Mobile Netw Appl (2008) 13:337–356
DOI 10.1007/s11036-008-0063-3

Cooperative Caching in Wireless Multimedia Sensor Networks

Nikos Dimokas · Dimitrios Katsaros · Yannis Manolopoulos

Published online: 11 June 2008

Abstract The recent advances in miniaturization and the creation of low-power circuits, combined with small-sized batteries have made the development of wireless sensor networks a working reality. Lately, the production of cheap complementary metal-oxide semiconductor cameras and microphones, which are able to capture rich multimedia content, gave birth to what is called *Wireless Multimedia Sensor Networks* (*WMSNs*). WMSNs will boost the capabilities of current wireless sensor networks, and will fuel several novel applications, like multimedia surveillance sensor networks. WMSNs introduce several new research challenges, mainly related to mechanisms to deliver application-level Quality-of-Service (e.g., latency minimization). To address this goal in an environment with extreme resource constraints, with variable channel capacity and with requirements for multimedia in-network processing, the caching of multimedia data, exploiting the cooperation among sensor nodes is vital. This article presents a cooperative caching solution particularly suitable for WMSNs. The proposed caching solution exploits sensor nodes which reside in "positions" of the network that allow them to forward packets or communicate decisions within short latency. These so-called "mediator" nodes are selected dynamically, so as to avoid the creation of hot-spots in the communication and the depletion of their energy. The mediators are not more powerful than the rest of the nodes, but they have some special role in implementing the cooperation among the sensors. The proposed cooperative caching protocol includes components for locating cached data as well as for implementing data purging out of the sensor caches. The proposed solution is evaluated extensively in an advanced simulation environment, and it is compared to the state-of-the-art cooperative caching algorithm for mobile ad hoc networks. The results confirm that the proposed caching mechanism prevails over its competitor.

Keywords cooperative caching · replacement policy · multimedia · wireless sensor networks

Research supported by a *Γ.Γ.Ε.Τ.* grant in the context of the project "Data Management in Mobile Ad Hoc Networks" funded by *ΠΥΘΑΓΟΡΑΣ ΙΙ* national research program.

N. Dimokas · Y. Manolopoulos
Department of Informatics, Aristotle University, Thessaloniki, Greece

N. Dimokas
e-mail: dimokas@csd.auth.gr

Y. Manolopoulos
e-mail: manolopo@csd.auth.gr

D. Katsaros (✉)
Department of Computer & Communication Engineering, University of Thessaly, Volos, Greece
e-mail: dimitris@delab.csd.auth.gr

1 Introduction

Wireless Sensor Networks [2, 18] (WSNs) have emerged during the last years due to the advances in low-power hardware design and the development of appropriate software, that enabled the creation of tiny devices which are able to compute, control and communicate with each other. A WSN consists of wirelessly interconnected devices that can interact with their environment by controlling and sensing "physical" parameters. WSNs attracted a huge interest from both the

research community and the industry, that continues to grow. This growing interest can be attributed to the many new exciting applications that were born as a result of the deployment of large-scale WSNs. Such applications range from disaster relief, to environment control and biodiversity mapping, to machine surveillance, to intelligent building, to precision agriculture, to pervasive health applications, and to telematics.

The support of such a huge range of applications will be (rather) impossible for any single realization of a WSN. Nonetheless, certain common features appear, in regard to the characteristics and the required mechanisms of such systems and the realization of these characteristics is the major challenge faced by these networks. The most significant characteristics shared by the aforementioned applications concern [18]:

- Lifetime: Usually, sensor nodes rely on a battery with limited lifetime, and their replacement is not possible due to physical constraints (they lie in oceans or in hostile environments) or it is not interesting for the owner of the sensor network.
- Scalability: the architecture and protocols of sensor networks must be able to scale up (or to exploit) any number of sensors.
- Wide range of densities: the deployment of sensor nodes might not be regular and may vary significantly, depending on the application, on the time and space dimension and so on.
- Data-centric networking: the target of a conventional communication network is to move bits from one machine to another, but the actual purpose of a sensor network is to provide information and answers, not numbers [17].

Recently, the production of cheap complementary metal-oxide semiconductor (CMOS) cameras and microphones, which can acquire rich media content from the environment, created a new wave into the evolution of WSNs. For instance, the Cyclops imaging module [29] is a light-weight imaging module which can be adapted to MICA2[1] or MICAz sensor nodes. Thus, a new class of WSNs came to the scene, the *Wireless Multimedia Sensor Networks* (*WMSNs*) [1]. These sensor networks, apart from boosting the existing application of WSNs, will create new applications: a) multimedia surveillance sensor networks which will be composed by miniature video cameras [22] will be able to communicate, to process and store data relevant to crimes and terrorist attacks; b) traffic avoidance and control systems will monitor car traffic and offer routing advices to prevent congestion; c) industrial process control will be realized by WMSNs that will offer time-critical information related to imaging, temperature, pressure, etc.

The novel applications of WMSNs challenged the scientific community because, as it is emphasized in [1], these applications require us to rethink the computation-communication paradigm of traditional WSNs. This paradigm has mainly focused on reducing the energy consumption, targeting to prolong the longevity of the sensor network. Though, the applications implemented by WMSNs have a second goal, as important as the energy consumption, to be pursued; this goal is the delivery of application-level quality of service (QoS) and the mapping of this requirement to network layer metrics, like latency. This goal has (almost) been ignored in mainstream research efforts on traditional WSNs.

The goal of Internet QoS in multimedia content delivery has been pursued in architectures like Diffserv and Intserv, but these protocols and techniques do not face the severe constraints and hostile environment of WSNs. In particular, WMSNs are mainly characterized by:

- Resource constraints: sensor nodes are battery-, memory- and processing-starving devices.
- Variable channel capacity: the multi-hop nature of WMSNs, which operate in a store-and-forward fashion because of the absence of base stations, implies that the capacity of each wireless link depends on the interference level among nodes, which is aggravated by the broadcasting operations.
- Multimedia in-network processing: is many applications of WMSNs, a single sensor node is not able to answer a posed question, but several sensor must collaborate to answer it. For instance, a sensor node with a camera monitoring a moving group of people, can not count their exact number and determine their direction, but it needs the collaboration of nearby sensors in order to cover the whole extent of the group of people. Therefore, sensor nodes are required to store rich media, e.g., image, video, needed for their running applications, and also to retrieve such media from remote sensor nodes with short latency.

Under these restrictions/requirements, the goal of achieving application-level QoS in WMSNs becomes a very challenging task. There could be several ways to attack parts of this problem, e.g., channel-adaptive streaming [14], joint source-channel coding

[1] http://www.xbow.com.

[11]. Though, none of them can provide solutions to all of the three aforementioned issues. In this paper, we investigate the solution of *cooperative caching multimedia content* in sensor nodes to address all three characteristics. In cooperative caching, multiple sensor nodes share and coordinate cache data to cut communication cost and exploit the aggregate cache space of cooperating sensors. The plain assumption we make, is that each sensor node has a moderate local storage capacity associated with it, i.e., a flash memory. Although, there exist flash memories with several gigabytes storage capacity, e.g., the NAND flash [23] and the trend is that they become cheaper, larger and more common, we do not assume extreme storage capabilities, so as to capture a broader field of applications and architectures.

Since the battery lifetime can be extended if we manage to reduce the "amount" of communication, caching the useful data for each sensor either in its local store or in the near neighborhood can prolong the network lifetime. Additionally, caching can be very effective in reducing the need for network-wide transmissions, thus reducing the interference and overcoming the variable channel conditions. Finally, it can speed-up the multimedia in-network processing, because, as it is emphasized in [1], the processing and delivery of multimedia content are not independent and their interaction has a major impact on the levels of QoS that can be delivered.

This paper investigates for the first time the technique of caching in the context of WMSNs. The need for effective and intelligent caching policies in sensor networks has been pointed out several times [10, 25] in the very recent literature, but no appropriate sophisticated policies have been proposed, although there are quite a lot of caching protocols in other fields (see relevant work in Section 2). This article proposes a novel and high-performance cooperative caching protocol, the *NICoCa* protocol named after the words *Node Importance-based Cooperative Caching*, and compares it with the state-of-the-art cooperative caching policy for mobile ad hoc networks (MANETs), which is the "closer" competitor. Using the J-Sim simulation environment [34], we perform an experimental evaluation of the two methods, which attests that the proposed *NICoCa* cooperative caching policy prevails over its competitor.

The rest of this article is organized as follows: in Section 2 we review the relevant work and record the contributions of the article. In Section 3 we present the details of the *NICoCa* protocol, and in Section 4 we present the results of the performance evaluation of the methods; finally, Section 5 concludes the paper.

2 Relevant work

The technique of caching has been widely investigated in the context of operating systems and databases and is still an attractive research area [24]. Similarly, caching on the Web has been thoroughly investigated for cooperative [12] and for non-cooperative [19] architectures. Wessels and Claffy [37] introduced the Internet cache protocol; Cache digests [30] and Summary Cache [12] enable proxies to exchange information about cached content. In [6] a cooperative hierarchical Web caching architecture was studied. However, the above architectures and protocols usually assume a fixed network topology and require powerful computation and communication capabilities.

In the context of wireless broadcast cellular networks, a number of caching approaches have been proposed [20]. These policies assume more powerful nodes than the sensor nodes, and one-hop communication with resource-rich base stations, which serve the needed data.

A number of data replication schemes [15, 16] and caching schemes [31, 35, 38] have been proposed in order to facilitate data access in MANETs. Data replication studies the issue of allocating replicas of data items to meet access demands. These techniques normally require a priori knowledge of the network topology.

Caching schemes however do not facilitate data access based on the knowledge of distributed data items. In SimpleCache [38] the requested data item has always been cached by the requester node. The node uses the cached copy in order to serve subsequent requests when they arrive. The requester node has to get the data from the data center in case of cache miss. However increasing the hop distance between the requester node and caching node will increase the response time for the request.

The most relevant research works to our protocol are the cooperative caching protocols which have been developed for MANETs. The main motive for the development of these protocols is the mobility of the nodes, and thus they all strive to model it or exploit it. Recently, a cooperative caching scheme, called CoCa, was proposed in [7, 8]. The CoCa framework facilitate mobile nodes to share their cached contents with each other in order to reduce the number of server requests and the number of access misses. The authors extended CoCa with a group-based cooperative caching scheme, called GroCoCa, in [9]. According to GroCoCa, the decision of whether a data item should be cached depends on two factors of the access affinity on the data items and the mobility of each node. The mobile

support station performs an incremental clustering algorithm to cluster the mobile nodes into tightly coupled groups based on their mobility patterns. In GroCoCa also the similarity of access patterns is captured by frequency-based similarity measurement. GroCoCa improves system performance at the cost of extra power consumption. Papadopouli and Schulzrinne [26] suggested the 7DS architecture. The authors deployed a couple of protocols in order to facilitate sharing and dissemination of information among users. It operates on two modes. The first one is a prefetch mode, based on the information and user's future needs and the second one is an on-demand mode, which searches for data items in a one hop multicast basis. Depending on the collaborative behavior, a peer-to-peer and server-to-client mode are used. Unlike our approach, this strategy focuses also on single-hop wireless environment and on data dissemination, and thus the cache management including cache admission control and replacement is not well explored. Sailhan and Issarny [32] proposed a collaborative cache management strategy among mobile terminals interacting via an ad hoc network. The issue that the authors addressed was on setting an ad hoc network of mobile terminals that cooperate to exchange Web pages. The proposed solution aims at improving the Web latency on mobile terminals while optimizing associated energy consumption. It is implemented on top of Zone Routing Protocol. The authors proposed a fixed broadcast range based on the underlying routing protocol.

The Zone Cooperative (ZC) [5], the Cluster Cooperative (CC) [4] and the ECOR [33] protocols attempt to form clusters of nodes based either in geographical proximity or utilizing widely known node clustering algorithms for MANETs [3]. In ZC, mobile nodes belonging to the neighborhood (zone) of a given node form a co-operative cache system for this node since the cost for communication with them is low both in terms of energy consumption and message exchanges. Each node has a cache to store the frequently accessed data items. The data items in the cache satisfy not only the node's own requests, but also the data requests passing through it from other nodes. For a data miss in local cache, the node first searches the data in its zone before forwarding the request to the next node that lies on a path towards the data center. As a part of cache management, a value-based replacement policy based on popularity, distance, size and time-to-live was developed to improve the data accessibility and reduce the local cache miss ratio. Simulations experiments revealed improvements in cache hit ratio and average query latency in comparison with other caching strategies. In CC, the authors present a scheme for caching in MANETs. The goal of CC is to reduce the cache discovery overhead and provide better cooperative caching performance. The authors partitions the whole MANET into equal size clusters based on the geographical network proximity. In each cluster, CC dynamically chooses a"super" node as cache state node, to maintain the cluster cache state information of different nodes within its cluster domain. The cache state node is defined as the first node that enters the cluster. The cluster cache state for a node is the list of cached data items along with their time-to-live field. The cache replacement policy is similar to that in ZC. However, the ZC protocol is problematic in terms of communication overhead and energy consumption, because every node that lies on the path towards the data center has to broadcast the packet to nodes in its zone in order to discover if there is a cached copy that satisfies the requested data item. In contrast with the ZC protocol, the CC protocol reveal a smaller overhead in terms of messages exchange between nodes, because the intermediate node (the nodes between a requesting node and the node which holds the requested data) broadcast the packet only to cluster state node. In ECOR, each mobile node forms a cooperation zone (CZ) with mobile nodes in proximity by exchanging messages to share their cached data items in order to minimize bandwidth and energy cost for each data retrieval. When a data request arrives, the node first searches the data in its CZ before forwarding the request to the data center. According to ECOR each node broadcast every modification of the cached data items to nodes that belong to cooperation zone. Each node maintains a cache hint table for the cache information of all nodes in its proximity. However, in ECOR appears a great number of exchanged messages and energy consumption in case of large node density and big cooperation radius.

The only protocols that deviated from such approaches and tried to exploit both data and node locality in an homogeneous manner are described in [38] and are the following: CachePath, CacheData, and HybridCache. In CacheData, intermediate nodes may cache data to serve future requests instead of fetching data from the "Data Center". An intermediate node caches passing by data item locally when it finds that data item is popular and does not cache the data item if all requests for it are from the same node. This rule is designed in order to reduce the cache space requirement because the mobile nodes have limited cache spaces. In CachePath, a mobile node may cache the information of a path to a nearby data requester while forwarding the data and use the path information to redirect future requests to the nearby caching site.

By caching the data path for each data item, bandwidth and the query delay can be reduced since the requested data can be obtained through fewer number of hops. The authors proposed also some optimizations techniques. An intermediate node can save only the destination node information because the path from the current router to the destination node can obtained by the underlying routing protocol. Also, the intermediate node need to record the data path when it is closer to the caching node than the data center. One major drawback of CachePath is that the cached path may not be reliable and using it may increase the overhead. A cached path may not be reliable because either the data item has become obsolete or the caching node can not be reached. The hybrid protocol HybridCache combines CacheData and CachePath while avoiding their weaknesses. In HybridCache, when a mobile node forwards a data item, it caches the data or the path based on some criteria. These criteria include the data item size, the time-to-leave of the data item and the number of hops that a cached path can save and denoted as H_{save}. H_{save} value is the difference between the distance to the data center and the distance to the caching node. H_{save} must be greater than a system tuning threshold. One drawback with these methods is that caching information of a node cannot be shared if the node does not lie on the path between the data requester and the data source. Moreover, the threshold values used in these heuristics must be set carefully in order to achieve good performance. The only works on caching in WSNs concern the placement of caches [28, 36] and thus they are not strictly relevant to our considered problem.

2.1 Motivation and contributions

The protocols proposed so far for cooperative caching in MANETs present various limitations. Those protocols, which first perform a clustering of the network and then exploit this clustering (the cluster-heads, (CH)), in order to coordinate the caching decisions, inherit the shortcomings of any bad CH selection. For instance, in [4, 5], the nodes which form the cluster are assumed to reside within the same communication range, i.e., they are with on-hop distance from the other nodes of the cluster. Additionally, the nodes do not cache the data originating from an one-hop neighbor. Thus, CHs which do no reside in a significant part of data routes can not serve efficiently their cluster members, because they do not have fast access (short latency) to requested data. The cooperation zone which is formed in [33] by selecting an optimal radius, implies a large communication overhead, because every node within that radius must send/receive any changes to the caches of the other nodes within the radius. Finally, the HybridCache policy is tightly coupled to the underlying routing protocol, and thus if a node does not reside in the route selected by the routing protocol can not cache the data/path, or conversely, can not serve the request even if it holds the requested data.

Motivated by the weaknesses of the current cooperative protocols and the unique requirements of the WMSNs, which are mainly static and not mobile, we propose a cooperative caching policy which is based on the idea of exploiting the sensor network topology, so as to discover which nodes are more important than the others, in terms of their position in the network and/or in terms of residual energy. Incorporating both factors into the design of the caching policy we ensure both network longevity and short latency in multimedia data retrieval. In summary, the article's contribution are the following:

- Definition of a metric for estimating the importance of a sensor node in the network topology, which will imply short latency in retrieval.
- Description of a cooperative caching protocol which takes into account the residual energy of the sensor nodes.
- Development of algorithms for discovering the requested multimedia data, and maintaining the caches (cache replacement policy).
- Performance evaluation of the protocol and comparison with the state-of-the-art cooperative caching protocol for MANETs, using an established simulation package (J-Sim).

3 The *NICoCa* cooperative caching protocol for WMSNs

One of the main parts of the proposed protocol is the estimation of the importance of sensors relative to the network topology. The intuition is that if we discover those nodes, which reside in a significant part of the (short) paths connecting other nodes, then these are the "important" nodes; then they may be selected as coordinators for the caching decisions, i.e., as "mediators" to provide information about accessing the requested data or even as caching points.

3.1 Measuring sensor node importance

A WMSN is abstracted as a graph $G(V, E)$, where V is the set of its nodes, and E is the set of radio connections between the nodes. An edge $e = (u, v),\ u, v \in E$ exists if and only if u is in the transmission range of

v and vice versa. All links in the graph are bidirectional, i.e., if u is in the transmission range of v, v is also in the transmission range of u. The network is assumed to be in a connected state. The set of neighbors of a node v is represented by $N_1(v)$, i.e., $N_1(v) = \{u : (v, u) \in E\}$. The set of two-hop nodes of node v, i.e., the nodes which are the neighbors of node v's neighbors except for the nodes that are the neighbors of node v, is represented by $N_2(v)$, i.e., $N_2(v) = \{w : (u, w) \in E$, where $w \neq v$ and $w \notin N_1$ and $(v, u) \in E\}$. The combined set of one-hop and two-hop neighbors of v is denoted as $N_{12}(v)$.

Definition 1 (Local network view of node v) The local network view, denoted as LN_v, of a graph $G(V, E)$ w.r.t. a node $v \in V$ is the *induced subgraph* of G associated with the set of vertices in $N_{12}(v)$.

A *path* from $u \in V$ to $w \in V$ has the common meaning of an alternating sequence of vertices and edges, beginning with u and ending with w. The *length* of a path is the number of intervening edges. We denote by $d_G(u, w)$ the *distance* between u and w, i.e., the minimum length of any path connecting u and w in G, where by definition $d_G(v, v) = 0, \ \forall v \in V$ and $d_G(u, w) = d_G(w, u), \ \forall u, w \in V$. Note that the distance is not related to network link costs (e.g., latency), but it is a purely abstract metric measuring the number of hops.

Let $\sigma_{uw} = \sigma_{wu}$ denote the number of shortest paths from $u \in V$ to $w \in V$ (by definition, $\sigma_{uu} = 0$). Let $\sigma_{uw}(v)$ denote the number of shortest paths from u to w that some vertex $v \in V$ lies on. Then, we define the *node importance* index $\mathcal{NI}(v)$ of a vertex v as:

Definition 2 The $\mathcal{NI}(v)$ of a vertex v is equal to:

$$\mathcal{NI}(v) = \sum_{u \neq v \neq w \in V} \frac{\sigma_{uw}(v)}{\sigma_{uw}}. \quad (1)$$

Large values for the $\mathcal{NI}$ index of a node v indicate that this node v can reach others on relatively short paths, or that the node v lies on considerable fractions of shortest paths connecting others. Illustration of this metric is presented in Fig. 1 (with well formed and vague node clusters).

Apparently, when estimating the $\mathcal{NI}$ index for each sensor node using the whole network topology we obtain a very informative picture of which nodes reside in a large number of shortest paths between other nodes. Though, it would not be practical to use this global information in the desing of a cooperative caching protocol, because it would require the exchange of a huge number of messages between the nodes to learn this information. Moreover, it would require a lot of communication rounds; for the establishment of broadcast tree with a leader/root node, for learning the neighborhoods for the estimation and propagation of the NI indexes and so on. These rounds would also increase the latency and compromise the protocol's robustness when changes in the topology would take place. Therefore, if the idea of NI index could be successfully applied in neighborhoods to identify nodes which reside in a large number of shortest paths, then we could desing a localized algorithm based on it. Indeed, using only the information contained in the local network view of a node, we can still obtain this information, quite accurately. For instance, computing the NI indexes for all the nodes that are contained in LN_8, we obtain the information of Table 1.

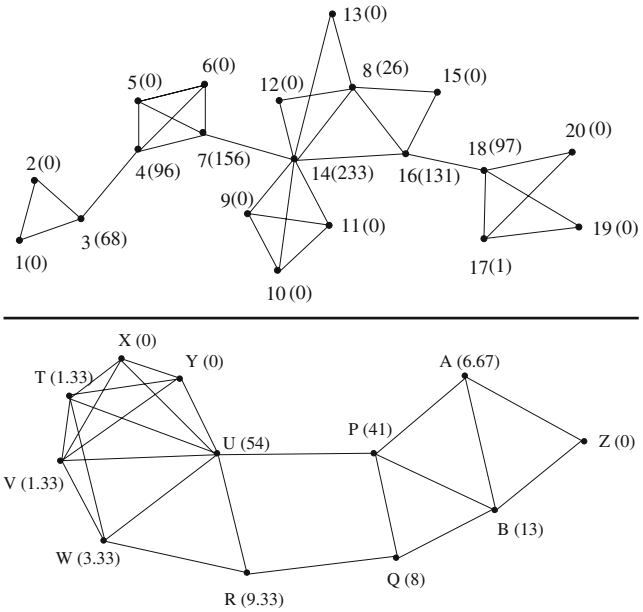

Figure 1 Calculation of $\mathcal{NI}$ for two sample graphs. The *numbers in parentheses* denote the $\mathcal{NI}$ index of the respective node considering the whole WMSN topology

We observe that although the absolute values of the NI indexes are reduced as a consequence of the smaller network size, the relative ranking of the 1-hop neighbors of the node 8 w.r.t. their NI index is the

Table 1 $\mathcal{NI}$ index of the nodes belonging to LN_8 (from the local perspective of node 8)

n(ode)	$\mathcal{NI}(n)$	*n(ode)*	$\mathcal{NI}(n)$	*n(ode)*	$\mathcal{NI}(n)$
7	0	11	0	15	0
8	14	12	0	16	23
9	0	13	0	18	0
10	0	14	65		

same with that obtained when we considered the whole network. Apparently, the larger the neighborhood we consider the more accurate the picture of nodes' importance we get. Although, we can not come up with a closed form that would describe when and at what degree the relative rankings change when we consider small neighborhoods, instead of the whole network, because it depends on the graph characteristics, our investigation showed that in more than 98% of the cases, the relative rankings remain unchanged.

At a first glance, the computation of the $\mathcal{NI}$ seems expensive, i.e., $O(y * x^2)$ operations in total for a 2-hop neighborhood, which consists of x nodes and y links. For our undirected network, we can use breadth-first traversal to count the number and find the length of shortest paths from any particular node (source) to any other node (target) in time linear w.r.t. the number of edges, i.e.,$O(x * y)$. Repeating this procedure for any possible source, we conclude that computing the length and number of shortest paths between any possible pair can be done in time $O(y * x^2)$. Fortunately, we can do better than this and exploit some redundancy in the computations. This procedure gives an $O(x * y)$ method for computing the NI indexes for the nodes. For more information concerning the calculation and the use of this metric in broadcasting protocols the interested user can consult the work reported in [21].

3.2 Housekeeping information in the *NICoCa* protocol

W.l.o.g. and adopting the model presented in [38], we assume that the ultimate source of multimedia data is a *Data Center*. This is not restrictive at all and simply guarantees that every request, if it is not served by other sensor nodes and if does not expire, will finally be served by the Data Center.

Firstly, it is assumed that each node is aware of its 2-hop neighborhood. This information is obtained through periodic exchange of "beacon" messages. We assume that we are able to determine an assignment of time slots to the sensor nodes such that no interference occurs, i.e., no two nodes transmit in the same time slot. With this assignment, the nodes can transmit beacon messages to discover their neighbors and also to acquire the channel for transmitting their data requests. This assignment is determined using the D2-coloring algorithm [13]. Although more traditional methods can be used for channel arbitration, this approach contributes in reducing the latency by avoidind interference. Then, every node calculates the $\mathcal{NI}$ index of its 1-hop neighbors. The node uses this information in order to characterize some of its neighbors as *mediator* nodes; the minimum set of neighbors with the larger $\mathcal{NI}$ which "cover" its 2-hop neighborhood are the mediator nodes for that node; The node is responsible for notifying its neighbors about which of them are its mediators. Thus, a node can be either a mediator or an ordinary node. Notice here that there is no need for the node to know the "importance" of its neighbors w.r.t. the whole network; instead the node needs to know their importance in its neighborhood. Of course, if the notion of neighborhood is extended to cover the whole network (i.e., each node is aware of its k-neighborhood, where k equals the network diameter), then it is obvious that the node would have perfect knowledge of the exact NI indexes of all network's nodes.

The sending of requests for data is carried out by an ordinary sensor (or ad hoc) routing protocol, e.g., ad-hoc on-demand distance vector (AODV) [27]. A node always caches a datum which has requested for. A node is aware of its remaining energy and of the free space in its cache. Each sensor node stores the following metadata related to a cached multimedia item:

- the dataID, and the actual multimedia data item,
- the data size (s_i),
- a Time-To-Live (TTL) interval,
- for each cached item, the timestamps of the K most recent accesses to that item. Usually, $K = 2$ or 3.
- each cached item is characterized either as O (i.e., own) or H (i.e., hosted). If an H-item is requested by the caching node, then its state switches to O.

When a node acquires the multimedia datum he has requested for, then it caches it and broadcasts a small index packet containing the dataID and the associated TTL, its remaining energy and its free cache space. The mediator nodes which are also 1-hop neighbors of this node store this broadcasted information. Notice here that this set of mediator nodes includes the mediators that the broadcasting node has selected, and also any other mediators which have been selected by nearby nodes. In summary, every mediator node stores the remaining energy and the free cache space for each one of its 1-hop neighbors, and for each dataID that has heard through the broadcasting operation, the TTL of this datum and the nodes that have cached this datum.

3.3 The cache discovery component protocol

When a sensor node issues a request for a multimedia item, it searches its local cache. If the item is found there (a local cache hit) then the K most recent access timestamps are updated. Otherwise (a local cache miss), the request is broadcasted and received by the

mediators. If none of them responds (a "proximity" cache miss), then the request is directed to the Data Center.

When a non one-hop mediator node receives a request, it searches its local cache. If it deduces that the request can be satisfied by a neighboring node (a remote cache hit), then stops the request's route toward the Data Center, and forwards the request to this neighboring node. If more than one nodes can satisfy the request, then the node with the largest residual energy is selected. If the request can not be satisfied by this mediator node, then it does not forward it recursively to its own mediators. This is due to the fact that these mediators will most probably be selected by the routing protocol as well (AODV) and thus a great deal of savings in messages is achieved. Therefore, during the procedure of forwarding a request toward the Data Center, no searching to other nodes is performed apart from the nodes which reside on the path toward the Data Center.

For example, suppose that sensor node SN_i in Fig. 2 issues a request for the data item x that is placed in data center DC_1 and which has been cached by sensor nodes SN_g and SN_h. The black shaded nodes are the mediators nodes of the sensor network. Based on the information presented in Fig. 1, we can easily understand that node SN_i has selected as its mediators the nodes SN_a and SN_b. With their turn, these sensors would select as their mediators the nodes SN_c and SN_d (for node SN_a) and SN_d (for node SN_b).

In the beginning sensor node SN_i searches its own cache. If it deduces that data item is not available in local cache, it sends a proximity search request in neighboring mediator nodes SN_a and SN_b. Upon receiving the search request, each mediator searches in the proximity cache table. If data item found, each mediator replies with an index packet that contains the dataID and the remaining energy of the sensor node that has the uppermost battery power and has cached the data item. SN_i, upon the receipt of index packets, selects the sensor node that has the smallest energy consumption and sends the request packet. Caching node responds with a reply packet containing the requested data item.

If no neighboring sensor node is caching the data item, SN_i sends a request packet to the data center DC_1, as shown in Fig. 2. When SN_x ($x \in \{d, e\}$) receives a request packet, it searches in local cache and in proximity cache table. If the data item is not found, SN_x forwards the request through the path to the DC_1. When sensor node SN_e found that the requested data item has been cached by some neighboring nodes, it chooses the node that has the smallest energy dissipation and redirects the request packet to the caching node. The caching node sends a reply packet containing the data item x along the routing path until it reaches the original requester. Once the requester node receives the data item, it notifies its one-hop mediators about the new caching item by sending an index packet containing item's dataID. In case of not enough cache capacity, it triggers the cache replacement protocol to determine the data items that should be evicted from the cache.

For every issued request one of the following four cases may take place:

1) Local hit: the requested datum is cached by the node which issued the request. If this datum is valid (the TTL has not expired) then the *NICoCa* is not executed.
2) "Proximity" hit: the requested datum is cached by a node in the 2-hop neighborhood of the node which issued the request. In this case, the mediator(s) return to the requesting node the "location" of the node which stores the datum.
3) Remote hit: the requested datum is cached by a node and this node has at least one mediator residing along the path from the requesting node to the Data Center.
4) Global hit: the requested datum is acquired from the Data Center.

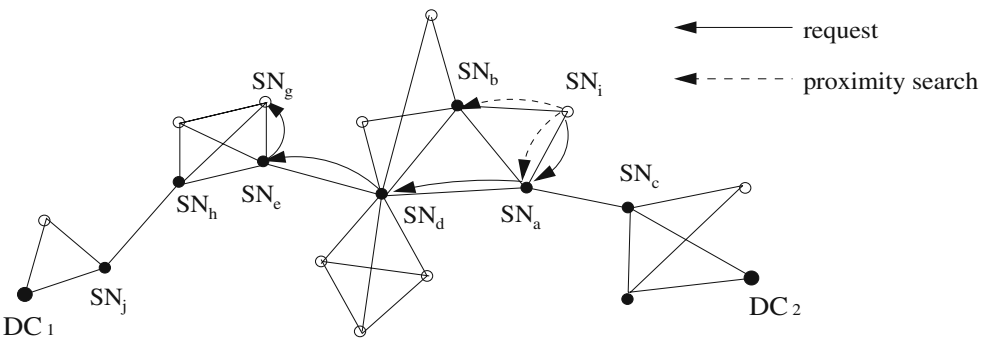

Figure 2 A request packet from sensor node SN_i is forwarded to the caching node SN_g

Table 2 Simulation parameters

Parameter	Default value	Range
# items (N)	1000	
S_{min} (KB)	1	
S_{max} (KB)	10	
S_{min} (MB)	1	
S_{max} (MB)	5	
Zipfian θ	0.8	
# nodes (n)	500	100–1000
Bandwidth (Mbps)	2	
Waiting interval (t_w)	10 s for items with KB size	
	100 s for items with KB size	
Client cache size (KB)	800	200 to 1200
Client cache size (MB)	125	25 to 250
Zipfian skewness (θ)	0.8	0.0 to 1.0

3.4 The cache replacement component protocol

Even though the cache capacity of individual sensors may be in the order of gigabytes (e.g., NAND flash) the development of an effective and intelligent replacement policy is mandatory to cope with the overwhelming size of multimedia data generated in WMSNs. The *NICoCa* protocol employs the following four-step policy:

STEP 1. In case of necessity, before purging from cache any other data, each sensor node first purges the data that it has cached on behalf of some other node. Each cached item is characterized either as O (i.e., own) or H (i.e., hosted). In case of a local hit, then its state switches to O. If the available cache space is still smaller than the required, execute Step 2.

STEP 2. Calculate the following function for each cached datum i: $cost(i) = \frac{s_i}{TTL_i} * \frac{now - t_{K-th\ access}}{K}$. The candidate cache victim is the item which incurs the largest cost.

STEP 3. Inform the mediators about the candidate victim. If it is cached by some mediator, then this information returns back to the node and purges the datum. If the datum is not cached by some mediator(s), then it is forwarded to the node with the largest residual energy and the datum is purged from the cache of the original node. In any case, the mediators update their cached metadata about the new state.

STEP 4. The node which caches this purged datum, informs the mediators with the usual broadcasting procedure.

The pseudocode for the complete algorithm *NICoCa* is presented in the Appendix.

4 Performance evaluation

We evaluated the performance of the *NICoCa* protocol through simulation experiments. We conducted a large number of experiments with various parameters, and compared the performance of *NICoCa* to a state-of-the-art cooperative caching policy for MANETs, namely *HybridCache* [38].

4.1 Simulation model

We have developed a simulation model based on the J-Sim simulator [34]. In our simulations, the AODV [27] routing protocol is deployed to route the data

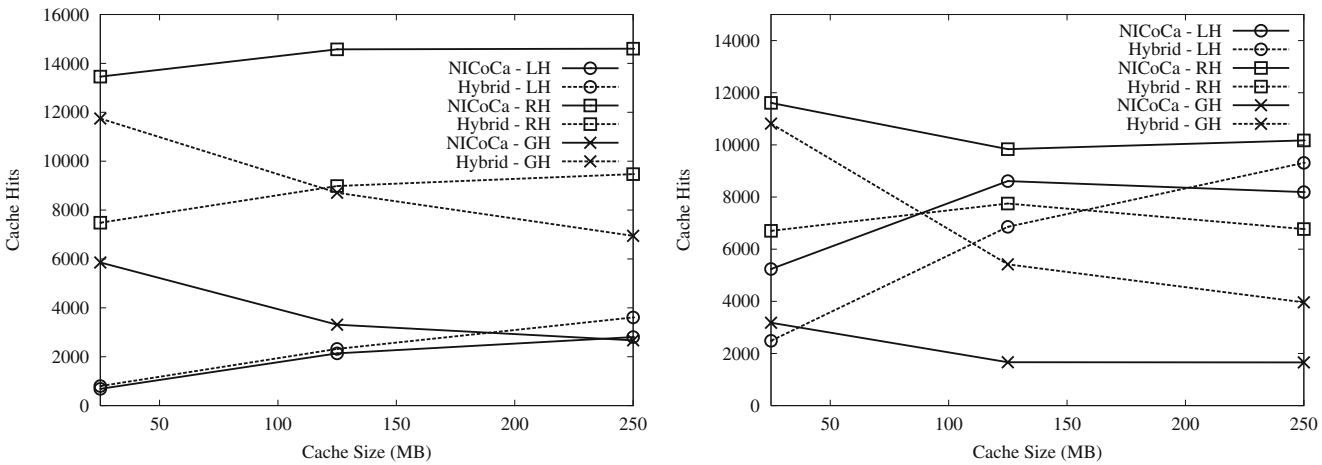

Figure 3 Impact of sensor cache size on hits (MB-sized files, $\theta = 0.0$ and $\theta = 0.8$) in a sparse WMSN ($d = 7$) with 100 sensors

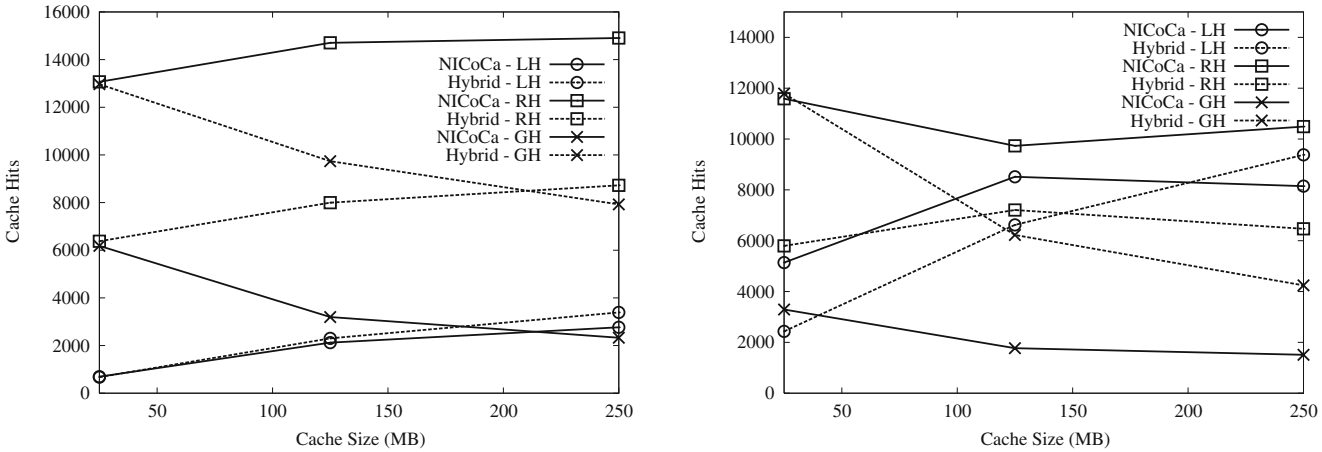

Figure 4 Impact of sensor cache size on hits (MB-sized files, $\theta = 0.0$ and $\theta = 0.8$) in a dense WMSN ($d = 10$) with 100 sensors

traffic in the WSN. We use IEEE 802.11 as the MAC protocol and the free space model as the radio propagation model. The wireless bandwidth is 2 Mbps.

We tested the protocols for a variety of sensor network topologies, to simulate sensor networks with varying levels of node degree, from 4 to 10. We also conducted experiments by choosing the number of nodes between 100 and 1000. In addition, in our experiments we evaluate the protocol efficiency under two different set of data item sizes. Each data item has size that is uniformly distributed from 1KB to 10KB for the first set, and from 1MB to 5MB for the second.

The network topology consists of many square grid units where one or more nodes are placed. The number of square grid units depends on the number of nodes and the node degree. The topologies are generated as follows: the location of each of the n sensor nodes is uniformly distributed between the point ($x = 0$, $y = 0$) and the point ($x = 500$, $y = 500$). The average degree d is computed by sorting all $n * (n - 1)/2$ edges in the network by their length, in increasing order. The grid unit size corresponding to the value of d is equal to $\sqrt{2}$ times the length of the edge at position $n * d/2$ in the sorted sequence. Two sensor nodes are neighbors if they placed in the same grid or in adjacent grids. The simulation area is assumed of size $500\,m \times 500\,m$ and is divided into equal sized square grid units. Beginning with the lower grid unit, the units are named as $1, 2, \ldots,$ in a column-wise fashion.

The client query model is similar to what have been used in previous studies [38]. Each sensor node generates read-only queries. After a query is sent out, if the sensor node does not receive the data item, it waits for an interval (tw) before sending a new query. The access pattern of sensor nodes is: a) location independent, that is, sensor nodes decide independently the data of

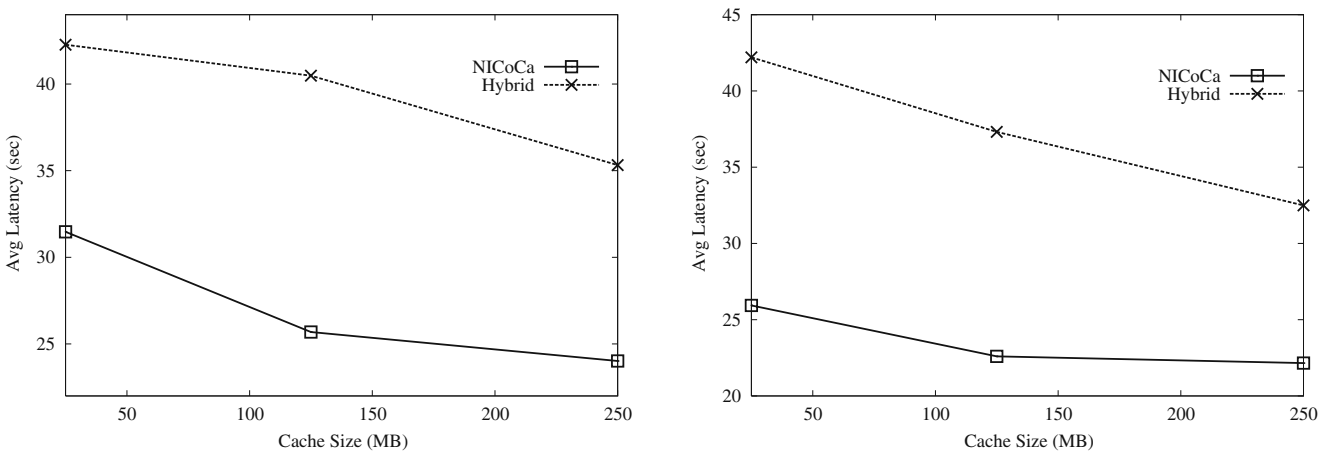

Figure 5 Impact of sensor cache size on latency (MB-sized files, $\theta = 0.0$ and $\theta = 0.8$) in a sparse WMSN ($d = 7$) with 100 sensors

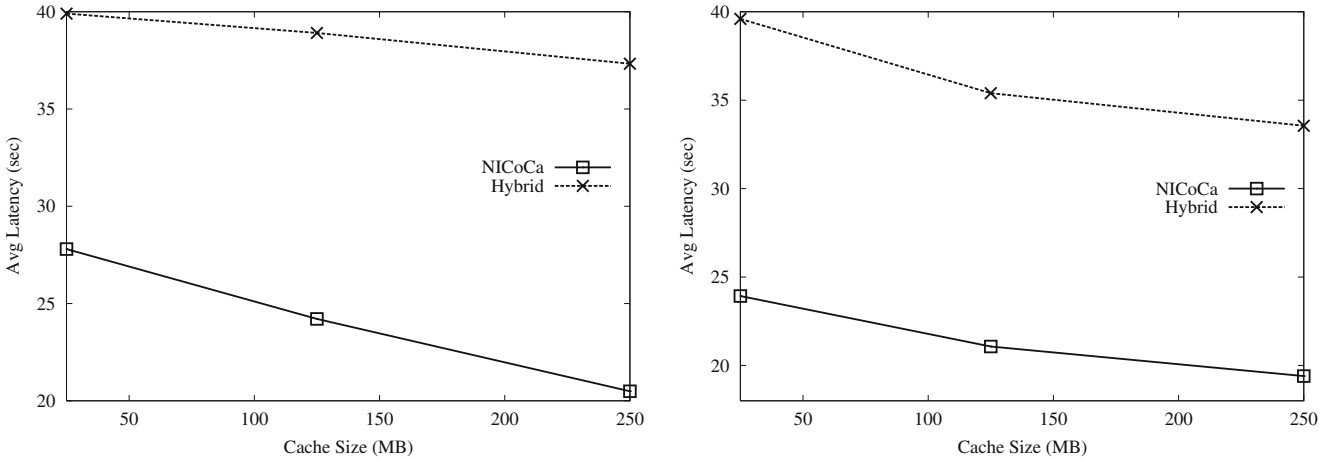

Figure 6 Impact of sensor cache size on latency (MB-sized files, $\theta = 0.0$ and $\theta = 0.8$) in a dense WMSN ($d = 10$) with 100 sensors

interest; each sensor node generates accesses to the data following the uniform distribution, and b) Zipfian with $\theta = 0.8$, where groups of nodes residing in neighboring grids (25 grids with size $100\,m \times 100\,m$) have the same access pattern. We tested the protocols both for zipfian access pattern and for uniform access pattern. In case of zipfian access pattern we conducted experiments with varying θ values between 0.0 and 1.0.

Similar to [38], two data centers are placed at opposite corners of the simulation area. Data Center 1 is placed at point $(x = 0, y = 0)$ and Data Center 2 is place at point $(x = 500, y = 500)$. There are $N/2$ data items in each data center. Data items with even ids are stored at Data Center 1 and data items with odd ids are stored at Data Center 2. The size of each data item is uniformly distributed between smin and smax. We assumed that data items are not updated. The data centers serve the queries on a first-come-first-served basis. The system parameters are listed in Table 2.

4.2 Performance metrics

The measured quantities include the number of hits (local, remote and global), the average latency for getting the requested data and the message overhead. It is evident that a small number of global hits implies less network congestion, and thus fewer collisions and packet drops. Moreover, large number of remote hits proves the effectiveness of cooperation in reducing the number of global hits. A large number of local hits does not imply an effective cooperative caching policy, unless it is accompanied by small number of global hits, since the cost of global hits vanishes the benefits of local hits.

4.3 Evaluation

We performed a large number of experiments varying the size of the sensornet (in terms of the number of its sensor nodes), varying the access profile of the sensor nodes, and the cache size relative to the aggregate size of all data items. In particular, we performed experiments for 100, 500, and 1000 sensors, for cache size equal to 1%, to 5% to 10% of the aggregated size of all distinct multimedia data, for access pattern with θ equal to 0.0 (uniform access pattern) to 1.0 (highly skewed access pattern), for average sensor node degree equal to 4, 7 (very sparse and spare sensornet) and 10 (dense sensornet), and for data item size equal to a few kilobytes (KB) and also equal to a few megabytes (MB). For each different setting we measured the number of hits (local, remote, global), the latency[2] and the message overhead. In the sequel of the article we will present only a representative set of the results, since there are many of independent parameters and three dependent performance metrics. We partition the graphs in two large groups w.r.t. whether they deal with small KB-sized files or large MB-sized multimedia files.

4.3.1 Experiments with MB-sized data items

The purpose of this set of experiments is to examine the performance of the caching algorithms when they have to deal with large multimedia files, e.g., video files, queried by the sensornet.

[2]The latency is measured in seconds, which does not corresponds to the usual time metric, but to internal simulator clock.

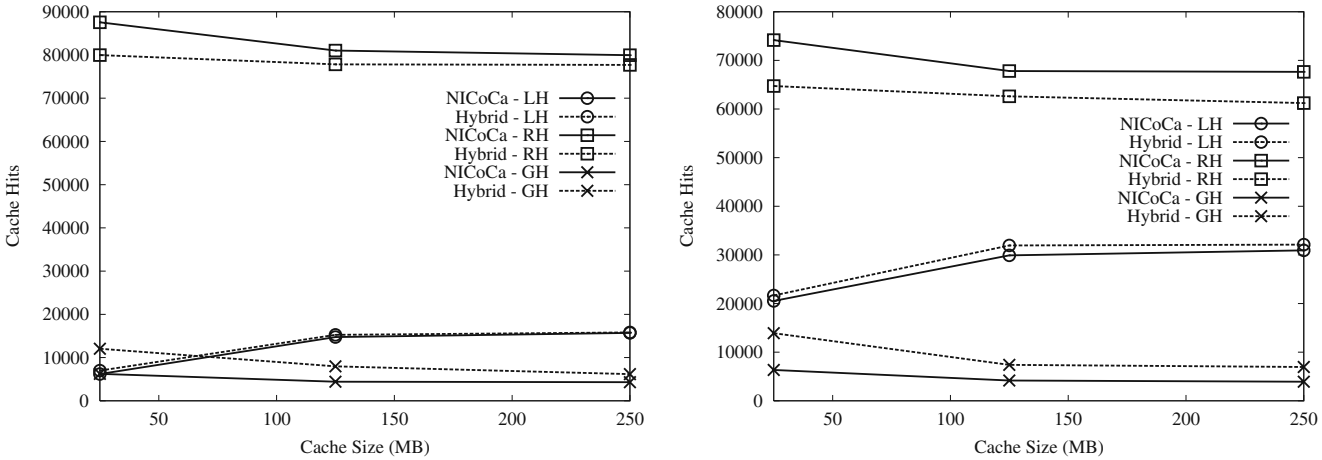

Figure 7 Impact of sensor cache size on hits (MB-sized files, $\theta = 0.0$ and $\theta = 0.8$) in a sparse WMSN ($d = 7$) with 500 sensors

Figures 3 and 4 show the number of hits achieved by the two protocols for a small (sparse and dense, respectively) sensornet for both uniform and skewed access pattern. The first observation is that, as expected, the number of local hits increases for both protocols as the access pattern becomes more skewed. The interesting point is that, although for uniform access patterns *HyrbidCache* is slightly better than *NICoCa* w.r.t. the local hits, the situation is reversed when the requests are concentrated to a smaller number of files, which can be attributed to the more efficient replacement and admission policy of the *NICoCa*. With respect to the number of global hits, *NICoCa* achieves half that of hybrid and the performance gap widens as we move to dense sensor deployments; actually *NICoCa* maintains almost constant the number of global hits. The reason behind this is the relative performance of the algorithms w.r.t. the remote cache hits. For sparser deployments *NICoCa* is two times better than *HyrbidCache*, and this difference becomes more evident for denser networks. Thus, it proves to be a more effective cooperation scheme, due to the fact that it strives to exploit the network topology. These relative performance results are straightforwardly reflected to the access latency incurred by the algorithms (Figs. 5 and 6).

When we move to larger sensornets with 500 (Fig. 7–8) and with 1000 nodes (Fig. 12), the superiority of the *NICoCa* caching algorithm in terms of hits is still evident, but the results are not so impressive, because we constrain more sensor nodes to be dispersed in the same geographical region, thus creating replicas of the same data. This superiority is reflected to the access latency as well (Figs. 9 and 10). We observe that as

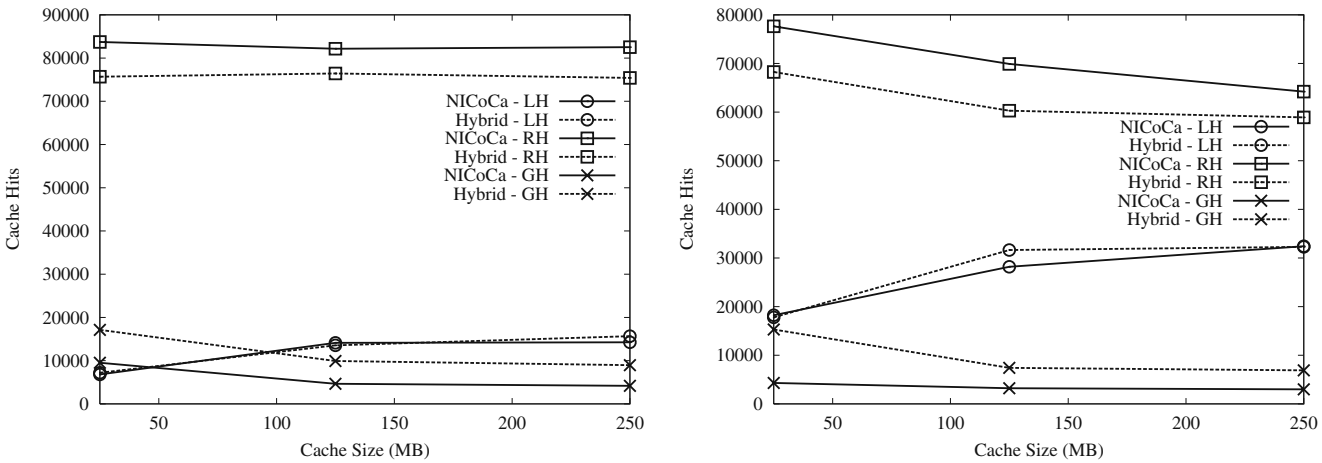

Figure 8 Impact of sensor cache size on hits (MB-sized files, $\theta = 0.0$ and $\theta = 0.8$) in a dense WMSN ($d = 10$) with 500 sensors

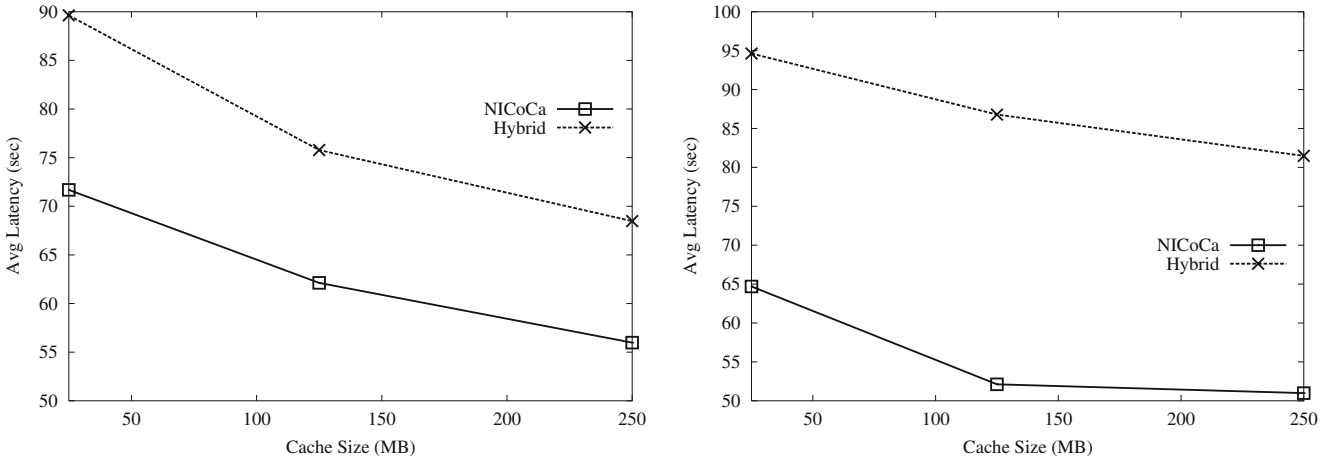

Figure 9 Impact of sensor cache size on latency (MB-sized files, $\theta = 0.0$ and $\theta = 0.8$) in a sparse WMSN ($d = 7$) with 500 sensors

we move to larger sensornets, the latence gradually increases, because the denser deployment (more nodes in the same region) has a negative effect on the efficiency of communication, aggravating the collisions and packet drops.

At this point it is interesting to note the total number of messages that are communicated between the sensor nodes, which is also the metric that models the total network energy dissipated (Figs. 11). For a dense sensornet with uniform and skewed access pattern, *NICoCa* sends at most half of the messages sent out by *HyrbidCache* and the situation becomes more favorable for *NICoCa* as the access pattern becomes more skewed, which is expected. These results are confirmed for larger sensornets with 1000 nodes (Figs. 12, 13, and 14).

In summary, for all network topologies *NICoCa* achieves more remote hits and less global hits than *HyrbidCache*. This performance gap widens in favor of *NICoCa* as we move from sparse to denser WMSN. It is striking that for very dense sensor deployments, *NICoCa* achieves double the remote hits of *Hyrbid Cache* and only half of its global hits. Examining the local hits, we observe that for sparse sensor networks *HyrbidCache* achieves slightly more local hits than *NICoCa*, but this gap vanishes completely when moving to denser network topologies. Besides, this small gain of *HyrbidCache* for sparse topologies is not advantageous at all, since it incurs global hits as many as twice the number of its local hits.

4.3.2 Experiments with KB-sized data items

A significant question arises whether these relative results still hold when the sensornet has to deal with smaller multimedia files, with size equal to a few kilobytes. Although we expect that WMSNs will deal with

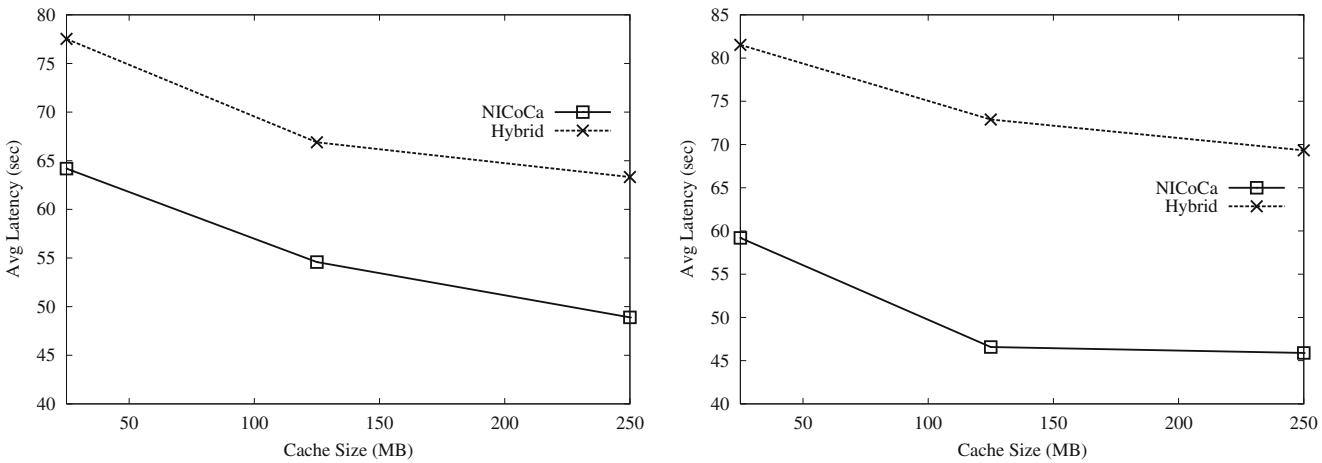

Figure 10 Impact of sensor cache size on latency (MB-sized files, $\theta = 0.0$ and $\theta = 0.8$) in a dense WMSN ($d = 10$) with 500 sensors

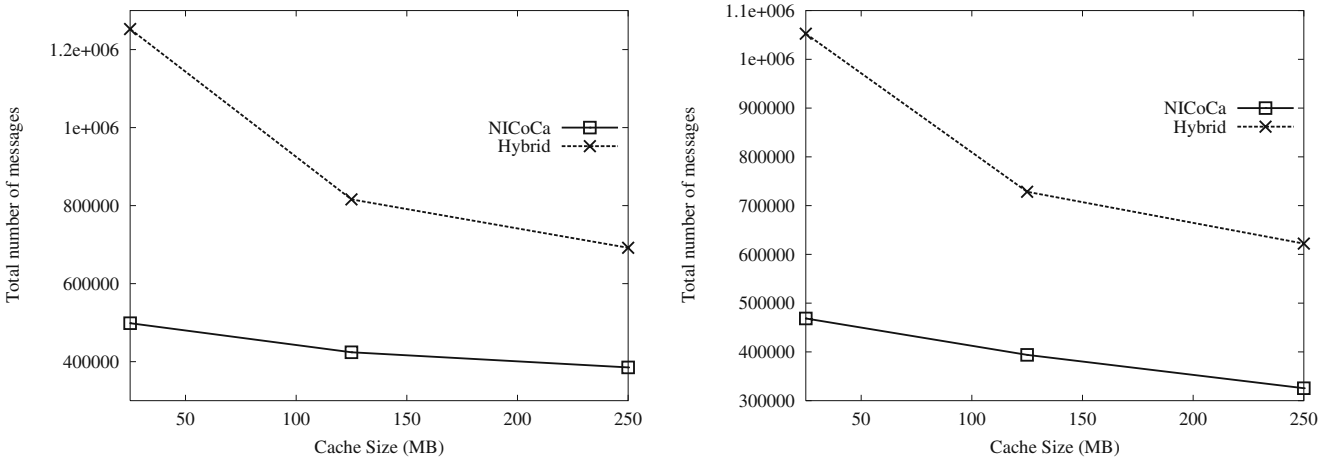

Figure 11 Impact of sensor cache size on number of messages (MB-sized files, $\theta = 0.0$ and $\theta = 0.8$) in a dense WMSN ($d = 10$) with 500 sensors

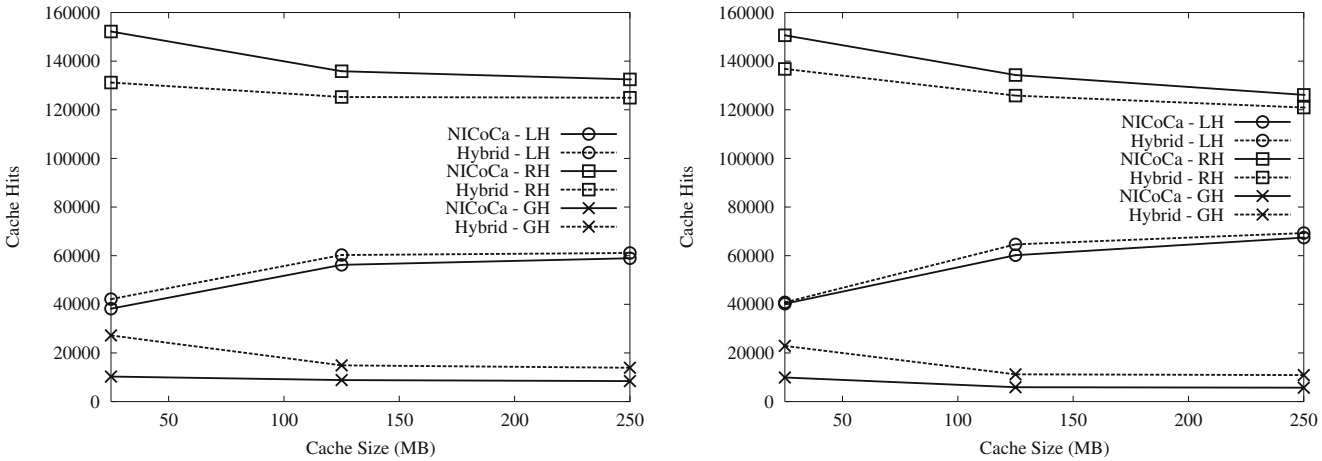

Figure 12 Impact of sensor cache size on hits (MB-sized files, $\theta = 0.8$) in a dense WMSN ($d = 10$) with 1000 sensors

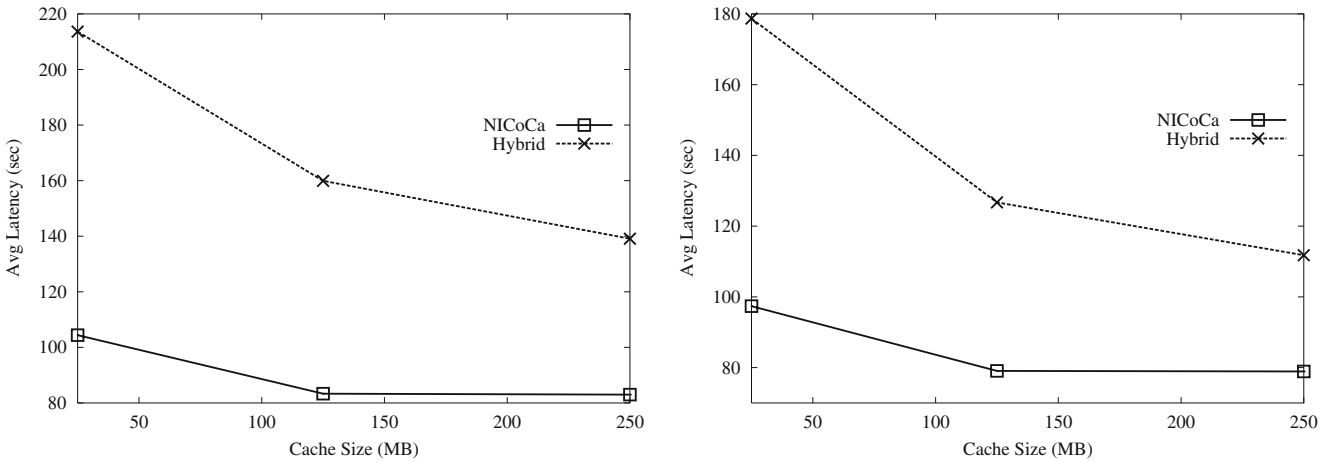

Figure 13 Impact of sensor cache size on latency (MB-sized files, $\theta = 0.8$) in a dense WMSN ($d = 10$) with 1000 sensors

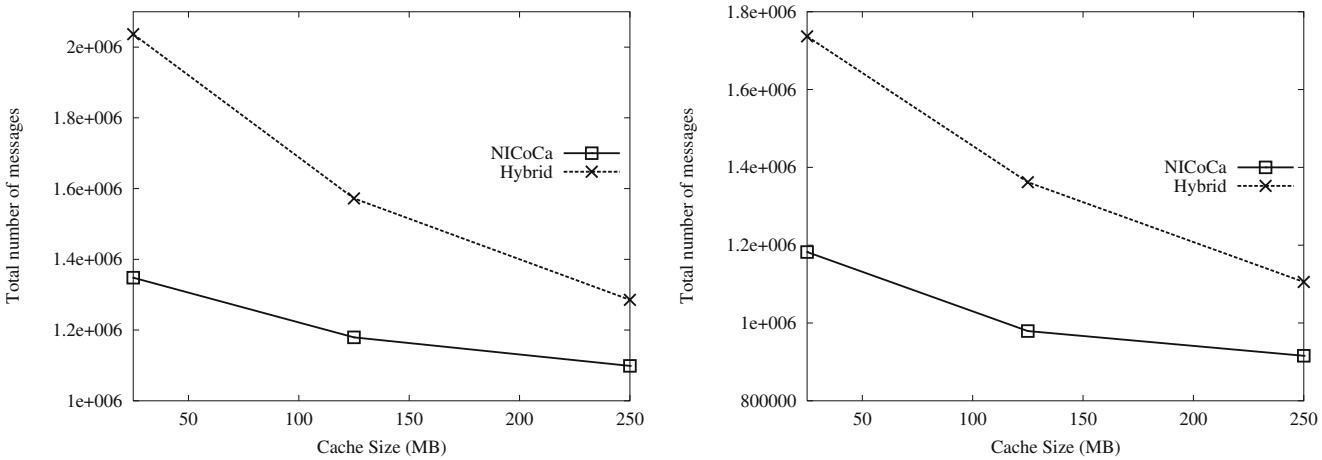

Figure 14 Impact of sensor cache size on the number of messages (MB-sized files, $\theta = 0.8$) in a dense WMSN ($d = 10$) with 1000 sensors

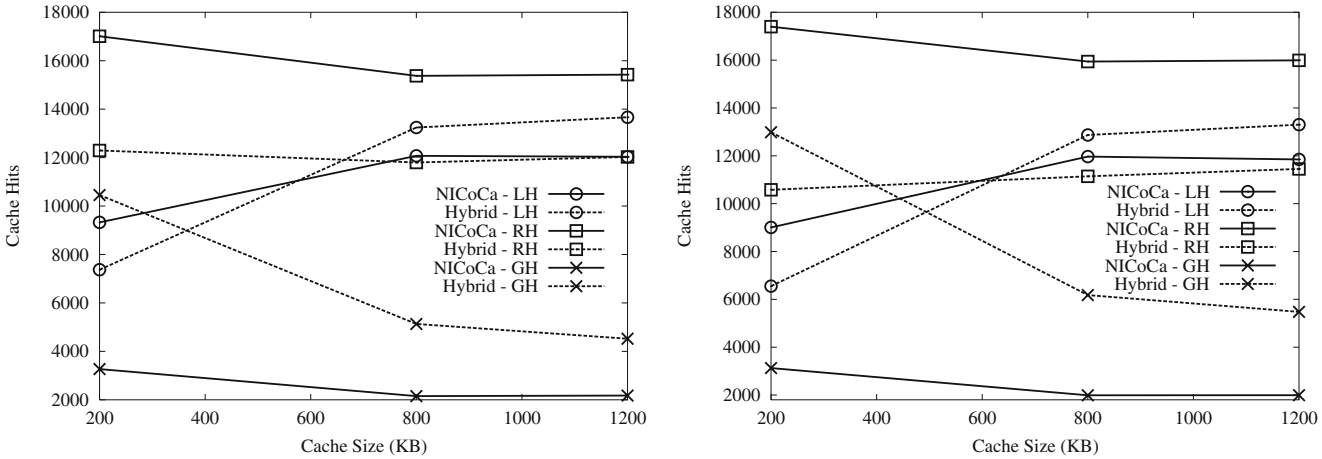

Figure 15 Impact of sensor cache size on hits (KB-sized files, $\theta = 0.8$) in a sparse and dense WMSN ($d = 7$ and $d = 10$) with 100 sensors

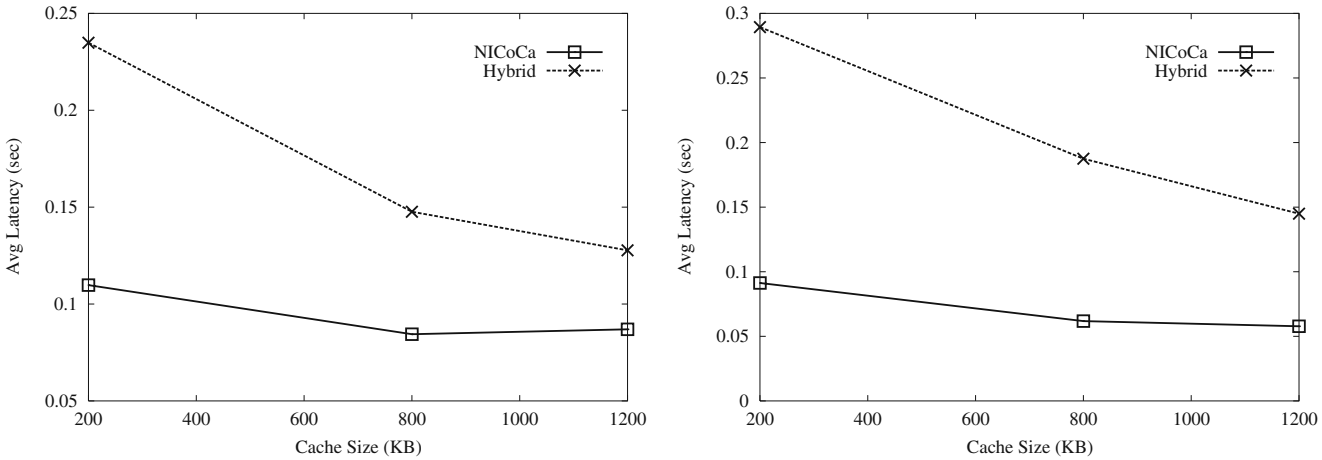

Figure 16 Impact of sensor cache size on latency (KB-sized files, $\theta = 0.8$) in a sparse and dense WMSN ($d = 7$ and $d = 10$) with 100 sensors

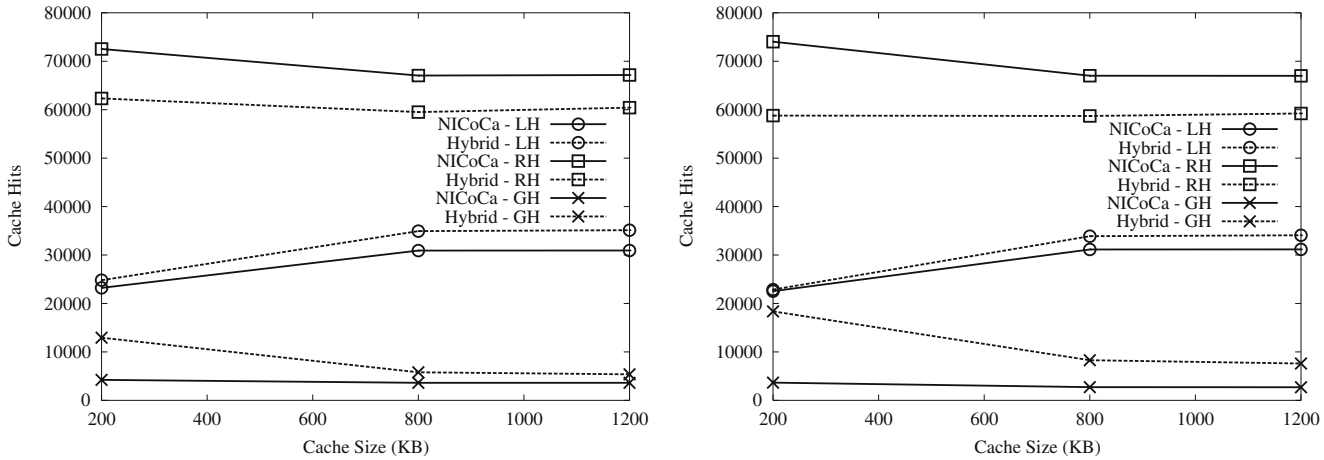

Figure 17 Impact of sensor cache size on hits (KB-sized files, $\theta = 0.8$) in a sparse and dense WMSN ($d = 7$ and $d = 10$) with 500 sensors

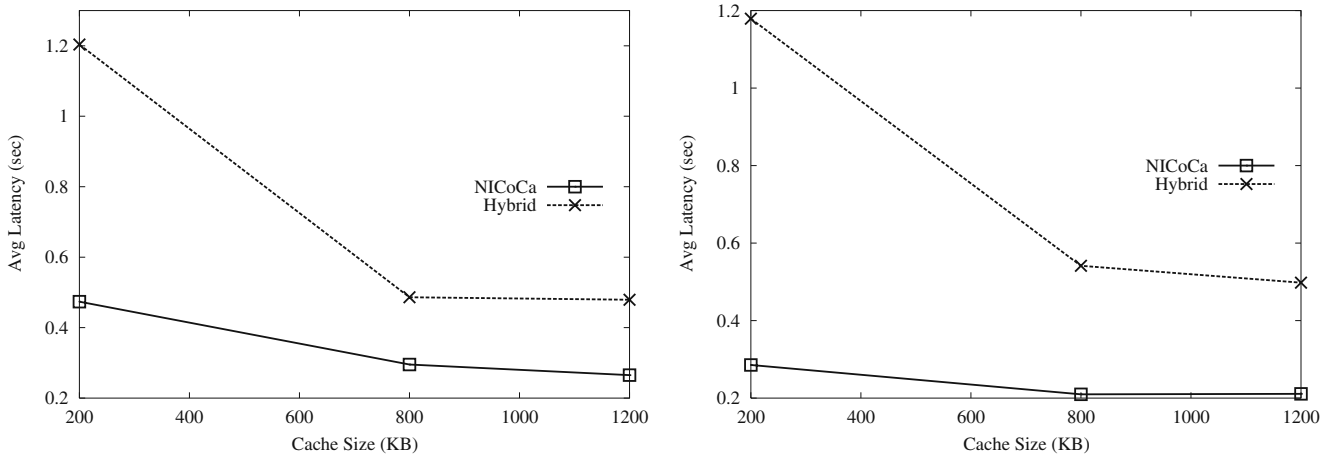

Figure 18 Impact of sensor cache size on latency (KB-sized files, $\theta = 0.8$) in a sparse and dense WMSN ($d = 7$ and $d = 10$) with 500 sensors

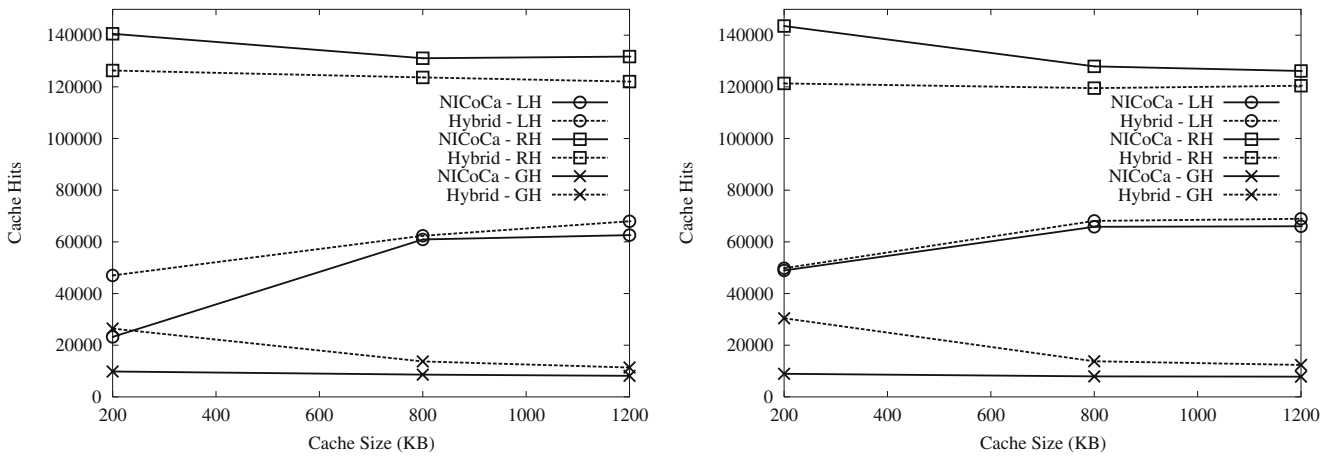

Figure 19 Impact of sensor cache size on hits (KB-sized files, $\theta = 0.8$) in a sparse and dense WMSN ($d = 7$ and $d = 10$) with 1000 sensors

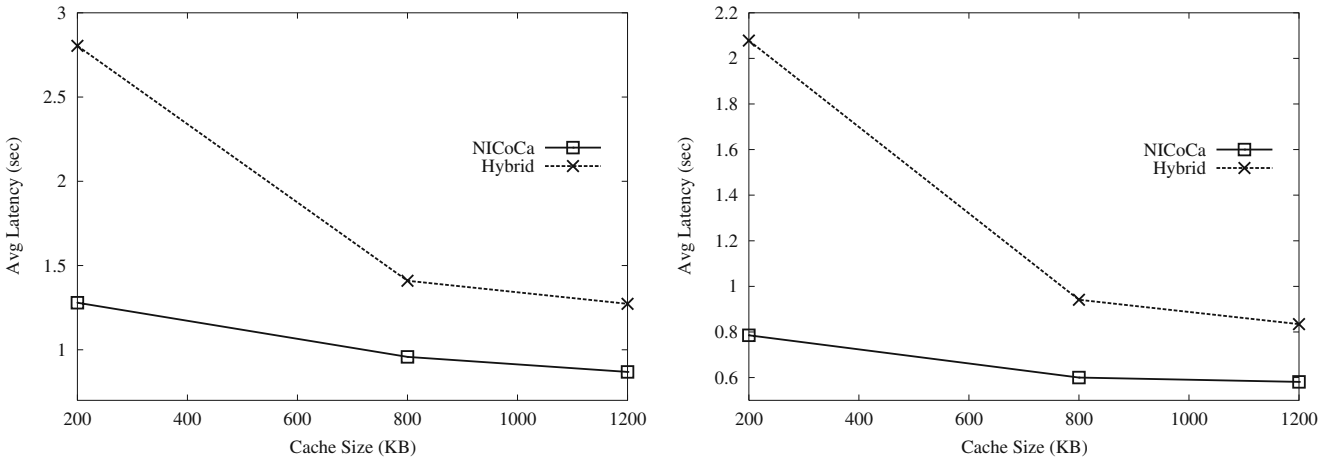

Figure 20 Impact of sensor cache size on latency (KB-sized files, $\theta = 0.8$) in a sparse and dense WMSN ($d = 7$ and $d = 10$) with 1000 sensors

MB-sized images of video files, it might be the case that the sensor nodes will exchange smaller images as well. To investigate the performance of the cooperative caching protocols for this case, we performed the same set of experiments but for KB-sized files and here we demonstrate a subset of the results obtained.

The general observations that we recorded for the case of large MB-size files, still hold for this case; *NICoCa* achieves significantly smaller number of global hits and larger number of remote hits than *Hyrbid Cache* does. It is not worthy to comment on each individual performance graph (Fig. 15, 16, 17, 18, 19, and 20), since in all cases *NICoCa* is the clear winner; it achieves again 25% more remote hits and 50% less global hits than *HyrbidCache*, which is only marginally better than *NICoCa* in terms of local hits.

5 Summary and conclusions

The recent advances in miniaturization, the creation of low-power circuits, and the development of cheap CMOS cameras and microphones, which are able to capture rich multimedia content, gave birth to what is called *WMSNs*. WMSNs are expected to fuel many new applications and boost the already existing. The unique features of WMSNs call for protocol designs that will provide application-level QoS, an issue that has largely been ignored in traditional WSNs. Taking a first step toward this goal, this article develops a cooperative caching protocols, the *NICoCa* protocol, suitable for deployment in WMSNs. The protocol "detects" which sensor nodes are most "central" in the network neighborhoods and gives to them the role of mediator in order to coordinate the caching decisions. The proposed protocol is evaluated with J-Sim and its performance is compared to that of a state-of-the-art cooperative caching protocol for MANETs. The obtained results attest the superiority of the proposed protocol which is able to reduce the global hits at an average percentage of 50% and increase the remote hits due to the effective sensor cooperation at an average percentage of 40%. The performance of the protocol is particularly high for the delivery of large multimedia data.

Appendix

The *NICoCa* cooperative caching protocol

// d_i: data item i, $i \in [1 \ldots 1000]$
// request(d_i): Request for data item i
// N_i: Node i
// FS: Free cache space
// RE: Remaining energy
// PCT: Proximity Cache Table
// ipacket: An index packet that contains d_i's id, FS and RE

(A) Cache Discovery Algorithm
if(d_i is in local cache of requester node) **then**
    send ipacket to CHs;
    return;
if(requester node is CH and d_i's id in PCT) **then**
    select caching node with largest RE;
    send request(d_i) to caching node;
else
    requester node sends request(d_i) to CHs;
when CHs answers or time elapsed
if(caching nodes found) **then**
    select caching node with largest RE;
    send request(d_i) to caching node;

else
    send request(d_i) to data center;
when N_i receives request(d_i)
if(N_i has a valid copy) **then**
    send d_i to requester node;
else if(N_i is CH **and** d_i's id in PCT) **then**
    select caching node with largest RE;
    redirect request(d_i) to caching node;
else
    forward request(d_i) to caching node;

(B) Replacement Policy
while(current node has not enough FS)
    Select a valid d_i with largest value and store it temporary;
    Send to CHs d_i's id;
    Remove the valid d_i;
when a CH gets d_i's id
if(CH gets d_i's id and d_i's id not in PCT) **then**
    select caching node with largest RE and FS;
send answer to requester node;
when current node get answers from CHs
foreach(temporary stored d_i)
    if(there is no other caching node) **then**
        Select caching node with least RE and largest FS;
        Send d_i to new caching node;
    Remove temporary stored d_i;

(C) Cache Admission Policy
when the packet with d_i obtained from current node
if(current node is packet's destination) **then**
    if(there is enough FS) **then**
        cache d_i;
        send ipacket to CHs;
    else
        call Replacement Policy ;
when CH gets an ipacket
if(CH get ipacket) **then**
        store d_i's id, RE and FS in PCT;

References

1. Akyildiz I, Melodia T, Chowdhury KR (2007) A survey of wireless multimedia sensor networks. Comput Netw 51(4): 921–960
2. Akyildiz I, Su W, Sankarasubramaniam Y, Cayirci E (2002) A survey of wireless sensor networks. IEEE Commun Mag 40(8):102–116
3. Basagni S, Mastrogiovanni M, Panconesi A, Petrioli C (2006) Localized protocols for ad hoc clustering and backbone formation: a performance comparison. IEEE Trans Parallel Distrib Syst 17(4):292–306
4. Chand N, Joshi RCRC, Misra M (2006) A zone co-operation approach for efficient caching in mobile ad hoc networks. Int J Commun Syst 19:1009–1028
5. Chand N, Joshi RCRC, Misra M (2007) Cooperative caching strategy in mobile ad hoc networks based on clusters. Wirel Pers Commun 43(1):41–63
6. Che H, Tung Y, Wang Z (2002) Hierarchical web caching systems: modeling, desing and experimental results. IEEE J Sel Areas Commun 20(7):1305–1314
7. Chow C-Y, Leong HV, Chan ATS (2004) Cache signatures for peer-to-peer cooperative caching in mobile environments. In: Proceedings of the IEEE international conference on advanced information networking and applications (AINA), vol 1. IEEE, Piscataway, pp 96–101
8. Chow C-Y, Leong HV, Chan ATS (2004) Peer-to-peer cooperative caching in mobile environments. In: Proceedings of the IEEE international conference on distributed computing systems workshops (ICDCSW). IEEE, Piscataway, pp 528–533
9. Chow C-Y, Leong HV, Chan ATS (2007) GroCoca: group-based peer-to-peer cooperative caching in mobile environment. IEEE J Sel Areas Commun 25(1):179–191
10. Diao Y, Ganesan D, Mathur G, Shenoy P (2007) Rethinking data management for storage-centric sensor networks. In: Proceedings of the conference on innovative data systems research (CIDR). Asilomar, 7–10 January 2007, pp 22–31
11. Eisenberg Y, Luna CE, Pappas TN, Berry R, Katsaggelos AK (2002) Joint source coding and transmission power management for energy efficient wireless video communications. IEEE Trans Circuits Syst Video Technol 12(6):411–424
12. Fan L, Cao P, Almeida AZ, Broder JM (2000) Summary cache: a scalable wide-area web cache sharing protocol. IEEE/ACM Trans Netw 8(3):281–293
13. Gandhi R, Parthasarathy S (2004) Fast distributed well connected dominating sets for ad hoc networks. Computer Science Department, University of Maryland at College Park, Tech. Rep. CS-TR-4559
14. Girod B, Kalman M, Liang YJ, Zhang R (2002) Advances in channel-adaptive video streaming. Wirel Commun Mob Comput 2(6):573–584
15. Hara T (2003) Replica allocation methods in ad hoc networks with data update. ACM/Kluwer Mob Netw Appl 8(4): 343–354
16. Hara T, Madria SK (2006) Data replication for improving data accessibility in ad hoc networks. IEEE Trans Mob Comput 5(11):1515–1532
17. Huang GT (2003) Casting the wireless sensor net. Technol Rev 106:51–56
18. Karl H, Willig A (2006) Protocols and architectures for wireless sensor networks. Wiley, New York
19. Katsaros D, Manolopoulos Y (2004) Caching in web memory hierarchies. In: Proceedings of the ACM symposium on applied computing. Cyprus, 14–17 March 2004, pp 1109–1113
20. Katsaros D, Manolopoulos Y (2004) Web caching in broadcast mobile wireless environments. IEEE Internet Comput 8(3):37–45
21. Katsaros D, Manolopoulos Y (2006) The geodesic broadcast scheme for wireless ad hoc networks. In: Proceedings of the IEEE international symposium on world of wireless, mobile multimedia (WoWMoM). IEEE, Piscataway, pp 571–575
22. Kulkarni P, Ganesan D, Shenoy P, Lu Q (2005) Sens Eye: a multi-tier camera sensor network. In: Proceedings of the ACM international conference on multimedia (MM). Singapore, 6–11 November 2005, pp 229–238

23. Mathur G, Desnoyers P, Ganesan D, Shenoy P (2006) Ultra-low power data storage for sensor networks. In: Proceedings of the ACM international conference on information processing in sensor networks (IPSN). Nashville, 19–21 April 2006, pp 374–381
24. Megiddo N, Modha DS (2003) ARC: a self-tuning, low overhead replacement cache. In: Proceedings of the USENIX conference on file and storage technologies (FAST). San Francisco, 31 March–2 April 2003, pp 115–130
25. Nath S, Kansal A (2007) FlashDB: dynamic self-tuning database for NAND flash. In: Proceedings of the ACM international conference on information processing in sensor networks (IPSN). Cambridge, 25–27 April 2007, pp 410–419
26. Papadopouli M, Schulzrinne H (2001) Effects of power conservation, wireless coverage and cooperation on data environments. In: Proceedings of ACM symposium on mobile ad hoc networking and computing (MOBIHOC). Long Beach, 4–5 October 2001, pp 117–127
27. Perkins CE, Royer E (1999) Ad hoc on-demand distance vector routing. In: Proceedings of the IEEE workshop on mobile computing systems and applications. IEEE, Piscataway, pp 90–100
28. Prabh KS, Abdelzaher TF (2005) Energy-conserving data cache placement in sensor networks. ACM Trans Sensor Netw 1(2):178–203
29. Rahimi M, Baer R, Iroezi OI, Garcia JC, Warrior J, Estrin D, Srivastava M (2005) Cyclops: in situ image sensing and interpretation in wireless sensor networks. In: Proceedings of the ACM international conference on embedded networked sensor systems (SenSys). San Diego, 2–4 November 2005, pp 192–204
30. Rousskov A, Wessels D (1998) Cache digests. Comput Netw ISDN Syst 30(22–23):2155–2168
31. Sailhan F, Issarny V (2002) Energy-aware web caching for mobile terminals. In: Proceedings of the IEEE international conference on distributed computing systems workshops (ICDCSW). IEEE, Piscataway, pp 820–825
32. Sailhan F, Issarny V (2003) Cooperative caching in ad hoc networks. In: Proceedings of the IEEE international conference on mobile data management (MDM). IEEE, Piscataway, pp 13–28
33. Shen H, Das SK, Kumar M, Wang Z (2004) Cooperative caching with optimal radius in hybrid wireless networks. In: Proceedings of the international IFIP-TC6 networking conference (NETWORKING), ser. Lecture Notes on Computer Science, vol 3042. Athens, 9–14 May 2004, pp 841–853
34. Sobeih A, Hou JC, Kung L-C, Li N, Zhang H, Chen W-P, Tyan H-Y, Lim H (2006) J-Sim: a simulation and emulation environment for wireless sensor networks. IEEE Wireless Commun Mag 13(4):104–119
35. Takaaki M, Aida H (2003) Cache data access system in ad hoc networks. In: Proceedings of the IEEE spring semiannual vehicular technology conference (VTC), vol 2. IEEE, Piscataway, pp 1228–1232
36. Tang B, Das S, Gupta H (2005) Cache placement in sensor networks under update cost constraint. In: Proceedings of the (ADHOC-NOW), ser. Lecture Notes on Computer Science, vol 3738. Springer, Heidelberg, pp 334–348
37. Wessels D, Claffy K (1998) ICP and the Squid Web cache. IEEE J Sel Areas Commun 16(3):345–357
38. Yin L, Cao G (2006) Supporting cooperative caching in ad hoc networks. IEEE Trans Mob Comput 5(1):77–89

Nikos Dimokas was born in Giannitsa, Greece in 1978. He received B.Sc. and M.Sc. in Computer Science from University of Crete, Greece, in 2001 and 2004, respectively. Between May 2004 and December 2005 he worked as a research and development engineer in the Institute of Computer Science at Foundation of Research and Technology Hellas (FORTH). Currently, he is a Ph.D. candidate at the Department of Informatics of Aristotle University of Thessaloniki, Greece. His research interests include wireless sensor networks and vehicular ad hoc networks.

Dimitrios Katsaros was born in Thetidio (Farsala), Greece in 1974. He received a BSc in Computer Science from Aristotle University of Thessaloniki, Greece (1997) and a Ph.D. from the same department on May 2004. He spent a year (July 1997–June 1998) as a visiting researcher at the Department of Pure and Applied Mathematics at the University of L'Aquila, Italy. Currently, he is a lecturer at the Department of Computer and Communication Engineering of University of Thessaly (Volos, Greece). He is editor of the book "Wireless Information Highways" (2005), co-guest editor of a special issue of IEEE Internet Computing on "Cloud Computing" (2009–2010), and translator for the greek language of the book "Google's PageRank and Beyond: The Science of Search Engine Rankings". His research interests are in the area of distributed systems, including the Web and Internet, social networks analysis, mobile and pervasive computing, flash storage devices, mobile/vehicular ad hoc networks, wireless sensor networks, and delay/disruption tolerant networks.

Yannis Manolopoulos was born in Thessaloniki, Greece in 1957. He received a B.Eng. (1981) in Electrical Eng. and a Ph.D. degree (1986) in Computer Eng., both from the Aristotle Univ. of Thessaloniki. Currently, he is Professor at the Department of Informatics of the same university. He has been with the Department of Computer Science of the Univ. of Toronto, the Department of Computer Science of the Univ. of Maryland at College Park and the Univ. of Cyprus. He has published over 200 papers in journals and conference proceedings. He is co-author of the books "Advanced Database Indexing", "Advanced Signature Indexing for Multimedia and Web Applications" by Kluwer and of the books "R-Trees: Theory and Applications", "Nearest Neighbor Search: a Database Perspective" by Springer. He has co-organized several conferences (among others ADBIS2002, SSTD2003, SSDBM2004, ICEIS2006, ADBIS2006, EANN2007). His research interests include Databases, Data mining, Web Information Systems, Sensor Networks and Informetrics.