Mobile Agent based Architecture for Wireless Sensor Networks

Rijubrata Bhaumik
Helsinki University of Technology
rbhaumik@cc.hut.fi

Abstract

Wireless Sensor Networks(WSN) have emerged as one of the key growth areas for wireless networking and computing technologies. Usually in sensor networking, traditional client-server computing architecture is employed where data at multiple nodes is transfered to a processing element. In the mobile-agent based computing paradigm, the motto is to move the computation to the data rather than data to the computation. Using this paradigm, the communication cost, specially over low bandwidth links is greatly reduced. In this paper we study the recently proposed paradigm known as mobile agent based distributed sensor network (MADSN) by Qi et. al [6]. The paper also addresses the advantages and the challenges associated with this paradigm.

KEYWORDS: mobile agents, Wireless sensor networks

1 Introduction

Wireless Sensor Networks(WSN) have emerged as a new computing paradigm spearheading the cause of pervasive computing. WSN consist of tiny sensors to sense the environment with the goal of collecting information to make an informed decision. Example applications include habitat monitoring surveillance, medical care and structural monitoring. The lengthy deployment intervals and not so conducive environment implies restraint on energy usage as human interaction might not be possible. So life time of a node is effectively determined by its battery life. The main drainage of battery is due to transmitting and receiving data among nodes and the processing elements. Apart from battery life there is a constraint on bandwidth as the sensors are usually deployed on a low bandwidth wireless link. So data movement should be as minimum as possible. Depending upon the inter node distance, there is a lot of redundancy in the data collected. Some amount of redundancy is required to add robustness to the system as the nodes themselves are not very reliable, so it should be a part of the optimal deployment strategy. By moving the software code (Mobile Agent) itself to the nodes a large amount of the spatial redundancy in closely located nodes can be eliminated. The Mobile Agent (MA) is a special kind of software which visits the network either periodically or on demand and performs data processing autonomously while migrating from node to node.

The need to reprogram the wireless sensor network may arise due to changes in application requirements or bug fixes. Over a period of time, protocols keep changing due to performance or security. We would also want to use the sensor data for a variety of applications. A "one-deployment, multiple applications" trend hence demands a dynamic approach to programmability. So instead of flashing the nodes with the same code and doing a remote code update from time to time to all nodes might not be a good option. A better option might be to introduce MAs in the system which can hop from node to node within the same network, do the relevant processing and then transmit the result back preferably from the last node it needs to visit or a node in the nearby neighborhood of the last node having the highest battery life left. MAs also add extensibility and scalability to the network as the MAs can be programmed to support adaptive network load balancing or to carry out task adaptive fusion processes. In the client server model sensor data is passed to a central location for fusion. Given the vulnerability of the wireless link, transmitting all non critical sensor data can compromise the security. So the incremental fusion done by a MA at every hop might be a better solution. Traditionally, in Distributed Sensor Networks (DSN), the sensor nodes are flashed with the code before deployment. If we need to reprogram the network, the code update approach which is described in the next section in detail is followed. This process is bandwidth intensive and hence not energy efficient. So, the MA based approach is gaining prominence. The following table shows the main difference between a DSN and a MADSN.

1.1 DSN vs MADSN

Table 1.

Features	DSN	MADSN
Element moving in the network	Data	Computation
Bandwidth	High	Low
Scalable?	No	Yes
Extensible?	No	Yes
Affected by network reliability?	Yes	Yes
Fault Tolerable?	Yes	Yes

Table 1: DSN vs MADSN

2 Related Work

A lot of work in the area of wireless sensor network in recent times has been focusing on providing high level abstractions of complex low level concepts to application programmers. As the usage of wireless sensor networks becomes more varied, the need for middlewares for complexity encapsulation is even more felt. An approach to develop Sensor network middleware has been proposed by C.Fok et al[5] in the form of Agilla, which proposes an agent based approach. UCLA's SensorWare [4] is a system which falls under the category of active sensor frameworks, which are very close to the mobile agent based approach. In Berkley, a similar framework called Maté [16] has been developed. It is a tiny communication-centric virtual machine built on top of TinyOS [8]. Maté and SensorWare share the same goal and objective, but differ each other in implementation. Maté targets the Berkley designed family of sensor nodes called motes which are extremely resource constrained. Programming for Maté is also a difficult task and due to ultra compact instruction set, even a medium size task may lead to a big program. SensorWare targets a richer platform usually having a 1 Mbyte program memory and a 128 Kbytes of RAM. It also allows easy programming with a high level scripting language. MIT's Pushpin [11] is the realization of the "paintable computer" programming targeting hardware even more resource constrained than the motes. In all the above 3 frameworks, the code is made autonomously mobile enabling its migration and replication throughout the network without user intervention. From an application point of view, there exists systems like [10], where researchers have already built a mobile agent based WSN to monitor the vital signs in a patient.

3 WASN Programmability

There is an inherent need to reprogram the wireless sensor network due to requirement changes, so while designing the wireless sensor network we need to focus on the dynamic programmability aspect of the sensor network. There are three main ways of dynamically programming the wireless sensor network - i) the code update approach, ii) database model approach and iii) mobile agent based approach.

3.1 Code Update approach

The code executing in the nodes is dynamically updated where a central entity is responsible for the control of the tasks. The update can take place on various abstraction levels, lowest being a flash memory update or re flashing, or at a higher level which use a more modular approach allowing multiple users, like Impala [1]. Often changes are incremental and small, so a full flash update is expensive. So, Reijers et al [18] devised a Unix diff-like approach using various patch list commands to minimize the traffic. In this approach, it should be possible to update a running code which can be done by building the code in the EEPROM and using a small piece of code in RAM to load the image in the Flash memory. The system should be resilient to packet losses. The code update can be global, that is all the nodes in the network are flashed, or the user can statically select the nodes. The selection can be based on various criteria like location, id or any other static property. Code update approaches do not take into account the programmability of the WASN as an aggregate, instead it just views the network as a mere aggregation of nodes.

3.2 Database Model Approach

In this approach, the WASN is viewed as a distributed database system. Multiple users inject database like queries to be autonomously distributed in the network. The query's task is to retrieve the required information by finding the appropriate nodes and possibly aggregate the data preferably using custom collaboration among a set of nodes, as they are routed back to the user. Heidemann et al [7] used a low level naming scheme using directed diffusion [9] as the underlying protocol. Cougar [15] developed at Cornell uses SQL, with minimal additions to the language, to create a similar distributed database system. When a query arrives at a node, the query resolver which is present in every node, decides which nodes to pass the query for optimal performance. TinyDB [12] developed in Berkley provides some optimizations like exploitation of shared medium and hypothesis testing. Although this model helps us to program the network as a whole, yet it is not expressive enough to implement any distributed algorithm. Being declarative in nature, the user cannot specify the algorithm to extract the data, as the complications of embedded and distributed programming is abstracted. So the data extraction is done in predefined ways and hence only specific requirements specially the aggregation types, have optimal performance.

3.3 Mobile agent based approach

The agent based computing paradigm addresses a number of issues faced in the field of WASN like -
(1)*Scalability*: Network performance is unaffected by the increase in the number of nodes and if the agent architecture can support adaptive load balancing, then much of the redesign can be done automatically.
(2)*Reliability*: Mobile agents can be sent when network connection is alive and they can return the required information when the connection is re-established, so reliability of the network does not affect performance very much.
(3)*Extensibility and task adaptability*: Mobile agents can be used to carry out task specific integration process. Hence with the same network, different tasks and hence different applications can be achieved. So it extends the functionality of the WASN.
(4)*Energy awareness*: The itinerary of the MA is determined by the information gain and energy constraints. If the itinerary of the MA is not already known before hand, then the genetic algorithm based solution to compute an approximation to the optimal source-visiting sequence as proposed by Q.Wu [19] can be used.Mobile agent based Directed diffusion [13] can also determine the path traversed by a MA. Moreover, MA usually have a smaller footprint than the sensed data. So in every way, mobile agent paradigm is more energy aware than other paradigms.
(5)*Progressive accuracy*: A MA is always carrying a partially integrated result generated as it migrates from one node to another. Assuming the MA follows the gradient along the increase of information gain, we can terminate the itinerary whenever the integration accuracy surpasses a given threshold, thereby saving network bandwidth and computation of more visits to other nodes.
Qi et al. [6] give a quantitative approach to the performance

evaluation of the mobile agent based system compared to the usual client-server paradigm with respect to execution time as well as energy consumption.

4 System architecture and design of MAWSN

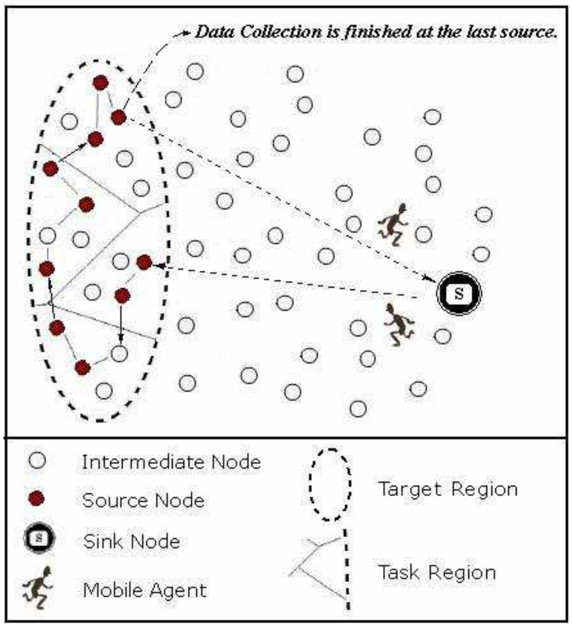
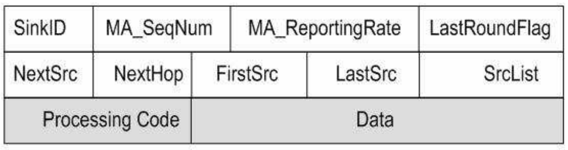


Figure 1: Architecture of MAWSN

Fig. 1 [14] shows the basic architecture of a MAWSN as presented by Chen et al. [14], where a sink queries multiple targets simultaneously by means of the Mobile Agent. The MA moves from node to node as shown in the figure aggregating and integrating data. The output of a task maybe a simple aggregation done by collaborative process or a complex return of a large volume of sensed data like a picture from an image sensor. For efficiency reasons, multiple concurrent task requests can be clubbed in one packet and the results for these tasks can be concatenated in a single packet to save communication overhead, if the minimum quality of service, like latency bound is not violated. Usually target region is far away from the sink, so the energy savings can be significant. The multiple tasks should be initiated simultaneously using a single packet only if (1) all the tasks which are combined can be processed by the common code part of the MA packet, to avoid communication overhead for additional processing code, (2) with respect to the distance between the center of the combined task region and the sink, the task area are likely to be close to each other. (3) all the tasks are requested concurrently by the application.

We can assume that the sink is aware of the set of nodes which the MA is going to visit and the itinerary is already setup. The routing mechanism can be thought of as a one phase-pull Directed Diffusion [9]. A more MA specific directed diffusion is presented by Chen et al. in [13].

An example of a packet structure for a MA is shown in Fig. 2. *Sink_ID* and *MA_SeqNum* identifies a packet uniquely.

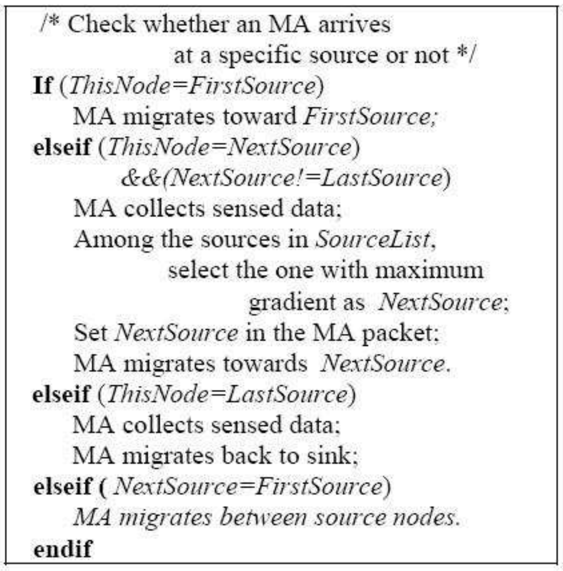


SinkID	MA_SeqNum	MA_ReportingRate		LastRoundFlag
NextSrc	NextHop	FirstSrc	LastSrc	SrcList
Processing Code		Data		

Figure 2: MA Packet

MA_SeqNum is incremented the sink dispatches a new MA packet. The *SourceList* which specifies the set of nodes the MA should visit, has the *FirstSource* and the *LastSource* marking the starting and the end point of the MA's itinerary. *NextSource* specifies the next destination node to be visited and the *NextHop* specifies the immediate next hop of a current node. A *round* is defined as period between the collection of the packet from the *FirstSource* to the collection of the packet from the *LastSource*. *LastRoundFlag* indicates that the current round is the last round in the data collection process and is set by *FirstSource*.

The payload of a MA packet consists of two types of data, the *ProcessingCode*, code used to process the sensed data and the *Data* which is actually the aggregate data. At the start of every round, *FirstSource* will create a new MA by copying from a stored one with the frequency of *MA_ReportingRate*. During the migration phase when the MA arrives at a sensor node, the current node's id is checked to see if it reached on the destination source. If not, it continues to migrate to the correct node, else the MA collects the locally processed data, deletes the current target source from the *SourceList* and chooses the *NextSource* as the next destination. Fig. 3 illustrates the pseudo code describing the migration algorithm followed by a MA in a round.



```
/* Check whether an MA arrives
        at a specific source or not */
If (ThisNode=FirstSource)
    MA migrates toward FirstSource;
elseif (ThisNode=NextSource)
        &&(NextSource!=LastSource)
    MA collects sensed data;
    Among the sources in SourceList,
            select the one with maximum
                gradient as NextSource;
    Set NextSource in the MA packet;
    MA migrates towards NextSource.
elseif (ThisNode=LastSource)
    MA collects sensed data;
    MA migrates back to sink;
elseif ( NextSource=FirstSource)
    MA migrates between source nodes.
endif
```

Figure 3: Mobile Agent Migration Algorithm

5 Challenges

Challenges usually encountered while programming with mobile agents as discussed in [2]. Contrary to the belief that mobile agents optimize resource access and network overhead, quantitative analysis have shown that mobile agents are not as effective in practice as they are supposed to be. Mobile agent based systems are difficult to design, while most distributed applications can be addressed by well known techniques. Implementation, testing and debugging of mobile agent based systems is a very daunting task. Mobile agents are difficult to authenticate and control and are very similar to worms in their functioning. Mobile agents also lack a shared language/ontology to interact with the environment. So some researchers like Giovanni Vigna [2] feel that other forms of code mobility or simpler solutions should be tried to avoid the issues that have to be addressed while dealing with mobile agents.

6 Conclusion

The mobile agent based WSN promises a lot of efficiency when compared to the more traditional client server approach. Also Chen et al. [14], for a given set of parameters, have derived the conditions where MA based WSN exhibits better performance than the client server approach in terms of packet delivery ratio, energy consumption and end-to-end delay. Given the autonomous nature of the viral like migration of MAs, significant improvements in security measures need to be taken. Mobile agent based system help to program the network as a whole and truly follow the "one-deployment, multiple applications" principle by injecting task specific code and not just tune in parameters. Boulis [3] has clearly shown the advantage of the mobile based paradigm compared to the other approaches like code update or the distributed database model using a target tracking system. Qi et al. [17] has shown the usage of mobile agent to carry out distributed sensor integration tasks by taking an example of target classification to illustrate the efficiency of the MA computing model.

References

[1] T. Liu and M. Martonosi. Impala: A Middleware System for Managing Autonomic, Parallel Sensor Systems. In *Proceedings of the ACM SIGPLAN Symposium on Principles and Practices of Parallel Programming*, 2003.

[2] Vigna, G. Mobile agents: ten reasons for failure. In *Mobile Data Management, 2004. Proceedings. 2004 IEEE International Conference on*, volume 16, pages 298– 299, June 2004.

[3] Athanassios Boulis . Programming Sensor Networks with Mobile Agents. In *ACM International Conference On Mobile Data Management*, volume x, pages 252–256, May 2005.

[4] Athanassios Boulis, Chih-Chieh Han, and Mani B. Srivastava. Design and Implementation of a Framework for Efficient and Programmable Sensor Network. `www.ee.ucla.edu/~simonhan/simon_paper/SensorWare-Mobisys03.pdf`.

[5] Chien-Liang Fok, Gruia-Catalin Roman and Chenyang Lu. Mobile agent middleware for sensor networks: An application case study. Technical report, 2005. `http://cse.seas.wustl.edu/techreportfiles/getreport.asp?399`.

[6] Hairong Qi, Yingyue Xu and Xiaoling Wang. Mobile-agent-based Collaborative Signal and Information Processing in Sensor Network. In *Proceedings of the IEEE*, volume 91, pages 1172 – 1183, Aug 2003.

[7] J. Heidemann, R. Govindan, and F. Silva. Building efficient wireless sensor networks with low-level naming,Ť sosp. In *In SOSP*, 2001.

[8] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. Pister. System architecture directions for networked sensors. In *In Architectural Support for Programming Languages and Operating Systems*, pages 93–104, 2000.

[9] C. Intanagonwiwat, R. Govindan, and D. Estrin. Directed diffusion: a scalable and robust communication paradigm for sensor networks. In *ACM International Conference on Mobile Computing and Networking (MOBICOM'00*, pages 56–67, 2000.

[10] John Herbert; John Oapos;Donoghue; Gao Ling; Kai Fei; Chien-Liang Fok. Mobile Agent Architecture Integration for a Wireless Sensor Medical Application. In *Web Intelligence and Intelligent Agent Technology Workshops, 2006. WI-IAT 2006 Workshops. 2006 IEEE/WIC/ACM International Conference on*, pages 235 – 238, Dec 2006.

[11] J. Lifton, D. Seetharam, M. Broxton, and J. Paradiso. Pushpin computing system overview: a platform for distributed, embedded, ubiquitous sensor networks. In *in F. Mattern and M. Naghshineh (eds): Pervasive 2002, Proceedings of the Pervasive Computing Conference*, pages 139–151. Springer Verlag, 2002.

[12] S. Madden, R. Szewczyk, M. J. Franklin, and D. Culler. Supporting aggregate queries over ad-hoc wireless sensor networks. In *In Workshop on Mobile Computing and Systems Applications*, pages 49–58, 2002.

[13] Min Chen, Taekyoung Kwon, Yabghee Choi and Victor C.M. Leung . Mobile Agent-Based Directed Diffusion in Wireless Sensor Networks. In *The IEEE Conference on Local Computer Networks*, volume 17, page 529, Nov 2005.

[14] Min Chen, Taekyoung Kwon, Yong Yuan and Victor C.M. Leung. Mobile Agent Based Wireless Sensor Networks. In *Journal of computers*, volume 1, April 2006.

[15] J. G. P. Bonnet and P. Seshadri. Querying the physical world. *IEEE Personal Communications*, 7:10–15, 2000.

[16] D. C. Philip Levis. Maté: a tiny virtual machine for sensor networks. In *ACM Proceedings of the 10th international conference on Architectural support for programming languages and operating systems*, pages 85 – 95, 2002.

[17] H. Qi and X. Wang. Multisensor data fusion in distributed sensor networks using mobile agents. In *In Proceedings of 5th International Conference on Information Fusion*, pages 11–16, 2001.

[18] N. Reijers and K. Langendoen. Efficient code distribution in wireless sensor networks. In *WSNA '03: Proceedings of the 2nd ACM international conference on Wireless sensor networks and applications*, pages 60–67, 2003.

[19] Wu, Q.; Rao, N.S.V.; Barhen, J.; Iyenger, S.S.; Vaishnavi, V.K.; Qi, H.; Chakrabarty, K. On computing mobile agent routes for data fusion in distributed sensor networks. In *Knowledge and Data Engineering, IEEE Transactions on*, pages 740 – 753, 2004.