Distributed Computing Paradigms for Collaborative Processing in Sensor Networks

Yingyue Xu, Hairong Qi, Phani Teja Kuruganti
Department of Electrical and Computer Engineering
The University of Tennessee, Knoxville, TN 37996-2100
{yxu4, hqi, teja}@utk.edu

***Abstract*— In sensor networks, collaborative processing between multiple sensor nodes is essential in order to complement for each other's sensing capability, tolerate faults, and provide reliable information. The client/server-based paradigm is typical for distributed processing. However, it is not the most efficient in the context of sensor networks. In this paper, we present a mobile-agent-based paradigm to carry out collaborative processing, where instead of each sensor node sending local information to a processing center, as is typical in the client/server-based computing, the processing code is moved to the sensor nodes through mobile agents. This approach has great potential in providing energy-efficient and scalable collaborative processing with low latency. We design two metrics (*execution time* and *energy consumption*) and use simulation tools to quantitatively measure the performance of different computing models in collaborative processing. Experimental results show that the mobile agent paradigm performs much better when the number of nodes is large while the client/server paradigm is advantageous when the number of nodes is small. Based on this result, we develop a cluster-based hybrid computing paradigm to combine the advantages of both paradigms. We analyze two different scenarios in hybrid computing and simulation results show that there is always one scenario that performs better than either the client/server- or mobile-agent-based paradigm.**

I. Introduction

The advances in Micro Electro Mechanical System (MEMS) and wireless communication have enabled the development of low-cost, low-power, untethered and tiny sensor nodes, equipped with significant processing, memory, and wireless communication capabilities. Large number of such sensor devices can be densely and randomly deployed to form a new kind of network - the sensor network.

A lot of challenging problems have been presented by the sensor network. We summarize some important issues here: (1) Energy. Sensor networks are energy constrained because sensor devices are battery operated. Once deployed, it is usually impossible to replace the battery. So, how to save energy, how to prolong the lifetime of individual sensor node and the whole sensor network is a major challenge in sensor networks. (2) Scalability. Huge amount of sensor nodes could be deployed in a sensor network. Therefore, the many protocols proposed for ad hoc networks may not be suitable for sensor networks. (3) Reliability. Sensor nodes are often deployed in harsh, disastrous or inaccessible environment. Such environment makes sensor nodes prone to failures. Furthermore, the wireless transmission in sensor networks has high bit error rate (BER) and low bandwidth. In such dynamic networks, fault-tolerance and reliability is another challenge. (4) Realtime response. Although performance is not the first priority, in some applications, emergent information has to be provided in time. So how to reduce latency in presence of limited energy is also worth studying. This paper presents mobile-agent-based approaches to performing collaborative information processing, which respond to the challenges posed by the sensor network.

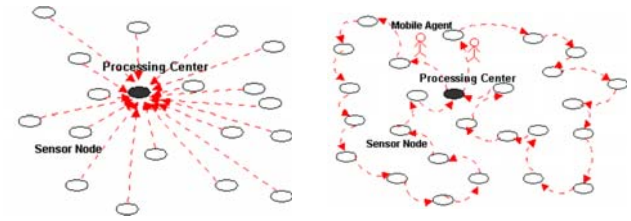
II. Mobile-agent-based Computing Paradigm

A sensor network forms a typical distributed environment and the client/server paradigm has been one of the most popular models adopted in distributed computing [1]. In this paradigm, a server offers a set of services, resources, and know-how needed for service execution. The client requests the execution of a service. As a response, server performs the requested service by executing the corresponding service know-how and accessing the involved resources at the server. In sensor networks, the processing center is the server and the sensor nodes are the clients. Figure 1 (a) illustrates the client/server-based paradigm. Although widely used, the disadvantages of this model are also dramatic, especially for sensor networks [2], [3]. First of all, there should be some super-nodes acting as processing centers, which need much higher energy, storage and computing capabilities. These will largely reduce the lifetime of the whole sensor network, especially in autonomic and homogeneous sensor networks. Secondly, in some particular applications, such as data processing or data fusion, large amounts of data are moved from the clients to the server. In a system with many clients, the total bandwidth requirements may exceed available bandwidth, resulting in poor performance of the system.

Because the number of sensor nodes is usually large, using the client/server-based computing will create huge network traffic. The mobile-agent-based computing [2], on the other hand, transfers the partially integrated results and executable code from one node to another and the processing can be done locally on the sensor nodes.

In the mobile-agent-based paradigm, the service know-how is owned by the server, but most of the resources are located at the clients. The server sends out mobile agents carrying service know-how. The mobile agents complete the service using resources available at the clients. In sensor networks, the processing center sends out the mobile agents who will migrate

to each node and conduct data processing until they return to the processing center with results. Figure 1 (b) illustrates the mobile-agent-based paradigm.

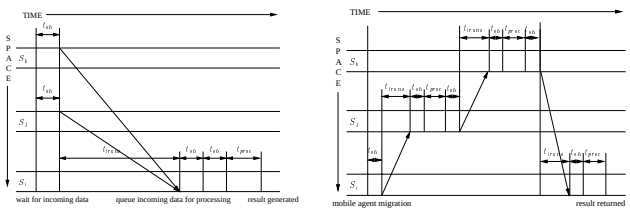


(a) client/server based (b) mobile-agent based

Fig. 1. Different computing paradigms.

We define mobile agent as an entity of four attributes: identification, itinerary, data space, and service know-how. Identification uniquely identifies each mobile agent. Data space is the agent's data buffer which carries a partially integrated result. Itinerary is the route of migration. It can be fixed or dynamically determined based on the current network status. Service know-how is the processing task (or execution code) carried with the agent. In order to better illustrate how these two computing models perform, Fig. 2 presents a temporal and spatial comparison of the life cycle of their migration units. In the client/server-based model, the migration unit is "data", while in the mobile-agent-based model, the migration unit is "mobile agent". The life cycle of both units is composed of three components: t_{trans}, t_{oh}, and t_{proc}. We use t_{trans} to represent the time spent in transferring the migration unit from one node to the other. t_{oh} represents the overhead time: In the case of client/server-based model, it is the time spent on file access, here we assume the data a node collects is large and has to be stored in a file; and in the case of mobile-agent-based model, it is the time used to create, dispatch, and receive the mobile agent. Finally, t_{proc} represents the processing time.

In the client/server-based model (Fig. 2 (a)), the clients (S_j and S_k) first read the data files into memory using t_{oh}, then transfer them to the server (S_i) using t_{trans}. Here, we assume S_j and S_k are identical nodes and can start data transfer at the same time. Therefore, these data files might arrive at the server simultaneously but can only be processed serially. It takes S_i t_{oh} amount of time to receive each data file and open it. After receiving all the incoming data files, the server can start processing, which would take t_{proc} amount of time.



(a) Data migration. (b) Mobile agent migration.

Fig. 2. Life cycle of data (client/server-based) and mobile agent (mobile-agent-based) migration in time and space.

In the mobile-agent-based model (Fig. 2 (b)), a mobile agent is created at node S_i using t_{oh}. It then migrates to S_j using t_{trans}. S_j spends t_{oh} to parse the mobile agent and read the data file on the node and t_{proc} to do information processing. After t_{oh} amount of time, the agent is created at node S_j and migrates to S_k. After the same procedure, it finishes migration and returns to S_i.

The mobile agent model has many advantages over the client/server model and is very suitable for distributed environment like sensor networks: (1) Network bandwidth requirement is reduced. Instead of passing large amounts of raw data over the network through several round trips, only the agent with small size is sent. This is especially important for real-time applications and where the communication is through low-bandwidth wireless connections. (2) Energy. Since the total amount of data transmission is reduced, the energy usage can also be reduced as radio transmission is the most energy consuming activity in sensor networks [4]. Moreover, mobile agent can terminate migration and return when the accuracy of the results satisfy the requirement, thus further reduce energy usage and execution time since unnecessary data transfers are avoided. (3) Scalability. For the client/server-based computing, there will be increased queuing length as the number of clients increases. As a result, it may cause longer processing time and more possible drops at the server side. Unfortunately, in sensor networks, the number of nodes may be hundreds or even thousands. On the other hand, the mobile-agent-based computing may not be affected as the number of nodes increase. (4) Reliability. When a node is down, mobile agent can bypass that node and migrate to the next available node.

III. Performance Evaluation

Even though the mobile-agent-based computing is promising and seems more appropriate for sensor networks, there has not been any simulation work done to actually evaluate its performance in collaborative processing in sensor networks. The mobile-agent-based computing may not always perform better since mobile agents also introduce overhead, which mainly comes from the agent creation and dispatch time. On the other hand, the client/server-based computing needs to transfer data files to the processing center, which also causes overhead due to file accesses. Therefore, a performance evaluation of these two computing paradigms is essential.

In order to simplify the analytical model and simulation, as well as to make the two computing models comparable, we make the following assumptions of the sensor network: (1) When certain event occurs in an area, all sensor nodes can detect it and collect the raw data so that there are always the same amount of sensor nodes participating in the collaborative information processing; (2) There is no event occurring simultaneously in the area; and (3) Sensor nodes are not mobile.

We design two metrics for the performance evaluation: execution time and energy consumption.

A. Execution time metric

The *execution time* is the time spent to finish an information processing task. In the mobile-agent-based paradigm, it starts from the time a mobile agent is created to the time the mobile agent returns with results. In the client/server-based paradigm, it is from the time the clients send out requests to the time the server generates results. Execution time consists of three components, data transfer time (t_{trans}), overhead time (t_{oh}), and data process time (t_{proc}). Since in both computing models, we have exactly the same amount of data to process and the only difference is where data processing takes place, t_{proc} is the same for both paradigms. Hence, we exclude this component from our model. The other components of the execution time depend on the network transfer rate v_n, the data file size s_f, the mobile agent size s_a, the overhead of file access o_f, the overhead of agent o_a, the number of sensor nodes p, and the balance between the number of agents m and the number of sensor nodes n that each agent migrates (the server is not included). We assume each agent migrates the same number of nodes. Notice that $p = m \times n$.

Thus, for the client/server paradigm, the total execution time is

$$t_{cs} = t_{tran-cs} + 2mno_f \tag{1}$$

where the overhead time is $t_{oh} = 2mno_f$ (assuming the time used to read and write the data file is the same) and $t_{tran-cs}$ is the data transfer time which will be obtained from simulation.

For the mobile agent paradigm, the total execution time is

$$t_{ma} = t_{trans-ma} + 2(m+n)o_a \tag{2}$$

where the agent overhead time is $t_{oh} = 2(m+n)o_a$, since it takes $2mo_a$ for the dispatcher to send and receive m mobile agents, and $2no_a$ for the nodes to send and receive each mobile agent. $t_{trans-ma}$ is the agent transfer time and is obtained from simulation as well.

B. Energy consumption metric

Sensor nodes rely on the power supplied to the node by a battery which in reality has finite power. Minimization of energy consumption is important for extended battery lifetime and subsequently the sensor network lifetime [5]. Thus comparing the energy consumed by the sensor node in different paradigms proves to be an important performance metric. Similar to the formulation of the execution time, the energy consumed can be divided into three different parts, energy required for processing the data (e_{proc}), for processing the overhead (e_{oh}), and for transfer of data (e_{trans}).

Energy consumed in processing can be obtained either at the physical measurement or from the simulation-based methods at the instruction level. However, [5] shows that the power consumption of a processor can be predicted within 8% accuracy and 99% confidence, as the product of processor average power and the software execution time and there is no need to consider the individual assembly instructions to accurately predict power and energy consumption.

For the same reason that we excluded t_{proc} from the execution time, here, we exclude e_{proc} from the energy consumption model as no matter where the data processing takes place, it consumes the same power in terms of the whole sensor network.

The overhead energy for one transmission can be computed by $2 \times p_{processor} \times t_{oh}$, where $p_{processor}$ is the power consumption of the node processor in full load and the multiplier 2 considers both dispatching and receiving that use the same amount of energy in the overhead processing. We chose ARM Thumb processor at 40MHz with $p_{processor} = 75mW$ [4] for simulation purposes. The number of data migrations in client/server-based computing is mn while the number of mobile agent migrations in mobile-agent-based computing is $m(n+1)$. Therefore, the model used to calculate energy consumption is

$$e_{cs} = e_{trans-cs} + 2mnp_{processor}o_f \tag{3}$$

$$e_{ma} = e_{trans-ma} + 2m(n+1)p_{processor}o_a \tag{4}$$

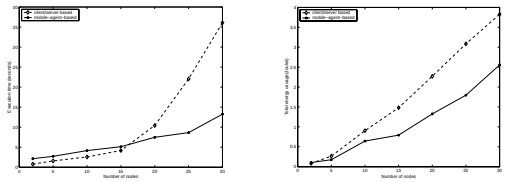
where $e_{trans-cs}$ and $e_{trans-ma}$ are the transfer energies for the client/server and mobile-agent-based paradigms and will be obtained from simulation.

IV. Experiments and Simulation Results

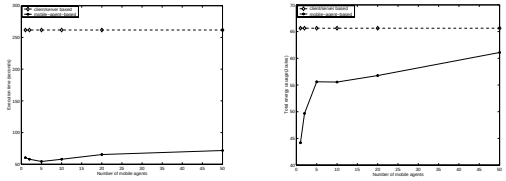
We set up a basic network and the related parameters are as follows: $n = 10, m = 1, s_f/s_a = 10:1, s_a = 1$Kb, $o_f/o_a = 1:4, o_a = 0.05$s. We use Network Simulator 2 [6] to simulate the basic network. In our simulation, nodes are randomly deployed in a 10m $\times$ 10m area. IEEE 802.11 is the MAC layer protocol. The radio transmission power is 0.6W and receiving power is 0.3W. Random motion is disabled. We design four experiments to test the effect of different parameters on the two metrics. In each experiment, we change one of the parameters of the basic network and keep all the others unchanged.

A. The effect of the number of nodes (p)

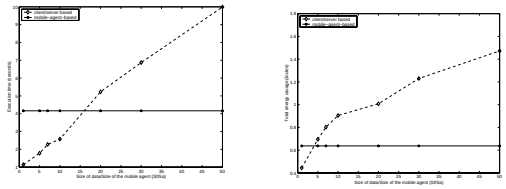
In this experiment, we change the number of nodes p from 2 to 30. The execution time measurement is shown in Fig. 3 (a)(left). It can be observed that the execution time using both paradigms grows as the number of nodes increases. But the client/server model grows much faster than the mobile agent model. This is because as the number of nodes increases, the server has to deal with more connections requested by the clients at the same time which elongates the execution time. On the other hand, the mobile agent model is less influenced by the number of nodes because there are much less connections at one time for the mobile agent model. The figure also shows that, for $p < 17$, the client/server model performs a little better than the mobile agent model. This happens because the mobile agent model needs more connections than the client/server model in order to send and receive mobile agents. Another reason is that the overhead of the mobile agent



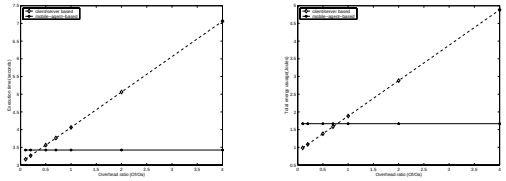
(a) The effect of the number of nodes (p).



(b) The effect of the number of mobile agents (m).



(c) The effect of data size/mobile agent size (s_f/s_a).



(d) The effect of the overhead ratio (o_f/o_a).

Fig. 3. Performance analysis of the effect of different parameters on the execution time (left column) and energy consumption (right column).

surpasses the overhead of the client/server model. Therefore, in a network with fewer nodes, the client/server model may have less execution time than that of the mobile agent model. However, if the number of node is large, the mobile agent model will perform better. Figure 3 (a)(right) shows the total energy the network consumed. We observe that the mobile agent computing always consumes less energy, because the amount of data transmitted are much reduced in the mobile agent computing, thus saving total energy.

B. The effect of number of mobile agents (m)

In this experiment, we fix the node number at 100 and observe the effect of the number of mobile agents to the performance of different computing paradigms. Without loss of generality, we assume each agent migrates the same number of nodes. We expect a constant profile from the client/server-based computing since it is irrelevant to the number of mobile agents. We can see from Fig. 3 (b) that the mobile agent model always has a less execution time and energy usage than the client/server model because the node number is large. Interestingly, the execution time of the mobile agent model decreases as the number of mobile agents increases and reaches the lowest point when there are five mobile agents. Then the execution time begins to increase. This is because more mobile agents will reduce the number of nodes each agent migrates, thus reduce the execute time. However, more mobile agents also cause more connections and more overheads. So a proper number of mobile agents can make the model perform more efficiently.

C. The effect of data size/mobile agent size (s_f/s_a)

In this experiment, we change the size of the data file s_f, but fix the other parameters and let $s_a = 1k$. We can see from Fig. 3 (c) that the execution time and energy consumption using the mobile-agent based computing are constant since the data are located at the local nodes and no data transmission is needed. When the data size is less than 16Kb, the client/server-based computing has less execution time than the mobile-agent based. However, the larger the data size, the more advantageous the mobile-agent based paradigm. This is because as s_f increases, more data needs to be transferred, thus increasing the time used by the client/server model. For the same reason, when the data size is less than 4Kb, the client/server-based computing consumes less energy.

D. The effect of the overhead ratio (o_f/o_a)

In this experiment, we fix all other parameters in the basic network, and observe the effect of the overhead ratio o_f/o_a (between 0.1 and 4.0) on the performance of different computing paradigms. We can see that when the ratio is greater than 0.4 (Fig. 3 (d)(left)) and 0.8 (Fig. 3 (d)(right)), the client/server-based computing starts to perform worse than the mobile-agent model since the larger the o_f, the longer the execution time and the more energy consumed.

V. A CLUSTER-BASED HYBRID COMPUTING PARADIGM

From the above experiments, we can draw the conclusion that the mobile agent paradigm performs much better when the number of nodes is large. On the other hand, mobile agent paradigm will suffer longer network latency when the number of nodes is small because of its bigger overhead. So, the mobile agent paradigm is more suitable for large networks while the client/server paradigm has advantages in small networks. Based on this conclusion, we propose a cluster-based hybrid computing paradigm combining the advantages of both client/server and mobile agent paradigms. The idea is to divide the network into several clusters. Each cluster has a cluster head which manages several nodes. We design two schemes to meet different network conditions.

Scheme A: The mobile agent paradigm is used *within* a cluster and the client/server paradigm is adopted *between* cluster heads. Within each cluster, the cluster head sends out one mobile agent processing data on the nodes within that

cluster. After the mobile agent returns, the cluster head sends the partial result returned by the agent to the processing center. We should restrict the number of clusters so that the number of nodes within a cluster is large, and the number of cluster heads is small. Thus we can take full advantages of both the client/server paradigm and the mobile agent paradigm.

Scheme B: The client/server paradigm is used *within* a cluster and the mobile agent paradigm is adopted *between* cluster heads. The nodes within a cluster send data to the cluster head and the cluster head behaves like a processing center that generates the partial integrated result. After all the cluster heads obtain the result, a mobile agent is dispatched from the processing center to each cluster head to integrate these results. We should choose this scheme when the number of clusters is large, and the number of nodes within a cluster is relatively small.

We use the same metrics to evaluate the performance of the hybrid model. The execution time model for scheme A of the hybrid paradigm is:

$$t_{hy-A} = t_{trans-wc} + 2(m + n_{node})o_a + t_{trans-bc} + 2n_{cluster}o_f \tag{5}$$

where $t_{trans-wc}$ and $t_{trans-bc}$ are the network transfer time spent within one cluster and between clusters, respectively, and are obtained from simulation, n_{node} is the number of nodes within one cluster, $n_{cluster}$ is the number of clusters, and m is the number of the mobile agents a cluster head dispatchs. Similarly, for scheme B, the execution time is:

$$t_{hy-B} = t_{trans-wc} + 2n_{node}o_f + t_{trans-bc} + 2(m + n_{cluster})o_a \tag{6}$$

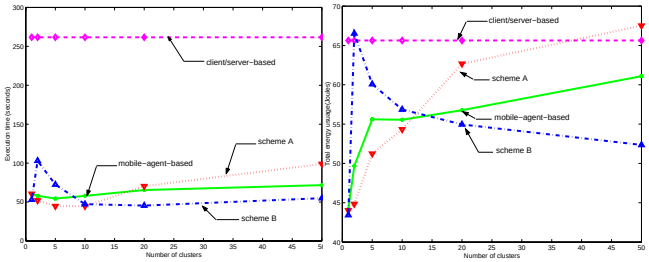
The energy model for scheme A is:

$$\begin{aligned} e_{hy-A} &= e_{trans-wc} + 2m(n_{node} + 1)p_{processor}o_a \\ &\quad + e_{trans-bc} + 2mn_{cluster}p_{processor}o_f \end{aligned} \tag{7}$$

where $e_{trans-wc}$ and $e_{trans-bc}$ are the total energy usage within one cluster and between clusters, respectively. For scheme B, it is:

$$\begin{aligned} e_{hy-B} &= e_{trans-wc} + 2mn_{node}p_{processor}o_f \\ &\quad + e_{trans-bc} + 2m(n_{cluster} + 1)p_{processor}o_a \end{aligned} \tag{8}$$

Again, we use ns-2 to evaluate the performance of the two schemes designed for the hybrid paradigm and compare it with the client/server and mobile agent paradigms. We set the number of nodes $p = 100$ and choose $m = 1$ such that the number of clusters in the hybrid model is the same as the number of mobile agents used in the mobile agent model. The other parameters in the experiment are the same as those in the basic network. The experimental results are shown in Fig. 4. From Fig. 4 (a) we can see that scheme A of the hybrid paradigm has the lowest execution time when $n_{cluster} < 11$. This is where the network is configured to be suitable for the client/server paradigm used among the cluster heads. When $n_{cluster} > 11$, scheme B performs the best where the mobile agent paradigm is adopted among the cluster heads. From Fig. 4 (b), we can observe similar pattern: scheme A is the most energy efficient when $n_{cluster} < 12$ and beyond



(a) t_{cs} vs. t_{ma}. (b) e_{cs} vs. e_{ma}.

Fig. 4. The comparison of three computing paradigms.

this point, scheme B becomes the least energy consumption approach when $n_{cluster} > 14$.

Based on the above experiments, we can see that the cluster-based hybrid computing paradigm can always be advantageous to both the client/server and mobile agent paradigms if we choose the proper scheme according to the network clustering conditions.

VI. Conclusions

This paper focussed on the performance evaluations of three distributed computing paradigms for collaborative information processing in sensor networks. We presented simulation models using execution time and energy usage as metrics. We designed and implemented several scenarios to investigate the effect of different parameters on the performance of the computing paradigms. We first showed the pros and cons of the client/server paradigm and the mobile agent paradigm. Based on this analysis, we proposed a cluster-based hybrid computing paradigm to benefit from both paradigms. We identified two schemes in this paradigm and showed that they are advantageous to both paradigms under different network clustering conditions in terms of the execution time and energy usage. In the context of sensor networks with hundreds or even thousands of nodes, high node failure rate, unreliable communication links, and reduced bandwidth, the mobile-agent-based computing and the cluster-based hybrid computing provide energy-efficient and scalable solutions for collaborative processing with low latency.

References

[1] A. Fuggetta, G. P. Picco, and G. Vigna, "Understanding code mobility," *IEEE Trans. on Software Engineering*, vol. 24, no. 5, pp. 342–361, 1998.

[2] H. Qi, S. S. Iyengar, and K. Chakrabarty, "Multi-resolution data integration using mobile agents in distributed sensor networks," *IEEE Trans. on Syst., Man, and Cybern. C*, vol. 31, no. 3, pp. 383–391, Aug. 2001.

[3] A. Helal *et al.*, *Any Time, Anywhere Computing*. Kluwer Academic Publishers, 1999.

[4] D. Estrin, A. Sayeed, and M. Srivastava, "Mobicom 2002 tutorial: Wireless sensor networks."

[5] J. Russell and M. Jacome, "Software power estimation and optimization for high performance, 32-bit embedded processors," in *Proceedings of ICCD'98*, Oct. 1998.

[6] L. Breslau, D. Estrin, K. Fall, *et al.*, "Advances in network simulation," *IEEE Computer*, vol. 33, no. 5, pp. 59–67, May 2000.