Information Processing Letters 110 (2009) 62–70

Contents lists available at ScienceDirect

Information Processing Letters

www.elsevier.com/locate/ipl

# An analytical study of central and in-network data processing for wireless sensor networks

Mohamed Medhat Gaber[a,*], Uwe Roehm[b], Karel Herink[b]

[a] Monash University, Caulfield East, Melbourne, Victoria, 3145, Australia
[b] University of Sydney, SIT Building J12, NSW 2006, Australia

ARTICLE INFO

*Article history:*
Received 5 June 2009
Received in revised form 18 September 2009
Accepted 20 October 2009
Available online 22 October 2009
Communicated by X. Zhou

*Keywords:*
Wireless sensor network
Performance evaluation

ABSTRACT

This paper compares the performance of centralized and in-network data processing for wireless sensor networks (WSNs) under various deployment conditions on the real sensor hardware *Sun SPOT* from *Sun Microsystems*. We define several criteria to measure the quality of responses in WSN applications. Guided by an extensive experimental study, we discuss in detail the performance impacts of different deployment factors on algorithms that implement both centralized and in-network computing. Finally, performance guidelines are given to algorithm designers for WSN applications.

## 1. Introduction

It has been established that in-network processing in wireless sensor networks is the acceptable processing model due to the energy constraints of battery powered sensor nodes. However, this hypothesis does not always hold in all scenarios with different network topologies when considering other factors. In this paper, we perform an analytical comparison, based on extensive experimental study, between centralized and in-network processing setting up a set of criteria for the designer to choose between the two models of operation.

We have implemented our experimental study on the Sun SPOT sensor nodes [1] from Sun Microsystems. This gives a clear strength and significance to the produced results as opposed to simulation results. The same aggregate query runs in all the experiments for a fair comparison of the results. We have chosen to run the average (*AVG*) query. Although other experiments with different queries could also be performed, our results are able to establish the different guidelines to other processing tasks in wireless sensor networks. This is a result of the similarity of many of these tasks with the chosen query.

The paper is organized as follows. Section 2 provides the different possible scenarios of data processing for centralized and in-network processing in a WSN. The experimental setup and results are discussed in Section 3. Section 4 discusses the related work. Finally, the paper is concluded in Section 5.

## 2. Data processing alternatives in wireless sensor networks

Our research makes use of the two implementation options to develop centralized computing paradigm algorithms. The first is the *central data optimized* depicted in Fig. 1(a). This is a simple algorithm that collects all available data from the nodes and transfers it to a central off network computing facility. This leads to high response accuracy since all processing is done off the network. It also leads to complete data reusability as all the data are transferred off the network and can be stored in full resolution. The negative effects of this approach are associated with high data transfer costs.

The second option for the centralized computing paradigm is the *central result optimized* algorithm shown in

* Corresponding author.
*E-mail address:* mohamed.m.gaber@gmail.com (M.M. Gaber).

```
1. Receive request from parent node
2. FOR EACH child node
       Forward request to child
3. Retrieve local data
4. Send local data to parent node
5. FOR EACH child node
       Receive response value
       Send response value to parent node
```

(a) Central Data Optimized

```
1. Receive request from parent node
2. FOR EACH child node
       Forward request to child
3. Retrieve local data
4. Perform local data transformation
5. Store response value
6. FOR EACH child node
       Receive response value
       Store response value
7. Transform stored response values
   into a single representative value
8. Send transformed value to parent node
```

(b) Central Result Optimized

```
1. Receive request from parent node
2. FOR EACH child node
       Forward request to child
3. Retrieve local data
4. Perform local data processing
5. Store response value
6. FOR EACH child node
       Receive response values
       Store response values
7. Send all response values to parent node
```

(c) Local

```
1. Receive request from parent node
2. FOR EACH child node
       Forward request to child
3. Retrieve local data
4. Perform local data processing
5. Store response value
6. FOR EACH child node
       Receive response value
       Store response value
7. Aggregate stored response values into
   a single representative value
8. Send aggregate value to parent node
```

(d) Local Optimized

**Fig. 1.** Algorithms.

Fig. 1(b). This algorithm applies transformations to in-network data to minimize the data transfer overhead, thereby remedying the main shortfall of the central data optimized algorithm. The data are then transferred to a location off the network. The computations that produce query results are performed off network on a central server. The benefits of using this algorithm are high response accuracy, since all processing is done off the network, as well as low data transfer overhead. The largest pitfall of this algorithm is that data transformations are performed in the network; therefore they are limited by the computational capabilities of the nodes. In addition the transformed data delivered to the central location may not be further reusable; this depends purely on the type of transformation. This algorithm also requires cache/storage on each local node.

Similarly, we have made use of the two available choices to develop the algorithms for the in-network computing paradigm. The *local* algorithm depicted in Fig. 1(c) represents the first one. This is a simple algorithm that computes a local result for each node that participates in a query. The result is appended to a string of all results gathered from children nodes and forwarded to the parent node. The amount of data overhead is directly proportional to the size of the network. However, since only local results are transferred the overall data transfer overhead tends to remain much lower (depending on the size of data cache) than that of the central data optimized algorithm. This algorithm has the potential to introduce computational error attributed to limited computing resources available onboard the sensor node. It also offers a limited amount of data reusability as local results from each participating node are passed to the base-station for further processing. Storage is also needed, and it must be large enough for the subtree results; not to mention that it typically will have to be written to flash memory.

The second algorithm is the *local optimized* shown in Fig. 1(d). This is also referred to as *in-network aggregation* in the WSN literature. This algorithm is similar to the local algorithm, but additionally makes use of in-network data aggregation. Each node that participates in a query undergoes the following steps. It computes a local result, and then collects aggregate results from all of its direct children. Finally, it computes a single aggregate value that represents its own local result as well results received from its children. This single aggregate value is then forwarded to the parent node. This algorithm minimizes the data transfer overhead by passing only a single value between nodes at a time regardless of the network size. This algorithm has larger potential for computational error than the local algorithm. This is due to the recursive aggregation behavior that causes computational error to increase with every additional iteration. Even with simple aggregate functions like *COUNT* and *SUM*, overflows can be encountered, especially is large networks.

## 3. Experimental study

All experiments were performed on a wireless sensor network built using the Sun SPOT wireless sensor nodes modules. The base-station is a node connected to an iBook G4 apple notebook via a USB cable. All the source code implementing the in-network techniques as well as base-station functionality is written in Java programming language using a combination of Java SE 5 on the desktop and J2ME on the nodes. There are several key environmental factors that affect the behavior of a WSN as a whole. These are:

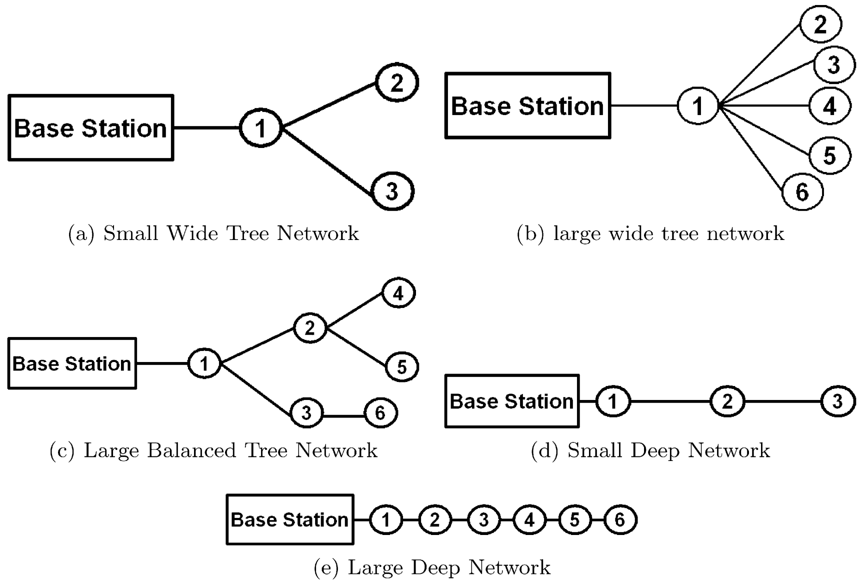

Base Station
(a) Small Wide Tree Network
(b) large wide tree network
(c) Large Balanced Tree Network
(d) Small Deep Network
(e) Large Deep Network

**Fig. 2.** Network topology.

1. Network topology.
2. Amount of data that each sensor can hold (cache).
3. Domain of data collected by sensors (dataset).

For our experiments, we have used five different topologies shown in Fig. 2. Three levels of cache sizes have been used as follows: 5 (small), 25 (medium) and 125 (large) elements. It has to be noted that the cache size refers to the memory buffer where data are stored, not the hardware cache. Finally, we have used three ranges for the domain of the data sets as follows: (1) Small value set in the range between 0 and 3, (2) medium value set in the range between 0 and 1100, and (3) large value set in the range between 10 000 and 11 100. We have used the following quality criteria to assess the performance of the processing algorithm:

1. Number of bytes transferred.
2. Number of messages transferred.
3. Response Time (*RT*): the time between query dissemination and the time when the first message containing (possibly partial) response arrives to the base-station.
4. Execution Time (*ET*): the time between query dissemination and the time when the last message containing response arrives to the base station.
5. Response Percentage Error (*RPE*): the amount of error in the response.
6. Data reusability.

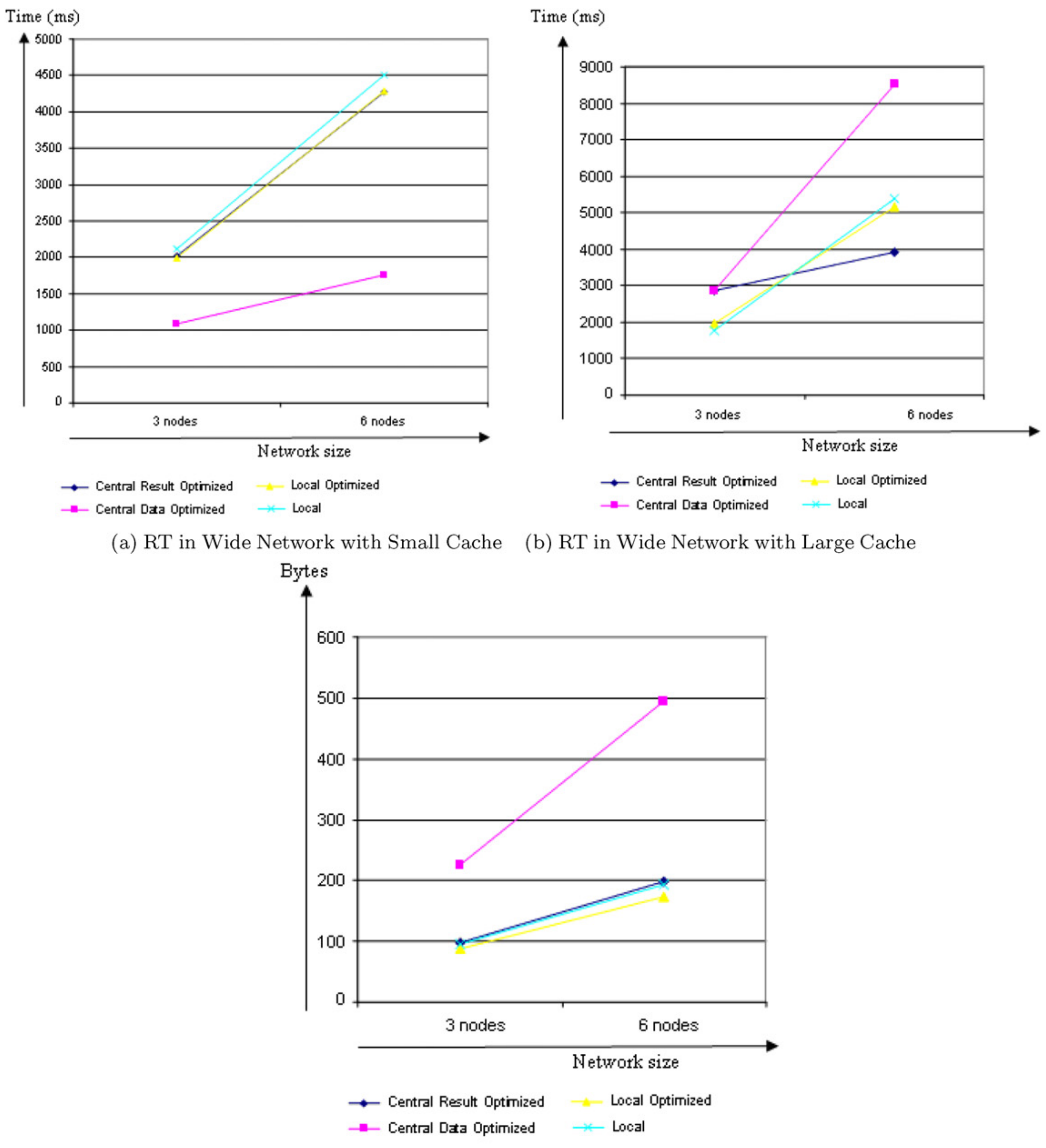
### 3.1. *Impact of network size*

Figs. 3, 4, and 5 show the different effects of the network size on the different performance criteria. The experiments were repeated with different sizes of cache memory. In Fig. 3(a), we compare the quality of responses of the four algorithms in terms of response time using small cache size and small values in a wide network. We observe that the central data optimized algorithm is the least affected by growing the network size. This is explained by the fact that the central data optimized algorithm is implemented without any additional data processing. Therefore responses are sent to the base-station as soon as they become available without any delay. In contrast, all other algorithms do additional processing that causes nodes to wait for replies from their children before they send a response. This effectively makes the response time equal to the execution time for all algorithms other than the central data optimized algorithm. It is important to note that with increasing the cache size, the response time of the central data optimized algorithm will degrade due to high network resource contention as shown Fig. 3(b).

In Fig. 3(c), we illustrate the effects of network size on the number of bytes transferred in the network during query execution. The graph follows expected nearly linear growth for all algorithms other than central data optimized algorithm. In the case of the central data optimized algorithm, doubling the number of nodes results in 2.2 times increase of the size of transferred data. This trend holds for all cache sizes and the very same trend is also reflected in the number of messages that are passed through the network during query execution.

Figs. 4(a) and 4(b) show that the impact of network size on execution time is similar in both balanced and wide networks. It can also be observed that the balanced network topology yields better results than wide network topology in terms of execution times for all network sizes and algorithms. This is caused by the throughput bottleneck at the root node of the wide network. We again see that cache size strongly degrades the performance of the central data optimized algorithm.

Fig. 5(a) shows that the effects of the network size on the response time using small caches. While the response time of central data optimized algorithm is largely unaf-



(a) RT in Wide Network with Small Cache (b) RT in Wide Network with Large Cache

(c) Number of Bytes transferred in wide network with small cache

**Fig. 3.** Impact of network size on wide network.

fected by increasing the network size, response times of the three remaining algorithms increase. Fig. 5(b) shows the results of the same experiment with lager cache size. Although the central data optimized algorithm is required to transfer much larger quantity of data, it still remains the best choice for its minimal response time. The same does not hold true for balanced or wide topology. This property of the deep topology is explained as follows. At the time the first node receives a query request from the base-station, it has relatively large amount of time (depending on the network depth) to transmit its data back to the base station before the request propagates all the way through the network, and then responses are transmitted back through the network to the base station.

We now examine the effects of network size on the execution time. In case of the deep network topology, the performance of the four algorithms is comparable in small networks using small cache size. When the size of the network increases, the local optimized algorithm becomes the best performing one. On the other hand, the central data optimized algorithm comes the last. However, the performance of all the algorithms is in reasonably close proximity with the slowest time 36% behind the fastest time

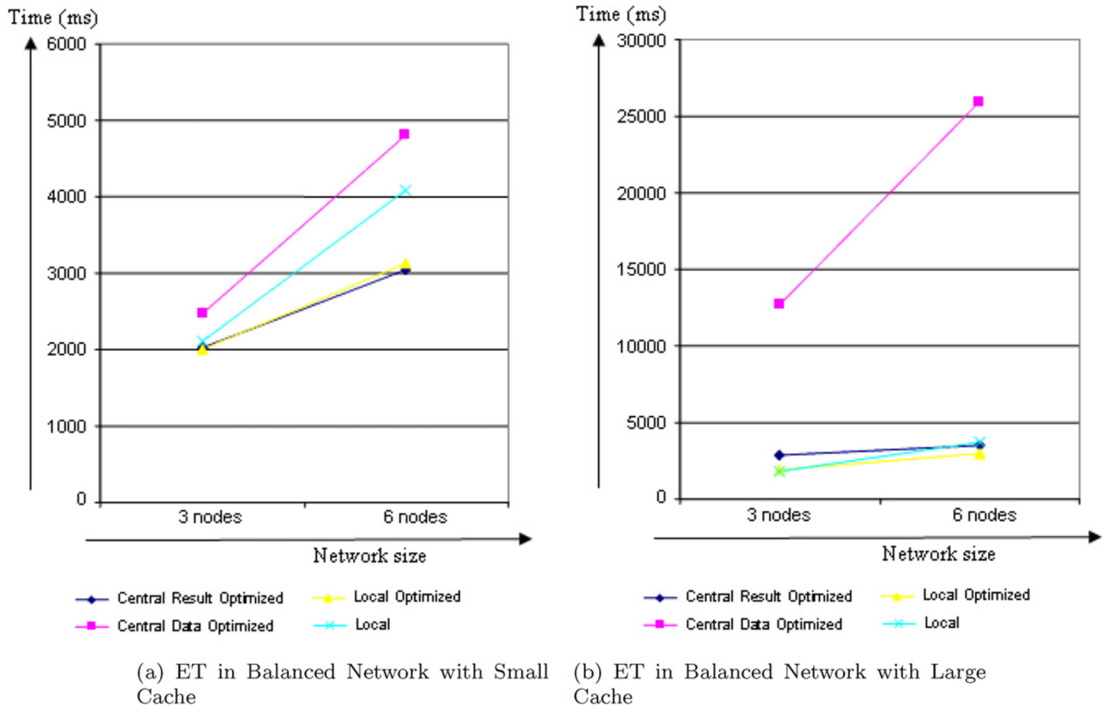


**Fig. 4.** Impact of network size on balanced network.

as shown in Fig. 5(c). The performance gap widens considerably once the cache size is increased as illustrated in Fig. 5(d). This is a common trend observed in all topologies. The central data optimized algorithm performs poorly in terms of execution time with large caches, and this effect is further magnified with increasing the network size.

### 3.2. *Impact of dataset domain*

The domain of the dataset mainly impacts the response percentage error. We have run a set of experiments with setting the cache memory to its small size setting. Fig. 6 shows the results of these experiments.

We can observe from Fig. 6(a) that the amount of percentage error with small and medium values varies among different topologies in a random fashion with values between 6% and 21% approximately. In the case of large values, we can see that response error values in the range between 0.38% and 0.99%. It is important to note that the potential for error in the case of large values is limited by the value range of 10 000–11 100, i.e. maximum possible error value of 9.9%, in which case the smallest error using large values of 0.38% is equivalent to 3.8% error when using medium values. This suggests that the response error is not affected by the dataset domain, but rather by the potential for error and value distribution within the network.

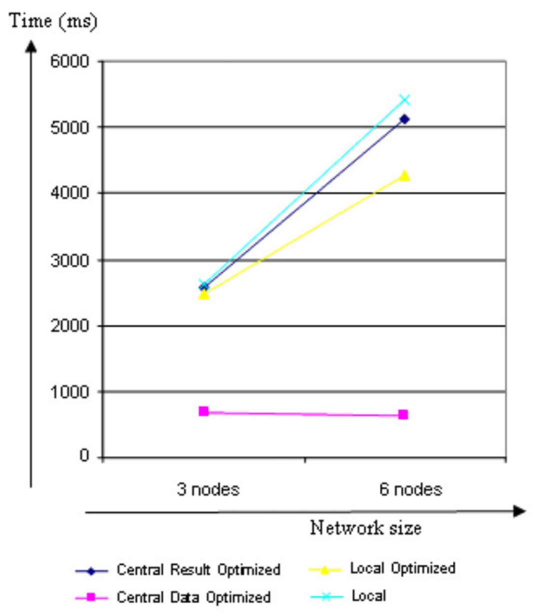
In Fig. 6(b), we illustrate the effects of dataset domain on the local optimized algorithm in various network topologies. We observe that the small value dataset with value in range from 0 to 3 produces large execution percentage error, while the medium value dataset with value range from 0 to 1100 produces much smaller execution percentage error. This effect is caused by rounding error which is lager in operations on smaller data values.

In Fig. 6(c), we project the effects of dataset domain on the local algorithm in various network topologies. We observe much the same effects as in Fig. 6(b), i.e., that the small value dataset with value in the range from 0 to 3 produces large execution percentage error, while the medium value dataset with value range from 0 to 1100 produces much smaller execution percentage error. It could be also observed that the percentage error produced by the local algorithm is smaller than or equal to the error produced by the local optimized algorithm. This holds true across all network topologies and data sets.

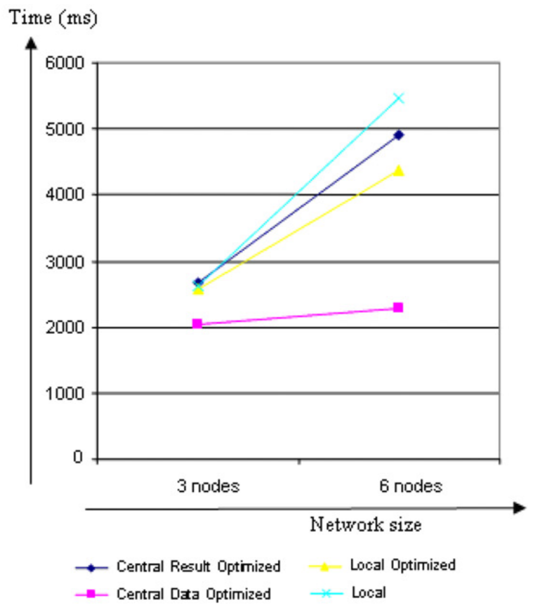
### 3.3. *Impact of cache size*

Fig. 7 shows the impact of varying the cache size on the response time, the execution time and the amount of data transferred in all the different topologies.

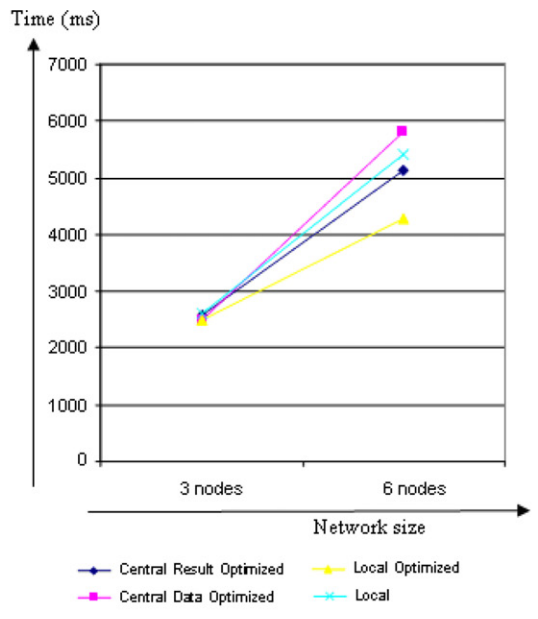
In Fig. 7(a), we illustrate the impact of cache size on the response time in various topologies with small values using central data optimized algorithm. The graph conveys several important points. Firstly, we observe that even though the cache size grows exponentially, the growth of response time for all implemented algorithms is much slower. This is attributed to the constant network packet size of 96 bytes as well as to the implementation of random time-out periods during network collisions. Secondly, we observe that the large wide topology is the most neg-



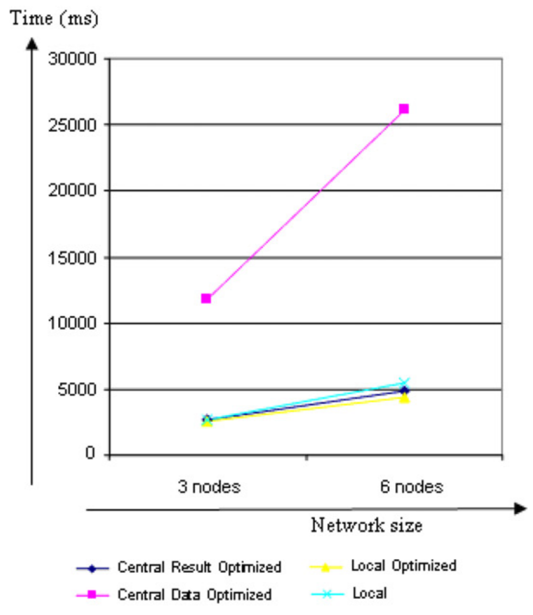
(a) RT in Deep Network with Small Cache



(b) RT in Deep Network with Large Cache



(c) ET in Deep Network with Small Cache

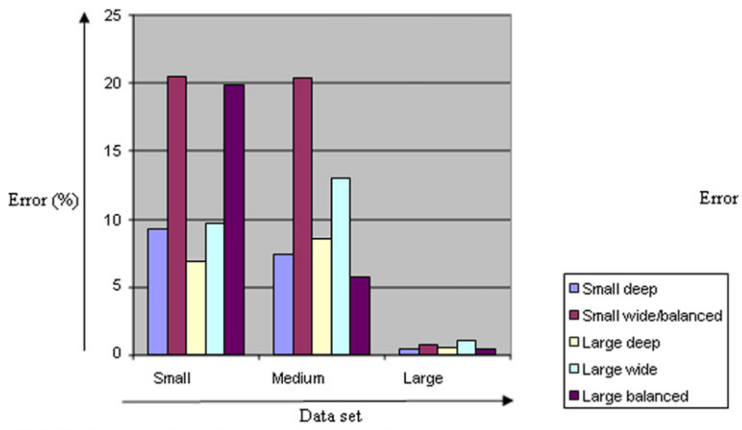


(d) ET in Deep Network with Large Cache

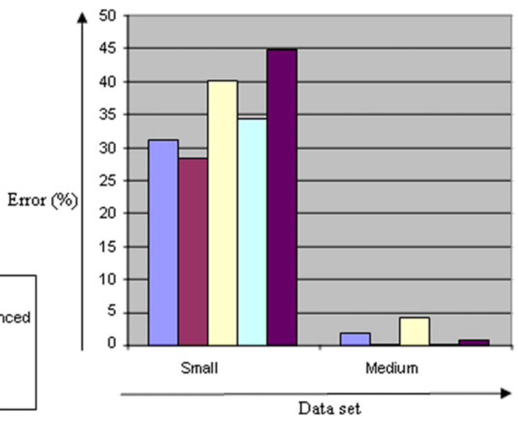
**Fig. 5.** Impact of network size on deep network.

atively affected by increasing the cache size. This is attributed to the very high network resource contention rates in large wide topologies. As the number of child nodes attempting to send to a single parent node increases, the amount of time required to complete the data transfer increases. A point of interest is that "wide" is the only topology that copes badly with increasing the network size. On the other hand, the network size has little effect upon the response times in all the other topologies.

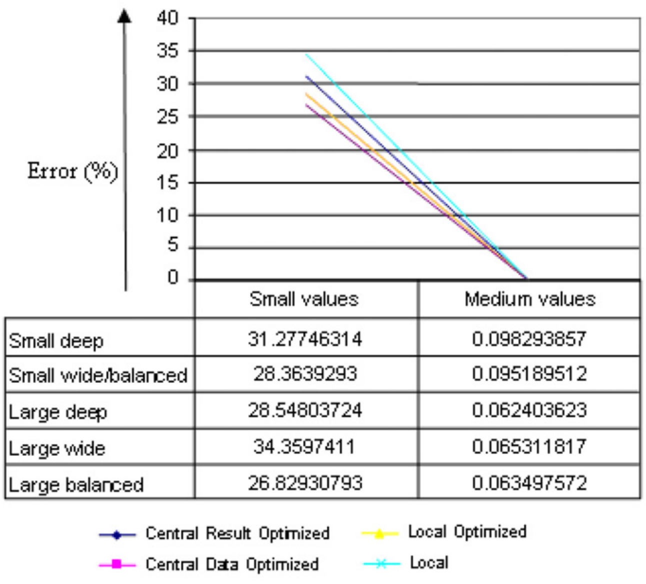
Fig. 7(b) demonstrates the impact of cache size on the execution time in various topologies with small values using central data optimized algorithm. A noticeable difference between Figs. 7(b) and 7(a) is that the network size plays an important role in result quality in terms of execution time. This is predictable, because execution time measures the entire period between the time of request dissemination and the time of arrival of the final packet of response data to the base-station. It is also clear that re-



(a) RPE with Small Cache using Central Data Optimized Algorithm



(b) RPE with Small Cache using Local Optimized Algorithm



| | Small values | Medium values |
|---|---|---|
| Small deep | 31.27746314 | 0.098293857 |
| Small wide/balanced | 28.3639293 | 0.095189512 |
| Large deep | 28.54803724 | 0.062403623 |
| Large wide | 34.3597411 | 0.065311817 |
| Large balanced | 26.82930793 | 0.063497572 |

(c) RPE with Small Cache using Local Algorithm

**Fig. 6.** Impact of dataset domain.

sponse quality suffers noticeably as cache size increases. For example, in the case of large balanced topology, as the cache size increases from 5 elements to 25 elements, the execution time in turn increases from 4801 ms to 25 961 ms, which is a substantial increase.

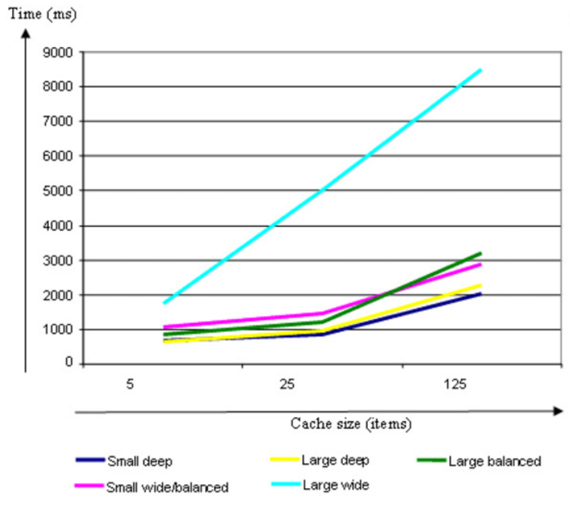
In Fig. 7(c), we observe that the cache size has very direct impact on the number of bytes transferred. Due to the fact that the number of messages has a linear direct relationship with the number of bytes transferred, we observe the same trend when considering the number of messages transferred. While both values are very closely correlated, it is important to consider them separately as each measurement has its own significance in terms of power usage. While we are unable to measure power usage directly, it can be derived from the amount of power demanding operations that a sensor node undertakes during query execution. Depending on the platform and networking implementation, either number of messages (in case of datagram connectivity), or number of bytes (in case of stream connectivity) can be the determining measure of power use for network communication.

### 3.4. *Performance guidelines*

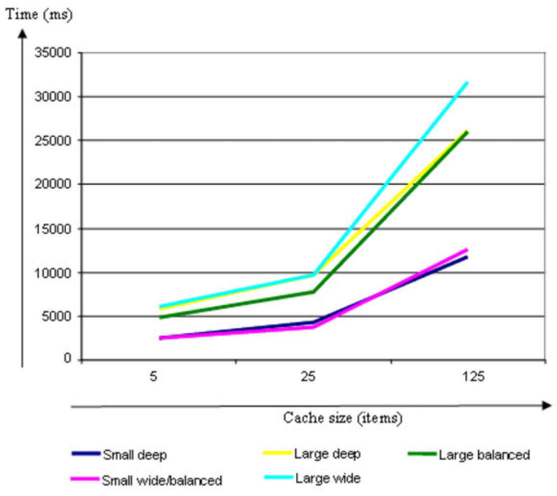
Having examined the previously discussed results, we have been able to come up with a set of performance guidelines detailed in Table 1.

## 4. Related work

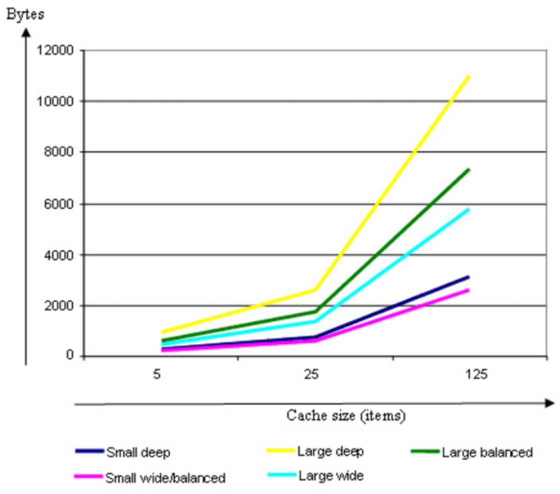
To the best of our knowledge, a detailed analytical comparison between centralized and in-network data processing in wireless sensor networks has not been con-



(a) RT with Small Values using Central Data Optimized Algorithm



(b) ET with Small Values using Central Data Optimized Algorithm



(c) Data Transferred in Bytes with Small Values using Central Data Optimized Algorithm

**Fig. 7.** Impact of cache size.

ducted. Thus, we can identify the following indirect related work.

The hardware and software implementation of our experimental platform is described in detail in [1]. Although our research makes extensive usage of the data-centric localized algorithms, the idea is the main focus of [3,5,6]. As our experiments used data aggregation, this processing task has been the main focus of [7] and [2].

Query processing in wireless sensor networks has been addressed in developing four main system prototypes, namely TinyDB [8], Cougar [10], SSDQP [9] and Nile [4]. However, these systems have not addressed the trade-off between centralized and in-network query processing.

## 5. Conclusion and future work

Our research clearly outlines the key differences between in-network and centralized processing using a concrete implementation of both computing paradigms on real hardware as opposed to simulation results. Our experimental results show that centralized computing is a viable alternative to in-network computing under various circumstances especially when the result accuracy, the response time, and the data reusability are of primary concern.

The research presented in this paper offers a variety of future directions. One way to extend our research is by improving the currently implemented algorithms by compressing or approximating the data, and then analyzing the impact they have on the quality of the produced results.

**Table 1**
Performance guidelines.

| Quality requirement | Optimal algorithm | Conditions |
|---|---|---|
| Execution accuracy | Central result optimized | None |
| | Central data optimized | None |
| Response accuracy | Central result optimized | None |
| Response time | Central data optimized | Topology: Small wide/balanced, Large balanced |
| | | Cache size: Small, Medium |
| | | Topology: Large wide |
| | | Cache size: Small |
| | | Topology: Small deep, Large deep |
| | | Cache size: All |
| | Local optimized | Topology: Large wide |
| | | Cache size: Medium |
| | Local optimized | Topology: Large wide |
| | | Cache size: Large |
| | Local optimized | Topology: Small wide/balanced |
| | | Cache size: Large |
| | Local optimized | Topology: Large balanced |
| | | Cache size: Large |
| Execution time | Local optimized | None |
| Bytes transferred | Local optimized | None |
| Messages transferred | Local | All encapsulated results can be sent in a single message |
| | Local optimized | None |
| | Central result optimized | None |
| Data reusability | Central data optimized | High data resolution required |
| | Local | Low data resolution required |

## References

[1] C. Cifuentes, D. Cleal, J. Daniels, D. Simon, D. White, Java(TM) on the bare metal of wireless sensor devices – The Squawk Java virtual machine, in: Proceedings of the 2nd International Conference on Virtual Execution Environments, VEE, Canada, 2006, pp. 78–88.

[2] A. Deligiannakis, Y. Kotidis, N. Roussopoulos, Hierarchical in-network data aggregation with quality guarantees, in: Proceedings of EDBT, 2004, pp. 658–675.

[3] D. Estrin, R. Govindan, J. Heidemann, S. Kumar, Next century challenges: Scalable coordination in sensor networks, in: Proceedings of the 5th Annual ACM/IEEE International Conference on Mobile Computing and Networking, Seattle, Washington, United States, ACM Press, 1999, pp. 263–270.

[4] M.A. Hammad, T.M. Ghanem, W.G. Aref, A.K. Elmagarmid, M.F. Mokbel, Data stream management systems and architectures, in: J. Gama, M.M. Gaber (Eds.), Data Stream Processing Techniques in Sensor Networks, Springer, 2007.

[5] C. Intanagonwiwat, R. Govindan, D. Estrin, Directed diffusion: A scalable and robust communication paradigm for sensor networks, in: Proceedings of the 6th Annual International Conference on Mobile Computing and Networking, Boston, Massachusetts, United States, ACM Press, August, 2000, pp. 56–67.

[6] C. Intanagonwiwat, R. Govindan, D. Estrin, J. Heidemann, F. Silva, Directed diffusion for wireless sensor networking, IEEE/ACM Transactions on Networking (TON) 11 (1) (2003) 2–16.

[7] S. Madden, M.J. Franklin, J.M. Hellerstein, W. Hong, TAG: A tiny aggregation service for ad hoc sensor networks, in: Proceedings of the 5th Symposium on Operating Systems Design and Implementation (OSDI), Boston, Massachusetts, United States, ACM Press, 2002, pp. 131–146.

[8] S. Madden, M.J. Franklin, J.M. Hellerstein, W. Hong, TinyDB: An acquisitional query processing system for sensor networks, ACM TODS (2005).

[9] B. Scholz, M.M. Gaber, T. Dawborn, R. Khoury, E. Tse, Efficient time triggered query processing in wireless sensor networks, in: Yann-Hang Lee, Heung-Nam Kim, Jong Kim, Yongwan Park, Laurence Tianruo Yang, Sung Won Kim (Eds.), Embedded Software and Systems, Third International Conference, ICESS 2007, Daegu, Korea, May 14–16, 2007, in: Lecture Notes in Computer Science, vol. 4523, Springer, ISBN 978-3-540-72684-5, 2007, pp. 391–402.

[10] Y. Yao, J. Gehrke, The cougar approach to in-network query processing in sensor networks, SIGMOD Record 31 (3) (September 2002) 9–18.