Wireless Netw (2011) 17:717–737
DOI 10.1007/s11276-010-0311-x

High performance, low complexity cooperative caching for wireless sensor networks

Nikos Dimokas · Dimitrios Katsaros · Leandros Tassiulas · Yannis Manolopoulos

Published online: 23 December 2010

Abstract During the last decade, Wireless Sensor Networks have emerged and matured at such point that they currently support several applications such as environment control, intelligent buildings, target tracking in battlefields. The vast majority of these applications require an optimization to the communication among the sensors so as to serve data in short latency and with minimal energy consumption. Cooperative data caching has been proposed as an effective and efficient technique to achieve these goals concurrently. The essence of these protocols is the selection of the sensor nodes which will take special roles in running the caching and request forwarding decisions. This article introduces two new metrics to aid in the selection of such nodes. Based on these metrics, we propose two new cooperative caching protocols, *PCICC* and *scaPCICC*, which are compared against the state-of-the-art competing protocol, namely *NICoCa*. The proposed solutions are evaluated extensively in an advanced simulation environment and the results confirm that the proposed caching mechanisms prevail over its competitor. The evaluation attests also that the best policy is always *scaPCICC*, achieving the shortest latency and the least number of transmitted messages.

Keywords Wireless sensor networks · Social networks · Cooperative caching · Cache management · Replacement policy

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

N. Dimokas · Y. Manolopoulos
Department of Informatics, Aristotle University of Thessaloniki, Thessaloniki, Greece
e-mail: dimokas@delab.csd.auth.gr

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

D. Katsaros (✉) · L. Tassiulas
Department of Computer & Communication Engineering, University of Thessaly, Volos, Greece
e-mail: dkatsar@inf.uth.gr

L. Tassiulas
e-mail: leandros@uth.gr

1 Introduction

Wireless Sensor Networks [1, 2] (WSNs) emerged during the last decade due to the advances in low-power hardware and the development of appropriate software. A wireless sensor network consists of wirelessly interconnected devices (each being able to compute, control and communicate with each other) that can interact with their environment by controlling and sensing "physical" parameters. WSNs have fueled a huge number of applications, such as disaster relief, environment control and biodiversity mapping, machine surveillance, intelligent building, precision agriculture, pervasive health applications, target tracking in battlefields, and so on.

Although there is no single realization of a WSN to support so many diverse applications, there are some common characteristics of these networks, that need to be efficiently addressed in all these applications:

- Lifetime: Sensor nodes rely on a battery with limited lifetime, and their replacement is not possible due to physical constraints (they lie in oceans or in hostile environments) or it is of no interest to the owner of the sensor network.

- Scalability: The architecture and protocols of sensor networks must be able to scale up with (or to exploit) any number of sensors.
- Data-centric networking: The target of a conventional communication network is to move bits from one machine to another, but the actual purpose of a sensor network is to provide information and answers, not simply connections. Therefore, each sensor needs to construct responses to quite complex queries whose processing may involve the cooperation with many neighboring sensors, as the following example demonstrates.

Consider for instance a sensornet deployed in a modern battlefield, with sensor nodes dispersed in a large area; each sensor node is equipped with a micro-camera that can take a photograph of a very narrow band around its position. The sensors update the photographs they take (storing a prespecified number of the immediate past images, so as to be able to respond to historic queries), and share (on demand) with each other the new photographs, in order to built a more complete view of the region that is being monitored. The sharing is necessary because every micro-camera can capture a limited view of the whole region, either due to the sensor node's position or because of the obstacles that exist nearby the sensor node. Therefore, every sensor may request and receive a large number of photographs taken by some other sensor(s) through multi-hop communication. Afterwards, each sensor is able to respond to queries about "high-level" events, e.g., enemy presence.

In this hostile environment, it is obvious that sensor battery recharging is not a trivial task. Also, the location of the sensors has not been decided by some placement algorithm (the sensors were dropped by an unmanned aircraft), and the communication among them is strictly multihop.

The vast majority of applications running over WSNs require the optimization of the communication among the sensors so as to serve the requested data in short latency and with minimal energy consumption. The battery lifetime can be extended if the 'amount' of communication is reduced, which in turn can be done by caching useful data for each sensor either in its local store or in the near neigbhborhood. Additionally, caching can be very effective in reducing the need for network-wide transmissions, thus reducing the interference and overcoming the variable channel conditions. The cooperative data caching has been proposed as an effective and efficient technique to achieve these goals [3–5] (for more details cf. Sect. 2).

The fundamental aspect in all the proposed cooperative caching schemes for sensor networks is the identification of the nodes which will implement the aspects of the cooperation concerning the caching decisions, i.e., towards which nodes will the data request will be forwarded? which nodes will decide about which data will be cached in which nodes? and so on.

1.1 Motivation and contributions

The early proposals for cooperative caching in WSNs and mobile ad hoc networks [5, 6], did not pay particular attention to the selection of nodes that will have special roles in the cooperation protocol. The work [3] pointed out the singificance of the careful selection of these nodes; it was argued there, that these nodes should be "central" in the sensornet topology. Based on this, the authors proposed a cooperation scheme where the sensors with special role were selected based on their *ability to influence the communication between pairs of other nodes*, e.g., when these sensors lie in the communication path of the pair of nodes. This ability was quantified by calculating the *Node Importance index*— $\mathcal{NI}$ for each sensor, which is a localized version of the well-known betweenness centrality index [7] used in social network analysis.

Nevertheless, the $\mathcal{NI}$ metric suffers from the following disadvantages:

- Its computation by a sensor requires detailed knowledge of the connectivity of the sensor's one-hop neighbors, i.e., the sensor must exchange the set of its one-hop neighbors with every one-hop neighbor; thus more and larger packets travel in the network.
- The calculation of $\mathcal{NI}$ index, although quite fast, is not a $O(1)$ complexity operation, which might be an issue when the sensornet topology changes quite fast.
- The $\mathcal{NI}$ index might be misleading, since it is affected a lot by the existence of isolated nodes in the borders of the network (this issue will be explained in Sect. 5.1 with an example).
- Poor robustness, since the removal of a single node might change significantly the 'influence' values of many other neighboring nodes, and consequently the performance of the protocol.
- When the $\mathcal{NI}$ index is calculated over wider neighborhoods, and not simply for the two-hop neighborhood of a sensor node, it might change singificantly, since more connections and nodes come into play.

Motivated by the aforementioned shortcomings, this article proposes two new metrics to evaluate the significance of a sensor to undertake special roles in the cooperation, and based on these it describes the new cooperative caching protocols respectively. In particular, the article makes the following contributions:

- It describes a new centrality metric for sensor nodes, the *μ-power community index* ($\mu - \mathcal{PCI}$), which is

more informative than the node degree, it is more robust and computed faster compared to $\mathcal{NI}$; in addition, it is not affected by any isolated nodes.
- It describes a scaled version of the $\mu - \mathcal{PCI}$ (*scaPCI*), that improves the selection of the sensors with special role. The new metric requires the same communication cost with $\mathcal{NI}$ in order to be calculated, but it is more robust than $\mu - \mathcal{PCI}$ and $\mathcal{NI}$; in addition it is not affected by any isolated nodes;
- It develops two new cooperative caching[1] protocols for WSNs, the *PCICC* (PCI Cooperative Caching) and *scaPCICC*; these protocols are compared against the state-of-the-art protocol via simulation analysis.

The rest of this article is organized as follows: In Sect. 2, we review the related work, and in Sect. 3, we describe the problem been addressed in the current article. Section 4 describes the network model, i.e., any assumptions made in the present work; in Sect. 5 we describe the new centrality metrics and the components of the proposed cooperative caching protocols. Section 6 presents the simulation environment that was built to investigate the performance of the proposed protocols, and also describes the experiments and obtained results after the comparison of the protocols with the competing state-of-the-art scheme, and finally Sect. 7 conludes the article.

2 Related work

Caching is a significant technique to improve the performance of wired and wireless networks. Cooperative caching has attracted significant attention in the literature concerning various types of distributed systems: the Web [9], file servers [10], cellular networks [11], and so on. Nevertheless, the very limited capabilities of sensor nodes (in terms of energy, storage, and computation), the particularities of the wireless channel (variable capacity), and the multi-hop fashion of communication, turns the solutions proposed in the aforementioned environments of limited usefulness.

In the context of wireless sensor networks and ad hoc networks, it is beneficial to cache frequently accessed data items not only to reduce the average query latency, but also to save wireless bandwidth and extend the battery lifetime. A number of data replication schemes [12, 13] and caching schemes [5, 14] have been proposed in order to facilitate data access in mobile ad hoc networks (MANETs). Hara [12, 13] proposed a number of replica allocation techniques to increase data accessibility in MANETs. According to overall network topology and access frequency, the replicated data items are relocated periodically. However, these techniques normally require a priori knowledge of the network topology, a significant communication overhead and energy dissipation in order to relocate data items periodically and to preserve the consistency of data items.

In distributed systems over wireless networks based on multihop communication, cooperative caching has been proven a very efficient strategy to shorten the communication latency and conserve energy. Nuggehalli et al. [15] addressed the problem of energy-conscious cache placement in wireless ah hoc networks, and [16] considered the cache placement problem of minimizing total data access cost in ad hoc networks with multiple data items, and presented a polynomial-time centralized approximation algorithm to attack the problem, since it is NP-hard. Similarly, the work in [17] adresses the problem of cache placement in wireless sensor networks by constructing a static multicast tree based on the Steiner tree concept.

Though all these works address static cache placement issues, and are not strictly relevant to our problem, which implies dynamic sensornets. The work in [18] describes some techniques to reduce energy conservation in sensornet; one of them, namely data vanishing, resembles an emulated cache in the sense that while data are moving from the origin sensor to the sink (base station) via an aggregation/routing tree, the duplicate data are discarded. Still these techniques have no relation to the traditional concept of cooperative caching that we address here. Finally, the recent work reported in [19] addresses problems of cooperative caching in sensornets, but the proposed protocols tightly integrate consistency policies, which fall beyond the scope of this article (the interested reader can read a companion paper [8]).

2.1 Cooperative data caching in WSNs

In [20], the proposed strategy uses flooding technique to discover the nearest node that cached a copy of the requested data item. Although flooding may reduce the response delay, it imposes a communication overhead in terms of large number of messages that are being broadcasted. The authors try to reduce the overhead performing a limited flooding to nodes within k hops from the requester node, where k is the number of hops from the requester node to the data center. However, the communication overhead is still high when either the k is large or the network density is high.

A protocol based on maintaining detailed history (source and position of every "passing" datum) in nodes of a MANET and defining cooperation zones (setting a parameter z) has been proposed in [21], namely COOP. Each node maintains a table where previously received

[1] The issues pertaining the cache coherence aspect are discussed in the companion paper [8].

requests are recorded. Initially, a node checks its table after its local misses and before flooding a request. If matching entries are found, the node compares its distances to these matching caches and the original data source, and selects the closest node to redirect the request. In case of table miss, the requester node is using an adaptive z-hop flooding technique. If adaptive flooding also fails, the request is sent to data center. The proposed protocol [21] is unscalable due to the difficulty of properly configuring the parameter z and the cache space that is required for the requests table. It also suffers from a big communication overhead due to its flooding technique, especially in large and dense network topologies. In a recent work [22], the authors of COOP significantly improved upon the original scheme, by not exceeding neighborhood radius values beyond 3. Still, the long chain of resolution stages (about 6) jeopardizes the worst-case performance of the protocol. The flooding technique is also used in [14], where flooding is limited by imposing a threshold on the route existence probability. As a query packet is forwarded through a multi-hop communication path, its route existence probability decreases. However, defining an appropriate existence probability is topology-dependent.

Cooperative caching strategies in hybrid mobile ad hoc networks (i.e., with the presence of Access Points) are described in [23]. The authors proposed a cache for Internet-based MANETS (IMANET) where, by caching the data items in the local caches of the mobile nodes and making them available to other mobile nodes, members of the IMANET can efficiently access the data items. Thus, the aggregated local caches of the mobile nodes can be considered as a large, unified cache for the IMANET. The work reported in [6] considered cache replacement issues for wireless ad hoc networks but in the context of a very limited form of cooperation; a node which requests a data searches either in its local cache or in the caches of its one-hop neighbors (otherwise forwards the request to a fictitious data center). Thus, the authors try to analyze the impact of energy on cache replacement and propose an energy-efficient cache replacement policy based on coordinated replacement.

In ECOR [4], each mobile node forms a *cooperation zone* (CZ) with mobile nodes in proximity by exchanging messages to share their cached data items in order to minimize bandwidth and energy cost for each data retrieval. When a data request arrives, the node first searches the data in its CZ before forwarding the request to the data center. The authors developed an analytical model in order to determine the optimal radius of the cooperation zone based on mobile node's location, data popularity and node density. According to ECOR, each node broadcasts every modification of the cached data items to nodes that belong to its cooperation zone. Each node maintains a cache hint table for the cache information of all nodes in its proximity. However, ECOR exhibits a great number of exchanged messages and energy consumption in case of large node density and big cooperation radius. There is also a drawback in terms of communication overhead in case of big cooperation radius when a node incorrectly redirects a request packet to a node in its proximity.

In order to further save bandwidth for each data retrieval, PReCinCt [24] incorporates a novel cooperative caching scheme that caches relevant data among a set of peers in a region. The authors define a new cache replacement policy by considering not only each peer's own access frequency but also the importance of data items to other peers in the same region. The proposed caching strategy, Greedy-Dual Least Distance (GD-LD), uses a utility function to evaluate the importance of each data item based on a combination of three factors: the popularity of the item in the region, size of the item, and the region distance between the requesting and responding peers.

Yin and Cao [5] design and evaluate three cooperative caching schemes to efficiently support data access in MANETs and reduce query delay. In CacheData, the intermediate nodes may cache frequently accessed data items to serve future requests. In CachePath, the intermediate nodes may cache the path to a nearby requester node while forwarding the data and use the path information to redirect future requests to the nearby caching node. CachePath records the path when it is closer to the caching node than the data center. A hybrid protocol HybridCache was also proposed, which improves the performance by taking the advantages of CacheData and CachePath schemes while avoiding their weakness. In HybridCache, when an intermediate node forwards a data item, it caches the data or the path based on some criteria. These criteria include the data item size, the time-to-leave of the data item and the number of hops that a cached path can save, denoted as H_{save}. This value is the difference between the distance to the data center and the distance to the caching node. H_{save} must be greater than a system tuning threshold. One drawback with these methods is that caching information of a node cannot be shared if the node does not lie on the path between the data requester and the data source. Moreover, the threshold values used in these heuristics must be set carefully in order to achieve good performance. The HybridCache was proved inferior to *NICoCa*, described in [3] which took special consideration to select appropriate "central" nodes to carry out and coordinate the cooperation.

The *NICoCa* protocol [3] is based on the definition of a metric which quantifies for each sensor node how much "control" it can exert on pairs of cooperating nodes, assuming that these nodes use shortest paths-based communicating protocols. This metric is based on the well-known concept of betweenness centrality, and it is actually

a localized version of it. Each sensor node gets an "influence value", i.e., a number which represents the percentage of shortest paths between pairs of neighboring nodes that pass through this node. Then sensors with high "influence" are identified as the mediators which coordinate the caching and request forwarding decisions. The *NICoCa* protocol has a discovery and a cache replacement component to address the respective needs.

2.2 Research in centrality metrics

The research on centrality metrics has a fourty years history [7], when centrality metrics, such as the degree, closeness and betweenness centrality were introduced. Later, the concept of spectral centrality—based on the adjacency matrix spectrum—was investigated [25, 26], and more recently, amalgamation of earlier centrality metrics have been proposed [27, 28]. The literature on centrality metrics is quite large to be listed here, but we have to emphasize that new concepts are needed so as to be used in the design of communication protocols.

3 Problem description and protocol design requirements

In the online cooperative caching problem there are several goals that need to be optimized, such as energy consumption, access latency, number of copies of items to different caches so as to avoid waste of the aggregate cache space, and so on. These goals are often conflicting; for instance, in order to avoid waste of the aggregate cache space, a node is restricted from caching an item even though it has a relative high access rate for that item (a fact that would imply caching that item). Therefore, it is rather unfeasible to formulate the online cooperative caching problem using a single equation that would encompass all these factors. Alternatively, we prefer to express it as an optimization problem with the goal of optimizing one of these metrics, i.e., access latency and making a 'best-effort' for the rest of the factors. Thus, we provide the following formulation for the cooperative caching problem.

Given an ad hoc network of sensor nodes $G(V, E)$ with $|dataID|$ data items $D_1, D_2, , D_{|dataID|}$, 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 *cooperative caching problem* is an online problem, with the goal being the selection of a set of sets $M - \{M_1, M_2, , M_{|dataID|}\}$, where M_j is a set of sensors that store a copy of D_j, to minimize the total access cost:

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

and fulfilling the memory capacity constraint that:

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

Since earlier work [3] suggested that the smart selection of the so-called 'mediator' nodes is a crucial factor in addressing energy and latency considerations, one of the aims of the present work is to design centrality metrics to help us in the selection of the mediator nodes which will be robust and easy to compute (without the need of complex calculations or many rounds of message exchanges). The primary performance metrics that we are interested in optimizing are access delay and energy consumption.

4 Network model

We assume a WSN consisting of N sensors; no assumptions are made about the network diameter and/or network density. We consider only the following generic properties of the WSN:

- Sensor nodes are static, the communication links are bidirectional, and sensors communicate in a multi-hop fashion.
- The computation and communication capabilities are the same for all network nodes, and sensors do not need GPS-like hardware.
- Sensors use a routing protocol (e.g., AODV) to send requests to source nodes. Each sensor node has a cache space of C bytes, and calculates its power community index ($\mathcal{PCI}$). The neighboring nodes with the biggest $\mathcal{PCI}$ values are set as community caching nodes. Therefore, a node can be in two states: ordinary or community caching node (CCN).
- Similarly to [3, 5] and without loss of generality, we assume that the ultimate provider of the data items is a Data Center. Thus, every request can be satisfied even if a datum has been evicted from the caches of all sensors. The existence of the Data Center does not affect the protocols' performance; serves only the need to avoid having "failed" requests.

5 The new cooperative caching schemes

In our earlier discussion, we emphasized the impact of the selection of the sensors to assign to them special roles in

the cooperation, and, consistently with the findings in [3], we set as our target the discovery of the sensors that are more "central" in the sensornet. The quest for such nodes has been described also in [29]. The 'central' nodes are able to control the communication among others: for instance, (a) in routing protocols for sensornets, such nodes can be selected to forward the packets because, due to their position, they will succeed in reducing the routing latency, (b) in disconnection-tolerant mobile sensor networks, such nodes can be selected as data mules to carry messages, until they find the chance to pass these messages to sensors which are closer to the packets' final destination, and so no. Therefore the significance of such central sensors varies depending on the application and the protocol, and thus we use the word 'influence' to depict the ability of the central nodes to affect (usually for optimization purposes) the communication among other sensors. As another example, assume that we need to elect a 'leader' in a network to send notifications to or to receive alarms from all other nodes of the network within small delay. For such purposes we would select a node whose hop distance to all other network nodes is the smallest. Such an indication is offered by the closeness centrality metric.

On the other hand, the notion of centrality is susceptible to many interpretations; the $\mathcal{NI}$ index proposed one such interpretation, with the disadvantages recorded in Subsect. 1.1 Here, we follow a different route and define a new centrality index as an amalgam between the $\mathcal{NI}$ index and the sensor node degree. The next subsection provides its definition, whereas Subsect. 5.2 describes an enhanced version of the centrality index. Finally, Subsect. 5.3 describes the components of the cooperative caching protocol.

5.1 The power community index

We model our WSN as a graph $G(V, E)$, where V is the set of sensor nodes SN_1, SN_2,..., and $E \subseteq V \times V$ is the set of radio connections between the nodes. The existence of a link $(u, v) \in E$ also means that $(v, u) \in E$, and that sensor nodes u and v are within the transmission range of each other, in which case u and v are also called one-hop neighbors of each other. The *degree* D_i of a sensor node SN_i is the number of direct connections (links) of SN_i to other sensors. The sensor network is assumed to be in a connected state.

In some cases, the sensor degree has been used as a measure of centrality. Looking at Fig. 1, we see that the nodes 3, 4, 7, 6 are equally central with respect to their degree; they all have a degree equal to 4. In addition, if we compute the betweenness centrality for each sensor in the whole graph, then node 7 is the most "central", followed

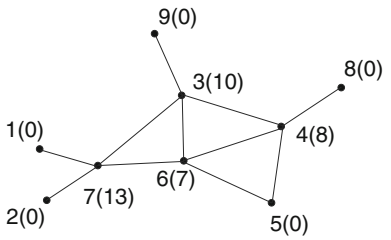


Fig. 1 The $\mathcal{NI}$ indexes (the *numbers in parentheses*) for a small sample graph

by nodes 3,4 and then comes node 6. This is somehow counter-intuitive, since node 6 has *all network nodes* at its vicinity (at a distance two-hops away). Starting from this observation, we propose a new centrality metric, called the *μ-Power Community Index* ($\mu - \mathcal{PCI}$), defined as follows:

Definition 1 The *μ-Power Community Index* of a sensor v is equal to k, such that there are up to $\mu \times k$ sensors in the μ-hop neighborhood of v with degree greater than or equal to k, and the rest of the sensors with in that neighborhood have a degree less than or equal to k.

Note that while calculating the degree of sensors which reside within the region defined by the μ-hop neighborhood of sensor v, it is possible that to this degree can contribute connections (links) to sensors outside of this region. Such cases are not precluded by the above definition, and as a matter of fact should not be precluded, since the definition tries to discover the sensors that can exert the maximum "influence" to other sensors.

It is clear that sensor nodes which have more connections (larger degree) are more likely to be "powerful", since they can directly affect more sensor nodes. But, their power depends also on the degrees of their one-hop neighbors. Large values for the $\mu - \mathcal{PCI}$ index of a sensor v indicate that this sensor v can reach others on relatively short paths (just like the $\mathcal{NI}$ index), or that the sensor v lies on a dense area of the sensor network (just like the indication provided by the sensors degree).

Since our target is WSNs applications, we are interested in a localized version of this metric; setting $\mu = 1$, we have the plain *Power Community Index* ($\mathcal{PCI}$), defined as:

Definition 2 The *Power Community Index* of a sensor v, $PCI(v)$, is equal to k such that there are no more than k one-hop neighbors of v with degree k or more, and the rest of the one-hop neighbors of v have a degree k or less.

With this definition in mind, we see in Fig. 1 that $PCI(7) = PCI(4) = 2$, whereas $PCI(6) = PCI(3) = 3$. Calculation of the $\mathcal{PCI}$ indexes for a larger graph is shown in Fig. 2.

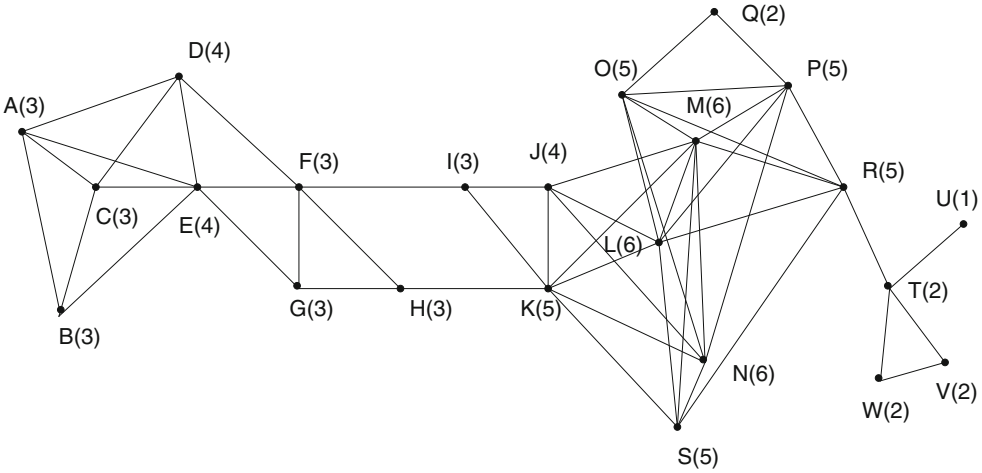


Fig. 2 Calculation of $\mathcal{PCI}$ for a sample graph. Each node is characterized by a pair of ID($\mathcal{PCI}$)

Apparently, when computing the $\mu - \mathcal{PCI}$ indexes for each sensor, we obtain a very informative picture of which sensors reside in significant positions of the network. However, due to energy and bandwidth constraints imposed by a WSN, our efforts focus on the localized distributed computation of $\mu - \mathcal{PCI}$, i.e., the $\mathcal{PCI}$. It is very easy to confirm that, even when we calculate the $\mathcal{PCI}$ of the sensors, the picture about the relative significance of the sensors remains very accurate. Thus, we consider only the $\mathcal{PCI}$ index of a sensor, in order to maximize the communication and energy savings.

The definition of $\mathcal{PCI}$ offers some great advantages, because the sensors, in order to compute their $\mathcal{PCI}$, need to exchange only only their degrees, and not detailed connectivity. Recall that the $\mathcal{NI}$ index [3] is based on the discovery of the number of shortest paths that are passing through each sensor. In order to calculate the $\mathcal{NI}$ index, each sensor needs the one-hop neighborhood of each and every one-hop neighbor; thus the calculation of the $\mathcal{NI}$ includes three steps: (a) initially, each node broadcasts its ID; (b) then, each sensor broadcasts the IDs of its one-hop neighbors; and, (c) each node calculates the shortest paths for its local network and decides about the important nodes. However, the calculation of $\mathcal{PCI}$ requires that at the second step each sensor broadcasts only an integer number (i.e., its degree), and at the third step no need for sophisticated computations are need. Thus, the calculation of $\mathcal{PCI}$ imposes less communication and computational cost.

5.2 The scaled power community index

It is obvious that a sensor node is powerful and can influence many other sensor nodes when the degrees of their one-hop neighbors are large. In this case, large values for the $\mathcal{PCI}$ of a sensor node v indicate that this sensor node v can reach others on relatively short paths, or that the sensor node v lies on a dense area of the sensor network. Although the $\mathcal{PCI}$ is a very useful metric with low computational and communication cost, it cannot always depict the significance of a node. The $\mathcal{PCI}$ of a sensor node v is calculated based on the degrees of sensors that are one-hop neighbors. However, the degree of a sensor may comprise connections (links) to sensors that are also one-hop neighbors of v. It is clear that $\mathcal{PCI}$ calculation can be affected by connections that are established among one-hop neighbors of a sensor node v. In this case, sensor node v does not influence many other sensor nodes, since the majority of sensor nodes are inside the communication range.

Looking at Fig. 3, we see that sensor node A has the same $\mathcal{PCI}$ value for both topologies, since nodes B, C, D, E, have the same degree; nodes B, C have degree equal to 4, while node D has degree equal to 5 and node E has degree equal to 6. Thus, the $\mathcal{PCI}$ value of node A is equal to 4 for both topologies. This is somehow inaccurate, since node A has at its vicinity (at a distance two-hops away) 7 nodes in the left topology and 16 nodes in the right topology. This result demonstrates that $\mathcal{PCI}$ cannot always capture the importance of a node, and this happens when *cliques* exist.

A clique of a graph is a subgraph such that all nodes of this subgraph are connected to each other. The existence of cliques in a graph is quantified by the *clustering coefficient*. The clustering coefficient [30], provides local information and is calculated as:

$$C(v) = \frac{2 \times L_v}{d_v \times (d_v - 1)}. \quad (2)$$

where d_v is the degree of node v and L_v is the number of links among the d_v neighbors of node v. The clustering coefficient of the whole network is the average of all C_v's. It is obvious, that small values of $C(v)$ indicates that one-hop neighbors of node v have a few connections among them.

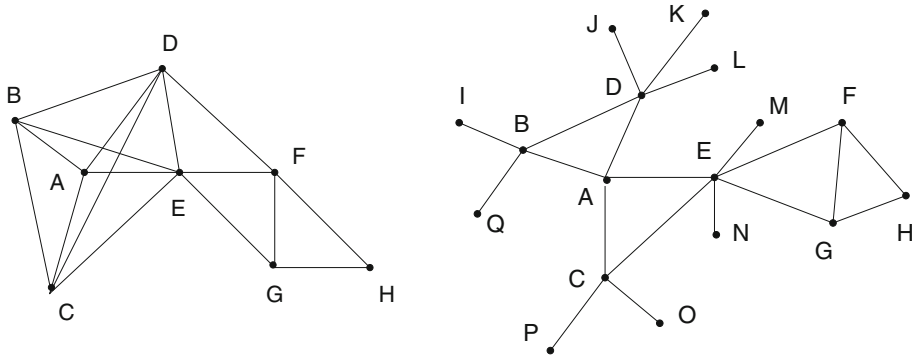


Fig. 3 An example of misbehavior of $\mathcal{PCI}$. In both cases $\mathcal{PCI}$ of node *A* has the same value, while in the *right figure* the number of nodes that it can influence are significant more, than those in *left figure*

Starting from the above observation that $\mathcal{PCI}$ cannot always indicate the importance of a sensor node and taking into account the clustering coefficient technique, we propose a new centrality metric, called the *scaled Power Community Index* (*scaPCI*), defined as follows:

Definition 3 The $scaPCI(v)$ of a vertex v is equal to:

$$scaPCI(v) = \frac{PCI(v)}{C(v)}. \quad (3)$$

The *scaPCI* is less complex to calculate than *NI* and it offers more advantages compared to $\mathcal{PCI}$, since its implementation is simple and the computational cost low. Although the $\mathcal{PCI}$ has a lower communication cost, the detailed connectivity that is being broadcasted by each sensor in *scaPCI* results in a more accurate selection of nodes that will have special roles in the cooperation protocol. The *scaPCI* calculation include three steps (like *NI*): (a) initially, each node broadcasts its ID; (b) then, each sensor broadcasts the IDs of its one-hop neighbors; and, (c) each node v calculates the $scaPCI(v)$. However, the *scaPCI* like *PCI* performs better than *NI*, since *NI* is affected a lot by the existence of isolated nodes in the borders of the network. Calculation of the *scaPCI* indexes for a large graph is shown in Fig. 4.

5.3 The *scaPCICC* and *PCICC* protocols

Apparently, the centrality metrics introduce earlier could have been incorporated into existing cooperative caching protocols, e.g., [3] which make use of mediator-type nodes to coordinate the caching decisions. But, their robustness and ease of calculation (both in terms of computation time and it terms of the number of message exhanges) allow for the desing of protocols which can be slightly more complex e.g., in the replacement decisions or in the coordination of neighborhoods by message exhanges, so as to further improve the performance of the protocols. Thus, it is worth striving for 'better' protocols, instead of 'pluging' these metrics into earlier works. In any case though, these metrics are of independent interest and we feel that they can be used for message carrying in delay-tolerant networks, in vehicular networks.

In the rest of this subsection we describe the details of the proposed cooperative caching schemes; i.e., the metadata kept at each sensor, the algorithm for the selection of the sensors which will coordinate the caching decisions, the forwarding of data requests, the cache replacement policy and the cache admission strategy. The two protocols follow the same principles except that *PCICC* calculates the $\mathcal{PCI}$ metric and *scaPCICC* calculates the *scaPCI* metric. In the following of this subsection we will present the whole operation of the *scaPCICC* protocol and explain only the different operations of *PCICC*.

At the very first step, it is supposed that each sensor is aware of the number and identity of its one-hop neighbors; this is achieved with the exchange of "HELLO" messages. Then, each node of the sensor network broadcasts its one-hop neighbors to its neighborhood. Thus, each node can obtain the list of its one-hop neighbors and the one-hop neighbors that each neighboring node has. The only difference between the *scaPCICC* and the *PCICC* protocol is that each node of the sensor network executing *PCICC* protocol broadcasts only its degree to its neighborhood. The information that is being broadcasted is only two numbers; the sensor identifier and its degree. Therefore, each node can obtain the list of its one-hop neighbors and the degree that each neighboring node has. We assume that we are able to determine an assignment of time slots to the sensor nodes such that no interference occurs, i.e., no two nodes transmit in the same time slot. Such a scheme can be found using the D2-coloring algorithm from [31].

When each sensor has gathered the "connectivity" information for its neighborhood, it then calculates and broadcasts its *scaPCI*. According to *PCICC* every sensor node calculates and broadcasts its $\mathcal{PCI}$. This is because each node needs the scaled Power Community Indexes (Power Community Indexes for *PCICC*, respectively) of neighboring nodes in order to characterize some of its neighbors as *community caching* nodes (CCNs). This

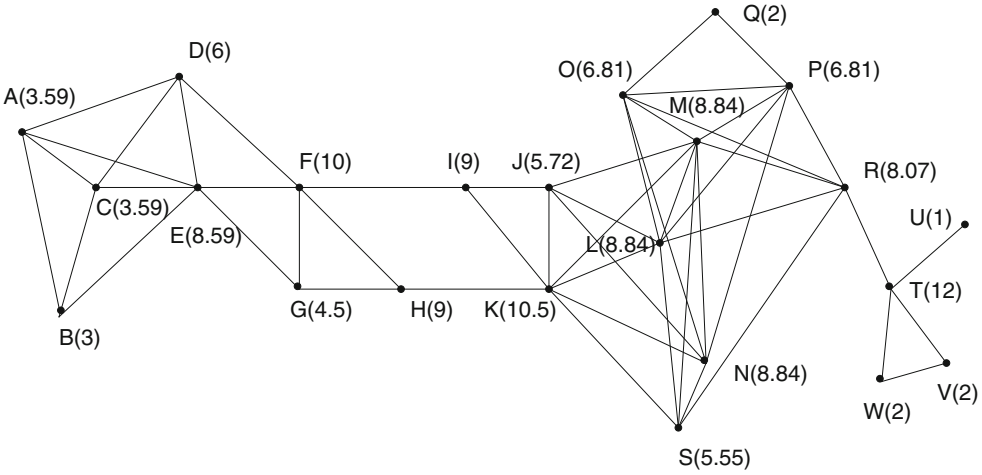


Fig. 4 Calculation of *scaPCI* for a sample graph. Each node is characterized by a pair of ID(*scaPCI*)

means that a node is a CCN for a specific set of nodes and not for any node in the network. This in turn means that in the worst case, every node could be a CCN for some other node; fortunately due to the properties of the centrality metrics this can not happen, because 'central' nodes are seen as such by the majority of their neighboring nodes. A sensor node v specifies its CCNs as the minimum set of its one-hop sensors, with the larger *scaPCI* values, which "cover" the two-hop neighborhood of node v. In other words, the sensor v constructs a minimum dominating set for its two-hop neighborhood (using the *scaPCI* value for making selections); the CCNs are the sensors which belong to this dominating set and are notified by the node itself. Apparently, there is no need for the node to know the "importance" of its neighbors with respect to the whole network; instead the node needs to know their importance in its neighborhood.

It is supposed that each sensor is aware of its remaining energy and of the free cache space; Additionally, each sensor node stores the following data/metadata:

- The dataID, the actual data item, and its access rate.
- The latency to obtain an data item (using exponential smoothing).
- The size $Size_i$ of datum i.
- A TTL interval (Time-To-Live) for each datum.
- For each cached item, the timestamps of the K most recent accesses to that item (usually, $K = 2$ or 3).
- Each cached item is characterized either as O (i.e., own) or H (i.e., hosted). If an H-item is requested by the caching node, then its state switches to O.
- Each O cached item is also identified as community (it derived from the two-hop neighborhood) and network (it derived outside the two-hop neighborhood).

scaPCICC uses a cache discovery algorithm (described in the next two paragraphs and exemplified in Fig. 6) to find the node who has cached the requested datum.

When a sensor node issues a request for a data item, it searches its local cache. If the item is found there (a local cache hit), then the K most recent access timestamps are updated. Otherwise (a local cache miss), the request is broadcasted and received by the CCNs. If none of them responds (a "community" cache miss), then the request is routed to the Data Center. Although, this store-process-forward mechanism may impose an enormous worst-case performance, this approach will work better when the cooperation and the replacement policy are effective, since it will drastically reduce a surge of messages travelling in the network that are practically unnecessary in the case when the requested item is already stored in the neighborhood of the requesting node.

When a a CCN that is not in the requestor's neighborhood intercepts a request, it searches its local cache. If it deduces (CCNs maintain local indexes) that the request can be satisfied by a neighboring node (a remote cache hit), then stops the request's route toward the Data Center, and forwards the request to this neighboring node. If more than one nodes can satisfy the request, then the node with the largest residual energy is selected. If the request can not be satisfied by this CCN, then it forwards the request toward the Data Center. Figure 5 shows the behaviour of *scaPCICC* protocol for a node's request. In summary, for every request issued by a sensor, one of the following four cases may take place:

1. Local hit (LH): The requested data item is cached by the node which issued the request. If this data item is valid (the TTL has not expired) then no further action is taken.
2. "Community" hit (CH): The requested data item is cached by a node in the two-hop neighborhood of the node which issued the request. In this case, the CCN(s) return to the requesting node the "location" of the node which stores the data item.

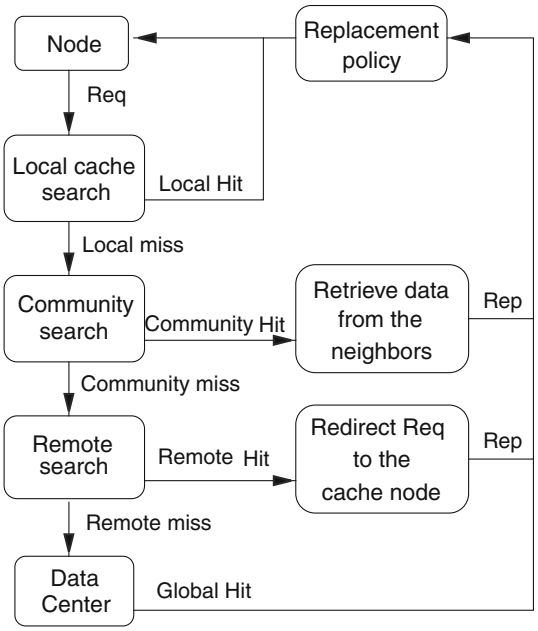


Fig. 5 The *scaPCICC* protocol

3. Remote hit (RH): The requested datum is cached by some node, and this node has at least one CCN residing along the path from the requesting node to the Data Center.
4. Global hit (GH): The requested data item is obtained from the Data Center.

When a node receives the data item that has requested for, then it caches it and broadcasts a small index packet containing the dataID and the associated TTL, its remaining energy and its free cache space. The CCNs which are also one-hop neighbors of this node store this broadcasted information. Every CCN stores the remaining energy and the free cache space for each one of its one-hop neighbors, and for each dataID that it has heard through the broadcasting operation, the TTL of this datum and the nodes that have cached this datum.

Figure 6 illustrates an example of the cache discovery phase. Here, the sensor node G is assumed to contain in its local cache the data item d_i that sensor node Q has requested. Additionally, the sensor nodes E, F and H are CCNs of G, while sensor node O is a CCN of Q. Initially, node Q sends the request to its CCN. The CCN performed a community search and in case of community cache miss a failure message is returned to requester node and node Q sends the request to the Data Center (node A). When an intermediate node receives a request packet, it searches in local cache and in community cache table. If the data item is not found, the request is forwarded through the path to the Data Center. Finally, node F receives the request and the requested data item is discovered in community cache table. The request is redirected to caching node G, and a reply message is sent back to the requester node.

5.3.1 The cache replacement component

The *scaPCICC* protocol utilizes an effective and efficient replacement policy in order to manage the cache space properly. A cache replacement policy is required when a SN attempts to cache a data item, but the cache is full. In replacement operation, one or more data items are evicted out of the local cache (to provide sufficient space) and the new one is cached.

When a node caches the datum, it broadcasts a small index packet containing the dataID and the associated TTL, its remaining energy and its free cache space. The CCNs which are also 1-hop neighbors of this node, store this broadcasted information. Every CCN node stores the remaining energy and the free cache space for each one of its 1-hop neighbors, and for each dataID that it has heard

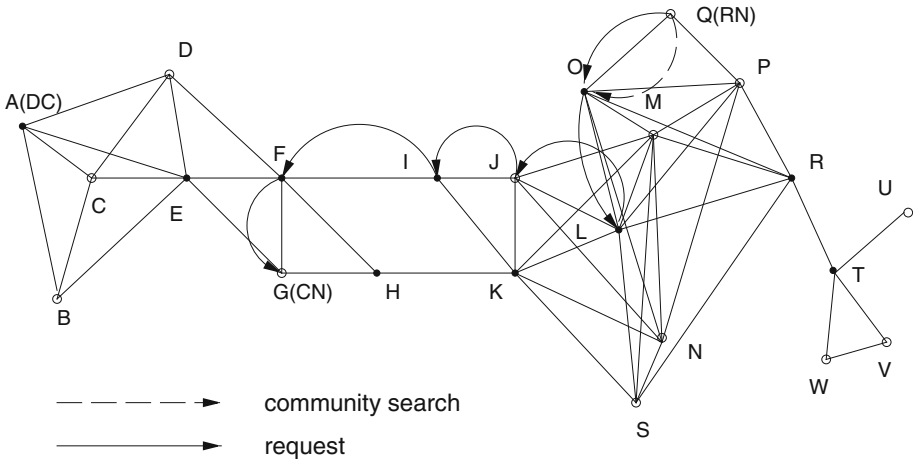


Fig. 6 A request packet from node Q is forwarded to the caching node G

through the broadcasting operation, the TTL of this datum and the nodes that have cached this datum. The *scaPCICC* protocol employs the following policy:

A. Initially each sensor node first evicts the data item that it has cached on behalf of some other node. Each cached item is characterized either as O (i.e., own) or H (i.e., hosted). In case of a local hit, then its state switches to O. If the available cache space is still smaller than the required, execute B.
B. Each *O* cached item is also identified as community (it originated from the two-hop neighborhood) and network (it originated outside the two-hop neighborhood)—CCNs provide this information. A sensor node evicts a number of *community* data items executing C, until sufficient space is provided. If the available cache space is still smaller than the required, execute C for the network cached data items.
C. We have developed a cost-based cache replacement function, where data items with the greatest costs are those that are removed from the cache. For each cached object i the following function is calculated: $\text{cost}(i) = \frac{\text{Lat}_i \times \text{Size}_i}{TTL_i \times AR_i}$. When a SN gets a reply message, it calculates the incurred latency (Lat). The smaller the latency of a data item is, the more likely to remain to cache. The access rate (AR) indicates the frequency that a cached item is being requested, while time-to-live (TTL) value determines the validity of a cached data item. A data item remains in cache when AR and TTL are big. Finally, the bigger the size of a data item is, the more likely to be removed from the cache.
D. Inform the CCNs about the candidate victims. If the data item is also cached by another node in the community, then CCNs transmit a delete message and the data item is evicted out of the local cache. Otherwise, each CCN sends a message that contains the node that has the largest residual energy and enough space to cache the data item. In this case, the node purges the data item and send it to the node with the largest residual energy. Finally, the CCNs update their cached metadata about the new state.
E. The node which caches this purged data item, informs the CCNs with the usual broadcasting procedure.

5.3.2 Cache admission control

When a sensor node receives the requested data, a cache admission control is triggered in order to determine whether it should be cached or not. Caching a data item might not be the appropriate solution since it can lead to lower probability of cache hits [32]. In this paper, the cache admission control policy allows a sensor node to cache every requested data item while there is enough available cache space. However, when the local cache is full, the policy favours caching of data items that are not cached in the two-hop neighborhood. This is because, the data items that lie in the community can be fetched in a relatively small interval of time, in contrast with those items that are many hops away.

Initially, each node may cache every requested data item including those derived from two-hop neighborhood. The data items that originated from outside the two-hop neighborhood are identified with bit 0, while those derived from two-hop neighborhood with bit 1. Thus, the cached data items are divided as community (those with bit 1) and network (those with bit 0). When the local cache is full and a new requested data item arrived, the cache admission control policy checks first if the response derived from the two-hop neighborhood. In this case, sensor node checks if any community data items are already cached and evicts a number of them until enough free space is provided. Finally, the requested data item is cached. However, if there aren't any community data items, the requested data item is not cached. The requested items that did not originate from the two-hop neighborhood are always cached. If the local cache is full then cache replacement policy is triggered.

6 Performance evaluation

We evaluated the performance of the *scaPCICC* and *PCICC* protocols through simulation experiments; we conducted several experiments with various parameters, and compared the performance of *scaPCICC* and *PCICC* to the state-of-the-art cooperative caching policy for WMSNs, namely *NICoCa* [3].[2]

6.1 Simulation model

All protocols have been implemented and evaluated with the J-Sim wireless network simulator [33]. In our simulations, the AODV [34] routing protocol is deployed to route the data traffic in the wireless sensor network. Although other routing schemes, like GPSR [35], 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

[2] In [3], the *NICoCa* protocol was compared against the Hybrid caching scheme [5], for many data/request distributions and many network topologies, and *NICoCa* proved superior in all cases.

Table 1 Radio characteristics

Operation	Energy dissipated
Transmitter/receiver electronics	$E_{\text{elec}} = 50$ nJ/bit
Transmit amplifier if $d_{toBS} \leq d_0$	$e_{fs} = 10$ pJ/bit/m^2
Transmit amplifier if $d_{toBS} > d_0$	$e_{mp} = 0.00134$ pJ/bit/m^4
Data aggregation	$E_{DA} = 5$ nJ/bit/signal

the radio propagation model. The wireless bandwidth is 2 Mbps. The radio characteristics used in our simulations are summarized in Table 1. 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 have been tested for a variety of sensor network topologies, to simulate sensor networks with varying values of node degree, from 4 to 10. Thus, we are able to simulate both sparse and dense sensor deployments. The denser the topology is, the larger the number of cliques is; though our simulator can not control precisely the number of generated cliques. We experiment with various sizes of the sensornet; we present here the results for two cases, namely when the number of sensors is 100 and 500. The distribution of the sizes of the data items is uniform between 1 and 10 KB. In order to filter out the statistical error and remove any bias from our results, we run each examined protocol for each network/data configuration at least 100 times.

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

The client query model is similar to what have been used in previous studies [3, 5]. Each sensor node generates read-only queries. After a query is sent out, if the sensor node does not receive the data item, it waits for an interval (tw) before sending a new query. The access pattern of sensor nodes follow the well-known Zipfian distribution with parameter θ (for $\theta = 0.0$, we get a uniform access pattern; for values of θ around 1, the access pattern is highly skewed). All sensors across the sensornet 'obey' the same θ, but they 'show preference' to different sets of data items. The sensors residing in the same grid (25 grids with size 100 × 100) m have more or less the same access pattern, i.e., they obey the same θ and they show preference for the same set of data items. The preferences across different grids are different. Thus, we model a locality in the data requests. We conducted experiments with varying θ values between 0.0 and 1.0. Here, we present the results for two representative cases, i.e., $\theta = 0.0$ and $\theta = 0.8$.

Table 2 Simulation parameters

Parameter	Default value	Range
Data items (N)	1,000	
S_{min} (KB)	1	
S_{max} (KB)	10	
Nodes (n)	500	100–500
Requests per node	250 (total)	200–300
Bandwidth (Mbps)	2	
Waiting interval (tw)	10 s	
Cache size (KB)	800	200–1,200
Zipfian skewness (θ)	0.8	0.0–1.0

Similar to [3, 5], two Data Centers are placed at opposite corners of the simulation area. Data Center 1 is placed at point ($x = 0$, $y = 0$) and Data Center 2 is placed at point ($x = 500$, $y = 500$). There are $N/2$ data items in each data center. Data items with even ids are stored at Data Center 1 and data items with odd ids are stored at Data Center 2. We assumed that data items are not updated. The system parameters are listed in Table 2.

6.2 Performance metrics

The measured quantities include the average query latency, the message overhead (as a direct metric of energy consumption) and the number of hits (local, remote and global) as a measure of the effectiveness of the cooperation. The query latency is the time elapsed between the time when the query is sent and the time when the data is transmitted back to the requester; the 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 for locating and retrieving data. Since the number of messages due to the routing scheme is the same for all schemes under study, thus we ignore the overhead for routing messages. It is evident that a small number of global hits implies less network congestion, and thus fewer collisions and packet drops. Moreover, a large number of remote hits proves the effectiveness of cooperation in

reducing the number of global hits. A large number of local hits does not imply an effective cooperative caching policy, unless it is accompanied by small number of global hits, since the cost of global hits overshadows the benefits of local hits.

6.3 Evaluation

We performed a large number of experiments varying the size of the sensornet (in terms of the number of its sensor nodes), varying the access profile of the sensor nodes, and the cache size relative to the aggregate size of all data items. In particular, we performed experiments for cache size equal to 1%, to 5–10% of the aggregated size of all distinct data, for access pattern with θ starting from 0.0 (uniform access pattern) to 1.0 (highly skewed access pattern), and for average sensor node degree equal to 4 (sparse sensornet) and 10 (dense sensornet). Each data item size is equal to a few kilobytes (KB). For each different setting we measured the number of hits (local, remote, global), the latency and the message overhead. The latency is measured in seconds. Figures 7, 8, 9, 10, 11, and 12 show the performance comparison of the protocols for a sensornet with 100 nodes, while Figs. 13, 14, 15, 16, 17, and 18 for a sensornet with 500 nodes.

The Figs. 11, 12, 17 and 18 depict the efficiency of *scaPCICC* and *PCICC* over *NICoCa* in terms of hits. These graphs should be interpreted as follows: *the line corresponding to Global Hits, represents the (percentage) reduction in global hits achieved by scaPCICC and PCICC with respect to the global hits achieved by NICoCa*; *the line corresponding to Remote Hits, represents the (percentage) increase in remote hits achieved by scaPCICC and PCICC with respect to the remote hits achieved by NICoCa*; *the line corresponding to Local Hits, represents the (percentage) decrease in local hits achieved by scaPCICC and PCICC with respect to the local hits achieved by NICoCa*.

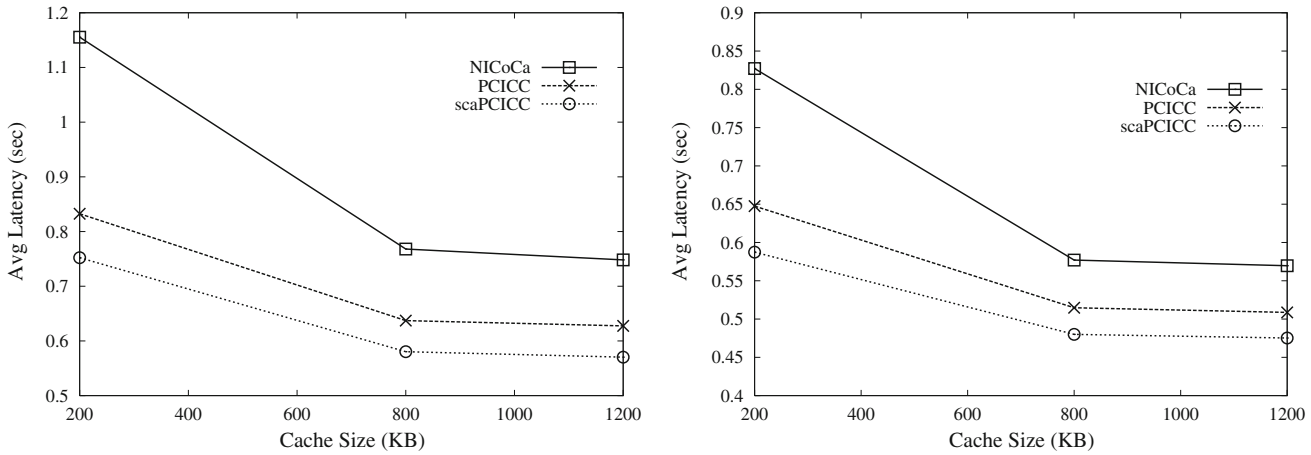


Fig. 7 Impact of sensor cache size on latency ($\theta = 0.0$ and $\theta = 0.8$) in a sparse WSN ($d = 4$) with 100 sensors

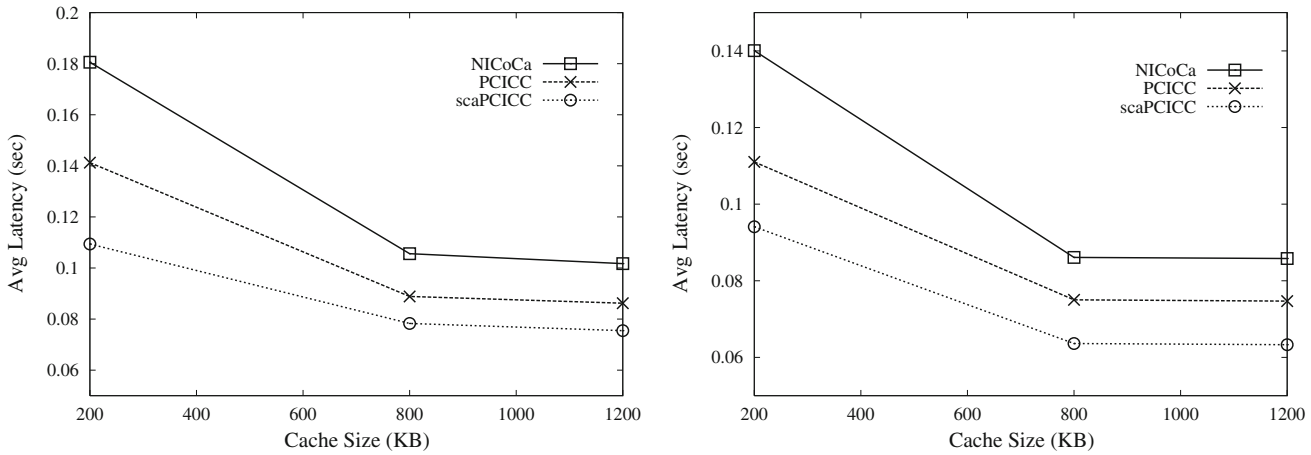


Fig. 8 Impact of sensor cache size on latency ($\theta = 0.0$ and $\theta = 0.8$) in a dense WSN ($d = 10$) with 100 sensors

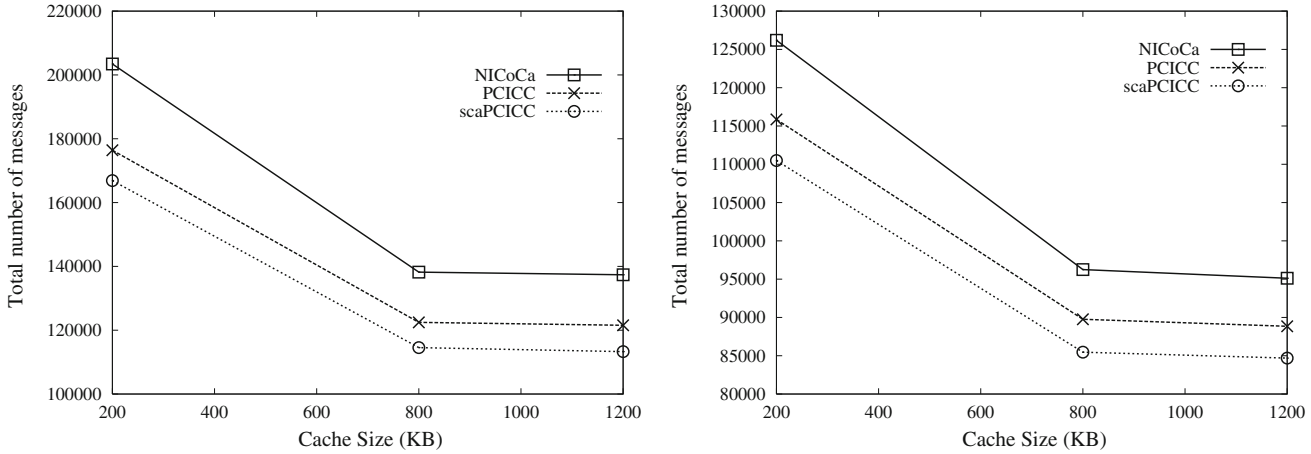


Fig. 9 Impact of sensor cache size on number of messages ($\theta = 0.0$ and $\theta = 0.8$) in a sparse WSN ($d = 4$) with 100 sensors

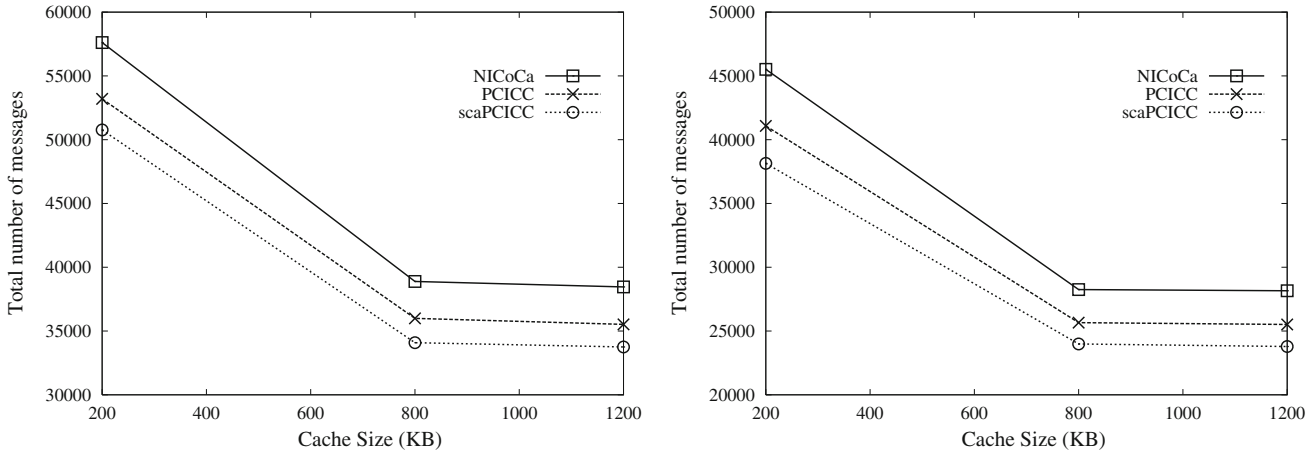


Fig. 10 Impact of sensor cache size on number of messages ($\theta = 0.0$ and $\theta = 0.8$) in a dense WSN ($d = 10$) with 100 sensors

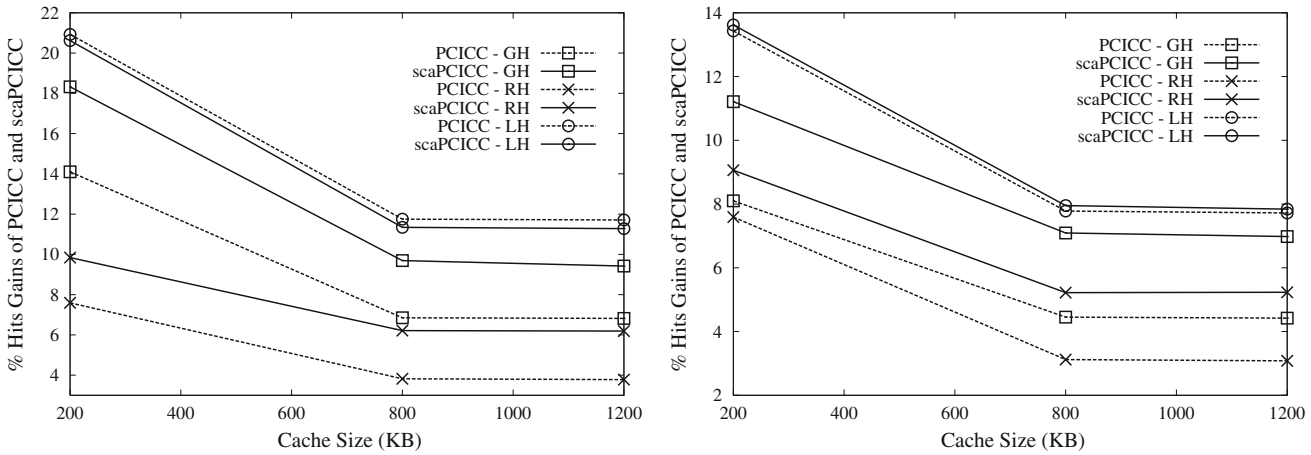


Fig. 11 Impact of sensor cache size on hits ($\theta = 0.0$ and $\theta = 0.8$) in a sparse WSN ($d = 4$) with 100 sensors

As expected, all cooperative caching schemes exhibit better performance for all metrics with increasing cache size; therefore caching is indeed a useful technique, irrespectively of the network topology. The second generic observation is that the proposed *scaPCICC* and *PCICC* protocols are superior to its competitor for all data/network

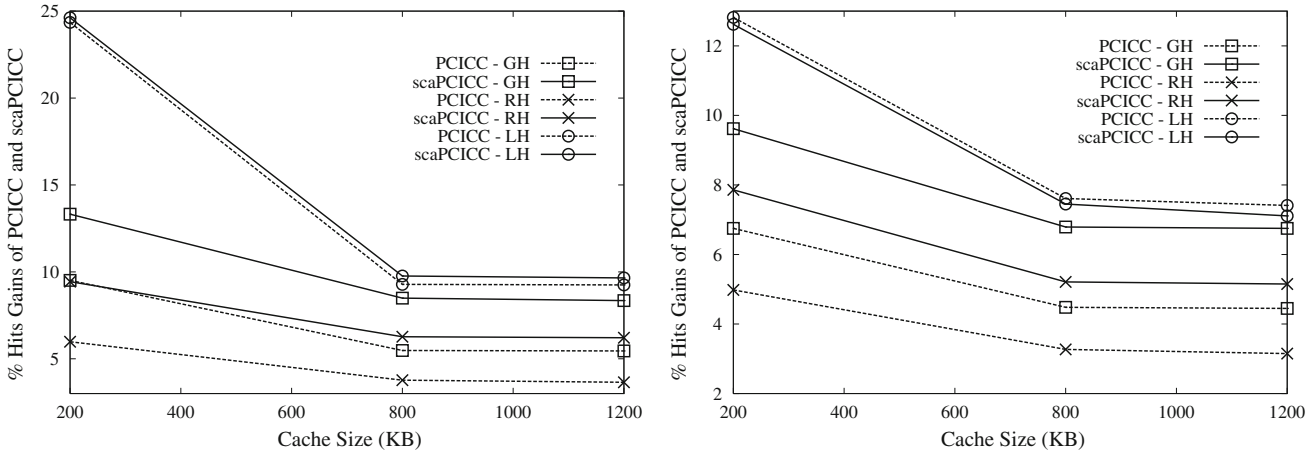


Fig. 12 Impact of sensor cache size on hits ($\theta = 0.0$ and $\theta = 0.8$) in a dense WSN ($d = 10$) with 100 sensors

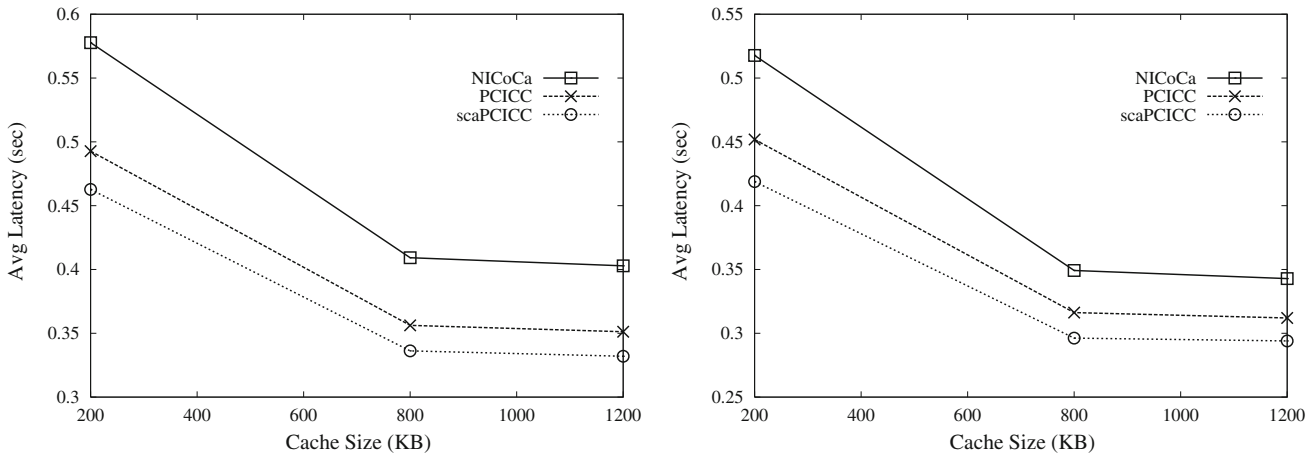


Fig. 13 Impact of sensor cache size on latency ($\theta = 0.0$ and $\theta = 0.8$) in a sparse WSN ($d = 4$) with 500 sensors

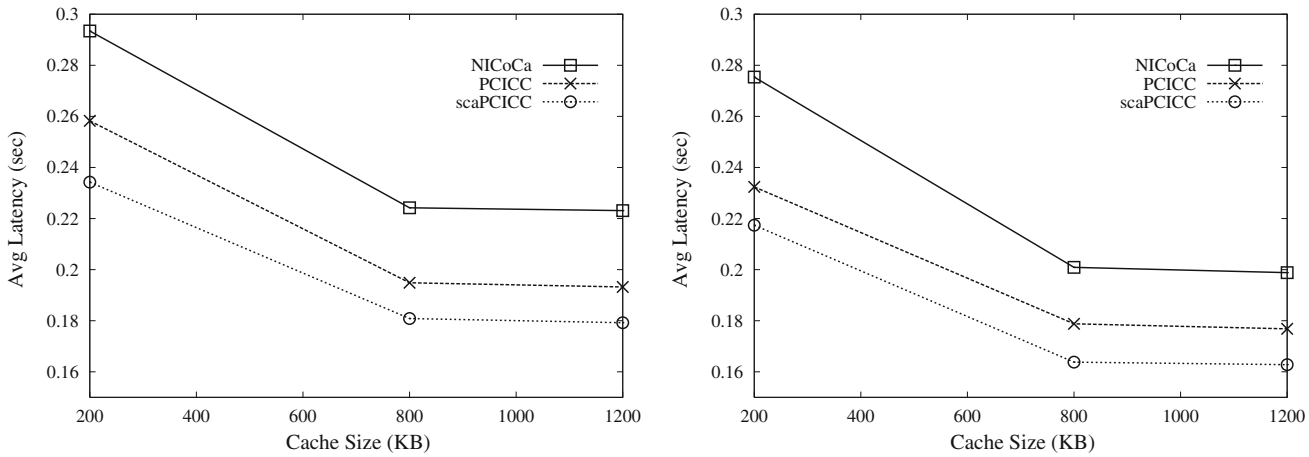


Fig. 14 Impact of sensor cache size on latency ($\theta = 0.0$ and $\theta = 0.8$) in a dense WSN ($d = 10$) with 500 sensors

configurations. As the cache space in each sensor increases toward an infinite cache, that could ideally accommodate all items, the actual performance gap (latency and travelling messages) between the protocols diminishes.

The performance of the protocols with respect to the average latency incurred for varying cache sizes for both sparse and dense sensor network deployments are depicted in Figs. 7, 8, 13 and 14. The dominant observation is that

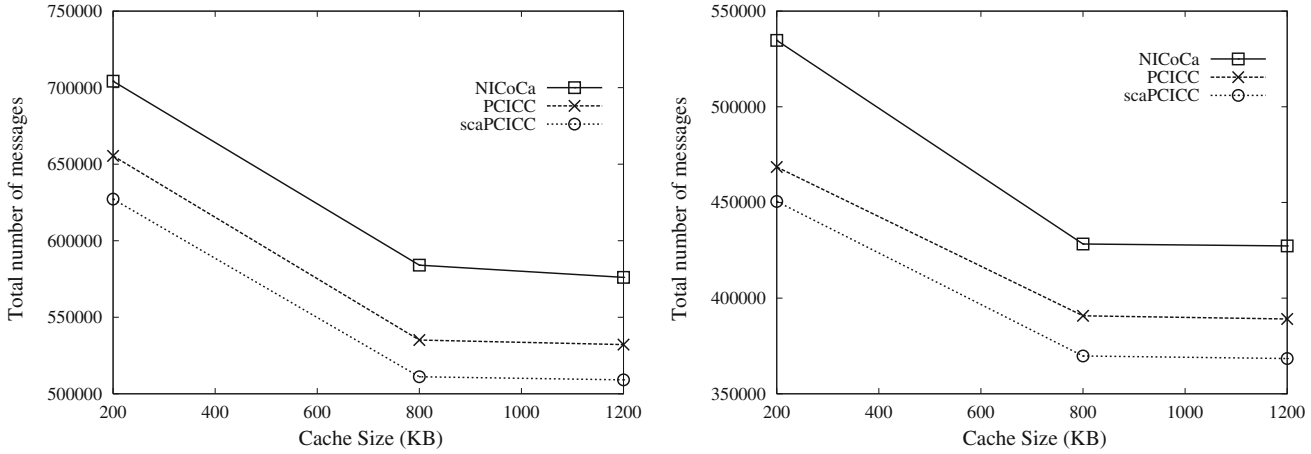


Fig. 15 Impact of sensor cache size on number of messages ($\theta = 0.0$ and $\theta = 0.8$) in a sparse WSN ($d = 4$) with 500 sensors

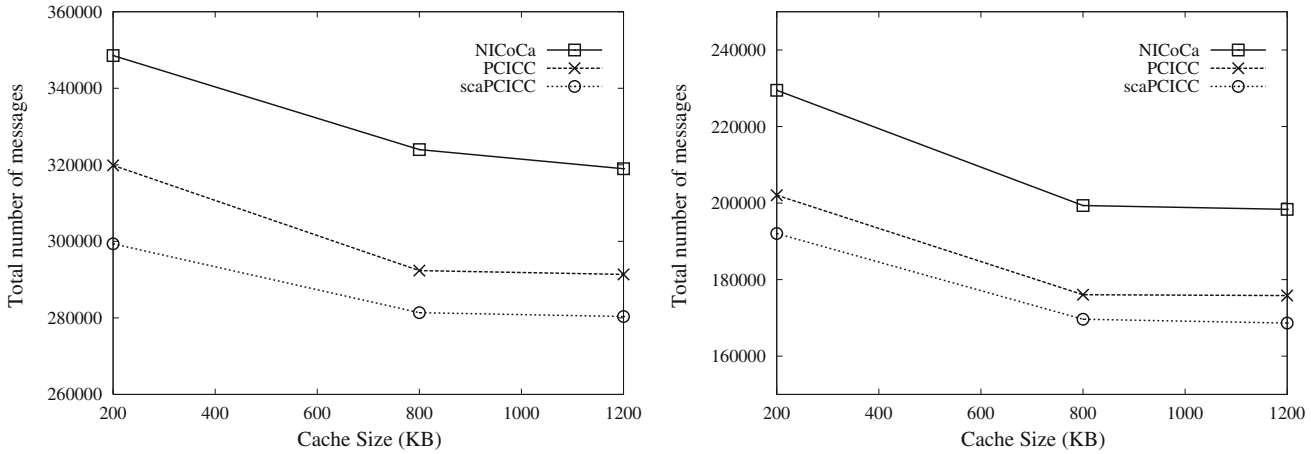


Fig. 16 Impact of sensor cache size on number of messages ($\theta = 0.0$ and $\theta = 0.8$) in a dense WSN ($d = 10$) with 500 sensors

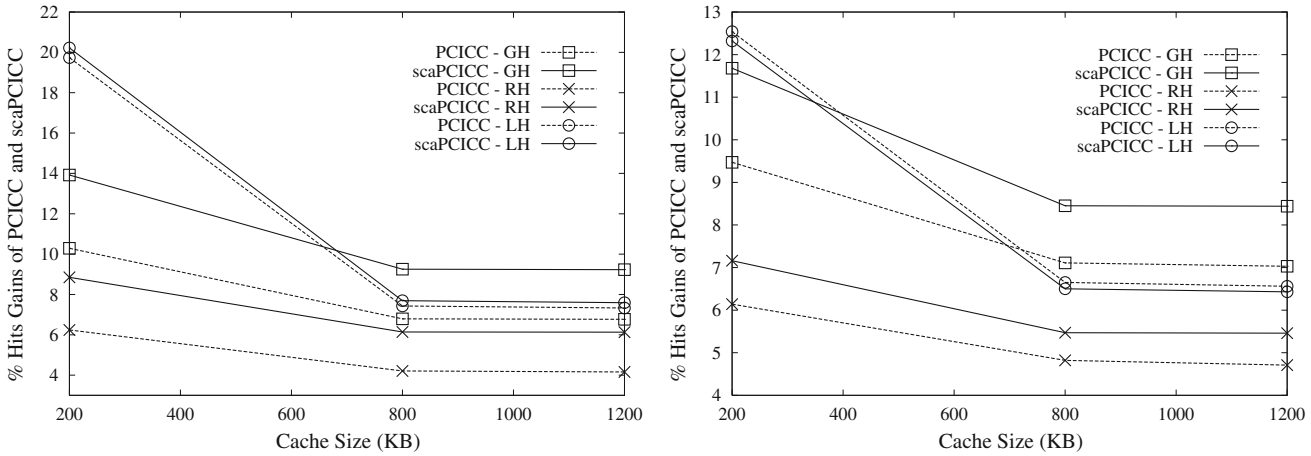


Fig. 17 Impact of sensor cache size on hits ($\theta = 0.0$ and $\theta = 0.8$) in a sparse WSN ($d = 4$) with 500 sensors

caching is more beneficial for sparse networks, since it can balance the (relatively) longer paths to the data that increase the latency. The latency incurred by *scaPCICC* is 10–18% smaller than that of NICoCa. Due to the central points that community cache nodes are located, the query requests are served faster than that of *NICoCa*.

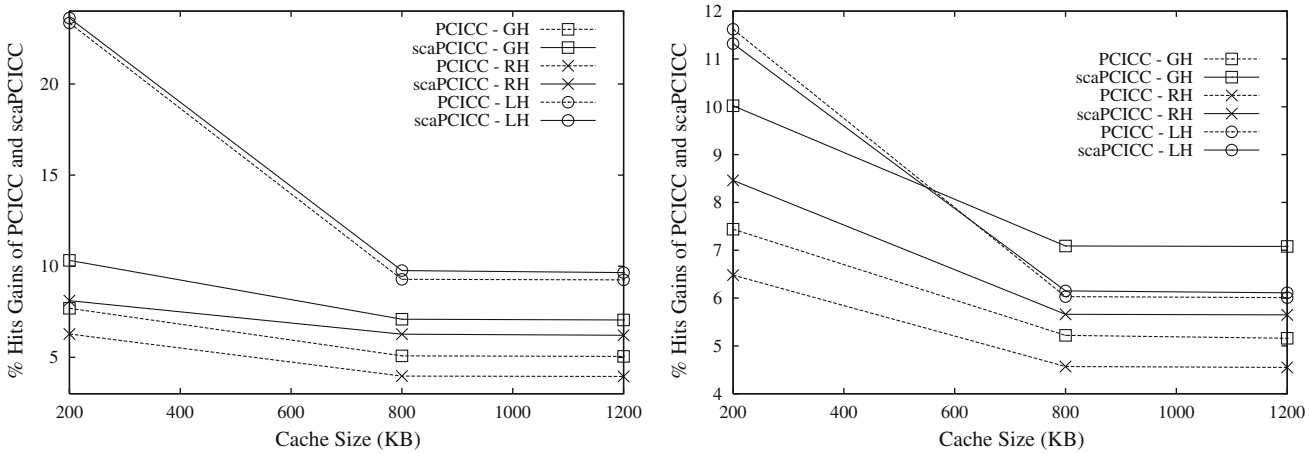


Fig. 18 Impact of sensor cache size on hits ($\theta = 0.0$ and $\theta = 0.8$) in a dense WSN ($d = 10$) with 500 sensors

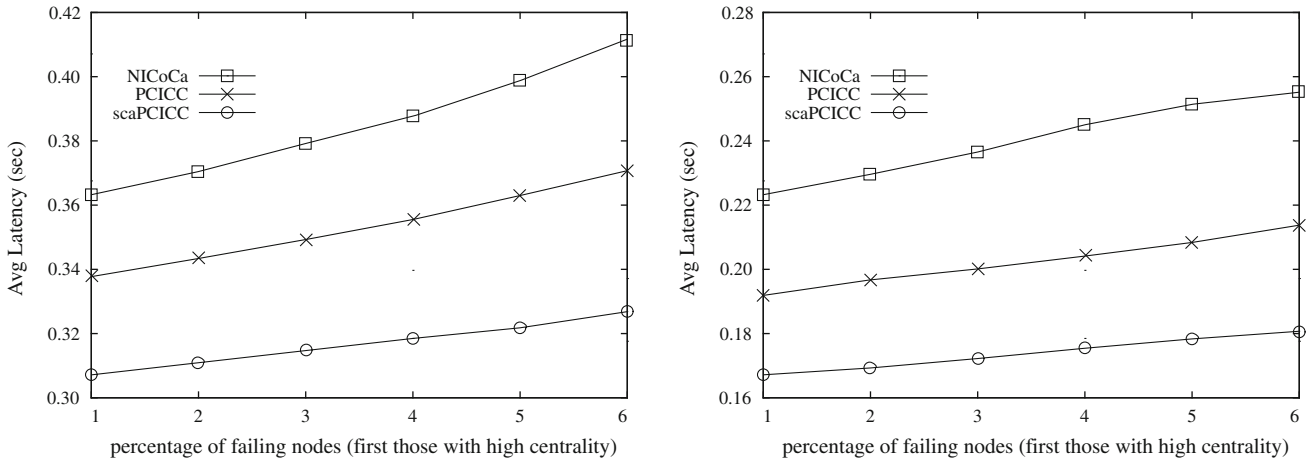


Fig. 19 Impact of failing sensors with high centrality on latency ($\theta = 0.8$) in a sparse ($d = 4$) and in a dense WSN ($d = 10$) with 500 sensors and cache of 800KB

We also 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 Figs. 9, 10, 15 and 16. The relative results follow the same trends that we observed in the previous experiment; there is a close connection between the number of 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. When a sensor node broadcasts a large number of messages, the energy dissipation increases. Thus, the message overhead metric exhibits also the energy consumption, since the fewer messages result in smaller energy consumption.

Finally, it is interesting to note that the *scaPCICC* protocol achieves an average of 13% fewer global hits than *NICoCa*, and around 10% more remote hits. The results are depicted in Figs. 11, 12, 17 and 18. The reduction in global hits and simultaneous increase in remote hits proves that *scaPCICC* achieves a more successful cooperation, and this is directly attributed to the selection of CCNs. Therefore, the proposed centrality metric is indeed useful and it is able to better capture the "significant" nodes in the sensornet. In terms of local hits, both protocols (*scaPCICC* and *PCICC*) have similar performance, but the different admission policy adopted by the proposed protocols results in fewer local hits and larger number of remote cache hits.

We argued from the beginning of this paper that the selection of the sensors that will coordinate the caching decisions is of critical importance. To provide one more evidence for this, we performed the following experiment. For a fixed cache size (i.e., 800 KB), we gradually removed (a percentage of) the sensors with high centrality (those with high *NI* for the *NICoCa* protocol, those with high $\mathcal{PCI}$ for the *scaPCICC* protocol and so on) and we measured the incurred latency. The results are depicted in Fig. 19. As expected the latency increases when more and

more sensors with high centrality value "deplete" their energy and are not part of the network. This increase is more steep for the sparse networks, since significantly fewer sensors have considerably large centrality values; on the other hand, in dense sensornets the impact of the node removal is less 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 experiment 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 latency incurred for the competing protocols, but, with the IEEE 802.15.4 as the MAC layer. The results are illustrated in Fig. 20 (it is the analogous of Figs. 13(b), 14(b)). Although there exist 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 detail 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. 21. There is no noticeable difference in the performance of the

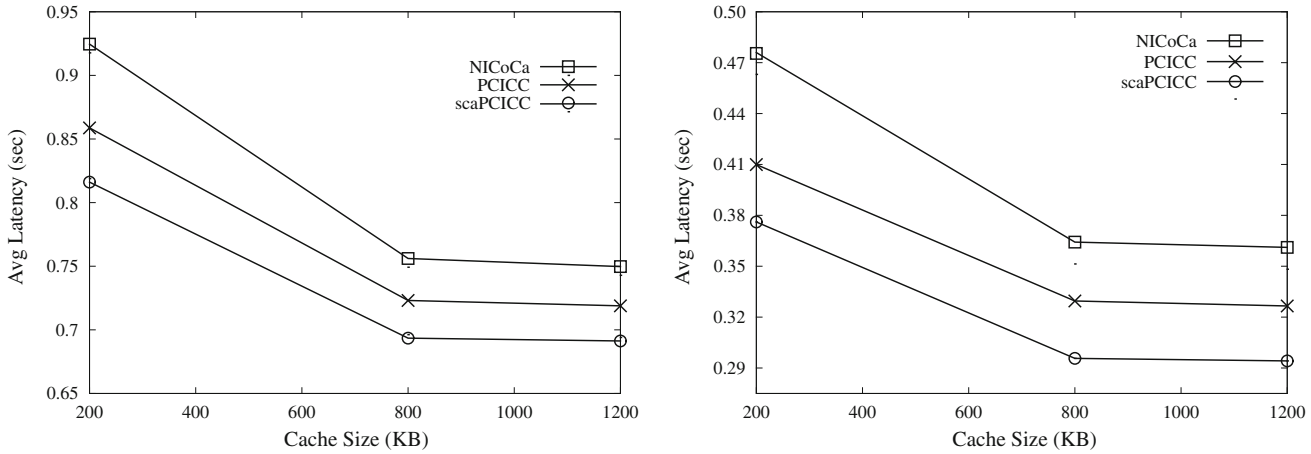


Fig. 20 Impact of sensor cache size on latency ($\theta = 0.8$) in a sparse ($d = 4$) and in a dense ($d = 10$) WSN with 500 sensors and 802.15.4 MAC protocol

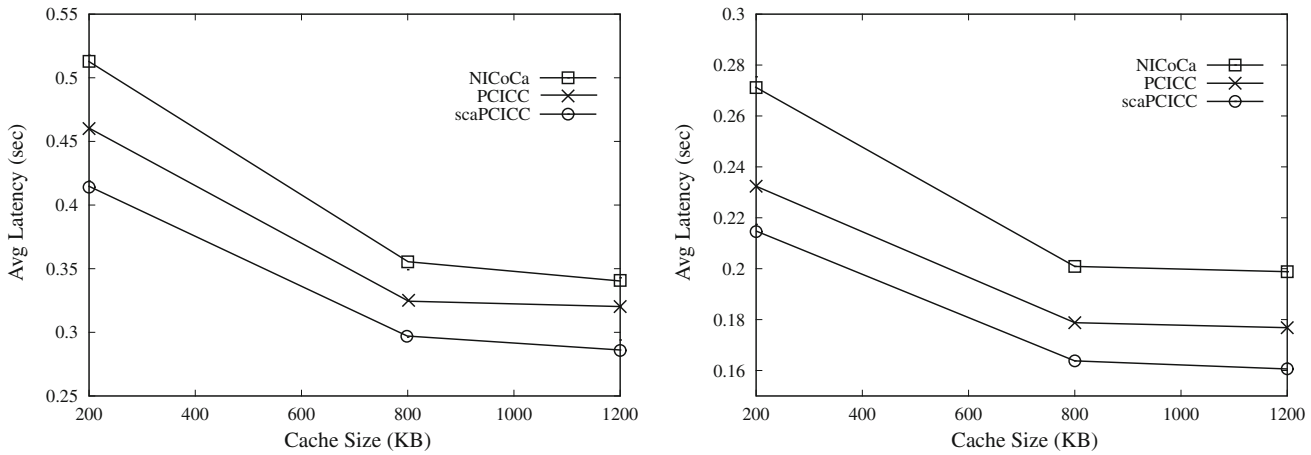


Fig. 21 Impact of sensor cache size on latency ($\theta = 0.8$) in a sparse ($d = 4$) and in a dense ($d = 10$) WSN with 500 sensors using the GPSR protocol

cooperative caching algorithms (contrast Figs. 13(b), 14(b)), which is expected since the cooperation scheme is the major factor that affects the performance, and not the routing scheme; a similar observation was also made in a companion paper [8] which concerned cooperative cache consistency.

In general, the performance of each protocol improves in dense sensornets, since we constrain more sensor nodes to be dispersed in the same geographical region, thus creating more replicas of the same data and providing more alternative paths to the data. This better performance is reflected to the access latency, hit ratio and message overhead.

In summary, for all network topologies *scaPCICC* achieves more remote hits and less global hits than *PCICC* and *NICoCa*. This performance gap is slightly better in favor of *scaPCICC* as we move from dense to sparse WSNs. Finally, all protocols achieve significant performance gains for skewed access pattern ($\theta = 0.8$). This is because the number of neighboring nodes that request the same data items is increased. Therefore, the cooperative caching protocols perform better.

7 Conclusions

The proliferation of applications based on wireless sensor networks depends mainly on the ability of the underlying protocols to scale to large number of sensors, to conserve energy and provide answers with short latency. Cooperative data caching has been proposed as an effective and efficient technique to achieve these goals concurrently. The essence of these protocols is the selection of the sensor nodes which will take special roles in running the caching and request forwarding decisions. The article introduced two new centrality metrics to aid in the selection of such nodes, and proposed two new cooperative caching protocols (*PCICC* and *scaPCICC*), which, based on simulation analysis, proved superior to the state-of-the-art competing protocol, achieving average gains in energy consuption and latency around 20%.

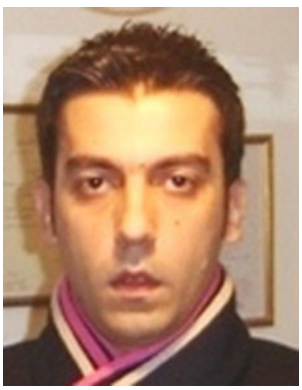
Acknowledgments The authors wish to thank the anonymous reviewers for their valuable comments and suggestions.

References

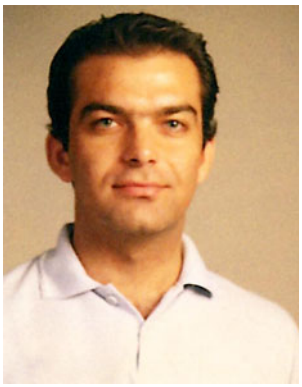
1. Akyildiz, I., Su, W., Sankarasubramaniam, Y., & Cayirci, E. (2002). A survey on sensor networks. *IEEE Communications magazine, 40*(8), 102–114.
2. Karl, H., & Willig, A. (2006). *Protocols and architectures for wireless sensor networks*. New York: Wiley.
3. Dimokas, N., Katsaros, D., & Manolopoulos, Y. (2008). Cooperative caching in wireless multimedia sensor networks. *ACM Mobile Networks and Applications, 13*(3–4), 337–356.
4. Shen, H., Das, S. K., Kumar, M., & Wang, Z. (2004). 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, pp. 841–853).
5. Yin, L., & Cao, G. (2006). Supporting cooperative caching in ad hoc networks. *IEEE Transactions on Mobile Computing, 5*(1), 77–89.
6. Li, W., Chan, E., & Chen, D. (2007). Energy-efficient cache replacement policies for cooperative caching in mobile ad hoc network. In *Proceedings of the IEEE WCNC* (pp 3349–3354).
7. Freeman, L. C. (1977). A set of measures of centrality based on betweenness. *Sociometry, 40*(1), 35–41.
8. Dimokas, N., Katsaros, D., & Manolopoulos, Y. (2010). Cache consistency in wireless multimedia sensor networks. *Ad Hoc Networks, 8*(2), 214–240.
9. Fan, L., Cao, P., Almeida J. M., & Broder, A. Z. (2000). Summary cache: A scalable wide-area Web cache sharing protocol. *IEEE/ACM Transactions on Networking, 8*(3), 281–293.
10. Annapureddy, S., Freedman, M. J., & Mazières, D. (2005). Shark: Scaling file servers via cooperative caching. In *Proceedings of USENIX NSDI* (pp. 129–142)
11. Hara, T. (2002). Cooperative caching by mobile clients in push-based information systems. In *Proceedings of ACM CIKM* (pp. 186–193).
12. Hara, T. (2003). Replica allocation methods in ad hoc networks with data update. *ACM Mobile Networks and Applications, 8*(4), 343–354.
13. Hara, T., & Madria, S. K. (2006). Data replication for improving data accessibility in ad hoc networks. *IEEE Transactions on Mobile Computing, 5*(11), 1515–1532.
14. Sailhan, F., & Issarny, V. (2002). Energy-aware Web caching for mobile terminals. In *Proceedings of the IEEE international conference on distributed computing systems workshops (ICDCSW)* (pp. 820–825).
15. Nuggehalli, P., Srinivasan, V., & Chiasserini, C. F. (2003). Energy-efficient caching strategies in ad hoc wireless networks. In *Proceedings of ACM MobiHoc* (pp. 25–34).
16. Tang, B., Gupta, H., & Das, S. R. (2008). Benefit-based data caching in ad hoc networks. *IEEE Transactions on Mobile Computing, 7*(3), 289–304.
17. Prabh, K. S., & Abdelzaher, T. F. (2005). Energy-conserving data cache placement in sensor networks. *ACM Transactions On Sensor Networks, 1*(2), 178–203.
18. Rahman, M. A., & Hussain, S. (2007). Effective caching in wireless sensor networks. In *Proceedings of the international IEEE conference on advanced information networking and applications workshops (AINAW)* (Vol. 1, pp. 43–47).
19. Sharma, T. P., Joshi, R. C., & Misra, M. (2009). Cooperative caching for homogeneous wireless sensor networks. *International Journal of Communication Networks and Distributed Systems, 2*(4), 424–451.
20. Law, W., Kumar, M., & Venkatesh, S. (2002). A cooperative cache architecture in supporting caching multimedia objects in MANETs. In *Proceedings of the international workshop on wireless mobile multimedia.*
21. Du, Y., & Gupta, K. S. (2005). COOP: A cooperative caching service in MANETs. In *Proceedings of ICAS-ICNS* (pp. 58–63).
22. Du, Y., Gupta, K. S., & Varsamopoulos, G. (2009). Improving on-demand data access efficiency in MANETs with cooperative caching. *Ad Hoc Networks, 7*(3), 579–598.

23. Lim, S., Lee, W. C., Cao, G., & Das, C. R. (2006). A novel caching scheme for improving internet-based mobile ad hoc networks performance. *Ad Hoc Networks, 4*(2), 225–239.
24. Shen, H., Joseph, M. S., Kumar, M., & Das, S. K. (2005). PReCinCt: A scheme for cooperative caching in mobile peer-to-peer systems. In *Proceedings of the international parallel and distributed processing symposium (IPDPS)*.
25. Brin, S., Page, L., Motwani, R., & Winograd, T. (1999). *PageRank citation ranking: Bringing order to the Web*. Tech. Rep. 1999-66, Computer Science Department, Stanford University.
26. Bonacich, P., & Lloyd, P. (2001). Eigenvector-like measures of centrality for asymmetric relations. *Social Networks, 23*(3), 191–201.
27. Hwang, W., Kim, T., Ramanathan, M., & Zhang, A. (2008). Bridging centrality: Graph mining from element level to group level. In *Proceedings ACM SIGKDD* (pp. 336–344).
28. Nanda, S., & David Kotz, D. (2008). *Localized bridging centrality for distributed network analysis*. Tech. Rep. 2008-612, Computer Science Department, Dartmouth College.
29. Erramilli, V., Crovella, M., Chaintreau, A., & Diot, C. (2008). Delegation forwarding. In *Proceedings of ACM MobiHoc* (pp. 251–259).
30. Watts, D. J., & Strogatz, S. H. (1998). Collective dynamics of 'small-world' networks. *Nature, 393*, 440–442.
31. Gandhi, R., & Parthasarathy, S. (2004). *Fast distributed well connected dominating sets for ad hoc networks*. Tech. Rep. CS-TR-4559, Computer Science Department, University of Maryland at College Park.
32. Cao, G., Yin, L., & Das, S. (2004). Cooperative cache based data access framework for ad hoc networks. *IEEE Computer, 37*(2), 32–39.
33. Sobeih, A., Hou, J. C., Kung, L. C., Li, N., Zhang, H., Chen, W. P., Tyan, H. Y., & Lim, H. (2006). J-Sim: A simulation and emulation environment for wireless sensor networks. *IEEE Wireless Communications magazine, 13*(4), 104–119.
34. Perkins, C. E., & Royer, E. (1999). Ad hoc on-demand distance vector routing. In *Proceedings of the IEEE workshop on mobile computing systems and applications* (pp. 90–100).
35. Karp, B., & Kung, H. T. (2000). GPSR: Greedy perimeter stateless routing for wireless networks. In *Proceedings of the ACM/IEEE international conference on mobile computing and networking (MobiCom)* (pp. 243–254).

Author Biographies



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 2009), and translator for the greek language of the books 'Google's PageRank and Beyond: The Science of Search Engine Rankings' and "Introduction to Information Retrieval". His research interests are in the area of distributed systems, including the Web and Internet, mobile and pervasive computing, mobile/vehicular ad hoc networks, wireless sensor networks.

Leandros Tassiulas (S′89, M′91, SM/06, F/07) was born in 1965, in Katerini, Greece. He obtained the Diploma in Electrical Engineering from the Aristotelian University of Thessaloniki, Thessaloniki, Greece in 1987, and the M.S. and Ph.D. degrees in Electrical Engineering from the University of Maryland, College Park in 1989 and 1991 respectively. He is Professor in the Department of Computer and Telecommunications Engineering, University of Thessaly, Greece and Research Professor in the Department of Electrical and Computer England the Institute for Systems Research, University of Maryland College Park since 2001. He has held positions as Assistant Professor at Polytechnic University New York (1991–1995), Assistant and Associate Professor University of Maryland College Park (1995–2001) and Professor University of Ioannina Greece (1999–2001). His research interests are in the field of computer and communication networks with emphasis on fundamental mathematical models of communication networks, architectures and protocols of wireless systems, sensor networks, high-speed internet and satellite communications. He lead several research projects in the above areas funded by the National Science Foundation, Office of Naval Research, Airforce Office of Scientific Research, Army Research Laboratory, Army Research Office in USA while currently he leads a number of projects funded by the European Commission, Information Society Technologies program. He spent time with IBM T.J.Watson research laboratory as a visiting researcher and he consults regularly with industry. He published in excess of 180 papers and holds one patent. Dr. Tassiulas received a National Science Foundation (NSF) Research Initiation Award in 1992, an NSF CAREER Award in 1995, an Office of Naval Research, Young Investigator Award in 1997, a Bodosaki Foundation award in 1999 and the INFOCOM '94 best paper award. He is a fellow of IEEE.

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 University 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 University of Toronto, the Department of Computer Science of the University of Maryland at College Park and the University 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.