**ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ**

**Τμήμα Μηχανικών Η/Υ, Τηλεπικοινωνιών και Δικτύων**

Διπλωματική εργασία του φοιτητή

**ΜΠΑΣΑΡΑ ΠΑΥΛΟΥ**

με τίτλο

# «Cooperative Caching in Wireless Multimedia Sensor Networks»

Επιβλέποντες καθηγητές:

**ΚΑΤΣΑΡΟΣ ΔΗΜΗΤΡΙΟΣ**

**ΜΠΟΖΑΝΗΣ ΠΑΝΑΓΙΩΤΗΣ**

# CONTENTS

## ABSTRACT

**Wireless Sensor Networks** is a fast growing and exciting research area that has attracted considerable research attention in the recent past. This has been fueled by the recent tremendous technological advances in the development of low-cost sensor devices equipped with wireless network interfaces, combined with small sized batteries. The creation of large-scale sensor networks interconnecting several hundred to a few thousand sensor nodes, opens up several technical challenges and immense application possibilities. Sensor networks find application spanning several domains including military, medical, industrial and home networks. The availability of low-cost hardware such as CMOS cameras and microphones has fostered the development of *Wireless Multimedia Sensor Networks* (WMSNs), i.e., networks of wirelessly interconnected devices that are able to ubiquitously retrieve multimedia content such as video and audio streams, still images and scalar sensor data from the environment. To address this goal in an environment with extreme resource constraints, with variable channel capacity and with requirements for multimedia in-network processing, the caching of multimedia data, exploiting the cooperation among sensor nodes is vital. Here we present a cooperative caching solution suitable for WMSNs. Our caching solution exploits the sensor nodes which *reside in positions* of the network that allow them to forward packets or communicate decisions within short latency. These nodes are called *mediators*.
These nodes are selected dynamically to avoid the creation of hot-spots in the communication and the depletion of their energy. Mediators are ordinary nodes who have some special role in implementing the cooperation among the sensors. Our solution is evaluated extensively in an advanced simulation environment. Our cooperative protocol includes components for locating cache data as well as for implementing data purging from sensor caches.

**Keywords** cooperative caching, replacement policy, multimedia, wireless sensor networks

## 1. Introduction

Wireless sensor networks (WSNs) have drawn the attention of the research community in the last few years, driven by a wealth of theoretical and practical challenges. This growing interest can be largely attributed to new applications enabled by large-scale networks of small devices capable of harvesting information from the physical environment, performing simple processing on the extracted data and transmitting it to remote locations. Significant results in this area over the last few years have ushered in a surge of civil and military applications. As of today, most deployed wireless sensor networks measure scalar physical phenomena like temperature, pressure, humidity, or location of objects. More recently, the availability of inexpensive hardware such as CMOS cameras and microphones that are able to ubiquitously capture multimedia content from the environment has fostered the development of Wireless Multimedia Sensor Networks (WMSNs).

Wireless multimedia sensor networks will not only enhance existing sensor network applications such as tracking, home automation, and environmental monitoring, but they will also enable several new applications such as:

- Multimedia surveillance sensor networks
- Storage of potentially relevant activities
- Traffic avoidance, enforcement and control systems.
- Advanced health care delivery
- Automated assistance for the elderly and familyMonitors
- Environmental monitoring
- Person locator services
- Industrial process control

The support of such huge range of applications will be rather impossible for any single realization of a WSN. Nonetheless, certain common features appear in common among those applications and the realization of these characteristics is the major challenge faced by these networks. The most significant characteristics shared by those application are listed below:

- *Lifetime*: Usually , sensor nodes rely on battery with limited lifetime ,and their replacement is not possible due to physical constraints ( they lie in hostile environments ).

- *Scalability*: the architecture and protocols of sensor networks must be able to scale up 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

- *Data-centric networking*: the target of a communication network is to move bits from a node to another, but the purpose of a sensor network is to provide information and answers not numbers.

Many of the above applications require the sensor network paradigm to be rethought in view of the need for mechanisms to deliver multimedia content with a certain level of quality of service (QoS). Since the need to minimize the energy consumption has driven most of the research in sensor networks so far, mechanisms to efficiently deliver application level QoS, and to map these requirements to network layer metrics such as latency and jitter, have not been primary concerns in mainstream research on classical sensor networks.

The target of 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 WMSNs. WMSN are 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*: 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 restriction the goal of QoS in WMSNs becomes a very challenging task. In our implementation we investigate the solution of *cooperative caching multimedia content* in sensor to address those characteristics. In cooperative caching, multiple sensor nodes share and coordinate cache data to cut communication cost and exploit the aggregate cache space of cooperating sensors. The plain assumption we make, is that each sensor node has a moderate local storage capacity associated with it, i.e., a fash memory. Although, there exist fash memo-
ries with several gigabytes storage capacity, e.g., the NAND fash and the trend is that they become cheaper, larger and more common, we do not assume extreme storage capabilities, so as to capture a broader field of applications and architectures.

The battery lifetime can be extended if we can reduce the amount of communication in the network by storing the useful data for each node in it’s local cache or in it’s near neighborhood. This way battery lifetime of each node can be prolonged. Caching also can reduce the network wide transmissions thus reducing the interference and also overcoming the various channel conditions. Finally the multimedia in-network processing is faster thus better QoS since processing and delivery of multimedia content are not independent and their interaction has a major impact on the level of QoS.

We present our caching protocol in the context of WMSNs the *Node Importance-based Cooperative caching* protocol. We used Oment++ (with MIXIM) simulator to run our experiments.

## 2. Relevant work

The technique of caching has been widely investigated in the context of operating systems and databases and it is still a very fertile research area. Similarly, caching on the Web has been thoroughly investigated for cooperative and for non-cooperative architectures . These environments though do not face the extreme resource constraints of WMSNs, and thus the proposed caching protocols are not appropriate for the case of WMSNs.

There are also a number of caching approaches for wireless broadcast cellular networks . These policies assume more powerful nodes than the sensor nodes, and one-hop communication with resource-rich base stations, which serve the needed data.

The most relevant research works to our protocol are the cooperative caching protocols which have been developed for mobile ad hoc networks (MANETs). The main motive for the development of these protocols is the mobility of the nodes, and thus they all strive to model it or exploit it. For instance, the GroCoca  attempts to organize nodes into groups based on their data request pattern and their mobility pattern. The ECOR , the Zone Co-operative and the Cluster Cooperative protocols attempt to form clusters of nodes based either in geographical proximity or utilizing widely known node clustering algorithms for MANETs. The only protocols that deviated from such approaches and tried to exploit both data and node locality in an homogeneous manner are the following: *CachePath*, *CacheData*, and *HybridCache*. In CacheData, intermediate nodes (the nodes between a requesting node and the node which holds the requested data) cache the data to serve future requests instead of fetching data from the “Data Center". In *Cache Path*, mobile nodes cache the data path and use it to redirect future requests to the nearby node which has the data instead of the faraway Data Center. A high performance amalgamation of them is the Hybrid- Cache, which can further improve the performance by taking advantage of *Cache Data* and *Cache Path* while avoiding their weaknesses.

The only works on caching in wireless sensor networks concern the placement of caches and thus they are not strictly relevant to our considered problem.

## 2.1 Motivation and contributions

The protocols proposed so far for cooperative caching in MANETs present various limitations. Those protocols which first perform a clustering of the network and then exploit this clustering (actually, the cluster-heads, CH) in order to coordinate the caching decisions, inherit the shortcomings of any bad CH selection. 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 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‘t cache the data/path, or conversely, can’t serve the request even if it holds the requested data.

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

- Definnition 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 sensors.
- Development of algorithms for discovering the requested multimedia data, and maintaining the caches.

- Performance evaluation of the protocol using an established simulation package (OMNET++).

## 3. The Node Importance –based Cooperative Caching protocol

One of the basic parts of the proposed protocol is the estimation of the importance of a sensor node in the network topology. The idea is that if we can discover those nodes which reside in a significant part of paths connecting other nodes these nodes can be selected as “mediators” and serve as access to requested data or caching points.

### 3.1 Sensor Node’s 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 $e = (u,v)$, $u, v \varepsilon E$ exists, if and only if $u$ is in the transmission range of $v$ and vice versa. The network is assumed to be in a connected state. The set of neighbors of a node $v$ is represented by $N_1(v)$, i.e., $N_1(v) = \{ u : (v,u) \varepsilon E\}$. 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., $N_2(v) = \{w : (u,w) \varepsilon E\}$, where *w is different from v* and *w doesn't belong to* $N1$ and $(v,u) \varepsilon E\}$. The combined set of one-hop and two-hop neighbors of $v$ is denoted as $N_{12}(v)$.

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

### *3.2 Routing Betweenness-Centrality* (*RBC*) OF INDIVIDUAL VERTICES

Assume that a packet is introduced to the network by source vertex $s$ and destined to leave the network at target vertex $t$. Let $\delta_{s,t}(v)$ be the probability that this packet will pass through the vertex $v$. We will refer to $\delta_{s,t}(v)$ and its variants as *pairwise dependency* of $s$ and $t$ on the intermediate $v$. $\delta_{s,t}(v) \cdot T(s, t)$ is the expected number of packets sent from $s$ to $t$ that pass through $v$. Note that for special cases where $v$ equals $s$, $t$, or both it holds that $\delta s,t\ (s) = \delta s,t\ (t) = 1$. $\delta s,t\ (v)$ can be recursively computed for arbitrary $v\ \varepsilon\ V$ based on the loop-free routing strategy $R(s, u, v, t)$. Let $Preds,t\ (v)$ be a set of all immediate predecessors of $v$ on theway to $t$: $Preds,t\ (v) = \{u \mid R(s, u, v, t) > 0\}$. Let $u$ be a predecessor of $v$ on theway from $s$ to $t$. The probability that a packet will pass through $v$ after visiting $u$ is $R(s, u, v, t)$. Hence, the pairwise dependency of $s$ and $t$ on $v$ can be computed using pairwise dependency of $s$ and $t$ on $v$'s predecessors.

$$\delta s,t\ (s) = 1 \qquad (1)$$

$$\delta s,t\ (v) = \sum_{u\ \varepsilon\ Preds,t\ (v)} \delta_{s,t}(u) \cdot R(s, u, v, t).$$

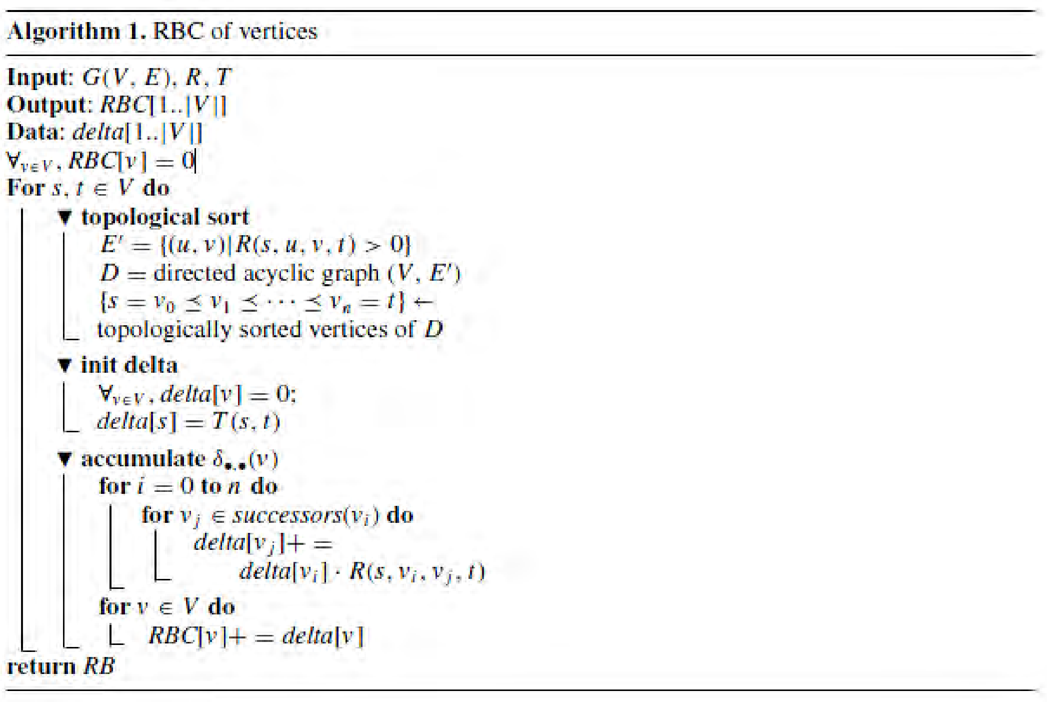
Since we assume loop-free routing, $Preds,t$ defines a directed acyclic graph (DAG)[Harary et al. 1965] as shown in Figure 2. Therefore, we can compute $\delta s,t\ (v)$ for all $v\ \varepsilon\ V$ in $O(m)$ in the worst case. All we need to do is topologically sort the DAG induced by $Preds,t$ and iteratively apply Eq.(1) on all vertices starting from $s$.

Let RBC of a vertex $v$ ($\delta_{\bullet,\bullet}(\mathrm{v})$) be the expected number of packets that pass through $v$.

$$\delta_{\bullet,\bullet}(\mathrm{v}) = \sum_{s,t\ \varepsilon\ V} \delta s,t\ (v) \cdot T\,(s, t). \qquad (2)$$

$\delta_{\bullet,\bullet}(v)$ can be regarded as the potential of $v$ to inspect or alter communications in the network. Eq. (2) resembles the original definition of SPBC with two exceptions. First, each $\delta s,t\ (v)$ is multiplied by the number of packets sent from $s$ to $t$ to compute the traffic load on $v$. Second, end points are included in the summation to accommodate communications originating from (or destined to) the investigated node. Algorithm 1 computes the RBC of all individual vertices in $O(n2m)$ time using Eqs. (1) and (2).

Here is the algorithm of RBC of vertices:



**Algorithm 1.** RBC of vertices

```
Input: $G(V, E), R, T$
Output: $RBC[1..|V|]$
Data: $delta[1..|V|]$
$\forall_{v \in V}, RBC[v] = 0$
For $s, t \in V$ do
    ▼ topological sort
        $E' = \{(u, v) | R(s, u, v, t) > 0\}$
        $D$ = directed acyclic graph $(V, E')$
        $\{s = v_0 \preceq v_1 \preceq \cdots \preceq v_n = t\} \leftarrow$
        topologically sorted vertices of $D$
    ▼ init delta
        $\forall_{v \in V}, delta[v] = 0;$
        $delta[s] = T(s, t)$
    ▼ accumulate $\delta_{\bullet,\bullet}(v)$
        for $i = 0$ to $n$ do
            for $v_j \in successors(v_i)$ do
                $delta[v_j] += $
                    $delta[v_i] \cdot R(s, v_i, v_j, t)$
        for $v \in V$ do
            $RBC[v] += delta[v]$
return $RB$
```

It is composed of an outer loop that iterates over all *s*-*t* pairs of vertices and of three inner stages. In the first stage the algorithm creates the routing DAG with single source *s* and single sink *t*. In the second stage the *delta* array is initialized .Entry *delta[v]* of this array represents the expected number of packets from *s* to *t* that pass through *v*: $\delta s,t\ (v) \cdot T\ (s, t)$. Finally in the third stage the expected number of packets from *s* to *t* that pass through each one of the vertices is computed and these probabilities are accumulated according to Equation 2 to form RBC values of all vertices. Most of the following algorithms will use the same template and similar content.

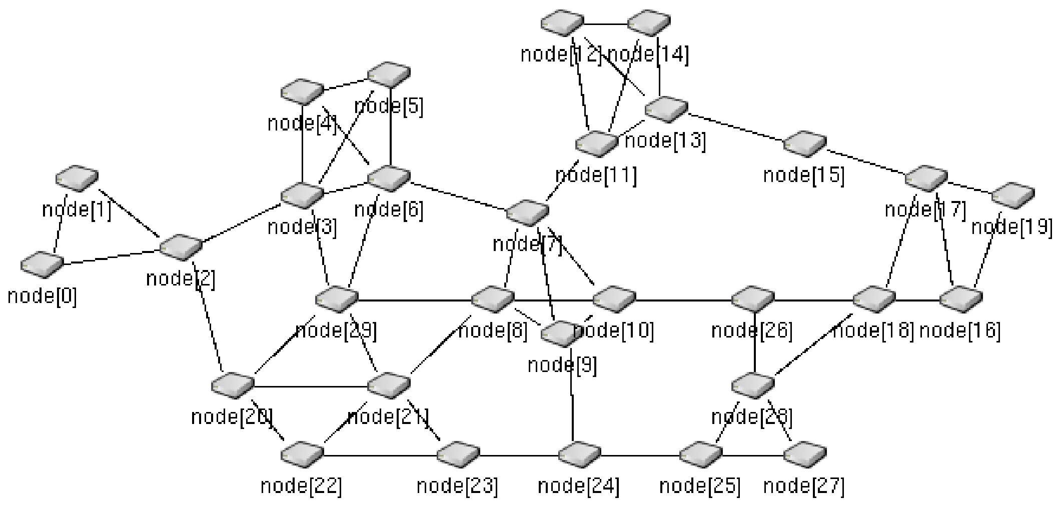


**Figure 1** : Small sized network , Data centers: are node[0] , node[16]

| node | RBC | node | RBC | node | RBC | node | RBC | node | RBC |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 7150 | 7 | 33029 | 14 | 12154 | 21 | 18921 | 28 | 35463 |
| 1 | 7150 | 8 | 19304 | 15 | 29040 | 22 | 24077 | 29 | 18930 |
| 2 | 23061 | 9 | 23287 | 16 | 21375 | 23 | 32152 | | |
| 3 | 20987 | 10 | 27347 | 17 | 33764 | 24 | 39911 | | |
| 4 | 11850 | 11 | 23537 | 18 | 39411 | 25 | 33316 | | |
| 5 | 11850 | 12 | 12154 | 19 | 13306 | 26 | 34581 | | |
| 6 | 23639 | 13 | 28768 | 20 | 24652 | 27 | 18884 | | |

**Table 1**: RBC value of vertices of complete network of figure 1

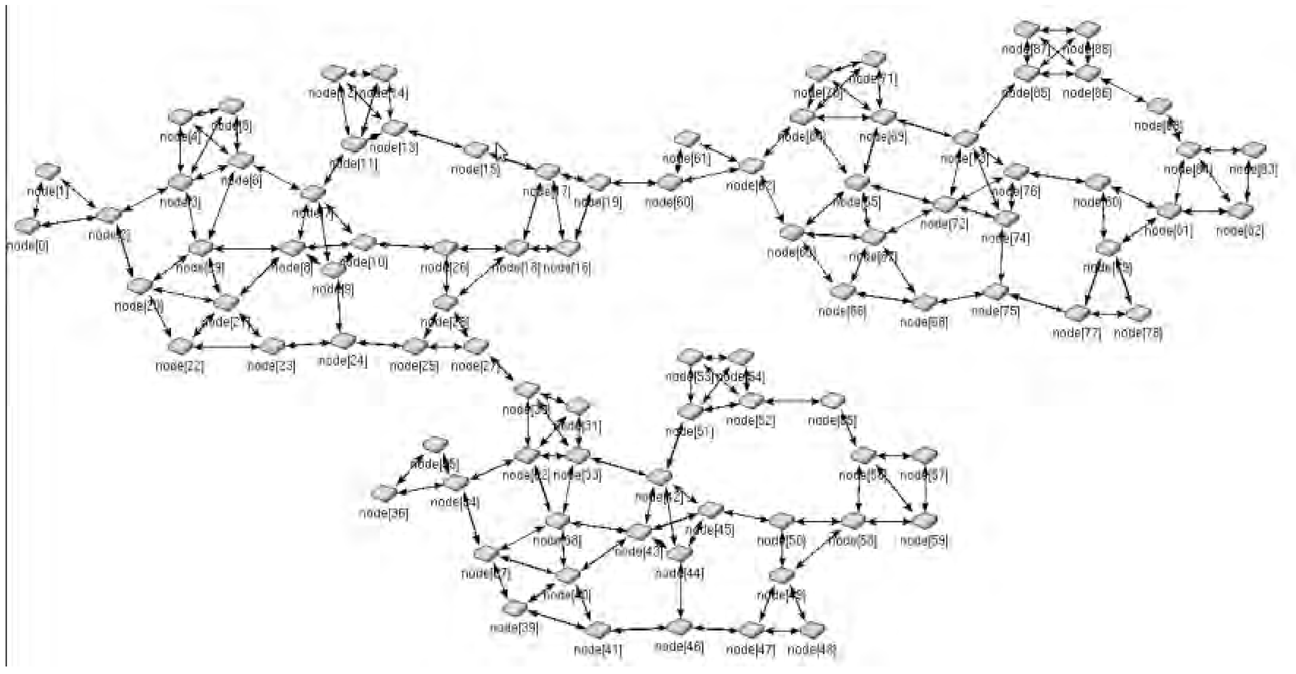


**Figure 1** : Large sized network , Data centers are: node[36] , node[71]

Apparently, when estimating the *RBC value* for each sensor node using the whole network topology we obtain a very informative picture of which nodes reside in a large number of shortest paths between other nodes. Though it would not be practical use this gloabal information because it would require the exchange of a huge number of messages to acquire this information. Fortunately, it is very easy to confirm that, even when we calculate the *RBC value* of the nodes taking into account only their *k*-hop (*k* = 2 or 3) neighborhood, the picture about the relative importance of the nodes remains very accurate.

## 3.3 Housekeeping information in our protocol

In our protocol we assume that the ultimate source of multimedia items is a Data Center. A Data Center guarantees that every request if it’s not served by other sensor node will finally be served.

We assume that every node is aware of it’s two-hop neighborhood by periodic exchange of “beacon messages”. Then every node computes it’s RBC value of it’s one-hop neighbors. Every node uses this information to choose some of his neighbors as it’s *mediators* . The minimum set of mediators with the larger RBC value that cover the two-hop neighborhood of each node are it’s *mediators*. When each node chooses it’s *mediators* it notifies them. This way each node knows it’s importance in the neighborhood. A sensor node can be either an ordinary node or a mediator.

The sending of requests for data is carried out by an ordinary routing protocol e.g AODV. A node always caches a datum he requested for and is aware of it’s remaining energy and it’s free cache space. Every node stores the following data of the requested item:

- the data_id and the actual multimedia item
- the data_size of item ($S_i$)
- TTL : time to live interval
- for every item in cache we keep the K most recent accesses to that item (K=2 or 3) .
- each cache item is either “own” or “hosted”. Hosted items are stored on behalf of another node . If a hosted item is requested by the node it changes to owned

When a node receives the multimedia item it caches it and broadcasts an index packet containing the data_id, the items TTL interval and the remaining energy of the node. The mediator nodes of the requester stores this information in their proximity table.

### 3.4 The cache discovery protocol

When a sensor node makes a request firstly checks it's own cache. If the item is found there (cache hit) then the K most recent timestamps are updated. If not then the requesting node searches it's proximity table. If found there (proximity hit)  the requester chooses the node with the most energy who has the item and forwards the request. If not in proximity table (proximity miss) the request is broadcasted and received by the mediator nodes. If none of them has the item cached or has info for the item in it's proximity table then the request is forwarded to the Data Center.

When a non one hop mediator receives a request it searches it's local cache and if not there it searches the proximity table. If item can be satisfied by a neighboring node then the route towards the Data Center stops and the request is forwarded to that neighboring node (remote hit). If more than one nodes can satisfy the request then the node with the largest energy is selected. If the request can't be satisfied then the node doesn't forward the request to it's own mediators and no searching is performed apart from the nodes which reside on the path to the Data Center.

For example suppose that node 14 in figure 2 issues a request for a data item x that is stored in Data Center 1 ( DC1) and cached by the nodes 3 and 4. We can easily see that the node 14 has selected as it's mediators the nodes 11 and 13.

Firstly node 14 searches it's cache and proximity table. If item not found there then the request is forward to the mediators. After receiving the request the mediators search their cache and the info stored at the proximity table. If data item found, then each mediators which has cached the item sends an index packet to the requester that contains the data_id and the remaining energy of the sensor node . If more than one mediator has the item then the node with the most battery is selected by the requester, to send the item.

If no mediator node has item or has information for it in it's proximity table then the requester forwards the request towards the Data Center. When node 11 receives the request as usual it searches it's cache and proximity table. If item is not found there then node 11 forwards the request towards the Data Center. When sensor 6 found that some nodes have the requested item x then it chooses among those nodes the one with the most battery and forwards the request. The caching node sends back the item to the requester. When the requester receives the multimedia item it broadcast an index packet containing the info about the new items id and it's remaining energy to it's mediators. For every request there are four cases:

- **Local Hit**: the requested item is in the requesters local cache
- **Proximity Hit**: the requested item is cached within the 2 hop neighborhood of the requester
- **Remote Hit**: the request is cached by a node found towards the path to the Data Center
- **Global Hit**: the request is found at a Data Center

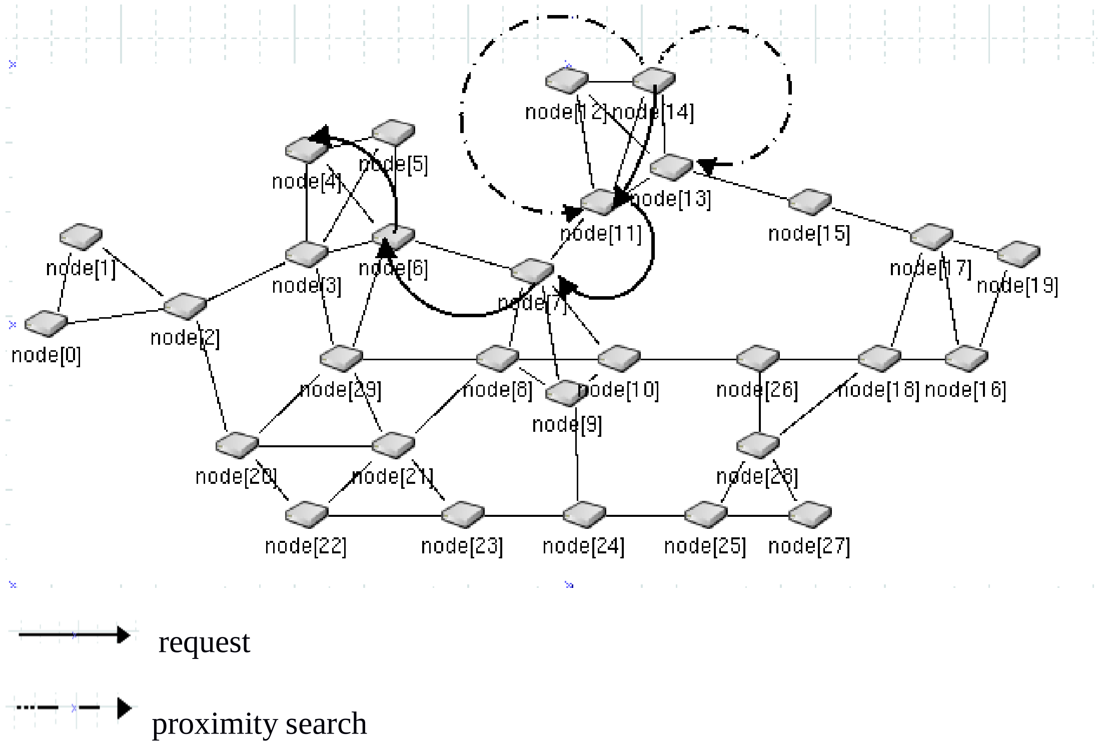


**Figure 2:** Node 14 issues a request for item x. Nodes 3 and 4 have cached the item and node 4 has more battery than node 3. Nodes 0 and 16 serve as DC1 and DC2

If remaining free cache is not enough to store the new item then it triggers the caching replacement policy.

## 3.5 Cache Replacement Policy

The development of an effective and intelligent cache replacement policy is mandatory to deal with the overwhelming size of multimedia data generated in WMSN's. Our replacement protocol in steps:

**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*: $cost(i) = (si / TTLi)*(now- t_{k\text{-}th\ access} / K)$, where $t_{k\text{-}th\ access}$ is the timestamp of the *K*-th access. 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.

## 4. Performance Evaluation

### 4.1 Our simulation model

We run some tests on small sized and large sized networks through simulation experiments in OMENT++ with various parameters. We developed a simulation model based on the OMENT++(Mixim) Simulator. We use IEEE 802.11 as the MAC protocol and the free space model as the radio propagation model . Wireless bandwidth is set to 2 Mbps. We evaluate the protocol efficiency under two set of data item sizes. Every data item has size uniformly distributed from 1KB to 10KB for the first test and 1MB to 5MB for the second.

Each sensor node generates queries. The access pattern of sensor nodes is location independent meaning that sensor nodes decide independently the data of interest; each sensor node generates accesses to the data following the uniform distribution. Two Data Centers ara place at opposite corners of the network . Half of multimedia items are stored in DC1 and the other half to DC2. Items with even id are in DC1 and items with odd id in DC2. Data Centers serve the queries on first-come-first-served basis. System Parameters are listed in Table 2.

| Parameter | Value | Range |
|---|---|---|
| # items(N) | | 1*(#nodes) to 2*(#nodes) |
| Smin(KB) | 1 | |
| Smax(KB) | 10 | |
| Smin(MB) | 1 | |
| Smax(MB) | 5 | |
| Bandwidth(Mbps) | 2 | |
| Cache size | | 1% - 10% of total size of items |
| #nodes | | 30 - 100 |

**Table 2** Simulation parameters

## 4.2 Evaluation

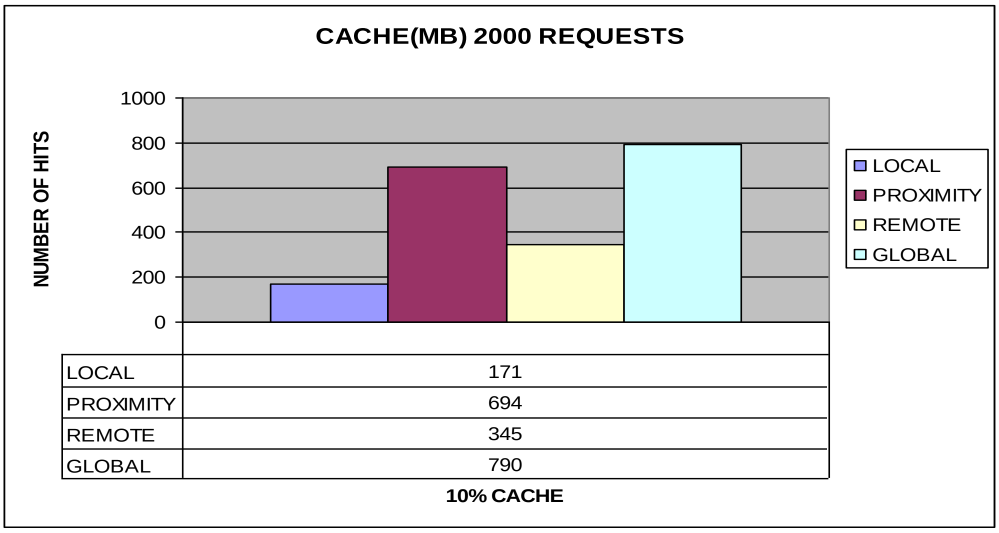
We measure the hits that occur in our simulation experiments. It is evident that a small number of global hits implies less network congestion and thus fewer collisions an packet drops. Moreover large number of remote-proximity hits proves the effectiveness of cooperation in reducing the number of global hits. A large number of local hits doesn't imply an effective cooperative caching policy unless it is accompanied by a large number of global hits since the cost of global hits vanishes the benefits of local hits.

We performed various experiments in small sized and large sized networks with varying cache size. In particular we performed experiments for cache size 1% , 5% and 10 of the aggregated size of all items for uniform access pattern for small KB sized items and large multimedia MB sized items. We will present a set of our results.

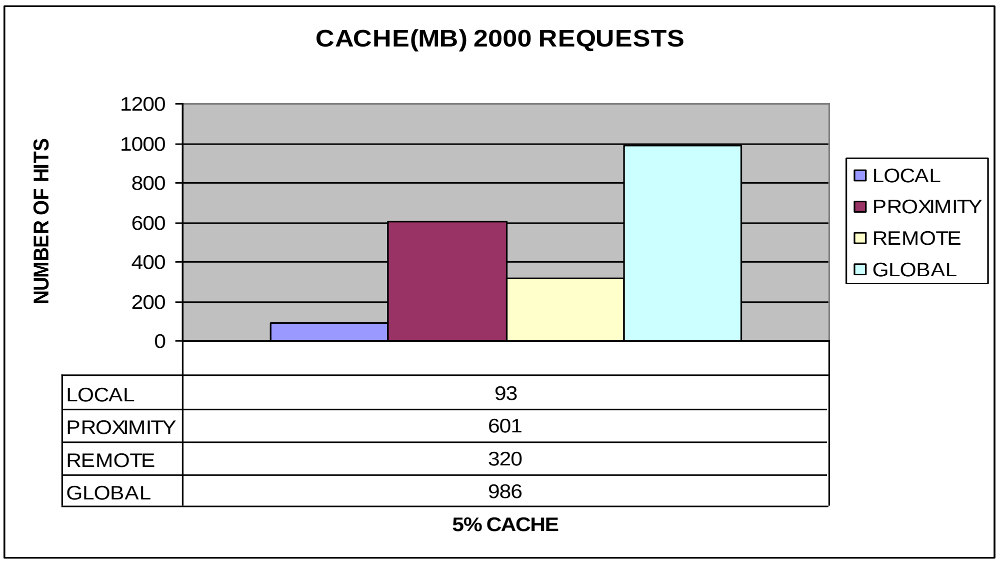
### 4.3.1 Eperiments with MB-sized items

Here we present a set of our results to show the performance of our cache protocol when we have to deal with large multimedia files, like video files queried by the sensornet.

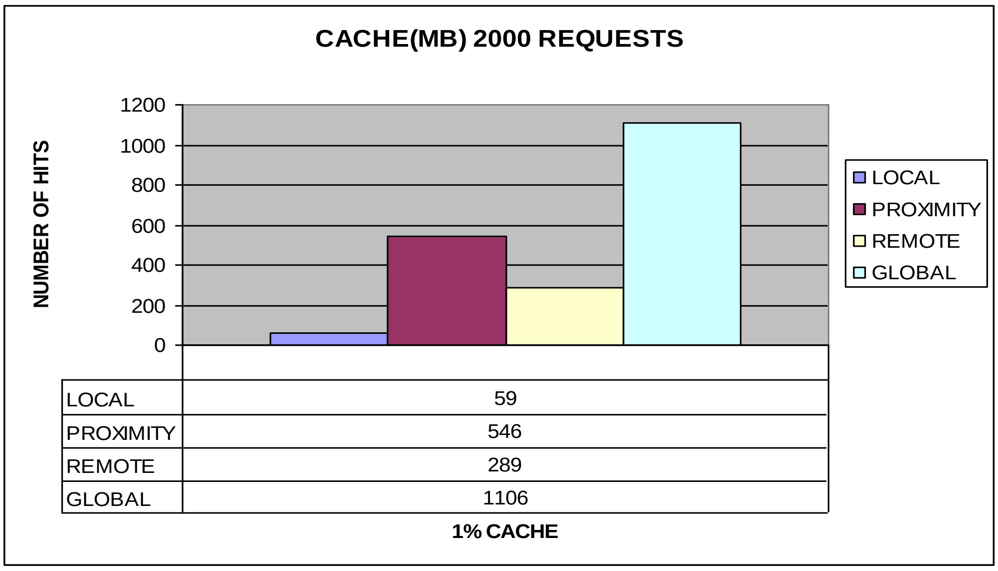
#### 4.3.1.1 Small sized networks



| | 10% CACHE |
|---|---|
| LOCAL | 171 |
| PROXIMITY | 694 |
| REMOTE | 345 |
| GLOBAL | 790 |



| | 5% CACHE |
|---|---|
| LOCAL | 93 |
| PROXIMITY | 601 |
| REMOTE | 320 |
| GLOBAL | 986 |

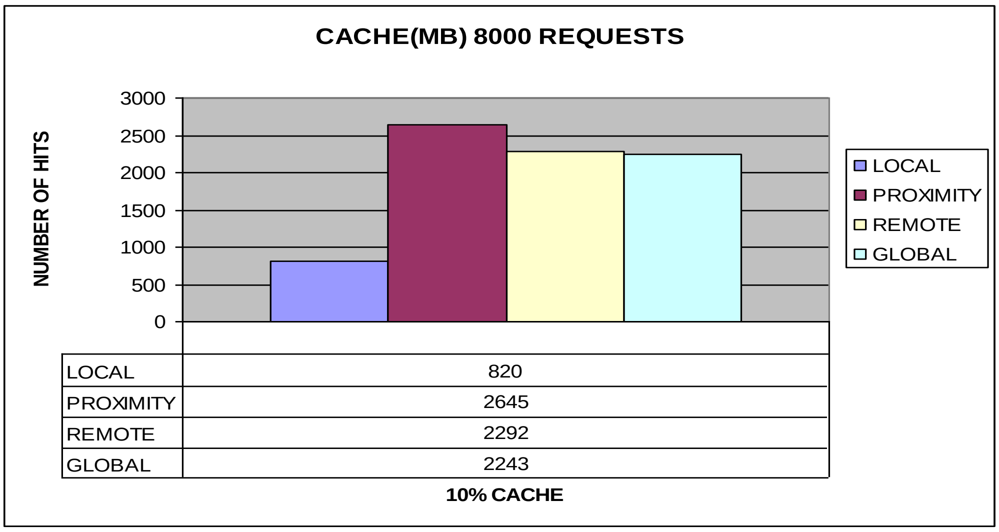


| | 1% CACHE |
|---|---|
| LOCAL | 59 |
| PROXIMITY | 546 |
| REMOTE | 289 |
| GLOBAL | 1106 |

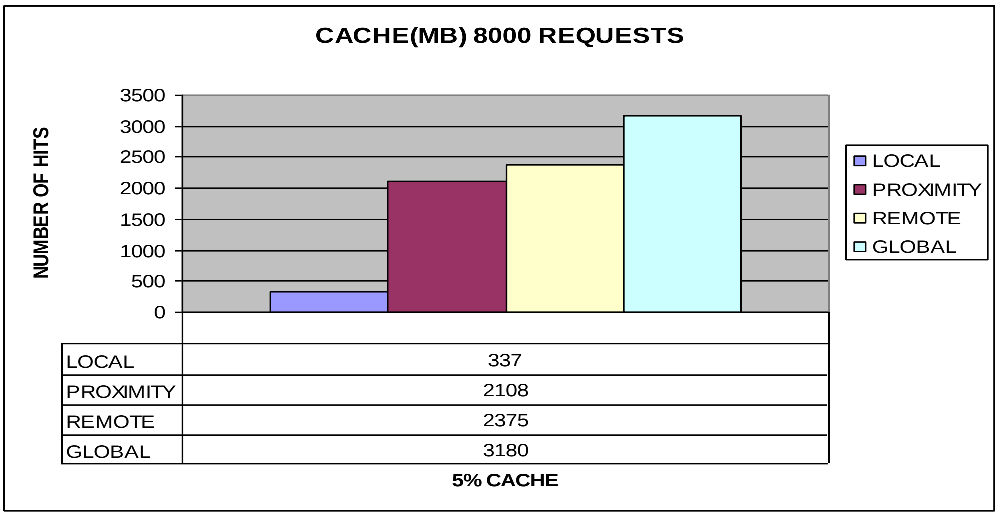
**Figure 3:** An example of our protocol's performance with 2000 requests for MB-sized items. As the cache size decreases the number of local hits decreases and the cache replacement protocol is triggered more often. When a node doesn't have enough space to store the new

*item it computes the cost of all the items it has stored by:* $cost(i) = (si / TTLi)*(now\text{-}t_{k\text{-th access}} / K)$. *The item with the max cost is the victim! When a node decides which item is to be deleted it checks if a mediator node has this item. If true then the item is purged from the cache. If false then the "victim" item is send to the mediator with max residual energy and then is purged..*

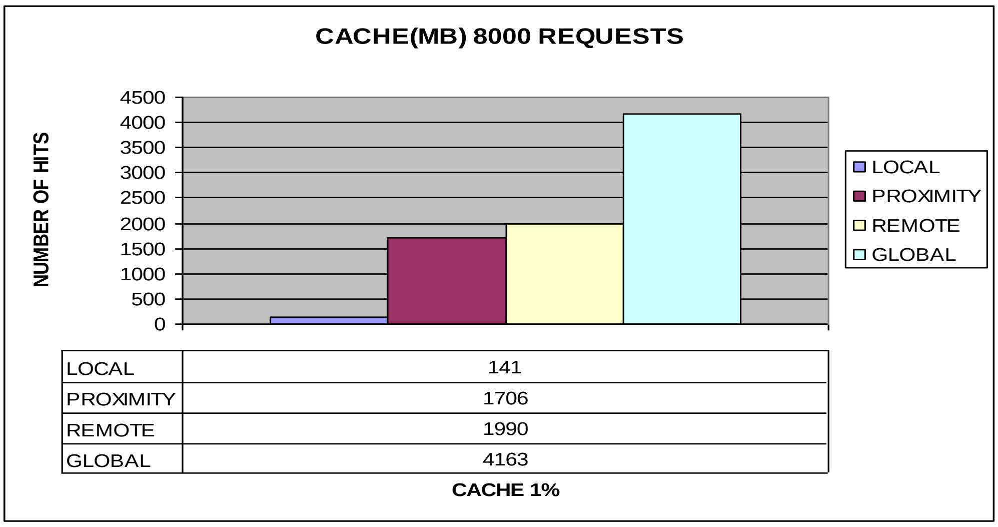
### 4.3.1.2 Large sized networks



| | |
|---|---|
| LOCAL | 820 |
| PROXIMITY | 2645 |
| REMOTE | 2292 |
| GLOBAL | 2243 |



| | |
|---|---|
| LOCAL | 337 |
| PROXIMITY | 2108 |
| REMOTE | 2375 |
| GLOBAL | 3180 |



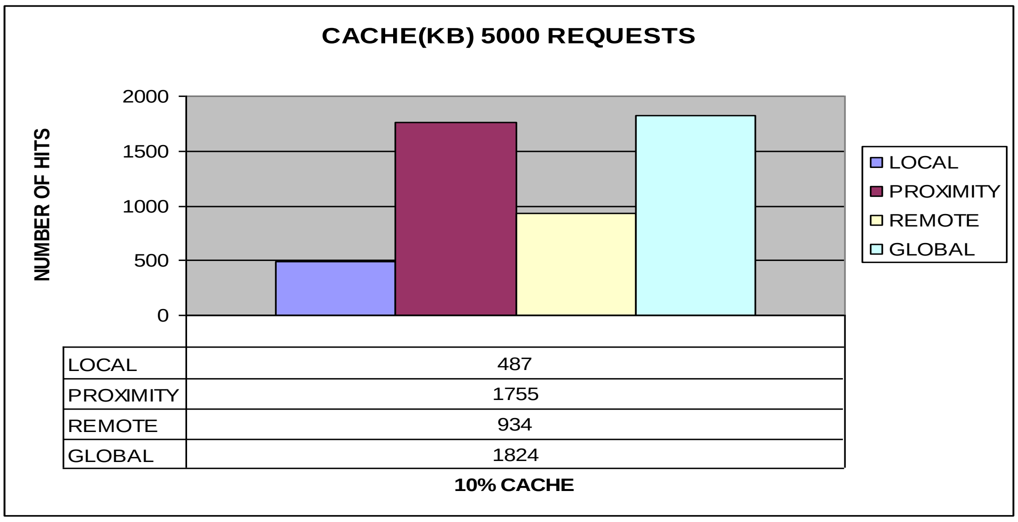
| | CACHE 1% |
|---|---|
| LOCAL | 141 |
| PROXIMITY | 1706 |
| REMOTE | 1990 |
| GLOBAL | 4163 |

**Figure 4:** Impact of cache size on number of hits for a large number of requests. As the number of requests increases our protocols effectiveness becomes more evident. When a request isn't found in local cache our protocol, is trying to possibly locate the request using what knowledge there may be in its neighborhood. Starting with the neighboring node with max RBC value and choosing nodes in descending order of the RBC, until all the 2-hop neighborhood of the requesting node is covered, a group of the 1-hop neighborhood of the requester is selected. Those nodes are called mediators. If the item is found within the 2 hop neighborhood of the requester by the mediators, we have a proximity hit.

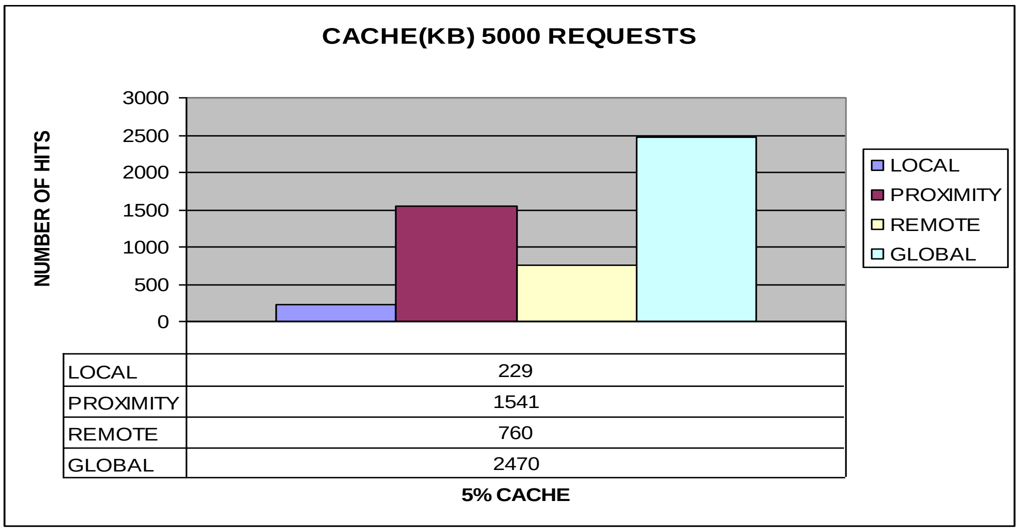
### 4.3.2 Experiments with KB-sized items

Here is the set of our results when the sensornet has to deal with small multimedia files, with size equal to a few kilobytes. Although we expect that WMSNs will deal with MB-sized images or video files it might occur to exchange smaller images as well. To investigate this occasion we performed the same set of experiments for KB-sized files.

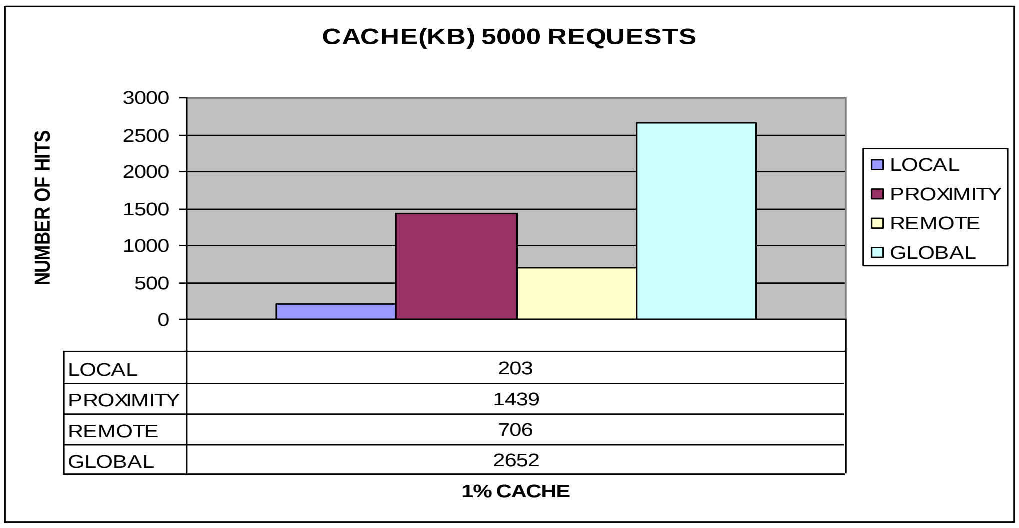
#### 4.3.2.1 Small sized networks



| | 10% CACHE |
|---|---|
| LOCAL | 487 |
| PROXIMITY | 1755 |
| REMOTE | 934 |
| GLOBAL | 1824 |



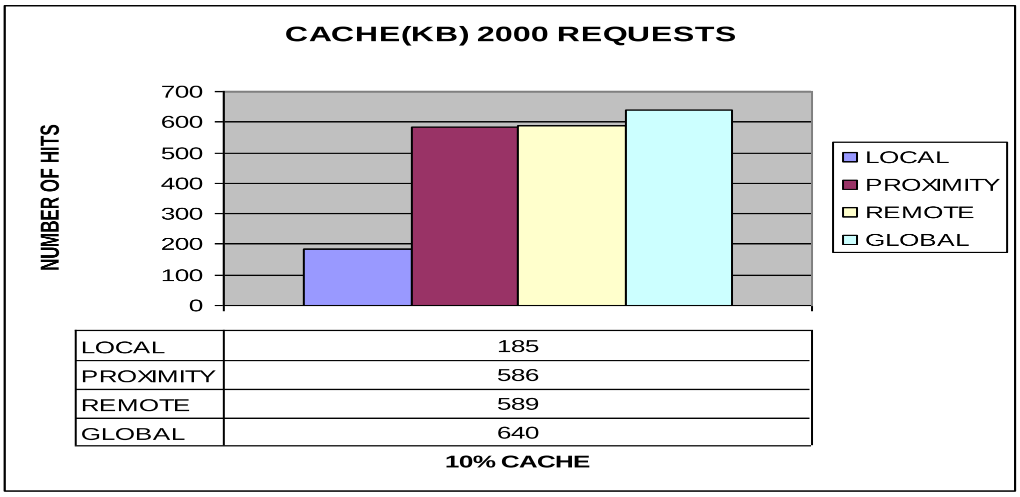
| | |
|---|---|
| LOCAL | 229 |
| PROXIMITY | 1541 |
| REMOTE | 760 |
| GLOBAL | 2470 |



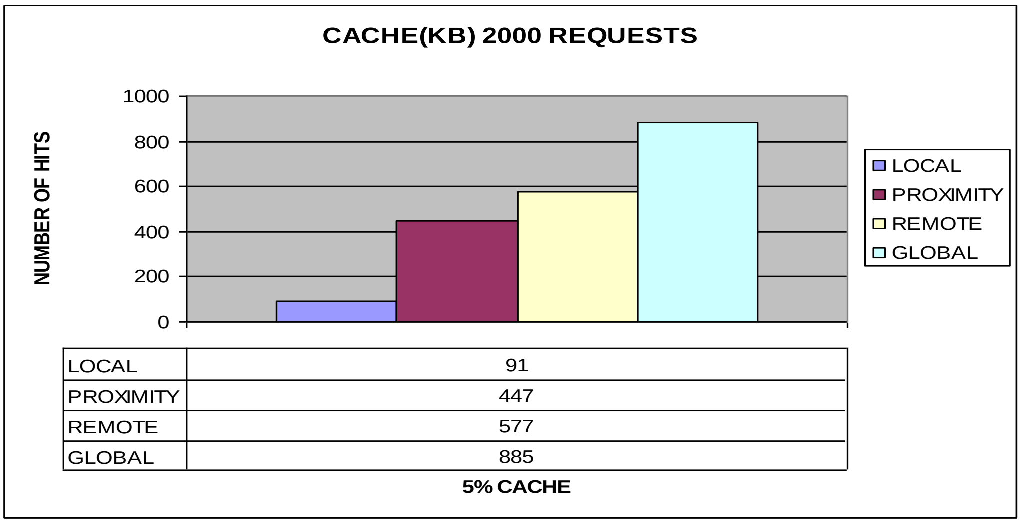
| | |
|---|---|
| LOCAL | 203 |
| PROXIMITY | 1439 |
| REMOTE | 706 |
| GLOBAL | 2652 |

**Figure 5:** As the size of cache decreases the number of global hits increases. With a small sized cache not only local hits but also remote and proximity hits are decreased, since in every node, less items can be stored thus fewer items reside within the 2 hop neighborhood of every requester. Even in the case of limited cache our protocol still is effective. In the worst cache for cache size 1% our experiments showed that at average less than 60 percent of queries are served at the global centers .

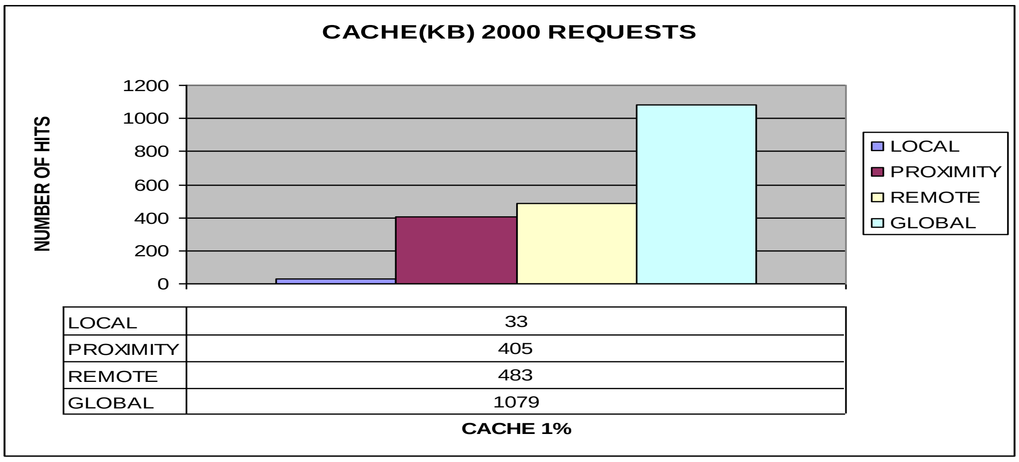
### 4.3.2.2 Large sized networks



| | |
|---|---|
| LOCAL | 185 |
| PROXIMITY | 586 |
| REMOTE | 589 |
| GLOBAL | 640 |



| | |
|---|---|
| LOCAL | 91 |
| PROXIMITY | 447 |
| REMOTE | 577 |
| GLOBAL | 885 |



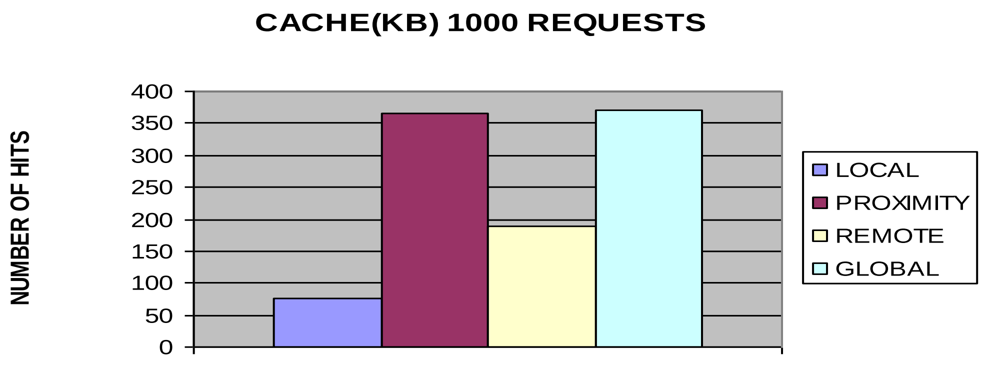
| | |
|---|---|
| LOCAL | 33 |
| PROXIMITY | 405 |
| REMOTE | 483 |
| GLOBAL | 1079 |

**Figure 6**: Impact of sensor cache size on hits. When the request isn't found by the mediators the query within the 2 hop neighborhood is stopped and is forwarded to the Data Center. When a node which resides in the path to the Data center receives a requests it searches it's local cache and if it's not found there, the node checks it's proximity table. If found then we have a remote hit. If not then the path to the Data Center is resumed!

### 4.3.3 Proximity Parameter Evaluation

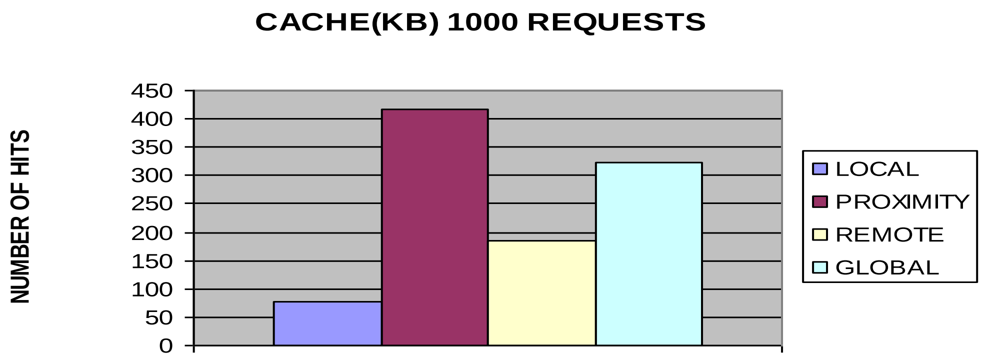
When a node caches an item, it broadcasts to it's mediators, the id of the item it cached and it's remaining energy. Here we see the impact of the proximity parameter, meaning the number of items that a mediator node knows that they reside within its one hop neighborhood.

### 4.3.3.1 Small sized networks



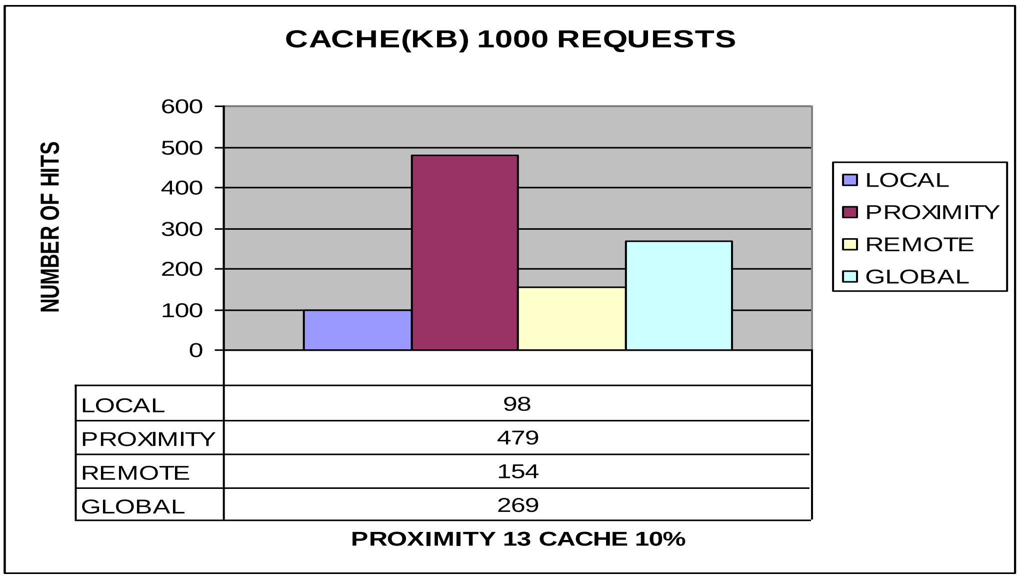
| | |
|---|---|
| LOCAL | 76 |
| PROXIMITY | 365 |
| REMOTE | 189 |
| GLOBAL | 370 |

PROXIMITY 7 CACHE 10%



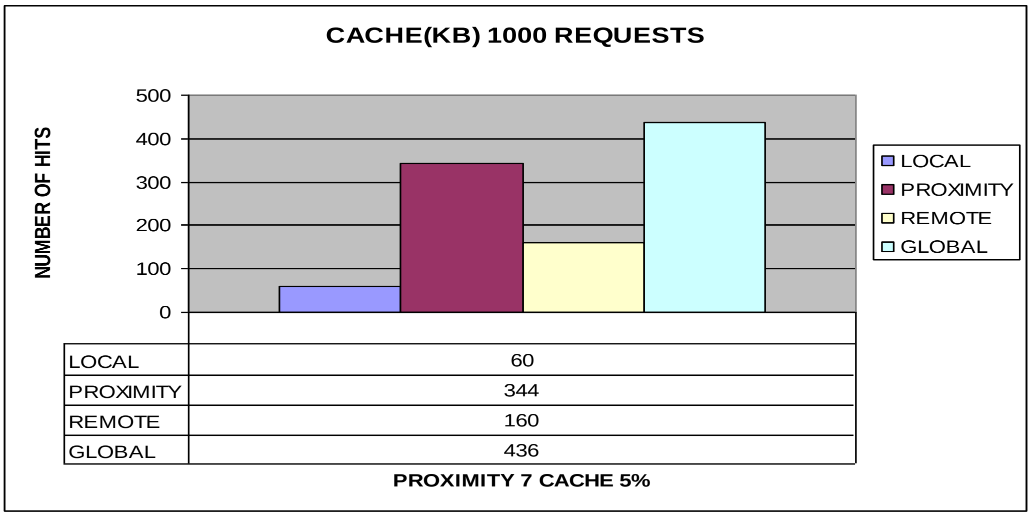
| | |
|---|---|
| LOCAL | 77 |
| PROXIMITY | 417 |
| REMOTE | 184 |
| GLOBAL | 322 |

PROXIMITY 10 CACHE 10%

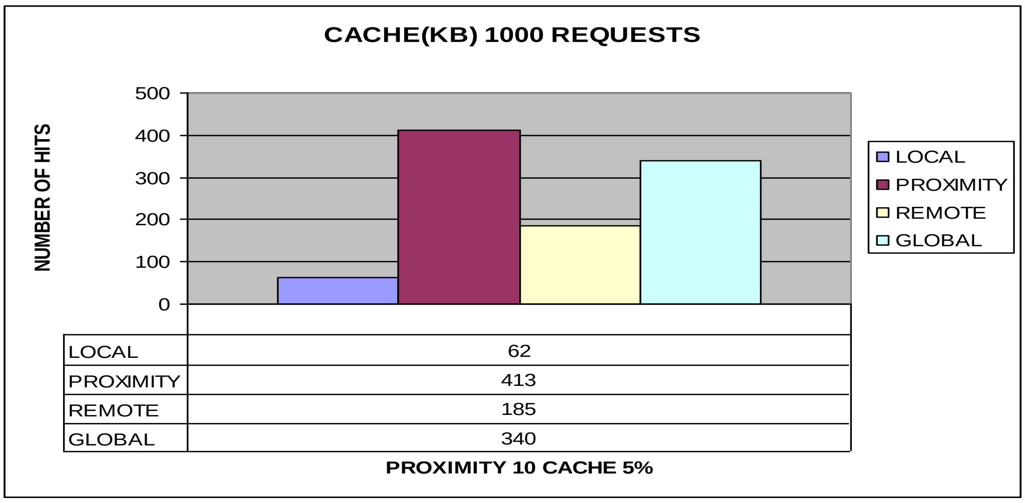


| | |
|---|---|
| LOCAL | 98 |
| PROXIMITY | 479 |
| REMOTE | 154 |
| GLOBAL | 269 |

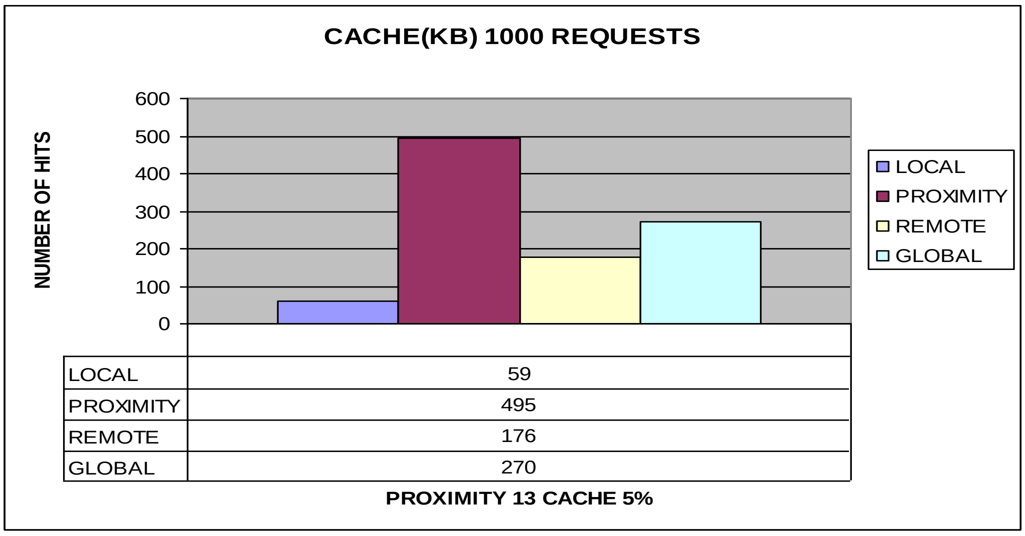
**Figure 7:** For 1000 requests cache size at 10% and proximity value 7, 10 and 13. It is evident that by increasing the proximity parameter the number of global hits are decreased remarkably, since a mediator nodes is aware of more items that reside in it's neighborhood.



| | |
|---|---|
| LOCAL | 60 |
| PROXIMITY | 344 |
| REMOTE | 160 |
| GLOBAL | 436 |



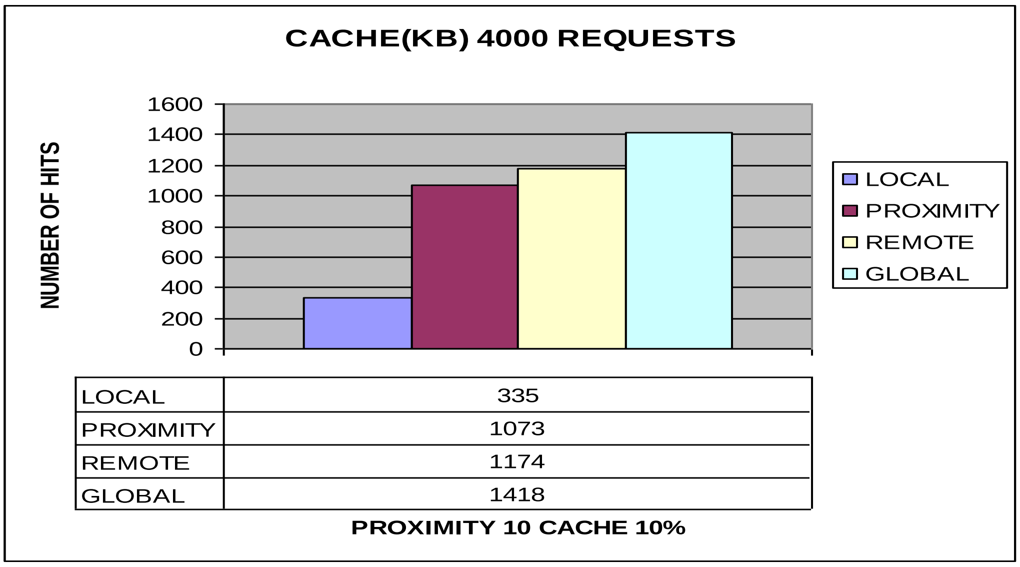
| | |
|---|---|
| LOCAL | 62 |
| PROXIMITY | 413 |
| REMOTE | 185 |
| GLOBAL | 340 |



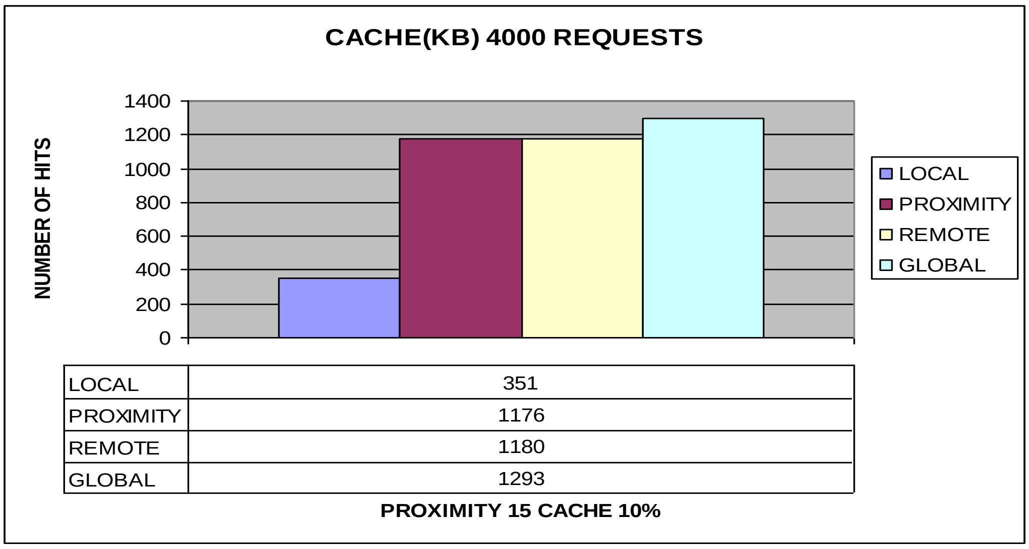
| | |
|---|---|
| LOCAL | 59 |
| PROXIMITY | 495 |
| REMOTE | 176 |
| GLOBAL | 270 |

**Figure 8:** Impact on the number of hits with cache size 5%. As we see the impact of smaller cache on our protocols effectiveness, is smaller when we increase the value of proximity parameter.

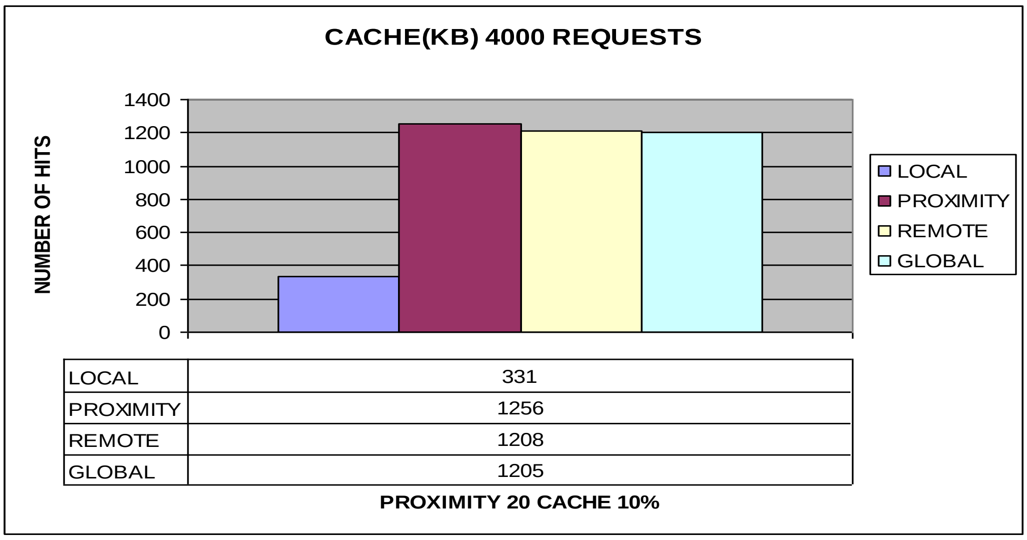
### 4.3.3.2 Large sized networks



| | |
|---|---|
| LOCAL | 335 |
| PROXIMITY | 1073 |
| REMOTE | 1174 |
| GLOBAL | 1418 |

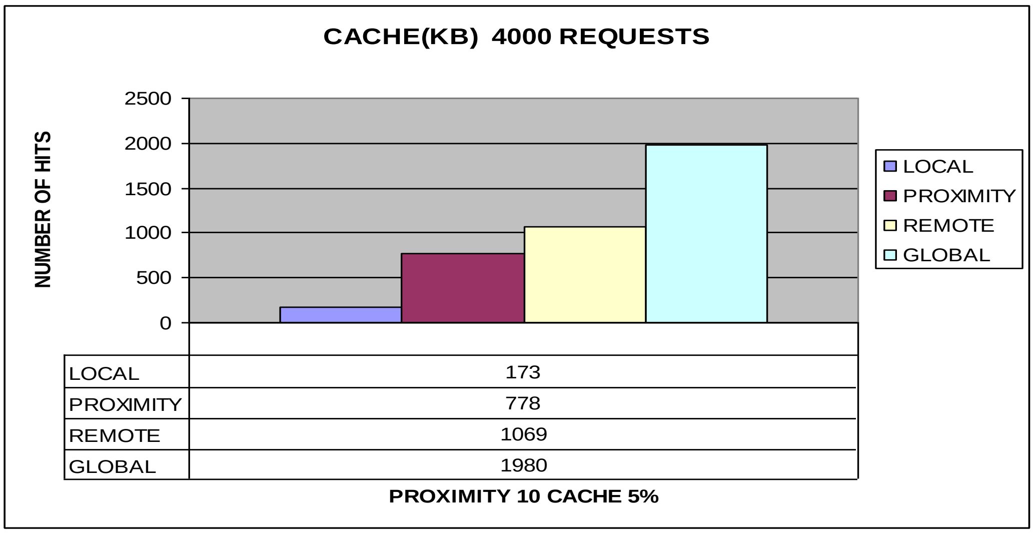


| | |
|---|---|
| LOCAL | 351 |
| PROXIMITY | 1176 |
| REMOTE | 1180 |
| GLOBAL | 1293 |

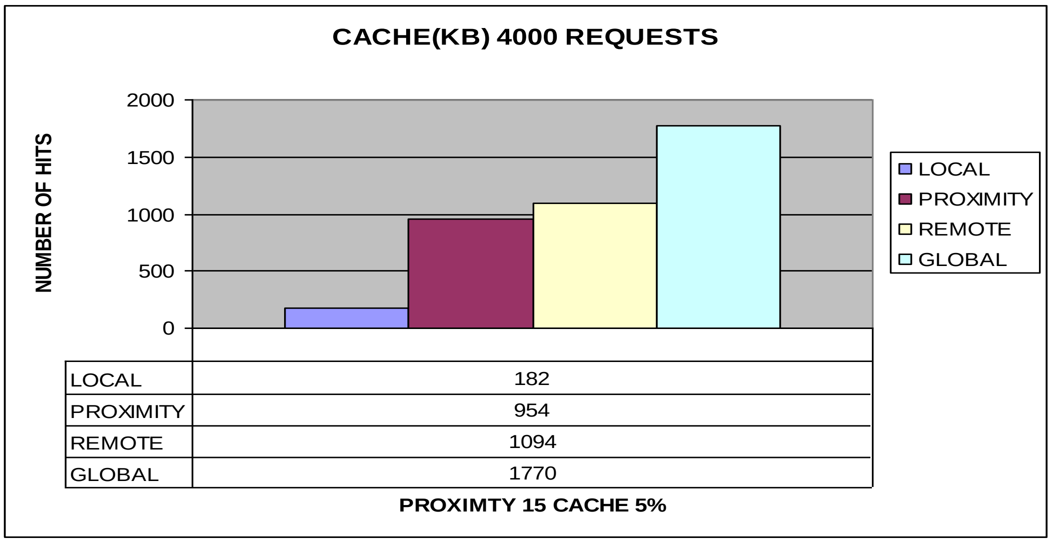


| | |
|---|---|
| LOCAL | 331 |
| PROXIMITY | 1256 |
| REMOTE | 1208 |
| GLOBAL | 1205 |

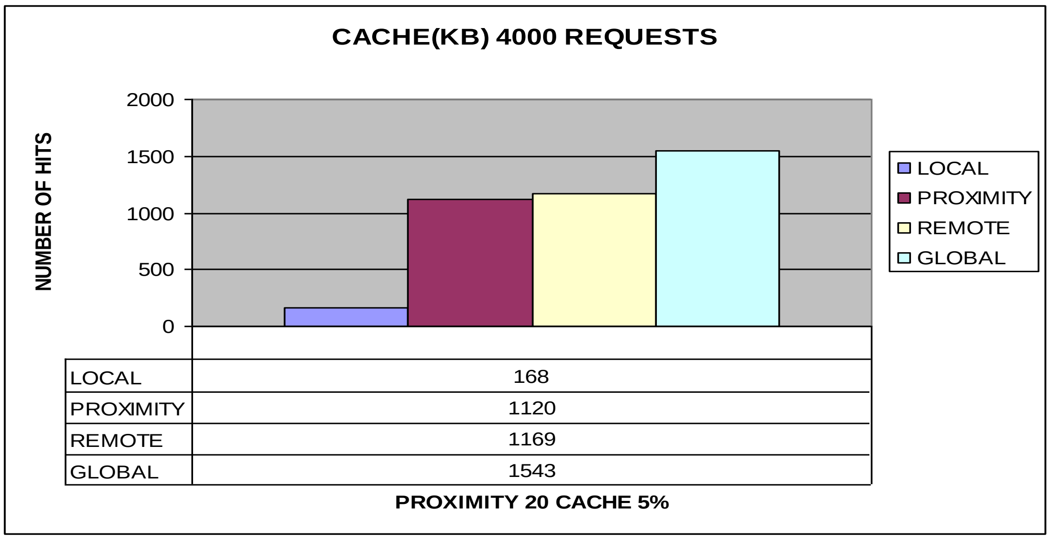
**Figure 9:** Showing the difference in our protocols effectiveness for proximity parameter value 10,15 and 20 for large size networks, KB-sized items and 4000 requests.



| | |
|---|---|
| LOCAL | 173 |
| PROXIMITY | 778 |
| REMOTE | 1069 |
| GLOBAL | 1980 |



| | |
|---|---|
| LOCAL | 182 |
| PROXIMITY | 954 |
| REMOTE | 1094 |
| GLOBAL | 1770 |



| | |
|---|---|
| LOCAL | 168 |
| PROXIMITY | 1120 |
| REMOTE | 1169 |
| GLOBAL | 1543 |

**Figure 10:** We can see that the case of "proximity value 20 with cache size 5%" is near to the case of "proximity value 10 with cache size 10%".

By increasing the proximity parameter we achieve better results. The impact it has on the memory space is minimum since what we store, is the id of the item and the id of the neighboring node which is stored in. The global hits are decreased and on the other hand proximity and remote hits are increased.

## 5. Summary and Conclusion

The recent advances in miniaturization, the creation of low-power circuits, and the development of cheap CMOS cameras and microphones gave birth to *Wireless Multimedia Sensor Networks*. WMSNs are expected to fuel many new applications and boost the already existing ones. The unique features of WMSNs call for protocol designs that will provide application-level QoS, an issue that has largely been ignored in traditional WSNs. Taking a first step toward this goal, this article develops a cooperative caching protocol, the *Node Importance*

*Cooperative Caching* protocol, suitable deployment in WMSNs. The protocol “detects” dynamically which sensor nodes are most “central” in the network neighborhoods and gives to them the role of mediator in order to coordinate the caching decisions. The proposed protocol is evaluated with OMNET++.

## 6. REFERENCES

**[1].** Nikos Dimokas , Dimitrios Katsaros,  Yannis Manolopos . Cooperative Caching in Wireless Multimedia Sensor Networks

**[2].**Dimitrios Katsaros , Yannis Manolopos. Web Caching in Broadcast Mobile Wireless Environments IEEE internet Computing

**[3].** Shlomi Dolrv , Yuval Elovici , Rami Puzis . Routing Betweenness Centrality

**[4].**Ian F. Akyildiz , Tommaso Melodia , Kaushik R. Chowdhury . A survey on wireless multimedia sensor networks

**[5].** Lodewijk F.W. van Hoesel, Stefan O. Dulman, Paul J.M Havinga . Design of a low-power testbed forWireless Sensor Networks and verification

**[6].** N. Chand, R. C. R. C. Joshi, and M. Misra. Cooperative caching strategy in mobile ad hoc networks based on clusters. *Wireless Personal Communications*, 2006.

**[7].** 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.

**[8].** 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.

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

**[11].** 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.

**[12].** 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.