

Book Title: Cooperative Wireless Communications

Yan Zhang, H. H. Chen and M. Guizani

April 5, 2008

Contents

- 1 Cooperative Caching in WMSNs 1
 - 1.1 Introduction 2
 - 1.2 Relevant work 5
 - 1.2.1 Caching in the Web 5
 - 1.2.2 Caching in wireless cellular networks 6
 - 1.2.3 Caching in MANETs and WSNs 6
 - 1.2.4 Why a new caching protocol? 9
 - 1.3 The *NICoCa* cooperative caching protocol for WMSNs 10
 - 1.3.1 Measuring sensor node importance 11
 - 1.3.2 Housekeeping information in the *NICoCa* protocol 12
 - 1.3.3 The cooperative cache discovery component protocol 14
 - 1.3.4 The cache replacement component protocol 16
 - 1.4 Performance evaluation 17
 - 1.4.1 Simulation model 17

| | | |
|-------|-------------------------------|----|
| 1.4.2 | Performance metrics | 19 |
| 1.4.3 | Evaluation | 19 |
| 1.5 | Conclusion | 22 |

Chapter 1

Cooperative Caching in Wireless Multimedia Sensor Networks

Nikos Dimokas, Dimitrios Katsaros, Yannis Manolopoulos

ABSTRACT. The advancement in wireless communication and electronics has enabled the development of low-cost wireless sensor networks. Especially, the production of cheap CMOS cameras and microphones, which are able to capture rich multimedia content, have fueled a new research and development area, that of *Wireless Multimedia Sensor Networks (WMSNs)*. WMSNs will boost the capabilities of current wireless sensor networks, and several novel applications, like multimedia surveillance sensor networks, storage of relevant activities and so on will be developed. WMSNs introduce several new research challenges, mainly related to mechanisms to deliver application-level Quality-of-Service (e.g., latency minimization). Such issues have almost completely been ignored in traditional WSNs, where the research focused on energy consumption minimization.

To address this goal in an environment with extreme resource constraints, with variable channel capacity and with requirements for multimedia in-network processing, the efficient and effective caching of multimedia data, exploiting the cooperation among sensor nodes is vital. This chapter describes a cooperative caching scheme particularly suitable for WMSNs. The presented scheme has been evaluated extensively in an advanced simulation environment, and it has been compared to the state-of-the-art cooperative caching algorithm for mobile ad hoc networks.

1.1 Introduction

Wireless Sensor Networks [2, 18] (WSNs) have emerged during the last years due to the advances in low-power hardware design and the development of appropriate software, that enabled the creation of tiny devices which are able to compute, control and communicate with each other. A wireless sensor network consists of wirelessly interconnected devices that can interact with their environment by controlling and sensing “physical” parameters. WSNs attracted a huge interest from both the research community and the industry, that continues to grow. This growing interest can be attributed to the many new exciting applications that were born as a result of the deployment of large-scale WSNs. Such applications range from disaster relief, to environment control and biodiversity mapping, to machine surveillance, to intelligent building, to precision agriculture, to pervasive health applications, and to telematics.

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

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

The production of cheap CMOS cameras and microphones, which can acquire rich media content from the environment, created a new wave into the evolution of wireless sensor networks. For instance, the Cyclops imaging module [30] is a light-weight imaging module which can be adapted to MICA2 (<http://www.xbow.com>) or MICAz sensor nodes. Thus, a new class of WSNs came to the scene, the *Wireless Multimedia Sensor Networks (WMSNs)* [1]. These sensor networks, apart from boosting the existing application of WSNs, will create new applications: a) multimedia surveillance sensor networks which will be composed by miniature video cameras [22] will be able to communicate, to process and store data relevant to crimes and terrorist attacks; b) traffic avoidance and control systems will monitor car traffic and offer routing advices to prevent congestion; c) industrial process control will be realized by WMSNs that will offer time-critical information related to imaging, temperature, pressure, etc.

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

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

- Resource constraints: sensor nodes are battery-, memory- and processing-starving devices.
- Variable channel capacity: the multi-hop nature of WMSNs, which operate in a store-and-forward fashion because of the absence of base stations, implies that the capacity of each wireless link depends on the interference level among nodes, which is aggravated by the broadcasting operations.

- Multimedia in-network processing: In many applications of WMSNs, a single sensor node is not able to answer a posed question, but several sensors must collaborate to answer it. For instance, a sensor node with a camera monitoring a moving group of people, can not count their exact number and determine their direction, but it needs the collaboration of nearby sensors in order to cover the whole extent of the group of people. Therefore, sensor nodes are required to store rich media, e.g., image, video, needed for their running applications, and also to retrieve such media from remote sensor nodes with short latency.

Under these restrictions/requirements, the goal of achieving application-level QoS in WMSNs becomes a very challenging task. There could be several ways to attack parts of this problem, e.g., channel-adaptive streaming [14], joint source-channel coding [11]. Though, none of them can provide solutions to all of the three aforementioned issues. In this chapter, the solution of *cooperative caching multimedia content* in sensor nodes is being investigated in order to address all three characteristics. In cooperative caching, multiple sensor nodes share and coordinate cache data to cut communication cost and exploit the aggregate cache space of cooperating sensors.

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

This chapter investigates the technique of caching in the context of WMSNs. The need for effective and intelligent caching policies in sensor networks has been pointed out several times [10, 25] in the literature, but no appropriate sophisticated policies have been proposed, although there are quite a lot of caching protocols in other fields (see relevant work in Section 1.2). This chapter presents also a novel and high-performance cooperative caching

protocol, the *NICoCa* protocol named after the words *Node Importance-based Cooperative Caching*, and the comparison with the state-of-the-art cooperative caching policy for mobile ad hoc networks, which is the “closer” competitor. An experimental evaluation of the two methods is being exhibited at the end of chapter.

The rest of this chapter is organized as follows: in Section 1.2 the solutions proposed so far in the area of cooperative caching in WSNs and MANETs are being described and the benefits achieved in terms of communication cost, query latency, energy dissipation and network lifetime prolongation are being investigated. The details of the *NICoCa* protocol are being presented in Section 1.3, and the results of the performance evaluation of the methods are being exhibited in Section 1.4; finally, Section 1.5 concludes the chapter.

1.2 Relevant work

1.2.1 Caching in the Web

The technique of caching has been widely investigated in the context of operating systems and databases and is still an attractive research area [24]. Similarly, caching on the Web has been thoroughly investigated for cooperative [12] and for non-cooperative [19, 26] architectures. Wessels and Claffy [39] introduced the Internet cache protocol (ICP) to support communication between caching proxies by using message exchange. Cache digests [31] and summary cache [12] enable proxies to exchange information about cached content. In [6] a cooperative hierarchical Web caching architecture was studied. However, the problem addressed in wireless networks is different from that in wired networks. The above architectures and protocols usually assume a fixed network topology and require powerful computation and communication capabilities. Due to the constraint resources (i.e., bandwidth, battery power and computing capacity) of sensor nodes, these techniques will not adapt well to the wireless sensor network.

1.2.2 Caching in wireless cellular networks

In the context of wireless broadcast cellular networks have been proposed a number of caching approaches [20, 38]. These policies assume more powerful nodes than the sensor nodes, and one-hop communication with resource-rich base stations, which serve the needed data.

1.2.3 Caching in MANETs and WSNs

A number of data replication schemes [15, 16] and caching schemes [32, 36, 40] have been proposed in order to facilitate data access in mobile ad hoc networks (MANETs). Data replication studies the issue of allocating replicas of data items to meet access demands. These techniques normally require a priori knowledge of the network topology.

Caching schemes however do not facilitate data access based on the knowledge of distributed data items. In SimpleCache [40] the requested data item has always been cached by the requester node. The node use the cached copy in order to serve subsequent requests when they arrive. The requester node has to get the data from the data center in case of cache miss. However increasing the hop distance between the requester node and caching node will increase the response time for the request.

In the research area of mobile ad hoc networks have been developed a number of caching protocols. The proposed caching protocol exploit the cooperation between mobile caches in order to decrease query latency and energy dissipation. The main motive for the development of these protocols is the mobility of the nodes, and thus they all strive to model it or exploit it. A cooperative caching scheme, called CoCa, was proposed in [8, 7]. The CoCa framework facilitate mobile nodes to share their cached contents with each other in order to reduce the number of server requests and the number of access misses in a single hop wireless mobile network. The authors extended CoCa with a group-based cooperative caching scheme, called GroCoCa, in [9]. According to GroCoCa, the decision of whether a data item should be cached depends on two factors of the access affinity on the data items and the mobility of each node. The mobile support station performs an incremental clustering algorithm to cluster

the mobile nodes into tightly coupled groups based on their mobility patterns. In GroCoCa also the similarity of access patterns is captured by frequency-based similarity measurement. GroCoCa improves system performance at the cost of extra power consumption. Papadopouli and Schulzrinne [27] suggested the 7DS architecture. The authors deployed a couple of protocols in order to facilitate sharing and dissemination of information among users. It operates on two modes. The first one is a prefetch mode, based on the information and user's future needs and the second one is an on-demand mode, which searches for data items in a one hop multicast basis. Depending on the collaborative behavior, a peer-to-peer (P2P) and server-to-client mode are used. This strategy focuses also on single-hop wireless environment and on data dissemination. Sailhan and Issarny [33] proposed a collaborative cache management strategy among mobile terminals interacting via an ad hoc network. The issue that the authors addressed was on setting an ad hoc network of mobile terminals that cooperate to exchange Web pages. The proposed solution aims at improving the Web latency on mobile terminals while optimizing associated energy consumption. It is implemented on top of Zone Routing Protocol (ZRP). The authors proposed a fixed broadcast range based on the underlying routing protocol.

The Zone Cooperative [5], the Cluster Cooperative [4] and the ECOR [34] protocols attempts to form clusters of nodes based either in geographical proximity or utilizing widely known node clustering algorithms for MANETs [3]. In Zone Cooperative (ZC), mobile nodes belonging to the neighborhood (zone) of a given node form a co-operative cache system for this node since the cost for communication with them is low both in terms of energy consumption and message exchanges. Each node has a cache to store the frequently accessed data items. The data items in the cache satisfy not only the node's own requests, but also the data requests passing through it from other nodes. For a data miss in local cache, the node first searches the data in its zone before forwarding the request to the next node that lies on a path towards the data center. As a part of cache management, a value-based replacement policy based on popularity, distance, size and time-to-live was developed to improve the data accessibility and reduce the local cache miss ratio. Simulations experiments revealed improvements in cache hit ratio and average query latency in comparison with other caching strategies. In Cluster Cooperative (CC), the authors present a scheme for caching in mobile ad hoc networks. The goal of CC is to reduce the cache discovery overhead and provide better

cooperative caching performance. The authors partition the whole MANET into equal size clusters based on the geographical network proximity. In each cluster, CC dynamically chooses a “super” node as cache state node, to maintain the cluster cache state information of different nodes within its cluster domain. The cache state node is defined as the first node that enters the cluster. The cluster cache state for a node is the list of cached data items along with their time-to-live field. The cache replacement policy is similar to that in ZC. In ECOR, each mobile node forms a cooperation zone (CZ) with mobile nodes in proximity by exchanging messages to share their cached data items in order to minimize bandwidth and energy cost for each data retrieval. When a data request arrives, the node first searches the data in its CZ before forwarding the request to the data center. 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 broadcast every modification of the cached data items to nodes that belong to cooperation zone. Each node maintains a cache hint table for the cache information of all nodes in its proximity.

The only protocols that tried to exploit both data and node locality in an homogeneous manner are described in [40] and are the following: CachePath, CacheData, and Hybrid-Cache. In CacheData, intermediate nodes may cache data to serve future requests instead of fetching data from the “Data Center”. An intermediate node caches passing by data item locally when it finds that data item is popular and does not cache the data item if all requests for it are from the same node. This rule is designed in order to reduce the cache space requirement because the mobile nodes have limited cache spaces. In CachePath, a mobile node may cache the information of a path to a nearby data requester while forwarding the data and use the path information to redirect future requests to the nearby caching site. By caching the data path for each data item, bandwidth and the query delay can be reduced since the requested data can be obtained through fewer number of hops. The authors proposed also some optimizations techniques. An intermediate node can save only the destination node information because the path from the current router to the destination node can be obtained by the underlying routing protocol. Also, the intermediate node need to record the data path when it is closer to the caching node than the data center. One major drawback of CachePath according to the authors is that the cached path may not be reliable

and using it may increase the overhead. A cached path may not be reliable because either the data item has become obsolete or the caching node can not be reached. The hybrid protocol HybridCache combines CacheData and CachePath while avoiding their weaknesses. In HybridCache, when a mobile node forwards a data item, it caches the data or the path based on some criteria. These criteria include the data item size, the time-to-leave of the data item and the number of hops that a cached path can save and denoted as H_{save} . H_{save} value is the difference between the distance to the data center and the distance to the caching node. H_{save} must be greater than a system tuning threshold. However, according to these methods the caching information of a node cannot be shared if the node does not lie on the path between the data requester and the data source. Moreover, the threshold values used in these heuristics must be set carefully in order to achieve good performance.

The only works on caching in wireless sensor networks concern the placement of caches [29, 37] and thus they are not examined to this chapter. Remotely related to the topic of this chapter are the caching policies for MANET routing protocols [41].

1.2.4 Why a new caching protocol?

The protocols proposed so far for cooperative caching in MANETs present various limitations. Those protocols, which first perform a clustering of the network and then exploit this clustering (the cluster-heads, (CH)), in order to coordinate the caching decisions, inherit the shortcomings of any CH selection. For instance, in [5, 4], the nodes which form the cluster are assumed to reside within the same communication range, i.e., they are with on-hop distance from the other nodes of the cluster. Additionally, the nodes do not cache the data originating from an one-hop neighbor. Thus, CHs which do not reside in a significant part of data routes can not serve efficiently their cluster members, because they do not have fast access (short latency) to requested data. The cooperation zone which is formed in [34] by selecting an optimal radius, implies a large communication overhead, because every node within that radius must send/receive any changes to the caches of the other nodes within the radius. Finally, the HybridCache policy is tightly coupled to the underlying routing protocol, and thus if a node does not reside in the route selected by the routing protocol

can not cache the data/path, or conversely, can not serve the request even if it holds the requested data.

This chapter presents a cooperative caching policy that taking into account the unique requirements of the WMSNs, which are mainly static and not mobile and tries to avoid the shortcomings of the current cooperative protocols. The cooperative caching policy is based on the idea of exploiting the sensor network topology, so as to discover which nodes are more important than the others, in terms of their position in the network and/or in terms of residual energy. Incorporating both factors into the design of the caching policy the authors ensure both network longevity and short latency in multimedia data retrieval. In summary, the core features that is being presented in this chapter are the following:

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

1.3 The *NICoCa* cooperative caching protocol for WMSNs

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

among neighboring nodes and the energy balancing of them, is managed by mediator nodes. The mediator nodes include indices about the cached data in neighborhood, the remaining energy and the free cache space of its neighboring node. Thus, whenever a data request reach a mediator node, it decides which neighboring node with the highest remaining energy could answer it. Mediator nodes took place also during the cache replacement phase that happened in a neighboring node. The cooperation between the node that replaces a cached data and the mediators is taken place in order not to permanent evict the data item from the neighborhood. According to the free cache space and the cached data items that neighboring nodes have, mediator nodes decide where to place the replaced cache data. If the replaced data item exists somewhere else in neighborhood, then it is simple deleted.

1.3.1 Measuring sensor node importance

A wireless multimedia sensor network is abstracted as a graph $G(V, E)$, where V is the set of its nodes, and E is the set of radio connections between the nodes. An edge $e = (u, v)$, $u, v \in E$ exists if and only if u is in the transmission range of v and vice versa. All links in the graph are bidirectional, i.e., if u is in the transmission range of v , v is also in the transmission range of u . The network is assumed to be in a connected state. The set of neighbors of a node v is represented by $N_1(v)$, i.e., $N_1(v) = \{u : (v, u) \in E\}$. The set of two-hop nodes of node v , i.e., the nodes which are the neighbors of node v 's neighbors except for the nodes that are the neighbors of node v , is represented by $N_2(v)$, i.e., $N_2(v) = \{w : (u, w) \in E, \text{ where } w \neq v \text{ and } w \notin N_1 \text{ and } (v, u) \in E\}$. The combined set of one-hop and two-hop neighbors of v is denoted as $N_{12}(v)$. **Definition** [Local network view of node v]. The local network view, denoted as LN_v , of a graph $G(V, E)$ w.r.t. a node $v \in V$ is the *induced subgraph* of G associated with the set of vertices in $N_{12}(v)$.

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

latency), but it is a purely abstract metric measuring the number of hops.

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

$$\mathcal{NI}(v) = \sum_{u \neq v \neq w \in V} \frac{\sigma_{uw}(v)}{\sigma_{uw}}. \quad (1.1)$$

Large values for the \mathcal{NI} index of a node v indicate that this node v can reach others on relatively short paths, or that the node v lies on considerable fractions of shortest paths connecting others. Illustration of this metric is presented in Figure 1.2 (with well formed and vague node clusters).

A very informative picture of which nodes reside in a large number of shortest paths between other nodes is being obtained when estimating the \mathcal{NI} index for each sensor node using the whole network topology. The picture about the relative importance of the nodes remains also very accurate, even when the \mathcal{NI} indexes of the nodes is being calculated taking into account only their k -hop ($k = 2$ or 3) neighborhood. For $k = 1$ the \mathcal{NI} index of a sensor node is equivalent to its degree. For more information concerning the calculation and the use of this metric in broadcasting protocols the interested user can consult the work reported in [21].

1.3.2 Housekeeping information in the *NICoCa* protocol

W.l.o.g. and adopting the model presented in [40], the cooperative caching protocol assume that the ultimate source of multimedia data is a *Data Center*. This is not restrictive at all and simply guarantees that every request, if it is not served by other sensor nodes and if does not expire, will finally be served by the Data Center.

Firstly, it is assumed that each node is aware of its 2-hop neighborhood. This information is obtained through periodic exchange of “beacon” messages. It is also considered 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 [13]. Then, every node calculates the \mathcal{NI} index of its 1-hop neighbors. The node uses this information in order to characterize some of its neighbors as *mediator* nodes; the minimum set of neighbors with the larger \mathcal{NI} which “cover” its 2-hop neighborhood are the mediator nodes for that node; The node is responsible for notifying its neighbors about which of them are its mediators. Thus, a node can be either a mediator or an ordinary node.

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

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

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

1.3.3 The cooperative cache discovery component protocol

When a sensor node issues a request for a multimedia item, it searches its local cache. If the item is found there (a local cache hit) then the K most recent access timestamps are updated. Otherwise (a local cache miss), the request is broadcasted and received by the mediators. If none of them responds (a “proximity” cache miss), then the request is directed to the Data Center.

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

Based on the above idea, we describe the cache discovery protocol and the cooperation among nodes, that is taking place, in order to determine the path to the sensor node having the requested item or to the data center. For example, suppose that sensor node SN_i in Figure 1.1 issue a request for data item x that is placed in data center DC_1 and has been cached by sensor nodes SN_g and SN_h . The shaded nodes are considered to be the mediator nodes of the sensor network. In the beginning sensor node SN_i searches its own cache. If it deduces that data item is not available in local cache, it sends a proximity search request in neighboring mediator nodes SN_a and SN_b . Upon receiving the search request, each mediator searches in the proximity cache table. If data item found, each mediator replies with an index packet that contains the dataID and the remaining energy of the sensor node that has the uppermost battery power and has cached the data item. SN_i , upon the receipt of index packets, selects the sensor node that has the smallest energy consumption and sends the

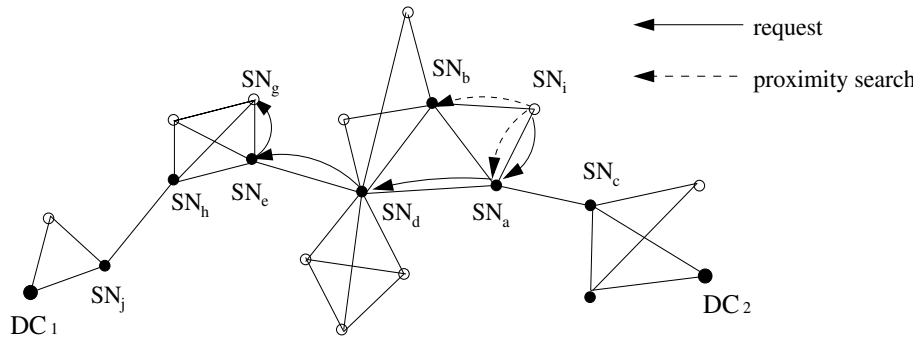


Figure 1.1: A request packet from sensor node SN_i is forwarded to the caching node SN_g .

request packet. Caching node responds with a reply packet containing the requested data item.

If no neighboring sensor node is caching the data item, SN_i sends a request packet to the data center DC_1 , as shown in Figure 1.1. When SN_x ($x \in \{d, e\}$) receives a request packet, it searches in local cache and in proximity cache table. If the data item is not found, SN_x forwards the request through the path to the DC_1 . When sensor node SN_e found that the requested data item has been cached by some neighboring nodes, it chooses the node that has the smallest energy dissipation and redirects the request packet to the caching node. The caching node sends a reply packet containing the data item x along the routing path until it reaches the original requester. Once the requester node receives the data item, it notifies its one-hop mediators about the new caching item by sending an index packet containing item's dataID. In case of not enough cache capacity, it triggers the cache replacement protocol to determine the data items that should be evicted from the cache.

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

1. Local hit (LH): the requested datum is cached by the node which issued the request. If this datum is valid (the TTL has not expired) then the *NiCoCa* is not executed.
2. “Proximity” hit (PH): the requested datum is cached by a node in the 2-hop neighborhood of the node which issued the request. In this case, the mediator(s) return to the requesting node the “location” of the node which stores the datum.

3. Remote hit (RH): the requested datum is cached by a node and this node has at least one mediator residing along the path from the requesting node to the Data Center.
4. Global hit (GH): the requested datum is acquired from the Data Center.

1.3.4 The cache replacement component protocol

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

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

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

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

STEP 4. The node which caches this purged datum, informs the mediators with the usual broadcasting procedure and the cached item is characterized as H (i.e., hosted).

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

1.4 Performance evaluation

The performance of the *NICoCa* protocol has been evaluated through simulation experiments. A large number of experiments with various parameters, and a comparison between the performance of *NICoCa* with the state-of-the-art cooperative caching policy for MANETs, namely *HybridCache* [40] is being presented in this chapter.

1.4.1 Simulation model

A simulation model based on the J-Sim simulator [35] has been developed. In the simulations, the AODV [28] routing protocol is deployed to route the data traffic in the wireless sensor network. Also, IEEE 802.11 has been chosen as the MAC protocol and the free space model as the radio propagation model. The wireless bandwidth was 2 Mbps.

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

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

| Parameter | Default value | Range |
|-------------------------------|---|-------------|
| # items (N) | 1000 | |
| S_{min} (KB) | 1 | |
| S_{max} (KB) | 10 | |
| S_{min} (MB) | 1 | |
| S_{max} (MB) | 5 | |
| # requests per node | 250 | 200–300 |
| # nodes (n) | 500 | 100–1000 |
| Bandwidth (Mbps) | 2 | |
| Waiting interval (t_w) | 10 sec for items with KB size 100 sec for items with MB size | |
| Client cache size (KB) | 800 | 200 to 1200 |
| Client cache size (MB) | 125 | 25 to 250 |
| Zipfian skewness (θ) | 0.8 | 0.0 to 1.0 |

Table 1.1: Simulation parameters.

The client query model is similar to what have been used in previous studies [40]. 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 (t_w) before sending a new query. The access pattern of sensor nodes is: a) location independent, that is, sensor nodes decide independently the data of interest; each sensor node generates accesses to the data following the uniform distribution, and b) Zipfian with $\theta = 0.8$, where groups of nodes residing in neighboring grids (25 grids with size $100m \times 100m$) have the same access pattern. The protocols have been tested both for Zipfian access pattern and for uniform access pattern. For the case of the Zipfian access, the experiments were conducted with varying θ values between 0.0 and 1.0.

Similar to [40], two data centers are placed at opposite corners of the simulation area. Data Center 1 is placed at point ($x = 0, y = 0$) and Data Center 2 is place at point ($x = 500, y = 500$). There are $N/2$ data items in each data center. Data items with even ids are stored at Data Center 1 and data items with odd ids are stored at Data Center 2. The size of each data item is uniformly distributed between s_{min} and s_{max} . The data items are considered static, i.e., not updated. The data centers serve the queries on a FCFS (first-come-first-served) basis. The system parameters are listed in Table 1.1.

1.4.2 Performance metrics

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

1.4.3 Evaluation

A large number of experiments was performed by varying the size of the sensor network (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, experiments were performed for 100, 500, and 1000 sensors, for cache size equal to 1%, to 5% to 10% of the aggregated size of all distinct multimedia data, for access pattern with θ equal to 0.0 (uniform access pattern) to 1.0 (highly skewed access pattern), for average sensor node degree equal to 4, 7 (very sparse and sparse sensor network) and 10 (dense sensor network), and for data item size equal to a few kilobytes (KB) and also equal to a few megabytes (MB). For each different setting, the performance measure includes the number of hits (local, remote, global), the latency and the message overhead. (The latency is measured in seconds, which does not correspond to the usual time metric, but to an internal simulator clock.) The remote hits comprise of proximity hits and remote hits. For *HybridCache* scheme, proximity hits is always zero. Proximity hits determine the number of hits that are generated when a sensor node inside requester node's two-hop neighborhood responds to the request. In the sequel of the chapter we will present only a representative set of the results, since there are many of independent parameters and three dependent performance metrics. The graphs were partitioned in two large groups w.r.t. whether they deal with small KB-sized files or large MB-sized multimedia files.

Experiments with MB-sized data items

The purpose of a first set of experiments was to investigate the performance of the caching algorithms when they have to deal with large multimedia files, e.g., video files, queried by the sensor network.

All figures show that both schemes exhibit better average query latency, hit ratio (local, remote and global) and message overhead when varying the cache size from 25MB to 250MB. This is because more required data items can be found in the local cache as the cache gets larger. Figures 1.3 and 1.4 show the number of hits achieved by the two protocols for a small (sparse and dense, respectively) sensor network for both uniform and skewed access patterns. The *NICoCa* scheme achieves always higher number of remote hits than *HybridCache* scheme because the chances of some neighboring nodes tuning in the required data item is higher due to cooperative caching. When the cache size is small, more required data can be found in local and *proximity* cache for *NICoCa* scheme as compared to *HybridCache* scheme which utilizes only the local cache, thus alleviating the need for remote and global cache access. It is worth noting that *NICoCa* always reaches the near optimum performance when the cache size is equal to 125MB. This demonstrates the low cache space requirement.

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

are straightforwardly reflected to the access latency incurred by the algorithms (Figures 1.5 and 1.6).

NiCoCa achieves low average query latency as compared to *HybridCache*. Nodes can access most of the required data items from local and *proximity* cache, so reducing the query latency. When the cache size increases the average query latency decreases for both schemes. That's happens because more data are been cached near to requester nodes. In this case, *NiCoCa* also outperforms over *HybridCache*. The performance of *NiCoCa* improves as the node density increases, while the performance of *HybridCache* maintains the same or slightly better in some cases. With an increase in node density, the size of each proximity region increases and thus the requested data item could be retrieved from the neighboring nodes than from data center or remote caches. This results in decreasing the average query latency.

When we move to larger sensornets with 500 (Figure 1.7) and with 1000 nodes (Figure 1.10), the performance gains of the *NiCoCa* caching algorithm in terms of hits is still evident, but the results are not so impressive, because more sensor nodes are dispersed in the same geographical region, thus creating replicas of the same data. This performance gain is reflected to the access latency as well (Figures 1.8). Moving to larger sensornets, the latence gradually increases, because the denser deployment (more nodes in the same region) has a negative effect on the efficiency of communication, aggravating the collisions and packet drops.

At this point it is interesting to note the total number of messages that are communicated between the sensor nodes, which is also the metric that models the total network energy dissipated (Figures 1.9). For a dense sensornet with uniform and skewed access pattern, *NiCoCa* sends at most half of the messages sent out by *HybridCache* and the situation becomes better for *NiCoCa* as the access pattern becomes more skewed, which is expected. These results are confirmed for larger sensornets with 1000 nodes (Figures 1.10). The reason is that due to cache cooperation within a proximity region *NiCoCa* gets data from nearby nodes instead of far away data center or remote caches. Therefore, the data requests and replies need to pass by a significant smaller number of hops and sensor nodes have to process

lower number of messages.

Experiments with KB-sized data items

A significant question arises whether these relative results still hold when the sensornet has to deal with smaller multimedia files, with size equal to a few kilobytes. Although it is expected that WMSNs will deal with MB-sized images of video files, it might be the case that the sensor nodes will exchange smaller images as well. To investigate the performance of the cooperative caching protocols for this case, the same set of experiments was performed but for KB-sized files and here we present the results.

The general observations that was recorded for the case of large MB-size files, still hold for this case; *NICoCa* achieves significantly smaller number of global hits and larger number of remote hits than *HybridCache* does. It is not worthy to comment on each individual performance graph (Figure 1.11–1.12), since in all cases *NICoCa* has a better performance; it achieves again 25% more remote hits and 50% less global hits than *HybridCache*, which is only marginally better than *NICoCa* in terms of local hits.

1.5 Conclusion

The recent advances in miniaturization, the creation of low-power circuits, and the development of cheap CMOS cameras and microphones, which are able to capture rich multimedia content, gave birth to what is called *Wireless Multimedia Sensor Networks (WMSNs)*. WMSNs are expected to fuel many new applications and boost the already existing. The unique features of WMSNs call for protocol designs that will provide application-level QoS, an issue that has largely been ignored in traditional wireless sensor networks. Taking a first step toward this goal, this chapter presented a cooperative caching protocols, the *NICoCa* protocol, suitable for deployment in WMSNs. The protocol “detects” which sensor nodes are most “central” in the network neighborhoods and gives to them the role of mediator in order to coordinate the caching decisions. The *NICoCa* protocol is evaluated with J-Sim

and its performance is compared to that of a state-of-the-art cooperative caching protocol for MANETs. The results attest the performance gains of the *NICoCa* protocol which is able to reduce the global hits at an average percentage of 50% and increase the remote hits due to the effective sensor cooperation at an average percentage of 40%. The performance of the protocol is particularly high for the delivery of large multimedia data.

APPENDIX

The *NiCoCa* cooperative caching protocol

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

(A) Cache Discovery Algorithm

```
if(  $d_i$  is in local cache of requester node ) then
    send ipacket to CHs;
    return;

if( requester node is CH and  $d_i$ 's id in PCT ) then
    select caching node with largest RE;
    send request( $d_i$ ) to caching node;
else
    requester node sends request( $d_i$ ) to CHs;
    when CHs answers or time elapsed
    if( caching nodes found ) then
        select caching node with largest RE;
        send request( $d_i$ ) to caching node;
    else
        send request( $d_i$ ) to data center;
    when  $N_i$  receives request( $d_i$ )
    if(  $N_i$  has a valid copy ) then
        send  $d_i$  to requester node;
    else if(  $N_i$  is CH and  $d_i$ 's id in PCT ) then
        select caching node with largest RE;
        redirect request( $d_i$ ) to caching node;
```

else

forward request(d_i) to caching node;

(B) Replacement Policy

while(current node has not enough FS)

Select a valid d_i with largest value and store it temporary;

Send to CHs d_i 's id;

Remove the valid d_i ;

when a CH gets d_i 's id

if(CH gets d_i 's id and d_i 's id not in PCT) **then**

select caching node with largest RE and FS;

send answer to requester node;

when current node get answers from CHs

foreach(temporary stored d_i)

if(there is no other caching node) **then**

Select caching node with least RE and largest FS;

Send d_i to new caching node;

Remove temporary stored d_i ;

(C) Cache Admission Policy

when the packet with d_i obtained from current node

if(current node is packet's destination) **then**

if(there is enough FS) **then**

cache d_i ;

send ipacket to CHs;

else

call Replacement Policy ;

when CH gets an ipacket

if(CH get ipacket) **then**

store d_i 's id, RE and FS in PCT;

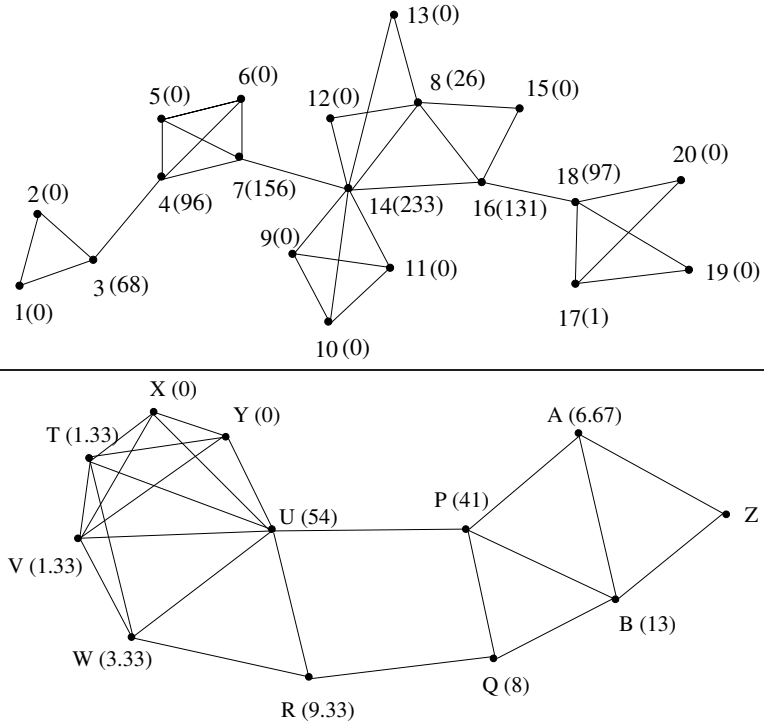


Figure 1.2: Calculation of \mathcal{NI} for two sample graphs. The numbers in parentheses denote the \mathcal{NI} index of the respective node considering the whole WMSN topology.

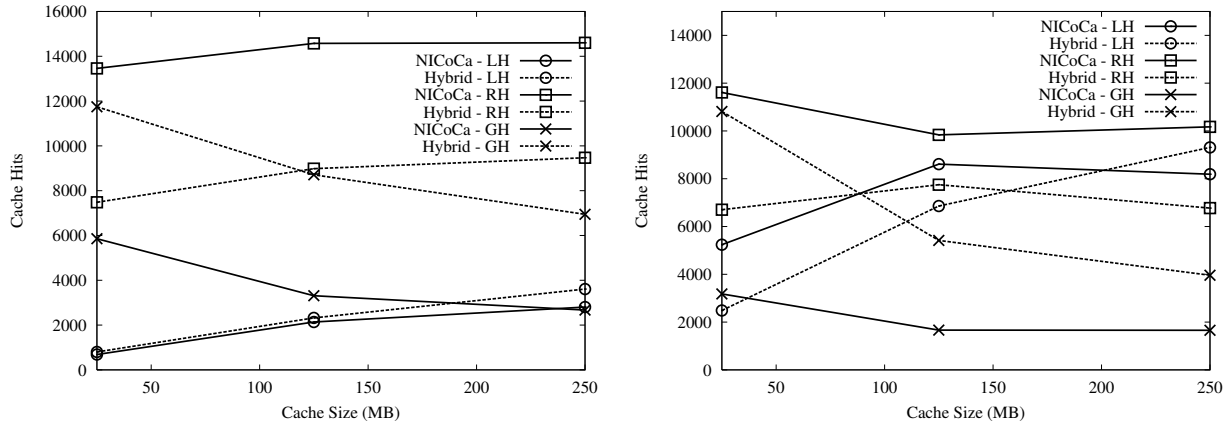


Figure 1.3: Impact of sensor cache size on hits (MB-sized files, $\theta = 0.0$ and $\theta = 0.8$) in a sparse WMSN ($d = 7$) with 100 sensors.

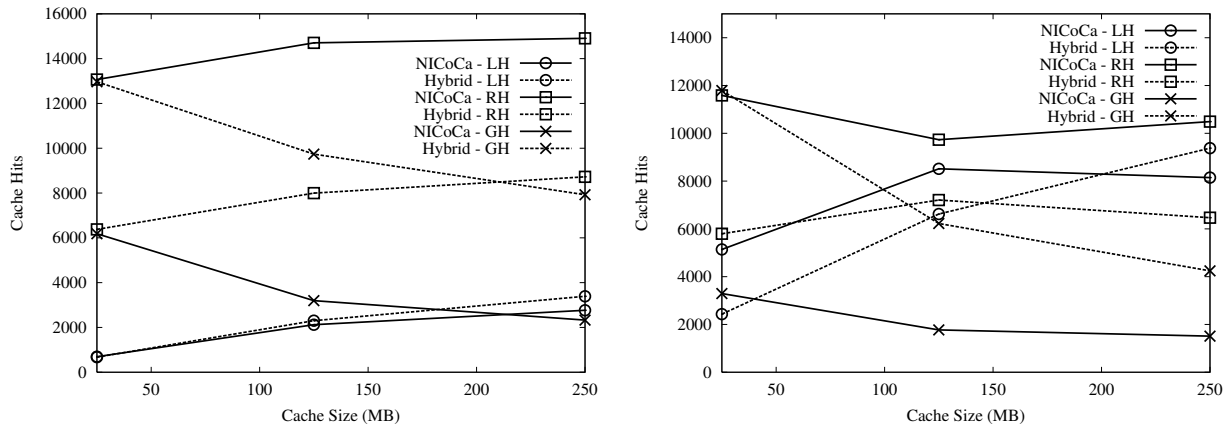


Figure 1.4: Impact of sensor cache size on hits (MB-sized files, $\theta = 0.0$ and $\theta = 0.8$) in a dense WMSN ($d = 10$) with 100 sensors.

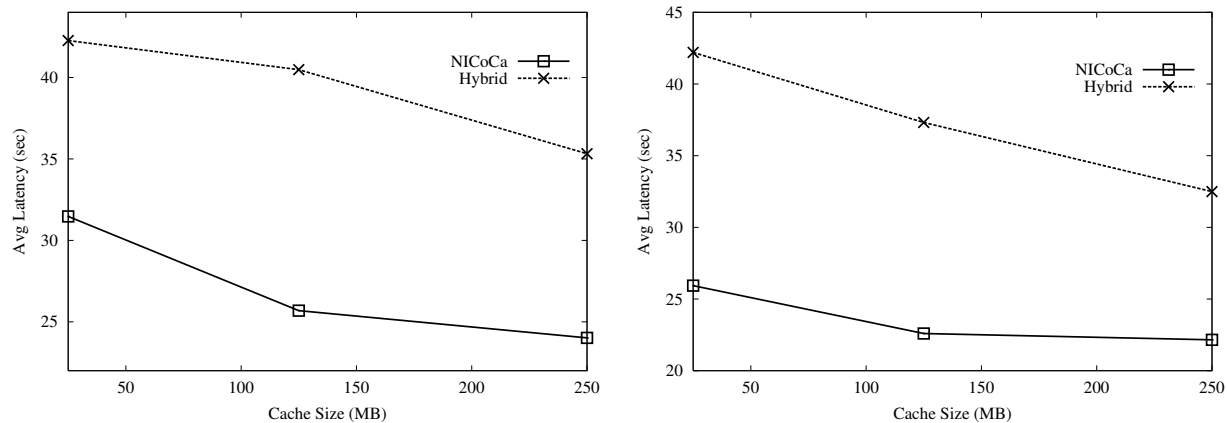


Figure 1.5: Impact of sensor cache size on latency (MB-sized files, $\theta = 0.0$ and $\theta = 0.8$) in a sparse WMSN ($d = 7$) with 100 sensors.

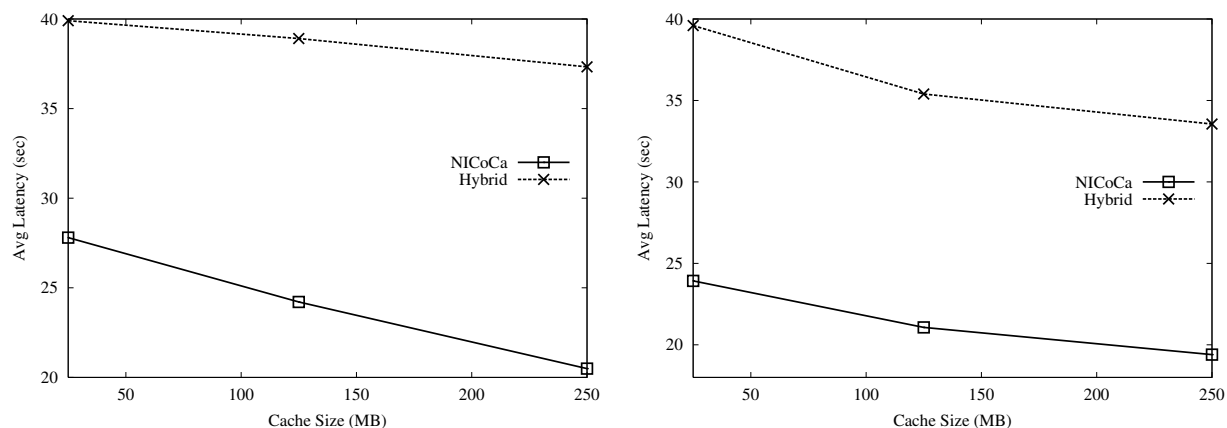


Figure 1.6: Impact of sensor cache size on latency (MB-sized files, $\theta = 0.0$ and $\theta = 0.8$) in a dense WMSN ($d = 10$) with 100 sensors.

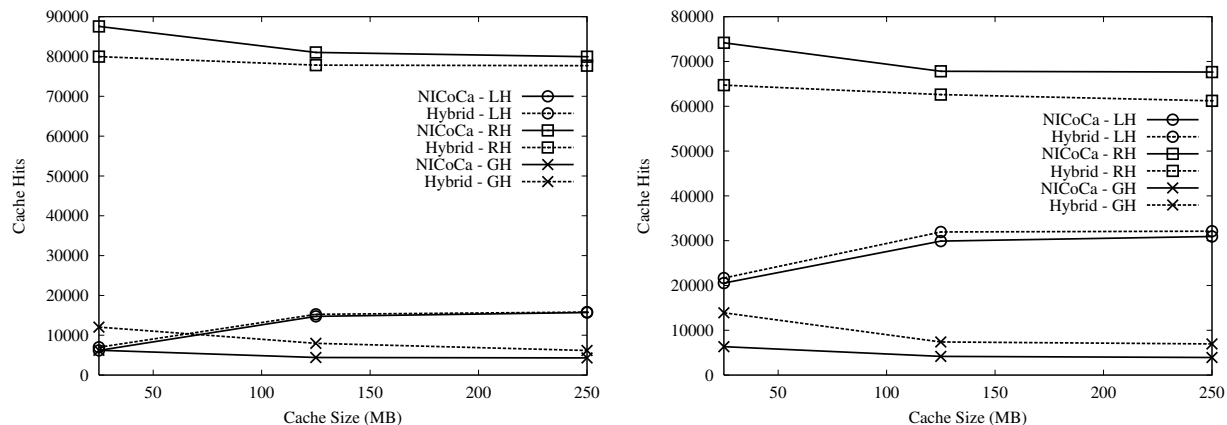


Figure 1.7: Impact of sensor cache size on hits (MB-sized files, $\theta = 0.0$ and $\theta = 0.8$) in a sparse WMSN ($d = 7$) with 500 sensors.

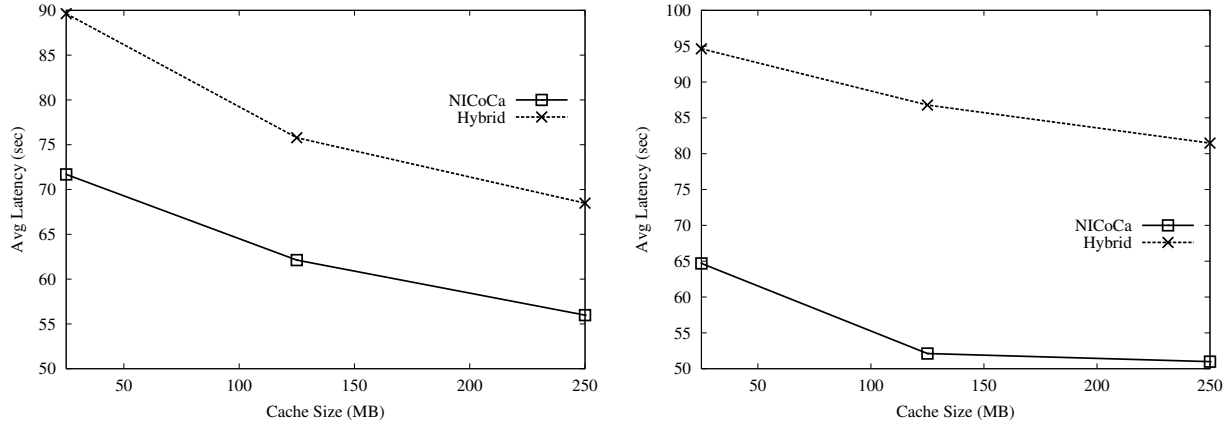


Figure 1.8: Impact of sensor cache size on latency (MB-sized files, $\theta = 0.0$ and $\theta = 0.8$) in a sparse WMSN ($d = 7$) with 500 sensors.

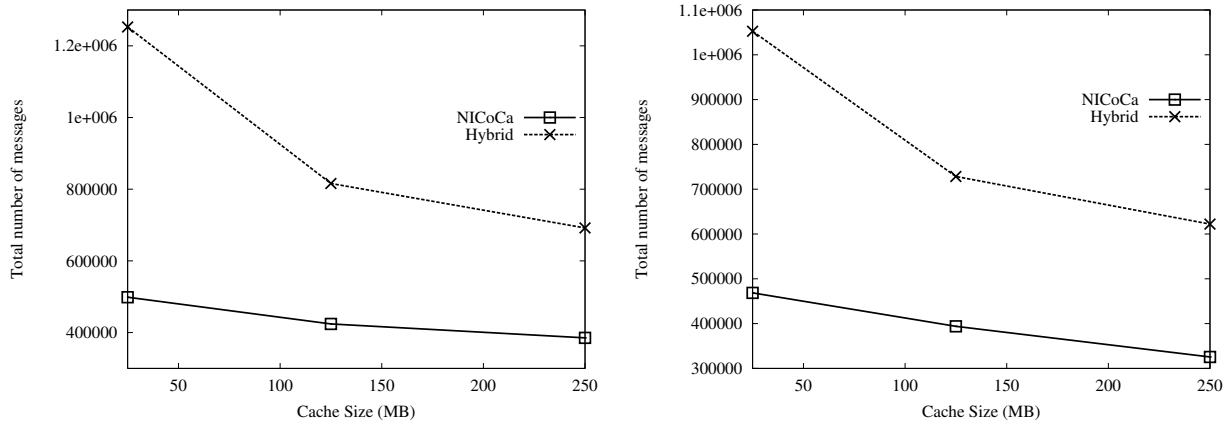


Figure 1.9: Impact of sensor cache size on number of messages (MB-sized files, $\theta = 0.0$ and $\theta = 0.8$) in a dense WMSN ($d = 10$) with 500 sensors.

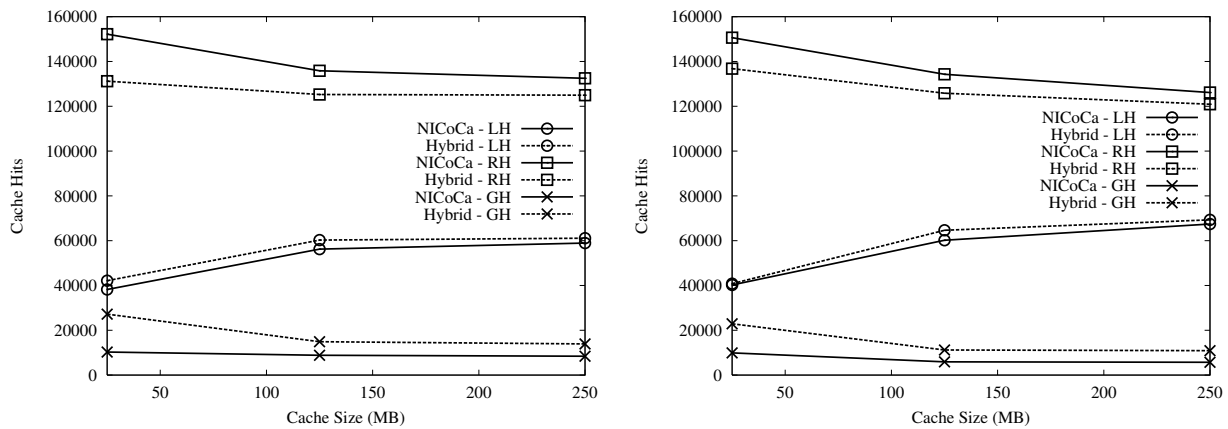


Figure 1.10: Impact of sensor cache size on hits (MB-sized files, $\theta = 0.8$) in a dense WMSN ($d = 10$) with 1000 sensors.

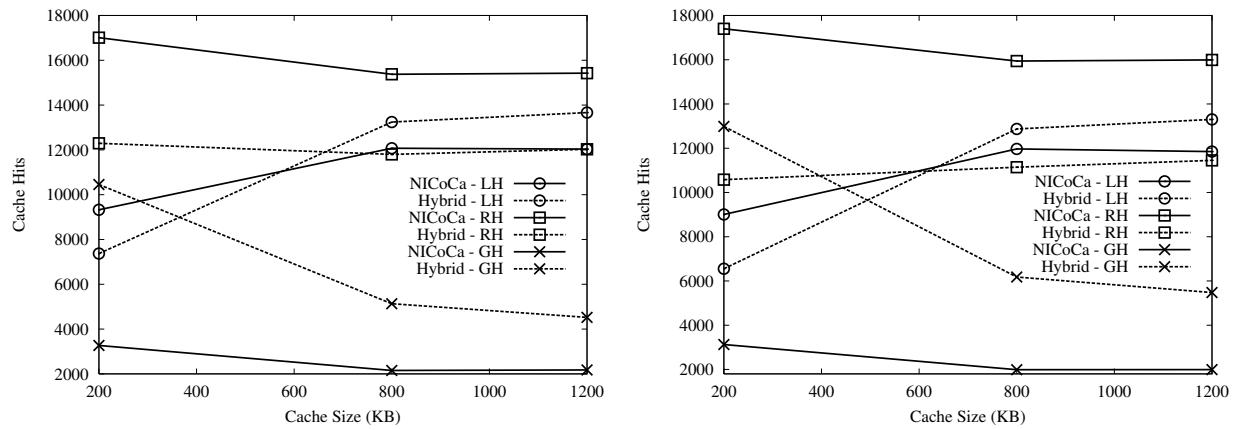


Figure 1.11: Impact of sensor cache size on hits (KB-sized files, $\theta = 0.8$) in a sparse and dense WMSN ($d = 7$ and $d = 10$) with 100 sensors.

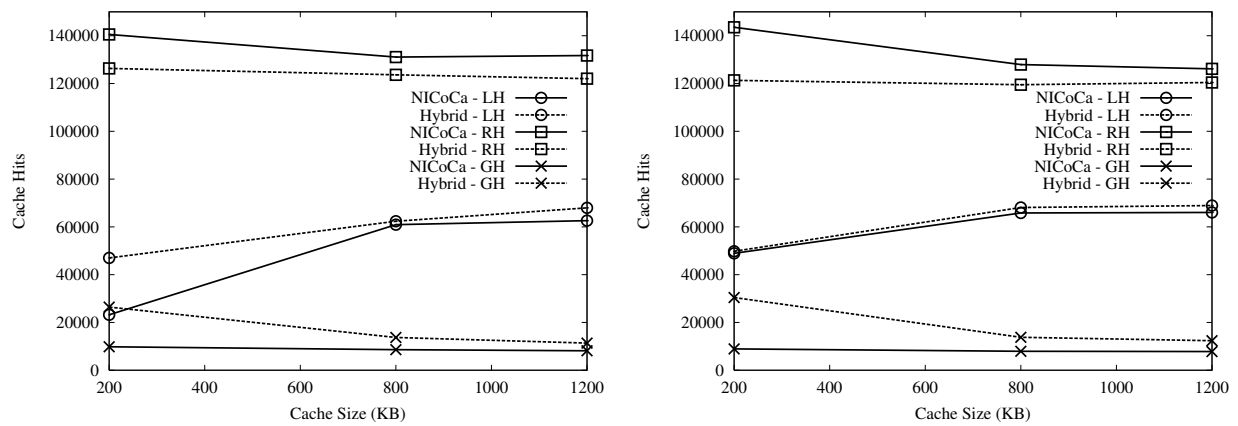


Figure 1.12: Impact of sensor cache size on hits (KB-sized files, $\theta = 0.8$) in a sparse and dense WMSN ($d = 7$ and $d = 10$) with 1000 sensors.

References

- [1] I. Akyildiz, T. Melodia, and K. R. Chowdhury. A survey of wireless multimedia sensor networks. *Computer Networks*, 51(4):921–960, 2007.
- [2] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. A survey of wireless sensor networks. *IEEE Communications magazine*, 40(8):102–116, 2002.
- [3] S. Basagni, M. Mastrogiovanni, A. Panconesi, and C. Petrioli. Localized protocols for ad hoc clustering and backbone formation: A performance comparison. *IEEE Transactions on Parallel and Distributed Systems*, 17(4):292–306, 2006.
- [4] N. Chand, R. C. R. C. Joshi, and M. Misra. A zone co-operation approach for efficient caching in mobile ad hoc networks. *International Journal of Communication Systems*, 19:1009–1028, 2006.
- [5] N. Chand, R. C. R. C. Joshi, and M. Misra. Cooperative caching strategy in mobile ad hoc networks based on clusters. *Wireless Personal Communications*, 43(1):41–63, 2007.
- [6] H. Che, Y. Tung, and Z. Wang. Hierarchical Web caching systems: Modeling, desing and experimental results. *IEEE Journal on Selected Areas in Communications*, 20(7):1305–1314, 2002.
- [7] C.-Y. Chow, H. V. Leong, and A. T. S. Chan. Cache signatures for peer-to-peer cooperative caching in mobile environments. In *Proceedings of the IEEE International Conference on Advanced Information Networking and Applications (AINA)*, volume 1, pages 96–101, 2004.
- [8] C.-Y. Chow, H. V. Leong, and A. T. S. Chan. Peer-to-peer cooperative caching in mobile environments. In *Proceedings of the IEEE International Conference on Distributed Computing Systems Workshops (ICDCSW)*, pages 528–533, 2004.
- [9] C.-Y. Chow, H. V. Leong, and A. T. S. Chan. GroCoca: Group-based peer-to-peer cooperative caching in mobile environment. *IEEE Journal on Selected Areas in Communications*, 25(1):179–191, 2007.
- [10] Y. Diao, D. Ganesan, G. Mathur, and P. Shenoy. Rethinking data management for storage-centric sensor networks. In *Proceedings of the Conference on Innovative Data Systems Research (CIDR)*, pages 22–31, 2007.
- [11] Y. Eisenberg, C. E. Luna, T. N. Pappas, R. Berry, and A. K. Katsaggelos. Joint source coding and transmission power management for energy efficient wireless video communications. *IEEE Transactions on Circuits and Systems for Video Technology*, 12(6):411–424, 2002.
- [12] 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.
- [13] 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.
- [14] B. Girod, M. Kalman, Y. J. Liang, and R. Zhang. Advances in channel-adaptive video streaming. *Wireless Communications and Mobile Computing*, 2(6):573–584, 2002.

- [15] T. Hara. Replica allocation methods in ad hoc networks with data update. *ACM/Kluwer Mobile Networks and Applications*, 8(4):343–354, 2003.
- [16] T. Hara and S. K. Madria. Data replication for improving data accessibility in ad hoc networks. *IEEE Transactions on Mobile Computing*, 5(11):1515–1532, 2006.
- [17] G. T. Huang. Casting the wireless sensor net. *Technology Review*, pages 51–56, Jul. 2003.
- [18] H. Karl and A. Willig. *Protocols and Architectures for Wireless Sensor Networks*. John Wiley & Sons, 2006.
- [19] D. Katsaros and Y. Manolopoulos. Caching in Web memory hierarchies. *Proceedings of the ACM Symposium on Applied Computing*, pages 1109–1113, 2004.
- [20] D. Katsaros and Y. Manolopoulos. Web caching in broadcast mobile wireless environments. *IEEE Internet Computing*, 8(3):37–45, 2004.
- [21] D. Katsaros and Y. Manolopoulos. The geodesic broadcast scheme for wireless ad hoc networks. In *Proceedings of the IEEE International Symposium on World of Wireless, Mobile Multimedia (WoWMoM)*, pages 571–575, 2006.
- [22] P. Kulkarni, D. Ganesan, P. Shenoy, and Q. Lu. SensEye: A multi-tier camera sensor network. In *Proceedings of the ACM International Conference on Multimedia (MM)*, pages 229–238, 2005.
- [23] G. Mathur, P. Desnoyers, D. Ganesan, and P. Shenoy. Ultra-low power data storage for sensor networks. In *Proceedings of the ACM International Conference on Information Processing in Sensor Networks (IPSN)*, pages 374–381, 2006.
- [24] N. Megiddo and D. S. Modha. ARC: A self-tuning, low overhead replacement cache. In *Proceedings of the USENIX Conference on File and Storage Technologies (FAST)*, pages 115–130, 2003.
- [25] S. Nath and A. Kansal. FlashDB: Dynamic self-tuning database for NAND flash. In *Proceedings of the ACM International Conference on Information Processing in Sensor Networks (IPSN)*, pages 410–419, 2007.
- [26] G. Pallis, A. Vakali, and J. Pokorny. A clustering-based prefetching scheme on a Web cache environment. *Computers & Electrical Engineering*, 2007. To appear.
- [27] M. Papadopouli and H. Schulzrinne. Effects of power conservation, wireless coverage and cooperation on data environments. In *Proceedings of ACM Symposium on Mobile Ad Hoc Networking and Computing (MOBIHOC)*, pages 117–127, 2001.
- [28] C. E. Perkins and E.M. Royer. Ad hoc On-demand Distance Vector routing. In *Proceedings of the IEEE Workshop on Mobile Computing Systems and Applications*, pages 90–100, 1999.
- [29] K. S. Prabh and T. F. Abdelzaher. Energy-conserving data cache placement in sensor networks. *ACM Transactions On Sensor Networks*, 1(2):178–203, 2005.
- [30] M. Rahimi, R. Baer, O. I. Iroezi, J. C. Garcia, J. Warrior, D. Estrin, and M. Srivastava. Cyclops: In situ image sensing and interpretation in wireless sensor networks. In *Proceedings of the ACM International Conference on Embedded Networked Sensor Systems (SenSys)*, pages 192–204, 2005.

- [31] A. Rousskov and D. Wessels. Cache digests. *Computer Networks and ISDN Systems*, 30(22–23):2155–2168, 1998.
- [32] F. Sailhan and V. Issarny. Energy-aware Web caching for mobile terminals. In *Proceedings of the IEEE International Conference on Distributed Computing Systems Workshops (ICDCSW)*, pages 820–825, 2002.
- [33] F. Sailhan and V. Issarny. Cooperative caching in ad hoc networks. In *Proceedings of the IEEE International Conference on Mobile Data Management (MDM)*, pages 13–28, 2003.
- [34] 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.
- [35] 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.
- [36] M. Takaaki and H. Aida. Cache data access system in ad hoc networks. In *Proceedings of the IEEE Spring Semiannual Vehicular Technology Conference (VTC)*, volume 2, pages 1228–1232, 2003.
- [37] B. Tang, S. Das, and H. Gupta. Cache placement in sensor networks under update cost constraint. In *Proceedings of the (ADHOC-NOW)*, volume 3738 of *Lecture Notes on Computer Science*, pages 334–348, 2005.
- [38] L. Tassiulas and C. Su. Optimal memory management strategies for a mobile user in a broadcast data delivery system. *IEEE Journal on Selected Areas in Communications*, 15(7):1226–1238, 1997.
- [39] D. Wessels and K. Claffy. ICP and the Squid Web cache. *IEEE Journal on Selected Areas in Communications*, 16(3):345–357, 1998.
- [40] L. Yin and G. Cao. Supporting cooperative caching in ad hoc networks. *IEEE Transactions on Mobile Computing*, 5(1):77–89, 2006.
- [41] X. Yu. Distributed cache updating for the dynamic source routing protocol. *IEEE Transactions on Mobile Computing*, 5(6):609–626, 2006.