# Application-aware Virtual Networks in WSN

Muhammad Shoaib Siddiqui and Choong Seon Hong
Department of Computer Engineering, Kyung Hee University
shoaib@networking.khu.ac.kr, cshong@khu.ac.kr,

## Abstract

Heterogeneity in WSN is common with multi-functional sensor nodes constituting a network. On the fly code updates, dynamic configuration and application awareness are the needs of the hour for the near future. We have introduced the concept of Virtual Sensor Networks (VSN), which creates application-aware virtual network for better management, configuration and resource efficiency. In this paper, we propose protocols and mechanism for creation and working of VSN. Through simulations we support the concept and show that the proposal can achieve better results than ordinary WSN.

## 1. Introduction

Wireless Sensor Network (WSN) [1] is a large scale distributed wireless network consisting of self-organizing sensor nodes which monitor the environment for a certain phenomenon. Sensors attached to sensor nodes create sensed data, which is aggregated by the sensor node and delivered to a single (sometimes more than one) destination i.e. called a Sink. The size of the sensor nodes is very small [2] and have fewer resource requirements, which makes their availability relatively inexpensive compared to other network devices.[*]

Due to their characteristics, WSN is a critical part of today's monitoring and surveillance systems. WSN has a promising potential in the future; from monitoring an environment to disaster management and from simple configuration to autonomous management; each field would depend upon sensor networks for gathering data and information and providing cheap execution at remote areas of interest.

WSN are often associated with scarce resources 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 rest of the paper is articulated as follows. Section 2 gives related work in the field of VSN. Section 3 provides describes the VSN concept and the ingredients of the VSN creation and maintenance protocol. Section 4 gives the mechanism. Section 5 provides simulation results and section 6 concludes our work.

## 2. 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.

## 3. 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

[*] This research was supported by Future-based Technology Development Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (No.20100020728).
Dr. CS Hong is the corresponding author.

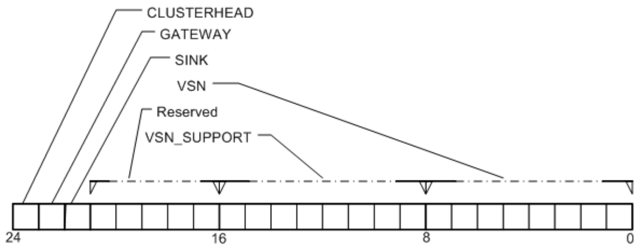

Figure 1. Role bit array scheme with role bits, support-role bits and reserved bits.

application running on them. For making the WSN application-aware, 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.

## 3.1. 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. For example, we have sensor nodes which have multiple sensors, such as Hanback ZigBee Mote, which can have a temperature sensor, a humidity sensor, an RF sensor and a photo-sensor (light sensor) at a same time. So there can be 4 roles that such a sensor node can have, such as TEMPERATURE, HUMIDITY, RF, and PHOTO (light sensing).

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.

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 eight lower significant bits for eight different types of sensors, next eight bits for support roles, one bit for gateway node, one for a sink node and one of a cluster-head node, while 5 other bits are reserved for future use or to be used by the user.

## 3.2. 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 as shown in Table 1. 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.

Table 1. Neighbor table fields and their descriptions.

| Field | Description |
|---|---|
| Node ID | Identity of the neighbor node. |
| Role | Role array of the neighbor node |
| Sequence Number | Sequence number in the last message received from the neighbor node |

## 3.3. VSN Table:

VSN table is used to maintain information about each VSN to which the sensor node belongs to. 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 as shown in Table 2.

Table 2. VSN table fields and their descriptions.

| Field | Description |
|---|---|
| Node ID | Identity of the next node in VSN. |
| Role | Role array of the node |
| Sequence number | Sequence number in the last message received from the neighbor node |
| Hop count | Hop count value for the node |

## 3.4 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

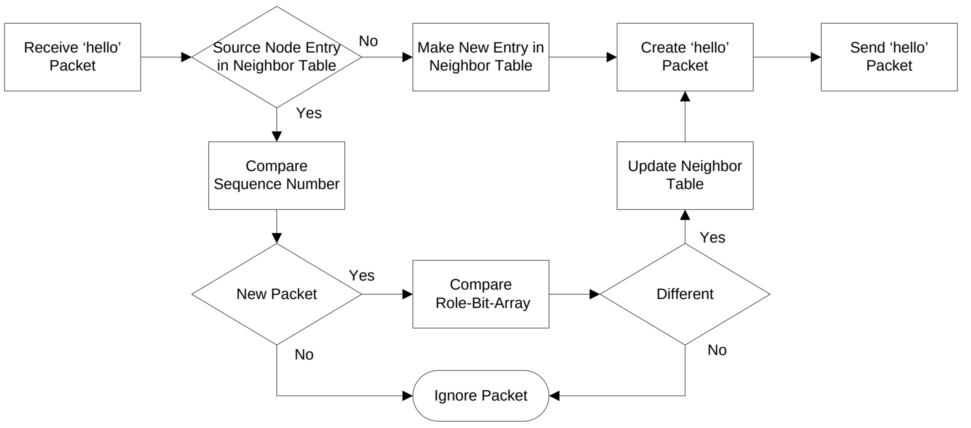

Figure 2. The mechanism followed by a node when it receives a 'hello' packet.

intermediate node (for the source routing of the reply packet).

## 4. Mechanism

### 4.1. 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). The flow of the mechanism is shown in Fig. 2.

### 4.2. 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. The flow of the mechanism is shown in Fig. 3.

If an entry for the source node does not exist then the role bit array of the receiving node and the source node are compared (only the last bytes are compared). If we have a total match, then an entry is 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 partial match then the VSN is updated in a similar manner, however, 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 and a reply packet is sent using unicast. If there is no match in the role bit array then VSN is not updated but the request packet is forwarded in a similar way.

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 (which is enlisted in the reply packet). 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

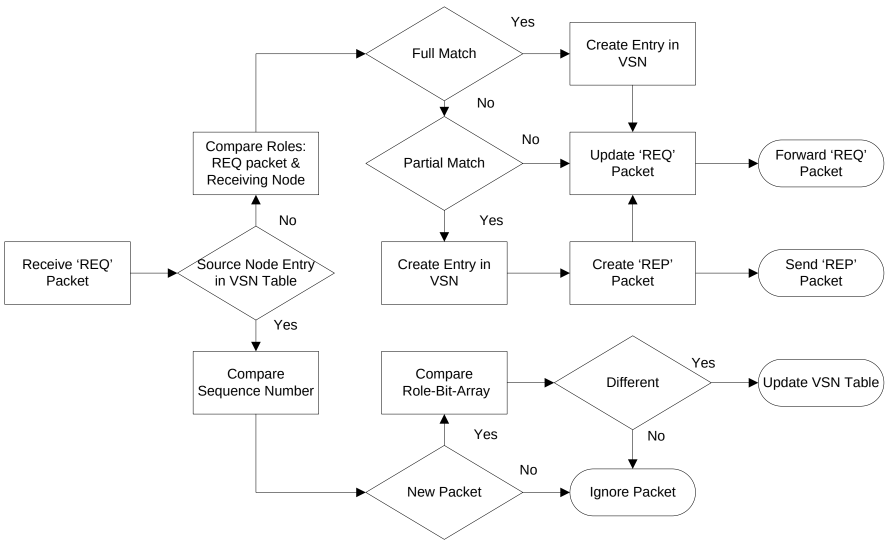

Figure 3. Mechanism followed by a node after receiving a VSN Request (REQ) packet.

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.

## 5. Simulation & Results

We performed the simulation in NS-2 [6]. The network model was consisted of 48 (4 x 12 grid) nodes placed within an area of 1000 x 1000 m$^2$. 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.

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 4 and Fig. 5 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.

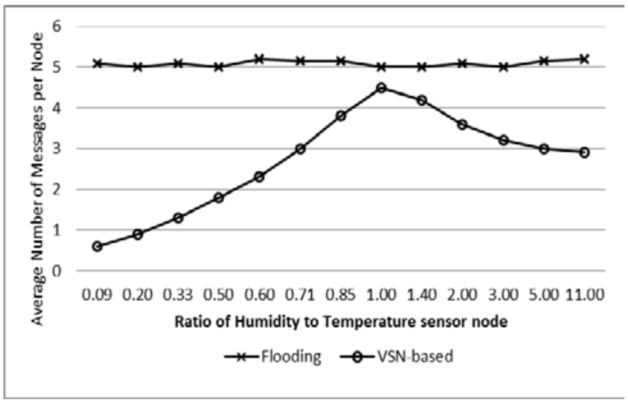

Figure 4. Simulation results for avergare number of messages per node.

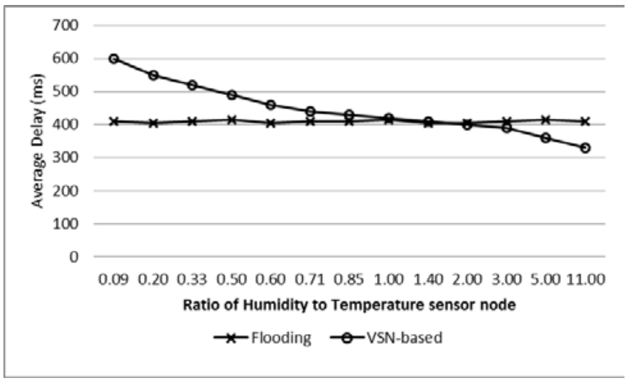

Figure 5. Simulation results for avergare delay

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 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).

## 5. 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. VSN based

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.

## 5. 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.

[6]. “UCB/LBNL/VINT Network Simulator - ns 2” URL: http://www.isi.edu/nsnamjns.