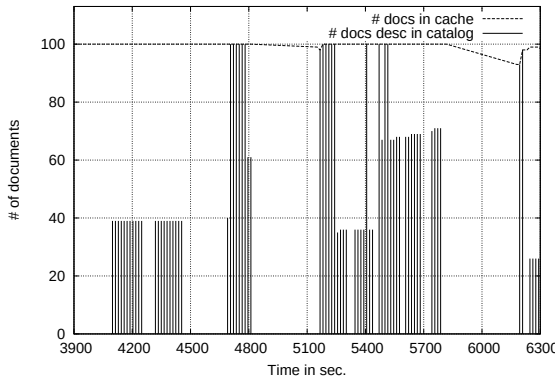


Fig. 20: Evolution of catalog size and cache

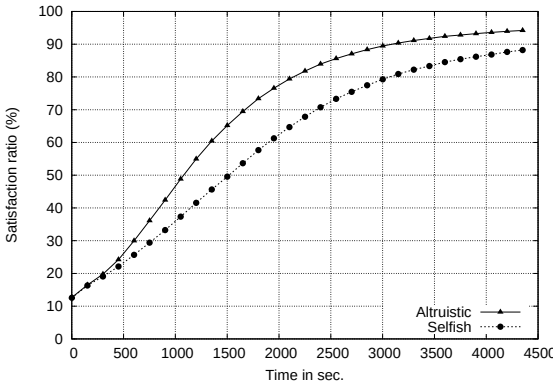


Fig. 21: Satisfaction ratio (of document de-occupancy over time on a single mobile hostlivery) over time when mobile hosts behave either altruistically of selfishly

obtained during a simulation where the capacity of the cache on the considered host was set to 100 documents. Each document was given a 75 minute lifetime, and the scope of multi-hop forwarding was set to 2 hops.

In the figure we can first observe that, during the interval considered, the cache is almost continuously full. Cached documents are discarded as soon as they become obsolete, but new documents obtained from neighbour hosts fill in the gap soon afterwards. Yet the number of descriptors inserted in the catalog the host constructs periodically (at most every 15 seconds) is often smaller than the number of documents it maintains in its cache, and sometimes falls down to 0 (empty catalog). This is because the host builds its catalog by selecting only documents that can interest its neighbours. Sometimes the host has no neighbour at all, sometimes its neighbours have interest profiles that do not intersect its own, so that it cannot propose them any document they might be interested in. Sometimes its neighbours are interested in only a small subset of the documents it maintains in its cache, so the catalog only concerns this subset.

These results confirm that by adapting continuously its catalog based on its neighbours' profiles, a host can contribute to reduce the weight of its periodic announces and, more generally, the amount of work expected from any neighbour that must receive and analyse these announces. Of course, our efforts for reducing the size of each host's catalog proves even more effective when each host maintains a large number of documents in its cache, while its neighbours present highly selective interest profiles. In some simulation scenarios (not detailed here) we have considered hosts capable of maintaining up to 10.000 documents in their cache, whereas the selectivity of their neighbours' profiles was such that only a very small fraction (actually less than 1 %) of these documents had to be proposed in each catalog. The fine-tuning of each catalog proves very profitable in such circumstances.

Altruistic behaviour. In Section 4 we have explained how each host can be configured so as to behave as an altruistic carrier for documents it is not especially interested in, without compromising its chance of collecting documents that match its own interest

profile. Yet this possibility has not been used in the simulation runs whose results have been presented so far. Let us now observe how the global performance of document dissemination is affected when hosts are allowed to behave altruistically.

In Fig. 21 we observe how the satisfaction ratio of document delivery evolves over time, depending on whether the mobile hosts adopt either an altruistic or a selfish behaviour. A host is said to behave selfishly when it only accepts to store, carry, and forward only documents it is itself interested in. This corresponds to the behaviour we have actually considered in al lthe results presented so far. A host is said to behave altruistically when it accepts to receive and store in its cache documents it overhears on the wireless medium, even though these documents present no interest to him. The results presented in Fig. 21 were obtained during a simulation where the capacity of each host's cache was set to 300 documents, which in that particular case is slightly larger than the capacity required for carrying only interesting documents, and therefore allows that the remaining space be used for carrying non-interesting ones. Each document was given a 75 minute lifetime, and the scope of multi-hop forwarding was set to 2 hops.

In this figure we can observe that when the hosts are allowed to behave as altruistic carriers, documents can indeed disseminate faster---and therefore more efficiently---in the network. For example, with selfish hosts the documents can be received on average by 57 % of all interested subscribers in less than 30 minutes, whereas with altruistic hosts this figure is about 72 % (hence a 15 % improvement).

Of course, this observation confirms again the natural expectation that documents disseminate better when they can be transported by a larger number of mobile carriers. Yet it is worth recalling that, with our approach, this improvement comes at very little cost, since each host basically collects non-interesting documents by overhearing their transmission on the wireless medium, and since an altruistic host never removes an interesting document from its cache in order to make room for a non-interesting one.

6 Related work and discussion

The concept of content-based networking has originally been introduced in [5]. Since then it has been refined in a number of papers such as [4] and [3]. In [3] the authors notably propose several levels of predicate languages that can be used to filter messages based on their content, including languages that apply regular expressions either to attribute names or to flat messages. In our system, document differentiation currently relies on a rather simplistic model: differentiation is performed by comparing document descriptors only (rather than the whole content of these documents), and a subscriber's selection predicate is defined as a conjunction of regular expressions that only apply to attribute values. Improving the expressiveness of this model is one of our objectives in the near future.

Many papers have been published in the last few years that address the problem of supporting communication in disconnected MANETs [32, 24]. Some of these papers actually assume that mobility patterns are known in advance or can be controlled as needed

(e.g. [33, 19]), while others make no such assumption and propose to rely on redundancy in order to improve the reliability of delay-tolerant transmission. In the latter category, it is usually proposed to rely on more or less controlled forms of epidemic or probabilistic propagation schemes [29, 11, 23, 27, 28]. Some papers specifically consider communication between user-carried devices, and propose to drive message forwarding in the network by predicting how users move or meet, or by identifying what communities each user belongs to [17]. Indeed, in these papers the basic assumption is that users tend to exhibit regular mobility and/or social interaction patterns, which can be identified (more or less automatically) and then used to select the best carriers for messages addressed to a particular user. For example [20] defines a probabilistic approach whereby the probability to deliver a message to its destination is calculated based on a delivery predictability metric that is derived from the history of node encounters. Similarly, [22] attempts to predict node contacts, using a model of prediction over time series that allows to forecast co-location probability. [2] proposes a context-based approach, whereby each host must maintain a history of context information pertaining to each host (or user) it has encountered in the past. Whenever a message is sent in the network the sender must provide meta-information about the destination (such as the recipient's residence or work address), so this information can be matched against that available in each potential carrier's history in order to calculate delivery probabilities. [16] describes a protocol for social-based forwarding, whereby user communities are identified automatically (using an approach similar to that described in [17]), and users that belong to the same community as a message recipient are selected as best carriers for that message.

In most of the above-mentioned papers the objective is to reach a set destination, specified by the sender. In contrast in content-based communication the sender does not necessarily know who the recipients of its message are, or even if they exist at all. Several papers about content-based communication have already been published, but the algorithms and protocols they define can only be used in stable, wired networks, or in fully connected MANETs [9, 21, 25, 13]. These papers usually propose to construct and maintain content-based *routing structures* in order to forward messages efficiently between publishers and subscribers. A notable exception with that respect is the protocol defined in [1]. Like ours this protocol does not attempt to build any structure to support routing decisions. Instead it too relies on broadcast transmissions, while deferring to hosts that receive a message the decision to forward this message to potential subscribers, based on an estimation of their distance to these subscribers. Yet this protocol requires that temporaneous end-to-end paths exist between senders and receivers. It could not run satisfactorily in a disconnected MANET.

Content-based dissemination in disconnected MANETs is addressed specifically in [8], which describes an approach whereby a content-driven multi-hop routing structure (limited to a given horizon) is built around each host. A utility-based function is used in order to select the best forwarders for each kind of message, and mobile carriers help disseminate messages between non-connected parts of the network. Our protocol relies on a slightly different approach. Instead of attempting to construct and maintain a routing structure, it relies on periodic broadcast transmissions (also limited to a given ``horizon'' from the sender), whereby each host periodically informs its neighbours about the documents it is carrying and that match their interest profiles. Upon receiving such a catalog a host can

request the transmission of a document it is actually missing. Thus no document is sent in the network unless it has been requested explicitly by a client host.

[10] defines the Autonomous Gossipping (A/G) algorithm, that allows neighbour hosts to opportunistically exchange documents they are missing, based on their respective advertised profiles. In the A/G algorithm, information dissemination is actually depicted as an epidemic process: each host is considered as being more or less vulnerable to being ``infected'' by one or another kind of data item. One difference between our protocol and the A/G algorithm is that the latter only relies only on direct interactions between one-hop neighbours, whereas ours supports interaction in connected fragments of the network through multi-hop transmissions. Simulations show that this possibility for a host to reach n-hop neighbours makes the dissemination of information more effective when islands actually appear in the network, as it helps compensate for the selectivity of each host's interest profile. Moreover, to the best of our knowledge the A/G algorithm was never actually implemented (except as a simulator), whereas our protocol has been fully implemented in a middleware platform, so it can now run either in real conditions, or coupled to a simulator.

[31] proposes to use a clustering algorithm to create a Publish/Subscribe overlay, in which centrality nodes somehow behave as brokers between message publishers and subscribers. [7] describes a protocol for Publish/Subscribe that is derived from the protocol for unicast routing presented in [22], but that can account for the interests of users. This protocol exploits predictions based on metrics of social interactions to identify the best message carriers. In fact, the basic assumption is that users with common interests tend to meet with each other more often than with other users. The routing algorithm exploits this property by selecting as carriers for messages hosts which have often been co-located with interested subscribers in the past.

Admittedly there are circumstances when people with similar interests tend to meet regularly. This is for example the case when these people are co-workers (or fellow students), or when they are members of the same family, the same sports club, or the same game club. Yet there are also cases when the fact that people share similar interests does not imply that they are members of the same closely-knit---or even loosely-knit---community. For example people with a keen interest in football or rugby do not necessarily meet very often. Indeed some of them meet frequently in stadiums, but many others simply watch matches on TV. People who wish to keep informed about weather forecasts or about weekly TV programs do not necessarily meet very often either. Similitude between their interest profiles simply means that they can share information occasionally, but this similitude is hardly correlated with their mobility or co-location pattern. In such conditions it is very unlikely that any history-based approach (that basically attempts to predict future movements and/or contacts between users based on an observation of past mobility and/or contacts) can prove very efficient.

Another problem with history-based prediction techniques is that they do not scale up very well, because of the overhead implied by history maintenance. Imagine for example that every citizen in a medium-size city---say 20.000 inhabitants---carries a digital device capable of short-range, ad hoc communication (such as a PDA or smart-phone with a built-in Wi-Fi interface). To the best of our knowledge, it is still unclear whether history-based algorithms, in which each host must continuously collect and maintain data about any

other host it meets while roaming the network, can scale up to such a large network.

It is our conviction that although history-based prediction techniques can prove very efficient in small networks, there is also a need for systems that can run in larger networks. We claim that the system we designed can indeed run in a large disconnected MANET, as it does not attempt to build any history of a mobile host's encounters with other hosts. Indeed, in our system each host only maintains very little information about its neighbours, and forgets everything about them as soon as its neighbourhood changes. The main limitations of the system are therefore the number of neighbours a host can have at any time (although this constraint mostly depends on the characteristics of the underlying wireless technology), and the number and size of the documents it can exchange with neighbours while they are co-located.

In order to prove our claim that our system can indeed perform satisfactorily in a large network, it would be most interesting to run simulations with a large number of mobile hosts. Unfortunately, the simulator we used for evaluating our system is not distributed and must therefore run on a single workstation. The CPU speed and memory available on this workstation are thus the limiting factors during simulations. With this simulator we could actually run simulations with up to 10.000 hosts, but in that case each host could only maintain a very small cache, so the number of documents disseminated during these simulations was not very impressive. We could also run simulations with more than 100.000 documents disseminating in the network, but then it is the number of hosts that was limited. Raising this constraint, so we can simulate realistic scenarios in large networks, is an important item in our agenda.

7 Conclusion

In this paper we have presented a new system for content-based communication in disconnected MANETs. Unlike other protocols that rely on costly methods for constructing and maintaining content-driven routing structures, ours does not attempt to build any such structure. Instead it exploits transient contacts between mobile hosts that get close enough to one another, allowing these hosts to exchange documents according to their respective interest profiles. Communication between non-connected fragments of the network is performed thanks to mobile hosts, each host serving as a carrier for documents it maintains in a local cache. In our system a host is primarily expected to collect and carry information it is itself interested in, but it can also behave as an altruistic carrier for non-interesting documents, as long as this behaviour does not compromise its chance of collecting interesting ones. Simulation shows that our protocol is effective at propagating documents between senders and interested receivers. Its use of temporaneous multi-hop forwarding helps disseminate documents in connected fragments of the network, which in turn has a positive influence on this dissemination in the whole, disconnected network. By adjusting the extension of multi-hop forwarding around each host, the resulting transmission overhead can be balanced against the benefit observed in document dissemination. With the current version of the protocol the number of hops used when broadcasting messages is set as a constant parameter. In the future we plan to investigate methods allowing each host to adjust this value dynamically, accounting for its current situation in the network (e.g. number and density of neighbours, interest profiles of these neighbours, history of

recent document exchanges in the neighbourhood, etc.). We also consider improving the way document differentiation is achieved by allowing more elaborate forms of attribute filtering.

References

[1] R. Baldoni, R. Beraldi, M. Migliavacca, L. Querzoni, G. Cugola, and L. Migliavacca. Content-Based Routing in Highly Dynamic Mobile Ad Hoc Networks. *Journal of Pervasive Computing and Communication*, 1(4):277--288, Dec. 2005.

[2] C. Boldrini, M. Conti, J. Jacopini, and A. Passarella. HiBOp: a History Based Routing Protocol for Opportunistic Networks. In *Proceedings of the IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks*, pages 1--12. IEEE, June 2007.

[3] A. Carzaniga and C. P. Hall. Content-Based Communication: a Research Agenda. In *Proceedings of the 6th international workshop on Software Engineering and Middleware*, pages 2--8. ACM, Nov. 2006.

[4] A. Carzaniga, M. J. Rutherford, and A. L. Wolf. A Routing Scheme for Content-Based Networking. In *Proceedings of IEEE INFOCOM 2004*, pages 918--928. IEEE, Mar. 2004.

[5] A. Carzaniga and A. L. Wolf. Content-based Networking: a New Communication Infrastructure. In *Proceedings of the NSF Workshop on an Infrastructure for Mobile and Wireless Systems*, number 2538 in LNCS, pages 59--68. Springer, Oct. 2001.

[6] T. Clausen and P. Jacquet. Optimized Link-State Routing Protocol (OLSR). IETF, RFC 3626, Oct. 2003.

[7] P. Costa, C. Mascolo, M. Musolesi, and G. P. Picco. Socially-Aware Routing for Publish-Subscribe in Delay-Tolerant Mobile Ad Hoc Networks. *IEEE Journal On Selected Areas In Communications (JSAC)*, 26(5):748--760, June 2008.

[8] P. Costa, M. Musolesi, C. Mascolo, and G. P. Picco. Adaptive Content-based Routing for Delay-tolerant Mobile Ad Hoc Networks. Technical report, UCL, Aug. 2006.

[9] P. Costa and G. P. Picco. Semi-Probabilistic Content-Based Publish-Subscribe. In *25th International Conference on Distributed Computing Systems (ICDCS 2005)*, pages 575--585. IEEE, June 2005.

[10] A. Datta, S. Quarteroni, and K. Aberer. Autonomous Gossiping: a Self-Organizing Epidemic Algorithm for Selective Information Dissemination in Mobile Ad-Hoc Networks. In *Semantics for Grid Databases, First International IFIP Conference (ICSNW 2004)*, number 3226 in LNCS, pages 126--143. Springer, June 2004.

[11] P. Eugster, R. Guerraoui, A.-M. Kermarrec, and L. Massouli□ From Epidemics to Distributed Computing. *IEEE Computer*, 37(5):60--67, May 2004.

[12] K. Fall. A Delay-Tolerant Network Architecture for Challenged Internets. In *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 27--34. ACM, Aug. 2003.

[13] U. Farooq, S. Majumdar, and E. Parsons. High Performance Publish/Subscribe Middleware for Mobile Wireless Networks. *International Journal of Mobile Information Systems*, 3(2):107--132, 2007.

[14] K. A. Harras, K. C. Almeroth, and E. M. Belding-Royer. Delay Tolerant Mobile Networks (DTMNs): Controlled Flooding in Sparse Mobile Networks. In *IFIP Networking Conference, Waterloo, Ontario, CANADA*, volume 3462 of *LNCS*, pages 1180--1192. Springer, May 2005.

[15] L. Hogie, P. Bouvry, and F. Guinand. The MADHOC simulator. http://www-lih.univ-lehavre.fr/ hogie/madhoc.

[16] P. Hui, J. Crowcroft, and E. Yoneki. BUBBLE Rap: Social Based Forwarding in Delay Tolerant Networks. In *Proceedings of the 9th ACM International Symposium on Mobile Ad Hoc Networking and Computing*, pages 241--250. ACM, May 2008.

[17] P. Hui, E. Yoneki, S.-Y. Chan, and J. Crowcroft. Distributed Community Detection in Delay Tolerant Networks. In *Proceedings of the 2nd ACM/IEEE international workshop on Mobility in the evolving internet architecture*, pages 1--8. ACM, Aug. 2007.

[18] M. Ikeda, L. Barolli, G. De Marco, T. Yang, A. Durresi, and F. Xhafa. Tools for Performance Assessment of OLSR Protocol. *International Journal of Mobile Information Systems*, 5(2):165--176, 2009.

[19] Q. Li and D. Rus. Sending Messages to Mobile Users in Disconnected Ad-hoc Wireless Networks. In *Proceedings of the Sixth Annual International Conference on Mobile Computing and Networking*, pages 44--55, Aug. 2000.

[20] A. Lindgren, A. Doria, and O. Schelen. Probabilistic Routing in Intermittently Connected Networks. In *Proceedings of the First International Workshop on Service Assurance with Partial and Intermittent Resources (SAPIR 2004)*, volume 3126 of *LNCS*, pages 239--254. Springer, Aug. 2004.

[21] R. Meier and V. Cahill. STEAM: Event-Based Middleware for Wireless Ad Hoc Network. In *International Conference on Distributed Computing Systems, Workshops (ICDCSW '02)*, pages 639--644. IEEE, July 2002.

[22] M. Musolesi, S. Hailes, and C. Mascolo. Adaptive Routing for Intermittently Connected Mobile Ad Hoc Networks. In *Proceedings of the IEEE 6th International Symposium on a World of Wireless, Mobile, and Multimedia Networks*, pages 183--189. IEEE, June 2005.

[23] M. Musolesi, C. Mascolo, and S. Hailes. Emma: Epidemic messaging middleware for ad hoc networks. *Personal Ubiquitous Computing*, 10(1):28--36, Aug. 2005.

[24] L. Pelusi, A. Passarella, and M. Conti. Opportunistic Networking: Data Forwarding in Disconnected Mobile Ad Hoc Networks. *IEEE Communications Magazine*, 44(11):134--141, Nov. 2006.

[25] M. Petrovic, V. Muthusamy, and H.-A. Jacobsen. Content-Based Routing in Mobile Ad Hoc Networks. In *Proceedings of the 2nd Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services (MobiQuitous'05)*, pages 45--55. IEEE, July 2005.

[26] A. Qayyum, L. Viennot, and A. Laouiti. Multipoint Relaying for Flooding Broadcast Messages in Mobile Wireless Networks. In *Proccedings of the 35th Annual Hawaii International Conference on System Sciences (HICSS'02)*, pages 3866--3875. IEEE, Jan. 2002.

[27] Y. Sasson, D. Cavin, and A. Schiper. Probabilistic Broadcast for Flooding in Mobile Ad Hoc Networks. Technical Report IC/2002/54, Swiss Federal Institute of Technology (EPFL), 2002.

[28] T. Spyropoulos, K. Psounis, and C. S. Raghavendra. Spray and Wait: an Efficient Routing Scheme for Intermittently Connected Mobile Networks. In *2005 ACM SIGCOMM workshop on Delay-tolerant networking (WDTN'05)*, pages 252--259. ACM, Aug. 2005.

[29] A. Vahdat and D. Becker. Epidemic Routing for Partially Connected Ad Hoc Networks. Technical report, Duke University, Apr. 2000.

[30] E. Vollset, K. Birman, and R. van Renesse. Chickweed: Group Communication for Embedded Devices in Opportunistic Networking Environments. In *3rd International Workshop on Dependable Embedded Systems, in conjunction with 25th Symposium on Reliable Distributed Systems (WDES 2006)*, Oct. 2006.

[31] E. Yoneki, P. Hui, S.-Y. Chan, and J. Crowcroft. A Socio-Aware Overlay for Publish/Subscribe Communication in Delay Tolerant Networks. In *Proceedings of the 10th ACM/IEEE International Symposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM)*, pages 225 -- 234. ACM, Oct. 2007.

[32] Z. Zhang. Routing in Intermittently Connected Mobile Ad Hoc Networks and Delay Tolerant Networks: Overview and Challenges. *IEEE Communications Surveys and Tutorials*, 8(1):24--37, Jan. 2006.

[33] W. Zhao, M. Ammar, and E. Zegura. A Message Ferrying Approach for Data Delivery in Sparse Mobile Ad Hoc Networks. In *Proceedings of ACM Mobihoc 2004*, pages 187--198, May 2004.

A Appendix: pseudo-code of the content-based dissemination protocol

Variables

$self$: node's own identifier.
$profile$: node's own interest profile.
$\mathcal{C}$: node's document cache.
$announce$: node's last comprehensive announce.
$neigh < keyprof > [i]$: contains a tuple composed of the last hash-key, profile, and catalog received from node i.
$\mathcal{S}$: set of identifiers of documents that have been broadcast recently.
$altruistic$: boolean flag. True if the node's altruistic behaviour is enabled.

Messages

ANNOUNCE$< nkeyprofcat >$: comprehensive form of the announce a host can send periodically. It contains the host's id, a hash-key, and the node's profile and catalog.
ANNOUNCE$< nkey >$: short form of the announce a host can send periodically. It only contains the node's id and a hash-key (whose value is the same as that of the last comprehensive announce sent by this host).
REQUEST$< docIds >$: message requesting the broadcast of the documents whose ids are specified in list *docIds*.
DOCUMENT$< descdata >$: message containing a document (descriptor and data).

Functions

broadcast(m): broadcast message m.
send(m, n): send message m to destination n.
hashKey(c, p): compute hash-key based on a host's catalog c and profile p.
createCatalog(neigh, C): create a catalog by selecting in cache C documents that match the current neighbours' profiles.
identifyMissingDocs(c, p, C): process catalog c based on profile p, and return a list of identifiers of interesting documents mentioned in c that are not already in cache C.
conditionsHaveChanged(): return true if something has changed that justifies updating the host's announce (i.e. at least one new neighbour has been discovered, a neighbour's profile has changed, a new document has been put in the local cache, or the node's profile has changed).

```
--- Periodic announce ---
send(ANNOUNCE)
 1 if conditionsHaveChanged() then
       // Prepare a comprehensive announce
       // (and store a copy in cache)
 2     C = C - {announce}
 3     cat = createCatalog(neigh, C)
 4     key = hashKey(cat, profile)
 5     announce = ANNOUNCE<self, key, profile, cat>
 6     C = C U {announce}
 7     msg = announce
 8     neigh = 0;
 9 else
       // Prepare a short announce
10     msg = ANNOUNCE<self, key>
11 S =
12 broadcast(msg)

--- Invoked on receipt of a comprehensive announce ---
receive(a: ANNOUNCE<n, key, prof, cat> )
13 neigh[n]=<key,prof>
14 docIds = identifyMissingDocs(cat, profile, C)
15 send(REQUEST<docIds>, n)

--- Invoked on receipt of a short announce ---
receive(a: ANNOUNCE<n, key>)
16 if neigh[n].key != key then
       // Failed to receive the comprehensive version
       // of this announce. Requesting one.
17     send(REQUEST<key>, n)

--- Invoked on receipt of a request ---
receive(r: REQUEST<docIds>)
18 forall id in docIds do
19     if id notin S then
           // A document is sent only once during a period
20         DOCUMENT d = C.get(id)
21         S = S U {id}
22         broadcast(d)

--- Invoked on receipt of a document ---
receive(d: DOCUMENT<desc,data>)
23 if ((desc.matches(profile) AND (d notin C))
24 OR (altruistic AND (C.notFull()))) then
25     C = C U {d}
```

Algorithm 1: Protocol functions