Feasibility Analysis of Achieving Mobile Agents for Wireless Sensor Network based on LooCI

Yuechun Wang, Ka Lok Man, Steven Guan, Danny Hughes

Abstract -- **Distributed sensing in combination with wireless communication techniques and self-organising deployment approaches enable Wireless Sensors Networks (WSN) to play a crucial role in our daily life. An increasing number of dynamic sensing applications have been developed, and there is a growing demand for middleware to bridge the gap between these applications and a diverse range of underlying operating systems. In this paper, we present a prototype middleware that combines a Mobile Agent-based WSN system in conjunction with the component based LooCI middleware. The system has both merits of agent-based WSN system as well as the benefits of LooCI, which include efficient energy use and providing the possibility of platform independence and component model programming language independence. The feasibility analysis along with potential implementation approaches of this system are proposed in this paper. The direction and value of on-going research based on the research work proposed in this paper is presented as well.**

Index terms -- Mobile Agent, LooCI, WSN, Middleware

I. Introduction

WITH features of environmental sensing, self-organization, and flexibility, Wireless Sensor Networks (WSNs) have independently arisen in various application scenarios such as space exploration and logistic tracking [1]. Two functionally different types of nodes are involved in WSN system, i.e. terminal resources sensor nodes and sink nodes. Terminal sensor nodes that highly distributed in geographical sensing area refer heterogeneous types and operation systems. These nodes have purpose of data collection and could be GPS module, RFID module and/or sensors for specific monitored parameters. Sink nodes, which played as connectors between terminal sensor nodes and higher service centre, transmit both data packets from terminal sensor nodes to back-end systems and control instructions from the back-end to sensor nodes. However, several critical problems such as restricted the lifespan of sensor nodes and limited energy consumption due to the resource limits of sensor nodes and the increasing scale and complexity of the end-to-end WSN system.

While there are many families and types of middleware discussed in [2], there is no single existing middleware can meet all of the demands of WSN systems while operating within the strict limitations of WSNs. One of the most crucial challenges for middleware design is how to deal with resource trade-offs. One example is the energy consumption trade-off between data processing performed on local sensor nodes and data transmission among sensor nodes. In addition, resource discovery and code management for dynamic applications cannot be solved using existing middleware approaches. When dealing with the trade-off as mentioned above or the conflict occurred during resource management, a reasonable resource allocation approach is important. With ability of data processing, mobile agent is one of the techniques that could achieve resource allocation ideally.

Mobile Agents (MA) [3] are one of the state-of-the-art techniques that contributes to extend lifespan of terminal resources sensor nodes by itinerating among target nodes to collect information and transmit to interested sink nodes in certain area. One of the design challenges of MA is the middleware system design. As MA is defined as an application-specific software, a well-designed middleware that supports potential add or update of MA based applications is necessary. Besides, middleware should be able to bridge the gap between various applications and different underlying operations in WSN system.

The Loosely-coupled Component Infrastructure (LooCI) [4] is an event-driven component-based middleware system, which is platform-independent and support for large scale distributed sensing applications.

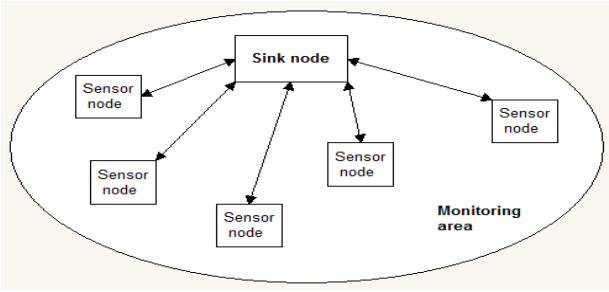
The remainder of this paper is organized as follows. Section II introduces Mobile Agent in WSN. Section III presents LooCI and its relevant work. The feasibility analysis of achieving MA on LooCI is proposed in Section IV and the conclusions of this paper are in Section V.

II. Mobile Agent in WSNs

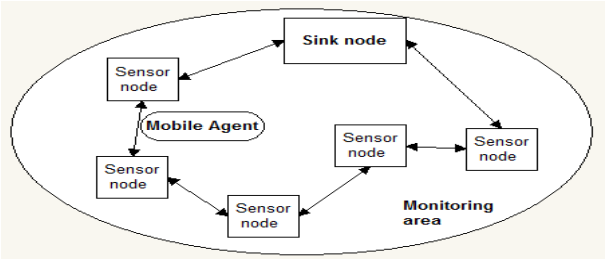
A Mobile Agent (MA), which is described as a special software that includes executed codes, can migrate among terminal sensor nodes to collect information and transmit to interested sink nodes in WSNs [5-7]. This definition indicates that MA is an application-specific software and has high mobility. Its ability of data processing decides that WSN system with MA can significant extend lifespan of

Yuechun Wang is with Xi'an Jiaotong-Liverpool University, China (email: yuechun.wang@xjtlu.edu.cn)
Ka Lok Man is with Xi'an Jiaotong-Liverpool University, China (email: ka.man@xjtlu.edu.cn)
Steven Guan is with Xi'an Jiaotong-Liverpool University, China (email: steven.guan@xjtlu.edu.cn)
Danny Hughes is with IBBT-DistriNet, KU Leuven, Leuven, B-3001, Belgium (email: danny.hughes@cs.kuleuven.be)

ISBN: 978-988-14047-7-0
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)



(a)



(b)

Fig. 1 Compare of the WSN system with and without MA. (a)The system without MA, and (b) the system with MA

terminal resources sensor nodes by reducing the energy consumption of data processing on local sensor nodes. Data fusion, localisation [2], re-allocation as well as update of tasks of resources sensor nodes are typical applications of MA.

To achieve MA in WSN system, there are three basics that should be addressed: 1) architecture of overall system, 2) MA itinerary arrangement which denotes the order of sensors to be visited by MA, and 3) middleware support. We focus on middleware system design and implementation in this paper.

Fig. 1 shows the comparison of the WSN system that with and without MA. In a monitoring area with randomly distributed sensor nodes and one sink node, the sink node in Fig.1 (a) needs to collect data from each terminal sensor node separately. Therefore, the bandwidth requirement of channels connected with the sink node is unduly high. When adding a MA into the same system as shown in Fig.1 (b), data exchange between each sensor node and sink node has been dramatically reduced. Instead, MA that transmitted by sink node travels around all the terminal sensor nodes to collect, pre-process, and deliver useful data. Therefore, the bandwidth requirement for the channels that connect with sink node is reduced. Through this approach, MA can reduce the energy consumption of data exchange between terminal sensor nodes and sink node as well as bandwidth requirements.

III. LooCI

The Loosely-coupled Component infrastructure (LooCI) as presented in [4] is a state-of-the-art middleware system that

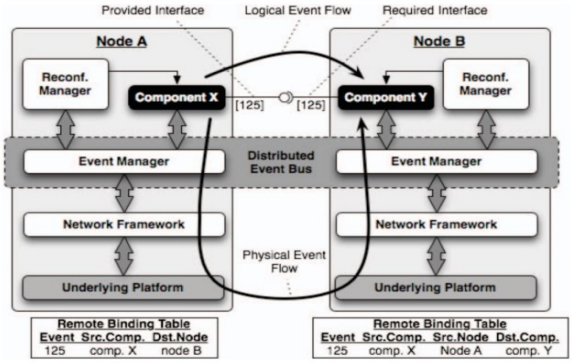


Fig. 2 LooCI hierarchical architecture

bridges the gap between distributed WSN applications and diverse platforms. As described in its name, LooCI is composed of a reconfigurable component model, a hierarchical system, and a distributed event bus. The architecture of LooCI is shown in Fig. 2 taken from [4]. These features denote that LooCI can support a separation of distributed applications from component implementation. Operating system that could support LooCI currently includes OSGi, Contiki, Squawk, and Android.

IV. MA on LooCI – Feasibility Analysis

A. Functional Needs

To clearly specify the functional needs of a mobile agent based middleware, two core components should be analysed and defined – MA and relative supporting platform.

Mobile Agent

Composed modules for each agent are shown in Fig. 3. Condition of an agent shows its current status so that agent manager can do further treatment on it. Self-control logistic contributes to self-organization of an agent, which decide the application requirement of the agent. Perceptron and effector present the 'communication' between agents and environment – environmental changing can be detected by agents and also agents can give out feedbacks. As introduced in section II, an agent is a special software that includes executed codes, and can migrate among terminal sensor nodes to collect information and transmit to interested sink nodes in WSNs. To achieve this feature, four elements are essential for an agent – *Agent ID* for unique defining the agent on sensor nodes, *Agent condition* which includes operation condition and migration condition,

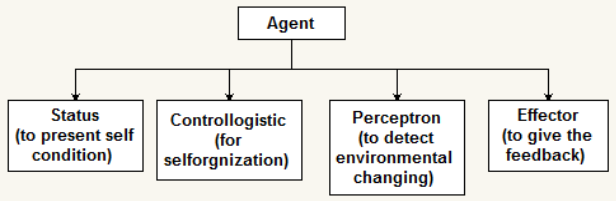


Fig. 3 Composed modules for the agent

ISBN: 978-988-14047-7-0
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

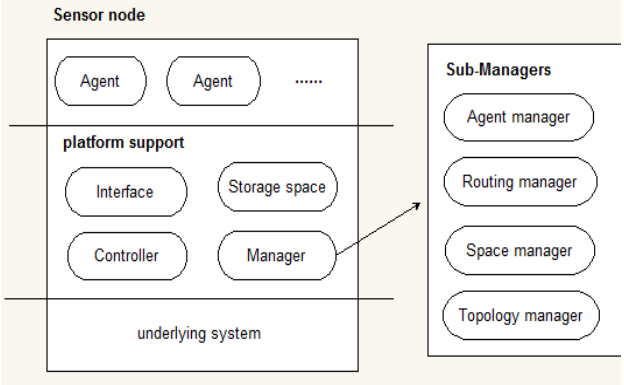


Fig. 4 Overall structure of the MA system

Agent code that includes main functional codes of the agent, as well as the *Storage space* for keeping data gathered from sensor nodes.

Supporting platform

The functionality of the operating platform includes providing reusable framework interface and providing a set of reusable components for application system developers.

As the overall structure of the system shows in Fig. 4, in the middleware layer includes four core modules (labelled as platform support in Fig. 4): the manager, interface, controller and storage space. The manager is composed of four sub-managers which are Agent manager, Routing manager, Space manager, and Topology manager. These interoperable managers have the following responsibilities: The Agent manager is in charge of agent operation and management. The routing manager is in charge of the establishment and maintenance of neighbour lists on sensor nodes. The Space manager is in charge of management and allocation of storage space for agents. Finally, the Topology manager is in charge of the sensor nodes status and serves that optimises the topology of the whole network.

In this paper, for the sake of simplicity, only the Agent manager and the Topology manager are presented in detail.

Fig. 5 illustrates the block diagram of operation for the Agent manager on supporting platform. The Agent manager achieves three fundamental functions: i) recognising agents and control messages; ii) allocating resources (e.g. storage space in Queues); and iii) processing agents. Hence components that constitute the Agent manager should be addressed as: i) a receiving monitor to recognise and receive agents around sensor nodes or control messages from controller; ii) an agent status detector to detect the updated status of agents; iii) three fix-length queues to store agents – Received Queue to collect received agents, Operation Queue to organise agents under 'Ready_to_work' status, and Delivery Queue to collect agents that ready to migrate to other nodes. Pseudo code for the Agent manager will be introduced in the next section.

Being different from the traditional WSN topology management which controls the sensor nodes through the backbone node, the topology manager of an agent-based network controls sensor nodes according to the status of agents on the node. As topology management is closely relative to routing management of the network, some functionality of topology manager is incorporated with the routing manager such as to update list of the neighbour nodes. The block diagram of the Topology manager is illustrated in Fig. 6. With the initial status of all nodes in the network set to idle, agent detection is followed on. If there are agents inside the monitoring area of a sensor node or any agent under operation on the node, the node

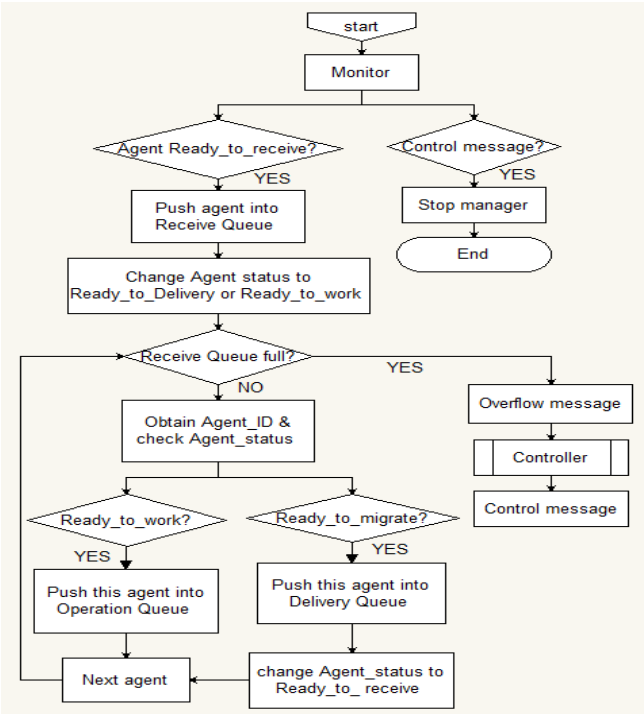


Fig. 5 Block diagram of Agent Manager

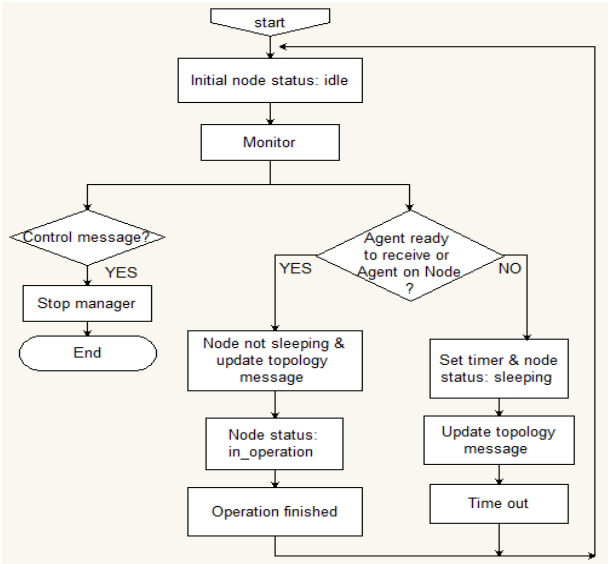


Fig. 6 Block diagram of Topology manager

ISBN: 978-988-14047-7-0
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

will maintain awake status rather than to sleep; and the topology message will be updated by the next-level nodes in the meantime. On the contrary, those sensor nodes which cannot detect any agent within monitoring area and do not have any agent operating on it will change to sleeping status with a timer; and send the topology message to next-level nodes. When the agent operation has finished or the time is up on the nodes that fell asleep, the status of sensor nodes will change to idle again. Based on the methodology introduced above, core modules for the Topology manager include an Agent detector, a timer, and a logger for recording node status. Besides, there should be a logger to record and update the topology message from neighbour nodes.

B. Analysis of importing MA in LooCI

As a component infrastructure and event driven middleware system, LooCI is a good option to achieve MA. The analysis of LooCI operating principle connected with functional requirements presented in section IV.A is as follows.

Three core sections of LooCI are codebase and component, event, and wires. Codebase is an executable code file with a node unique ID that can be deployed to a node while component is a runtime unit which executes the functionality of codebase. In the other words, codebases provide executable code and components are the instantiation of codebases, which means all the manager modules referred in section IV can be achieved by declaring new codebases and components in LooCI. Components communicate by exchanging events through two interfaces – provided interface which is for publishing, and required interface which is for subscribing. These two interfaces can achieve the message exchange among four modules on supporting platform layer: control messages, status of agent, neighbour list, etc. An event is the basic unit for network communication. Events are delivered between sensor nodes which are similar to the role of mobile agents. Every event contains an event ID and payload which indicates the agent ID as well as storage space for both data and code on agents could be achieved. Wires are the medium for component communication. Events that have been delivered among diverse components on local or remote sensor nodes transmit through wires.

As a conclusion, core modules of LooCI can satisfy the functional needs of the MA middleware. The challenge is how to declare the manager modules and set suitable properties for the component.

C. Potential Implementation approach

The implementation approach of Agent manager is presented in Pseudo-code 1. The *AgentManager()* function starts from detecting matching agents or control messages from controller. In case any control message from controller detected by the agent manager, all the processes stop immediately.

When an agent with status Ready-to-receive has been detected, manager will push this agent into Receive Queue. If the code size of an agent is not overflowed from the size of Received Queue, which denotes the storage resources are adequate for current agent, then agent manager will process the agent by checking the status that is modified based on the duty of current agent. In case that the code size of received agent is larger than rest of the space in Received Queue, manager will send overflow message to the Controller; once the controller acquires the overflow message from manager, it will send a control message as a feedback. Inside each queue, a pointer is used to mark the agent that manager currently dealing with.

Pseudo-code 1 Agent manager

```
1: function AgentManager ()
2: active monitor
3: read packageHead
4: for ( i = 0, I< MaxReceive) do
5:       if head (I) = = controlMessageHead then
6:           stop function
7:       elseif head (I) = = agentHead AND
8:                Agent_status (I) = =Ready_to_receive)  then
9:           push Agent (I) → Received Queue
10:          change Agent_status (I)
11:          if agentSize (I) > maxQueue  then
12:               return 1     /* overflow */
13:          else return 0    /* not overflow */
14:          end if
15:          if Agent_status (I) = = Ready_to_work   then
16:              push Agent (I) → Qperation_Queue
17:          elseif  Agent_status = Ready_to_migrate  then
18:              push Agent (I) → Delivery_Queue
19:          end if
20:          Agent_status → Ready_to_receive
21:      end if
22:      I + +
23: end for
24: end function
```

Pseudo-code 2 Topology manager

```
1: function TopologyManager ()
2: initial Node_status → idle
3: while (Node_status = idle)
4: active monitor
5: update boolean flag1
6: if flagP = = 1  then      /*  package been detected */
7:       read packageHead
8:       if head = = controlMessageHead then
9:          stop function
10:      elseif head = = agentHead  then
11:           topology_message → node_activate
12:           Node_status → in_operation
13:      end if
14:      check Agent_engine
15:      if Agent_engine → finished
16:           boolean flagA = 1
17:           Node_status → idle
18:      else  boolean flagA = 0
19       end if
20: elseif flagP = =0  then    /* nothing been detected */
21:      while(flagA)
22:          Node_status → sleeping
23:          topology_message → node_sleeping
24:          set timer
25:          flagA = 0
26:       check timer
27:      if timer = = 0  then
28:      Node_status = idle
29:      end if
30: end if
31: end function
```

ISBN: 978-988-14047-7-0
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

Traversal of the agents in Received Queue is achieved through the while loop. In case that Received Queue is not empty, agent ID will be recorded as reference for agent engine which deals with executable codes of agents. Agent status therefore changed on the basis of next-step operation. If an agent is expected to operate its functional codes on the node, the Agent manager will push it into Operation Queue in which the agents will be handled by agent engine one by one. Another status of an agent is Ready-to-Migrate which denotes the agent has certain target node and prepares for delivery from current node. The Agent manager will push the agents with Ready-to-Migrate status into Delivery Queue in which the agents will be processed by agent transmitter. Status of the agents hanging in Delivery Queue then changes to Ready-to-Receive that gives a sign to next-level sensor nodes.

The implementation approach of the Topology manager shown in Fig. 6 is presented in Pseudo-code 2. *TopologyManager()* function is used to decide and manage the status of sensor nodes in the network so that optimises the allocation of limited energy on sensor nodes as well as limited bandwidth resources of whole network. The function starts from initialising all the nodes status as idle. When a node is under idle status, which denotes that the node is awake without any operation, it detects the active agents in its monitoring area as well as control messages from controller.

We set a FlagP as a package receive sign. If Boolean FlagP returns 1, it indicates that the control messages or agents are detected; header of the package will be checked. When the package is identified as a control message, all the processes on manager will shut down. Otherwise an agent will be received. We set a FlagA as a sign of agents active on the node. If Boolean FlagA returns 1, it indicates that active agents are operating on node; vice versa. With active agents around/on, a node sends topology message 'active' to next-level nodes. Then the status of the node changes to in-operation while agent engine dealing with received agent; and status returns to idle until all the operation on agent engine finished. If both Boolean FlagP and FlagA return 0, this indicates that no package is detected within monitoring area and no agent is under operation on the node. Therefore, topology manager closes the communication module on nodes and set a timer in the meantime. The Topology message 'sleep' is sent to next-level nodes at the same time. The sensor node will not exchange messages with other nodes until the time is up. The status of node returns to idle when the node is awake. To clarify the functionality of FlagP and FlagA, feedback of both flags and their relative indication are presented in Table 1.

TABLE I
FEEDBACK AND INDICATION OF FLAGS IN TOPOLOGY MANAGER

Flag No.	feedback	Indication
FlagP	0	Nothing has been detected in monitoring area
	1	There are packages existed in monitoring area
FlagA	0	No agent is under operation on node
	1	There are agents under operation on node

V. CONCLUSIONS

This paper contributed a feasibility analysis of achieving Mobile Agents for WSN system based on LooCI is presented. We analysed the current development situation of dynamic sensing applications and the value of implementing the MA for dynamic sensing applications. We proposed the functional needs of the MA middleware as well as the connection between LooCI and MA system. Potential implementation approaches of Agent manager and Topology manager are presented by several pseudo-codes that based on analysis and functional needs. Further research directions based on this paper are to implement the approaches on LooCI and validating the approaches on an embedded sensor platform.

ACKNOWLEDGEMENT

This project is supported by the Research Development Fund (#RDF14-03-12) of the Xi'an Jiaotong-Liverpool University, Suzhou, China.

REFERENCES

[1]. K. L. Man, D. Hughes, S. U. Guan, and P. W. H. Wong. "Middleware Support for Dynamic Sensing Applications." in International Conference on Platform Technology and Service (PlatCon), 2016, pp. 1-4.

[2]. H. Fukuda and P. Leger. "An Efficient Agent Location Management for Wireless Sensor Network." in 2015 IEEE 17th International Conference on High Performance Computing and Communications (HPCC), 2015 IEEE 7th International Symposium on Cyberspace Safety and Security (CSS), 2015 IEEE 12th International Conferen on Embedded Software and Systems (ICESS), 2015, pp. 1014-1019.

[3]. C. L. Fok, G. C. Roman, and C. Lu. "Rapid Development and Flexible Deployment of Adaptive Wireless Sensor Network Applications." in 25th IEEE International Conference on Distributed Computing Systems (ICDCS'05), 2005, pp. 653-662.

[4]. D. Hughes, K. Thoelen, J. Maerien, N. Matthys, W. Horre, J. Del Cid, C. Huygens, S. Michiels, and W. Joosen. "LooCI: The Loosely-coupled Component Infrastructure." in 11th IEEE International Symposium on Network Computing and Applications (NCA), 2012, pp. 236-243.

[5]. M. Chen, S. Gonzalez, and C. M. Leung Victor, "Applications and design issues for mobile agents in wireless sensor networks," IEEE Wireless Communications. Vol. 14, No. 6, pp. 20-26, 2007.

[6]. M. Usman, V. Muthukkumarasamy, X. W. Wu, and S. Khanum. "Securing mobile agent based Wireless Sensor Network applications on middleware." in International Symposium on Communications and Information Technologies (ISCIT), 2012, pp. 707-712.

[7]. R. Amine, K. Amine, B. Khalid, Z. Elhoussaine, and O. Mohammed. "Knowledge discovery in WSN using mobile agents." in Intelligent Systems and Computer Vision (ISCV), 2015, pp. 1-6.