Available online at www.elixirpublishers.com (Elixir International Journal)

Network Engineering

Elixir Network Engg. 107 (2017) 47062-47071

Awakening to Reality

Elixir
ISSN: 2229-712X

Data Caching for Query Optimization in Multi-sink Wireless Sensor Networks

Vipan Arora and T P Sharma
NIT Hamirpur(H.P),India.

ARTICLE INFO

Article history:
Received: 21 April 2017;
Received in revised form: 29 May 2017;
Accepted: 10 June 2017;

Keywords
Wireless sensor networks,
Sensor Node,
Common root,
Caching zones,
Cumulative cache.

ABSTRACT

Wireless sensor networks are generally deployed in inaccessible terrains for monitoring certain physical parameters like temperature, pressure etc. Sensor nodes sense the event and send the sensed data towards sink nodes using multi hop path. Since radio communication is expensive in terms of energy consumption, the sensor nodes typically spend most of their energy reserve on communication during data dissemination and retrieval. Storing data in-network at specific locations within WSNs that minimize packet transmissions reduces the energy consumption, and hence extends its lifetime. Considering limited storage space, limited energy and other constraints of sensors, a data caching mechanism for multi sink sensor networks is proposed. The proposed scheme Data Caching for query optimization in multi sink sensor networks (DCQOM) extracts the common sub tree of the networks first and then forms two caching zones: One from source node to common root and other from common root to sink nodes. The sensor nodes in the caching zones cooperate among themselves to form a larger cumulative cache. A token-based cache admission control scheme is devised, which ensures proximity of cached data closer to sink(s). Simulation results show that DCQOM provides substantial energy saving and data availability as compared to other caching schemes.

1. Introduction

Wireless Sensor Network (WSN) consists of large number of sensor nodes (SNs) which are small in size, low cost and have limited memory, sensing, computation and wireless communication capabilities. SNs measure the ambient conditions from the environment surrounding them. In these applications, SNs are deployed to operate autonomously in unattended environments. In addition to the ability to probe its surroundings, each sensor has an on board radio to be used for sending the collected data to a base-station either directly or over a multi-hop path [1][2].

SNs are energy constrained devices. Energy consumption is generally associated with communication and more the communication more is the energy consumption. The solution to this problem could be the periodic replacement of the node battery. But, this is not viable every time as SNs may be inaccessible in some applications such as monitoring snow avalanches, tsunami or volcanoes etc. Further, different applications may require persistent long-term data collection because the gathered data make sense only if the data collection procedure lasts for months or even years without interruption. Therefore, data collection strategy must be carefully designed to reduce energy consumption at SNs, so as to prolong the network lifetime. [3][4].

In sensor networks, sending large amount of raw data directly to the sink can lead to several undesirable problems. First, the quality of data may be deteriorated by packets losses due to the limited bandwidth of SNs. Second, intensive data collection incurs excessive communication traffic and potentially results in network congestions. Also, excessive communication results in the exhaustion of SNs batteries. Third, intensive data collection leads to excessive energy consumption during sensing. Authors in [5][6] report that the lifetime of a sensor network can be increased from one month to more than eighteen months by lowering the data flow rates of SNs.

Data aggregation [7][8] can conserve transmission energy, but it is not suitable for multi-sink sensor networks in certain scenarios. In multi-sink sensor networks, queries from multiple sinks may query the same event. There are significant advantages of having multiple sinks in the network in terms of latency and energy consumption of information acquisition [9][10]. First, the efficient data gathering trees are formed by the nodes and then the best sink is chosen for transferring the data in multiple sink networks. Second, the mean distance between the nodes and the sink will be decreased in the multi-sink network leading to energy savings and increased lifetime. Third, multiple sinks deployment avoid the single bottleneck problem since in case of disconnection of one particular sink from the network, sensors can still transmit their data towards other sinks, and the network continues to function.

In majority of WSN deployments, SNs continuously sense event with some sensing frequency and pump data into the network. Data dissemination method further moves data towards sink by using suitable routing protocol and/or in-network data aggregation techniques. If sink do not need this data at the time it is generated, it must be stored temporarily in the network. SNs can hold only limited data in their local storages and hence to accommodate entire data generated due to event sensing at a particular point in time, it must be spread across distributed nodes throughout the sensor field. To retrieve such information from the network when sink does

Tele:
E-mail address: vipan.arora@gmail.com

not know the locations where data is spread is also a huge challenge. This can be achieved by caching useful data for each wireless sensor either in its local store or in the nearby neighborhood. Caching plays a useful role in reducing the number of communications from SN to sink by caching the useful data.

Since, large number of SNs may be active (i.e. sensing event) at a given time generating sensed readings with certain frequency. This results in huge amount of data which must be disseminated towards sink. In many scenarios, sink is also one of field SNs having similar hardware constraints except possibly having radio with higher transmit/receive range. Thus, sink can easily be swamped by these large numbers of sensed readings making impossible for it to store them in its limited local storage. This results in loss of precious data which may be required by sink in future to serve queries received from base station (BS) located outside the sensor field.

Even if sink is capable enough to at least receive and transmit all individual readings to base station, it soon drains its energy source. Moreover, not each individual reading is required at base station. Ideally, base station issues queries based on application (control or analysis) running there and sink serves these queries by further querying selected readings from recent past. Therefore, the issue here is to store these huge number of sensor readings generated in a given time window somewhere in WSN. Obvious choice seems to cache readings at nodes on data/query path from source SNs to sink, but it suffers from few problems. First, when path length between sink and active nodes (i.e. nodes sensing event) is small, the storage may still be very less to accommodate readings generated within a given time window. Secondly, readings will be cached throughout the path from sink up to the last node towards event, where as for better performance more and more cached readings must be cached nearer to sink.

To overcome above problems, in this paper, we propose Data Caching for query optimization in multi sink sensor networks (DCQOM) scheme. In this scheme, we extract the common sub tree of the sensor networks and then extract the common root of that sub tree. Two caching zones are proposed one from source node to common root (CZ1) and other from common root to sink nodes (CZ2). Apart from its own local storage, a SN uses storages of nodes from certain region around it to form larger cache storage known as cumulative cache (CMC). Proposed scheme increases data availability to sink and reduces both energy consumption and channel congestion. The ultimate goal of caching zones is to improve the data availability, so, data should be available on time by utilizing the minimum resources.

To increase data availability nearer to sink and to avoid unnecessary replication of a data item, a token-based cache admission control scheme is devised where node holding token can only cache or replace data item. A time-to-live (TTL) based scheme is also developed to avoid sender from fetching stale data and at the same time avoiding undesired data replication. Finally, a data item utility-based item replacement policy is developed so that in case of data item replacement, a data item with least utility is evicted from cache.

Rest of the paper is structured as follows. Section 2 describes related work. Section 3 describes system model. Section 4 describes about data/query path. Section 5 describes about extraction of common sub tree. Proposed caching zones and token based admission scheme are explained in section 6 and 7. Section 8 gives cache replacement policy. Complete simulation study including simulation setup and results and given in section 9. Section 10 concludes the work**.**

2. Literature Survey

Providing continuous information to the sink with uninterrupted communication is a big challenge in designing large-scale sensor networks. Tremendous research had been done in the area of designing the caching techniques for sensor networks. However, little work has paid attention to cooperative caching to increase data availability. This paper targets the problem of efficient data dissemination and tries to solve it by utilizing the memory of SNs by caching the data items in it. Caching in sensor networks is important, since caching sensed information at intermediate nodes can greatly reduce overall communication cost which is the main source of energy consumption.

Yuh-Jzer Joung [11] et al. proposed a data-centric query/storage method ToW. ToW is based on the concept of data centric storage where events and queries meet on some rendezvous node so that queries for the events can be sent directly to the node without being flooded to the network.

Rather than fixing the rendezvous point, they dynamically replicate the point and float them in between query nodes and event sensor nodes according to the relative frequencies of events and queries.However,ToW does not consider the concept of cooperative caching.Nuggehalli et al.[12] addressed the problem of optimal cache placement in ad hoc wireless networks and proposed a greedy algorithm, called POACH, to minimize the weighted sum of energy expenditure and access delay. Chand et al. [13] proposed a cooperative caching strategy that partitions network into non-overlapping clusters based on the physical network proximity. For a local cache miss, each client looks for data item in the cluster. If no client inside the cluster has cached the requested item, the request is forwarded to the next client on the routing path towards the server.

Nikos Dimokas et al.[14] proposed a new cooperative caching protocol (PCICC) as an effective and efficient technique. The essence of these protocols is the selection of the sensor nodes which will take special roles in running the caching and request forwarding decisions. In [15], the authors proved that many-to-many aggregation problem can be converted into multi independent optimization problems and proposed a data aggregation method to aggregate data from multiple sources to multiple destinations. This situation is similar to queries in multi-sink sensor networks, but they do not consider the data sharing among different destinations. A concurrent multi query optimization method TTMQO for single-sink sensor networks was given in [16]. This method can optimize multi query both at the base station (sink) and at the sensors, and it also realized the data sharing among different queries, but it is not suitable for multi-sink sensor networks because it does not consider the multi-sink situation. Sharma T.P et al[17] proposed cooperative caching(COCA) and exploited cooperation among SNs and their dual radio mode to realize efficient caching scheme. Some nodes on this path form cooperative zones in which SNs cooperate among themselves to realize a much larger CMC than individual SN’s storage. The caching scheme ensures maximum data availability to sink and that too in its close proximity. This scheme significantly enhances performance of the network in terms of lower overall energy consumption and fewer data item replacements needed. However, this scheme does not consider the caching for multiple sinks.

Shashi et al. [18] proposed a Steiner tree-based approach to find locations in the network to cache data that minimizes packet transmissions and reduces the power consumption in the network and hence extends its lifetime. Finding locations of the nodes for caching data to minimize communication cost corresponds to finding the nodes of a weighted minimum Steiner tree, whose edge weights depend on the edge's Euclidean length and its data traffic rate. Rahman and Sajid [19] proposed certain ways to improve energy efficiency in WSN that already use some routing technique. This emulates cache where latest sensed data are always known without requiring the sensors to sense and report continuously. They exploit data negotiation, data change expectancy and data vanishing for a given scenario.

In [20], the author proposed cooperative caching which ensures sharing of data among various nodes reduces the number of communications over the wireless channels. The authors proposed a cooperative caching scheme called ZCS (Zone Cooperation at Sensors) for wireless sensor networks. In ZCS scheme, one-hop neighbors of a sensor node form a cooperative cache zone and share the cached data with each other. However, this scheme does not consider the caching for multiple sinks. Jinbao Li et al[21] proposed a query mechanism based on data caching in Multi-Sink sensor networks. This mechanism extracts the common sub tree of a network firstly, and then caches a sink's query results at the common root of a common sub tree according to certain cashing strategies. When some other sinks submit the same query, it only needs to send the cached query results to the query sink. This mechanism can reduce communication cost and improve query efficiency. However, this scheme does not consider the cooperative caching among multiple sensor nodes.Nikos Dimokas et al[22] proposed a new energy efficient cooperative caching protocol (EBCCoCa).Mohammed Aalsalem et al[23] considered combined the two approaches of mobile gateways and multi-hop forwarding to address more constraints for sensor applications. They provided algorithmic solutions that attempt to balance the benefits of both approaches. Based on a given predetermined time deadline, a gateway tour will be established to involve a subset of the sensors. These selected sensors will work as gathering points and will aggregate the other sensors' data. The gathering points/nodes are selected with the aim of reducing the energy expenditures due to multi-hop forwarding. Authors in [24] proposed an effective cache resolution technique, which reduces the number of messages flooded in to the network to find the requested data, which in turn reduces the communication cost and bandwidth utilization

Most of the above works are based on single-sink sensor networks and are not suitable for multi-sink sensor networks. There are a lot of differences between single sink sensor networks and multi-sink sensor networks, such as the network topologies, data sharing among multiple sinks, and etc.To overcome above problems, in this paper, we propose Data Caching for query optimization in multi sink sensor networks (DCQOM) scheme. In this scheme, we extract the common sub tree of the sensor networks and then extract the common root of that sub tree. Two caching zones are proposed one from source node to common root (CZ1) and other from common root to sink nodes (CZ2). Apart from its own local storage, a SN uses storages of nodes from certain region around it to form larger cache storage known as cumulative cache (CMC). Proposed scheme increases data availability to sink and reduces both energy consumption and channel congestion. The ultimate goal of caching zones is to improve the data availability, so, data should be available on time by utilizing the minimum resources.

3. System Model and Assumptions

We assume a large-scale WSN comprising of homogeneous nodes, where each node is aware of its own location and coordinates (x,y) serve as node identification number. SNs are capable of communicating with each other through radio signals using Omni-directional antennas. However, each node is capable of communicating in dual mode i.e. using low power radio and high power radio. All nodes normally communicate using high power radio, but cluster head DNs, use low power radio while communicating with nodes in the zone around it.Network is represented as a directed graph G = (V, E), where V is the set of all nodes {S1, S2, …, Sn}and E is the set of edges between nodes that can directly communicate with each other in single hop i.e. are within communication range of each other. Thus, any pair of nodes, say *(i,j)*, can communicate if they are within distance *d* from each other i.e $dist(i, j) < d$. If node *Ni* can communicate directly with node *Nj*, a corresponding edge e_{ij} exists in *E*. The cardinality of V represents the total number of nodes N_t in the network, i.e. $N_t=|V|$. Without any loss of generality, each node in V is assigned a unique identifier. All SNs are static where as event may move slowly. All nodes have a common, maximum radio range equal to *d*.

The proposed caching scheme is implemented using two tier data dissemination (TTDD)[25] model. In this model, a virtual grid of square sized cells is constructed over entire sensor network. When an event occurs, the SNs initiates grid construction. The data source builds a grid structure throughout the sensor field and sets up the forwarding information at the sensors closest to grid points (henceforth called dissemination nodes). Also, initial path setup from SN to sinks is initiated by SNs by sending initial path setup message to their dissemination node immediately after detecting presence of an event

The query from sinks traverses two tiers to reach SN. The lower tier is within the local grid square of the sink's location and the higher tier is made of the DNs at grid points. The sink node floods the query within a cell. When the nearest DN for the requested data receives the query, it forwards the query to its upstream DN towards the source, which in turns further forwards the query, until it reaches SN.

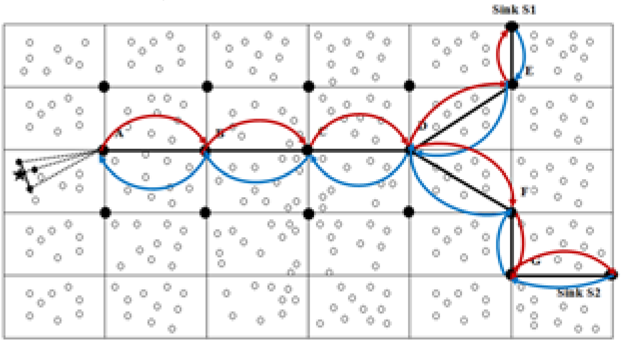

Figure 1

However, there is a little difference in sensing scenario and data/query path set up between TTDD and proposed scheme of DCQOM. In TTDD, once virtual grid is formed path setup is not initiated immediately but initiated only when presence of an event is detected by SNs. This is due to the fact that TTDD considers pure query driven scenario where data is accumulated at DN and subsequently forwarded towards sink only in response to a query from sink. In DCQOM, active SNs sense event periodically and inject sensed observations into the network. Queries from sink may be issued at any times which traverse towards DN(s) with the possibility of cache hits on their way. If required data is not found in caching zones, then at SN a special *FDSA* procedure is initiated to sense the event and aggregate new data item. Therefore, in DCQOM both periodic sensing and query driven data dissemination are considered.

4. Data/Query Path setup

In sensor networks with multiple sinks, different sink nodes may send out the same queries to the source nodes. The sending of same query to source nodes again and again results in wastage of time and energy. To resolve this problem, data can be cached on the common path between source to different sinks. When different sinks may issue the same query, they can obtain the results from those intermediate sensors on the common path, which is named as caching zones. The caching zones should be near to the sink so as to reduce the transmission time and energy consumption.

On detecting an event for the first time, active SNs sense event and send readings to their Initial dissemination point (IDP).To reduce the transmission of readings towards sink nodes; data aggregation is performed at IDP. During a SN sensing interval (SNSI), IDP aggregates similar readings from each active SN's in its cluster to yield a single data item. IDP aggregates these readings to generate initial data item D_{in} and forwards it to its upstream DN's normally towards sink. But in our scheme, we propose to setup two caching zones CZ1 and CZ2.

Source nodes sense data and sends towards multiple sinks. In the first phase, the common path between source and sinks is extracted so as to cache the data on the common path between source nodes and sinks. The sensed data is forwarded upwards towards common root i.e. Final Dissemination Point (FDP) of multiple sinks. All DNs gather information about upstream neighbor during grid formation process (i.e. out of its entire one hop DNs, it selects the one which is nearest to sink). Upstream DN also forwards it to its upstream DN and so on until D_{in} reaches at FDP. This sets up initial path for query and data flow i.e. caching zone CZ1 from IDP to FDP.

In the second phase, common root of subtree i.e. FDP now acts as source node for multiple sinks and sensed data is forwarded to its upstream DN's towards sinks nodes. The sensed data is cached between FDP and sink nodes. This path is used for caching the sensed data from FDP to sink nodes. The FDP decides which sink node will receive which data. This sets up initial path for query and data flow i.e. caching zone CZ2 from FDP to sink nodes. Event is assumed to be lying in one cell at a time and if it slowly moves to another cell path setup procedure similar to TTDD takes place. Figure 2 shows complete path setup scenario.

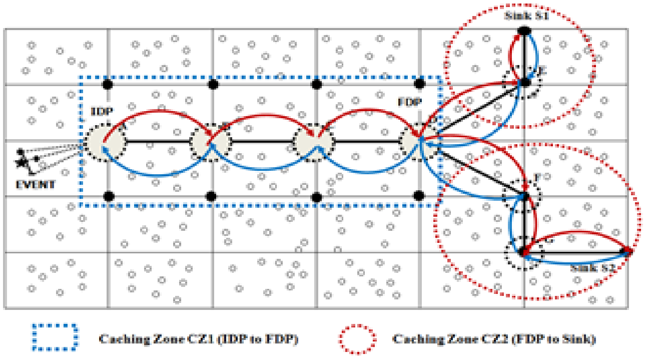

Figure 2. Caching zones CZ1 and CZ2.

After initial path set up, data items aggregated at IDP starts moving upwards towards FDP.Let S_i is the set of nodes in caching zones. Nodes in S_i cooperate among themselves and with DN by sharing their local caches to realize much larger cumulative cache (CMC). In the first phase, the data is stored in the CMC of FDP. Thereafter, the data is stored along the common path between FDP and IDP (Caching Zone CZ1) based on proposed token based cache admission control, which decides where to cache the data. Similarly, in the second phase, sensed data is stored in the respective sink nodes and after the sink nodes are full, the data is stored on the path from sink node to FDP (Caching Zone CZ2).

A query issued by sink nodes travels DN to DN towards FDP and cache discovery procedure tries to locate it for possible cache hit. In case query reaches FDP and still there is no cache hit, then it moves towards IDP. In case there is no cache till IDP, IDP broadcasts query in its cluster using high power radio. On receiving from IDP, each active SN senses the desired parameter and sends reading to its IDP. IDP aggregates these readings to generate data item needed and forwards it towards FDP which further forwards towards sink to serve the query.

5. Extraction of common sub tree

The extraction of common sub tree [21] is based on the spanning trees which are rooted at sinks. The spanning tree of sink S is a tree which is rooted at S and includes all the sensors in the network. In sensor networks with multiple sinks, different sink nodes may query the same event to source nodes. By sending the same query again and again results in wastage of time as well as energy. The solution to this problem lies in caching the data on the common path from source node to different sinks. When different sinks issue the same query, then instead of obtaining the results from source nodes, they can obtain the results from those intermediate sensors on the common path. These intermediate sensors form the caching zones . The caching zones should be near to the sink so as to reduce the transmission time and energy consumption. The two different sinks S1 and S2 query the data produced at source node A. The data transmission path for sink S1 is A-B-C-D-E-S1 where as query forwarding path for S1 is S1-E-D-C-B-A and data transmission path for sink S2 is A-B-C-D-F-G-S2 as query forwarding path for S2 is S2-G-F-D-C-B-A The query path D-C-B-A is common among both the sinks. If caching is done from node D towards node C and B then it will save lot of energy. To do so, we propose to extract common sub tree from different sinks to source. In a sensor network with *n* sinks, the spanning tree of each sink is denoted by T_1, T_2… T_n and the direction of the edges is from child to its parent.The common subtree of n sink's spanning tree is:

Subtree t is the common subtree of tree T_1, T_2… T_n if:

- S contains all the vertices of T and V(T) equals V(t).
- For any non-root node *s* in *t*, *s*'s parent node in *T1*,*T2*,…,*Tn* is the same with each other
- There does not exist any other subtree *t'* which belongs to *Ti*(1≤*i*≤*n*) such that *t* is a subtree of *t'* and *t'* satisfies the above two conditions

In the figure 2, query path D-C-B-A is common sub tree. Common root(Final Dissemination Point i.e FDP) is the nearest sensor to all the sinks in a CS.CR is the root of a CS. In Figure 2, D is the CR(FDP) of corresponding CS.Therefore, by caching the data at CR(FDP) can decrease energy consumption and response time of queries. For a sensor *S, S* is a common leaf node (CLN) if *S* is a leaf node and *S* has the same parent node in all the sinks' spanning trees. The edge between sensor *u* and sensor *v* is a common edge (CE) if there exists an edge between *u* and *v* in all the spanning trees and they have the same direction. Sensor *u* is a common inner node(CIN) if the edge between sensor *u* and sensor *v* is a CE, *u* is the child of *v*, and *u* is not a CLN.If *t* is a CS and *t'* is a subtree of *t* then *t'* is a common branch(CB). Common message packet(CMP) is a data packet transferred from a child to its parent along a CE and it is used to notify a parent node the subtree rooted at its child is a CB

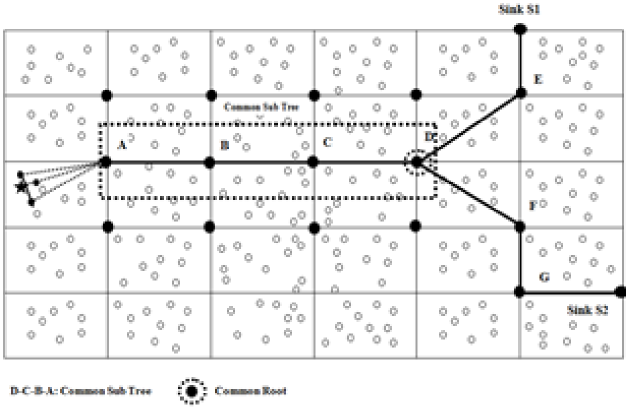

Figure 3. Extracting common root of sub tree.

Every SN maintains a common node flag (CNF). If the value of CNF is 1, then this SN is in a common sub tree(CS). Otherwise, it does not belong to any common sub tree(CS). The Common Branch Flag (CBF) indicates whether a sub tree is a common branch (CB) or not. If the value of CBF is 1, then this sub tree is common branch. Every SN has to maintain a set of CBFs to ensure the sub trees rooted at this sensor's children has a corresponding CBF. To extract a sub tree, we purpose Common Sub Tree Extraction Algorithm (CSEA).

Common Sub Tree Extraction Algorithm(CSEA)

Preliminaries:

CS	Common Sub tree
CR	Common root
CLN	Common leaf node
CE	Common Edge
CB	Common Branch
CMP	Common message packet
CIN	Common Inner Node
CNF	Common Node Flag

```
if (the current sensor S is a CLN)
    Set CNF=1
    S sends CMP to its parent along a CE
else if(S receives a CMP from its child C)
    S sets its CNF=1;
    S sets C's corresponding CBF=1;
    if (all S's children's corresponding CBFs are 1)
        if(S is a CIN)
            S sends a CMP to its parent along a CE
End
```

After extracting the sub tree, the next step is to find the common root (CR i.e FDP). After running CSEA for a certain period of time, one of the sinks sends out a command packet to all the sensors in the network to notify them that CSEA has been completed and each CS(FDP) can mark out their own CR(FDP). When a sensor receives any sink's CR(FDP) marking command packet, it will run Common Root Algorithm (CRA) as shown below to mark out its CR(FDP).

Common Root Algorithm (CRA)

Preliminaries

CS	Common Sub tree
CR	Common root
CLN	Common leaf node
CE	Common Edge
CB	Common Branch
CMP	Common message packet
CIN	Common Inner Node

```
sensor S receives a CR marking command packet from sink
if(S's CNF = 1)
    mark S as a CR
    for(for S's each child $C_i$)
        if($C_i$'s corresponding CBF=0 && $C_i$ is the child of S in Sink$_c$ spanning tree)
            S sends a CR marking command packet to Ci
End
```

6. Proposed Caching Zones

The proposed caching zones CZ1 and CZ2 are formed by exploiting the low power radio mode of SN's. In our approach, as shown in Figure 3, a circular region of radius $R_L/2$ around each DN, IDP and FDP is defined as caching zones on the common path from source to different sinks. Let S_i be the set of nodes in caching zones. Nodes in S_i cooperate among themselves and with DN by sharing their local caches to realize much larger cumulative cache (CMC). All nodes in S_i communicate with each other and with DN using low power radio for every cache management activity. The reason for taking radius of caching zones as $R_L/2$ is two-fold; first all cache nodes in caching zones should communicate in single hop with each other and second to utilize low power radio for caching so as to conserve energy. Each DN probes its caching zones by broadcasting special signal in it and each SN in S_i responds with a tuple comprising its id (geographical coordinates) and residual energy. This way, DN becomes aware of nodes available for caching and prepares an indexed list of node-ids in descending order of residue energies.

The query mechanism is as follows. Whenever a sink node receives a query, it will first check its own cache. If data item is found there and is valid, query is immediately processed and it won't be sent further. If data item is not found in local cache, sink will send the query downwards towards CMC formed around it by broadcasting query in its caching zone CZ2 using low power radio. Nodes in Caching zone having copy of data item in its local storage responds by transmitting data item to corresponding sink and query is serviced. In case of both local and CMC miss, sink sends query downwards to FDP on query/data path. At FDP it first checks local cache for data item and then CMC formed around it. If copy of data item is not found in its local cache or CMC, query is forwarded to next DN on query/data path towards IDP. In this way, query travels downstream towards IDP until data item is found at local cache or CMC of a DN.

In case copy of data item is not found in local cache or CMC of any of DNs including IDP, IDP initiates Fresh Data Sensing and Aggregation process (FDSA) to answer the query. In *FDSA*, IDP uses high power radio to broadcast a query which specifies type of parameters to be sensed. Active nodes immediately sense the required parameter about event and send it to IDP. IDP aggregates these individual readings from multiple sensors and infers the data item. Once data item is available, query is immediately serviced by sending data item towards sink following the path followed by query in reverse.

While data item moves upwards towards sink one hop at a time (i.e. DN to DN), a Cache Admission Control strategy at each DN decides whether to cache data item at its local cache or in CMC or not to cache at all. Further, in case a data item is needed to be evicted out of local cache or CMC at any node, a proper Item Replacement Policy is devised to accommodate new item in case cache is full.

7. Token Based Cache Admission Control

SNs are sensing physical phenomenon and data items are periodically generated at IDP and forwarded towards FDP and thereby to sink nodes. When query is issued by sink node, it travels towards FDP to IDP and is served either by cache hit (local or CMC) at some DN or by initiating DSA procedure at IDP (if it require data which is not cached in the network). In both the cases data items travel towards sink. A data item once utilized by sink to serve a query may again be required to serve another query since it still falls in a valid time window of second query. As sink's local storage is also limited, during first and subsequent usage of a data item many other queries might have been served by sink. This causes old data item to be overwritten by some other data item. Hence, even if a data item has been utilized by sink and stored in its local cache, still there is a need to cache it elsewhere in the network so that when needed, sink can fetch it again from the nearest location.

IDP	Initial Dissemination Point
FDP	Final Dissemination Point
ACM	Active Cache Manager is a node which can cache and replace data items.
DAS	Data Sensing and Aggregation process
CMC	Cumulative Cache
CT	Caching Token. A special token that empowers a node to become ACM
CIT	Cache Intention Token. A special message generated by node showing interest to become ACM
TTL	Time to live

To increase proximity of the data items nearer to sink, it is always better to start caching data items nearer to sink. First of all data items are cached at sink (in its local cache), then at its downward path towards FDP, then at downward path towards IDP, DN's cache (local cache and its CMC), so on until IDP's local cache and its CMC is full. But, this suffers from following problems:

(i) When a data item reaches at a DN or is first time aggregated at IDP, how this node knows whether to cache data item or forward it towards upstream DN? If it forwards data item without caching and there is no free space at any of upstream DN, data item might have to be transmitted back downwards for caching causing major overheads or might have to be accommodated by evicting some valid data item at some upstream DN which itself may be required soon. On the other side, if on the availability of free space a node caches data item before forwarding, multiple copies of a data items may exist and except for one none of which is utilized ever.

(ii) To solve above problem, there seems a simple approach that some flag is set or some other indication is provided at each DN to show whether its cache (local and CMC) is full or still has space to accommodate a data item. Once cache is full, node can pass this indication to downstream DN enabling downstream DN to decide on data item crossing through it. In this way caching can start from sink towards IDP. But, as each data item has a Time-to-Live (*TTL*) value and on its expiry data item becomes stale. While a downstream DN is caching, many data items at an upstream DN may become stale and thus provides opportunity to cache data items at this upstream DN which is nearer to sink. How can a downstream DN know about such an upstream DN?

To solve above problems, a token based cache admission control scheme is devised and is given as in flowchart in figure 4 In this scheme, no DN other than the one possessing a caching token (CT) caches data item. *CT* is a simple small sized message which tells a DN to be ready for caching data items crossing through it and all other DNs simply forward data items. *CT* passing between DNs is regulated both by sequential flow and another special token called as cache-intention-token (*CIT*). *CIT* is used by a DN to show its intention to cache data items. Following symbols/notations are used:

The caching process is in two phases. In the first phase caching is between sink nodes and FDP and in the second phase caching is between FDP and IDP.For caching between sink nodes and FDP,FDP acts as source whereas for caching between IDP and FDP,FDP acts as sink. Therefore, in our proposed scheme FDP acts as both source as well as sink.

Initially, sink possesses *CT* and caches all data items and no other DN caches data items at this point. A node (sink or any DN) which is currently active for caching is called here onwards as Active Cache Manager (ACM). When sink's local cache is full, it next uses CMC formed by cooperating nodes in its caching zone. Using low power radio sink broadcasts data item and id of a node where data item is to be cached so that no other than the node specified by id caches its copy. Nodes from caching zone are selected sequentially as per indexed list formed at sink as above i.e. first node with highest residue energy, then second highest and so on until all nodes in caching zone exhaust.

Once CMC is also full, sink finishes the role of ACM and passes CT to its first neighboring DN (called FDN here onwards) on data/query path towards FDP. This tells FDN to get ready to act as ACM. A DN including sink must have some minimum number of storage locations (Smin) free in its cache (local and CMC) to act as ACM, otherwise CT is simply forwarded to next downstream DN. If FDN becomes ACM, like sink it first caches data items crossing through it in its local storage and then in its CMC. Once its CMC is exhausted, it passes CT to its downstream neighboring DN. This way caching and token passing continues until ACM is found or cache of FDP is exhausted. On exhausting FDP's cache CT is forwarded to sink with special indication for intermediate DNs to forward it upwards until it reaches at sink and circle is completed.Similary,FDP now acts as sink for second phase and IDP acts as source. Same caching procedure is performed between IDP and FDP.

However, cached data item at local cache or CMC of a DN may become stale due to expiry of its TTL value and corresponding location thus becomes available to cache new data item. If number of available free locations in cache (local and CMC) reaches some minimum number Smin, DN becomes candidate to act as ACM. To increase cached data proximity nearer to sink, this DN proactively sends its intention to act as ACM by sending Cache-Intention-Token (CIT) to neighboring downstream DN and neighboring DN keep passing it to its downstream DN until it reaches at a node acting as ACM. When an ACM receives CIT from upstream DN, it passes CT back towards this upstream DN. On receiving CT, this upstream DN now becomes ACM and proximity of cached data nearer to sink is thus increased. Also, if CIT from multiple upstream DNs is received, the one nearest to sink is preferred.

The reason for sending CIT only downstream is that all upstream DNs are nearer to sink than DN itself and sending CIT to them will rather decrease the cached data proximity.

When FDP's CMC is exhausted, FDP sends *CT* back to sink on data/query path and completes circle first time. Since sink may not be at a single hop distance from FDP,*CT* moves hop by hop towards sink and each DN remembers the direction of *CT*. Once this circle is complete, the free-falling nature of token passing as in case 1 diminishes. This is due to the fact that if no downstream DN has *Smin* locations free to become ACM, the free-falling nature may continuously force *CT* to move around circle without finding a candidate for ACM.

Therefore, when *CT* after completing circle reaches at sink and sink even does not have at least *Smin* locations available (i.e. either completely full or <*Smin* available locations), still it is forced to act as ACM (called forced ACM). If all locations at sink are occupied, it uses item replacement policy to overwrite some existing data items to cache new ones and if some locations (<*Smin*) are available, it first utilizes those before applying item replacement policy

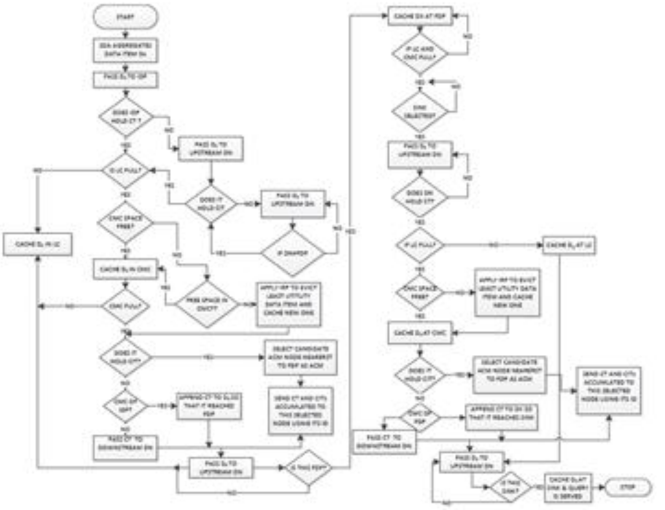

Figure 4. Flow Chart.

8. Cache Replacement Policy

There comes a situation when cache (local and CMC) of a DN (including sink and IDP) is full and new data item must be stored there. To accommodate new data item there is no option other than overwriting an existing data item in cache.

The victim must be selected based on some policy which results in minimum loss of information and minimum overheads.

All the parameters that are cached have Time To Live(TTL) value. As data items have TTL value, when they expires lots of storage space becomes free. So there has to a mechanism for requesting back the tokens. A DN including sink must have some minimum number of storage locations(L*min*) free in its cache to act as a ACM. So whenever some node have number of available free locations in cache(Local+CAZ)reaches L*min* , it becomes candidate to act as ACM. To increase the proximity of FAD(Frequently accessed data items) nearer the sink and LFAD(less Frequently accessed data items) nearer to the source, DN needs to request the token back.

Active nodes sense data from the event and hence data items are aggregated at IDP. Also, depending on the type of event property measured, different data items have different values of *TTL*. Hence, data items become invalid randomly in cache despite the fact that initially they are stored sequentially one after another in the order of arrival in cache. Data items are first stored in local cache of a node and then starting with the first cache node in set S_i till the last node in the set (Nodes in *SCS* are sequenced in decreasing order of remaining energies). A utility based criteria to find an item with least utility is used. But utility function is computed from two factors:

Popularity: The access probability reflects the popularity of a data item for a sink. An item with lower access probability should be chosen for replacement. Access probability Ai for data item Di in local cache of a node is given as:

$$Ai = {a_i}/{\sum_{k=1}^{N} \mathbf{a_k}}$$

Where a_i is the mean access rate to item D_i and N is the total number of data items in the local cache of a node. If data item is at CMC, it must be stored in local cache of one of SNs in S_i. Access probability of this data item is also calculated with respect to total number of data items in local cache of that particular SN and not total number of data items in CMC. However, if access probability of a data item with respect to total number of data items in CMC is computed, a large amount of overhead is required every time to compute access probability. This overhead includes communication among all nodes forming CMC and may prove costlier in terms of energy consumption and delay. Therefore, while dealing with CMC, one SN from *SCS* at a time is taken to find victim data item for replacement. This node selection to find victim data item in CMC takes same ordering of nodes as taken while filling cache first time.

Lifetime: An item with shorter TTL value remains valid only for limited period and hence is better candidate for replacement. But, if its access probability is high, it may still be accessed many times in its limited remaining valid time. Based on above factors, utility of data item Di is computed as follows:

$$\text{utility}_i = A_i \,.\, \text{TTL}_i$$

Item replacement policy based on utility of a data item is proposed as follows:

(i) Data item is replaced only from the cache (local or CMC) of the ACM.

(ii) Data item with least utility value as per function is first replaced in local cache of the DN. This continues until N new data items are replaced in local cache, where N is the maximum number of data items that can be stored in local cache of a node.

(iii) If N replacements have already been made in local cache of a DN holding LTR (i.e. ACM) and new data item is to be cached, it is cached in CMC (i.e. nodes in SCS) of that DN. Nodes in SCS are also selected in the same order they are selected during cache admission and victim data items are found. In each node N replacements are done.

(iv) Immediately after Nth replacement in last node of SCS, token is passed to downstream DN making it forced ACM. Any new data items are cached in new ACM's cache, either at some available locations or by applying above item replacement policy.

9. Performance Evaluation

In this section, we evaluate the performance of proposed scheme of Data Caching in multi sink sensor networks (DCQOM) by performing simulations under different scenarios. We consider 500 SNs. The performance is evaluated by comparing it with COCA[17] based on three metrics: Number of cache hits, Number of item replacements and Energy Consumed. *Number of cache hits* gives the total queries served from cache (local or cumulative) without the need to fetch data by initiating DSA process. This ultimately measures the effectiveness of the proposed *CC* scheme to serve queries. *Number of item replacements* performed indicates the capability of the scheme to retain valid data items in cache. Smaller number of data item replacements indicates better data items retaining capability in cache. Energy consumption is defined as the overall communication energy consumed by the network on data/query path. The default simulation setting has a square sensor field of size 700 × 700 m^2 in which *N* (500) SNs are uniformly distributed. Some of these SNs act as sources and generate data packet. Simulation model is run 100 times and the observation is based on the fixed number of SNs. There are two sinks in the sensor field. The size of data item/query packet is 64 bytes and default cache size is 320 bytes. The high power transmission range of each SN is 100 m and low power transmission range is 25 m.In first setup, Poisson distribution is used to generate queries with mean query generate time of 0.2 s to observe the effect of cache size on evaluation metrics. In second setup, mean query generate time is varied from 0.1 s to 1.2 s while keeping cache size fixed. We categorise all data items under ten classes, where data item from each class has different TTL value. All SNs manage small portion of their memory as cache where they can store data items and are able to delete them when their TTL values expire. Table 1 gives the summary of parameters used for simulation and are varied to study their effects

9.1. Effects of Mean Query Generate Time

In this section, we evaluate the performance in terms of number of cache hits (Cache size is fixed at 5 data items) to mean query generate time(T_m).Figure 5 shows the number of cache hits with T_m. Initially cache hits are more for both DCQOM and COCA and it reduces significantly with increasing T_m.The number of cache hits in DCQOM is 6% more as compared to COCA (Cooperative Caching).This is because DCQOM does caching of data items from source to common node and from common node to different sinks.

Table 1. Simulation Parameters.

Parameter	Value
Number of SNs	500
Size of sensing region	700X 700m^2
High power transmission range of a node (*RH*)	100m
Low power transmission range of a node (*RL*)	25m
Cell size	Square with 75*m* diagonal
Data item and query size	64 bytes
Default cache size	320 *bytes* (can hold 5 data items)
Initial energy level of each node	10 J
Simulation time	200 second
Energy consumed in transmitting data or query in high power mode	0.0024 *J*
Energy consumed in transmitting data or query in low power mode	0.0010 *J*
Energy consumed in receiving data or query in high power mode	0.0018 *J*
Energy consumed in receiving data or query in low power mode	0.0008 *J*
Node mobility	Stationary
Distribution Type of SNs	Uniform
Type of Sensor Field	Two dimensional plane

These results in more caching space available in DCQOM compared to COCA. This results in more cache hits in DCQOM as compared to COCA. Due to small T_m data items in cache remain valid for comparatively large number of queries issued by sink which results in more cache hits. However, when T_m is large, the difference in cache hits in DCQOM and COCA vanishes due to the fact that data items in cache become stale due to expiry of *TTL* despite cache having sufficient free locations. Figure 5 shows the number of cache hits with T_m

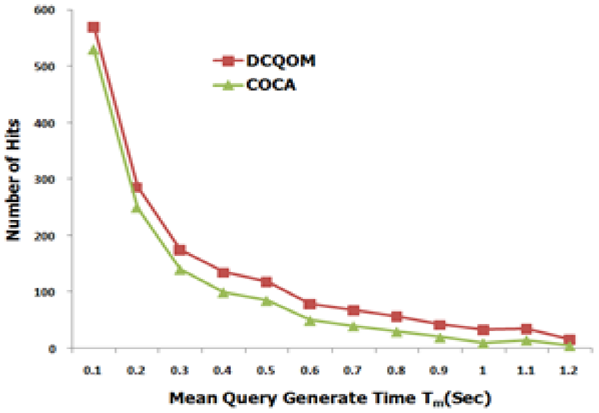

Figure 5. Effect of Mean Query Generate Time on Cache Hits in DCQOM.

Figure 6 shows the comparison between local and CMC cache hits. From the figure, it is clear that contribution of CMC hits to total hits decreases with increasing T_m. This is due to the reason that our token based cache admission control first fills local cache and then starts filling CMC, therefore for large T_m, data items in local cache keep on becoming stale, thereby pushing the data to CMC.

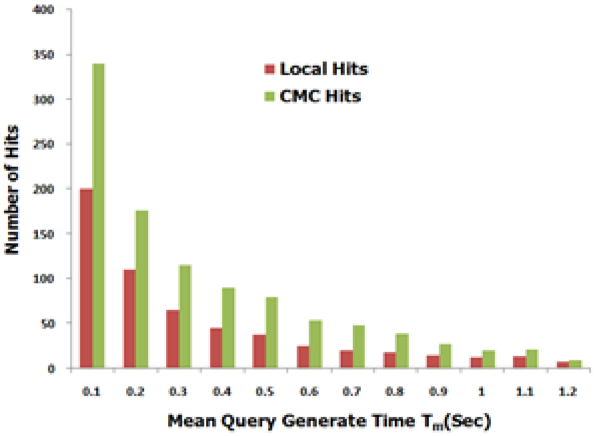

Figure 6. Effect of Mean Query Generate Time on Local and CMC Hits in DCQOM.

In this section, we evaluate the performance in terms of impact of mean query generate time T_m on data replacements. Figure 7 shows this comparison. It is clear from the figure, number of replacements in DCQOM scheme are much lower as compared COCA and Non corporate Caching scheme (NOCOCA). This is because in our DCQOM scheme, number of caching locations are much more as compared to COCA because of two phase caching. Since number of caching locations is more, it results in less number of replacements and the number of data items becoming stale exceeds the number of new data items to be cached.

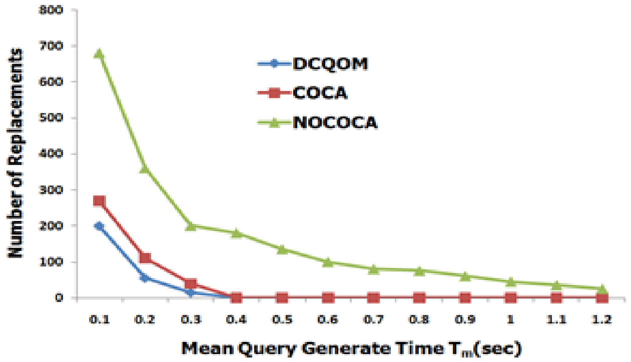

Figure 7. Effect of Mean Query Generate Time on Number of Data Item Replacements.

In this section, we evaluate the performance in terms of impact of mean query generate time T_m on overall energy consumption. The overall energy consumption of DCQOM is 12% less as compared to COCA. This is because DCQOM creates more caching space for sensed data and thereby reducing the overall communication in the network and thereby reducing the energy consumption .Figure 8 shows the impact of mean query generate time T_m on overall energy consumption.

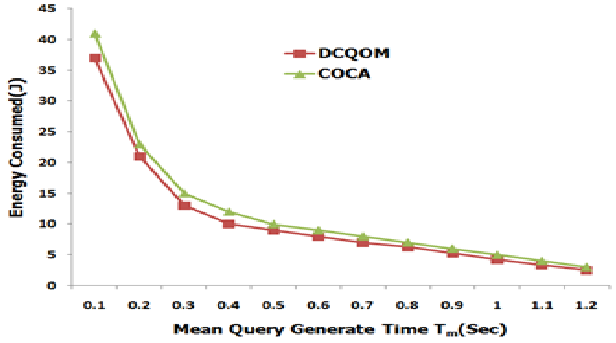

Figure 8.Effect of Mean Query Generate Time on Overall Energy Consumed.

9.2. Effects of Cache Size

In this section, we evaluate the performance in terms of effects of cache size on performance metrics. T_m is kept fixed at 0.2 seconds and cache size is varied to observe its effects on cache hits, data item replacements and overall energy consumed by the network. Figure 9 show that DCQOM have more caching locations for storing sensed data as compared to COCA. This is due to the two phase storage mechanism in DCQOM where caching is done from source to common root and from common root to different sinks. Beyond a particular cache size, cache hits do not increase because after certain time *TTL* values of many data items in cache start expiring making them stale.

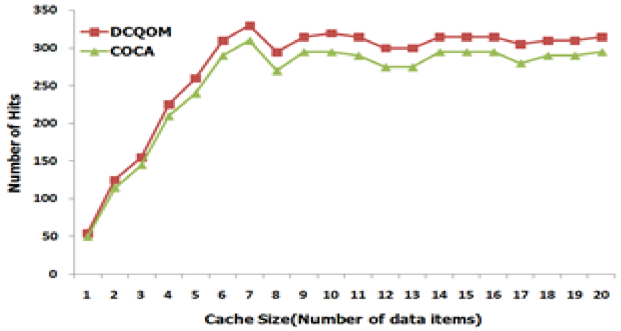

Figure 9. Effect of Cache Size on Cache Hits.

Figure 10 shows the comparison between local and CMC cache hits. From the figure, it is clear that contribution of CMC hits to total hits decreases with increasing cache size. This is due to the reason that our token based cache admission control first fills local cache and then starts filling CMC, therefore for large cache size, data items in local cache keep on becoming stale, thereby increasing the local cache hits.

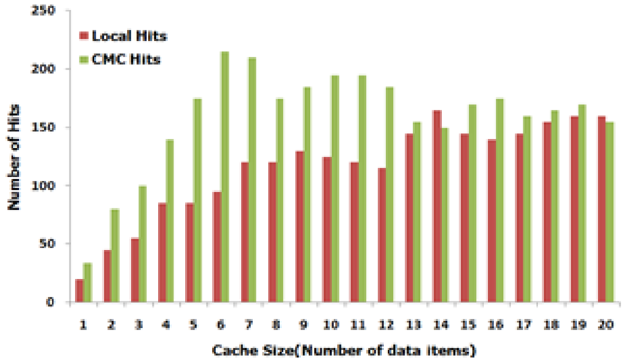

Figure 10. Effect of Cache Size on Local and CMC Hits in DCQOM.

In this section, we evaluate the performance in terms of impact of cache size on number of data item replacements both in DCQOM and COCA. Figure 11, it is clear, that number of replacements in DCQOM scheme are much lower as compared COCA .This is because in our DCQOM scheme, number of caching locations are much more as compared to COCA because of two phase caching. Since number of caching locations is more, it results in less number of replacements and the number of data items becoming stale exceeds the number of new data items to be cached.

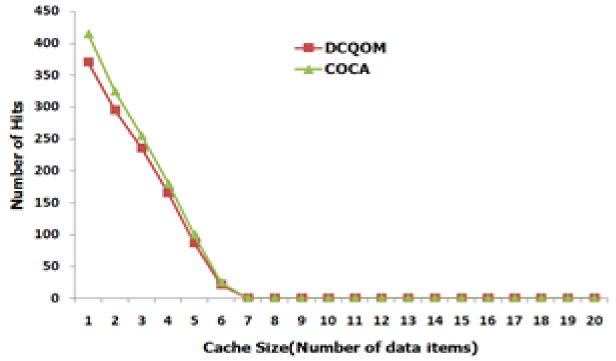

Figure 11. Effect of Cache Size on Number of Data Item Replacements.

In this section, we evaluate the performance in terms of cache size on overall energy consumption. The overall energy consumption of DCQOM is 12% less as compared to COCA. This is because DCQOM creates more caching space for sensed data and thereby reducing the overall communication in the network and thereby reducing the energy consumption .Figure 12 shows the impact of cache size on overall energy consumption.

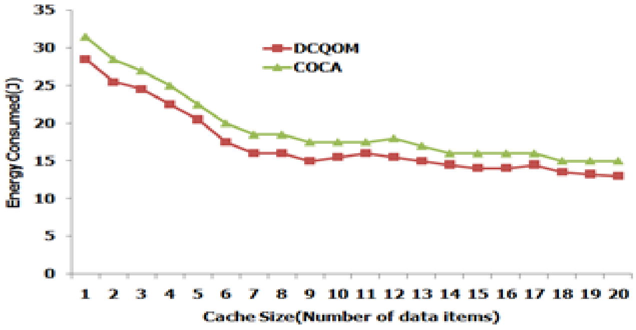

Figure 12. Effect of Cache Size on Overall Energy Consumed.

10. Conclusion and Future Work

The proposed scheme for Data Caching for query optimization in Multi Sink Sensor Networks (DCQOM) extracts the common sub tree of the networks first and then forms two caching zones: One from source node to common root and other from common root to sink nodes. The sensor nodes in the caching zones cooperate among themselves to form a larger cumulative cache. A token-based cache admission control scheme is devised, which ensures proximity of cached data closer to sink. Simulation results show that DCQOM provides substantial energy saving and data availability as compared to other caching schemes.

There are few directions for future research. First, our scheme forms two different caching zones to minimize the energy consumption .More sophisticated cost model that takes other factors, such as queuing effect and link quality, into consideration shall be further studied. Also, it is a long-term goal to design an efficient protocol that acts dynamically to any topology change in a wireless network for minimum energy consumption in sensor networks.

11. References

[1]I.F.Akyildiz, M.C.Vuran, O.Akan, W.Su: Wireless Sensor Networks: A Survey Revisited. In: Computer Networks Journal (Elsevier Science), Vol. 45, no. 3, 2004

[2]K.Akkaya, M.Younis: A Survey on Routing Protocols for Wireless Sensor Networks. In: Elsevier Ad Hoc Network Journal, Vol. 3, pp. 325- 349, 2005.

[3]S.Olariu, Q.Xu, A.Zomaya, An energy-efficient self-organization protocol for wireless sensor networks, in: Intelligent Sensors, Sensor Networks and Information Processing Conference (ISSNIP), IEEE, Melbourne, Australia, 2004, pp. 55–60.

[4]H.S.AbdelSalam, S.Olariu, “A lightweight skeleton construction algorithm for self-organizing sensor networks”, in: ICC, IEEE, 2009, pp. 1–5.

[5]C.Guestrin, P.Bodi, R.Thibau, M.Paski, and S.Madde, “Distributed Regression: An Efficient Frame Work for Modeling Sensor Network Data,” Proc. Third Int’l Symp. Information Processing in Sensor Network (IPSN), 2004.

[6]Ge-Ming Chiu, Li-Hsing Yen and Tai-Lin Chin“Optimal Storage Placement for Tree-Structured Networks with Heterogeneous Channel Costs “IEEE Transactions on computers, Vol. 60, NO. 10, October 2011.

[7]S.Madden et al, “A Tiny Aggregation Service for Ad Hoc SensorNetworks”, In OSDI Conf., 2002, pp. 131-146.

[8]Yao Y, Gehrke J., “The Cougar Approach to In-Network Query Processing in Sensor Networks”, SIGMOD Record. 2002, 31(3), pp.9-18.

[9]Mohamed Yacoab, M.Y., “Multiple Sink Based Compressive Data Aggregation Technique For Wireless Sensor Network”, International Journal of Wireless & Mobile Networks (IJWMN) Vol. 3, No. 2, pp 182-194 April 2011.

[10] Frank Yeong-Sung Lin and Yean-Fu Wen, “Multi-sink Data Aggregation Routing and Scheduling with Dynamic Radii in WSNs”, IEEE Communications Letters, Vol. 10, No. 10, October 2006.

[11]Yuh-Jzer Joung and Shih-Hsiang Huang “Tug-of-War: An Adaptive and Cost-Optimal Data Storage and Query Mechanism in Wireless Sensor Networks” Springer-Verlag Berlin Heidelberg 2008

[12]Nuggehalli,P., Srinivasan,V. and Chiasserini,C.F. (2003) ‘Energy-efficient caching strategies in ad hoc wireless networks’, MobiHoc, pp.25–34.

[13]Chand,N., Joshi,R.C. and Misra,M. (2006) ‘Cooperative caching strategy in mobile ad hoc networks based on clusters’, Springer Wireless Personal Communications, October, No. 1, pp.41–63.

[14]Nikos Dimokas, Dimitrios Katsaros, Leandros Tassiulas, Yannis Manolopoulos, “High Performance, Low Complexity Cooperative Caching for Wireless Sensor Networks”, IEEE 2009

[15]Adam Silberstein et al, “Many-to-Many Aggregation for Sensor Networks”, ICDE 2007, pp. 986– 995.

[16]Shili Xiang et al, “Multiple Query Optimization for Wireless Sensor Networks”, ICDE 2007, pp. 1339-1341.

[17]T.P.Sharma, R.C.Joshi and Manoj Misra “Cooperative caching for homogeneous wireless sensor networks” Int. J. Communication Networks and Distributed Systems, Vol. 2, No. 4, 2009.

[18]K.S.Prabh and T.F.Abdelzaher, “Energy Conserving Data Cache Placement in Sensor Networks,” ACM Transactions on Sensor Networks, Vol. 1, No. 2, pp. 178–203, 2005

[19]Rahman,M.A. and Hussain,S. (2007) ‘Effective caching in wireless sensor network’, IEEE 21st International Conference on Advanced Information Networking and Applications Workshops (AINAW’07), May, Vol. 1, pp.43–47.

[20]Narottam Chand,“Cooperative Data Caching in WSN”, International Scholarly and Scientific Research & Innovation 6(3) 2012

[21]Jinbao Li, Shouling Ji, Jinghua Zhu “Data Caching Based Queries in Multi-Sink Sensor Networks” 2009 Fifth International Conference on Mobile Ad-hoc and Sensor Networks

[22]Nikos Dimokas, Dimitrios Katsaros “Detecting Energy-Efficient Central Nodes for Cooperative Caching in Wireless Sensor Networks” 2013 IEEE 27th International Conference on Advanced Information Networking and Applications

[23]Mohammed Aalsalem, Khaled Almi'ani and Bassam Alqaralleh “A Data Caching Approach for Sensor Applications” IEEE 2012

[24]Preetha Theresa Joy, K.Poulose Jacob “A NOVEL CACHE RESOLUTION TECHNIQUE
FOR COOPERATIVE CACHING IN WIRELESS MOBILE NETWORKS” CS & IT-CSCP 2013

[25]TTDD: A Two tier Data Dissemination Model for Large scale Wireless Sensor Networks Haiyun Luo, Fan Ye, Jerry Cheng, Songwu Lu, Lixia Zhang UCLA Computer Science Department

[26]Jinbao Li, Shouling Ji et al, “Routing in Multi-Sink Sensor Networks Based on Gravitational Field”, ICESS 2008, pp. 368-375.

[27]Shouling Ji, Jinbao Li et al, “Routing in Multi-Sink Sensor Networks Based on Gravitation Field”ICESS2008.