Chapar: A Cross-Layer Overlay Event System for MANETs

Amir R. Khakpour and Isabelle Demeure

Ecole Nationale Supérieure des Télécommunications (TELECOM ParisTech)
46, rue Barrault, 75634 Paris Cedex 13, France
{amir.khakpour,isabelle.demeure}@telecom-paristech.fr

Abstract. In this paper, we present *Chapar*, an event system designed for mobile ad hoc networks that supports the publish-subscribe model as well as point-to-point and point-to-multipoint message sending. Chapar supports event persistency to resist transient disconnections and network partitioning. Following a cross-layer approach, Chapar is designed as an overlay network that uses the Multipoint Relays (MPRs) defined in OLSR as distributed brokers, and uses the OLSR routing table to disseminate the events. It therefore benefits from the way OLSR handles topology changes. The implementation performance is promising in the sense that no extra signaling is generated by mobility support and the generated overlay traffic is considerably less than the underlying routing protocol.

1 Introduction

A **M**obile **A**d hoc **NET**work (MANET)[1] is a self-configuring network of mobile nodes connected by wireless links. The nodes are mobile therefore the network topology changes over time. This infrastructureless nature nominate MANETs to be used for a set of new spontaneous services in domains such as military, fire fighting and gaming. However, MANETs are prone to frequent network disconnections and to network partitioning. Also, nodes may enter and leave the network and the network should be able to dynamically adapt to these fluctuating conditions. In these conditions, the use of the traditional client-server model that relies on the server accessibility should be avoided; symmetrical (distributed) models using asynchronous communications that are more robust to transient disconnections should be preferred. Hence, asynchronous publish/subscribe systems receive a great deal of attention in the realm of MANETs.

Contribution. In this paper, we present "Chapar"[1], a novel event system designed for MANETs that uses OLSR routing protocol [2] for event dissemination. OLSR is often said to have limited scalability, but it is well adapted to the small size ad hoc networks with relatively high traffic dissemination that we target in our project. Chapar supports the publish/subscribe model, where no destination address(es) is assigned to the event (the system delivers the events to the correponding events subscribers). It further supports point-to-point and point-to-multipoint message dispatching. In order to address

[1] Chapar is the first Mail and Message Delivery Service known in the History. It belonged to the Persian Empire hundred years B.C. [Reference: Allyn Huntzinger, "Persians in the Bible", Global Commission Inc, 2004.]

C. Giannelli (Ed.): Mobilware, LNICST 0007, pp. 325–339, 2009.

transient disconnections and network partitioning, Chapar supports event persistency (an event may be kept in replicated data containers until its expiration time elapses or it is delivered to all its subscribers). Contrary to other event systems [3,4] that rely on a single broker to handle event publications and subscriptions, Chapar uses a distributed approach to implement the event broker. This is to avoid having a single point of failure and a performance bottleneck (the single broker and its links to neighbors)[10]. Following a cross-layer approach, Chapar operates as an overlay network that uses the OLSR Multipoint Relays (MPRs) as brokers, and uses the OLSR routing table to disseminate the events. Using the underlying routing layer enables us to constitute the virtual multicast trees (with no additional signaling) to deliver events instead of using expensive unicast communication and flooding which is not scalable.

In addition, we show that Chapar yields good performance in particular, because it causes considerably less network overhead than other solutions. We also investigate the event system traffic in different network density conditions and compare the results with OLSR routing messages traffic. The proposed system considers node mobility and changes in network topology is handled by OLSR routing protocol.

Organization. The remainder of this paper is organized as follows. In Section 2, we review the requirements for the event layer that were specified within the Framework of the Transhumance project and that we addressed in the design of the Chapar system. Section 3 provides an overview on related work. We then present the data structures and algorithms used in Chapar, in Section 4. Section 5 is dedicated to Chapar evaluation and performance analysis. We conclude in Section 6.

2 Requirements and Definitions

Chapar is an event management system designed and implemented as part of the French National Research Agency (ANR) funded Transhumance Project [5], [6].

In this scheme, the event system is required to support the point-to-point, point-to-multipoint (group communication), publish-subscribe system and a combination of these. Following the *record-based* event model [7], the events are defined as composite messages including several attributes describing the event content. The subscription is done accordingly, by expressing the interest into these attributes or topics [8].

In Chapar, the events are either destroyed after they are dispatched (*real-time* events), or stored in the broker network for their lifetime (*memorized* events). The memorized events are notified not only to the subscribers who are absent at the event publishing time and eventually come back, but also to the subscribers who subscribed in the event lifetime period (time decoupling support).

The underlying routing protocol is OLSR, a proactive and table-driven protocol. The OLSR routing table held by each node contains the list of nodes available in the network along with their corresponding *next hop* nodes through which the node is accessible. OLSR is a link-state routing protocol using a set of nodes called *Multipoint Relays(MPRs)* to connect nodes to their 2-hop neighbors. In OLSR, nodes periodically send *HELLO* messages to their neighbors to advertise their link status. The HELLO messages also contain the nodes' neighbor set. This is used by neighbors to determine their own MPR set and also to identify if they are MPR or not. Each MPR periodically

generates and sends *Topology Control* (TC) messages to all of the nodes. It contains the list of nodes that the TC sender has chosen as its MPR (MPR Selector Set (MS)). Finally, each node is able to calculate its routing table based on received TC messages.

3 Related Work

There are many related work regarding event systems and Message-Oriented Middleware (MOM) [23,24,25]; in this section, we focus on publish/subscribe systems for mobile networks such as JEDI [7] and the systems presented in [9], [10],[11], and [12].

We identify three strategies for event dissemination in distributed publish-subscribe systems: (1) event forwarding, (2) subscription forwarding and (3) hierarchical forwarding. In *event forwarding* (aka *message flooding*), an event is forwarded through an acyclic graph, called *dispatching tree*, to all nodes in the network. Those whose subscription matches the event are notified and others just forward the event to the next hop in the tree. Although, this strategy supports mobility (if the tree is modified by the new node location), it is not scalable. With this strategy, it would be impossible to store events in order to support subscribers that may be unreachable at publishing time. This makes the event system unreliable when network partitioning occurs.

In *subscription forwarding*, a subscriber sends a subscription message to all its neighbors and each node holds a table including the subscriptions that it received. Once an event is published, it is checked against all the subscriptions and then is forwarded to the neighbor who has forwarded or generated the corresponding subscription. This strategy is employed by *filter-based event routing* [13] which is used by many content-based event systems such as SIENA [14] and the systems described in [8], [15], and [16]. Since the subscription tables are based on the nodes' neighbors, these protocols cannot support mobility and frequent network topology changes. To cope with this problem, some mobility extensions are provided such as the *MoveInMaster* and *MoveOutMaster* operations that allow SIENA to build a new multicast tree and reroute the events to the displaced nodes.

Finally in the *hierarchical forwarding* strategy, events are notified based on a rooted tree topology. In this tree which includes all of the nodes in the network, the subscription messages are forwarded upwards the "root" in the dispatching tree and then routed downwards to the subscribers. This strategy is used by JEDI to forward events from publisher to subscriber(s). In JEDI, nodes called *Dispatching Servers* (DSs) are organized in a tree and forward subscription messages to the root. When a published event is handed over to the root, each DS verifies if any of their descendants has subscribed to this event and forwards them a copy if appropriate. Handling mobility, similar to the SIENA mobility extension, JEDI provides *MoveIn* and *MoveOut* operations.

However, to the best of our knowledge, none of the proposed publish-subscribe systems provide a comprehensive solution addressing network partitioning which is quite likely to happen especially in sparse networks. In addition, the aforementioned publish-subscribe systems mainly support push-based [17] publish-subscribe system (except for JEDI) and they are not able to store events to be *pulled* later by the subscribers.

Some other related works concerns applications and services on mobile ad hoc networks using overlay network with cross-layer approach (using routing layer information). Delmastro et al. [18] proposed *CrossROAD*, an API on P2P system on mobile networks using the OLSR routing information as the underlying layer to handle mobility and achieve better performance in terms of reducing the overhead of the overlay data structures. Similar work presented in [19] uses the same concept for group communication within nodes in a mobile ad hoc network. Chapar however is quite different. We are proposing a publish/subscribe system building upon the underlying routing protocol.

4 Chapar Data Structures and Algorithms

The main idea behind Chapar is to use the OLSR MPRs as the event system *Broker Nodes (BN)*. OLSR is therefore used to support mobility, to enable self-configuration of the broker network, to provide one-hop publishing and notification, and to build the virtual multicast trees from publisher to subscriber(s).

In this section, we first present the main data structures used in Chapar. We then explain the main algorithms in the event system.

4.1 Tables and Data Structures

Chapar uses three main data structures: the Node Vector (NV) and two tables maintained by the broker nodes (BNs) namely *Filter Mapping Table (FMT)* and *Memorized Event Table (MET)*. We describe each data structure in turn.

The *Nodes Vector (NV)* is a bitmap where each bit represents one node in the network. The basic assumption is that the nodes addresses follow an ordering such that we are able to define a hash function $H_{NV}() : \{a_1, a_2, \cdots, a_N\} \rightarrow [1..N]$ to map the address to a specific bit index (figure 1(a)). For instance, if we assign an IPv4 class C address pool for at most ($N = 255$) nodes in the network, the $H_{NV}()$ is defined to return the last byte of the node address.

If identifying $H_{NV}()$ is not feasible due to randomness of the nodes addresses, an alternative solutions called Bloom Filters [22] could be used. However, this solution is costly, because to have negligible false positive error, an 8 times longer bit string bitmap is required. Moreover, using logical operations to calculate the list of nodes for each step may not be possible any more [20].

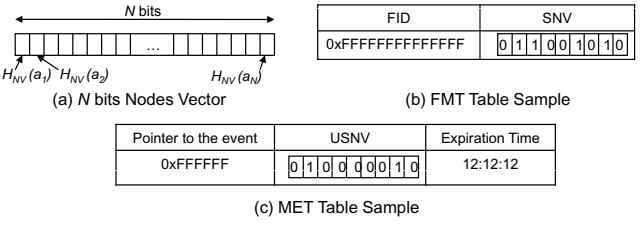


Fig. 1. Chapar's different data structures examples

Each broker node (BNs) maintains a *Filter Mapping Table (FMT)* to keep the subscription information. In this table, each filter is mapped to the list of the subscribers represented by an NV called Subscribers Nodes Vector (SNV). The second table maintained by BNs is the *Memorized Event Table (MET)* that holds the memorized messages with their corresponding Unavailable Subscribers Nodes Vector (USNV). Figure 1(b) and 1(c) show examples of FMT and MET tables respectively.

The FMT table is modified upon the arrival of subscriptions and un-subscription messages, whereas the MET table is updated upon the arrival of a memorized event and when an absent subscriber returns (USNV purging).

4.2 Chapar Functions and Algorithms

In this section, we present the main functions of the publish/subscribe system and their underlying algorithms.

Subscription/Un-subscription: Each subscriber dispatches the subscription message to one BN in its neighborhood. In case it is already the BN, the subscription message is looped back to itself.

A subscriber defines a filter by expressing its interest in some event attributes. Each subscription message contains a *Filter ID (FID)*. The FID is a hash string that is calculated as concatenation of hash result of each attribute. Note that if the subscriber does not express its interest in a specific attribute, the hash result for that attribute is 0.

Once a BN (or MPR) receives the subscription message, it adds the subscription to its FMT table, and forwards it to all neighbor broker nodes. This process is repeated until all the BNs update their FMT table by the new subscription. The un-subscription messages are treated similarly, yet has inverse effect on the FMT table.

Figure 2 demonstrates an example of an event and subscription/unsubscription messages in Transhumance middleware. This structure follows the record-based event systems where different fields of the event describe its characteristics. However, the fields may differ for each application. In fact the only required fields in the event are the Event Destination NV (EDNV) field that represents the event destination nodes, and the event class flags by which different types of the events are distinguished. The required fields

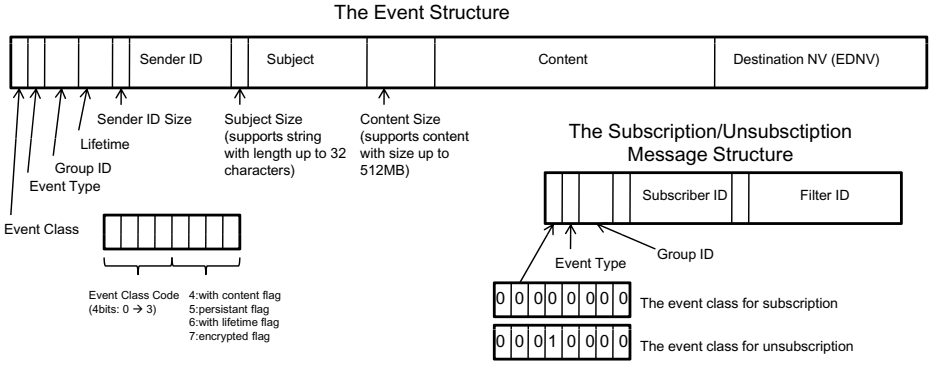


Fig. 2. The examples of the event and subscription/unsubscription message in Transhumance

Algorithm 1. Lookup Process Algorithm

1: **for** each filter in FMT **do**
2: **if** (corresponding FID) $\wedge$ (EID) $\neq$ (corresponding FID) **then**
3: Filter's corresponding SNV $\vee =$ the filter SNV;
{*(it is to calculate the union of all of the SNVs of those filters matched against the incoming event)*}
4: **end if**
5: **end for**
6: $EDNV = ASNV = SNV \wedge ANV$ {(ASNV= Available Subscribers Nodes Vector and ANV= Available Nodes Vector)}

for the subscription/unsubscription messages are the event type, the filter ID, and subscriber ID that are used for updating the FMT.

Publishing and Notification: Publishing and notification are one-hop communication. The publisher sends an event to one of its BNs. The event is then notified by the BN to its adjacent subscriber. If the publisher or the subscriber is a BN, the publishing and the notification are done through self-publishing and self-notification, respectively.

The real-time events are disseminated through a multicast tree from the publisher BN to the subscribers, whereas the memorized events are flooded in the BN network up to all BNs receive them and keep them in their MET until their lifetime is elapsed.

The broker network functionality description: Once the published event is received by the first BN, it is looked up in the FMT (Lookup Process) to specify who the event's subscribers are. Then the event EDNV field is set by the subscribers who are available for notification. The event subsequently is ready to be notified (Event Notification Process) or to be forwarded to other BNs (Event Multicasting Process).

Lookup Process: Upon the arrival of an event coming from the publisher, the Event ID (EID) is calculated by concatenating the hash result of the event attributes (same as FID). The hash result of each attribute is compared using Lookup Process Algorithm 1.

In the lookup algorithm, we are investigating whether the event (or event ID) matched against any filter ID or not. We basically verify if (FID[i] $\wedge$ EID ==FID[i]) is held or not for each record (filter) i in FMT. Using this algorithm, the false positive errors are possible, however, the more bits we assign for the hash results, the less the probability of false positives. False positive is calculated using the following formula[2]:

$$P_{false\ positive} = \frac{\sum_{i=1}^{n-1} \binom{n}{i} \cdot (2^i - 1)}{2^{2n}} \tag{1}$$

where n denotes the hash result length for each attribute. Formula (1) shows that we can increase n to minimize the false positive error. For example, the false positive error is reduced to ≈ 0.01 when $n = 16$. Note that we assume that the hash function is uniformly random.

[2] The formula detail justification and proof is available in [20].

Taking advantage of FID and EID definition, Chapar supports dissemination of the encrypted events. Since the subscription in all types of the publish/subscribe systems is based on the value of different attributes, publish/subscribe systems usually do not support security features and encrypted events. However, in Chapar the encrypted event could be sent along with its EID so the first BN does not need to calculate the EID from event attributes and proceeds with the Lookup Process by the EID embedded in the event.

Note that this Lookup Algorithm is designed based on the project specification. However, other filters and filter matching algorithms can be used to determine the SNV.

Event Notification: The event notification is a one-hop event delivery. Thus, the BN who has received the event checks whether the event's subscriber(s) is itself, and/or is one of its non-BN neighbors. Let NSNV represents the set of nodes that are notified in current step, HNV represents the broker Host NV, NNV represents the set of neighbors, and BNV represents the preceding broker who sent the event. Therefore,

$$NSNV = EDNV \ \wedge \ (HNV \ \vee \ (NNV \oplus BNV)) \tag{2}$$

When the NSNV nodes are notified, the new EDNV is calculated as follows:

$$EDNV_{new} = EDNV_{old} \oplus NSNV \tag{3}$$

The next step, Event Multicasting, proceeds with new EDNV.

The event notification for memorized events follows the same procedure for the nodes that are available at the publishing time. For the absent nodes, the BN MET table periodically calculates the new NSNV to notify the absent subscribers that are present in its neighborhood. The NSNV is calculated as follows:

$$NSNV = USNV \ \wedge \ NNV \tag{4}$$

And subsequently the new USNV is:

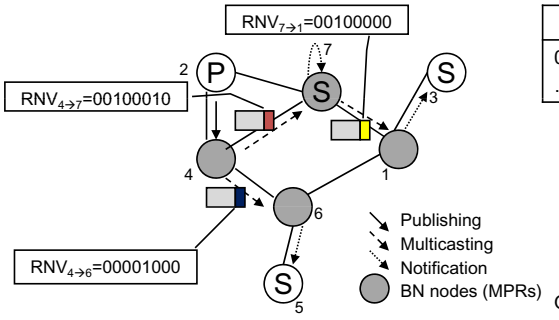
$$NSNV_{new} = USNV_{old} \oplus NSNV \tag{5}$$

To avoid event duplication for a returned absent node that may frequently connect to different BNs, a periodic MET purging process is also required. Thus, the USNV should be recalculated as follows based on the available nodes in the network (ANV):

$$USNV_{new} = USNV_{old} \oplus (USNV_{old} \ \wedge \ ANV) \tag{6}$$

Event multicasting and flooding: Memorized events, similar to the subscription messages, are flooded in the BN network. Every BN stores a copy of the event in its MET table. However, the real-time events are delivered through a virtual multicast tree to the subscribers and the events are destroyed as soon as they are delivered.

As it is mentioned, once an event is published and forwarded to the publisher BN, the event EDNV field is set after the lookup process. The BN then execute the Event Notification process to notify the events to its neighbors that are in fact the event subscribers. It then recalculates the EDNV using (3). In this stage, the BN requires to forward the



FID	SNV
0xFB010050	00110011
…	//m=8 nodes

FMT of node 4

Dest	Next	Hops
1	7	2
2	2	1
3	7	3
5	6	2
6	6	1
7	7	1

OLSR Routing Table for node 4

// node 4 is the root in multicast rooted tree (first MPR connected to publisher)
ANV = 11111110 //node 8 is missing
SNV = 00101011
ASNV = 00101010 //ASNV = ANV & SNV

NNV_4 = 01000110	NNV_7 = 11010000	NNV_6 = 10011000	NNV_1 = 00100110
BNV_4 = 00000110	BNV_7 = 10010000	BNV_6 = 10010000	BNV_1 = 00000110
$NSNV_4$= 00000000	$NSNV_7$ = 00000010	$NSNV_6$ = 00001000	$NSNV_1$ = 00100000

Fig. 3. Chapar real-time event publishing, multicasting and notification based on the routing table

event to the rest of the BNs for event delivery. Thus, it groups the subscribers based on their corresponding next-hop node (using the BN routing table). For each group, the EDNV is changed to the group members represented by an RNV (Residual NV) and is handed over to the corresponding next-hop. This procedure is repeated on each BN until the event is notified to all available subscribers. In fact in this paradigm, for each event a rooted multicast tree is built where each group represents a branch in the tree. This multicasting scheme supports mobility, as the multicast tree is built for each published event virtually using the current routing table. Note that we assume that the event propagation delay is less than the network mobility rate.

Figure 3 shows how an event (real-time event) published by node 2 is multicasted to the subscribers. In this example, an event is received by node 4 (node 2 MPR or BN). Node 4 looks up the event in its FMT table. Once the event ID is matched to a filter(s) it derives the SNV to know who are the subscribers (nodes 3, 5, 7, and 8). Node 4 then calculates ASNV, using its routing table to identify the hosts who are subscribers and available at the moment (nodes 3, 5, and 7). The calculated ASNV is set as the event destination (EDNV). Node 4 then calculates NSNV, the set of subscribers using formula (2) that it can notify before it forwards the event to next hop (NSNV in this stage contains no node). Node 4 then calculates the new EDNV (3) (in this example same as the old EDNV) and groups the subscribers into two groups: one group is the set of subscribers that can be reached by node 7 (nodes 3 and 7), and one group is the set of subscribers that can be reached by node 6 (node 5). For each group, the EDNV is changed to RNV and the event is sent to next corresponding hop. This process is repeated in node 7 and 6. Finally the event is notified to all subscribers.

For the events with the designated destination (*i.e.*, point-to-point (P2P) and point-to-multipoint (P2MP)), the root of the multicast tree is the event publisher node. However, for the *multimode* (combination of the pub/sub system, P2P and P2MP), the tree root is the event publisher broker node, but the primitive EDNV is calculated as follows:

$$EDNV_{new} = EDNV_{original} \vee ASNV \tag{7}$$

Special cases: As pointed out, the OLSR MPRs are used to form the broker network. However, in some network topologies, there could be no MPR in the network. For instance, a full-mesh network where all of the nodes are adjacent and can communicate through one-hop communication does not have any MPR. Also in some cases, the broker network is reduced to one node; this should be avoided since the availability of at least two brokers for redundancy purposes is one of the basic requirements. Thus, for these special cases which are detectable by all nodes thanks to nodes' routing table [20], we need an election mechanism to assign some nodes as broker. In the election mechanism we proposed for the Transhumance Project, nodes (at least two) with the *lowest* index in the ANV are appointed broker nodes. Another special case which is likely to happen addresses the single nodes send subscription messages or dispatch memorized events while they are secluded from the network temporarily. Because the middleware is not concerned with node connection status, these messages and events will be lost. To avoid this loss, when nodes do not contain any host in their routing table (single-node status), they are automatically appointed BN and perform self-subscription and self-publishing and save the information in their local tables. When they reconnect to the network, the FMT and MET changes are propagated to rest of the nodes through the Table Consistency Check Process.

Table Consistency Check: Due to network partitioning and node transient disconnections, divergence of the nodes' tables is inevitable. To maintain table consistency, a periodic process is used to synchronize the content of the tables of different BNs.

In this process, each BN calculates the hash of each table and dispatches it to its adjacent BNs. The incoming hash strings including MET hash and FMT hash are checked by the adjacent BN's tables hash result, if MET hash is different, the receiver sends the list of all MET records. On the other hand, if FMT hash is different, the receiver sends the list of hash of each FMT record. Then, the node compares the received lists with its own tables, if it has a record which is not included in the lists, for memorized message the event will be resent, and for FMT, the complete FMT record will be sent.

The repetition of such process helps BNs to complete each other's tables by information they have and their adjacent BNs do not. Note that using hash in this process provides us with a light signaling for table comparison.

The table consistency check induces some table false positive errors in FMT. These false positive errors occur when one BN (node A) receives an unsubscription message while another BN (node B) does not. In this case, the correct table should not include that specific subscription. However, after table consistency check process, node A's FMT will update the node B's FMT with an obsolete subscription. The probability of this kind of table false positive errors depends on the size and the density of the network, the likelihood of message loss, and the rate of dispatching unsubscription messages. Nonetheless, this error has negligible impact on the performance of the event system in which some events may be notified to some nodes which are not the actual subscribers.

Inheritance and Dismissal (Check-out Process): Following the self-configuring goals, we provide an inheritance function by which a non-BN node may retrieves the tables

from one of its adjacent BNs while it is appointed as BN. On the other hand, when a BN is dismissed, it drops the tables it has. However, there may be some information in the tables that other BNs do not have. Hence, a *Check-out* process is required before any table elimination. The Check-out process is done similarly to the table consistency check to notify the network about the contents which is not available in adjacent BN.

Because of node mobility, nodes status is flapping from MPR to non-MPR and vice versa. This causes frequent and unnecessary inheritance and check-out processes that should be avoided in networks with random mobility. Coping with this problem, after the BN dismissal, the node waits for the Dismissal Transient Time (DTT) before it drops the table. The DTT can be addressed also as the table expiration date. If the dismissed BN is reassigned to BN during DTT, it will not drop the tables and will not ask for new tables through inheritance process. The DTT should be determined according to the node mobility rate, subscription/memorized events generation rate, and the level of consistency expected from event system [20].

Instant Notification (Pull-based Notification): When a node subscribes to an event, the subscription filter is matched against all the events in MET, and in case it matches any of them, the node will be notified at once. Note that the notification is a neighbor-to-neighbor communication. Thus, the event notification in this model does not cause event duplication.

5 Chapar Evaluation and Performance Analysis

The main objective of the work presented in this paper is to create a reliable self-configuring event system resilient enough to network partitioning and node mobility. The resource awareness of the protocol is also important, since it is implemented in mobile nodes (PDAs) with limited power and bandwidth resources. Chapar is programmed and implemented as the event module of the Transhumance middleware on Nokia 770 [28]. The middleware including the event system is available on SourceForge [3] under LGPL licence and it was tested successfully [6,20]. An experiment with a game involving 8 users has clearly shown the efficiency of the persistent event system [21].

In this section, we discuss the system functionality and present preliminary performance evaluation. As the simulation tools, we used NS2 [27] with UM-OLSR [26] package for supporting the underlaying OLSR routing protocol. The detail of the simulation specifications are shown in Figure 4.

5.1 Functionality Analysis

Since Chapar uses the OLSR routing tables to build a real-time multicast tree in order to deliver the published event to subscribers, the notification is done regardless of node location and movement. Network partitioning/merging and node frequent disconnection are also properly handled using Table Consistency Check, Inheritance, and Check-out Process (They are explicitly investigated in [20]).

[3] http://sourceforge.net/projects/transhumance

Simulation Parameters	
Simulation Specifications	Tool: NS-2.28 + UM-OLSR [27] Duration: 3600s Simulation time slot (T): 30s
Network Specifications	Size: variable Number of nodes: 40 Network routing protocol: OLSR (RFC3626) TC_interval=5s, HELLO_interval=2s
Node Specifications	Radio-propagation model: Two Ray Ground Antenna: OmniDirectional MAC type: 802.11 Queue type and length: Drop-tail/Priority queue 50 Effective coverage range (R): 250 m
Mobility Specifications	Mobility Model: Random Speed: 1 m/s
Traffic Specifications	Subscription: Almost constant ($65B \leq size \leq 70B$) PDF: Uniform Subscription rate: 6 subscription/minute (3 subs/T) Memorized Messages: minimum: 94B, average: 362B, PDF: log-normal Memorized Events Rate: 2 event/minute (1 events/T)

Fig. 4. The simulation specifications

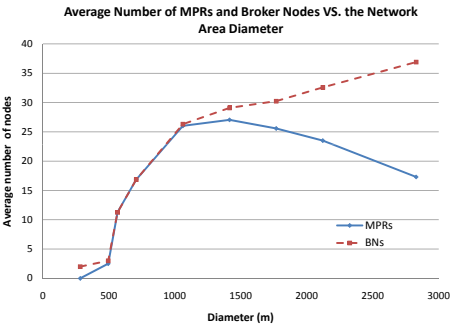


Fig. 5. Average number of MPRs and BNs

Table 1. Desirable properties check list for different event system functionalities in Chapar

	Orderedness	Consistency (avoiding event duplication)	Completeness
Subscription	√	suffers from duplicated copies in broker network	√
Real-time Event Publishing	√	√	√ with some exceptions
Memorized Event Publishing	Event reordering (partially)	suffers from duplicated copies in broker network and notification (partially)	√

The FMT and MET are replicated in each BN, and this redundancy prevents their content from being lost. There are three desirable properties for redundant systems [10] which are investigated in Chapar and shown in Table 1. *Orderedness* concerns the message reordering in event systems. For some applications, it is important that the events be notified in the same order they are published. *Consistency* addresses the message duplication in the distributed system. And in *completeness*, the system is expected to be as complete as the system with no replication.

5.2 Resource Awareness

The resource awareness is studied with respect to: memory usage, processes analysis, and signaling and bandwidth allocation [20].

Memory usage concerns the size of the FMT and the MET tables on broker nodes. First, using a subset of the nodes (the MPRs) as broker nodes reduces the number of nodes whose memory is occupied by event system tables. Second, using the NV bitmap and FID (a fixed bitmap) minimizes the size of the FMT. For instance, if the number of nodes is 40 and there are 100 available filters, the FMT table size is $\approx$ 1.5 KB. The MET size mainly depends on the size of the events. Therefore, it is recommended that the applications generate memorized messages with efficient sizes and specify the event lifetime value cautiously. To avoid overflows, tables sizes are bounded. Also, the

memorized events are generated with some upper-bounds to minimize the impact of the memorized messages on the event system. For instance, in Transhumance project, the memorized event sizes could not exceeds 2KB and the MET table size is bounded to 50 entries [20]. The process analysis addresses to the CPU usage and power-awareness of the system. All computations in Chapar are accomplished by simple logical operations thanks to NVs and simple hash functions. In [20], we have shown that even the lookup process which could be assumed as the heaviest procedure running on the system involves a limited number of operations and are adapted to tiny mobile devices (for 40 nodes and 5 filter ID fields is less than 100 AND operations for each FMT entry).

Signaling and bandwidth allocation refer to network traffic. Having minimum number of BNs is the key contribution in Chapar. It is important, since the number of BNs identifies not only the number the FMT and MET copies, but also the volume of the subscription and memorized messages generated and forwarded in the network. Figure 5 shows that the number of MPRs and BNs increase when the network area diameter grows and the network density declines. In dense networks ($D < 450$), there is no MPR in the network (special case), thus two BNs will be elected. In the semi-dense network ($450 < D < 1000$) the number of MPRs and BNs is the same and rises up to 45% of the total nodes. These two areas are recommended working area sizes. In the third area, the network experiences many network partitions and node seclusion. Thus, the number of MPRs declines whereas the number of BN rises because of the several occurrences of the special cases.

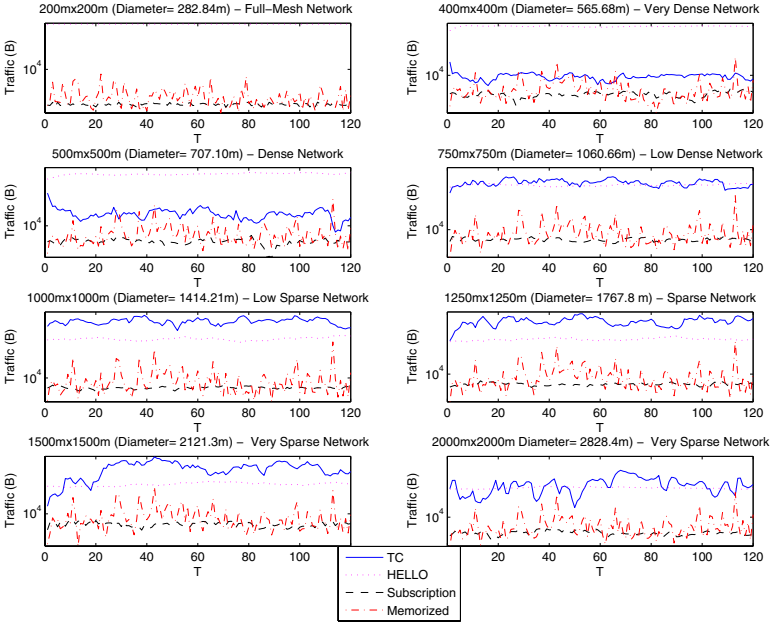


Fig. 6. The OLSR routing and event system total traffic in logarithmic scale in 120 time slots (3600s) with different densities

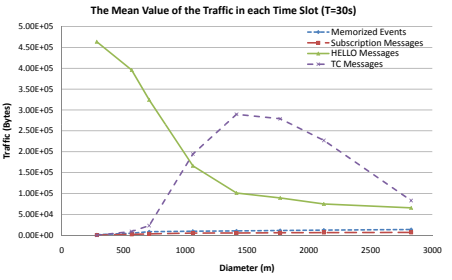


Fig. 7. The comparison between types of network traffic in different network densities

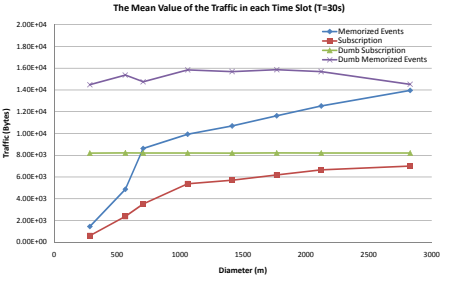


Fig. 8. Using MPRs as BNs VS using all nodes as BN

Figure 6 is the result of one hour ($120 \times time\ slot(T)$) simulation, of 40 nodes with random mobility. As described in Figure 4, the subscription traffic is generated at the fixed rate of 3 sub/T with uniformly random size ($65B < sub\_size < 70B$), whereas the memorized event is created and dispatched in the fixed rate of 1 (event/T) and random size (with lognormal distribution) between minimum of 64B and average of 362B. The lognormal traffic distribution enable to show the seldom large memorized events which is sent by some modules and applications. In fact, figure 6 demonstrates how the number of MPRs and BNs may effect the amount of generated traffic. Figure 7 (and also 8) shows this difference more clearly. The average number of memorized messages when the network is sparse is almost twice more than the number of memorized messages when the network is dense. Therefore, working on recommended area size could be crucial in network performance. Indeed, Chapar satisfies the Transhumance's requirements quite well, since the goal in Transhumance is to support the middle-scales ubiquitous ad hoc networks working in rural environment where users are not apart from each other considerably.

Figure 7 also compares the subscription messages traffic and memorized events traffic to OLSR messages. This figure clearly illustrates that the Chapar messages do not add significant traffic to the network in comparison to the OLSR routing messages. We have not shown the rest of event system traffic measurements since they are really negligible in compare to the subscription and memorized events traffic which should be received by all nodes in the network. Nonetheless, more details can be found in [20].

Finally, we show the effectiveness of Chapar event system in reducing the number of messages in the network. Figure 8 shows that using MPRs as BNs instead of using all nodes as BNs can decrease the traffic from 90% to 40% in the working area depending on the network density and the number of MPRs.

Unfortunately, we are not able to compare Chapar with the mentioned related works, because event systems are designed as middleware through which higher level applications may communicate. Thus, based on the system requirements and specifications, event systems are different in most of the cases. Therefore, a comprehensive comparison may not be feasible. Also, to the best of our knowledge there is no event system working on the mobile ad hoc networks that fully supports all publish/subscribe system decouplings. For instance, for time decoupling we use memorized messages with life

time and replicated event container for Chapar, which is not supported by any of current publish/subscribe systems. Moreover, the cross layer protocols are tightly bounded to the underlying network layers, so the feasibility of proper comparison between the cross-layer approaches and the independent ones is questionable.

6 Conclusion

In this paper, we have introduced Chapar, a novel event system designed for MANETs. Chapar supports the publish-subscribe model as well as point-to-point and point-to-multipoint message sending. Chapar uses a distributed reliable approach to implement the event brokers. It provides consistent and reliable event dissemination in harsh mobile environments. However, since Chapar is not dependent on any specific node and is a ubiquitous event system, it could be implemented on stable homogeneous networks. Chapar uses OLSR as an underlay network to propagate the events efficiently through the network and support mobility and self-management. It is also robust to the network partitioning and frequent topology changes thanks to Table Consistency Check, Inheritance, and Check-out Processes. We have shown that this protocol is light and well-designed for mobile devices thanks to NVs and simple logical operations. The network overhead of this protocol is also negligible compared to the OLSR routing messages.

Acknowledgment

The work presented in this paper was supported by the French National Research Agency (ANR) funded Transhumance Project. We are thankful to Javier Hernando for his support in Chapar implementation.

References

1. Chlamtac, I., Conti, M., Liu, J.: Mobile Ad hoc Networking: Imperatives and Challenges. Ad Hoc Networks 1(1), 13–64 (2003)
2. Clausen, T., Jacquet, P.: Optimized Link State Routing Protocol. RFC 3626, Internet Engineering Task Force (October 2003)
3. Hapner, M., Burridge, R., Sharma, R., Fiall, J., Stout, K.: Java Message Service. Sun Microsystems Inc., Santa Clara (2002)
4. OMG. CORBA Notification Service Specification. Object Management Group, Needham, MA (August 2002)
5. Paroux, G., Martin, L., Nowalczyk, J., Demeure, I.: Transhumance: A power sensitive middleware for data sharing on mobile ad hoc networks. In: ASWN 2007, Santander, Spain (May 2007)
6. Transhumance Project web page, `http://www.infres.enst.fr/~demeure/TRANSHUMANCE/\/index.html`
7. Cugola, G., Di Nitto, E., Fuggetta, A.: The JEDI Event-Based Infrastructure and Its Application to the Development of the OPSS WFMS. IEEE Tran. on Software Engineering 27(9), 827–850 (2001)
8. Eugster, P., Felber, P., Guerraoui, R., Kermarrec, A.: The many faces of publish/subscribe. ACM Computing Surveys (CSUR) 35(2), 114–131 (2003)

9. Cugola, G., Murphy, A., Picco, G.: Content-based Publish-Subscribe in a Mobile Environment. In: Bellavista, P., Corradi, A. (eds.) Mobile Middleware, pp. 257–285. Auerbach Publications (2006)
10. Huang, Y., Garcia-Molina, H.: Publish/Subscribe in a mobile enviroment. In: Proc. of the MobiDE 2001, Santa Barbara, CA, pp. 27–34 (May 2001)
11. Bacon, J., Moody, K., Bates, J., Hayton, R., Ma, C., McNeil, A., Seidel, O., Spiteri, M.: Generic support for distributed applications. Computer 33(3), 68–76 (2000)
12. Fiege, L., Gartner, F., Kastenm, O., Zeidler, A.: Supporting Mobility in Content-Based Publish/Subscribe Middleware. In: Proc. of ACM/IFIP/USENIX Int. Middleware Conference (Middleware 2003), Rio de Janeiro, Brazil, pp. 103–134 (June 2003)
13. Carzaniga, A., Rosenblum, D., Wolf, A.L.: Design and Evaluation of a Wide-Area Event Notification Service. ACM Tran. on Computer Systems (TOCS) 19(3), 332–383 (2001)
14. Cao, F., Singh, J.P.: Efficient event routing in content-based publish-subscribe service networks. In: Proc. of IEEE INFOCOM 2004, Hong Kong, China (2004)
15. Banavar, G., Chandra, T., Mukherjee, B., Nagarajarao, J., Strom, R., Sturman, D.: An efficient multicast protocol for content-based publish-subscribe systems. In: Proc. ICDCS 1999 (1999)
16. Chand, R., Felber, P.: A Scalable Protocol for Content-Based Routing in Overlay Networks. In: IEEE Int. Symposium on Network Computing and Applications (NCA 2003), Cambridge, MA (April 2003)
17. Hauswirth, M., Jazayeri, M.: A component and communication model for push systems. In: Proc. of the 7th European Software Engineering Conference, Toulouse, France, pp. 20–38 (September 1999)
18. Delmastro, F., Conti, M., Gregori, E.: P2P Common API for Structured Overlay Networks: A Cross-Layer Extension. In: Proc. of MDC 2006, Niagara Falls, NY (June 2006)
19. Conti, M., Crowcroft, J., Delmastro, F., Passarella, A.: P2P Support for Group-Communication Applications: a Cross-Layer Approach for MANET Environments. In: Demo Session of INFOCOM 2006, Barcelona, Spain (April 2006)
20. Khakpour, A., Demeure, I.: Designing and Prototyping an Event-based Communication System on Mobile Ad Hoc Networks, Technical Report 2008D009, Ecole Nationale Supéure des Télécommunication (July 2008)
21. Demeure, I., Gentès, A., Stuyck, J., Guyot-Mbodji, A., Martin, L.: Transhumance: a Platform on a Mobile Ad hoc NETwork Challenging Collaborative Gaming. In: 1st International Workshop on Collaborative Games (CoGames 2008), Irvine, CA, USA, May 19-23 (2008)
22. Bloom, B.H.: Space/time trade-offs in hash coding with allowable errors. Communications of the ACM 13(7), 422–426 (1970)
23. Tran, P., Greenfield, P., Gorton, I.: Behavior and Performance of Message-Oriented Middleware Systems. In: Proc. of the ICDCS 2002, pp. 645–654 (2002)
24. Jung, D.: Design of MOBILE MOM: Message Oriented Middleware Service for Mobile Computing. In: Proc. of the ICPP 1999, pp. 434–439 (1999)
25. Souto, E., Guimaraes, G., Vasconcelos, G.: A message-oriented middleware for sensor networks. In: Proc. of the MPAC 2004, vol. 77, pp. 127–134 (2004)
26. Ros, F.J.: Universidad de Murcia OLSR impelmentation for NS2, `http://masimum.dif.um.es/um-olsr/html/`
27. NS2, The Network Simulator, `http://www.isi.edu/nsnam/ns/`
28. Nokia 770 Internet Tablet, Technical Specifications, `http://europe.nokia.com/A4145105`