High Performance, Low Complexity Cooperative Caching for Wireless Sensor Networks*

Nikos Dimokas† Dimitrios Katsaros§ Leandros Tassiulas§ Yannis Manolopoulos†
†Department of Informatics, Aristotle University, Thessaloniki, Greece
§Department of Computer & Communication Engineering, University of Thessaly, Volos, Greece
{dimokas,manolopo}@delab.csd.auth.gr {dkatsar,leandros}@uth.gr

Abstract

During the last decade, Wireless Sensor Networks (WSNs) have emerged and matured at such point that currently support several applications like environment control, intelligent buildings, target tracking in battlefields, and many more. 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 a new metric to aid in the selection of such nodes. Based on this metric, we propose a new cooperative caching protocol, which is compared against the state-of-the-art competing protocols. The simulation results attest the superiority of the proposed protocol; the proposed solution achieves on the average 20% improvement w.r.t. the competing method for the examined performance measures.

1 Introduction

Wireless Sensor Networks (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, like 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 characteristics of the sensor network, 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 not interesting for the owner of the sensor network.
- Scalability: The architecture and protocols of sensor networks must be able to scale up (or to exploit) any number of sensors.
- Data-centric networking: The target of a conventional communication network is to move bits from one machine to another, but the actual purpose of a sensor network is to provide information and answers, not numbers.

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 multihop communication. Afterwards, each sensor is able to respond to queries about "high-level" events, e.g., enemy presence.

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

978-1-4244-4439-7/09/$25.00 ©2009 IEEE

In this hostile environment, it is obvious that sensor battery recharging is not an easy/possible task. Also, the locations of the sensors will not have been decided by some placement algorithm (the sensors were left to float 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 cooperative data caching has been proposed as an effective and efficient technique to achieve these goals [12, 15, 2] (for more details cf. Section 5).

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 [8, 15], did not pay attention to the selection of nodes that will have special roles in the cooperation protocol. The work [2] 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*. 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 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 1-hop neighbors, i.e., the sensor must exchange the set of its 1-hop neighbors with each and every 1-hop neighbor; thus 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 Section 3.1 with an example).
- When the $\mathcal{NI}$ index is calculated over wider neighborhoods, and not simply for the 2-hop neighborhood of a sensor node, might change singificantly, since more connections and nodes come into play.

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

- It describes a new centrality metric for sensor nodes, the *μ-power community index* (μ–$\mathcal{PCI}$), which is more informative than the node degree, it is more robust that $\mathcal{NI}$ and it is computed faster than $\mathcal{NI}$; in addition, it is not affected by any isolated nodes.
- It develops a new cooperative caching protocol for WSNs, the PCICC (PCI Cooperative Caching); this protocol is compared against the state-of-the-art protocol via simulation analysis, which attests the superiority of the proposed protocol.

The rest of this article is organized as follows: Section 2 describes the network model, i.e., any assumptions made in the present work; in Section 3 we describe the new centrality metric and the component of the proposed cooperative caching protocol. Section 4 presents the simulation environment that was built to investigate the performance of the proposed protocol, and also describes the experiments and obtained results after the comparison of the protocol with the competing state-of-the-art schemes. In Section 5 we survey the most important works relevant to this article, and finally Section 6 conludes the article.

2 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 multihop 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 [15, 2] 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.

3 The new cooperative caching scheme

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 [2], 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 [4]. The notion of centrality is susceptible to many interpretations; the $\mathcal{NI}$ index proposed one such interpretation, with the disadvantages recorded in subsection 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 subsection 3.2 describes the components of the cooperative caching protocol.

3.1 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 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.

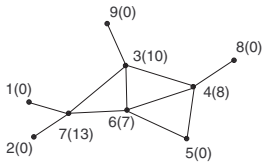


Figure 1. The $\mathcal{NI}$ indexes (the numbers in parentheses) for a small sample graph.

In some cases, the sensor degree has been used as a measure of centrality. Looking at Figure 1, we see that the nodes $3, 4, 7, 6$ are equally central w.r.t. 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 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 2-hops away). Starting from this observation, we propose a new centrality metric, called the μ-*Power Community Index* (μ–$\mathcal{PCI}$), defined as follows:

Definition 1 *The μ-Power Community Index of a sensor v is equal to k, if there are no more than $\mu * k$ sensors in the μ-hop neighborhood of v with degree equal to or greater than k, and the rest of the sensors within that region have a degree equal to or less than 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 percluded 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 other sensor nodes. But, their power depends also on the degrees of their 1-hop neighbors. Large values for the μ–$\mathcal{PCI}$ 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 $PCI(v)$ of a sensor v is equal to k, if there are no more than k 1-hop neighbors of v with degree equal to or greater than k, and the rest of the 1-hop neighbors of v have a degree equal to or less than k.*

With this definition in mind, we see in Figure 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 Figure 2.

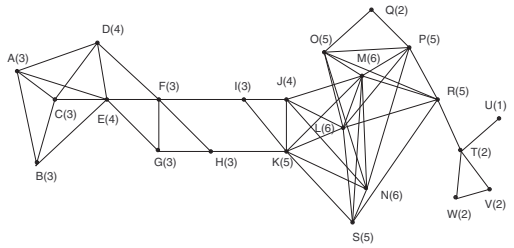


Figure 2. Calculation of $\mathcal{PCI}$ for a sample graph. Each node is characterized by a pair of ID($\mathcal{PCI}$).

Apparently, when computing the μ–$\mathcal{PCI}$ 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 μ–$\mathcal{PCI}$, 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 each sensor, in order to compute its $\mathcal{PCI}$, needs to exchange with its 1-hop neighbors only their degrees, and not detailed connectivity. Recall that the $\mathcal{NI}$ index [2] 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 NI include three steps: a) initially, each node broadcasts its ID; b) then, each sensor broadcasts the IDs of its 1-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.

3.2 The $PCICC$ protocol

This subsection describes the details of the proposed cooperative caching scheme; 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, and the cache replacement policy.

At the very first step, it is supposed that each sensor is aware of the number and identity of its 1-hop neighbors; this is achieved with the exchange of "HELLO" messages. Then, each node of the sensor network broadcasts its degree to its neighborhood. The information that is being broadcasted is only two numbers; the sensor identifier and its degree. Thus, 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 [6].

Then, every sensor calculates and broadcasts its $\mathcal{PCI}$. This is because each node needs the Power Community Indexes of neighboring nodes in order to characterize some of its neighbors as *community caching* nodes (CCNs). A sensor node v specifies its CCNs as the minimum set of its 1-hop sensors, with the larger $\mathcal{PCI}$ values, which "cover" the 2-hop neighborhood of node v.

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, and the actual data item.
- 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.

$PCICC$ uses a cache discovery algorithm (described in the next two paragraphs and exemplified in Figure 4) 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.

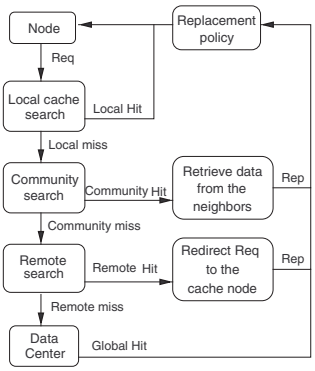


Figure 3. The $PCICC$ protocol.

When a non one-hop CCN receives a request, it searches its local cache. If it deduces that the request can be satisfied by a neighboring node (a remote cache hit), then stops the request's route toward the Data Center, and forwards the request to this neighboring node. If more than one nodes can satisfy the request, then the node with the largest residual energy is selected. If the request can not be satisfied by this CCN, then it forwards the request towards the Data Center. Figure 3 shows the behaviour of $PCICC$ 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.
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 has heard through the broadcasting operation, the TTL of this datum and the nodes that have cached this datum.

Figure 4 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.

3.2.1 The cache replacement component

$PCICC$ 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 an object, but the cache is full. In replacement operation one or more objects are evicted out of the local cache (due to providing sufficient space) and new one is cached. The $PCICC$ protocol employs the following policy:

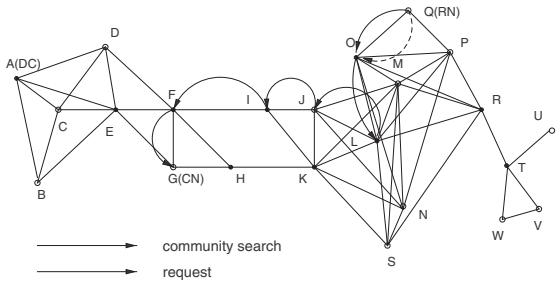


Figure 4. A request packet from node Q is forwarded to the caching node G.

A. Initially each sensor node first evicts the object 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. We have developed a value-based cache replacement function, where objects with the greatest values are those that are removed from the cache. For each cached object i the following function is calculated: $cost(i) = \frac{Lat_i * Size_i}{TTL_i * AR_i}$. When a SN gets a reply message, it calculates the incurred latency (Lat). The smaller the latency of an object 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 object. An object remains in cache when AR and TTL are big. Finally, the bigger the size of an object is, the more likely to be removed from the cache.

C. Inform the CCNs about the candidate victim. If the object is also cached by another node in the community, then CCNs transmit a delete message and the object is evicted out of the local cache. Otherwise, each CCN send a message that contains the node that has the largest residual energy and enough space to cache the object. In this case, the node purges the object and send it to the node with the largest residual energy. Finally, the CCNs update their cached metadata about the new state.

D. The node which caches this purged object, informs the CCNs with the usual broadcasting procedure.

4 Performance evaluation

We evaluated the performance of the $PCICC$ protocol through simulation experiments. We conducted a large number of experiments with various parameters, and

compared the performance of $PCICC$ to the state-of-the-art cooperative caching policy for WMSNs, namely NICoCa [2].[1] For the interest of space, we present here only the most important, representative experiments and respective results.

4.1 Simulation model

Both protocols have been implemented and evaluated with the J-Sim wireless network simulator [13]. In our simulations, the AODV [11] routing protocol is deployed to route the data traffic in the wireless sensor network. We use IEEE 802.11 as the MAC protocol and the free space model as the radio propagation model. The wireless bandwidth is 2 Mbps. The radio characteristics used in our simulations are summarized in Table 1.

Operation	Energy Dissipated
Transmitter/Receiver Electronics	$E_{elec} = 50nJ/bit$
Transmit Amplifier if $d_{toBS} \leq d_0$	$e_{fs} = 10pJ/bit/m^2$
Transmit Amplifier if $d_{toBS} \geq d_0$	$e_{mp} = 0.0013pJ/bit/m^4$
Data Aggregation	$E_{DA} = 5nJ/bit/signal$

Table 1. Radio characteristics.

The protocols has 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. 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 1KB and 10KB. 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 $n*(n-1)/2$ edges in the network by their length, in increasing order. The grid unit size corresponding to the value of d is equal to $\sqrt{2}$ times the length of the edge at position $n*d/2$ in the sorted sequence. Two sensor nodes are neighbors if they placed in the same grid or in adjacent grids. The simulation area is assumed of size $500m \times 500m$ and is divided into equal sized square grid units. Beginning with the lower grid unit, the units are named as $1, 2, \ldots$, in a column-wise fashion.

The client query model is similar to what have been used in previous studies [2, 15]. 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). The sensors residing in neighboring grids (25 grids with size $100m \times 100m$) have the same access pattern. 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 $\theta = 0.8$.

Similar to [2, 15], 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.

Parameter	Default value	Range
# items (N)	1000	
S_{min} (KB)	1	
S_{max} (KB)	10	
# nodes (n)	500	100–1000
Bandwidth (Mbps)	2	
Waiting interval (t_w)	10 sec	
Cache size (KB)	800	200 to 1200
Zipfian skewness (θ)	0.8	0.0 to 1.0

Table 2. Simulation parameters.

4.2 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 latency is the time elapsed between when the query is sent and 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. Because the number of messages due to the routing scheme is the same for

[1] In [2], the NICoCa protocol was compared against the Hybrid caching scheme [15], for many data/request distributions and many network topologies, and NICoCa proved superior in all cases.

both 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. 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 vanishes the benefits of local hits.

4.3 Evaluation

We performed a large number of experiments varying the size of the sensornet (in terms of the number of its sensor nodes), varying the access profile of the sensor nodes, and the cache size relative to the aggregate size of all data items. In particular, we performed experiments for cache size equal to 1%, to 5% to 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 $4, 7$ (sparse and moderate dense sensornet, respectively) 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, which does not corresponds to the usual time metric, but to an internal simulator clock. For the interest of space, we present only a representative set of the results, shown in Figures 5 – 8.

The first graph in each of these figures depicts the efficiency of 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 PCICC w.r.t. the global hits achieved by NICoCa; the line corresponding to Remote Hits, represents the (percentage) increase in remote hits achieved by PCICC w.r.t. the remote hits achieved by NICoCa; the line corresponding to Local Hits, represents the (percentage) increase in local hits achieved by PCICC w.r.t. the local hits achieved by NICoCa*.

As expected, both 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 PCICC protocol is superior to its competitor for all data/network 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.

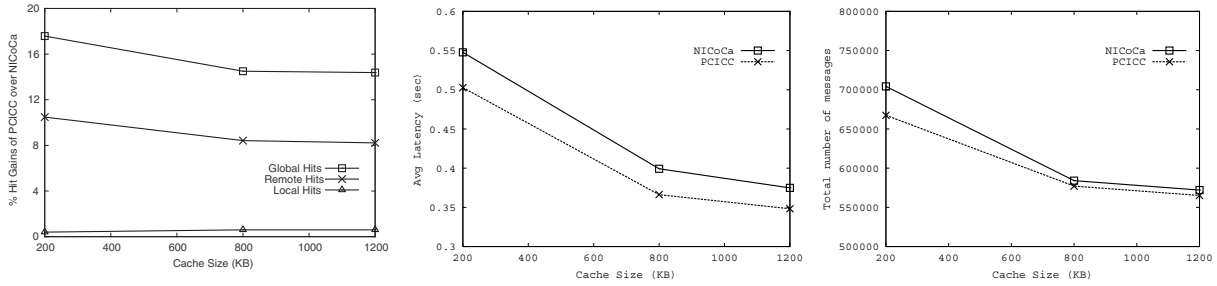
It is interesting to note that PCICC achieves an average of 20% fewer global hits than NICoCa, and around 13% more remote hits. The reduction in global hits and simultaneous increase in remote hits proves that PCICC achieves a more successful cooperation, and this is directly attributed to the seleciton 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 have similar performance, but the different admission policy adopted by PCICC provides a slightly better response for it.

In general, the performance of each protocol gets better in dense sensornets, since we constrain more sensor nodes 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.

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 Figures 5(b), 6(b), 7(b) and 8(b). The dominant observation is that 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 $PCICC$ is 10% to 16% 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.

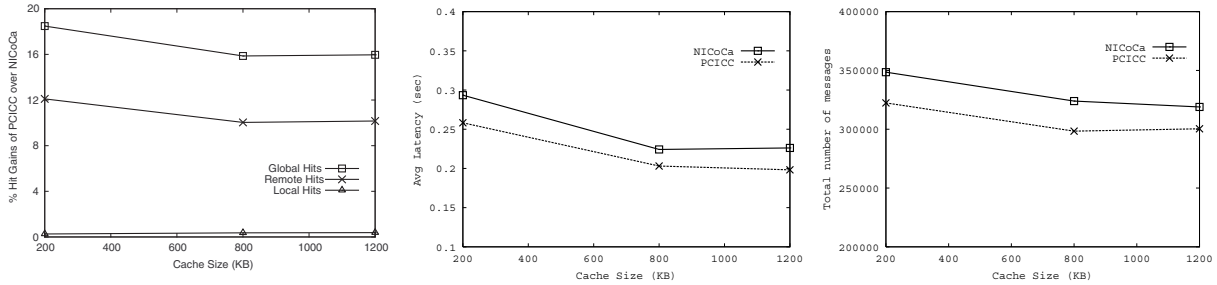
Finally, we evaluated the performance of the algorithms with respect to the number of transmitted messages for varying cache sizes for both sparse and dense sensor network deployments. The results are depicted in Figures 5(c), 6(c), 7(c) and 8(c). The relative results follow the same trends that we observed 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 impact in smaller energy consumption.

In summary, for all network topologies $PCICC$ achieves more remote hits and less global hits than NICoCa. This performance gap is slightly better in favor of $PCICC$ as we move from sparse to denser WSN. Finally, both protocols achieve significant performance gains for skewed access pattern ($\theta = 0.8$).



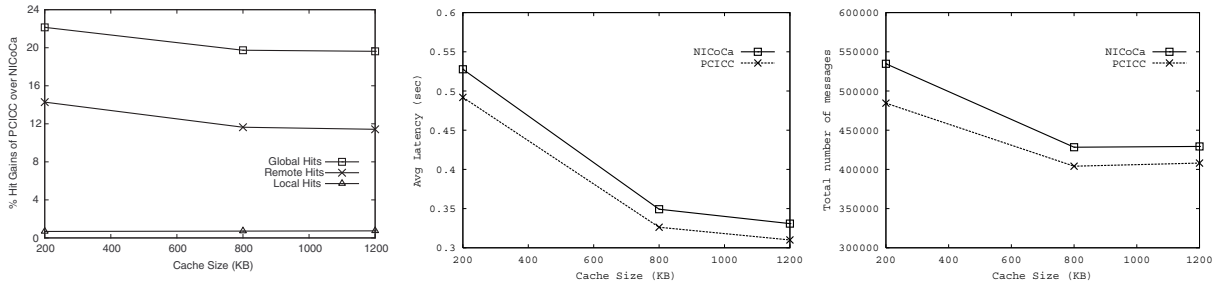
(a) (b) (c)

Figure 5. Impact of cache size on: (a) hits, (b) latency, (c) number of messages in a sparse WSN ($d = 4$) with $\theta = 0.0$.



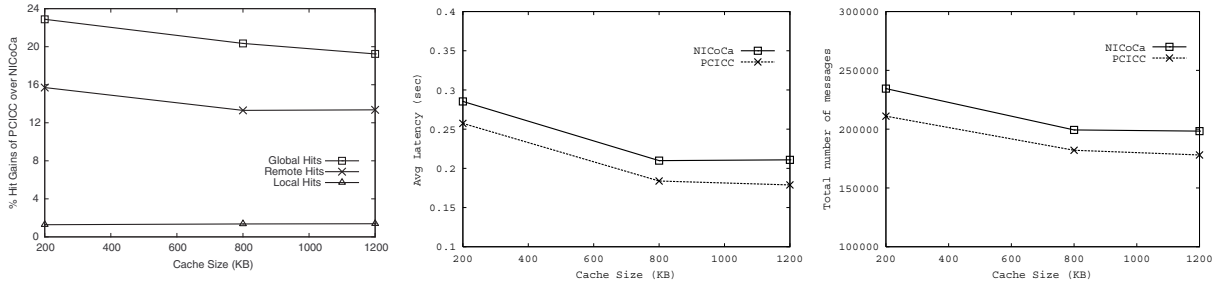
(a) (b) (c)

Figure 6. Impact of cache size on: (a) hits, (b) latency, (c) number of messages in a dense WSN ($d = 10$) with $\theta = 0.0$.



(a) (b) (c)

Figure 7. Impact of cache size on: (a) hits, (b) latency, (c) number of messages in a sparse WSN ($d = 4$) with $\theta = 0.8$.



(a) (b) (c)

Figure 8. Impact of cache size on: (a) hits, (b) latency, (c) number of messages in a dense WSN ($d = 10$) with $\theta = 0.8$.

5 Relevant work

Cooperative caching has attracted significant attention in the literature concerning various types of distributed systems; in the Web [5], in file servers [1], in cellular networks [7], and so on. Nevertheless, the very limited

capabilities of the sensor nodes (in terms of energy, storage, and computation), the particularities of the wireless channel (varible capacity), and the multi-hop fashion of communication, turns the solutions proposed in the aforementioned environments, of limited usefulness.

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. [10] addressed the problem of energy-conscious cache placement in wireless ah hoc network, and [14] 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. Though these works adress cache placement issues. The most recent important relevant works are those reported in [3, 9, 15, 8, 2]. Cooperative caching strategies but in hybrid mobile ad hoc networks (i.e., with the presence of Access Points) are described in [9]. 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 [3], but clearly such approaches are unscalable and not practical (who/how to set the many parameters?). The work reported in [8] 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 1-hop neighbors (otherwise forwards the request to a ficticious data center). Thus, remote hits can not happen take in this protocol. Yin & Cao proposed the Hybrid cooperative caching protocol, which exploited both data and node locality in an homogeneous manner, but this policy was proved inferior to NICoCa, described in [2] which took special consideration to select appropriate "central" nodes to carry out and coordinate the cooperation.

6 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 in 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 a new centrality metric to aid in the selection of such nodes, and proposed a new cooperative caching protocol (PCICC), which, based on simulation analysis, proved superior to the state-of-the-art competing protocol.

References

[1] S. Annapureddy, M. J. Freedman, and D. Mazières. Shark: Scaling file servers via cooperative caching. In *Proceedings of USENIX NSDI*, pages 129–142, 2005.

[2] N. Dimokas, D. Katsaros, and Y. Manolopoulos. Cooperative caching in wireless multimedia sensor networks. *ACM Mobile Networks and Applications*, 13(3–4):337–356, 2008.

[3] Y. Du and K. S. Gupta. COOP: A cooperative caching service in MANETs. In *Proceedings of ICAS-ICNS*, pages 58–63, 2005.

[4] V. Erramilli, M. Crovella, A. Chaintreau, and C. Diot. Delegation forwarding. In *Proceedings of ACM MobiHoc*, pages 251–259, 2008.

[5] L. Fan, P. Cao, and A. Z. Almeida, J. M. Broder. Summary cache: A scalable wide-area Web cache sharing protocol. *IEEE/ACM Transactions on Networking*, 8(3):281–293, 2000.

[6] R. Gandhi and S. Parthasarathy. Fast distributed well connected dominating sets for ad hoc networks. Technical Report CS-TR-4559, Computer Science Department, University of Maryland at College Park, 2004.

[7] T. Hara. Cooperative caching by mobile clients in push-based information systems. In *Proceedings of ACM CIKM*, pages 186–193, 2002.

[8] W. Li, E. Chan, and D. Chen. Energy-efficient cache replacement policies for cooperative caching in mobile ad hoc network. In *Proceedings of the IEEE WCNC*, pages 3349–3354, 2007.

[9] S. Lim, W.-C. Lee, G. Cao, and C. R. Das. A novel caching scheme for improving internet-based mobile ad hoc networks performance. *Ad Hoc Networks*, 4(2):225–239, 2006.

[10] P. Nuggehalli, V. Srinivasan, and C.-F. Chiasserini. Energy-efficient caching strategies in ad hoc wireless networks. In *Proceedings of ACM MobiHoc*, pages 25–34, 2003.

[11] C. E. Perkins and E. Royer. Ad hoc On-demand Distance Vector routing. In *Proceedings of the IEEE Workshop on Mobile Computing Systems and Applications*, pages 90–100, 1999.

[12] H. Shen, S. K. Das, M. Kumar, and Z. Wang. Cooperative caching with optimal radius in hybrid wireless networks. In *Proceedings of the International IFIP-TC6 Networking Conference (NETWORKING)*, volume 3042 of *Lecture Notes on Computer Science*, pages 841–853, 2004.

[13] A. Sobeih, J. C. Hou, L.-C. Kung, N. Li, H. Zhang, W.-P. Chen, H.-Y. Tyan, and H. Lim. J-Sim: A simulation and emulation environment for wireless sensor networks. *IEEE Wireless Communications magazine*, 13(4):104–119, 2006.

[14] B. Tang, H. Gupta, and S. R. Das. Benefit-based data caching in ad hoc networks. *IEEE Transactions on Mobile Computing*, 7(3):289–304, 2008.

[15] L. Yin and G. Cao. Supporting cooperative caching in ad hoc networks. *IEEE Transactions on Mobile Computing*, 5(1):77–89, 2006.