EURASIP Journal on Wireless Communications and Networking
a SpringerOpen Journal

RESEARCH Open Access

A cross layer interception and redirection cooperative caching scheme for MANETs

Francisco J González-Cañete*, Eduardo Casilari and Alicia Triviño-Cabrera

Abstract

The caching paradigm has widely been used in computers, data bases and the Internet to reduce the response time of the applications and to reduce the traffic load in the computer buses or in the networks. These advantages can also be applied in a Mobile Ad Hoc Network (MANET) as the caching paradigm could reduce the interferences due to the traffic overload and it could also increase the availability of the documents due to server disconnections. In this article, we propose a cooperative caching scheme for MANETs. The proposal is supported by the local cache that all the nodes in the MANET possess. In a collaborative way, a node could respond to the demand of other nodes if it is keeping a valid copy of the required document. As a novelty, the demands of the documents are not restricted to specific application messages but they are codified into the messages generated when the path to the server is being searched. In this way, the documents can be retrieved even when there is no route to the server. Furthermore, these expanded messages provide useful information about potential localizations of the documents. The proposed technique will use those additional location data to redirect the requests. By means of simulations, we have evaluated and compared our proposal to five other caching schemes as well as with the option of no caching. The obtained results indicate that our proposal outperforms the others in terms of document accessibility, delay and generated traffic load.

Keywords: ad hoc networks, MANETs, cooperative caching, distributed cache, local cache, redirection

1. Introduction

Wireless technologies are present in our daily lives. It is common to come across home appliances, mobile phones, personal digital assistants, electronic pads and tablets or laptops provided with one or more built-in radio interfaces. In some cases, these devices work autonomously without the support of any telecommunication operator. By establishing their own network, the devices are able to communicate among themselves. In the particular case that the direct connection between all the devices is not guaranteed, the mobile devices may form a Mobile Ad hoc Network (MANET). A key characteristic of MANETs relies on the fact that devices are expected to operate autonomously in multiple aspects. However, this autonomy often derives from the cooperation of them, such as it happens in the routing procedures. In a MANET, packets must be retransmitted by intermediate devices to ensure that they are received by the intended destination. This is why packets are said to hop from one node to another leading to multihop communications supported by the ad hoc routing protocols. Although routing protocols play an important role in MANETs, their performance can greatly be improved when some other technologies are adapted to multihop wireless communications. This article addresses the problem of adapting web caching techniques to these communication conditions.

Web caching is considered a helpful technique to share web documents in a network. By allowing the temporary storage of web documents in the clients or in the proxies, a document request can be satisfied even when the server storing the document is not reachable. In this way, the access to the documents is enhanced. In a MANET context, configuring the devices as potential proxies of the received or transmitted documents offers potential advantages owing to the fact that wireless retransmissions could be minimized. In this way, the mobile devices could share the documents in an efficient way. For instance, systems deployed for museums or for

* Correspondence: fgc@uma.es
Department Tecnología Electrónica, University of Málaga, Campus de Teatinos, ETSI Telecomunicación, Málaga, Spain

visiting theme parks usually offer a friendly graphical interface through which preloaded formatted content is shown. The content is generated by the service provider but often needs to be updated. In order to avoid the tedious task of continuously configuring the devices, web caching techniques could be applied to efficiently distribute the new content through the entire network.

An important aspect to consider is that web caching is supported by a memory structure where the documents are stored. Due to the finite storage space, replacement policies become necessary to decide which documents should be evicted. In this sense, replacement policies could also benefit from the knowledge about the topology of the communication network. On the other hand, the information disseminated along the network can have an expiration time, that is, the information is considered obsolete after a Time-to-Live (TTL) period. Thus, the expired documents have to be deleted from the mobile devices and requested again, if they are required, in order to maintain the information up to date. Replacement policies impact on the network performance.

Towards the goal of improving the MANET performance, we propose a novel caching scheme for multihop ad hoc networks. More precisely, our algorithm takes advantage of the messages generated by the routing protocols to become aware of more efficient approaches to distribute the caching messages. In addition, an efficient method to manage the information that the mobile devices keep about the location of the documents is used to distribute the requests along the network. Although some other caching mechanisms have already been customized for MANETs, from the evaluation results we state that our proposal yields significant improvements when compared to the previous solutions.

The rest of the article is structured as follows: Section 2 reviews the related study in the literature. The section also proposes a taxonomy to classify the existing caching mechanisms. In Section 3, the proposed caching scheme is thoroughly described. Section 4 details the simulation model used in this article. The performance of the proposal is evaluated in Section 5 with an extensive simulation study. Finally, Section 6 outlines the main conclusions and suggests possible future work.

2. Related study

The main objectives of a cooperative caching scheme in a wireless network are threefold. First, a cooperative caching scheme is expected to increase the accessibility to the documents in the MANET. Second, it aims at moderating the energy consumption in the mobile nodes (MN) by reducing the traffic across the network. Finally, it is desirable that the latency perceived by the users when retrieving the documents to be reduced. In order to achieve these goals, some cooperative caching schemes have been proposed. The approaches differ in how the document requests are propagated thorough the network. In this way, they can be divided into four non-exclusive categories: *broadcast-based*, *information-based*, *role-based* and *direct-request* mechanisms. In the *broadcast-based* approach, the MNs broadcast the requests in order to find a MN which can reply with the requested document. Since the Data Server (DS), which is the server storing the documents, is also a node in the network, it can also reply to these broadcast requests. As *broadcast-based* mechanism, we can mention MobEye [1], SimpleSearch [2] and Hamlet [3]. On the other hand, in the *information-based* methods, the MNs interchange or store information about where the documents are located in the network. This information is utilized to find distributed copies of the documents along the network when needed. Distributed Greedy Algorithm [4] and Wang [5] outstand as information-based caching schemes. Under the *role-based* approach, each MN has a specific functionality in the network as they are assumed to be organized in clusters. Depending on the architecture, some MNs are selected as information coordinators or clients. Using this methodology, the information coordinators deal with the task of storing distributed information about the location of the documents, while working as servers for the cluster to which they belong. Thus, the information coordinators mainly manage the document requests. Cluster Cooperative [6] and Denko [7] are illustrative examples in this category. Finally, in the *direct-request* method, the requests are directly sent to the server through the path discovered by the routing protocol. When traversing this path, the request can be replied by any of the relay nodes in the route as implemented in Gianuzzi [8].

The previous groups of caching mechanism are not mutually exclusive. We can find caching mechanism that fits both *information* and *broadcast* characteristics (COOP [9], IXP/DPIP [10]), *information* and *role*-based (COACS [11]), *information* and *direct-request* (CachePath/CacheData/HybridCache [12] and GroupCaching [13]), *broadcast* and *direct-request* (Zone Cooperative Caching [14]).

Next, we describe with more details the caching schemes that are going to be used to compare our proposal, that is, *MobEye*, *SimpleSearch*, *COOP*, *DPIP* and *CacheData*/*CacheData*/*HybridCache*. These caching schemes have been selected as they are the most representatives of each type.

In the *MobEye* caching scheme, when a MN needs a document not stored in its local cache, a request is broadcast to the entire network. When a MN receives the request, it checks in its local cache if there is a valid

copy of the document. If so, it replies with an acknowledgement (*ack*) message to the requester. The first *ack* message that reaches the requester will prompt the source to generate a specific request, named *confirm*, to the node that is known to maintain the document. The *MobEye* messages do not include any routing information so a routing protocol needs to be used in order to transmit the *confirm* message. Concerning the replacement policies, *MobEye* proposes the implementation of a Least Recently Used (LRU) within each MN.

SimpleSearch also proposes to broadcast the document requests. However, the broadcast requests are exclusively performed when the node sending the request holds two conditions. The first condition refers to the fact that the node does not possess a valid copy of the document in its local cache, so that the request needs to be propagated. On the other hand, the second condition relies on being a direct-request-based caching scheme. More particularly, broadcasting is avoided when the node sending the message maintains a route to the DS. Under these circumstances, the request is transmitted to the DS along the known route. In contrast, when using broadcast requests, the messages store the route (i.e., the addresses of the intermediate nodes) through which they are passing by. Thus, as soon as a MN that has a valid copy in its local cache receives the request, it replies with an *ack* message using the reverse route. As in the *MobEye* scheme, the requester sends a *confirm* message to the node that sent the *ack* message, which finally replies with the document requested. The *confirm* and *reply* messages use the same route as the request and *ack* messages, respectively. *SimpleSearch* also sets a maximum number of hops in the propagation of the broadcast requests in order to reduce the traffic in the network. *SimpleSearch* proposes three options for the local replacement policy based on the following two parameters: δ, defined as the number of hops to the node from which the document was served, and τ defined as shown in (1):

$$\tau = \frac{1}{t_{\text{cur}} - t_{\text{update}}} \tag{1}$$

where t_{cur} is the current time and t_{update} is the last access time of the document. The three proposed replacement policies for *SimpleSearch* are

1. TDS_D (Time and Distance Sensitive - Distance) which considers the distance in hops to the source node as the criterion to evict the documents from the local cache. Thus, the documents which were served from closer MNs are firstly evicted. In the case of a tie the deleted document will be the one with the lowest value of τ.
2. TDS_T (Time and Distance Sensitive - Time) evicts the documents with the lowest value of τ, that is, with the longest associated time since they were last updated.
3. TDS_N combines both previous methods, by discarding the documents with the lower value of the product ($\delta \times \tau$).

In addition, *SimpleSearch* also implements an admission control policy that only allows to store the documents in the local cache if they were served by a node that is located from a minimum number of hops away.

Alternatively, *COOPerative* (*COOP*) performs an adaptive flooding broadcast to search for the documents in the network. According to this scheme, any MN has to maintain a Recent Request Table (RRT) where they store information about the source (and its corresponding distance in hops) of the requests that the node forwards. Each entry in the RRT table has an associated expiration time based on the mean TTL of the forwarded documents. If the document to be requested is not stored in the local cache, the MN checks in its RRT table if there is an entry informing about the location of the document. If so, the document is directly requested to the stored source instead of broadcasting the request to the whole network. If there is not such information in the RRT table and the broadcast fails, the request is sent to the DS. As the replacement policy, *COOP* proposes to divide the documents into primary and secondary copies. A copy of a document in a MN is considered as primary if it is not stored in another node in the neighbourhood. Consequently, a copy is defined as secondary if there is at least one replica in the neighbourhood. Thus, *COOP* maintains two LRU lists with the primary and the secondary copies, respectively, considering that the documents stored in the secondary copy list must be deleted first. *Hello* messages have to be periodically broadcast one hop away in order to keep the list of neighbour nodes updated.

DPIP (Data Pull/Index Push), as well as *COOP*, also uses an additional data structure. In particular, each node maintains a table with an entry for each document in the system in the form of (*x*, *cached*, *cachednode*, *count*) where *x* is the document identification, *cached* is a Boolean variable indicating if this document is stored in the local cache, *cachednode* stores the address of the last node that previously requested the document and it is expected to have a copy of the document, while *count* is the number of copies of the document stored in the neighbourhood. When a node needs a document that is not stored in its local cache, the MN looks up the table to find a neighbour that has the document. If this information is not available, the MN sends a *data_pull* message to its neighbour nodes informing about the needed

document and the documents that will be evicted from its local cache upon receiving the requested document. When a node receives a *data_pull* message it updates its table and replies if one of the next conditions are true:

(a) The node has a copy of the requested document (which is sent to the requester).
(b) The node has an entry in its table informing about a node that previously requested the document. In this case the node replies with a *location_reply* message to the requester, notifying the node to which the request must be forwarded.

If there is no reply to the *data_pull* message before a certain time (*DPIP_timer*), the document is requested to the DS. On the other hand, the local replacement policy firstly evicts the documents with the greatest value of the variable *count*, aiming at reducing the number of redundant copies in the neighbourhood. As in COOP, the MNs periodically send *Hello* messages to update the list of neighbour nodes.

In the *CacheData*, *CachePath* and *HybridCache* strategies, the MNs directly request the documents to the DS. However, the intermediate nodes in the path followed by the requests can intercept or redirect the message basing on their local caches or on the information they manage. *HybridCache* is also supported by additional data. It stores the path for the document if its TTL is greater than a predefined threshold and the distance (in hops) to the DS minus the distance to the node having the document is greater than a predefined parameter. On the other hand, a MN in a route to the request originator stores the document if there is an entry in the cache of paths for the document. The *Size*Order (SXO)* policy is proposed as replacement strategy. This policy evicts from the cache the documents with the greatest value of the following metric (2):

$$\text{value}(d_i) = s_i \times \text{Order}(d_i) \tag{2}$$

where s_i is the size of the document d_i and k = Order (d_i) is the kth most frequently accessed data. As Order (d_i) is not available, it is derived from the MN access rate to d_i, denoted as a_i (3):

$$a_i = \frac{K}{T - T_{ai}(K)} \tag{3}$$

where T is the current time and $T(K)$ is the time when the Kth previous request to the document was performed being K the size of the request window. With the aim of reducing the processing time at each node, an optimization for *CacheData* was proposed in [15]. This enhancement proposes that the server decides which nodes in the response path have to store the document instead of leaving this decision to each node.

3. Proposed caching scheme: CLIR

In this section, the Cross Layer Interception and Redirection (*CLIR*) caching scheme is presented. This scheme proposes to benefit from the advantages of the previously proposed caching schemes while avoiding their flaws. The caching scheme is supported by five important features, which will be discussed next.

3.1. Local caching

For a MN in a MANET that requests information to DS or to the Internet through a gateway, the first strategy to reduce the traffic in the network is to implement a local cache for the requested documents. This cache can serve the requests of the node itself to the same document even if the DS is not available. In that case the requests will not introduce any traffic in the MANET.

As it refers to the replacement policy, many strategies have been proposed for the proxy caches in the Internet. Each policy is intended to process a specific type of traffic [16,17]. Therefore, if this type of traffic in the network is known, the most adequate replacement policy for it can be selected. Unfortunately, this is not always possible. Hence, we utilize for our caching scheme the LRU replacement policy because it is simple to implement, it is generally quite effective [18] and it does not have any configuration parameter to be 'heuristically' tuned.

3.2. Intercepcion caching

The next step in our proposal requires the MNs to cooperate in order to reply the requests of the other nodes using their local caches. In particular, we employ the Interception Caching scheme. To illustrate how this technique works, we use the example in Figure 1a. This figure shows a snapshot of a MANET where DS is the node that physically stores all the documents or provides access to external networks. Nodes 1, 2, 3, 4, 5 and 6 are user nodes that request documents to DS. In

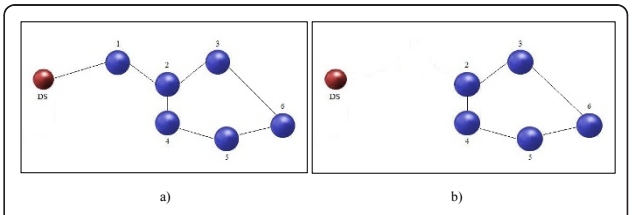

Figure 1 Example of MANET. **(a)** With connection to DS. **(b)** Without connection to the DS.

the figure, the lines connecting the nodes indicate the existing wireless links.

Let us suppose that node 2 requests the document *A*. The request will be forwarded using the routing protocol to DS, which will respond with the document *A* using the reverse route. Then, the node 2 will store *A* in its local cache. In the case that node 3 requests the same document *A* to DS, the request will reach node 2, which can reply to node 3 with the copy of the document *A* existing in its local cache by means of the interception caching technique. The Interception Caching technique reduces the number of hops necessary to obtain the document and, consequently, the number of protocol messages along the network.

3.3. Cross-layer intercepcion caching

The Interception Caching procedure exclusively works when the routing paths from the source of the requests to DS are known. However, this requirement is not always satisfied in a MANET. Let us suppose the situation where node 1 in Figure 1a has moved outside the coverage area of DS and hence node 2 cannot access DS (see Figure 1b). On the other hand, nodes 3 and 4 are located in the coverage area of node 2. When node 4 proceeds to request the document *A* to DS, it detects that DS is unreachable, i.e., there is no route to DS in its routing table. Under these circumstances, node 4 will trigger the procedures to discover a new route to DS. However, there is no path between node 4 and DS and, consequently, the route cannot be discovered. Thus, the request for the document cannot be emitted and, in turn, will not be replied even when node 2 has a valid copy of the document *A* in its local cache.

In order to avoid this problem, we propose to involucrate the routing algorithm in the process of looking for the documents in the MANET in the so called Cross Layer Interception procedure. Let us suppose that the Ad hoc On-Demand Distance Vector (AODV) MANET routing protocol [19] is utilized. The broadcast Route Request (*RREQ*) message is sent in order to create the route to DS. We propose that this message includes information about the demanded document. In this sense, the document identification is transported or 'piggybacked' into the AODV *RREQ* message. In our example, node 4 will broadcast a *RREQ* message in order to create the route to DS with the solicitation of the document *A*. When node 2 receives the *RREQ*, it checks if a request is piggybacked in the message. If so, it extracts the information and verifies if there is a copy of the document in its local cache. Then, node 2 responds to node 4 using a *RREP* (Route Reply) message including the piggybacked identification of the document. As the node 4 receives the *RREP* and hence the route between nodes 4 and 2 is created, node 4 directly sends the document request to node 2. By using this procedure, the nodes in the MANET can access to the disseminated documents even if DS is temporarily inaccessible.

Although *MobEye* and SimpleSearch employ a similar broadcast-based request method, when a node receives a broadcast request, it has to create a reverse route to the requester in order to send a unicast advertisement informing about the document location. If many mobiles nodes have a copy of the requested document, many routes have to be created by the routing protocol while just only one of them is actually needed. This operation unnecessarily increases the protocol traffic load in the network. This flaw is avoided in the proposed Cross-Layer Interception caching.

3.4. Redirection caching

Caching schemes such as *HybridCache*, *COOP*, or *DPIP* maintain tables of information about the nodes where the documents are located in the network. To keep these tables correctly updated, nodes are obliged to send periodic information. This may imply a heavy traffic overload, especially if there are many documents and requests are frequent. Moreover, these tables maintain information based on the requests of documents but they do not take into consideration if the document is not received by the requester and, consequently, the request originator does not finally store the document. The MNs could redirect a request using these tables to a node that has not actually received the document causing an error in the redirection process. In addition, the validity of this redirection information is only based on the TTL of the documents and it does not consider the possible eviction of documents in the remote caches. Thus, a document request could be redirected to a node even in the case when this node has deleted the document from its local cache. Finally, if a redirection error is detected because of incorrect or obsolete information in the tables, *HybridCache*, *COOP* and *DPIP* do not implement any mechanism to update the information they manage. Thus, they maintain the tables' entries even when a problem has been detected.

Aiming at avoiding these drawbacks, we propose the Redirection Caching procedure. This method also proposes to implement in all the nodes in the MANET a Redirection Cache structure with information about the location of the documents. However, it differs from the previous policies in the literature in the way that this Redirection Cache is managed. The idea is that each entry in the Redirection Cache must contain the tuple (*id*, *IP_GET*, *hops_GET*, *TTL_GET*, *IP_RESP*, *hops_RESP*, *TTL_RESP*) where *id* is the identification of a document, *IP_GET* is the IP address of the MN that performed the request, *hops_GET* stores the distance in hops to the requester node, *TTL_GET* is the time when

the document expires, while the rest of the information refers to the node that replied with the document (if it is not a server).

The TTL of the documents is learnt from the response messages containing the documents, in the way back to the requester. Accordingly, the Redirection Caching structure is managed using two LRU lists: the *KNOWN_TTL* list contains the entries related to documents whose TTL values are known while the *UNKNOWN_TTL* list contains the entries to documents with unknown TTL values. The space for each structure is dynamically assigned but the memory space reserved for both structures is set to a constant. If an entry of the *UNKNOWN_TTL* list is updated with the TTL of a document, it will be transferred to the head of the *KNOWN_TTL* list. When a new entry must be stored and there is not enough room because the reserved storage space is full, the oldest entry in the *UNKNOWN_TTL* list is evicted. If this list is empty, the oldest entry in the *KNOWN_LIST* will be deleted. An entry is also deleted if the information is obsolete because all the associated TTLs have expired.

Let us suppose in Figure 1a that node 3 requests the document *B* to DS. The request will pass through nodes 2 and 1. These nodes will create and entry for a document with identification (*id*) *B* and they will annotate in the *IP_GET* field the node 3 address. Similarly, the nodes will indicate in the *hops_GET* fields that the source node is at one and two hops away, respectively. The *TTL_GET* field will be undefined as it is still unknown. When DS replies with the document using the reverse route DS-1-2-3, the *TTL_GET* value will be updated with the expiration time of the document. The *IP_RESP*, *hops_RESP* and *TTL_RESP* values are left blank because the reply was delivered by DS. After this operation, if node 4 requests the same document *B* to DS, the request will reach node 2, which will check its Redirection Cache. Now, this Cache states that node 3 is located one hop away, so it is closer than DS, which is two hops away. Thus, node 2 will redirect the request to node 3. The *IP_RESP*, *hops_RESP* and *TTL_RESP* fields are filled when the responding node is not a server, that is, an interception or redirection operation is executed.

Every time a MN receives a request, it checks in its Redirection Caching structure if there is information to redirect the request to other node. In order to redirect a request, some conditions must be considered in the Redirection Caching procedure. These conditions are

1. A request only can be redirected once.
2. A request is redirected only if the associated expiration time (*TTL_GET* or *TTL_RESP*) is known.
3. The request is redirected if any of the next conditions are true:
   (a) The number of hops to the redirection node is lower than the number of hops to the destination of the request (e.g., DS).
   (b) The number of hops to the destination of the request is unknown.
   (c) The number of hops to the redirection node is unknown (because no route exists to the node) but the number of hops stored in the Redirection Cache for the redirection node is lower than the distance to the destination node and the destination node is located more than two hops away.

Condition 1 prevents redirection loops caused by multiple redirections. Condition 2 avoids the redirection of requests to nodes that are not guaranteed to have received the requested document. Let us suppose that node 6 requests the document *C* to DS using the route 6-5-4-2-1-DS and DS replies with the document but using the route DS-1-2-3-6. In this state, nodes 4 and 5 may know that node 6 requested the document *C* but they do not know if the document was received as the reply did not pass through them. By means of condition 2, these particular nodes will not be able to perform the redirection procedure for the document requested in these operations. Condition 3 declares the requirements that must be satisfied to redirect a request: condition (a) will assure that if the distance in hops to the redirection and original destination nodes are known (because there are valid routes created by the routing protocol), the request will be redirected to the closer one; if condition (b) is true, the request is redirected because the distance to the destination node (e.g., DS) is unknown as there is not a route created and it could even be inaccessible, so the redirection is performed; finally, condition (c) redirects a request if the distance to the redirection node is unknown (because a route to it does not exist) but, according to the Redirection Cache, it is closer than the destination node (which is known to be at least two hops away). Doing this, the request is redirected even if the redirection node is not in the neighbourhood with the hope of finding it in a closer position than the destination node. Condition 3 ensures that the wireless traffic induced by the redirection mechanism is lower than that required when it is not applied.

Previous studies only take into account the TTL associated to a document to set the expiration time of the information they manage. In order to consider the eviction of documents in the local caches, we propose to set as the validity time for the redirection information the minimum between the document TTL and the mean

time that the documents stay stored in the local cache. This value can be easily calculated by each node by observing its local cache. Thus, each node computes the time elapsed since each document has been stored and the instant in which it is evicted from the local cache. This time is employed to estimate the mean time that the documents are stored in the other nodes in the MANET (assuming that they exhibit similar caching dynamics). Using this mechanism, the redirection of requests to nodes that have already evicted the documents is diminished. As there are some cases when this method can also fail, we propose to send an error message when a node receives a redirected request and does not have the document in its local cache. In particular, the node will send a *REDIRECTION_ERROR* message to the node that redirected the request, which will permit to update the corresponding Redirection Caching structure. The message will also oblige the redirecting node to send the request to the original destination of the request. This mechanism avoids reiterative wrong redirections to a node that evicted a document before the expected time calculated by the Redirection Caching structure. The inclusion of the *REDIRECTION_ERROR* message is a novelty compared to the *COOP*, *Hybrid-Cache* and *DPIP* algorithms.

3.5. Middle reverse route caching

CLIR also implements the mechanism proposed in [20] that suggests that a node C in the middle of the route between the source node S of the reply of a document d and the destination node of the reply D should also store the document d in its local cache. This mechanism replicates the documents across the network in order to increase their availability. However, this mechanism is only performed if the distance from the requester to the replier is at least of four hops away in order to avoid an excessive replication of the documents in the neighbour nodes. Additionally, the algorithm divides the route between nodes S and D into three parts: nodes located in the first half of the route from S to C will register in their Redirection Cache that node S has a copy of d (if S is not a DS); nodes located in the second half of the route between S and C and in the first half of the route between C and D will store that node C has a copy of d; finally, nodes located in the final half of the route between C and D will record that node D has a copy of d. With this information, the next requests to document d can be directly sent to other nodes that are known to be closer than DS.

3.6. Algorithm and contributions

In order to clarify the functionality of *CLIR*, the pseudo code of the algorithm has been included as a resource. The algorithm has been divided into two sections: the algorithm when a node requests a document and when a node has to forward a request or a reply.

A previous initial version of *CLIR* was evaluated for static grid networks in [20,21]. In addition, that version was also evaluated in a MANET using the Random Waypoint and the Manhattan Grid mobility models [22]. The previous evaluated version included the Local caching and the Interception and Redirection caching. However, the Redirection Caching structure was infinite and it only redirected requests if a route to the redirection node was available. Consequently, the contributions of this paper to *CLIR* include:

1. The Routing Interception caching, which enhances the availability of the documents in the network even in the cases when the DS is unreachable, avoiding the creation of multiple unnecessary reverse routes to the requester to inform that the document is available.
2. The management of the Redirection Caching information. Since the available amount of storage space is limited, the Redirection Caching space is optimized in order to store as much useful information as possible.
3. The Redirection Caching procedure. This procedure is accomplished not only if the new node (to which the request is redirected) is closer than the original destination but also in other cases, which helps to improve the utility of the Redirection Caching.
4. The implementation of the document dissemination across the network and the integration into the Redirection Caching structure.
5. The performance of *CLIR* has been compared to other five caching schemes as well as the option of no caching. This is the first study that compares seven caching schemes at the same time. The related work used to compare the proposed caching scheme only with just another caching scheme.

4. Simulation testbed

By means of simulations we have evaluated the performance of the caching scheme proposed in this article. The simulations are based on the Network Simulator NS-2.33 which is a very popular simulation tool for the research on ad hoc networking [23].

Table 1 summarizes the main parameters of the simulated testbed. We suppose a default case with 50 MNs distributed in a 1000×1000 m^2 area. We also study the performance of a network with 25, 75 and 100 nodes in order to evaluate the influence of the density of nodes in the network. The MNs follow the modified Random Way Point (RWP) mobility pattern [24] moving at a

Table 1 Simulation parameters

Parameter	Default values	Other utilized values
Simulation area (m)	1000 × 1000	
Nodes	50	25-50-75-100
Servers	2	
Documents	1000	
Document size (bytes)	1000	
Timeout (s)	3	
TTL (s)	2000	250-500-1000-2000-∞
Mean time between requests (s)	25	5-10-25-50
Traffic pattern (Zipf slope)	0.8	0.4-0.6-0.8-1.0
Replacement policy	LRU	
Local cache size (number of documents)	35	5-10-35-50
Redirection Cache size (registers)	35	
Simulation time (s)	20000	
Warm-up time (s)	4000	
MAC protocol	802.11b	
Radio propagation model	Two Ray Ground	
Coverage radio (m)	250	
Ad hoc routing protocol	AODV	
Mobility pattern	TVCM	RWP
Speed (m/s)	Constant: 1	Constant: 1-3-5
Pause time (s)	0	

default constant speed of 1 m/s with no pause time after reaching each destination. This value corresponds to the mean walking person speed. We also test the scenario with constant speeds of 3 and 5 m/s to investigate the influence of the node mobility. Those speeds could correspond to a running person or a person riding a bicycle, respectively. In order to consider more realistic node mobility patterns, the Time-Variant Community Model (TVCM) [25] mobility model has also been used. TVCM is a realistic model obtained from actual traces of users' movements in wireless LAN in university campuses and corporate buildings. TVCM considers two parameters in the model. First, the model assumes that nodes have popular locations so they move to sites of preference (which are called communities). Additionally, TVCM considers periodical reappearances of the nodes. Taking into account these two features, TVCM proposes a mathematical model which incorporates a non-homogeneous behaviour in both space and time. The periodicity in time is controlled by a specific parameter (*period duration*). On the other hand, the *average length* parameter defines the mean value of an exponential distribution which characterizes the length that a node moves within a community. It is possible to generate the movement traces from the tool available at [26].

We consider 1,000 different documents (identified by a specific number) distributed among two fixed DSs with no mobility. The servers are located at positions (x, y) = (0,500) and (x,y) = (1000,500) in the simulation area respectively (with x and y coordinates expressed in meters). The DSs do not move as they are supposed to be access points that have direct access to the Internet using a wired medium. Each DS is directly connected to a Gateway or external network. To distribute the traffic between the DSs, documents with odd identification are placed in a server and even-numbered documents are located in the other one. In addition, each document has an associate TTL time that determines when the document expires, so that it is considered to be obsolete. The expired documents stored in the local caches are evicted in order to make room for fresh documents. We have considered an exponential distribution with a mean between 250 and 2000 s (depending on the experiment) for the TTL of the documents. In that way, we model both a high and low variability of the documents' lifetime. In this sense, we have also analysed the case of an infinite TTL for the documents, i.e. the documents never expire. The size of all the documents is constant and set to 1000 bytes.

Every node not being a server is programmed to generate requests to the servers along the simulation time. When a request is served, another request is generated by the same node after a certain time. The idle time between the reception of a response and the next request follows an exponential distribution with a mean time between 5 and 50 s (by default it is set to 25 s). By modifying all these parameters, we evaluate a wide range of patterns for the node activity (and consequently for the network load). A document is requested again if the response of the current request is not served before a defined timeout.

The pattern of requests of the documents follows a Zipf-like distribution which has been demonstrated to properly characterize the popularity of the Web documents in the Internet [27]. The Zipf law asserts that the probability $P(i)$ for the ith most popular document to be requested is inversely proportional to its popularity ranking (i) as shown in (4):

$$P(i) = \frac{\beta}{i^{\alpha}} \tag{4}$$

where parameter α is the slope of the log/log representation of the number of references to the documents as a function of its popularity rank (i) while β is the displacement of the function. Depending on the experiment, to select the document to be requested, we vary the selected slopes (α) from 0.4 to 1.0.

Finally, each node implements a local cache that employs the LRU replacement policy. This replacement policy discards the documents which have not been

requested for a longer period of time. All nodes have the same cache size which has been configured to fit a default value of 35 documents. Cache sizes with a capacity of 5, 10 and 50 documents have also been simulated aiming at testing the influence of the cache size. In order to avoid 'cold start' influences, that is, cache misses during the initial time of the simulation because the caches are empty; the local caches are 'warmed up' using the first 20% of the simulation time. As the simulation time is set to 20000 s, the warm-up time is set to 4000 s. Consequently, the statistics collected from the simulations are those corresponding to the time after this warm-up period. The Redirection Caching structure has been designed to allocate up to 35 entries.

The 802.1b MAC protocol with the Two Ray Ground propagation model and a coverage radio of 250 m are used. The AODV protocol (with the Local Repair and Intermediate RREP capabilities enabled) is selected as the MANET routing protocol.

5. Performance evaluation

We compare the performance results obtained by our proposal (*CLIR*) with those achieved when using the *MobEye* (*ME*), *SimpleSearch* (*SS*), *COOP*, *DPIP* and *HybridCache* (*HC*) cooperative caching schemes. As the source code for those caching schemes was not available, they have been implemented in NS-2 from scratch. In addition, we also compare those caching schemes with the case of a network without any caching method (No Caching - NC). The parameters and the replacement policy selected for the different caching schemes are those proposed by the corresponding authors. The TDS_T replacement policy is selected for the *SimpleSearch* algorithm because it is the one that obtains the best results. The admission control policy is set to not store in the local cache those documents that are served by a node which is located closer than five hops away. The RRT table size of *COOP* has been dimensioned to store 35 entries (the same value is configured for the *CLIR* Redirection Cache table) as it is not defined in [10]. Similarly, the time to wait for a reply after a broadcast request is set to half the value of the timeout for sending the request again (the study in [10] does not specify this value). The Index Vector table of *DPIP* is able to store information about all the documents in the network and the *DPIP_timer* is set to 150 ms. Finally, for the *HybridCache*, the threshold values for the redirection and documents' TTL have been set to two hops and 2000 s, respectively. For this algorithm, the window size K is set to two and each node is able to store information about the location of all the documents in the network.

In the shown figures, each represented point corresponds to the mean of the results obtained for five simulation runs with the same configuration parameters but different seeds. Each simulation varies the parameters related to the mobility, the starting point and the destination point of every node for every movement. Depending on the simulation, we modify the studied parameter while the rest of parameters are set to the default values. The figures include the 95% confidence interval for the different measured performance parameter.

We evaluate six performance parameters: the number of served documents, the traffic load, the document delay, the percentage of timeouts, the percentage of cache hits and the redirection cache hits. These parameters are calculated as follows:

- *Number of served documents*. It is the mean number of documents that were retrieved by the MNs during the simulation time. As the simulation time and the number of requests are kept constant, this performance parameter indicates which caching scheme is able to serve more documents during the same time.
- *Mean traffic load*. It measures the mean amount of data (in kilo bytes) that each MN generates or forwards. This measurement includes not only the traffic corresponding to document requests and replies, but also the overhead introduced by the routing protocol. Depending on the behaviour of the radio transceiver, this metric can give us an idea of the energy consumption of the MNs which are powered with limited batteries. Moreover, as more traffic is emitted in the network the wireless medium will experience more interferences and radio collisions.
- *Document delay*: It is defined as the time elapsed from the request of a document to the reception of the requested document. This metric only takes into account the requests that were successfully completed.
- *Percentage of timeouts*. It is the percentage of requests that were not served after the corresponding timeout when compared with the total amount of performed requests. The percentage of timeouts can be regarded as an indication of the quality of service perceived by the user because the lower the percentage of timeouts, the greater the accessibility to the documents.
- *Percentage of cache hits*. This metric is divided into local cache hits and remote cache hits. The local cache hits characterize the percentage of requests that were served by the local cache and hence, did not generate any traffic in the network. The remote cache hit is the proportion of requests that were served by a MN that is not a server. This metric describes the reduction of the server load as the greater the remote cache hit the lower the amount of requests served by the server.
- *Redirection cache hit*. It measures the proportion of redirections that were correctly performed, that is, the

occasions in which the requester received the document after the redirection of the request.

The performance of the algorithms is evaluated under different stress conditions: the speed, the number of nodes, traffic load, TTL of the documents, traffic pattern and cache size. Each next subsection describes the results obtained for each set of experiments. The results depicted in this article correspond to the simulations performed using the TVCM mobility model as the performance obtained by the RWP mobility model exhibits quite similar tendencies.

5.1. Effect of the speed

In this set of experiments, we vary the speed of the nodes in order to study how the caching schemes behave when the network topology varies. The topology changes affect the redirection caches as the information stored in them may become stale.

Figure 2a shows the mean number of received documents per node as a function of the speed of the nodes. As it can be observed, *DPIP* is the caching scheme that retrieves the fewest number of documents (even less than the NC scheme). This fact is caused by the high percentage of timeouts (Figure 2d) and the low percentage of redirection hits (Figure 2f). In fact, as the speed of the nodes increases, the number of retrieved documents is also increased because the percentage of redirection hits also rise. This effect is motivated by the fact that increasing the speed of a node eases the acquisition of more information from other nodes as it is in contact with a higher number of the network components. This behaviour is a common factor in all the simulation studies, and hence, *DPIP* will not be shown in the figures depicting the mean number of accessed documents for the other evaluated metrics in order to make the analysis easier. On the other hand, *MobEye*, *CLIR* and *COOP* retrieve a similar number of documents for all the considered speeds. *SimpleSearch* and *HybridCache* obtain a lower number of documents than the other caching schemes except for a speed of 1 m/s.

Figure 2b represents the mean traffic processed per node as a function of the speed of the nodes. As it can be contemplated from this figure, all the caching schemes except *HybridCache* and *CLIR* generate more traffic than the option of not using caching due to the use of broadcast request. *MobEye* performs indiscriminate broadcast request and it is the one that generates more traffic. As the broadcast process is restricted, the traffic generated is reduced as shown by the results obtained by *COOP*, *SimpleSearch* and *DPIP*. The speed of the nodes do not meaningfully affect the traffic load of the tested caching schemes except for *DPIP*, which generates more traffic as the speed increases. As the nodes' speed raises, the neighbourhood lists of *DPIP* are changed frequently and hence the requests cannot be directly sent to the nodes that are supposed to have the documents. Thus, the nodes have to perform a broadcast request more frequently, which intensifies the traffic load.

Figure 2c depicts the mean delay achieved by the requested documents. In this representation, *COOP* has been excluded because it obtains a delay at least five times greater than the other caching schemes. This can be explained by the timer defined by *COOP*. After broadcasting a request, *COOP* waits a certain time before assuming that the broadcast failed. Then, it sends the request directly to DS. This waiting time is a remarkable drawback of *COOP* as the timer must be very finely tuned for each scenario in order to obtain a good performance. In fact, if the timer is selected below the optimal value, the requests will be performed to the DS even before the broadcast requests could be replied. Alternatively, *CLIR* and *HybridCache* obtain the lowest delays. On the other hand, *SimpleSearch* achieves a similar delay to the option of NC because many of the documents are requested directly to DS. In addition, *SimpleSearch* needs four messages in order to retrieve the requested document (*get*, *ack*, *confirm* and *resp*) instead of the two messages (*get*, *resp*) needed by *HybridCache* and our proposal (*CLIR)*. Finally, *MobEye* and *DPIP* get the highest delays obtaining two and three times more delay than *CLIR*, respectively. The high delay of *MobEye* is due to the time elapsed in the exchange of the four messages needed to get a document. On the other hand, the delay of *DPIP* depends on the *DPIP_Timer* as it incurs in a lower delay bound.

Figure 2d represents the mean percentage of timeouts per node. As observed, only *DPIP* obtains a performance poorer than *NC*. *DPIP* assumes that the requests are going to be served because DS is always reachable but this is not always possible. When *DPIP* broadcasts the request to the neighbourhood, every node that receives the request annotates in its table that the requester node will have a copy of the requested document. Unfortunately, the document may be obtained much later because DS is not available. Consequently, the MNs may store incorrect information and subsequent requests for this document can be redirected to a node that does not actually have the document yet. In fact, as the speed of the nodes increases the percentage of timeouts decreases as the information stored in the tables is deleted because the nodes have left the neighbourhood. On the other hand, *MobEye* and *COOP* obtain better performance than *CLIR* as they use broadcast requests. Thus, they find the documents in the network with higher probability at the expense of increasing the traffic load. However, the difference is not significant in scenarios where nodes move with low speeds.

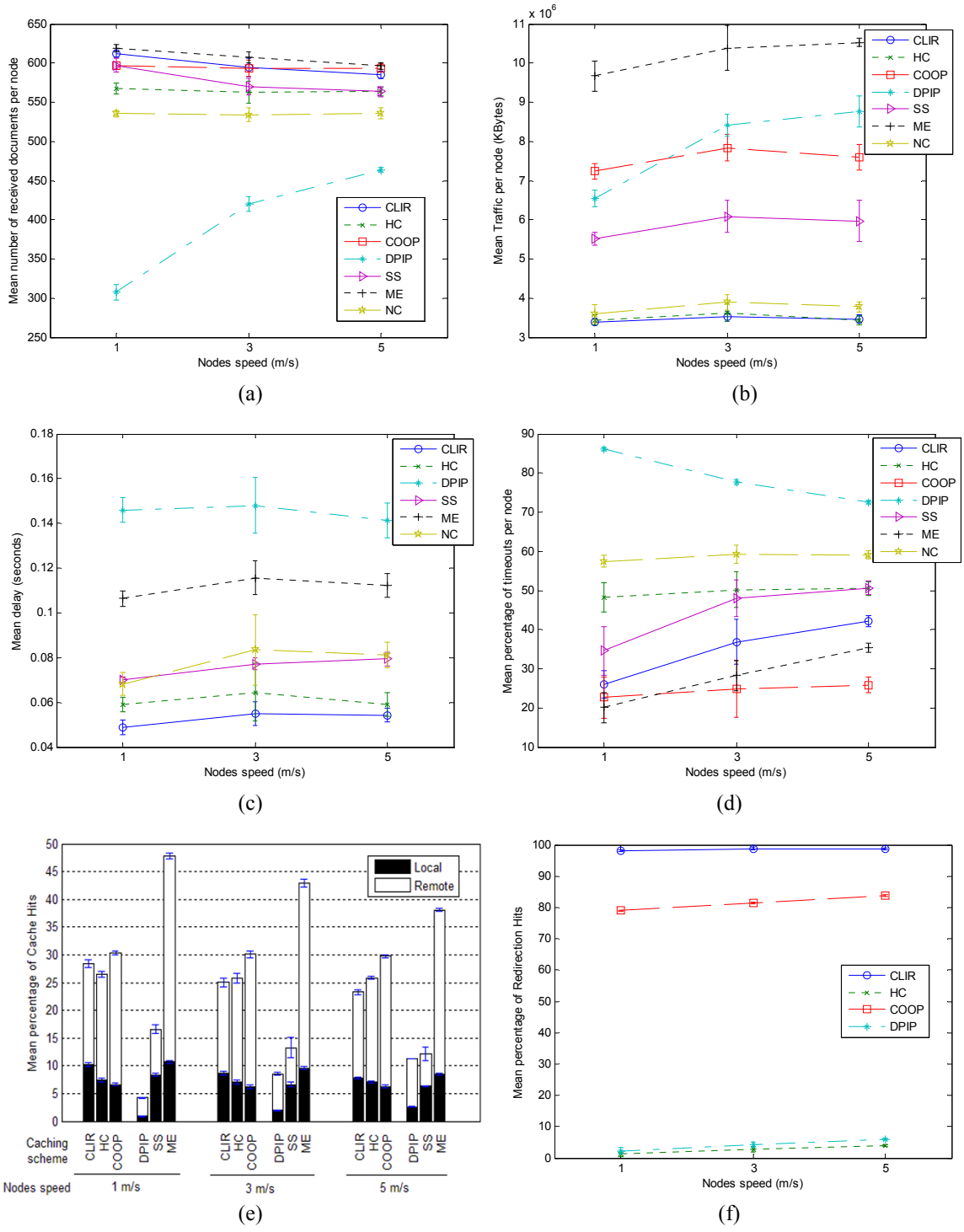

Figure 2 Speed of the nodes evaluation. **(a)** Mean number of received documents. **(b)** Traffic. **(c)** Delay. **(d)** Percentage of timeouts. **(e)** Cache hits. **(f)** Percentage of Redirection Hits.

Figure 2e represents the cache hits, showing both the local and remote cache hits. *DPIP* obtains the lower local cache hit ratio while the rest of caching schemes achieve similar performance. As expected, *MobEye* is the caching scheme with the highest number of remote hits because of the indiscriminate use of the broadcast technique to find the documents in the network. *CLIR*, *COOP* and *HybridCache* behave similarly for low speeds,

although the performance of *CLIR* and *HybridCache* is deteriorated as the speed increases while the *COOP* performance is not altered. *HybridCache* obtains a good remote hit rate at the expense of having a great redirection fail rate as shown in Figure 2f. Finally, *DPIP* and *SimpleSearch* achieve the worst hit ratio although *DPIP* performs better as the speed is increased because the information table that it manages must be updated more frequently. Thus, the information stored about the document location is up to date.

Figure 2f represents the mean percentage of redirection hits. *DPIP* and *HybridCache* redirection hit ratios are very close to zero. This fact demonstrates that the redirection mechanism that they implement is inappropriate. On the other hand, *COOP* and *CLIR* obtain a redirection hit ratio of about 80 and 100%, respectively. Clearly, *CLIR* outperforms the other caching schemes as it hardly ever fails when redirecting requests.

In conclusion, *CLIR* achieves one of the best document retrieval rates but generating the lowest traffic load in the network. The number of timeouts obtained is lower than those achieved by the other caching schemes except for *MobEye* and *COOP*, which perform slightly better but at the expense of doubling or tripling the traffic load. *CLIR* also obtains the lower delay when retrieving the documents. The management of the remote cache does not fail when redirecting requests.

5.2. Effect of the number of nodes

In this set of experiments, we analyse the behaviour of the caching schemes when the density of nodes varies. In this way, we evaluate how scalable the algorithms are.

Figure 3a depicts the mean number of received documents per node as a function of the existing number of MNs in the network. For low-density networks (e.g. with only 25 nodes), *MobEye* and *SimpleSearch* slightly improve the amount of documents downloaded by *COOP* and *DPIP*. On the other hand, although *HybridCache* performs better than the NC scheme, the number of documents retrieved is lower than the other caching schemes. As the node density increases, the previous commented behaviour is maintained for all the caching schemes. However, for networks with more than 50 nodes the performance of the *SimpleSearch* and *MobEye* declines. Conversely, *CLIR*, *COOP* and *HybridCache* maintain their number of retrieved documents. This fall in the performance is due to the increment of the number of timeouts (Figure 3d).

Figure 3b presents the mean traffic processed by the nodes. We can divide the caching schemes in three types as the number of nodes increases. Firstly, although *MobEye* (*ME* in the graphs) is one of the schemes that generates less traffic in a small network (25 nodes), the traffic load is highly increased as more nodes are present in the network. This is caused by the broadcast method used to request the documents. Secondly, the generated traffic using *COOP* and *DPIP* also grows with the number of nodes. Nevertheless, this growth is milder than in the case of *MobEye* because they perform a selective broadcast instead of a broadcast to the whole network. Finally, the increase of traffic load under *CLIR*, *HC*, *SimpleSearch* and *NC* is even smoother than the previously mentioned schemes.

Figure 3c shows the document delay as a function of the number of nodes in the network. *MobEye* and *SimpleSearch* obtain the lower delay in a 25-node network, but as the number of nodes increases, the delay perceived with *MobEye* is highly incremented. However, *SimpleSearch* maintains the delay unaltered. On the other hand, *CLIR* and *HybridCache* delay is reduced for networks with more than 25 nodes. Finally, *DPIP* and *COOP* present a very poor performance compared to the other schemes. This is due to the previously commented timer for *COOP* and the very low cache hit rate obtained by *DPIP* (Figure 3e), respectively.

Figure 3d represents the mean percentage of timeouts. The performance of all the schemes is similar for a network with 25 nodes. In particular, we observe a high percentage of timeouts in low-dense networks. This is due to the fact that the probability of creating a route to DS is very low under these circumstances and the DSs are inaccessible during a non-negligible part of the simulation time. Because of this unreachability, the documents are hardly served by DS and, in turn, the nodes cannot keep copies of the documents in their caches. Thus, the way that caches are managed has no impact on this parameter in low dense networks. As the number of nodes increases the percentage of timeouts is drastically reduced, except for *DPIP*, which does not reduce the percentage of timeouts. In addition, the timeouts for *MobEye* largely increase with a network with 100 nodes. This is again provoked by the broadcast method employed to request the documents. For a higher network density, the amount of broadcast messages and the generated traffic drastically increase (as shown in Figure 3b). Thus, the interferences and the saturation of the network degrade the obtained performance. Finally, we can mention that only *COOP* performs better than *CLIR* for high-density networks in terms of timeouts (Figure 3d).

Figure 3e illustrates the percentage of cache hits as a function of the number of nodes. The behaviour of the caching schemes is similar to the one obtained in Figure 2e except for the network with 25 nodes. As previously mentioned, with this node density, the performance is deteriorated. The remote hit ratio of *SimpleSearch* is reduced as the number of nodes increases because the probability of already having a

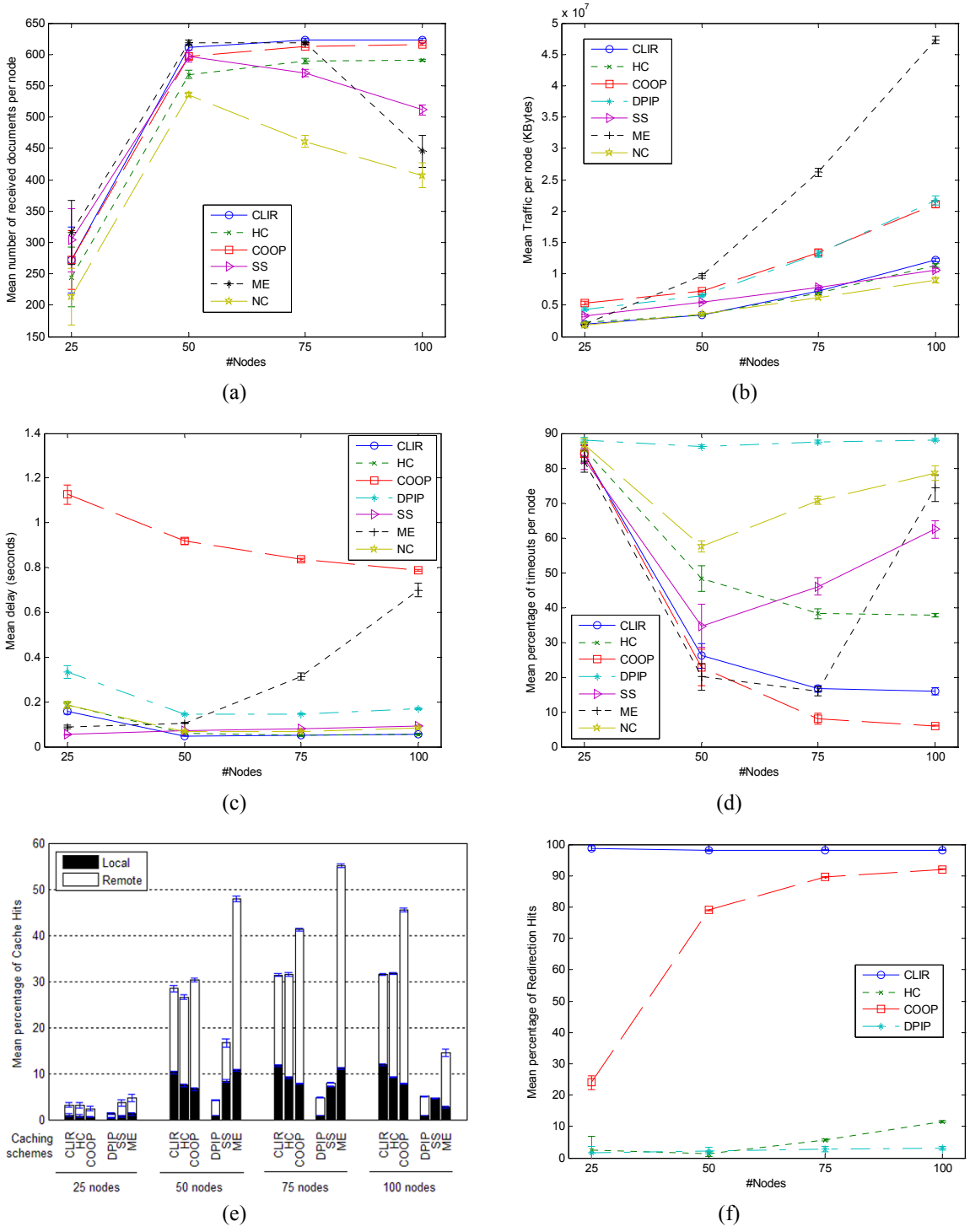

Figure 3 Number of nodes evaluation. **(a)** Mean number of received documents. **(b)** Traffic. **(c)** Delay. **(d)** Percentage of timeouts. **(e)** Cache hits. **(f)** Percentage of Redirection Hits.

created route to DS is also increased. Thus, the broadcast requests are not executed, and hence, the traffic load is also reduced (Figure 3b). On the other hand, the cache hits of *ME* are drastically reduced for a network with 100 nodes. This fact entails the reduction of the number of received documents (Figure 3a) and the increment of the delay (Figure 3c) and the percentage of timeouts (Figure 3d).

Finally, Figure 3f depicts the mean percentage of redirection hits. As in the previous study, the *CLIR* strategy obtains a redirection hit ratio close to 100%. The performance of *COOP* depends on the number of nodes and it is increased as more nodes are available. In addition, *HybridCache* and *DPIP* have a very low performance although the *HybridCache* redirection hits are slightly improved as the number of nodes increases.

As a result, *CLIR* obtains the best retrieval of documents and a low generated traffic, delay and percentage of timeout regardless of the number of nodes in the network. Thus, *CLIR* is observed to be scalable.

5.3. Effect of the mean time between requests

This set of tests is conducted in order to evaluate the performance of the caching schemes when the number of requests is changed and hence, the traffic load is also changed.

Figure 4a represents the mean number of received documents per node as a function of the mean time between requests. For high loaded networks (5 s between requests), *MobEye*, *COOP* and *CLIR* obtain a higher number of documents compared to other solutions. Conversely, *HybridCache* and *SimpleSearch* retrieve fewer documents (with a reduction of about 45%) than the previous caching schemes. As the mean time between requests is increased (and the traffic load decreases), the behaviour is similar although the divergences between the results of the different caching schemes are smoothed.

Similarly, Figure 4b shows the mean traffic processed by the nodes as a function of the traffic load. For high loaded networks (5 s of waiting time), *DPIP* and *SimpleSearch* perform better than the NC option while *HybridCache* exhibits a behaviour comparable to *NC*. *CLIR* introduces a bit more overhead while *COOP* and *MobEye* generates much more traffic load as they use broadcast requests. This difference is only perceived for a time between requests of 5 s. As the mean time between requests is increased, the traffic generation is obviously decreased. Under these circumstances, *CLIR* and *HybridCache* generate a lower traffic load than the option of NC.

Figure 4c depicts the mean delay obtained by all the strategies, excluding the *COOP* scheme. In order to make the analysis easier to follow, we have decided to exclude this result as it obtains a very high delay. *CLIR* gets the minimum delay for all the considered traffic loads. *SimpleSearch* performs in a similar way to the option of NC. On the other hand, *HybridCache* achieves a worse delay than NC for high loaded network although this delay is reduced as the network load is decremented. Finally, *MobEye* and *DPIP* obtain a very high delay.

Figure 4d illustrates the mean percentage of timeouts as a function of the mean time between requests. All the caching schemes, except *DPIP*, obtain a lower percentage of timeout than the NC option. *CLIR* performs better than *SimpleSearch* and *HybridCache* for high loaded networks (5 and 10 s of waiting time between requests) and worse than the broadcast schemes *COOP* and *MobEye*. Anyhow this difference is attenuated as the traffic load is diminished.

Figure 4e represents the percentage of cache hits as a function of the mean time between requests. From the figures, we observe that all the caching schemes except *CLIR*, *HybridCache* and *SimpleSearch* reduce their performance as the traffic load is decreased. *CLIR* and *SimpleSearch* use the broadcast request (*CLIR* at the AODV routing layer level and *SimpleSearch* at the application layer) when there is not a known route to DS. In AODV, routes are removed from the routing table after a period without being used. This period is defined by the Active Route Timeout (by default 10 s). Thus, if the waiting time between requests makes the routes to DS obsolete, *CLIR* is forced to emit a broadcast message. Therefore, the number of remote hits is improved. On the other hand, *HybridCache* obtains a higher redirection hit rate with low loaded networks (Figure 4f) and hence, its remote hit rate is incremented.

Figure 4f shows the mean percentage of redirection hits. As in previous studies, *CLIR* obtains a redirection hit ratio close to 100% although this parameter drops to a 95% for high loaded networks. *COOP* presents a redirection hit ratio of about 70-80% except for a mean time between requests of 5 s where this parameter falls to 50%. Finally, *DPIP* and *HybridCache* obtain a very poor redirection hit (near to 0%) although *HybridCache* achieves a performance close to 30% for very low loaded networks (50 s between requests).

As a result, *CLIR* is the best choice for all the tested traffic loads in terms of the delay. As the traffic load is decreased, the performance of *CLIR*, in terms of traffic generation and percentage of timeouts, is also improved.

5.4. Effect of the mean TTL

This group of experiments studies the influence of the life time of the documents in the performance of the network. The TTL defines how much time the documents could be stored in the local caches before they are considered obsolete.

Figure 5a shows the mean number of received documents per node as a function of the TTL of the documents. *MobEye* and *CLIR* are the policies retrieving more documents for all the considered TTLs, followed by *COOP* and *SimpleSearch*. *HibridCache* retrieves less documents for higher values of the mean TTL because the number of timeouts is increased (Figure 5d).

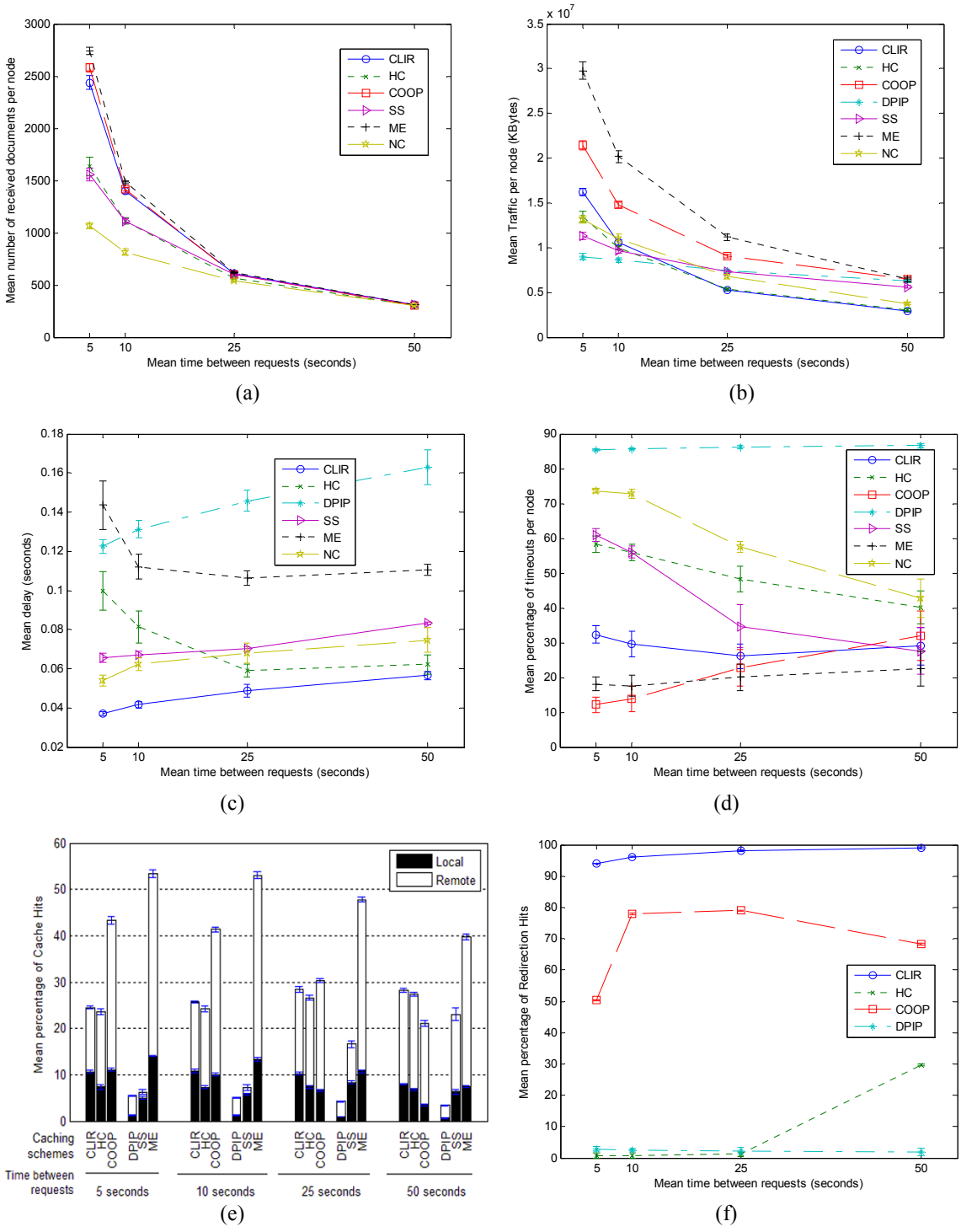

Figure 4 Mean time between requests evaluation. **(a)** Mean number of received documents. **(b)** Traffic. **(c)** Delay. **(d)** Percentage of timeouts. **(e)** Cache hits. **(f)** Percentage of Redirection Hits.

Figure 5b represents the mean traffic processed by the nodes as a function of the mean TTL of the documents. *CLIR* and *HybridCache* generate a similar traffic load for all the tested values of the TTL, although the traffic load is reduced when the documents live longer. In any case, they perform better than the option of NC. On the other hand, the rest of caching schemes generate more traffic than when NC is applied, specially *COOPS* and *ME*.

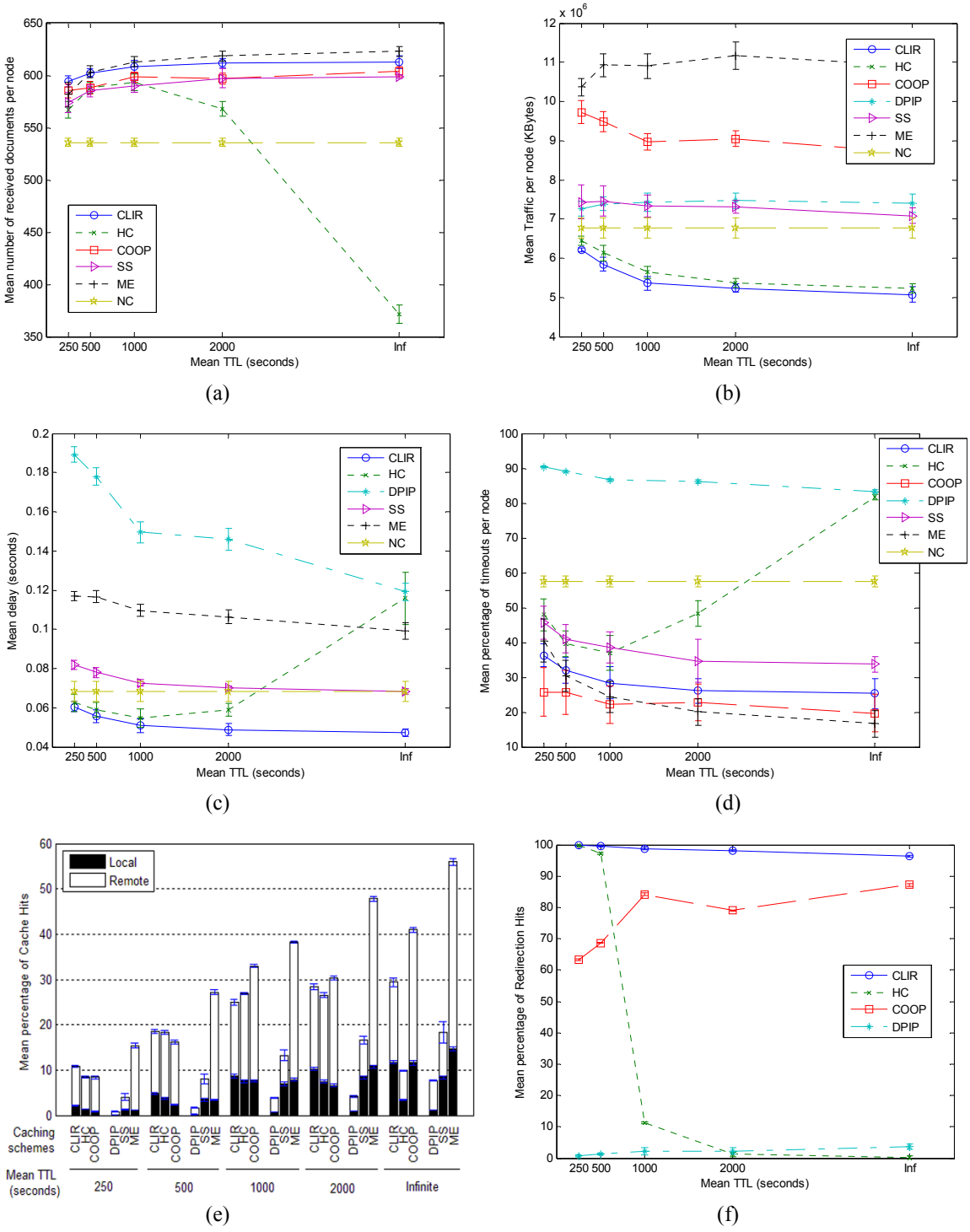

Figure 5 Mean TTL of the documents evaluation. (a) Mean number of received documents. (b) Traffic. (c) Delay. (d) Percentage of timeouts. (e) Cache hits. (f) Percentage of Redirection Hits.

Figure 5c depicts the mean delay as a function of the mean TTL of the documents. *CLIR* and *HybridCache* obtain a similar delay with very short-living documents. Conversely, as the TTL increases, the delay obtained by *HybridCache* augments, particularly if the documents never expire. This is caused by the incorrect management of the redirection policy, which is shown in Figure 5f. However, *CLIR* maintains its delay which is even

reduced if the validity of the documents is longer as the redirection technique is more effective. On the other hand, *SimpleSearch* performs similar to the option of NC. Finally, *MobEye*, *DPIP* and specially *COOP* obtain a very high delay compared to the option of NC.

Figure 5d illustrates the mean percentage of timeouts as a function of the mean TTL of the documents. The observed behaviour is comparable to the previous studies except for the *HybridCache* caching scheme. As previously commented, *HybridCache* is very sensitive to the TTL of the documents and hence, the number of timeouts rises as the TTL is incremented.

As the TTL increases, the storage of the documents in the cache is more useful. In this sense, Figure 5e shows that the cache (remote and local) hits increments when the TTL is increased. The schemes with the greatest cache hits are those supported by broadcast requests. This can be explained by the fact that these policies are able to find the documents in their local or remote caches as the documents are stored during more time.

The mean percentage of redirection hits as a function of the mean TTL of the documents is shown in Figure 5f. The *CLIR*, *COOP* and *DPIP* behaviours are the same as in the previous studies. However, *HybridCache* presents a high dependence on the mean TTL. The redirection mechanism works better with short-living documents (low TTL) but the performance degrades as the TTL increases.

As a conclusion, in the tested scenarios *CLIR* obtains the minimum delay for retrieving the documents. The scheme incurs in the lower overhead while it is able to retrieve a high number of documents.

5.5. Effect of the Zipf slope

In this set of experiments, we evaluate the impact of the Zipf slope on the performance of the caching schemes. The Zipf slope (α, see Equation (4)) characterizes the popularity of the documents, so when this parameter is higher, the frequency of requests for the most popular documents increases.

Figure 6a shows the mean number of retrieved documents per node as a function of the slope of the Zipf probability. *MobEye*, *CLIR*, *COOP* and *SimpleSearch* obtain between 620 and 580 documents for all the slopes. They retrieve more documents when the Zipf slope is close to one as the local and remote caches increase their performance as shown in Figure 6e. *HybridCache* obtains a number of documents similar to those of the NC scheme although its performance is increased significantly with the Zipf slope.

On the other hand, Figure 6b depicts the mean processed traffic per node as a function of the Zipf slope. As in previous studies, *CLIR* and *HybridCache* generate less traffic than the NC scheme, whereas the rest of caching schemes require a higher overhead in order to obtain an equivalent performance. In fact, the differences between *CLIR* and *HybridCache* with respect to the other caching schemes are increased with the Zipf slope.

Figure 6c compares the mean document delay. Only *CLIR* obtains a lower delay than that achieved with the NC scheme for all the slopes. However, *SimpleSearch* and *HybridCache* also reduce their delay to values lower than the NC option for slopes of 0.8 and 1.0, respectively. On the other hand, *MobEye* and *DPIP* obtain delays that double or triple the delay obtained by *CLIR*. Finally, the delay achieved by *COOP* is not depicted as it is close to one second.

In Figure 6d, the mean percentage of timeouts per node as a function of the Zipf slope is illustrated. As in previous studies, the broadcast-based schemes *MobEye* and *COOP* perform better than *CLIR*, although this improvement is decreased as the Zipf slope is close to one because of the local hits. *HybridCache* and *SimpleSearch* also obtain a lower timeout than the NC scheme. Nevertheless, *HybridCache* performs very similar to NC for lower slopes. Finally, *DPIP* is the caching scheme with the highest number of timeouts.

As the Zipf slope is increased the most popular documents are requested more times, and hence, the local cache will increase the ratio of local hits. The local cache hits do not generate any traffic in the network and the delay perceived by the user is zero. Figure 6e demonstrates this behaviour. The local cache hit of *CLIR* and *ME* is greater than the local cache of the other schemes for all the Zipf slopes.

Figure 6f represents the mean percentage of redirection hits as a function of the Zipf slope. *CLIR* obtains a redirection hit close to 100%. On the other hand, *COOP* achieves a performance from 85 to 70% as the Zipf slope is increased. Finally, as in the studies with the other parameters, *HybridCache* and *DPIP* obtain a percentage redirection hit close to 0% as they fail to redirect the requests.

As a conclusion, the proposed algorithm (*CLIR*) takes advantage of the popularity of the objects obtaining a high local and interception cache hit rate. Consequently, *CLIR* achieves a document retrieval in the same order as the broadcast-based caching schemes but notably reducing the traffic load and delay.

5.6. Effect of the cache size

The cache size is an important factor to take into account as the larger the cache, the more documents can fit into it. Thus, more documents can be stored and the percentage of cache hits and redirection hits can be incremented.

Figure 7a shows the mean number of received documents as a function of the cache size. *ME*, *CLIR*, *COOP* and *SimpleSearch* receive more documents than the NC

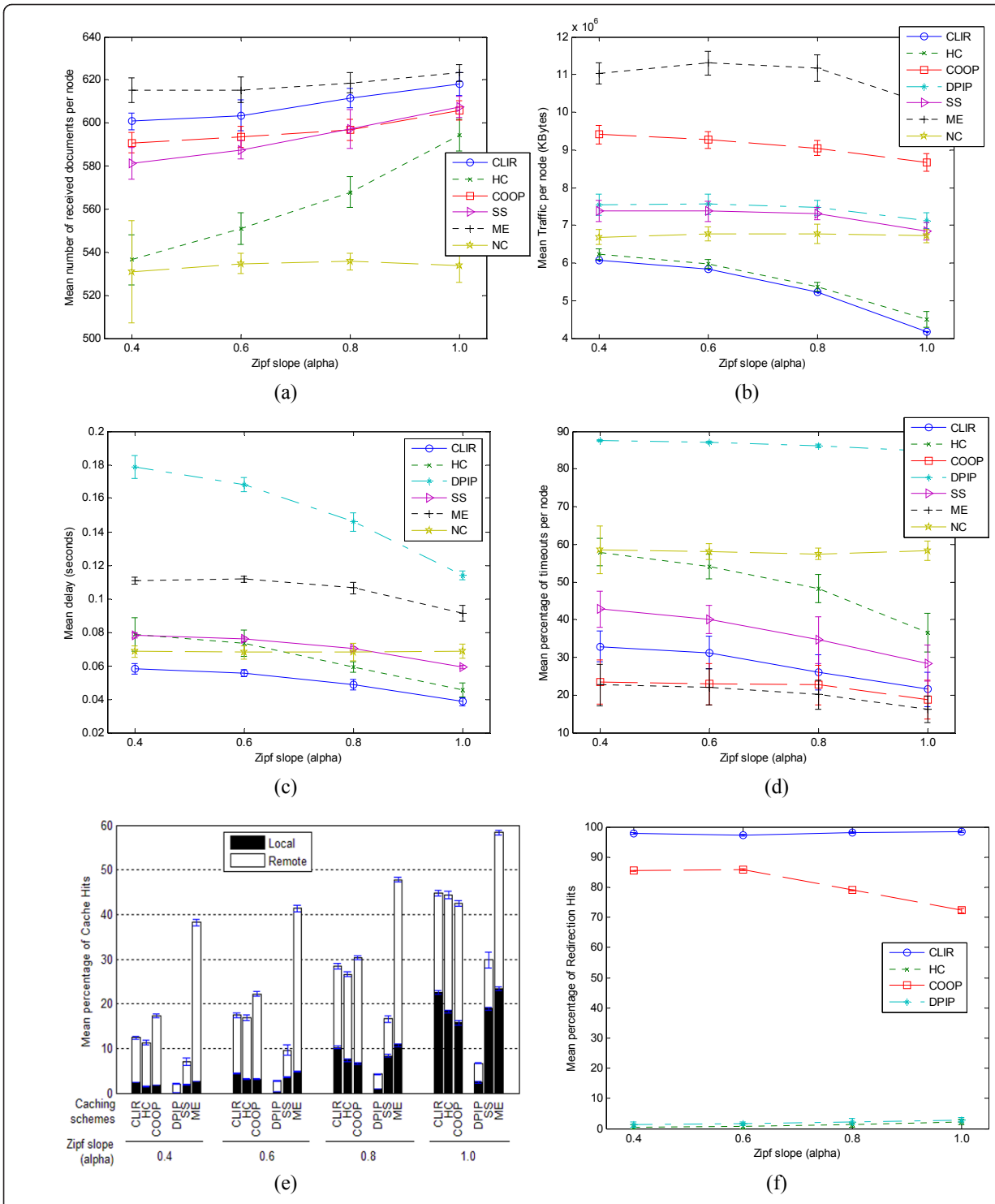

Figure 6 Slope of the Zipf distribution evaluation. **(a)** Mean number of received documents. **(b)** Traffic. **(c)** Delay. **(d)** Percentage of timeouts. **(e)** Cache hits. **(f)** Percentage of Redirection Hits.

option for all the cache sizes, respectively. On the other hand, the number of received documents in the *HybridCache* caching scheme is even lower than the NC scheme for cache sizes of 10 and 20 documents. In any case, the performance of *HybridCache* is inferior to the rest of the studied caching schemes.

Taking into account the mean traffic processed by the MNs as a function of the cache size, Figure 7b illustrates

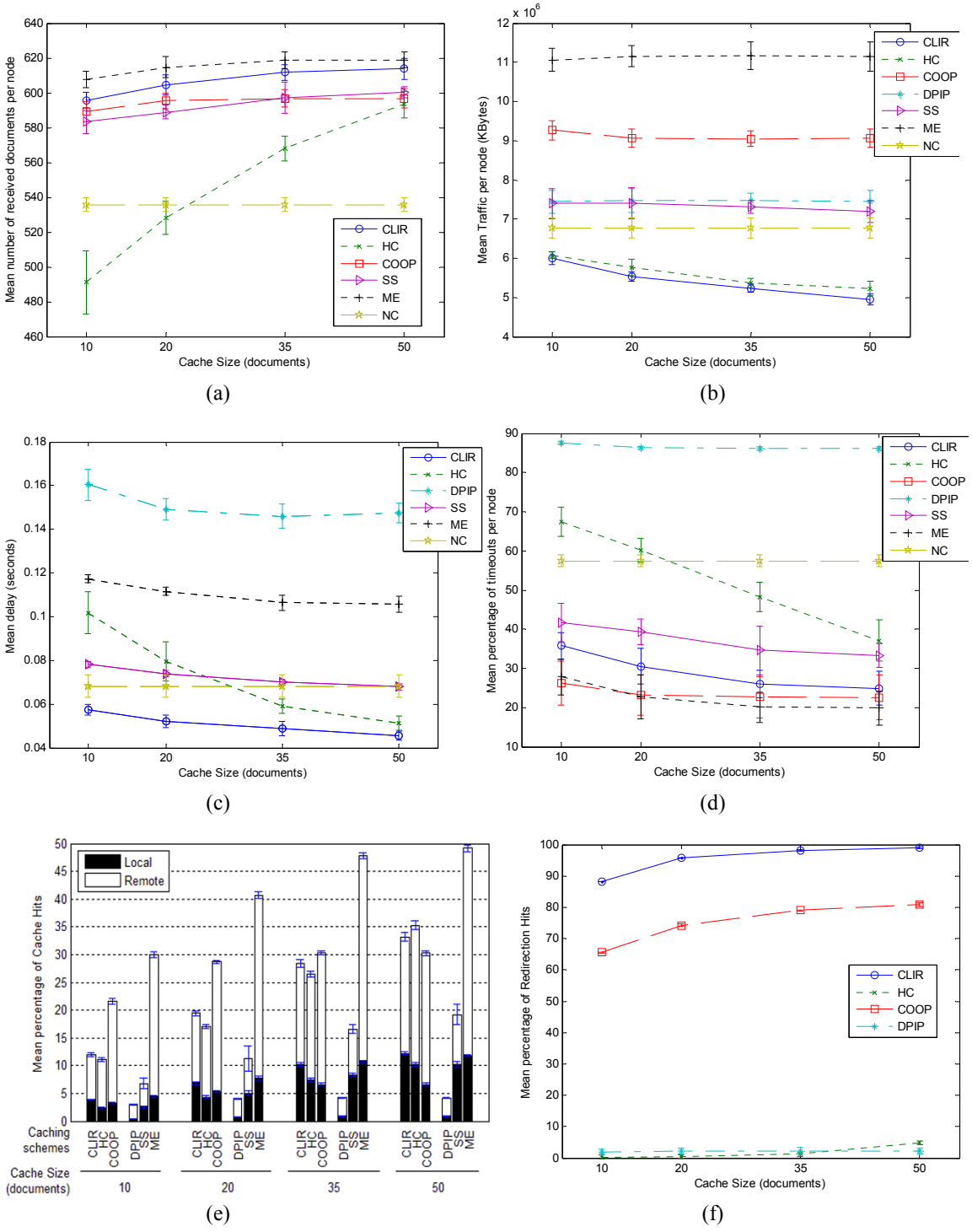

Figure 7 Cache size evaluation. **(a)** Mean number of received documents. **(b)** Traffic. **(c)** Delay. **(d)** Percentage of timeouts. **(e)** Cache hits. **(f)** Percentage of Redirection Hits.

the same behaviour as the previous studies. *CLIR* and *HybridCache* generate a lower traffic than the NC option. For these schemes, the generated traffic is diminished as the cache size augments because the local cache hits are increased (Figure 7e). The traffic generated by *SimpleSearch* and *DPIP* is similar to NC. In addition, *COOP* and *MobEye* increase the generated traffic in a 50 and 100%, respectively, when compared

with *CLIR*. Finally, the cache size does not influence the traffic load except for *HybridCache* and *CLIR*, which generate less traffic as the cache size is increased.

Figure 7c depicts the mean delay as a function of the cache size. *CLIR* is again the best caching scheme, although *HybridCache* obtain a similar delay for bigger cache sizes. *HybridCache* performs worse than the NC scheme for cache sizes of 10 and 20 documents due to the redirection cache hit, which is zero for those cache sizes (Figure 7f). This fact implies that the redirections always fail, and hence, the timeout will be triggered (Figure 7d) or the requests will be incorrectly redirected and finally forwarded again to DS. On the other hand, the delay obtained by *MobEye* and *DPIP* is about 100 or 150% greater than that of *CLIR*.

Figure 7d illustrates the mean percentage of timeouts as a function of the cache size. The percentage of timeouts of *COOP* and *MobEye* is lower than that of *CLIR*, although this difference is decreased as the cache size is increased and more documents can be stored in the cache. *SimpleSearch* obtains a better performance than the NC scheme, but it is slightly worse than *CLIR*. As commented before, the percentage of timeouts of *HybridCache* is even greater than the NC option for lower cache sizes. Finally, *DPIP* obtains the poorest performance.

Figure 7e shows the cache hits as a function of the cache size. As it could be expected, the number of local and remote cache hits is increased as more documents can be stored in the local caches.

Figure 7f represents the mean percentage of redirection hits as a function of the cache size. As the cache size is decreased, the number of documents that can be stored is decreased and the number of replacements increases. Thus, the probability of failing when a redirection process is performed is increased. For a cache size of 10 documents, the percentage of redirection hit is about 90% for *CLIR* and about 65% for COOP. For higher cache sizes, the redirection hits increases reaching values close to 100% for *CLIR* and 80% for *COOP*.

Under the tested conditions, *CLIR* obtains the lowest delay for all the cache sizes. In addition, the generated traffic load is also the lowest but retrieving a similar number of documents to that of the broadcast based caching schemes. Finally, the percentage of timeouts is reduced as the cache size increases until it is comparable to the one obtained by the broadcast based schemes.

6. Conclusions

In this article, the *CLIR* cooperative caching scheme has been presented. The scheme is proposed to improve the retrieval of information in MANETs. *CLIR* is an information and direct request caching scheme that implements a local cache in every mobile in the wireless network. This local cache can directly serve the documents that are requested by the MN as well as the requests that are forwarded by the other nodes in the MANET (request interception). Thus, a MN can reply to a node requesting a certain document using the copy stored in its local cache instead of forwarding the request to the DS*s*. On the other hand, the MNs manage information about the location of the documents disseminated in the network analysing the messages they forward. Using this information, the MNs may decide to redirect the requests to other MN that are closer to the requesting node than the DS*s*. In addition, the MNs also take advantage of the creation of the routes to the DS*s* servers in order to find the documents in the network.

We have evaluated the performance of the proposed caching scheme using different metrics: number of retrieved documents during the simulation time, traffic processed by each node, delay, response timeouts, cache hits and redirection cache hits. This evaluation has been conducted studying the influence of the speed of the nodes, the number of nodes in the network, the traffic load, the mean lifetime of the documents, the request pattern and the cache size. In addition, we have compared our proposal with the *HybridCache*, *COOP*, *DPIP*, *SimpleSearch* and *MobEye* caching schemes. Moreover, we have also compared the previous caching schemes with the option of NC.

From the extensive set of simulations that we have performed, we can conclude that the *CLIR* caching scheme retrieves a similar or even better amount of documents compared to the *COOP* and *MobEye* broadcast based caching schemes. However, *CLIR* is the caching scheme that generates less traffic load in the network, while *COOP* and *MobEye* achieve good results at the cost of overloading the MANET. On the other hand, *CLIR* is also the caching scheme that obtains the lowest delay for all the studied variables. Taking into account the percentage of timeouts, *COOP* and *MobEye* only achieve a better performance than *CLIR* for some of the studied parameters at the expenses of generating much more protocol traffic. These improvements make *CLIR* especially suitable for MANETs where the wireless medium offers a constrained bandwidth while the users may demand the same QoS requirements than in a wired network. Finally, *CLIR* achieves a local cache hit similar to the other caching schemes but obtaining the best redirection caching ratio, as it is always close to 100%. This result demonstrates that the management of the redirection information is performed efficiently.

From our study, we have also extracted the following conclusions:

- *DPIP* is the worse caching scheme for all the studied parameters. This is due to the supposition that the DS*s* are always accessible. This condition does not always hold in practical MANETs. In addition, the defined *DPIP_Timer* parameter value is not enough in some circumstances as the reply messages takes more time to be received than this timer.
- *MobEye* and *SimpleSearch* are not suitable for dense networks because of the use of broadcast requests.
- *HybridCache* is extremely sensitive to the mean lifetime of the documents and cache size as the redirection caching algorithm fails when the documents have a large expiration time or the cache size is small.
- The delay obtained by *COOP* depends on the selected timeout before sending the requests to the DSs.

Acknowledgements
We would like to thank Adela Isabel Fernandez Anta for revising the syntax and grammar of this article. This study was partially supported by the National Project No. TEC2009-13763-C02-01.

Competing interests
The authors declare that they have no competing interests.

Received: 11 July 2011 Accepted: 24 February 2012
Published: 24 February 2012

References

1. G Dodero, V Gianuzzi, Saving energy and reducing latency in MANET file access, in *Proceedings of the 26th International Conference on Distributed Computing Systems Workshops (ICDCSW'06)*, Lisbon, 15 4–7 July 2006
2. S Lim, WC Lee, G Cao, CR Das, A novel caching scheme for improving Internet-based mobile ad hoc networks performance. Ad Hoc Netw. **4**(2), 225 (2006). doi:10.1016/j.adhoc.2004.04.013
3. M Fiore, F Mininni, C Casetti, DF Chiasserini, To cache or not to cache? in *Proceedings of the IEEE Conference on Computer and Communications (INFOCOM 2009)*, Rio de Janeiro, Brazil 235 (2009)
4. B Tang, H Gupta, SR Das, Benefit-based data caching in ad hoc networks. IEEE Trans Mobile Comput. **7**(3), 289 (2008)
5. YH Wang, J Chen, CF Chao, C Chuang, A distributed data caching framework for mobile ad hoc networks, in *Proceedings of the 2006 International Conference on Wireless Communications and Mobile Computing*, Vancouver, Canada 1357 (2006)
6. N Chand, RC Joshi, M Misra, Cooperative caching in mobile ad hoc networks based on clusters. Int J Wirel Personal Commun. **43**(1), 41–46 (2007). doi:10.1007/s11277-006-9238-z
7. MK Denko, Cooperative data caching and prefetching in wireless ad hoc networks. Int J Bus Data Commun Netw. **3**(1), 1–15 (2007)
8. V Gianuzzi, File distribution and caching in MANET, Technical Report, DISI-TR-03-03 (DISI Tech University of Genova, 2003)
9. Y Du, S Gupta, COOP—a cooperative caching service in MANETs in *Proceedings of the Joint International Conference on Autonomic and Autonomous Systems and International Conference on Networking and Services (ICAS-ICNS 2005)*, Papeete, 58 23–28 (October 2005)
10. G Chiu, C Young, Exploiting in-zone broadcast for cache sharing in mobile ad hoc networks. IEEE Trans Mobile Comput. **8**(3), 384 (2009)
11. H Artail, H Safa, K Mershad, Z Abou-Atme, N Sulieman, COACS: a cooperative and adaptive caching systems for MANETs. IEEE Trans Mobile Comput. **7**(8), 961 (2008)
12. L Yin, G Cao, Supporting cooperative caching in ad hoc networks. IEEE Trans Mobile Comput. **5**(1), 77 (2006)
13. Y Ting, Y Chang, A novel cooperative caching scheme for wireless ad hoc networks: GroupCaching. in *Proceedings of the 2nd International Conference on Networking, Architecture and Storage (NAS 2007)*, Guilin, 62, 29–31 July 2007
14. N Chand, RC Joshi, M Misra, Cooperative caching in mobile ad hoc networks based on data utility. Mobile Inf Syst. **3**(1), 19 (2007)
15. J Zhao, P Zhang, G Cao, CR Das, Cooperative caching in wireless P2P networks: design, implementation and evaluation. IEEE Trans Parallel Distrib Syst. **21**(2), 229 (2010)
16. S Podlipnig, L Boszormenyi, A survey of web cache replacement strategies. ACM Comput Surv. **35**(4), 374 (2003). doi:10.1145/954339.954341
17. J Wang, A survey of web caching schemes for the Internet. ACM SIGCOMM Comput Commun Rev. **19**(5), 36 (1999)
18. A Hoof, Top40 cache algorithm compared to LRU and LFU, SNE MSc Research Project (2009)
19. CE Perkins, EM Belding-Royer, S Das, Ad hoc on demand distance vector (AODV) routing. IETF RFC 3561 (2003)
20. FJ González-Cañete, E Casilari, A Triviño-Cabrera, in *AdHoc-Now 2009: Proposal and evaluation of a caching scheme for ad hoc networks*, ed. by PM Ruiz, JJ Garcia-Luna-Aceves.*VIII International Conference on Ad Hoc Networks and Wireless*, Murcia, 23–25 September 2009. Lecture Notes in Computer Science, vol. 5793 (Springer, Heidelberg, 2009), p. 366
21. FJ González-Cañete, E Casilari, A Triviño-Cabrera, Proposal and evaluation of an application level caching scheme for ad hoc networks, in *Proceedings of the 5th International Wireless Communications and Mobile Computing Conference (IWCMC'09)*, Leipzig, 952 21–24 June 2009
22. FJ González-Cañete, E Casilari, Impact of the mobility model on a cooperative caching scheme for mobile ad hoc networks, ed. by Xin Wang *Moble Ad-Hoc Networks: Applicatios* (IntTech Publisher, Rijeka Croatia, 2011), p. 265
23. S Kurkowski, T Camp, M Colagrosso, MANET simulation studies: the incredibles. ACM Mobile Comput Commun Rev. **9**(4), 50 (2005). doi:10.1145/1096166.1096174
24. J Yoon, M Liu, B Noble, Random waypoint considered harmful, in *Proceedings of the 22nd Annual Joint Conference of the IEEE Computer and Communications (INFOCOM 2003)*, San Francisco, **2** 1312 (30 March–3 April 2003)
25. W Hsu, T Spyropoulos, K Psounis, A Helmy, Modeling time-variant user mobility in wireless mobile networks, in *Proceedings of the IEEE Conference on Computer and Communications (INFOCOM 2007)*, Anchorage, 758 (6–12 May 2007)
26. Weijen's Web site for Time-variant Community Mobility Model, http://nile.cise.ufl.edu/~weijenhs/TVC_model/. Accessed March 2011
27. LA Adamic, BA Huberman, Zipf's law and the Internet, Glottometrics **3**, 143 (2002)

doi:10.1186/1687-1499-2012-63
Cite this article as: González-Cañete *et al.*: A cross layer interception and redirection cooperative caching scheme for MANETs. *EURASIP Journal on Wireless Communications and Networking* 2012 **2012**:63.