Ad Hoc Networks 8 (2010) 214–240

Contents lists available at ScienceDirect

Ad Hoc Networks

journal homepage: www.elsevier.com/locate/adhoc

Cache consistency in Wireless Multimedia Sensor Networks

N. Dimokas [a], D. Katsaros [b,*], Y. Manolopoulos [a]

[a] Informatics Dept., Aristotle University, Thessaloniki, Greece
[b] Computer and Communication Engineering Dept., University of Thessaly, Volos, Greece

ARTICLE INFO

Article history:
Received 4 September 2008
Received in revised form 30 July 2009
Accepted 3 August 2009
Available online 8 August 2009

Keywords:
Cache consistency
Cache coherency
Replacement policy
Cooperative caching
Data dissemination
Wireless sensor networks

ABSTRACT

The production of cheap CMOS cameras, which are able to capture rich multimedia content, combined with the creation of low-power circuits, gave birth to what is called *Wireless Multimedia Sensor Networks (WMSNs)*. WMSNs introduce several new research challenges, mainly related to mechanisms to deliver application-level Quality-of-Service (e.g., latency minimization). Such issues have almost completely been ignored in traditional WSNs, where the research focused on energy consumption minimization. Towards achieving this goal, the technique of cooperative caching multimedia content in sensor nodes can efficiently address the resource constraints, the variable channel capacity and the in-network processing challenges associated with WMSNs. The technological advances in gigabyte-storage flash memories make sensor caching to be the ideal solution for latency minimization. Though, with caching comes the issue of maintaining the freshness of cached contents. This article proposes a new cache consistency and replacement policy, called *NICC*, to address the cache consistency issues in a WMSN. The proposed policies recognize and exploit the mediator nodes that relay on the most “central” points in the sensor network so that they can forward messages with small latency. With the utilization of mediator nodes that lie between the source node and cache nodes, both push-based and pull-based strategies can be applied in order to minimize the query latency and the communication overhead. Simulation results attest that *NICC* outperforms the state-of-the-art cache consistency policy for MANETs.

1. Introduction

The convergence of the Internet, communications, and information technologies, coupled with recent advances in engineering fields, paved the way for the production of inexpensive sensors, capable of achieving orders of magnitude spatial and temporal resolution and accuracy, compared to what was the situation a decade ago. This generation of the sensors established the now mature research and development area of wireless sensor networks [1] (WSNs). The deployment of networks of hundreds of sensors created a wealth of applications, like military applications (e.g., monitoring friendly/inimical forces and equipment, attack detection, targeting), environmental applications (e.g., microclimates, flood detection, precision agriculture), health applications (e.g., remote monitoring of physiological data, elderly assistance), commercial applications (e.g., inventory control).

More recently, the production of cheap CMOS cameras and microphones, which can acquire rich media content from the environment, created a new tidal wave into the evolution of wireless sensor networks. For instance, the Cyclops imaging module [2] is a light-weight imaging module which can be adapted to MICA2[1] or MICAz sensor

* Corresponding author. Tel.: +30 24210 74975; fax: +30 24210 74997.
E-mail addresses: dimokas@delab.csd.auth.gr (N. Dimokas), dkatsar@inf.uth.gr, dimitris@delab.csd.auth.gr, dkatsaro@csd.auth.gr (D. Katsaros), manolopo@delab.csd.auth.gr (Y. Manolopoulos).
URLs: http://delab.csd.auth.gr/~dimokas (N. Dimokas), http://www.inf.uth.gr/~dkatsar (D. Katsaros), http://delab.csd.auth.gr/~manolopo (Y. Manolopoulos).

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

1570-8705/$ - see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.adhoc.2009.08.001

nodes. Thus, a new class of WSNs came to the scene, the *Wireless Multimedia Sensor Networks (WMSNs)* [3].

Ian Akyildiz and his colleagues emphasized in their article [3] that WMSNs require us to rethink the computation–communication paradigm of traditional WSNs; this computation paradigm has almost exclusively focused on the preservation of the nodes' energy, with the ultimate target of prolonging the network's lifetime. Although the achievement of this goal remains fundamental in WMSNs, 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 data 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. WMSNs must address the following challenges:

- Energy, and processing capability constraints.
- Variable channel capacity, because the multi-hop nature of WMSNs implies that the capacity of each wireless link depends on the interference level among nodes.
- The need for multimedia in-network processing; sensor nodes are required to store rich media, 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. We could think of several ways to attack parts of this problem borrowing techniques from the 'wireless world', e.g., channel-adaptive streaming [4], joint source-channel coding [5]; though, none of them can provide solutions to all of the three aforementioned issues, esp. to in-network processing, thus alternative techniques should be pursued. Moreover, we could seek for solutions which have been developed in the context of the databases and WWW [6,7], but still these methodologies are not applicable because they do not suppose resource-starving nodes/links and moreover the patterns of communication between clients and servers is much more constrained, from many clients to a very specific (predefined) set of servers, but in our context, the pattern of communication is the most generic possible: any sensor can request and serve (relay) data. Therefore, even though the considered environment does not seem novel, a closer examination reveals that the challenges presented can not be dealt with the aid of already established techniques.

The success of the applications fuelled by WMSNs will be determined at a large degree by the technological advances related to the development of appropriate storage devices for sensor nodes, namely flash storage devices. The progress in this field is another indication of the rapid proliferation of WMSNs. Recent market predictions forecast that within the next three years the capacity of the flash memories (e.g., NAND) will be at the order of Terabytes. Presently, the NAND flash devices can store a few gigabytes data [8] and a lot of research work is devoted to developing appropriate data structuring mechanisms suitable for flash memories [9,10].

Under the assumption that each sensor node has a local storage capacity associated with it, we could collectively address the aforementioned challenges of WMSNs exploiting the technique of cooperative caching. In cooperative caching, multiple sensor nodes share and coordinate cache data without always having to visit the data centers. 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 and reduce the communication overhead and the data sources workload. Additionally, caching can be very effective in reducing the need for network-wide transmissions, thus reducing the interference and overcoming the variable channel conditions. Cooperative caching can also speed-up the multimedia in-network processing, because the processing and delivery of multimedia content are not independent. Thus, cooperative caching results in lower latency, energy dissipation and packet loss.

With caching comes the problem of maintaining the coherency of the cached data, namely the *cache consistency problem*. A lot of cache consistency policies have been proposed in the literature of databases, operating systems, and cellular wireless networks, and a handful of policies for ad hoc networks (see Section 2). Though, none of them can address the challenges of WMSNs. In our work, we study how to use application-level cache consistency to address these challenges. Example scenarios are illustrated in the following:

- In a battlefield sensor nodes are dispersed in a large target area where each sensor node is equipped with a communication device and a micro-camera that can take a photograph of its region. The sensors update the photographs taken and share with each other the new photographs, in order to built a more suitable view of the region that is being monitored. This is because every micro-camera can capture a limited view of the region either due to the sensor node's position or because of the obstacles that exist nearby the sensor node. However, every sensor node requests and receives a large number of photographs taken by other sensor nodes through multi-hop communication and constructs a large and accurate image of the monitored area. Every photograph has a unique identifier comprised by the region, the sensor and the time that the photograph has been taken. According to these features, a sensor can request a photograph of its interest and concludes about its freshness. Therefore, each military unit that lies in the battlefield can communicate with nearby sensor nodes to obtain an accurate view of the target region. This operation is crucial since each military unit can monitor more efficiently its corresponding perimeter.
- In a military scenario again [11], suppose there exists a sensor network which collects multimedia information about nearby activities. These data are queried in a dynamic fashion by soldiers to help them achieve their mission goals, to avoid possible sources of danger, and they are also queried by commanders to assess the progress of the whole mission. The queried data are real-time as well as long-term data about enemy activities (e.g., to answer a question such as where the supply lines are located). Thus, data must be cached at the sen-

sors to enable fast resolution of queries that span temporally short and long periods (e.g., days or even months), and such queries might originate from any part of the sensor network since the users (soldiers, commanders) are mobile and will submit their query to the nearest sensor.

- WMSNs can also be used in environmental monitoring. A large number of sensors can be deployed in a forest to prevent a disaster like fire or laying down refuse. This could be achieved through photographs taken by each sensor node. Rich multimedia data dissemination and sharing can offer an accurate view of the monitoring area. Thus, the human groups that monitor and protect the forest can communicate with a few sensors in order to obtain a detailed view of the forest area that is responsible for. This is because each sensor node can gather through multi-hop communication the pictures taken by other sensor nodes and create a large image of the corresponding subarea.

In this paper, we propose a novel and high-performance cache consistency protocol for cooperative caching, the *NICC* policy, named after the words *Node Importance-based Cache Consistency* for maintaining consistency in WMSNs. The proposed protocol is a hybrid push/pull algorithm that uses minimal message overheads to maintain data consistency. With the introduction of "mediator nodes", the caching invalidation performance can be improved greatly. Since there is no prior work on cache freshness for WMSNs, we compare the proposed protocol with the state-of-the-art consistency policy for mobile ad hoc networks, which is the "closer" competitor. Using the J-Sim simulation environment [12], we perform an experimental evaluation of the two methods, which attests that the proposed *NICC* 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 paper. Section 3 describes the system model and necessary preliminaries of our work. In Section 4 we present the details of the *NICC* protocol, and in Section 5 we present the results of the performance evaluation of the methods; finally, Section 6 concludes the article.

2. Relevant work

The issue of cache consistency is a thoroughly investigated area in the field of databases, operating/file systems, Web, and in wireless cellular (one-hop) networks. The article [13] contains an excellent categorization and survey of the most popular techniques developed for maintaining the cache freshness in these environments. In general, there exist two generic directions for notifying about the updates done in the data by the source nodes. Either the source nodes will disseminate, i.e., *push*, these changes via *invalidation reports* [14,15], or alternatively, the nodes caching the data, will query the source nodes, i.e., *pull* the updated data of interest [16,17].

In push-based strategies the source node needs to continuously or periodically send data to cache nodes. The modifications for a data item are pushed to the entire network using flooding. Each node, that receives the invalidation message, has to check its cache and invalidates the appropriate data. The advantage of the push scheme is the good consistency guarantees that provides and its simplicity. However this scheme suffers from unnecessary update transmissions, since not all previously interested nodes, are still interested in the updated data. It also increases the control message overhead due to subsequent broadcasts of the invalidation messages and entails message processing at every node.

In pull-based strategies, whenever a cache node needs data, the node can send a message to the server, and the server responds with the data. Thus, the cache nodes are responsible for maintaining consistency of their cached data items. The advantage of this scheme is the strong consistency guarantees and it is suitable for dynamic networks where there are frequent node disconnections and connections. However, this scheme suffers from too many *polling* messages travelling through the network, since the caching nodes have no way of knowing when or how often the updates take place in the source nodes. Also, the latency increases as the response time includes an additional delay, that it generated by the cache nodes querying the source nodes. Due to communication overhead the energy consumption will increase. Clearly, a hybrid among these policies would be the ideal solution, although some attempts to deploy them for ad hoc networks have been proposed in the literature [18].

Indeed, the articles [19,20] proposed such a hybrid policy for mobile ad hoc networks, and they comprise the most relevant research works to our proposed protocol.

The "Proximity Regions for Caching in Cooperative MP2P" (PReCinCt) scheme [20] divides the network topology into geographical regions, where each region is responsible for a set of keys representing the data. Each peer uses a region table to keep the location information of all regions in the whole network. Additionally, each key is then mapped to a region location based on a geographical hash function. A geographic-aided routing protocol is used to route traffic of each data retrieval. In order to further save bandwidth for each data retrieval, PReCinCt incorporates a cooperative caching scheme that caches relevant data among a set of peers in a region. When a peer requests a data item, it first floods the request in the region in which it resides (home region). If the data item cannot found in home region, the request is forwarded towards the destination region. The first node inside the destination region receiving the request message floods the message within the region to locate the peer caching the requested item. The PReCinCt scheme is also equipped with a cache consistency mechanism. The caching scheme considers data popularity, data size and region-distance during replacement to optimize cache content of peers. PReCinCt employs a hybrid push/pull mechanism to maintain data consistency among replicas in the network. However, the flooding technique increases the network traffic and battery depletion, especially when the region boundary are big and many sensor nodes reside in each region. Communication overhead is also generated when the location of each region has to be published in the entire network. Another drawback of the PReCinCt scheme is that each peer is location-aware. This imposes a constraint in sensor networks since each sensor should equipped with GPS-like hardware. Finally, it is not clear which is the optimal number of regions and the number of sensor nodes that reside in each of them. Thus, in a battlefield, where sensor nodes are dispersed in a large target area and are not equipped with GPS-like hardware, the regions is quite difficult to be defined dynamically.

The "Relay Peer-based Caching Consistency" (RPCC) policy [19] exploited "intervening" nodes, called relay nodes to implement this hybrid push/pull policy. With the help of relay peers between the source node and cache nodes, both push-based and pull-based strategies can be employed, which helps to reduce the communication overhead and query latency. The source host pushes the data to relay peers and cache hosts pull the data from relay peers. These operations can be performed asynchronously and simultaneously. The peers, that can listen to the invalidation messages sent from the source host of its cached data item, can become relay peers (each source host sent the invalidation message within a distance less than a number of hops). The authors use three parameters to define the relay peer selection criterion. The first parameter is the coefficient of peer access rate (CAR). The second parameter is the coefficient of stability of a node (CS), while the third parameter is the coefficient of the energy level of a node (CE). The authors predefine also three thresholds, one for each parameter, namely $th_{\mathrm{CAR}}, th_{\mathrm{CS}}$ and th_{CE}. A relay peer is considered candidate relay peer if it satisfies $(\mathrm{CAR} < th_{\mathrm{CAR}}) \wedge (\mathrm{CS} > th_{\mathrm{CS}}) \wedge (\mathrm{CE} > th_{\mathrm{CE}})$. Only after the cache peer receives an approval message from the source host, it can switch into a relay peer. However, the use of thresholds is a drawback, since it is quite difficult to define some values that are applicable to all network topologies and for different amount and different densities of sensor nodes. Additionally, a communication overhead generated during the broadcasting of invalidation messages from source hosts. This is because pure flooding technique is adopted. Thus, in case of a dense network topology the number of messages dispersed in network are increased significant.

Nevertheless, none of the two aforementioned policies considers the latency minimization criterion, which is vital for WMSNs. Moreover, they pay special attention in the design of the consistency policy so as to deal with MANET node mobility, which is almost absent in WMSNs. In particular, the method that RPCC uses to select the relays is based on a lot of administratively tuned parameters, which heavily rely on the query pattern of the nodes, thus refraining this policy for being adaptive. Our goal is similar to theirs but the *NICC* protocol proposed in this paper is more efficient in terms of communication overhead, energy dissipation and query latency.

Closely related works are the cache cooperation protocols developed for sensors [21,22] and ad hoc networks [23–28], but these protocols do not have a cache freshness maintenance component. The former protocols are based on centrality concepts in order to identify sensor nodes which control and coordinate the caching decisions; the *NICoCa* protocol [21] uses the well-known betweenness centrality metric for that task, and the *PCICC* protocol [22] is based on a novel metric that identifies sensors which are in dense areas of the network topology, curing at the same time some deficiencies of the betweenness centrality metric. In [29] the authors present an on demand relay placement algorithm in order to solve the problem of placing relay nodes in network regions where congestion is detected. Remotely related to the present work are the cache placement algorithms developed for wireless sensor networks [30,31] and ad hoc networks [32], the intrusion detection and intruder identification algorithms for cache consistency policies for ad hoc networks [33], and finally caching architectures and protocols for Internet-based wireless ad hoc and mesh networks [34–36].

2.1. Motivation and contributions

Motivated by the weaknesses of the current cooperative cache consistency protocols and the unique requirements of the WMSNs, we propose a cooperative cache consistency policy, namely the *NICC* protocol, 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. In summary, the article's contributions are the following:

- It introduces a new cooperative cache consistency protocol in order to maintain fresh data in WMSNs with the use of a metric that captures the significance of sensor node in the network topology. The discovery of significant sensor nodes will imply short latency in retrieval.
- Description of a cooperative cache consistency protocol.
- Development of the algorithm for disseminating the updates to caching sensor nodes and for purging out of the caches the less valuable data.
- Performance evaluation of the protocol and comparison with the state-of-the-art cooperative caching protocol for MANETs, namely RPCC, using an established simulation package (J-Sim).

3. System model

A WMSN consists of a collection of wireless interconnected sensor nodes deployed throughout the sensor field. There is no central server to maintain cache consistency. Each node is assigned a unique identifier in the system. We assume that l is the total number of sensor nodes and each node is denoted by $SN = \{SN_1, SN_2, \ldots, SN_l\}$. Each data item has also a unique identifier. We consider that m is the total number of data items and each data item is denoted by $D = \{D_1, D_2, \ldots, D_m\}$. Every data item is being assigned to each sensor node. Thus, each node is the data source of a unique data item (for example each sensor node with the use of micro-camera can capture an image). We assume that the data source of data item D_i is SN_i. Several copies of the same data item could be disseminated in the network and cached either by the mediator nodes or cache nodes. Each node has limited cache space, so only a portion of D can be cached locally. If the cache space of a node is full and it has to cache a new data item, the node needs to select an item from its cache for replacement. The modifications to a data item is being initiated only by the data source. Thus, only the data source has the most updated version of the data item.

- The sensor nodes are static. In the majority of applications, sensor nodes have no mobility.
- Links are bidirectional, thus if node SN_i can communicate with node SN_j then node SN_i is also reachable from node SN_j.
- The computation and communication capabilities are the same for all network nodes.
- One data item is being assigned to each sensor node and data items have varying sizes. Thus, each node constitute a data source.
- Every node can cache a number of data items, that it is being defined by the cache size.
- Every node calculates its significant neighbors (mediator nodes). Therefore, a node can be in two states: ordinary or mediator.
- Every node uses a routing protocol in order to send requests to source nodes.
- Every mediator node holds the following information for each cached data item:
  - DataID. It is the identification of the data item.
  - Source node ID that maintains the original copy of the cached data item.
  - Data item.
  - Size. The size of the data item is significant, since it contributes in the calculation of the cache replacement function.
  - Time to refresh. Each mediator node needs to know the time to refresh value, in order to attest the validity of cached data item. When time to refresh value is less than 0, data item is no longer valid.
  - Average number of hops of nodes requesting the data item. Every request contains a counter that computes the number of nodes that has passed by. Each sensor node increase the counter by one unit before forwarding the request. Thus, each mediator node is able to compute the average number of hops between the requesting nodes and itself. The bigger the distance between the requesting nodes and the mediator node, the more likely to remain in the cache during the cache replacement procedure.
  - Access rate. The access rate indicates the frequency that a cached item is being requested.
  - Update rate. The update rate determines the frequency that a data item is being modified
  - Version. The version of a data item is being assigned by the source node (the node that owns the request).

In the rest of the paper we use a number of notations. The complete list is presented in Table 1.

3.1. Problem formulation

Given an ad hoc network of sensor nodes $G(V,E)$ with $|dataID|$ data items $D_1, D_2, \ldots, D_{|dataID|}$, where data item D_j can be served by a sensor SN_o, a sensor may act as a server of multiple data items and has capacity of $|m_o|$ units. We use a_{ij} to denote the access frequency with which a sensor SN_i requests the data item D_j and d_{il} to denote the distance (in hops) between sensors i and l. The *cache consistency problem* is an online problem, with the goal being the selection of a set of sets $M - \{M_1, M_2, \ldots, M_{|dataID|}\}$, where M_j is a set of sensors that store a copy of D_j, to minimize the total access cost:

$$\tau(G,M) = \sum_{i \in V} \sum_{j=1}^{|dataID|} a_{ij} \times \min_{\{S_j\} \cup M_j} d_{il}$$

respecting the consistency requirements of D_j, and fulfilling the memory capacity constraint that:

$$|\{M_j | i \in M_j\}| \leqslant m_i \quad \text{for all } i \in V.$$

Since it can be easily be proven that this problem is NP-hard, in the next sections we develop online algorithms to achieve the aforementioned goals.

Table 1
List of notations.

Variable	Description
G	The graph corresponding to the ad hoc sensor network
V	The set of sensor nodes of G
E	The set of links between nodes of G
dataID	The id of data item
SN	Sensor node
src-req	The id of the requester node
src-med	The id of the mediator node
TTL	Time to live of an invalidation message
TTR	Time to refresh of data item at mediator node
DUI	Data update interval of data item at source node
CL	Consistency level (weak, delta, strong)
AR	Access rate of a data item
UR	Update rate of a data item
AvgHops	Average number of hops between the requesting and the responding node

4. The NICC consistency protocol for cooperative caching in WMSNs

In order to address the latency requirement of WMSNs, we attempt to identify some sensor nodes as being more “important” than the others, w.r.t. coordinating the procedure of cache coherency check and update messages forwarding. These nodes will be called “mediator” nodes, and will play the role of the representatives of the source (originator) nodes of the data in their communication with the cache nodes which store replicas of these data (sinks).

The mediator nodes constitute the main point of the cache coherence protocol. Each sensor node elects a number of mediator nodes that lie on the most “central” points in the sensor network neighborhood. The mediator selection procedure is localized with small communication overhead that is being performed once at the beginning of the network lifetime, since the sensor network is assumed relatively static. The mediator nodes lie between the source node and the cache nodes and they are closer to source node, in number of hops. The distance between source node and mediator nodes is defined by the TTL value that source node assigns to invalidation messages. When a mediator node receives an invalidation message and the distance from the source node is equal to TTL value, then it can cache the data item.

Because of the proximity between the source node and the mediator nodes, the communication is push-based, since the communication overhead will be relatively small. Thus, the source node and its mediator nodes can form an overlay network that could offer a valid data item to cache nodes. At the same time the communication between the mediator nodes and cache nodes is pull-based. Every time a new request is arrived in cache node, it has to poll the source node for fresh data. However, towards the path to the source node a mediator node can reply to polling message and offer valid data. Because the *push* and *pull* operations can be performed asynchronously and simultaneously, the communication overhead and the query latency can be reduced. Fig. 1 depicts an example that illustrates the *NICC* approach.

The next subsection describes the selection of mediator nodes, whereas Sections 4.2–4.4 present the major components of *NICC*.

4.1. Selection of mediator nodes

A wireless multimedia sensor network 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. 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 $N_{12}(v)$.

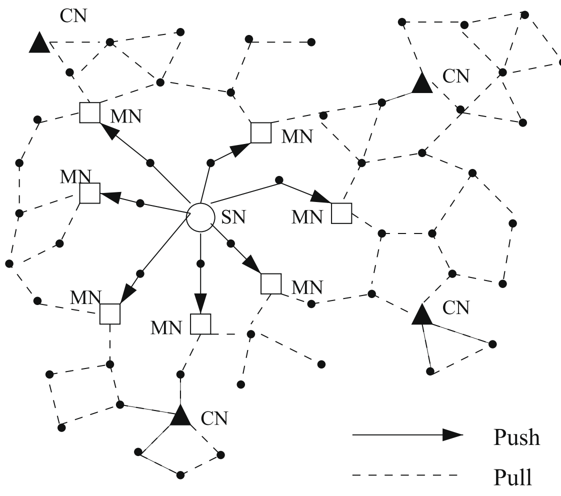


Fig. 1. An example of NICC approach.

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 $NI(v)$ of a vertex v as:

Definition 2. The $NI(v)$ of a vertex v is equal to:

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

Large values for the *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. Fig. 2 illustrates this metric, which is actually the betweenness centrality index as defined in the social networks literature [37]. Although, several other centrality metrics exist in the relevant literature (e.g., closeness centrality, spectral centrality), the *NI* index is the most appropriate for our target application, because the requests are directed from (potentially) any sensor to (potentially) any other sensor and thus we must seek for nodes that lie on many communicating paths among any pair of nodes. If, on the contrast, our target application was the dissemination of messages from a singe source to any other networks, then closeness centrality (or spectral centrality) would be more appropriate.

Apparently, when estimating the *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. Fortunately, even when we calculate the *NI* indexes of the nodes taking into account only their k-hop ($k=2$ or 3) neighborhood, the picture about the relative importance of the

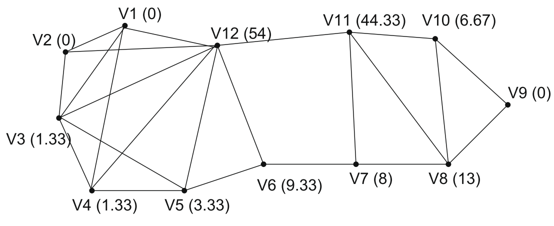


Fig. 2. Calculation of *NI* for a sample graphs. The numbers in parentheses denote the *NI* index of the respective node considering the whole WMSN topology.

Table 2
NI index of the nodes belonging to $LN_{V_{11}}$.

$n(ode)$	$NI_{V_{11}}(n)$	$n(ode)$	$NI_{V_{11}}(n)$	$n(ode)$	$NI_{V_{11}}(n)$
V_1	0	V_5	0	V_9	0
V_2	0	V_6	6	V_{10}	7
V_3	0	V_7	6.33	V_{11}	44.33
V_4	0	V_8	12.67	V_{12}	81.67

nodes remains very accurate. For instance, the *NI* index for the nodes belonging to $LN_{V_{11}}$ (see Fig. 2) are indicated in Table 2. For more information regarding the "localized" behaviour of this metric the interested reader is directed to [38]. The calculation of this localized metric, i.e., *NI* is quite low, and for neighborhoods comprised by v nodes and ϵ edges, each sensor computes this metrics for its neighbors in time $O(v * \epsilon)$ (for such and algorithm, consult [38]).

The next subsection explains the knowledge that each sensor node is supposed to have in order to execute the *NICC* cooperative cache consistency protocol.

4.2. Push–pull based approach

The source node "pushes" data and invalidation reports to all nodes within time-to-live (TTL) hops away. Each invalidation report is associated with a TTL, which determines the scope the invalidation report can reach. TTL value is initiated by the source node and it is decremented every time the data item passes from a sensor node. A node can listen the invalidation report when it is within a distance less than TTL number of hops from the source node. Additionally, we assume a sensor network in which the nodes exchange with their neighbors "Hello" messages (beacon messages), which contain the list of their neighbors. Thus, every node is aware of its two-hop neighborhood.

Mediator nodes are designated separately by each node of the network, according to the node importance value. Every node calculates the *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 *NI* which "cover" its 2-hop neighborhood are the mediator nodes for that node; mediator nodes undertake the task of rebroadcasting the invalidation reports of source nodes within TTL distance and they can use invalidation reports to check whether their cached copies are valid. The rebroadcasting of invalidation reports contributes to the reduction of communication overhead, that implies the push-based strategy. Thus, the number of messages that is being disseminated in order to transmit the invalidation reports are minimized. Each source node sends periodic invalidation reports – these intervals are set by the parameter of the system DUI (data update interval). The invalidation reports include dataID, data item (if changed, otherwise the null value), TTL (in number of hops), and version number. If the data item has been modified during the data update interval, the source node will issue an update message to its mediator nodes in order to inform them about the update. The update message include the modified data item. For every data item, each source node maintains the following information:

1. dataID,
2. data item,
3. size,
4. version.

The fact that a source node defines a TTL value, does not imply that every intermediate hop node (up to hopCount = TTL) will cache the data item. The data item is cached only in an elected mediator node whose distance from the source node (in number of hops) equals the TTL value. Thus, we create a ring of nodes around the source node where the invalidation reports are pushed and valid copies are cached. Every mediator node could cache data items from different source nodes.

The communication between mediator nodes and their cache nodes is pull-based. Cache nodes lies between mediator nodes and requester nodes and cache data items for subsequent requests. Every time an ordinary node asks for data from a source node, the request initially reaches a cache node. In case of strong consistency required (not just weak or *D* consistency), cache node forwards the request through the routing path until it reaches a mediator node of the source node, whose distance from the source node equals TTL hops. Every request includes the dataID, the version, src-req (the id of the requester node). If the mediator has a valid and fresher than request version of the saught data in its cache, then it sends a reply packet that includes dataID, data item, version, src-med (the id is the mediator node). If the version of the request is identical to that of mediator node then an AckPacket is sent which includes dataID, version, src-med (the id the mediator node). In case of an invalid data item, the mediator node holds the request until the next Invalidation report arrives and appropriate reply is sent. A mediator data item is considered valid when its TTR (time-to-refresh) value is greater than zero. In case that no mediator node along the routing path containing the seeked data item is not found, the request eventually reaches the source node. The source node sends the reply to the requester node. The requester node does not cache the data item since it is not able to receive the invalidation reports (we consider the case where strong consistency required – not just weak or *D* consistency). Thus, each time a requester node tries to communicate with either a mediator or a source node.

Based on the above idea, we describe the *NICC* protocol. For example, suppose that sensor node SN_a in Fig. 3 issue a request for data item x that is placed in data center *DC* and has been pushed to mediator nodes. TTL value is assumed to be 2. The square nodes are considered to be the mediators nodes of the sensor network, while triangle nodes are considered to be the cache nodes. In the beginning sensor node SN_a sends a request to cache node SN_c. Upon receiving the request, cache node checks the consistency level of the request. If *D* consistency is required and the *D* value equals zero or strong consistency required, cache node will forward the request to data center. Towards the path to data center, the request arrives to sensor node SN_f which

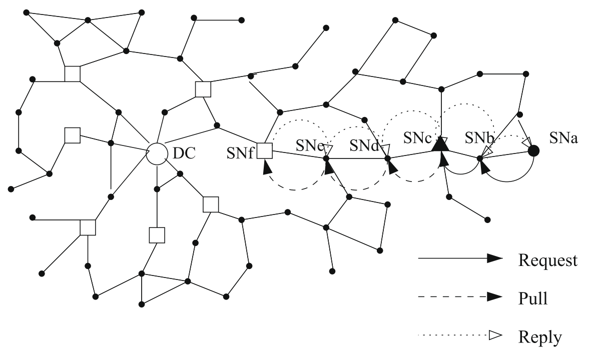


Fig. 3. An example of NICC operation when strong consistency or *D* consistency required and the *D* value equals zero.

is a mediator node for data item *x*. If the mediator has a valid and fresher version of the requested data item in its cache, then it sends a reply packet to cache node SN_c with the new version of data item. Cache node caches the new version of data item *x* and forwards the reply to SN_a. If the version of the request is identical to that of mediator node then an AckPacket is sent.

4.3. Consistency requirements

NICC can provide query requests with different levels of consistency. In this case, we introduce the concept of cache node. Cache nodes are placed between mediator nodes and the requester node. A request is initially forwarded to a cache node. If only weak-consistency is required, cache node can send immediately a reply back to the requester node. If *D* consistency is required and the *D* value is larger than zero, cache node can also serve immediately the request. However, if *D* value equals zero or strong consistency is required, the cache node has to poll the mediator node in order to check the consistency of data item. The mediator node can compare the versions and answer the cache node. If the mediator node has a valid copy of the data item, a reply message is sent back to the cache node. Otherwise, it will wait for the next invalidation message from the source node. Thus, the concept of cache node is used in case we must ensure 'delta' or 'weak' consistency. Cache nodes can be selected using the *NICoCa* protocol [21] or the *PCICC* protocol [22].

4.4. Cache replacement policy

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 cache replacement policy used in every node, either mediator or cache node (in the following discussion we examine mediator nodes) considers the following characteristics of the data items:

1. update rate,
2. access rate,
3. average number of hops of nodes requesting the data item,
4. data size.

When the mediator node gets a request message, it extracts the number of hops that the request has passed through. According to the number of hops, the mediator node calculates the average number of hops for each specific data item it stores. The bigger the average number of hops, the more likely to remain in the cache. This occurs in order to achieved query latency reduction, since the remote sensor node should not wait until the request reaches the source node. The access rate indicates the frequency that a cached item is being requested, while the update rate determines the frequency that a data item is being modified. The higher the rate access a data item, the more likely to remain in cache. The higher the rate of modification of a data item, the smaller the chance to remain in cache. Finally, the bigger the size of an object is, the more likely to be removed from the cache. Therefore, a mediator node, during cache replacement, calculates a value for each data item. Data item with the smaller value is replaced. The evaluation function, for each data item i located in the cache, is as follows:

$$Value(d_i) = \frac{\mathrm{AR}_i * \mathrm{AvgHops}_i}{\mathrm{UR}_i * Size_i},$$

where AR refers to access rate, AvgHops is the average number of Hops, and UR denotes update rate. The update rate of every data item is evaluated by mediator nodes based on the data item refresh rate.

4.5. Radio irregularity

For the design of all the components of the proposed cooperative cache consistency protocol, we assumed that the communication links among the sensors are bidirectional (see Section 3). Though, this assumption may not always hold, e.g., [39]. Even in this case, the protocol's semantics would not change. In particular, the *NI* metric would remain the same, but its calculation would be more CPU-intensive; its computational complexity would not be that reported in Section 4.1 using a breadth-first algorithm [38], but it would require the use of the Dijkstra's algorithm. The rest of the protocol components, i.e., push, pull, replacement policy would not require any changes.

The existence of radio irregularity could deteriorate the expected performance of the protocol when this irregularity would be located in "dense" areas of the network. With the assumption of bidirectionality, the sensors residing in dense areas usually have a quite large value for the *NI* index, and therefore can route messages from sources to destinations with small latency. High "concentration" of radio irregularity in such areas would alter the distribution of sensors with large *NI* values, and thus increase latency. Of course, such performance degradation would be expected for all cooperative caching protocols, because radio irregularity significantly reduces the number of alternative routes for messages and creates congestion and higher drop rates.

5. Performance evaluation

We evaluated the performance of the *NICC* protocol through simulation experiments. We conducted a large number of experiments with various parameters, and compared the performance of *NICC* to the state-of-the-art cache consistency policy for MANETs, namely RPCC [19].

We do not compare *NICC* to PReCinCt scheme [20], because it can not be applied in sensor networks since it requires by the sensor nodes to know the approximate geographic locations of other sensors. In PReCinCt scheme, it is not also obvious which is the exact number of regions and how to be set dynamically in a sensor network where sensor nodes are dispersed non-uniformly in a large target area and are not equipped with GPS-like hardware. Additionally, the number of nodes that reside in each region is obscure. According to *NICC* and *RPCC* protocols, a node does not need to be informed about the approximate locations of other nodes.

5.1. Simulation model

We have developed a simulation model based on the J-Sim simulator [12]. In our simulations, the AODV [40] MANET routing protocol is deployed to route the data traffic in the wireless multimedia sensor network. Although other routing schemes, like GPSR [41], are more appropriate for sensor networks, for reasons of fair comparison with the competitor we employ this routing protocol; though we conducted the experiments using GPSR as well to confirm that the relative performance of the algorithms remains the same, and, for the interest of space, we include only a couple of representative graphs. We use IEEE 802.11 as the MAC protocol and the free space model as the radio propagation model. The wireless bandwidth is assumed to be 2 Mbps. Additionally, the radio characteristics used in our simulations are summarized in Table 3. To test the validity of the experiments in more radio-starving environments, we also conducted experiments using the IEEE 802.15.4 as the MAC protocol.

The protocols has been tested 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 500. In order to achieve the protocols' network operation, we run each protocol at least 100 times for each different node degree. 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 1 KB to 10 KB for the first set, and from 1 MB to 5 MB for the second set.

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 according to the algorithm described in [42]. Initially, all the nodes are placed uniformly and independently. The location of each of the n sensor nodes is uniformly distributed between the point $(x = 0,\ y = 0)$ and the point $(x = 2000,$ $y = 2000)$. All $n * (n - 1)/2$ potential edges in the network are sorted by their lengths in ascending order. The length of the $n * d/2$th shortest edge is chosen as the transmission range *R*. The first $n * d/2$ edges in the list remain in the graph. Other edges are eliminated, making sure that the graph generated is a unit disk graph and has average degree equal to *d*. However, the J-Sim simulator artificially divides the whole simulated area into subareas where each subarea is a square grid unit. Two sensor nodes are neighbors if they placed in the same grid or in adjacent grids. In order to set the grid unit size, we compute the maximum side of square that fit in the circle with radius *R*. Therefore, 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. Finally, each candidate graph is checked for connectivity. If it is not connected, the procedure is repeated until a connected graph is generated.

Table 3
Radio characteristics used in our simulations.

Operation	Energy dissipated
Transmitter/receiver electronics	$E_{elec} = 50$ nJ/bit
Transmit amplifier if $d_{toBS} \leqslant d_0$	$e_{fs} = 10$ pJ/bit/m^2
Transmit amplifier if $d_{toBS} \geqslant d_0$	$e_{mp} = 0.0013$ pJ/bit/m^4
Data aggregation	$E_{DA} = 5$ nJ/bit/signal

The simulation area is assumed of size 2000 m$\times$ 2000 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 query model is similar to what have been used in previous studies [19]. Each sensor node generates read-only query requests with an exponential distributed query interval. They also generates an independent stream of updates to its source data with an exponential distributed update interval. The access pattern of sensor nodes is: (a) location independent, that is, sensor nodes decide independently the data of 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 400 m$\times$ 400 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 0.1 and 1.0. For the interest of space, we present only the results for $\theta = 0.8$ since we believe that it is the most representative access pattern. One data item is assigned to each sensor node and so the total number of data items is 100 in case of 100 sensor nodes topology and 500 in case of 500 sensor nodes topology. The simulation system parameters are listed in Table 4.

5.2. Performance metrics

The measured quantities include the number of hits (local, remote and global), the average query latency, the message overhead, the residual energy level of the sensor nodes and the impact of invalidation message's TTL on the system performance. The query latency is the time elapsed between the query is sent and the data is transmit-

Table 4
Simulation parameters.

Parameter	Default value	Range
# Items (N)	100	100–500
$S_{\min}$ (KB)	1	
$S_{\max}$ (KB)	10	
$S_{\min}$ (MB)	1	
$S_{\max}$ (MB)	5	
Zipfian θ	0.8	
# Nodes(n)	100	100–500
# Requests per node	200	100–200
Bandwidth (Mbps)	2	
Query interval	20 s for items with KB size 3 min for items with MB-size	
Update interval	2 min for items with KB size 20 min for items with MB-size	
Client cache size (KB)	75	10–100
Client cache size (MB)	37.5	15–65
TTL	3 hops	3–6

ted back to the requester, and average query latency is the query latency averaged over all the queries. A commonly used message overhead metric is the total number of messages injected into the network by the query process. The message overhead includes all the query and response messages of locating and retrieving data. The three different types of hits are the following:

1. Local hit (LH): the requested datum is cached by the node which issued the request, and it is valid.
2. Remote hit (RH): 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 source node for that datum.
3. Global hit (GH): the requested datum is acquired from its source node.

Evidently, 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 and coherency check. A large number of local hits does not imply an effective caching policy, unless it is accompanied by small number of global hits, since the cost of global hits vanishes the benefits of local hits. The independent parameter in all these experiments was the cache size of the sensors. In any case, the local sensor flash storage ranges from 1% to 10% of the total size of all distinct multimedia data.

5.3. Evaluation

We performed a 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 and 500 sensors, for cache size equal to 1%, 5% and 10% of the aggregated size of all distinct multimedia data, for access pattern with θ equal to 0.8 (skewed access pattern), for average sensor node degree equal to 4 (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). 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. Finally, we present the impact of different TTL values. Following the same methodology as that of RPCC, we set the TTL equal to 3. The effect of different TTL values will be discussed in Section 5.3.3.

5.3.1. Experiments with large data items

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

All figures show that both schemes exhibit better average query latency, hit ratio (local,remote and global) and message overhead by varying the cache size from 15 MB to 65 MB for both sparse and dense sensor deployments. This is because more required data items can be found in the local cache as the cache gets larger. The results are illustrated in Figs. 4 and 5. The first prevalent observation

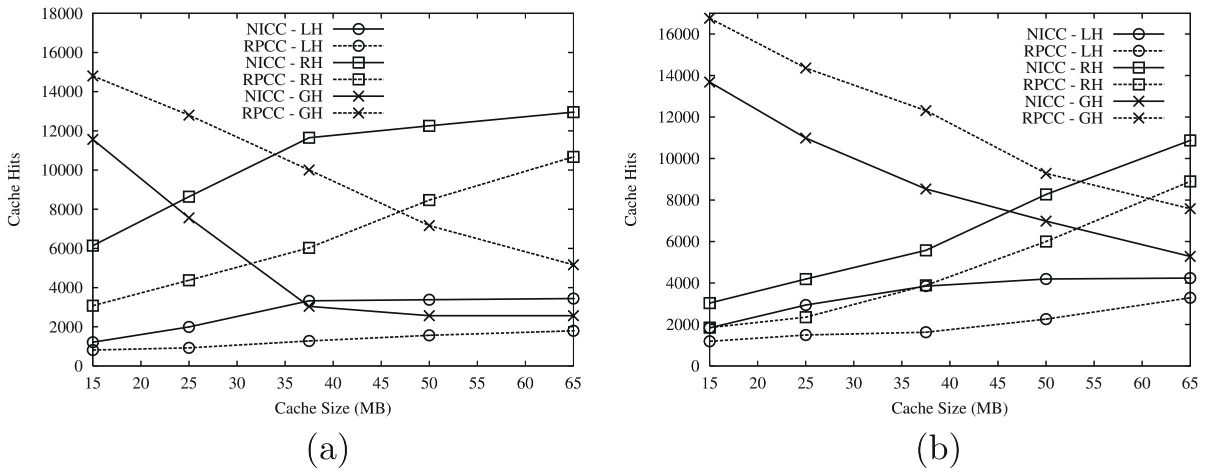


Fig. 4. Impact of cache size on hits (MB-sized files) in: (a) sparse ($d = 4$), (b) dense ($d = 10$) WMSN with 100 sensors.

is that the performance of the proposed algorithm is always better than the performance of its competitor. The *NICC* scheme achieves always higher number of remote hits than RPCC scheme, since the mediator nodes located in central points of the sensor network. Also, *NICC* achieves a smaller number of global hits than RPCC. Thus, source nodes do not become hot spots of communication and the workload is significant smaller than those of RPCC protocol. Additionally, *NICC* achieves a performance gain in terms of local hits.

The second prevalent observation is that the performance gap between *NICC* and RPCC is larger for the sparser sensor network deployments. This is true for the subsequent performance metrics as well. This is due to the fact that in denser deployments, NICC takes better decisions regarding the selection of the relay peers. This is apparent when examining the graphs depicting the relative remaining energy per sensor (cf. Figs. 14 and 15). Therefore, the ad hoc and highly parameterized methods for the selection of nodes which will have "special roles" (mediators vs. relay peers) have a profound impact on the performance of the consistency check algorithms.

Then, we evaluated the performance of the algorithms with respect to the average latency incurred for varying cache sizes, when the transmitted files have sized of a few megabytes for both sparse and dense sensor network deployments. The results are depicted in Figs. 6 and 7. The latency incurred by *NICC* is 20–30% smaller than that of RPCC. Due to cache replacement policy and the central points that mediator nodes located, the query requests are served faster than that of RPCC. We can also observe that in *NICC* the reduction of response time for cache sizes between 37.5 MB and 65 MB is not significantly high. It is worth noting that *NICC* always reach the near optimum performance when the cache size is equal to 37.5 MB. This demonstrates the low cache space requirement.

Additionally, we evaluated the performance of the algorithms with respect to the number of transmitted messages for varying cache sizes for both sparse and dense sensor network deployments. The results are depicted in

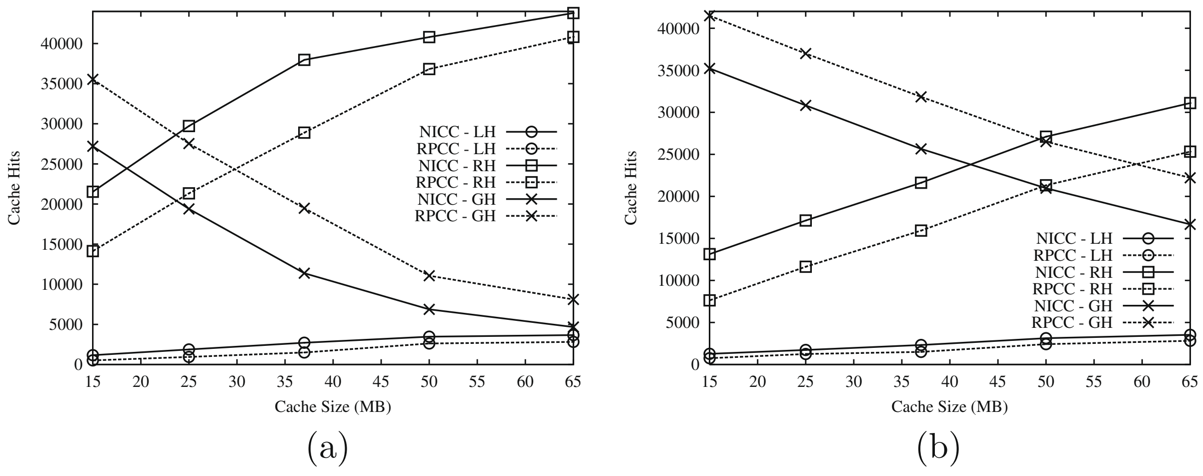


Fig. 5. Impact of cache size on hits (MB-sized files) in: (a) sparse ($d = 4$), (b) dense ($d = 10$) WMSN with 500 sensors.

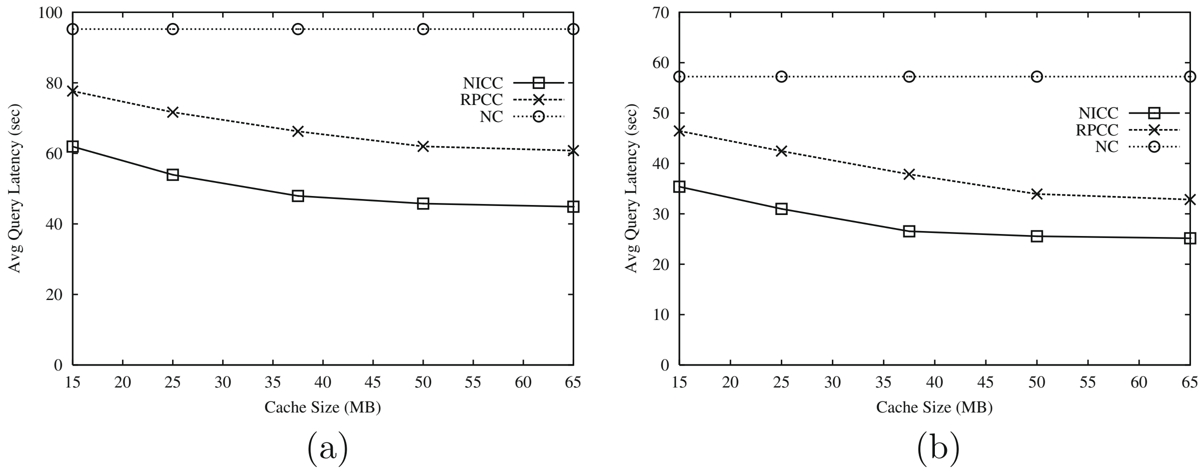
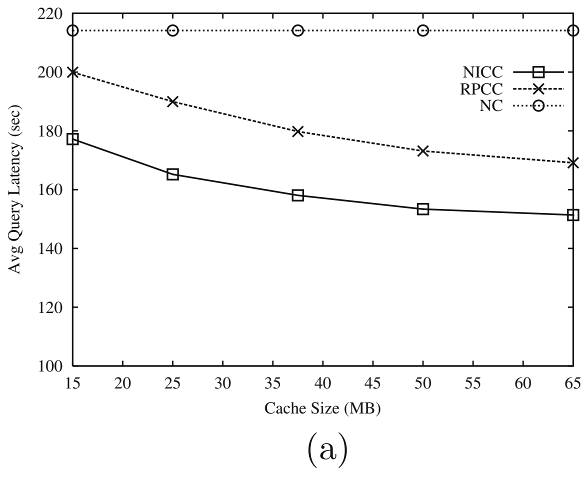


Fig. 6. Impact of cache size on latency (MB-sized files) in: (a) sparse ($d = 4$), (b) dense ($d = 10$) WMSN with 100 sensors.



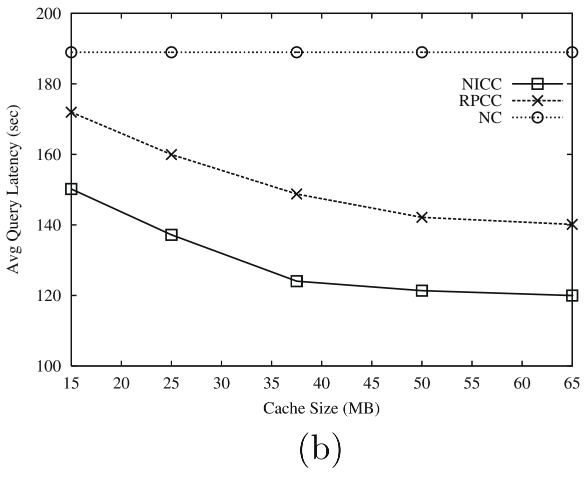
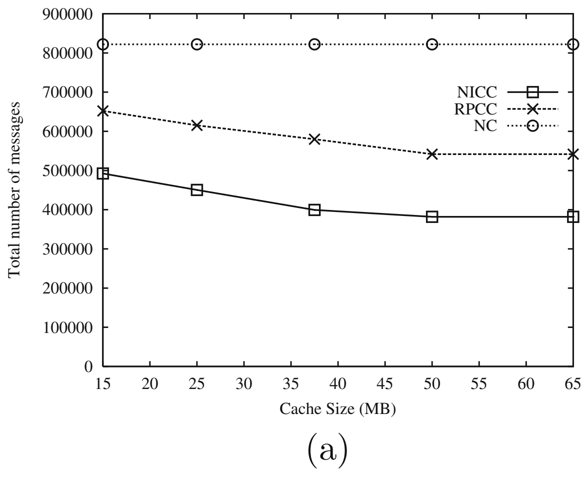


Fig. 7. Impact of cache size on latency (MB-sized files) in: (a) sparse ($d = 4$), (b) dense ($d = 10$) WMSN with 500 sensors.



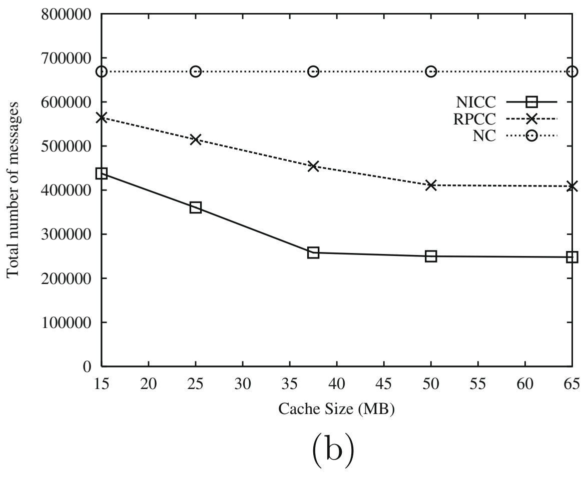
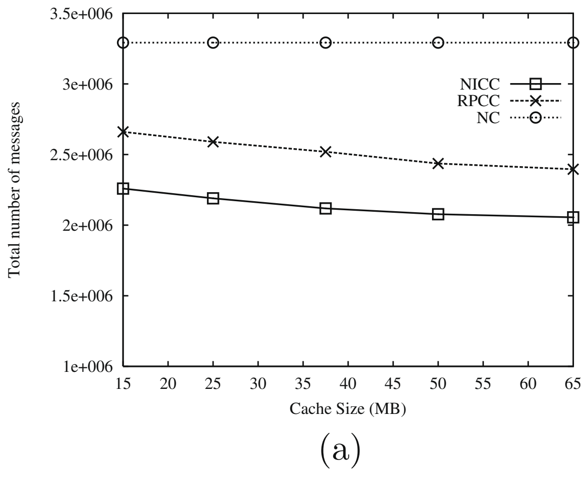


Fig. 8. Impact of cache size on number of messages (MB-sized files) in: (a) sparse ($d = 4$), (b) dense ($d = 10$) WMSN with 100 sensors.



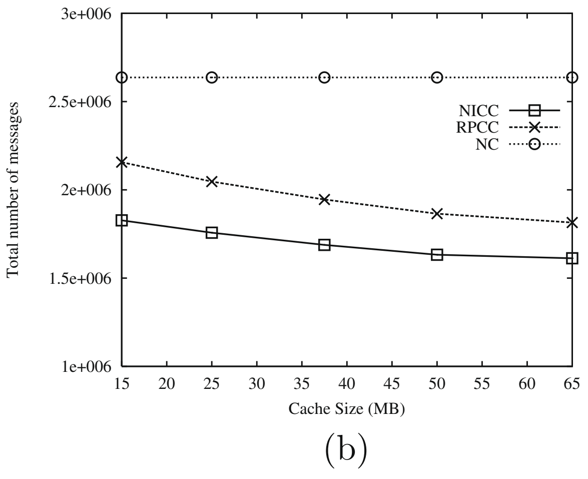


Fig. 9. Impact of cache size on number of messages (MB-sized files) in: (a) sparse ($d = 4$), (b) dense ($d = 10$) WMSN with 500 sensors.

Figs. 8 and 9. The relative results follow the same trends that we observed in the previous experiment; there is a close connection between the messages and latency, since the more the number of messages transmitted, the more intense the competition for the broadcast channels is, and thus many more collisions occur, which collectively aggravate the average latency. The number of messages in sparse networks is larger than those in dense networks. This is due to the number of hops that a message request should pass by until it reaches a responding node. In sparse networks the number of hops increases in contrast to dense networks.

Figs. 10 and 11 evaluates the performance in terms of query latency (in log scale) with varying values of request interval and update interval. Fig. 10a and Fig. 11a illustrate that query latency is a bit bigger for small values of request interval. This is because the network traffic increases as the request interval decreases. Thus, a delay occurs due to channel congestion. Both figures present a significant performance gain of *NICC* over RPCC. In *NICC* requests are served by mediator nodes that located in central points of network topology. So the round trip time of request is smaller. Fig. 10b and Fig. 11b show the results under different update interval. We note that there is not a significant difference of both protocols in terms of query latency as the update interval widens. Both protocols have a slightly worse performance for small update intervals, since more invalidation messages are disseminated by source nodes. This results in larger network traffic. Finally, we can observe that *NICC* performs also better than RPCC. These graphs also confirm the significance of the caching as a latency-reduction mechanism; the *NICC* protocols achieves 50% reduction in latency compared to the no-caching approach.

Fewer communication messages lead to less network traffic and energy communication. Therefore, a good algorithm should send a small number of messages. Figs. 12 and 13 evaluates the performance in terms of network traffic with varying values of request interval and update interval. As we can see, the RPCC protocol always leads

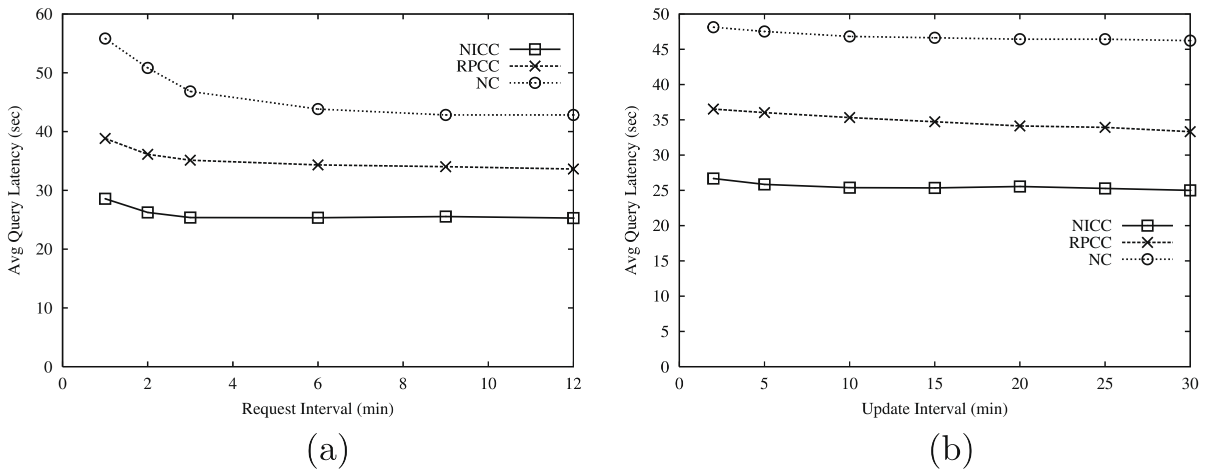


Fig. 10. Query latency (MB-sized files) according to: (a) request interval, (b) update interval for a WMSN with 100 sensors.

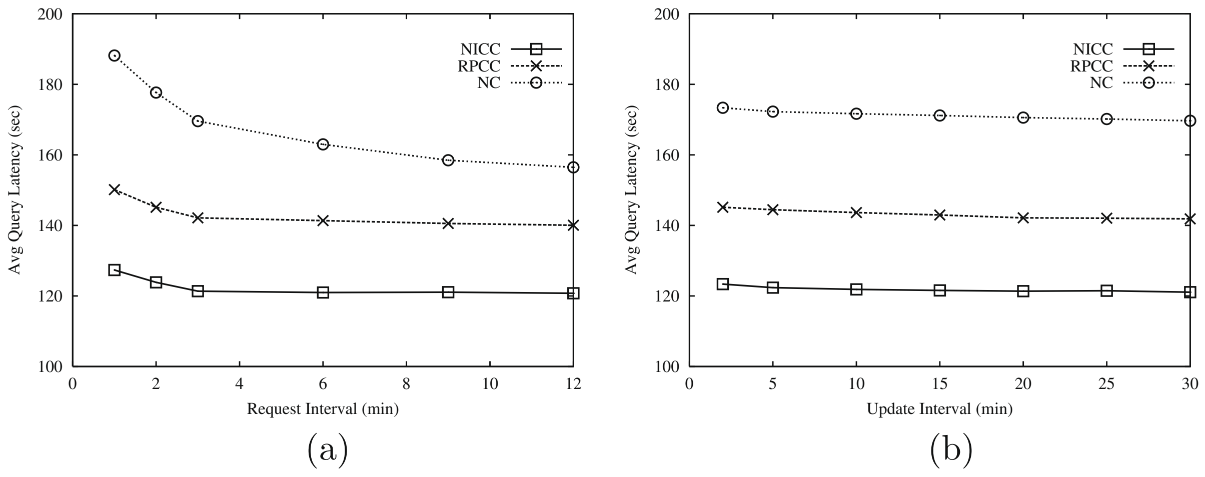


Fig. 11. Query latency (MB-sized files) according to: (a) request interval, (b) update interval for a WMSN with 500 sensors.

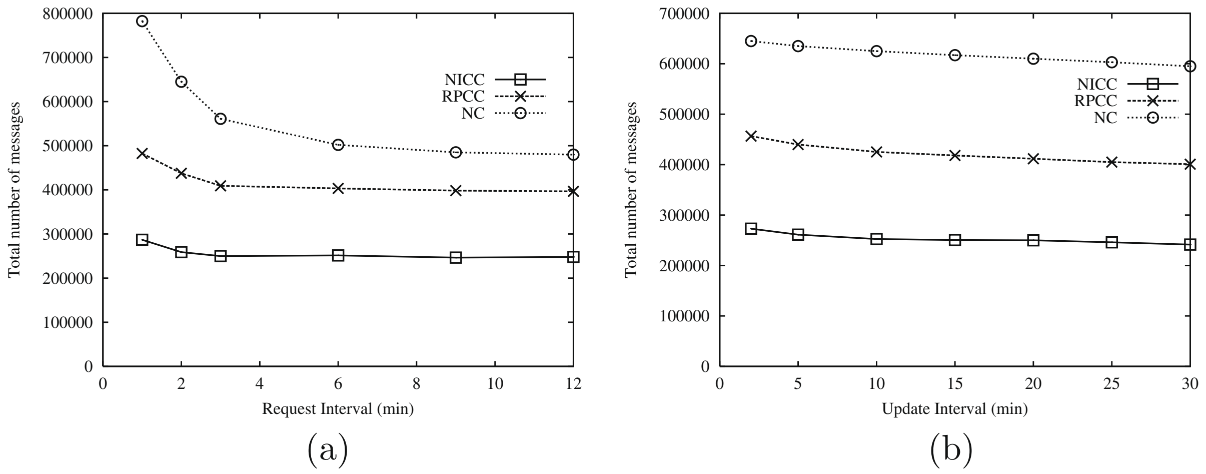


Fig. 12. Network traffic (MB-sized files) according to: (a) request interval, (b) update interval for a WMSN with 100 sensors.

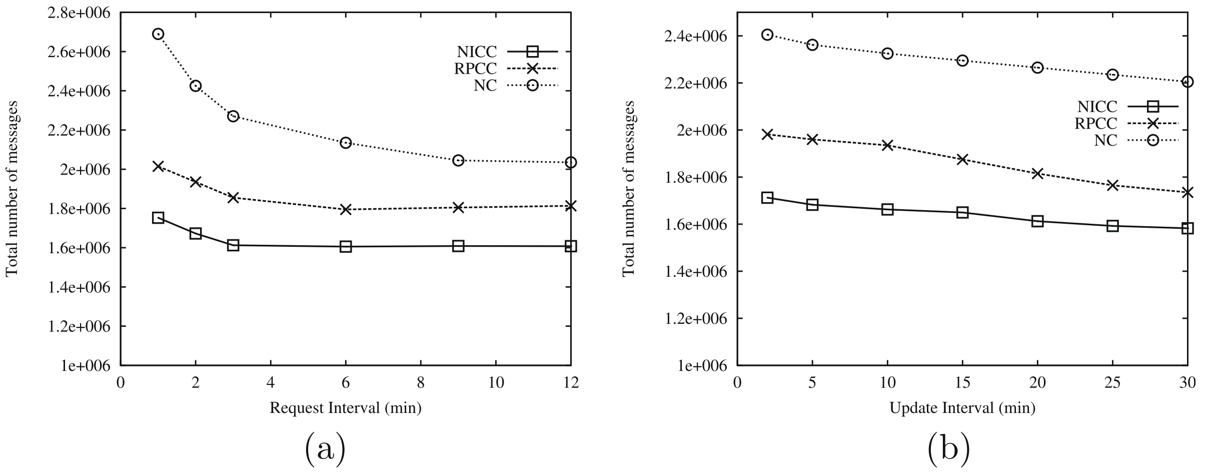


Fig. 13. Network traffic (MB-sized files) according to: (a) request interval, (b) update interval for a WMSN with 500 sensors.

to a heavy communication overhead. We can also observe that the request interval and update interval did not affect the performance of both protocols, except from the case of small values.

From the obtained results concerning the number of hits and transmitted messages, we draw the conclusion that the proposed protocol is more energy efficient than its competitor. To validate this conclusion we plotted the relative difference between the amount of consumed energy for each sensor node of the WMSN after the execution of *NICC* and after the execution of RPCC, taking averages after several runs of the simulation. The results are depicted in Figs. 14 and 15. It is indicative that there is not any single sensor which runs *NICC* with less remaining energy than the respective energy it has after the completion of RPCC. In *NICC* each sensor node elects the more significant nodes of its two-hop neighborhood according to its local network view. Therefore, nearby nodes may elect different nodes as mediator nodes. Mediator nodes election process depends on sensor nodes that originate and broadcast a message. Thus, energy consumption of sensor nodes is significantly less intense in *NICC* than by RPCC, since only mediator nodes rebroadcast an invalidation report and mediator nodes lie in 'central' points of network topology. Additionally, due to the distributed election of mediator nodes by each sensor node, the *NICC* protocol achieves a balanced energy consumption among sensor nodes, which is a desirable goal in protocol design according to [43] if we want to prolong the sensor network lifetime. *NICC* avoids the drawback of a centralized static election process of mediator nodes that will lead in a fast battery depletion of mediator nodes.

Moreover, for both cases, that of KB-sized items (Figs. 26 and 27) and MB-sized items, we observe that the energy gains of *NICC* are greater in the case of dense network topology, since the selection of mediator nodes is more accurate. Finally, all figures attest that the RPCC causes a big energy depletion of sensor nodes. Such situation is

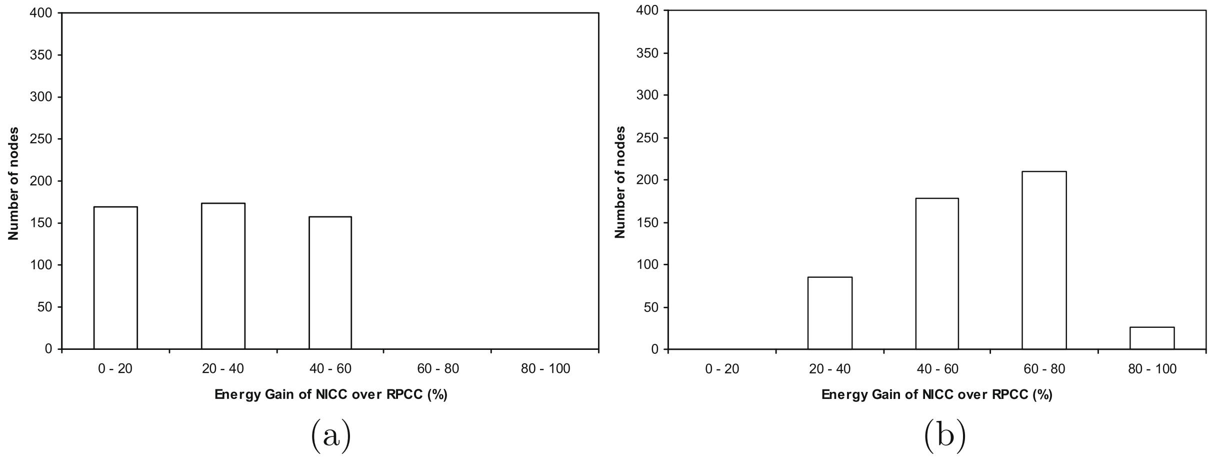


Fig. 14. % difference of energy consumption (MB-sized files) in: (a) sparse ($d = 4$), (b) dense ($d = 10$) WMSN with 500 sensors when it executes *NICC* w.r.t. when it executes RPCC.

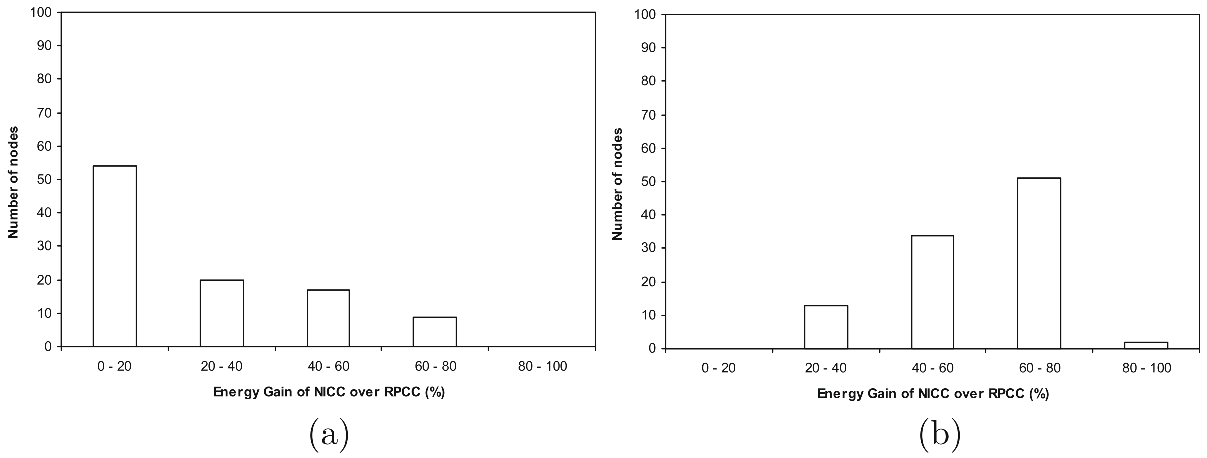


Fig. 15. % difference of energy consumption (MB-sized files) in: (a) sparse ($d = 4$), (b) dense ($d = 10$) WMSN with 100 sensors when it executes *NICC* w.r.t. when it executes RPCC.

undesirable, since may cause network partition and thus make the WMSN useless.

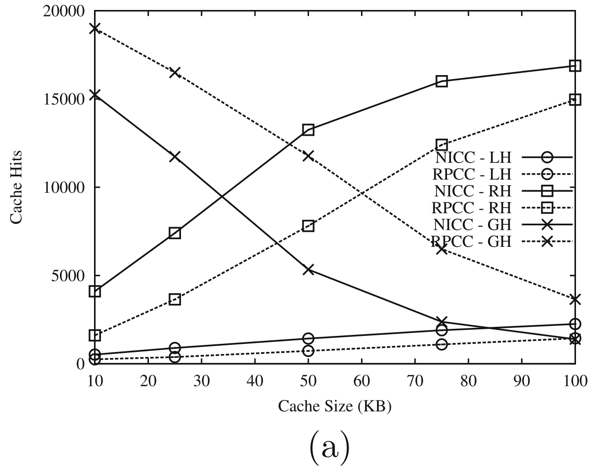
5.3.2. Experiments with small data items

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 it is considered that WMSNs will deal with 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 cache coherence 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; *NICC* achieves significantly smaller number of global hits and larger number of remote hits than *RPCC* does. It is not worthy to comment on each individual performance graph (Figs. 16–27), since in all cases *NICC* has a better performance; it achieves again 25% more remote hits and 50% less global hits than RPCC. The results also in terms of average query latency and number of messages are almost the same.

5.3.3. Impact of TTL on system performance

Each cache node can reach immediately a mediator node if there are many mediator nodes in the network. This implies that the network traffic between source node and mediator nodes will increase, since a great number of invalidation messages have to be disseminated in the network. However, the communication overhead between mediator nodes and cache nodes will be reduced. The number of hops that a request has to passed through, in order to reach a mediator node, will be minimized. On the contrary, if the number of mediator nodes is small, we will achieve a reduced communication overhead between source nodes and mediator nodes, but the communication



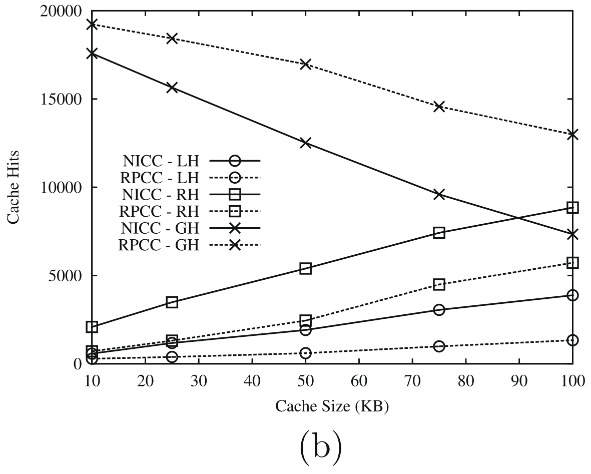
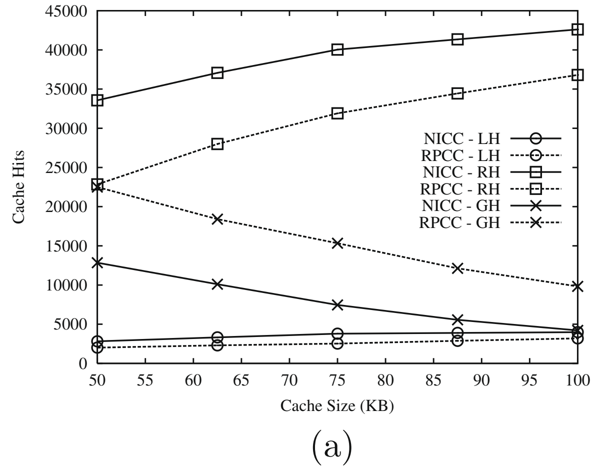


Fig. 16. Impact of sensor cache size on hits (KB-sized files) in: (a) sparse ($d = 4$), (b) dense ($d = 10$) WMSN with 100 sensors.



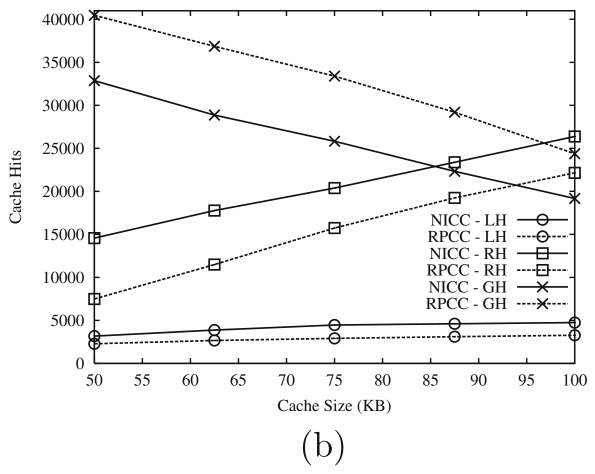
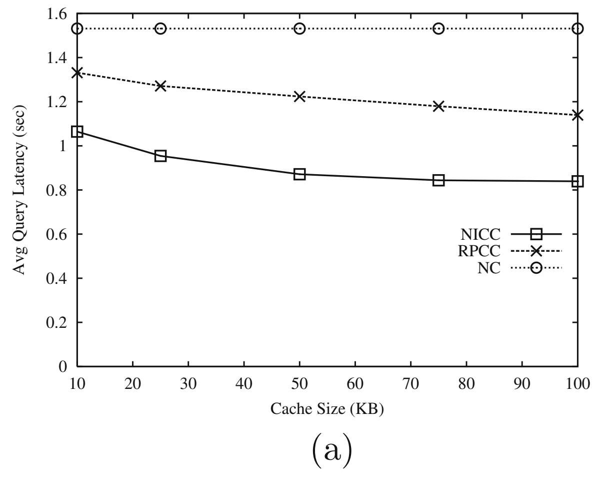


Fig. 17. Impact of sensor cache size on hits (KB-sized files) in: (a) sparse ($d = 4$), (b) dense ($d = 10$) WMSN with 500 sensors.



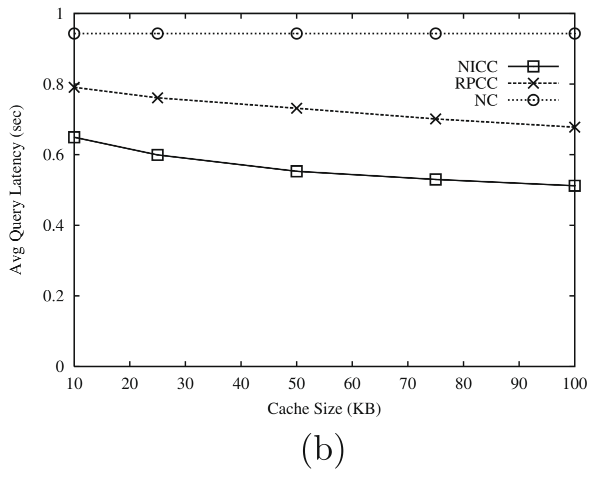


Fig. 18. Impact of cache size on latency (KB-sized files) in: (a) sparse ($d = 4$), (b) dense ($d = 10$) WMSN with 100 sensors.

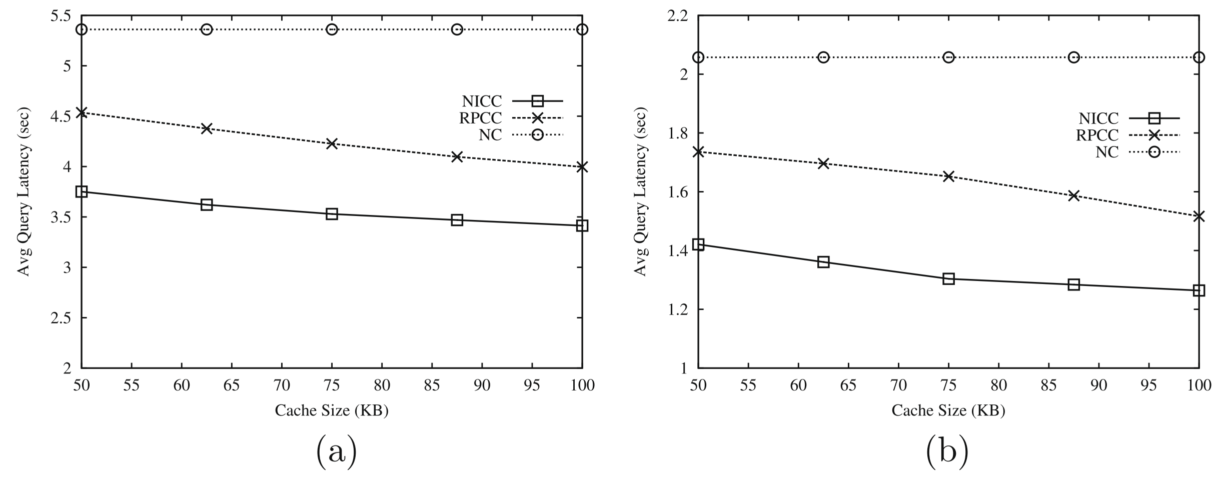


Fig. 19. Impact of cache size on latency (KB-sized files) in: (a) sparse ($d = 4$), (b) dense ($d = 10$) WMSN with 500 sensors.

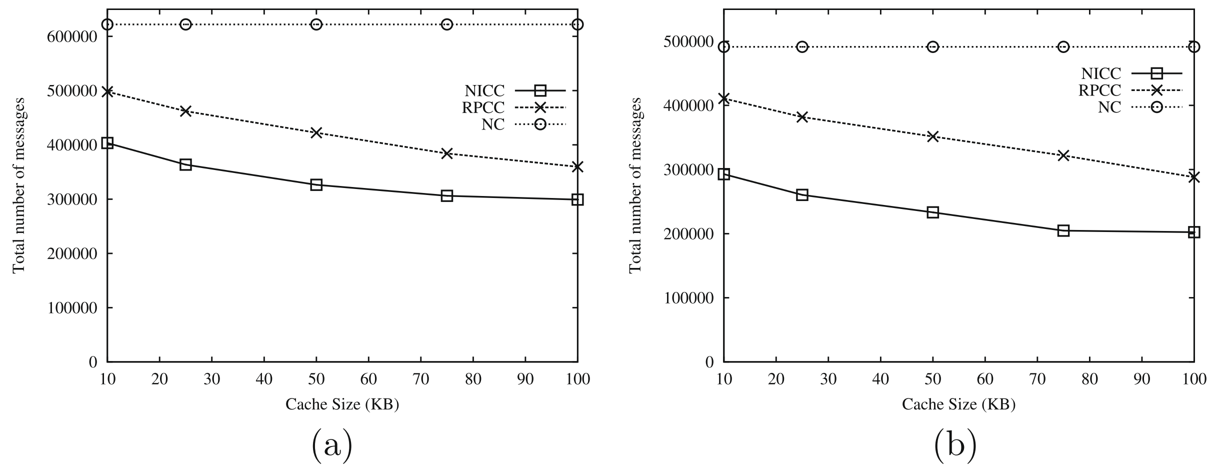


Fig. 20. Impact of cache size on number of messages (KB-sized files) in: (a) sparse ($d = 4$), (b) dense ($d = 10$) WMSN with 100 sensors.

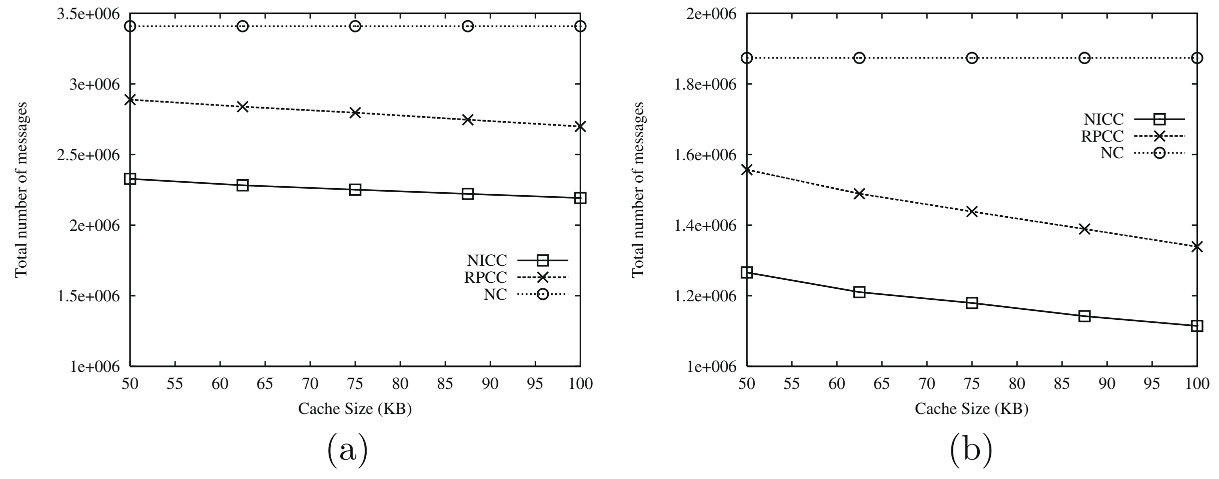


Fig. 21. Impact of cache size on number of messages (KB-sized files) in: (a) sparse ($d = 4$), (b) dense ($d = 10$) WMSN with 500 sensors.

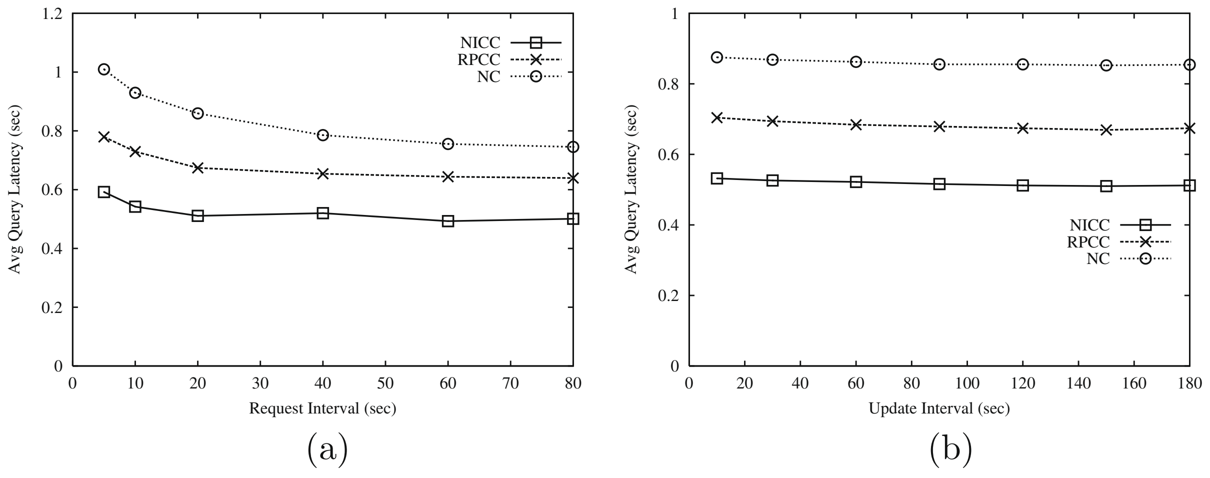


Fig. 22. Query latency (KB-sized files) according to: (a) request interval, (b) update interval for a WMSN with 100 sensors.

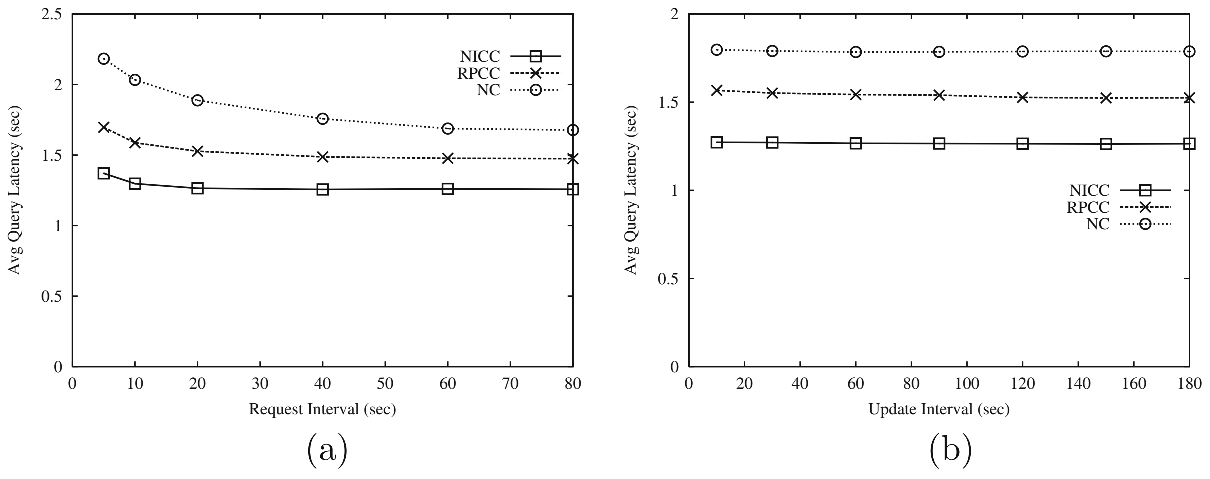


Fig. 23. Query latency (KB-sized files) according to: (a) request interval, (b) update interval for a WMSN with 500 sensors.

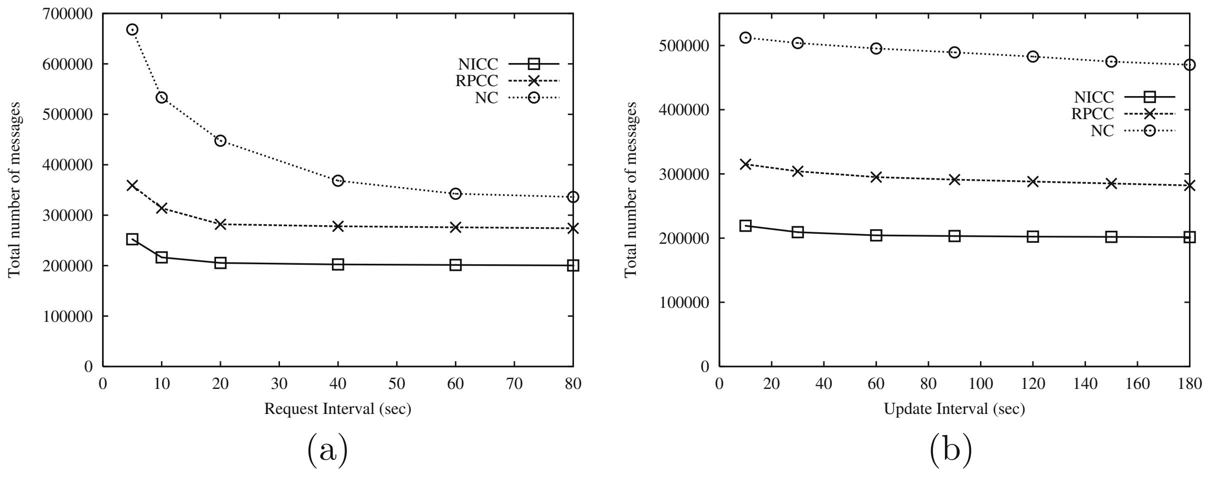


Fig. 24. Network traffic (KB-sized files) according to: (a) request interval, (b) update interval for a WMSN with 100 sensors.

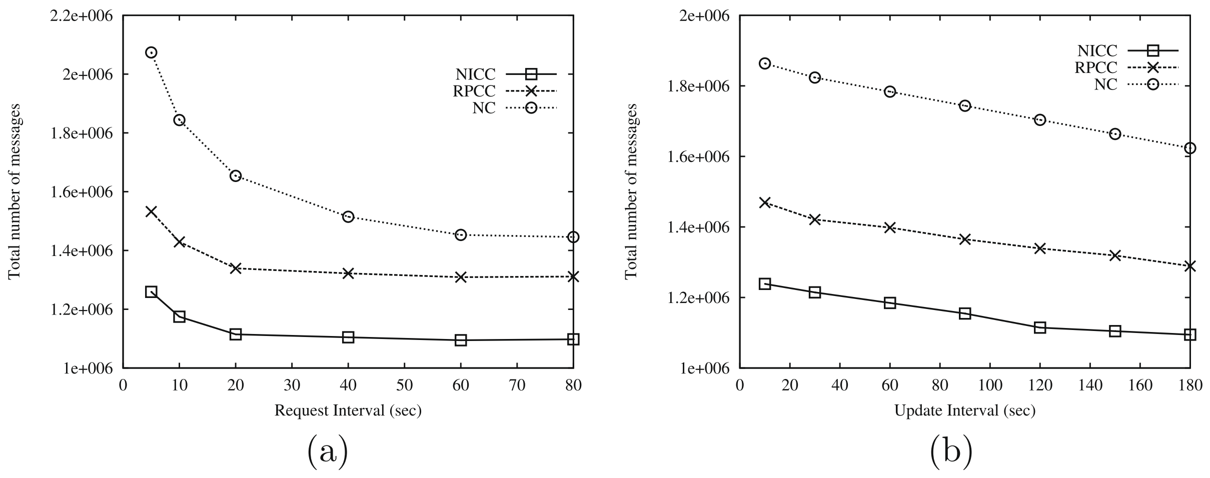


Fig. 25. Network traffic (KB-sized files) according to: (a) request interval, (b) update interval for a WMSN with 500 sensors.

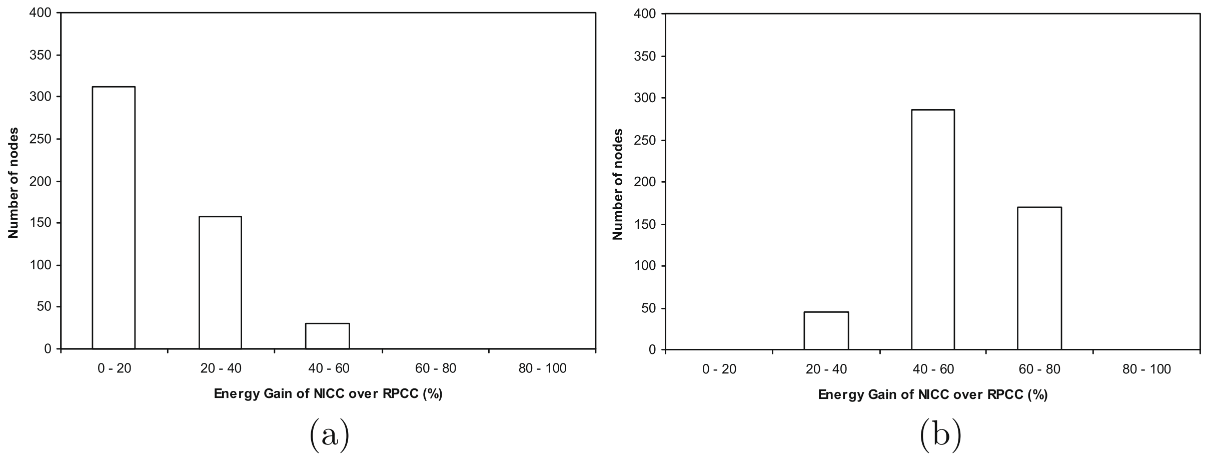


Fig. 26. % difference of energy consumption (KB-sized files) in: (a) sparse ($d = 4$), (b) dense ($d = 10$) WMSN with 500 sensors when it executes *NICC* w.r.t. when it executes RPCC.

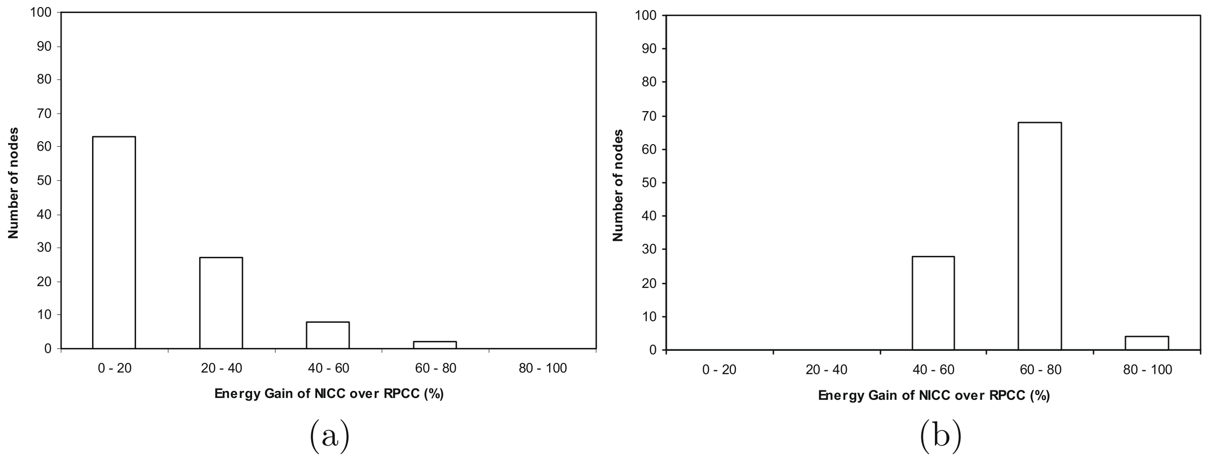


Fig. 27. % difference of energy consumption (KB-sized files) in: (a) sparse ($d = 4$), (b) dense ($d = 10$) WMSN with 100 sensors when it executes *NICC* w.r.t. when it executes RPCC.

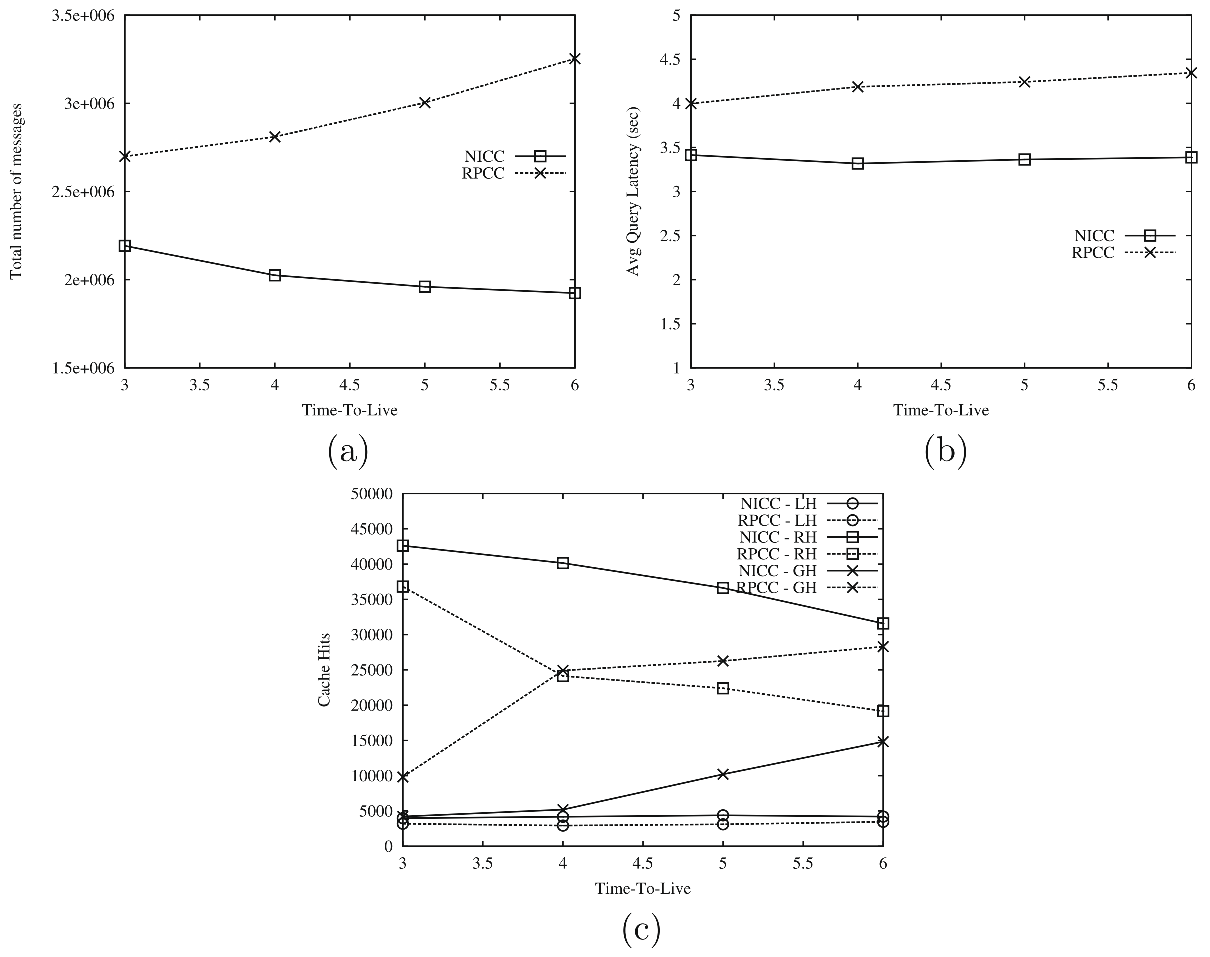


Fig. 28. Impact of TTL value on: (a) number of messages, (b) latency, (c) number of hits in WMSN with 500 sensors.

overhead between requester nodes and mediator nodes will be enlarged.

We vary the TTL value from 3 to 6 hops. The results are depicted in Fig. 28. *NICC* has a slightly better performance in terms of number of messages as the TTL value increases. However, the query latency remains almost the same in all cases. On the contrary, in RPCC, the number of messages increases significantly with increasing TTL value. Also, the query latency increases according to TTL value.

5.3.4. Cache replacement experiments

Cache replacement policy is a significant component of the *NICC* protocol. An effective and efficient replacement policy is requisite to deal with the huge amount of data generated in sensor networks. In these set of experiments, the cache replacement policy discussed in Section 4.4 is compared with GD-Size replacement policy [44], which is selected as the best replacement policy with *proven* (on-line) optimality.

Figs. 29 and 30 present the impact of cache replacement policies on latency. The proposed policy *NICC* outperforms the GD-Size policy (denoted as NICC_GDS) for all cache sizes. The first reason of the performance gains of the proposed policy is the parameter *AvgHops* that is included in the replacement function during the calculation of the utility value for a data item. As the distance between the requester node and the responding node increases, the cached data item is more possible to remain in the cache. Thus, the proposed policy attempts to keep in cache the data items that are requested by nodes that are further away. Caching these data items reduce latency, energy dissipation and save bandwidth for subsequent requests about the same items. The second reason is that GD-Size did not take into account the update rate and the popularity of a data item. The proposed policy favors data items that is not modified frequently and have a big access rate.

Figs. 31 and 32 evaluate the performance in terms of number of hits. We observe that the proposed policy achieves better results in local, remote and global cache hits as compared to those with GD-Size. This is because the proposed policy favors the remotely requested, small data items with high popularity. Additionally, the GD-Size policy did not take into account the update rate of a data item.

Finally, it is very interesting to present the difference between sparse and dense networks. In sparse networks (Fig. 29a, Fig. 30a, Fig. 31a, Fig. 32a) the performance gains

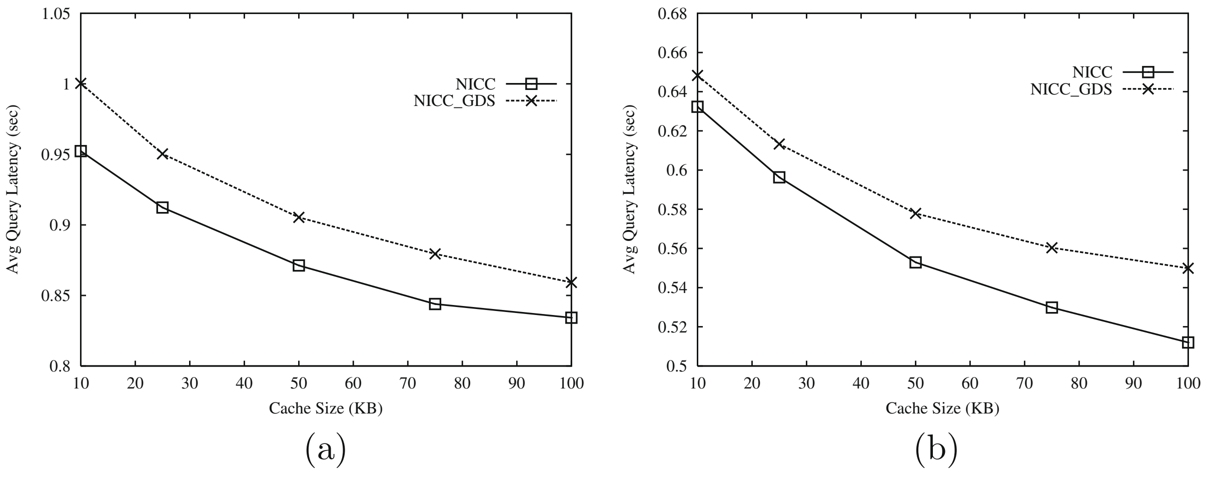


Fig. 29. Impact of cache replacement on latency in: (a) sparse $(d = 4)$, (b) dense $(d = 10)$ WMSN with 100 sensors.

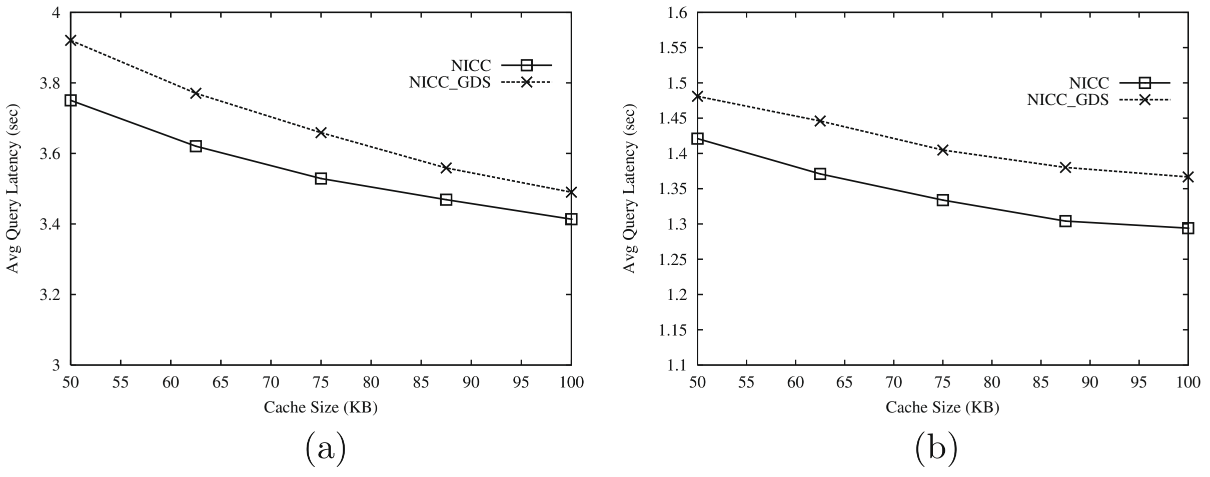


Fig. 30. Impact of cache replacement on latency in: (a) sparse $(d = 4)$, (b) dense $(d = 10)$ WMSN with 500 sensors.

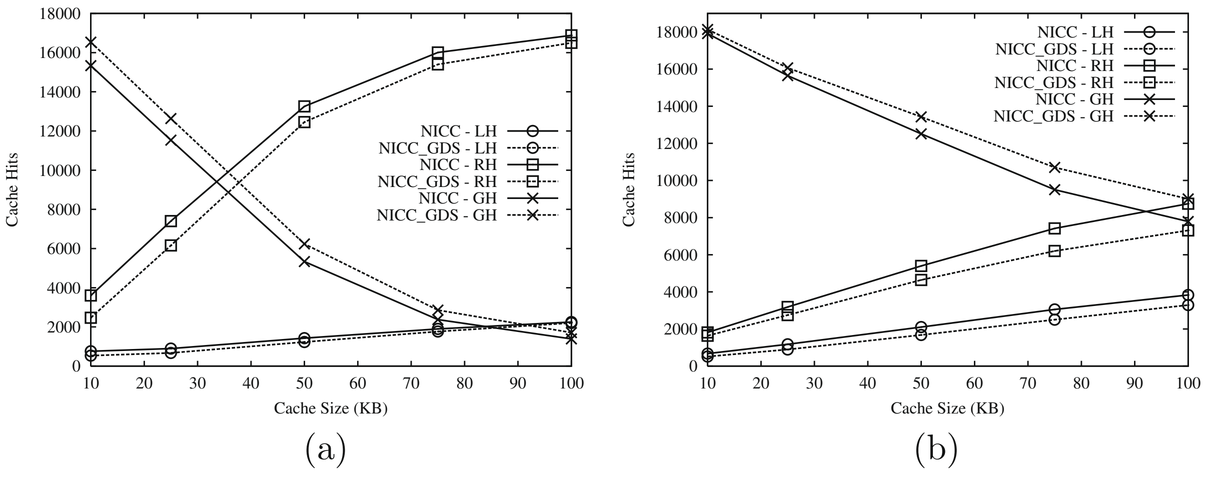


Fig. 31. Impact of cache replacement on hits in: (a) sparse $(d = 4)$, (b) dense $(d = 10)$ WMSN with 100 sensors.

of the proposed policy decrease as the cache size increases. This is because for large values of cache size the cache replacements reduce significantly, since there is enough cache space for many data items. Additionally, in sparse networks the number of neighboring nodes is small and thus the number of data items to be cached is also small. However in dense networks (Fig. 29b, Fig. 30b, Fig. 31b, Fig. 32b) the proposed policy performs better then GD-Size when the cache size increases. This is due to the large number of neighboring nodes. When the cache size is very small, the replacements of data items are very often, while the candidate data items to be cached are significantly more than those in sparse networks. Thus, the local and remote cache hits decrease while the global hits increase.

5.3.5. *Impact of network density and cache size on number of replicas and mediators*

In these set of experiments we examined the impact of cache size and node degree on the number of mediator nodes and replicas for *NICC* protocol. Fig. 33 presents the average number of replicas of a data item that have been cached in mediator nodes. When the cache space is small the number of replicas of a data item is also small. This is because the size of cache cannot allow a huge number of replicas to be cached. Additionally, when the cache size increases we observe that the number of replicas is also increases. However, varying the levels of node degree we detect a different system performance. This is due to the number of neighboring nodes. When the number of neighboring nodes is big, the number of nearby mediator nodes increases. Thus, the number of replicas that are pushed in mediator nodes is also big.

Fig. 34 illustrates the average number of replicas that have been cached in each mediator node. It is obvious that with increasing values of node degree and cache space, the number of cached replicas in mediator nodes is also augmented. This is because each mediator node has much more neighboring nodes and therefore it receives many replicas to be cached. Thus, each mediator node exploit better its cache space by caching more data items.

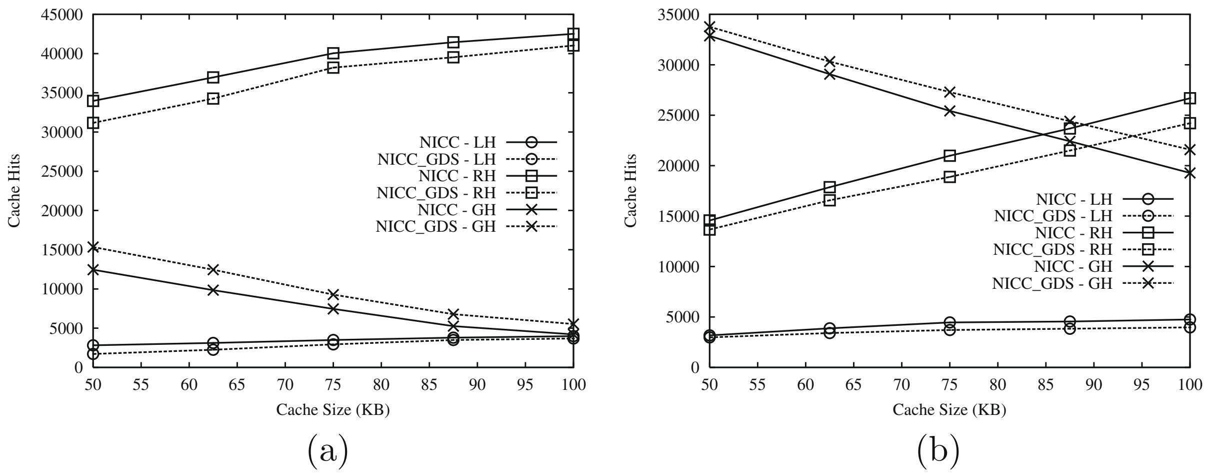


Fig. 32. Impact of cache replacement on hits in: (a) sparse ($d = 4$), (b) dense ($d = 10$) WMSN with 500 sensors.

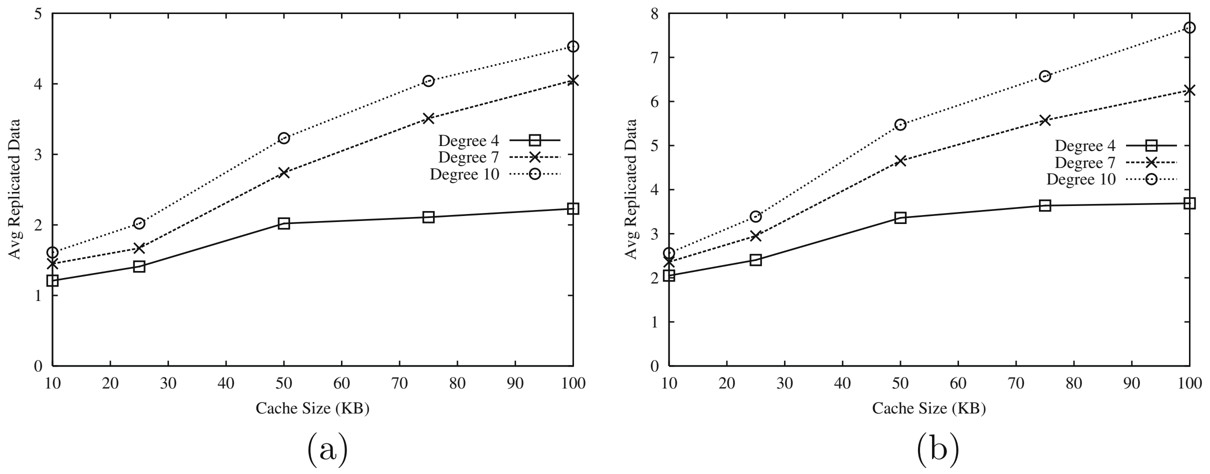


Fig. 33. Average number of replicas per data item for various degrees in a WMSN with: (a) 100, (b) 500 sensors.

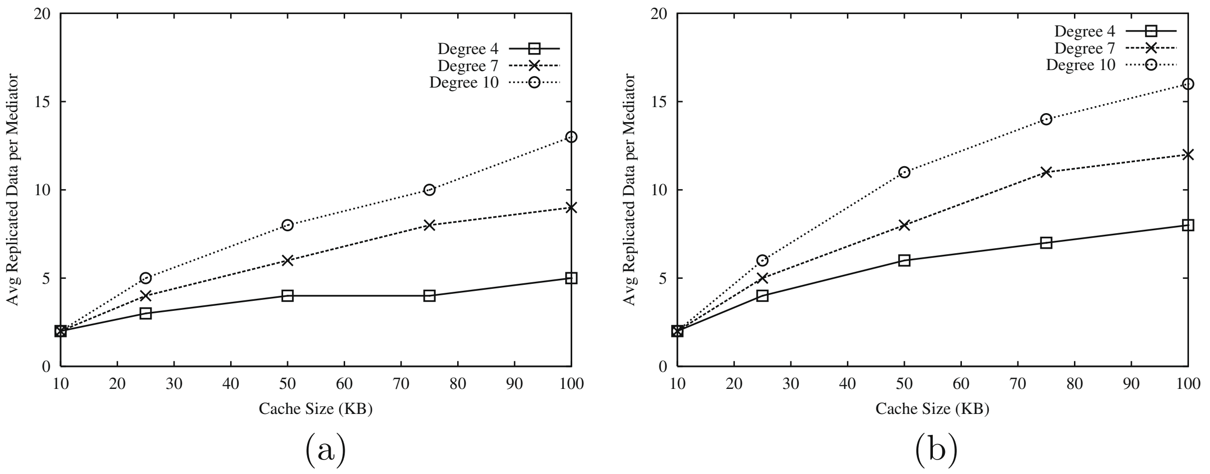


Fig. 34. Average number of replicas per mediator node for various degrees in a WMSN with: (a) 100, (b) 500 sensors.

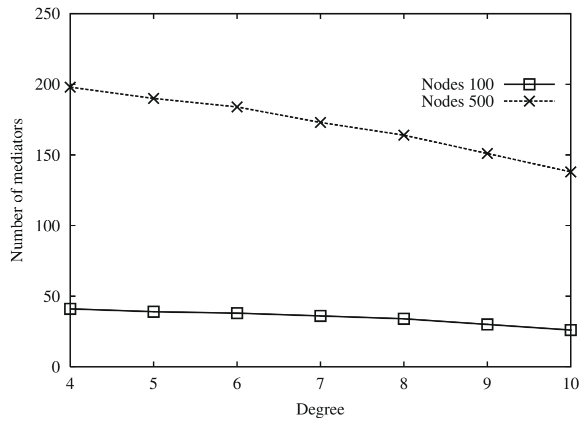


Fig. 35. Average number of mediator nodes for various degrees in a WMSN with 100 and 500 sensors.

Finally, in Fig. 35 we examine the average number of mediator nodes that are generated in the entire sensor network. We observe that for sparse sensor networks the number of mediator nodes is big enough. However, for dense sensor networks the number of mediator nodes is decreased between 25% and 30% of the entire set of sensor nodes. This is because many sensor nodes detect the same nodes as important.

Even though all the aforementioned results concern the AODV routing protocol with an IEEE 802.11 MAC layer, the relative performance results still holds in different protocol/layer configurations. To test this argument we performed a few more experiments: In the first set of experiments we experimented with the AODV routing protocol with an IEEE 802.15.4 as the MAC layer, and in a couple of experiments we investigated the GPSR protocol with an IEEE 802.11 MAC layer.

We investigated the impact of cache size on the number of hits, latency incurred and transmitted messages for the two protocols, but, with the IEEE 802.15.4 as the MAC

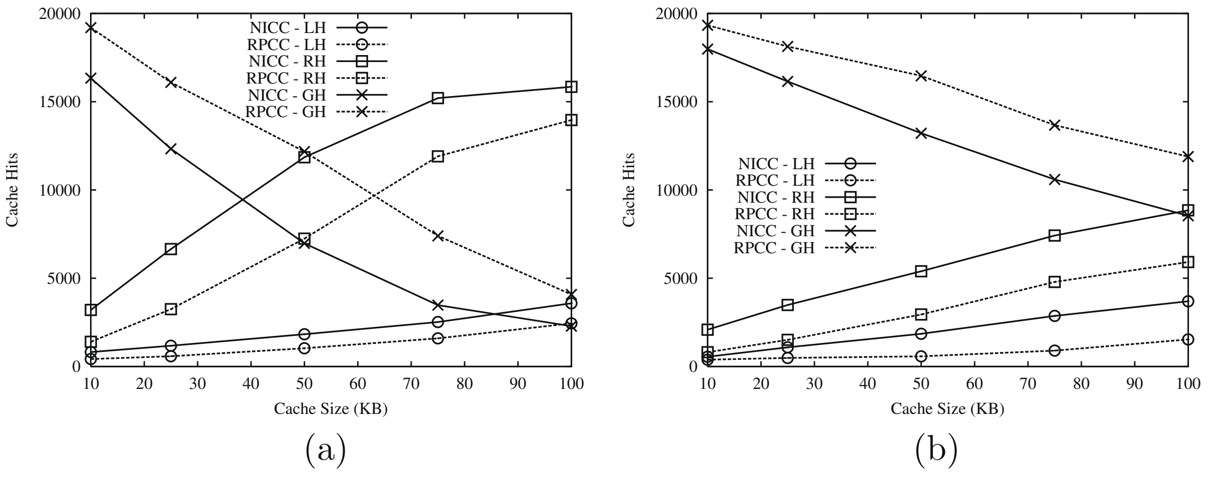


Fig. 36. Impact of sensor cache size on hits (KB-sized files) in: (a) sparse ($d = 4$), (b) dense ($d = 10$) WMSN with 100 sensors and 802.15.4 MAC protocol.

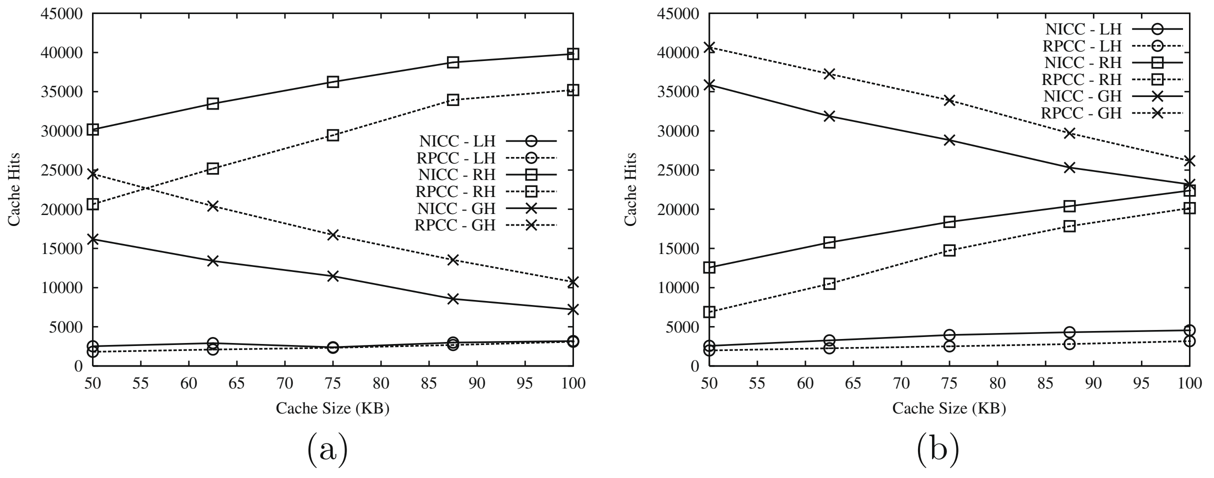


Fig. 37. Impact of sensor cache size on hits (KB-sized files) in: (a) sparse ($d = 4$), (b) dense ($d = 10$) WMSN with 500 sensors and 802.15.4 MAC protocol.

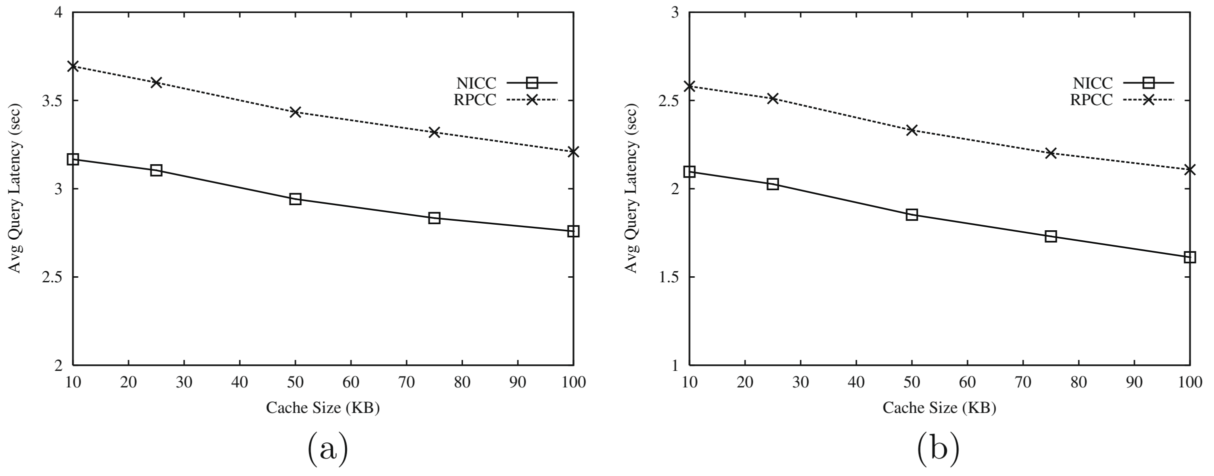


Fig. 38. Impact of cache size on latency (KB-sized files) in: (a) sparse ($d = 4$), (b) dense ($d = 10$) WMSN with 100 sensors and 802.15.4 MAC protocol.

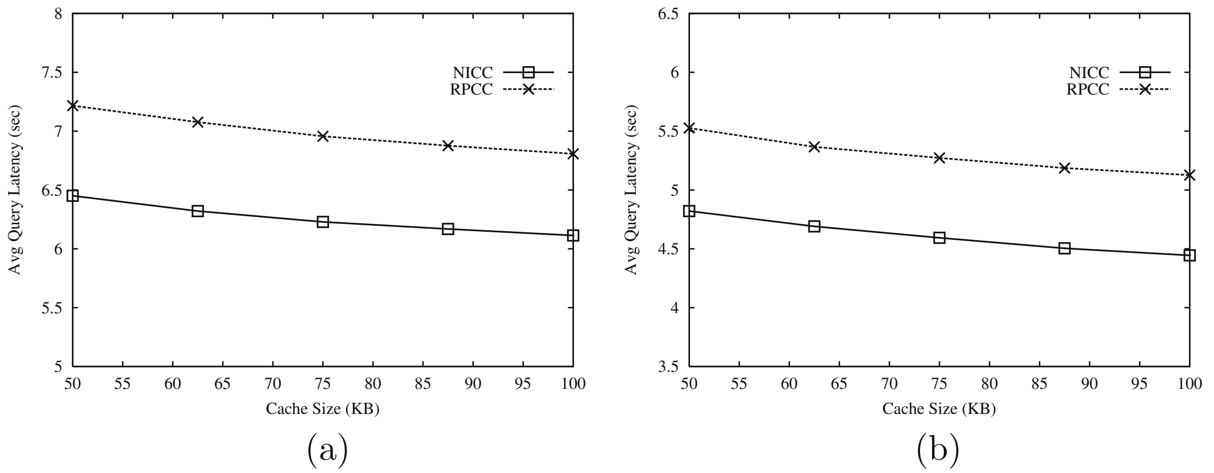


Fig. 39. Impact of cache size on latency (KB-sized files) in: (a) sparse ($d = 4$), (b) dense ($d = 10$) WMSN with 500 sensors and 802.15.4 MAC protocol.

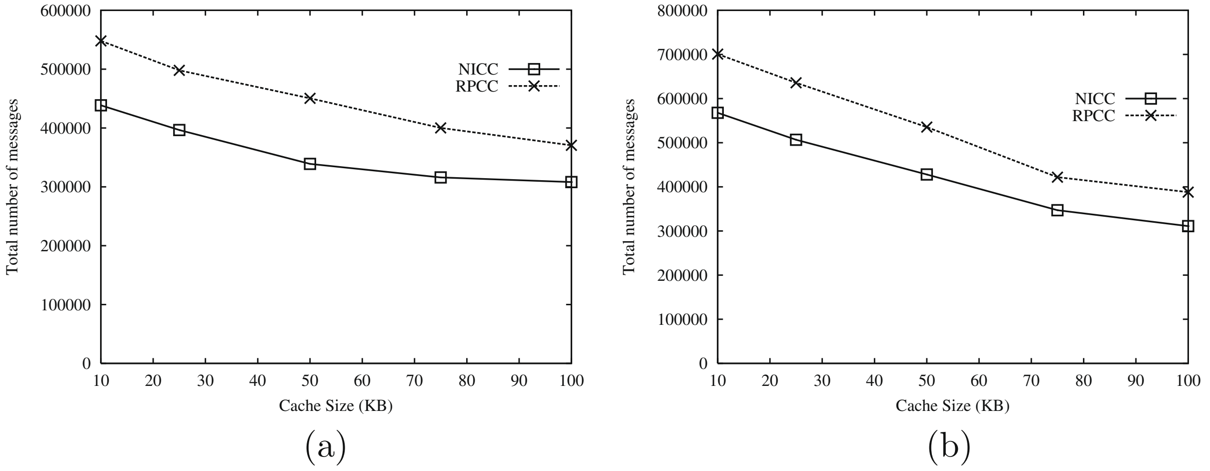


Fig. 40. Impact of cache size on number of messages (KB-sized files) in: (a) sparse ($d = 4$), (b) dense ($d = 10$) WMSN with 100 sensors and 802.15.4 MAC protocol.

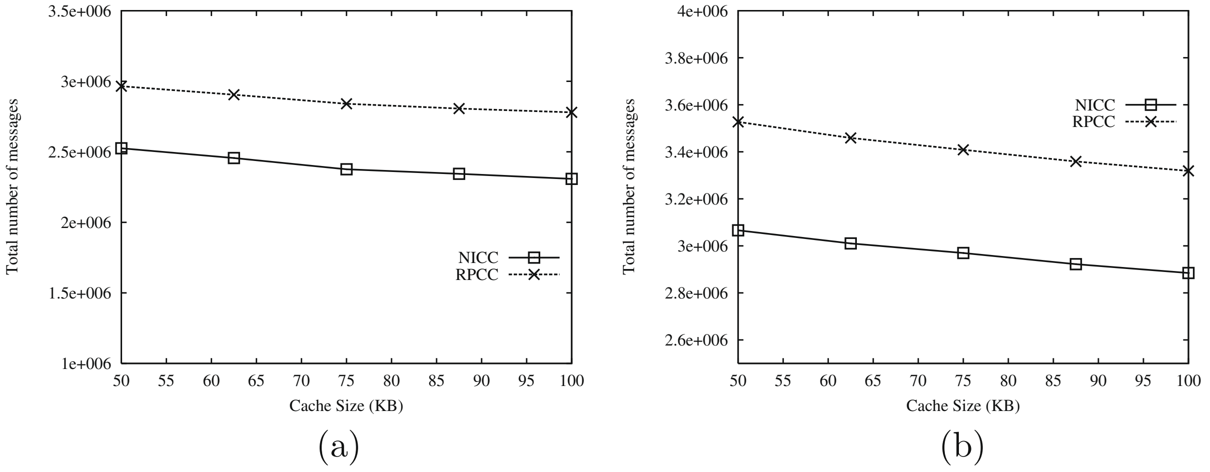


Fig. 41. Impact of cache size on number of messages (KB-sized files) in: (a) sparse ($d = 4$), (b) dense ($d = 10$) WMSN with 500 sensors and 802.15.4 MAC protocol.

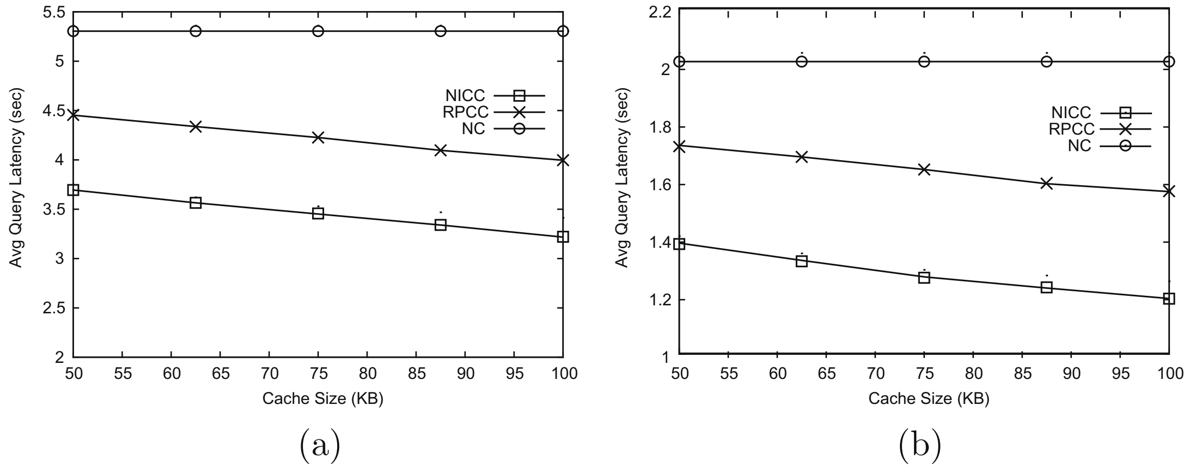


Fig. 42. Impact of cache size on latency (KB-sized files) in: (a) sparse ($d = 4$), (b) dense ($d = 10$) WMSN with 500 sensors using the GPSR protocol.

layer. The results are illustrated in Figs. 36–41 (they are the analogous of Figs. 16–21). Although there exists differences in the absolute values of the measured quantities because this MAC supports much lower bit-rates, the relative performance of the protocols and the generic trends remain unchanged. Therefore, there is no need to comment in details the performance of the algorithms, because the observations made earlier still hold for this MAC layer. The investigation of the impact of the cache size on the incurred latency when using the GPSR routing protocol in a sensornet with 500 nodes is depicted in Fig. 42. There is no noticeable difference in the performance of the cooperative cache consistency algorithms, which is expected since the cooperation scheme is the major factor that affects the performance, and not the routing scheme.

In conclusions, in all the conducted experiments, the NICC protocol proved superior to its competitor; the main reason for that is the wiser selection of the nodes that will play the role of the coordinator in the caching decisions. NICC takes into consideration the position of the sensor nodes, whereas RPCC makes a 'blind' selection.

6. Discussion

The recent advances in miniaturization, the creation of low-power circuits, and the development of cheap CMOS cameras and microphones gave birth to *Wireless Multimedia Sensor Networks*. WMSNs are expected to fuel many new applications and boost the already existing ones. Based on the recent progress in the design and development of NAND flash storage devices, which are expected to provide almost unlimited storage capacity to sensor nodes, this article identified cooperative caching as a vital solution for achieving both target of WMSNs, and proposed a cache consistency protocol, the *NICC* protocol, suitable 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 somehow coordinate the cache refresh decisions. The proposed protocol is evaluated with J-Sim and its performance is compared to that of a state-of-the-art cooperative cache consistency 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 30%, increase the remote hits due to the more effective sensor cooperation and cache refresh policy at an average percentage of about 25%, and provide better energy management than its competitor.

Acknowledgment

Research supported by the project "Control for Coordination of Distributed Systems", funded by the EU.ICT program, Challenge ICT-2007.3.7.

References

[1] H. Karl, A. Willig, Protocols and Architectures for Wireless Sensor Networks, John Wiley & Sons, 2006.
[2] M. Rahimi, R. Baer, O.I. Iroezi, J.C. Garcia, J. Warrior, D. Estrin, M. Srivastava, Cyclops: in situ image sensing and interpretation in wireless sensor networks, in: Proceedings of the ACM International Conference on Embedded Networked Sensor Systems (SenSys), 2005, pp. 192–204.
[3] I. Akyildiz, T. Melodia, K.R. Chowdhury, A survey of wireless multimedia sensor networks, Computer Networks 51 (2007) 921–960.
[4] B. Girod, M. Kalman, Y.J. Liang, R. Zhang, Advances in channel-adaptive video streaming, Wireless Communications and Mobile Computing 2 (6) (2002) 573–584.
[5] Y. Eisenberg, C.E. Luna, T.N. Pappas, R. Berry, A.K. Katsaggelos, Joint source coding and transmission power management for energy efficient wireless video communications, IEEE Transactions on Circuits and Systems for Video Technology 12 (6) (2002) 411–424.
[6] J. Liu, J. Xu, Proxy caching for media streaming over the Internet, IEEE Communications magazine 42 (8) (2004) 88–94.
[7] A.T.S. Ip, J.C.S. Lui, J. Liu, A revenue-rewarding scheme for cooperative proxy media streaming systems, ACM Transactions on Multimedia Computing, Communications and Applications 4 (1) (2008). Article No. 5.
[8] G. Mathur, P. Desnoyers, D. Ganesan, P. Shenoy, Ultra-low power data storage for sensor networks, in: Proceedings of the ACM International Conference on Information Processing in Sensor Networks (IPSN), 2006, pp. 374–381.
[9] S. Nath, A. Kansal, FlashDB: dynamic self-tuning database for NAND flash, in: Proceedings of the ACM International Conference on Information Processing in Sensor Networks (IPSN), 2007, pp. 410–419.
[10] C.-H. Wu, T.-W. Kuo, L.-P. Chang, An efficient B-tree layer implementation for flash-memory storage systems, ACM Transactions on Embedded Computing Systems 6 (3) (2007). Article 19.
[11] S. Tilak, N. Abu-Ghazaleh, W.B. Heinzelman, Storage management in wireless sensor networks, in: Mobile Wireless and Sensor Networks: Technology, Applications and Future Directions, IEEE/Wiley, 2006, pp. 257–281.
[12] A. Sobeih, J.C. Hou, L.-C. Kung, N. Li, H. Zhang, W.-P. Chen, H.-Y. Tyan, H. Lim, J-Sim: a simulation and emulation environment for wireless sensor networks, IEEE Wireless Communications Magazine 13 (4) (2006) 104–119.
[13] J. Cao, Y. Zhang, G. Cao, Data consistency for cooperative caching in mobile environments, IEEE Computer (2007) 60–66.
[14] D. Barbara, T. Imielinski, Sleepers and workaholics: caching strategies in mobile environments, The VLDB Journal 4 (4) (1995) 567–602.
[15] P. Cao, C. Liu, Maintaining strong cache consistency in the World Wide Web, IEEE Transactions on Computers 47 (4) (1998) 445–457.
[16] M.J. Franklin, M.J. Carey, M. Livny, Transactional client-server cache consistency: alternatives and performance, ACM Transactions on Database Systems 22 (3) (1997) 315–363.
[17] K.-L. Tan, J. Cai, B.C. Ooi, An evaluation of cache invalidation strategies in wireless environments, IEEE Transactions on Parallel and Distributed Systems 12 (8) (2001) 789–807.
[18] H. Hayashi, T. Hara, S. Nishio, Updated data dissemination methods for updating old replicas in ad hoc networks, Personal and Ubiquitous Computing 9 (5) (2005) 273–283.
[19] J. Cao, Y. Zhang, G. Cao, Consistency of cooperative caching in mobile peer-to-peer systems over MANET, in: Proceedings of the International Conference on Distributed Computing Systems Workshops (ICDCSW), 2005, pp. 573–579.
[20] H. Shen, M.S. Joseph, M. Kumar, S.K. Das, PReCinCt: a scheme for cooperative caching in mobile peer-to-peer systems, in: Proceedings of the International Parallel and Distributed Processing Symposium (IPDPS), 2005.
[21] N. Dimokas, D. Katsaros, Y. Manolopoulos, Cooperative caching in wireless multimedia sensor networks, in: Proceedings of the ACM SIGMM International Mobile Multimedia Communications Conference (MobiMedia), 2007, pp. 377–382.
[22] N. Dimokas, D. Katsaros, L. Tassiulas, Y. Manolopoulos, High performance, low complexity cooperative caching for wireless sensor networks, in: Proceedings of the IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2009.
[23] T. Hara, Cooperative caching by mobile clients in push-based information systems, in: Proceedings of the ACM International Conference on Information and Knowledge Management (CIKM), 2002, pp. 186–193.
[24] H. Shen, S.K. Das, M. Kumar, Z. Wang, Cooperative caching with optimal radius in hybrid wireless networks, in: Proceedings of the

International IFIP-TC6 Networking Conference (NETWORKING), Lecture Notes on Computer Science, vol. 3042, 2004, pp. 841–853.
[25] L. Yin, G. Cao, Supporting cooperative caching in ad hoc networks, IEEE Transactions on Mobile Computing 5 (1) (2006) 77–89.
[26] W. Li, E. Chan, D. Chen, Energy-efficient cache replacement policies for cooperative caching in mobile ad hoc network, in: Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC), 2007, pp. 3349–3354.
[27] C.-Y. Chow, H.V. Leong, A.T.S. Chan, GroCoca: group-based peer-to-peer cooperative caching in mobile environment, IEEE Journal on Selected Areas in Communications 25 (1) (2007) 179–191.
[28] N. Chand, R.C.R.C. Joshi, M. Misra, Cooperative caching strategy in mobile ad hoc networks based on clusters, Wireless Personal Communications 43 (1) (2006) 41–63.
[29] J. Mena, V. Kalogeraki, Dynamic relay node placement in wireless sensor networks, in: Proceedings of the IEEE International Symposium on Applications and the Internet (SAINT), 2008, pp. 8–17.
[30] K.S. Prabh, T.F. Abdelzaher, Energy-conserving data cache placement in sensor networks, ACM Transactions on Sensor Networks 1 (2) (2005) 178–203.
[31] B. Tang, H. Gupta, Cache placement in sensor networks under an update cost constraint, Journal of Discrete Algorithms 5 (3) (2007) 422–435.
[32] B. Tang, H. Gupta, S.R. Das, Benefit-based data caching in ad hoc networks, IEEE Transactions on Mobile Computing 7 (3) (2008) 289–304.
[33] W. Zhang, G. Cao, Defending against cache consistency attacks in wireless ad hoc networks, Ad Hoc Networks 6 (3) (2008) 363–379.
[34] S. Lim, W.-C. Lee, G. Cao, C.R. Das, A novel caching scheme for improving internet-based mobile ad hoc networks performance, Ad Hoc Networks 4 (2) (2006) 225–239.
[35] S.M. Das, H. Pucha, Y.C. Hu, Mitigating the gateway bottleneck via transparent cooperative caching in wireless mesh networks, Ad Hoc Networks 5 (6) (2007) 680–703.
[36] N. Loulloudes, G. Pallis, M.D. Dikaiakos, Information dissemination in mobile CDNs, in: R. Buyya, A.K. Pathan, A. Vakali (Eds.), Content Delivery Networks: Principles and Paradigms, Springer, 2008.
[37] W.S.K. Faust, Social Network Analysis: Methods and Applications, Structural Analysis in the Social Sciences, Cambridge University Press, 2005.
[38] D. Katsaros, Y. Manolopoulos, The geodesic broadcast scheme for wireless ad hoc networks, in: Proceedings of the IEEE International Symposium on a World of Wireless Mobile Multimedia (WoWMoM), 2006, pp. 571–575.
[39] G. Zhou, T. He, S. Krishnamurthy, J. Stankovic, Models and solutions for radio irregularity in wireless sensor networks, ACM Transactions On Sensor Networks 2 (2) (2006) 221–262.
[40] C.E. Perkins, E. Royer, Ad hoc on-demand distance vector routing, in: Proceedings of the IEEE Workshop on Mobile Computing Systems and Applications, 1999, pp. 90–100.
[41] B. Karp, H.T. Kung, GPSR: greedy perimeter stateless routing for wireless networks, in: Proceedings of the ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom), 2000.
[42] I. Stojmenovic, X. Lin, Loop-free hybrid single-path/flooding routing algorithms with guaranteed delivery for wireless networks, IEEE Transactions on Parallel and Distributed Systems 12 (10) (2001) 1023–1032.
[43] J.H. Chang, L. Tassiulas, Energy conserving routing in wireless ad-hoc networks, in: Proceedings of IEEE International Conference on Computer Communications (INFOCOM), 2000, pp. 22–31.
[44] P. Cao, S. Irani, Cost-aware WWW proxy caching algorithms, in: Proceedings of the USENIX Symposium on Internet Technologies and Systems (USITS), 1997, pp. 193–206.

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 B.Sc. 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 with 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" (September–October 2009), and translator for the greek language of the book "Google's PageRank and Beyond: The Science of Search Engine Rankings". His research interests lie are in the area of distributed systems, and in particular, on the Web/Internet, mobile and pervasive computing, mobile/vehicular ad hoc networks, wireless sensor 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.