Configuration of WSN using Application-aware Virtual Networks

M.S. Siddiqui, E.J. Cho, and C.S. Hong
Department of Computer Engineering, Kyung Hee University
Yongin, Korea
shoaib@networking.khu.ac.kr, {ejcho, cshong}@khu.ac.kr

Jongwon Choe
Department of Computer Science, Sookmyung Women's University
Seoul, Korea
chowjn@sookmyung.ac.kr

***Abstract*—In this paper, we propose protocols and mechanism for creation and working of Virtual Sensor Networks (VSN). Through simulations we support the concept and show that the proposal can achieve better results than ordinary WSN. We provide an application scenario for configuration management of WSN using the WSN concept .**

Keywords-virtual sensor network; wireless sensor network; network management, configuration management

I. INTRODUCTION

Wireless Sensor Network (WSN) [1] are often associated with scarce resources [2] hence, the early deployments of WSN were dedicated to a single application; an environment in which all the sensor nodes are looking for a certain event in a single interest domain. However, with recent increase in resources, such as CPU, memory etc., WSN is emerging with multifunctional sensor nodes and multi-domain networks collaborating within a WSN. The phenomenon is known as the Virtual Sensor Network (VSN) [3] [5].

We have developed a mechanism which utilizes the concepts of roles (based on application), which are assigned to each node. After the roles are assigned to each node, a distributed algorithm is executed to create different application-aware (role based) VSNs. Each node maintains VSN table to stay connected to its VSN, while supporting nodes are used to route messages in between the VSN nodes, if there is no direct connection.

The VSN phenomenon can be used for the dynamic configuration of WSN as it reduces the number of messages needed to be sent for the (re-)configuration of the network. Hence, we have applied our VSN proposal for the configuration management of WSN and provide detailed mathematical and simulation analysis.

The rest of the paper is articulated as follows. Section II gives related work in the field of VSN and configuration management. Section III provides describes the VSN concept and the ingredients of the VSN creation and maintenance protocol. Section IV gives the mechanism. Section V provides simulation results and section VI provides the configuration management mechanism in details. Section VII concludes our work.

II. RELATED WORKS

The concept of VSN is not new in fact many authors have used this concept for constructing resource efficient or application aware WSNs.

In [3] and [4], authors have constructed VSN for resource efficiency in concurrent applications. They have provided a cluster tree based self-organization of VSN. Nodes observing a same phenomenon create a tree based dynamic VSN. The authors compared their proposal with rumor routing and claim that their mechanism is efficient in reducing the power consumption at the sensor nodes. In [5], the authors have proposed an embedded agent based approach for creating VSN in a multipurpose WSN. By providing a certain degree of coverage and connectivity, hibernation of sensor nodes is used to save up the power consumption of the WSN.

The above mentioned papers failed to provide a protocol for dynamic VSN creation. Hence, we have developed a mechanism which utilizes the concepts of roles (based on application) for creating a VSN. We provide detailed protocol mechanism for the working of a VSN. We utilized the VSN concept for the configuration management of the WSN for changing attribute values and on-the-fly code update.

III. APPLICATION-AWARE VSN

"A VSN is formed by a subset of nodes of a WSN, with the subset dedicated to a certain task or an application at a given time." [3].

To create application-aware VSN, we need to group together similar nodes or node with same or similar application running on them. For making the WSN application-aware,

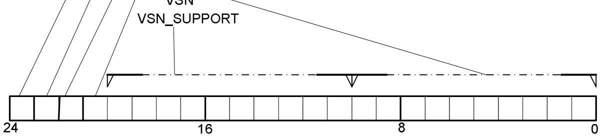

Figure 1. Role bit array scheme with role bits, and support-role bits.

[1] This research was supported by Basic Science Research Program through the National Research Foundation of Korea(
NRF) funded by MEST (No. 2010-0027645).
* Dr. C S Hong is the corresponding author.

we first identify the functionality of each sensor node and assign it a role. After that each node maintains its neighbor table and VSN table by communicating with its surrounding nodes and similar nodes, respectively. Follow afterwards are the description of each of these components which works together to enable the creation and maintenance of VSNs.

A. Roles:

A sensor node may have more than one type of sensor or it is a multi-functional device. The functionalities of a sensor node are each represented as a role of the sensor node, such as, TEMPERATURE, HUMIDITY, RF, and PHOTO (light sensing) etc.

Even if a node does not have a temperature sensor, it can be a part of a TEMPERATURE VSN as a supporting node. Then it can assume a role of TEMPERATURE_SUPORT. Similarly a node can be assigned a role of HUMIDITY_SUPORT, RF_SUPORT, and PHOTO_SUPORT. Here, we have mentioned only four types of sensors but that does not mean that protocol can only support these four kinds of sensors. The protocol is able to support 10 kinds of sensors and the selection of these sensors is based on the decisions of the network administrator.

Other than sensor based roles, a sensor node can be assigned a role based on its functionality in the network, such as, cluster-head, gateway or a sink node. We use a three byte array to store the role-set of a node. The mechanism is shown in Fig 1. The bit pattern should be same for the entire network so each node can comprehend the role of other nodes when it receives their role array. In Fig. 1, we assign ten lower significant bits for ten different types of sensors, next ten bits for support roles, one bit for gateway node, one for a sink node and one of a cluster-head node, while one bit is assigned to the leaf nodes.

B. Neighbor Table:

Each node maintains a neighbor table. Nodes use periodic ‘hello’ packets to keep aware of the neighbor nodes and their role. Neighbor table has three fields. Sequence number shows the value for the sequence number of the last packet received from that neighbor node. Sequence numbers are used to avoid stale information. Neighbor table are also used when a path for a VSN is no longer available and a new (or alternate) path is needed to be found.

C. VSN Table:

VSN table is used to maintain information about each VSN to which the sensor node belongs. For each role that is assigned to the sensor node, it maintains the next node in its VSN table, for forwarding data (or configuration) packets. Sequence number for the last packet received from that node and hop count (if known) is also maintained. VSN table has four fields. Sequence number is used to prevent the stale information obtained by a delayed packet, while hop count is used to assist the routing protocol, in case there are multiple paths are available within the VSN.

D. Message Packets

Hello packets are used by sensor nodes to maintain their neighbor table up-to-date. A hello packet contains source identifier of the sensor node which sends the hello packet, a three byte role array of the sensor node which specifies the roles of the source node and a sequence number for the hello packet, which is used to avoid stale and redundant information. When a node tries to create a VSN or connect to the rest of the VSN, it broadcasts a Request (REQ) packet. A Request packet consists of identifier of the source node, role bit array of the source node, sequence number for the Request message by the source node, a time to live value which is specified in hop count and the sequence of intermediate nodes’ identifiers.

When a node receives a Request packet which has enlisted role common to its own role, the node will send a Reply (REP) packet to the source node of this Request packet. A Reply packet contains the identifier of this node and the sender node of the request packet (now the destination node), role bit array of the replying nod, sequence number for the pair of source and destination node, time-to-live for this packet and the sequence of intermediate node (for the source routing of the reply packet). More details of the packet are discussed in section IV.

IV. MECHANISM

A. Hello Packets

A sensor node sends a ‘hello’ packet to its neighbor nodes, using a broadcast, to keep the neighbor table up-to-date. When a node receives a ‘hello’ packet, it checks the source node’s id in its neighbor table, if there is no entry then a new entry is made in the name of the source node, storing its id, its role bit array and the sequence number (obtained in the hello packet). Sequence number is updated with every hello packet, so that the receiving nodes can avoid the stale information. If there exists an entry for the source node in the neighbor table then the receiving node checks for the sequence number (compare the sequence number in the packet to the sequence number stored in the neighbor table), if sequence number shows that the packet is a new packet then if there is a change in the role bit array then the new information is stored and a reply to the hello packet is send, which is also a hello packet but as a unicast. If the information is not new then the hello packet is replied (or acknowledged).

B. Algorithm for VSN Creation

Virtual sensor networks are created in a distributed manner, initiated by sensor nodes to find similar nodes and maintain connection to them. A VSN is initiated when a sensor node broadcasts a request (‘REQ’) packet. This request packet contain field such as source node’s id, its role array, a time-to-live value and a sequence number for request packet from this node. When another sensor node receives a request packet, it checks in its VSN table for an existing entry of the source node. If an entry for the source node already exists then the sequence number is compared with the value in the VSN table. If the received packet is a new packet then the role array is compared with the value in the VSN table else this request packet is ignored. If there is any change, then that change is updated in the VSN table else the packet is ignored.

If an entry for the source node does not exist in the VSN table, then the role bit array of the receiving node and the source node are compared (only the last bytes are compared). If there is no match (not even for a single role) the request packet is updated as: (1) Time-to-live is decreased by one. (2) Node id of the receiving node is appended into the request packet.

Then the request packet is forwarded using broadcasting in the same way as done by the source node. If there is a match (for any role bit of the source node) but not a full match (full match implies a match for every role bit of the source node) then the Request packet is updated as mentioned before and forwarded using broadcasting. However, an entry is also created for the source node enlisting its id, role bit array, sequence number and number of hops in the VSN table. Also a reply ('REP') packet is created as: (1) Receiver node id is stored as Source node id, (2) Source node id (of request packet) is stored as destination node id, (3) Role bit array of the receiving node (of the request packet), (4) Role byte - Intersect of last byte of role bit array of the receiving and source nodes, (5) Sequence number for the receiving node, (6) Hop count is created by counting the appended ids (of the intermediate nodes) plus 1. (7) Intermediate nodes ids are again appended in the reply packet (for source routing)

The reply packet is sent back using a unicast. If there is a full match of role byte (match for every role of the source node) then the VSN is updated and a reply packet is sent to the source node in the similar manner but the request packet is not forwarded.

When a node receives a reply packet, it checks if its own node id exists in the reply packet as an intermediate node. If it does then it stripe of its own id from the reply packet and forwards the packet to the next node. After that the sensor node updates its role bit array as the support VSN of each role present in the role byte in the reply packet. For example, if the role byte of reply packet is 10101011 and the intermediate node role array is xxxxxxxx-uuuuuuuu-yyyyyyyy; then the intermediate node will update its role bit array as xxxxxxxx-1u1u1u11-yyyyyyyy. All the intermediate nodes work in the same way, at last the reply packet is received by the destination node (source node of the request packet). When the destination node receives the reply packet, it updates its VSN according to the values present in the reply packet. A new entry is created for the source node with the role bit array present in the source packet, the sequence number and the hop count.

V. Simulation & Results

We performed the simulation in ns-2. The network model was consisted of 48 (4 x 12 grid) nodes placed within an area of 1000 x 1000 m2. Each node has a propagation range of 150 meters with channel capacity 2 Mbps. The medium access control protocol used is IEEE 802.11 DCF. There are two types of sensor nodes i.e. Temperature and Humidity. Messages are sent to all Humidity sensor nodes only. We increase the ratio of humidity to temperatures sensor nodes from 1:47 to 47:1. We compared our scheme with the flooding mechanism. Packet is sent after 150 ms delay, hence there is packet loss due to collision. We compare both mechanism according to average delay and average messages per nodes. Although the mechanism is designed to support single node with multiple roles, however, in the simulation this functionality was not implemented.

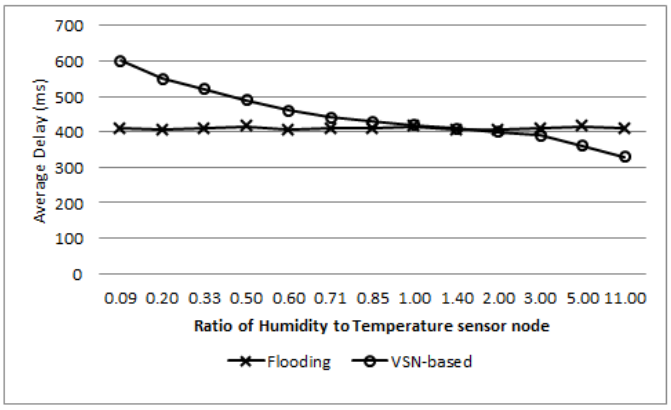

Figure 2. Simulation results for avergare number of messages per node.

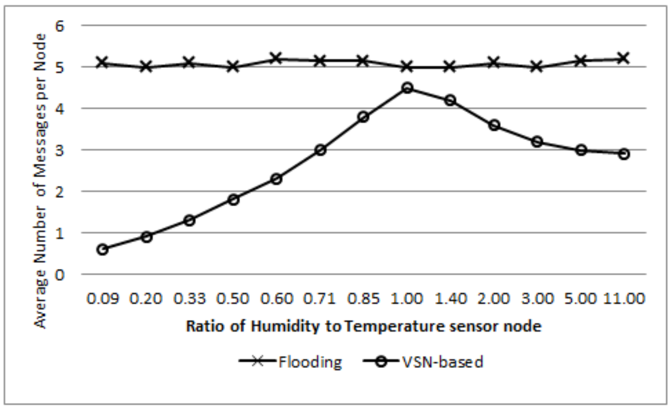

Figure 3. Simulation results for avergare delay.

We calculate the average number of messages required for delivering the packet to each humidity node. We divide the total number of data packets and divide it by the number of humidity nodes. The other metric that we measured is the average delay for all the packets to be delivered. For both metrics we monitor the trend with increasing the ratio of the number of humidity sensor nodes to the number of temperature sensor nodes.

We can observe from Fig 2 and Fig. 3 that the VSN-based system uses less number of packets hence, it consume less power of the sensor network. However, it induces some delay in the VSN-based system. This is due to the fact that flooding makes use of all the available nodes for data delivery while in the VSN based network only VSN nodes are used to deliver data.

According to our theoretical analysis, if we enable multiple roles at a single node then the increase in delay of average number of packet would be directly proportional to the number of nodes (as in the case of nodes without multiple roles). Only difference would be that the node with multiple roles would participate in message delivery of both kinds of VSN (i.e. HUMIDITY & TEMPERATURE).

VI. Application of VSN: Configuration Management

In a sensor network, each node is required to be configured according to the application it is running or reconfigured to optimize its performance. For this, each Sensor Node is armed with a Configuration Agent, which is responsible for providing the application and node configuration to and from the Configuration Manager (at the sink node). It also receives the configuration from the sink and applies them on the sensor node.

However, in a practical scenario, not all of the sensor nodes are required to be reconfigured at a time. In fact only certain nodes are needed to be updated with new configuration or with on-the-fly code updates. These updates are directed towards the nodes with similar applications or functionalities. For example, if a policy is changed for the temperature sensors with a new threshold for alarm; then only the nodes with the temperature sensors in it are needed to be updated. If we use the conventional scheme then the cost of updating or reconfiguration is very high and not feasible. This is because typically WSN uses flooding based protocols which require lot of messages and the communication has a huge cost in WSN. Hence, we utilize the concept of VSN to optimize the reconfiguration mechanism in WSN.

In a heterogeneous WSN, there can be more than one VSN. Hence, we provide configuration mechanism for the sensor nodes, according to the application that are running on the sensor nodes. The sink provides the basic configurations to each node according to the VSN it belongs to. The different colors (shades) of the sensor nodes imply the different VSN each sensor node belongs to. Now as shown in the simulation section, if one of the VSN is needed to be configured such that its attributes needed to be changed then a VSN based WSN performs better than a conventional WSN in terms of energy consumption.

Our Configuration management system based on VSN is consisted of two sub-systems, as shown in Fig 7. One is the server at a PC or Notebook, which is the configuration manager working as the sink node of the sensor network. The second sub-system is the configuration manager which resides and manages the configuration agents at the sensor nodes. Configuration Manager is responsible for applying the configuration parameter at the sensor node received from the Configuration Manager at the sink node.

Configuration Manager at the sink node is responsible for creating the configuration for the VSN nodes, and their applications. It also interacts with the user for changing the configuration within the sensor nodes using the Policy manager. Configuration manager is also responsible for assessing the configuration of the nodes such that basic requirements of the network are not affected by the change in configuration. After assessing the changes as correct, it notifies the sensor nodes about the new values.

Configuration manager has 3 important modules (shown in Fig. 4). Configuration Module is the main module which creates the configurations for each sensor node according to the application it is running. Transceiver module is used to communicate with the WSN nodes for maintaining VSN information (as sink node is part of every VSN) and sending new configurations to the VSN nodes. The Policy manager module is used to maintain policies which define the new configurations. The Configuration Manager makes reconfiguration decisions for the sensor node based on policies defined in the system. These policies are defined by a network manager or based on intuition & intelligence. The decision may be automated or monitored by a network manager. The automation of decision making is another study and out of the scope of this paper.

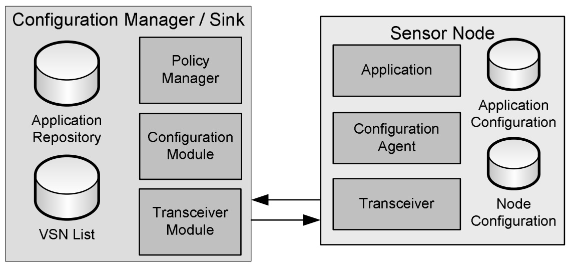

Figure 4. Configuration Management: Information Flow and Architecture of the system

VSN list is the directory of all the nodes with information about their application and the current configuration parameters. Application & Configuration repository contains the list of each application that is being run at the sensor nodes, along with its configurations.

The communication between the Configuration Manager at sink node and the configuration manager at the sensor node is very simple and based on the configuration agents. The Configuration agents have simple function calls.

VII. Conclusion

In this paper we have presented a VSN creation mechanism. We have defined the ingredients of enabling application-aware VSN in WSN. VSN creation is a self-organizing mechanism, which can be initiated by any node and works in a distributed way. Through simulation results, we can conclude that use of VSN significantly reduces the number of packets used to deliver data hence, saving battery life of WSN nodes. Using Configuration Management as an application scenario we support our claims. Configuration Management enable dynamic Virtual Sensor Networks for efficient and resource aware management. In the future, our goal is to design a framework for the WSN management around this mechanism.

References

[1] I. F. Akyildiz and I. H. Kasimoglu, “Wireless sensor and actor networks: research challenges”, Journal of Ad Hoc Networks (Elsevier), vol. 2, no. 4, Oct. 2004, pp: 351-367.

[2] Hill, J., Horton, M., Kling, R., Krishnamurthy, L., “The Platforms Enabling Wireless Sensor Networks,” Communications of the ACM, Vol. 47, No. 6, June 2004.

[3] Anura P. Jayasumana, Qi Han and Tissa H. Illangasekare, "Virtual Sensor Networks - A Resource Efficient Approach for Concurrent Applications" in International Conference on Information Technology ITNG 2007, pp: 111 - 115.

[4] H.M.N. Bandara, A.P. Jayasumana, and T.H. Illangasekare, "Cluster Tree Based Self Organization of Virtual Sensor Networks", in IEEE GLOBECOM Workshops, 2008, pp: 1 - 6.

[5] R. Tynan, G.M.P. O'Hare, M.J. O'Grady and C. Muldoon, "Virtual Sensor Networks: An Embedded Agent Approach" in IEEE International Symposium on Parallel and Distributed Processing with Applications, 2008, pp: 926 - 932.