# A Review of Distributed Scheduling Algorithms for Tree based Wireless Sensor Networks

Dr. Tejas Vasavada
Assistant Professor,
Lukhdhirji Engineering College, Morbi, India
and
Dr. Sanjay Srivastava
Professor,
Dhirubhai Ambani Institute of Information and Communication Technology, Gandhinagar, India

In this paper, distributed scheduling algorithms for data collection in tree-based Wireless Sensor Networks (WSNs) are reviewed. The algorithms are categorized based on type of convergecast addressed by them i.e. (i) Aggregated convergecast (ii) Raw convergecast (iii) General approaches. In aggregated convergecast scheduling algorithm, one slot per node is selected as all incoming packets are fully aggregated with packet of the given node. So, every node sends out only one packet. In raw convergecast, as packets are not aggregated, every node needs multiple time-slots. It is desired that algorithms for aggregated convergecast should be bottom-up in nature (i.e. run from leaf nodes towards the sink) to ensure aggregation freshness. In raw convergecast, the algorithms should be hybrid in nature. It means that the execution should take place in bottom-up and also in top-down manner. This ensures that every node transmits its own packet in the smallest possible time-slot and forward packets of children in the same TDMA cycle. The General approaches are not designed for any fix convergecast method, but they have other objectives like minimizing control overhead, minimizing schedule length or minimize energy consumption and many others. This work also presents a review of algorithms related to fault tolerance. The algorithms related to fault tolerance are aimed at quickly selecting new parent/slot when some existing parent dies. When the given node dies, the parent of the given node does not receive packets from the given node. So, the parent needs lesser time-slots as it has to forward lesser number of packets. Similarly, when the given node selects new parent due to death of current parent, the new parent needs extra slots to forward packets coming from the given node. Thus fault tolerance algorithms also take care of schedule adjustment due to change in workload of nodes. It is found that still there is scope of further research as follows: (i) Hybrid joint scheduling & tree formation algorithm for raw convergecast can be designed. (ii) The slot assignment should be elastic in nature. The given node should be assigned additional slots when required and slots should be revoked when not needed. (iii) The scheduling algorithm should have some provision of priority-based data transmission. When a node has urgent or high-priority data, it should be allowed to transmit it without waiting for its transmission turn.

Keywords: Sensor Networks,Tree Formation,Scheduling,Distributed Algorithms

## 1. INTRODUCTION

Wireless sensor nodes are very small electronic devices with capability to sense the environment. In addition to sensing, they are capable to store and process the information. The sensor nodes are deployed in an area where some quantities are to be observed. The quantities of interest are like pressure, temperature, humidity, solar radiation and many others. In last few years, sensor networks are deployed for many real applications. Some of them are like environmental research (Selavo [2007],Barrenetxea [2008]), volcano monitoring (WernerAllen [2006],W.Z.Song [2009]), water monitoring (Kim [2008]) and weather monitoring(Hartung [2006]) .

Every sensor network has one or more sink nodes present. All the sensor nodes send their observations towards the sink node. Often nodes are randomly deployed in the region. So, routing path is required from each node to the sink. Thus it is required to form some logical topology. In McGrath [2014] and Mamun [2012], various logical topologies are explained along with their comparison. Following are the well-known logical topologies: (i) Flat (i) Chain (iii)

Star (iv) Ring (v) Tree (vi) Cluster.

Flat means no logical topology. The basic technique used for data transmission is flooding. Due to flooding, nodes receive duplicate messages. As a result, more energy is consumed. As sensor nodes are energy-constrained devices, flooding affects lifetime of the network. So, flat topology is not a good choice. It is also difficult to predict the latecy as path is not fixed.

In Figure 1, chain topology is shown. The nodes are arranged in a linear chain (i.e. bus). The sink node is at one end of the chain. It is denoted as $S$. Every node sends packet to the next node towards the sink. The node receiving the packet forwards the packet to the next node. Thus finally packet reaches the sink. If any one node in the chain fails, network is partitioned into disconnected subsets. So, nodes which are disconnected from the sink can not send data to the sink. Thus chain topology has very low fault tolerance.

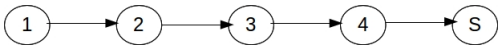


Fig. 1: Chain Topology

In Figure 2, star topology is shown. The star topology has sink in the center. All the nodes send data to the sink in one hop. The star topology seems less scalable as all the sensor nodes need to be at one hop distance from the sink. Its variants like cluster and tree (explained later) are more scalable. Ring topology is illustrated in the Figure 3. In case of ring topology, sensors and sink are arranged in the form of ring. The data transmission takes place in circular fashion. Every node receives packets from neighbor on one side and sends to the neighbor on the other side. Like chain, if any one node fails, network is partitioned. Thus ring topology also has low fault tolerance.

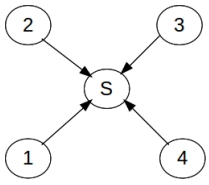


Fig. 2: Star Topology

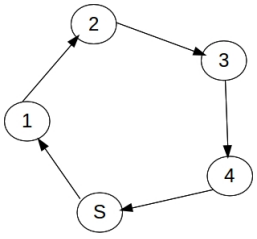


Fig. 3: Ring Topology

In the tree topology, every node has one parent and number of children nodes. The sink is the root of the tree. Every node sends packets towards the sink through parent node. One sample tree is shown in Figure 4. In a tree, every node receives packets from its children and may aggregate all incoming packets with its own packets and send a single packet to the parent node. The task of aggregation is possible in chain and ring also. But in both of them, every node receives packet from only one node. Whereas in tree, node may receive packets from multiple nodes. Thus tree seems most suitable topology for applications which require data aggregation. In case of chain or ring topology, maximum hop distance from sink to the the last node (i.e. leaf) is $(n-1)$, where $n$ is number of nodes in the network. Whereas in a balanced tree, it is on the order of $\log_2 n$. Thus packet latency is likely to be less in tree compared to chain or ring topology.

In Figure 5, cluster topology is illustrated. The network is divided into small regions. Each region is known as cluster. Each cluster has one node named as Cluster-Head (CH). All the nodes around the CH are cluster members. All the members send their packets to the CH. The CH may aggregate the incoming packets and send out a single packet. Each CH may be directly

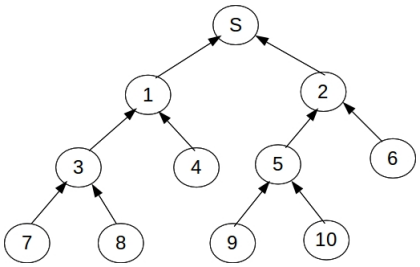


Fig. 4: Tree Topology

connected to the sink. Alternatively, second level of clustering may be formed such as CHs are members and they send data to their cluster-head.

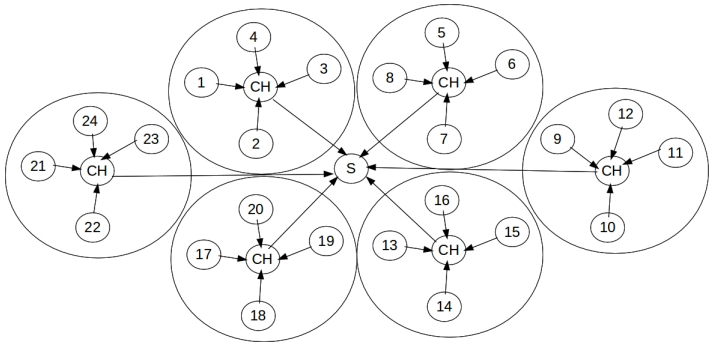


Fig. 5: Cluster Topology

The cluster members can be either at one-hop distance or multi-hop distance from the CH. If member nodes are at single hop distance from CH, the topology inside the cluster is a star topology with CH as the center node. If nodes are at multi-hop distance, the topology inside the cluster is a tree with CH as the root. Even in the tree, we can think that multiple star topologies are present because a node may be connected to multiple children nodes. Thus the given node is in center node of star topology. This is why it is mentioned earlier that tree and clusters are extensions of star topology.

From above discussion on topologies, it is concluded that flat topology is not useful in practice because of its poor performance with respect to network lifetime. The tree topology is more suitable for applications which need aggregation and also results in smaller latency than chain and ring topologies. Even a cluster can be considered as a tree. Thus due to robustness of the tree, it is a very popular topology. Our focus is also on tree-based networks.

Tree formation can take place in either centralized or distributed manner. The works presented in O.D.Incel [2012], Soua [2013], Pan [2008], Malhotra [2011], Ghosh [2010] and Hia [2011] are some examples of centralized algorithms. In case of centralized approach, sink runs tree formation and slot assignment algorithms. Then it propagates parent and slot information to all the nodes. So, sink must know entire topology. If deployment is regular, topology information may be fed to the sink. But nodes are randomly deployed in many practical applications. Thus it is required that nodes have to send their positions to the sink node as input to scheduling and tree formation algorithms. Thus every node should have GPS (Global Positioning System) installed. An alternate is to run localization algorithms.

As explained in subsequent sections, when distributed approach is used, every node performs parent and slot selection itself. When a node wants to select new parent due to failure of existing

parent, it can perform parent (and also slot) selection itself in case of distributed approach. But, if pure centralized approach is used, sink has to find new parent. Thus tree repairing and schedule maintenance need to be done centrally. The distributed approach results in faster tree repairing and schedule maintenance compared to the centralized approach as decision is taken locally.

It is easy to implement centralized algorithms. The centralized approach considers the network as a unit disk graph. So, it fails to take into account effects of real-time interference. In contrast, in distributed algorithms, every node selects a parent and slot taking into account its local neighborhood. So, the decision also takes into account real time interference. As distributed algorithms seem more appropriate considering decentralized nature of sensor networks, our focus is also on distributed parent selection and slot assignment.

In Ghosh [2011] and M.Bagga [2014], many scheduling algorithms for tree based sensor networks are reviewed. But they have covered mostly centralized approaches. In M.Bagga [2014], only aggregated convergecast scheduling is considered. Here, we are going to present algorithms for both i.e. aggregated and raw convergecast. As tree repairing and schedule maintenance both should be done effectively, we have also reported papers addressing both. We could not find any paper which reviews distributed scheduling algorithms along with approaches to recover from node failure. So, this review paper seems to be the first of its kind.

## 2. DISTRIBUTED SCHEDULING ALGORITHMS

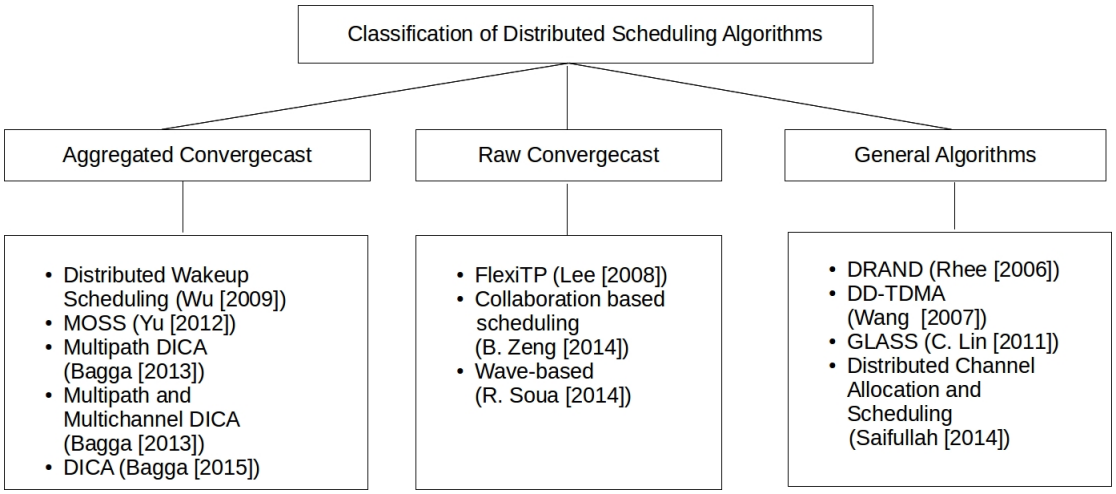


Fig. 6: Classification of Distributed Scheduling Algorithms

The basic objective of every scheduling algorithm is to assign one or more time slots to edges or nodes of tree. When a node selects a transmission slot, the slot should be such that data transmission done during that slot does not collide with packets received by neighboring nodes. Thus before selecting a transmission slot, node should determine which slots are already used by its neighbor nodes. The papers differ in the proposed method of finding collision free schedule. Otherwise their objective is same i.e. to select a transmission slot which does not create collisions.

In Figure 6, classification of distributed scheduling algorithms is given. There are three main categories: (i) Those designed specially for aggregated convergecast (ii) Those designed specially for raw convergecast (iii) General approaches i.e. designed neither for aggregated convergecast nor for raw convergecast. But adaptable to any of them.

In aggregated convergecast, parent node aggregates packets coming from children with its own packet. So, every node is assigned one transmission slot. If a node has $n$ children, it has $n$ reception slots and one transmission slot. Parent is assigned higher slot than children so that it

can aggregate packets of children with its own packet and forward in the same TDMA frame. This reduces data delivery delay. One way of assigning higher slot to parent is to do bottom-up scheduling. That is, scheduling should start from leaf nodes and continue till sink.

In raw convergecast, parent node does not aggregate packets coming from children with its own packet. Every node is assigned one slot to transmit its own packet and additional slots to transmit packets coming from its children. Thus if a node has $n$ children, it needs $(n + 1)$ transmission slots and $n$ reception slots. Here parent node can be assigned slot earlier than its children. As data is not aggregated, parent node should send its own packet as soon as it is generated. This would result in better data delivery delay.

The methods presented in DRAND(Rhee [2006]), GLASS(C.Lin [2011]), Distributed Channel Allocation and Scheduling (DCAS) (Saifullah [2014]) and DD-TDMA(Wang [2007]) are general approaches of slots selection. Even they are not designed specifically for tree structure. As mentioned earlier, packet delivery delay in aggregated convergecast can be minimized if parent is assigned higher slot than children. Raw convergecast requires every node to select more than one slots. As aggregation is not done, parent node can select a lower slot than children to forward its own packet. These methods do not have such features. So, they are not directly applicable to aggregated or raw convergecast. But they may be adapted for either of the two. Their objectives are like reducing control overhead of slot selection (GLASS (C.Lin [2011])) or to reduce the schedule length (DD-TDMA (Wang [2007]), DCAS (Saifullah [2014])).

### 2.1 Aggregated Convergecast

In this subsection, some important papers attempting scheduling for aggregated convergecast are summarized. At the end of the subsection, a qualitative comparison is also given.

In Wu [2009], distributed algorithms for tree formation and slots assignment are proposed. Tree formation works as follows. The sink initiates the process by flooding FORM_TREE message in network. Upon receiving FORM_TREE, a node periodically broadcasts HELLO message to its 2-hop neighbors for some time. The HELLO message contains followings: (i) 1-hop neighborhood of sender (ii) parent node ID, if selected. So nodes receiving this message can know about interference at transmitting node.

From candidate parents (i.e. those who have already joined the tree), node chooses the parent which is nearest to sink. If more than one candidate parents are at the same depth, the one with the least interference is selected as parent. Selecting the least interfered node results in better slots reuse. After selecting parent, node unicasts JOIN_TREE message to sink. When sink receives JOIN_TREE from all the nodes, it considers that tree formation is complete. It initiates scheduling next. Scheduling works as explained below.

The sink floods the network with ASSIGN_SLOT packet. The sink selects slot $K-1$ as its sending slot. Here $K$ is the largest slot in frame. Then periodically broadcasts GET_SLOT(sink,K-1) to its 2-hop neighbors for some time. When the given node $v_i$ receives a GET_SLOT message from parent, it selects a slot $s_i$ one less than that selected by parent. This slot is tentative. It broadcasts ($v_i$,$s_i$ ) to its 2-hop neighbors and starts a timer.

If some neighbor has already selected the same slot, it sends REJECT message. Upon reception of REJECT message, node selects a time slot one less than previously selected slot and repeats the procedure i.e. broadcast to 2-hop neighbors. If given node does not receive REJECT by the time timer expires, it finalizes the slots $s_i$ and informs the same by broadcasting GET_SLOT($v_i$,$s_i$) message to 2-hop neighbors. Now children of $v_i$ will continue the same procedure. This procedure will continue until all the nodes including leaf nodes select their slots.

In Yu [2012], joint tree formation and slots assignment is proposed. It is known as MOSS (Many to One Sensors to Sink) communication protocol. In joint approach, parent and slot selection takes place at the same time. So, one algorithm does both i.e. tree formation and scheduling.

Sink initiates the process by broadcasting PADV (Parent Advertisement) message. Its one hop neighbors receive the message. PADV contains the time slot selected by sender. Nodes receiving PADV keep record of parent's id and parent's slot. Then these nodes send SEL (Select) message

| Approach | Paper | Parent selection criteria |
|---|---|---|
| Sequential | DWS (Wu [2009]) | Interference |
| Joint (top-down) | MOSS (Yu [2012]) | Distance from given node |
| Joint (bottom-up) | DICA (Bagga [2015])<br>Multipath DICA (Bagga [2013a])<br>Multipath Multichannel DICA (Bagga [2013b]) | No. of unscheduled Neighbours |

Table a: Summary of Aggregation Convergecast Scheduling Algorithms

back to sink (i.e. parent) indicating that sink is selected as parent. SEL message also contains the time-slot selected by child node. The sink (i.e. parent) sends SCH (Schedule) message in reply. From SCH message, child comes to know whether its slots selection is successful or not.

If selection of slot is not successful, child has to try again. If successful, child node broadcasts PADV message and whole procedure is repeated. Now child becomes parent of some other nodes. A node may receive multiple PADV messages. That is, it may have more than one candidate parents. It selects the node as parent which is nearest to the sink in terms of hop count. If more than one candidate parents have same hop count, selection is done based on distance between given node and candidate parent node. The nearest candidate is selected as parent. This would save transmission power. At the end, all the nodes including the leaf nodes are scheduled.

In Bagga [2015], a distributed joint bottom-up tree formation and scheduling algorithm is proposed. It is named as DICA (Distributed algorithm for Integrated tree Construction and data Aggregation). As the first step, network is divided into levels. A node is in level L if it is L hops away from sink. Every node has information about its level L and neighbors in level L-1, L and L+1. Once leveling is done, parent selection and schedule assignment takes place in bottom-up manner. That it starts from leaf nodes and continues till the sink.

Every node at Level L is allowed to select parent nodes from level L-1, L or L+1. Node selects such a parent to whom it can transmit in lowest possible time slot. For a time slot T, node prepares a candidate parent set. It consists of nodes for whom following is true: (i) they don't receive in slot T (ii) they don't transmit in slot T (iii) they don't overhear from other nodes in slot T.

From the candidate parent set, the node with the minimum number of unscheduled neighbors is selected as parent. This selection criteria is aimed at maximizing slots reuse. Parent is scheduled only after all its children are scheduled. As parent has to aggregate children's data with its own data, it should wait until children send their data.

The ideas presented in Bagga [2013a] and Bagga [2013b] are like extensions of work presented in Bagga [2015]. In Bagga [2015], single channel is assumed and only one path is established from every sensor node to sink node. In Bagga [2013a], multiple paths are established from every sensor to sink. This results in better fault tolerance and load balancing. In Bagga [2013b], data aggregation scheduling utilizing multiple paths as well as multiple channels is done. Use of multiple channels reduces schedule length as interfering nodes can use same slot at different frequency.

2.1.1 *Discussion.* In the Table a, comparison of the three aggregation convergecast scheduling algorithms is presented. It is seen that they differ in the criteria of selecting the parent. The DICA(Bagga [2015]) and MOSS(Yu [2012]) are joint approaches. That is, every node selects parent and slot at the same time. In contrast, DWS(Wu [2009]) does parent and slot selection in two different phases, first parent selection followed by slot selection. As explained in DICA(Bagga [2015]), joint approach is better than sequential approach. In sequential approach, tree is formed first. Then scheduling algorithm is applied to the same tree. Thus the structure of the tree controls the performance of the scheduling algorithm. But in joint approach, tree and schedule are built hand in hand. So, there is no question of tree structure controlling the performance of

scheduling algorithm.

Among the three algorithms, the bottom-up approach is the most suitable. As mentioned earlier, to maintain aggregation freshness, scheduling in aggregated convergecast must take place in bottom-up fashion. That is, parent should be assigned higher time slot than children. In DWS(Wu [2009]), parent is assigned higher slot than children. It is not perfect bottom-up approach. As mentioned in Ren [2012], many times networks are heterogenous in nature. Different types of sensors are present in the same network. For example, temperature, pressure, humidity and solar radiation sensors are present in the same region.

All these different types of sensors are part of the same tree. In that case, perfect aggregation may not be feasible at every node. Let us assume that a temperature sensor is a parent of two different sensor nodes, one pressure sensor and the other temperature sensor. The parent can aggregate temperature reading coming from the child with its own reading. But, it can not aggregate pressure reading with the temperature reading. So, two packets would come out from the parent node i.e. one pressure packet and the other is temperature packet. If all the three nodes were temperature sensors, only a single packet would come out of the parent.

Thus in heterogeneous network, a node should be scheduled only after its children are scheduled. Otherwise, it would not estimate the number of slots required. The MOSS(Yu [2012]) is not suitable in this case as it is top-down. In addition, DWS(Wu [2009]) would not be efficient because when a node selects a slot, all its ancestors may be required to select additional slots. The DICA(Bagga [2015]) is likely to work well for both i.e. homogeneous and heterogeneous networks as it is bottom-up in nature.

### 2.2 Raw Convergecast

Some important papers addressing raw convegecast are summarized in this subsection along with their qualitative comparison.

In Lee [2008], FlexiTP (Flexible TDMA Protocol) is proposed. It handles tree formation, scheduling and fault tolerance. Fault tolerance part is mentioned in Section 3. Tree formation is initiated by sink. It generates a token. Token passes through tree in Depth First Search manner. When a node receives a token, it broadcasts a signal. All nodes who receive this signal become children of the node. Then node forwards the token to child whose ID is lowest. Every node repeats this procedure. Finally token reaches to sink after all the nodes find their children.

Once tree formation is complete, sink initiates slots assignment. It also involves use of tokens. When a node receives a token, it selects its transmission slots. A node selects slots in such a way that its transmission does not interfere with two-hop neighbors. After claiming a slot, node shares its slot information with one and two hop neighbors. When a node selects a slot, its parent also requires a slot to forward node's data. Thus selection of a slot by a node leads to slots selection by all the nodes till sink.

In B.Zeng [2014], collaboration based distributed scheduling algorithm is proposed. It is assumed that tree is already formed using some known algorithm. Basic idea behind slot selection is as follows. Node knows its total workload. It is sum of its own workload and that of its children. Based on workload and available time slots, node selects a slice. Here slice is a sequence of slots for transmission. Then it broadcasts a request packet. The request packet contains slice. All the neighbor nodes receive request packet. Every neighbor sends a reply packet. The reply packet either grants the requested slice or rejects it. If transmission of data packet during requested slice does not create collision, neighbor node grants the request. Else it rejects the request. If request is granted by all neighbors, node fixes the slice as its transmission slots. It also broadcasts an acknowledgement (ACK) message as confirmation. If any one neighbor node rejects the request, node does not send ACK.

In R.Soua [2014], scheduling algorithms for raw convergecast are proposed. It is assumed that nodes can use more than one channels for transmission. Two versions of same algorithm are proposed : centralized and distributed. The aim is to reduce schedule length and data generated by a node should reach the sink in same cycle. It is considered that different nodes have different

| Approach | Paper | Parent Selection criteria |
|---|---|---|
| Hybrid | FlexiTP (Lee [2008]) | Lowest ID |
| Bottom Up | Collaboration based (B.Zeng [2014]) | Not addressed |
| Top Down | Joint Channel & Slot Assignment (R.Soua [2014]) | Not addressed |

Table b: Summary of Raw Convergecast Scheduling Algorithms

traffic requirements. Accordingly different frames have different number of slots active. Each frame is known as a wave.

Centralized algorithm works as follows: sink has idea about traffic load of each node and entire topology. It sorts nodes in descending order of number of slots required to transmit their data. Nodes are processed for slot and channel assignment in order of priority. Every node is assigned lowest available channel and slot for transmission in given frame. As channels are limited, a node can not send all the packets in a single frame. Thus multiple waves are required. Sink calculates slot and channel allocation for each wave.

Distributed approach works as follows. Nodes exchange hello packets to know about two hop neighbors and their traffic load. First wave is computed by sink as sink has information about entire topology and traffic available at each node. Sink forwards the first wave to nodes. Thus every node knows its allocated channel and time slot for first wave. Rest of the waves are computed by individual nodes locally.

2.2.1 *Discussion.* In Table b, the three raw convergecast scheduling algorithms are compared. In Collaboration based scheduling (B.Zeng [2014]) and Joint channel and slot assignment (R.Soua [2014]) algorithms tree is assumed to be present. The FlexiTP(Lee [2008]) first forms the tree and then scheduling is done. This means that all the three algorithms execute scheduling algorithm on the existing tree. As mentioned earlier, joint slot and parent selection would result in smaller schedule length. Every node selects a single parent. Packets are transmitted to the same parent in multiple slots. If joint approach were used, multiple parents may be selected to confirm data transmission in the smallest slot

In raw convergecast, bottom-up slot assignment is not desirable. As aggregation is not done, given node need not wait for its children to send the packets. It could transmit earlier than children. That is, it may use smaller slot. Even pure top-down approach is also not suitable. As mentioned in R.Soua [2014], node knows it total workload i.e. rate at which packets are coming from children. So, node could select the required time-slots. In that case, the node is assigned smaller slots than its children. It could not forward incoming packets in the same cycle.

It is better to use hybrid approach like in FlexiTP(Lee [2008]). In FlexiTP, every node selects time slot to transmit its own packet. But when a node selects a time-slot, all its ancestors select additional time-slots to send the packets coming from the given node. Thus complete scheduling involves multiple top to bottom and bottom to top slot assignments. That is why we call it 'hybrid'. It has two advantages. First, every node could send its own packet as early as possible. Second, packets coming from descendants can be forwarded in the same TDMA cycle.

In FlexiTP, every node sends the token to the child with the lowest ID. Now that child gets a chance to expand its subtree. So, it finds its children and becomes parent of some nodes. That is why, parent selection criteria is written as 'lowest ID' in Table b. As the other two algorithms assume that tree is already formed using some algorithm, it is concluded that they do not address parent selection.

In collaboration based algorithm (B.Zeng [2014]) and Joint slot and channel allocation algorithm (R.Soua [2014]), every node selects all required slots in one go. So the control overhead would be less than Hybrid approach of FlexiTP(Lee [2008]. As a result, energy consumption would also be reduced. The same thing is verified through simulation results of B.Zeng [2014]. Thus this a trade off between control overhead and latency. In FlexiTP(Lee [2008]), due to hybrid

approach, packets of every node would reach to the sink in the same TDMA cyle. But control overhead would be high compared to the other two approaches. In collaboration based approach, node's own packets would be delayed. In case of joint scheduling and tree formation (R.Soua [2014]), packets would reach to sink in the next cycle.

### 2.3 Miscellaneous Approaches

In this subsection, generalized methods of scheduling (i.e. not designed for a specific convergecast) are reviewed along with their qualitative comparison.

In Rhee [2006], DRAND (Distributed RANDomized Algorithm) is proposed. Every node selects its own time slot. Time slots selection requires 3-way message exchange. A node willing to select slot first broadcasts REQUEST message. All its neighbors receive this message. In response, every neighbor sends GRANT message back. GRANT message contains id of sender and list of time-slots used in one hop neighborhood of sender. Once requesting node receives GRANT messages from all neighbors, it has an idea about which slots are used in its two-hop neighborhood. It selects the minimum slot not used in its two-hop neighborhood and broadcasts RELEASE message. It contains id of sender and the slot selected. Thus all neighbors of given node update their list of used slots. Message complexity of DRAND is on the order of number of two hop neighbors of given node.

In Wang [2007], DD-TDMA (Deterministic Distributed TDMA) scheduling is proposed. It is proposed to minimize the distance between transmission and reception slots of every node. If transmission and reception slots of a node are nearby, it is not desirable to keep the node in sleep mode in intermediate slots. It is better to keep the node in idle mode than to put into sleep because frequent on/off switching consumes more energy than idle listening. Scheduling algorithm similar to DRAND is proposed. It is shown through simulations that DD-TDMA results in lesser number of slots than DRAND. Its running time and message complexity both are lesser compared to D-RAND.

In Saifullah [2014], Distributed Channel Allocation and Scheduling protocols (DCAS) are proposed. As per IEEE 802.15.4 standard, a node can operate over multiple channels. Sensor nodes are tiny. Every node normally contains one radio transceiver. So it can operate on one channel at a time. Distributed algorithms for Receiver Based Channel Assignment (RBCA) and Link Based Channel Assignment (LBCA) are proposed.

The general approach followed by RBCA and LBCA is to form different conflict graphs (i.e. one for RBCA and the other for LBCA). In both the conflict graphs, nodes are presented as vertices. In conflict graph of RBCA, an edge is present between two vertices if they are interfering receivers. Whereas in conflict graph of LBCA, an edge is present between two vertices if they are interfering senders. Edges in conflict graph are assigned different channels. The total number of channels are always fixed (for example, 16 in 802.15.4). To guarantee interference free transmission, interfering nodes using the same channel are assigned different transmission slots.

In C.Lin [2011], a distributed and scalable time slots assignment protocol called GLASS (Grid based LAtin Square Scheduling) is proposed. In distributed scheduling algorithms, nodes have to exchange control messages with neighbors to decide time slots. This creates a lot of network traffic and results in more energy consumption at nodes. The GLASS protocol is aimed at reducing scheduling overhead. In GLASS, network is divided into cells forming a virtual grid. Each cell of grid is of size R x R. The value of R is 2.1r, where r is ratio range.

The GLASS protocol works in three steps. In the first step, every sensor finds its cell. The algorithm for this is executed locally in each sensor node. It requires every sensor to know its location. Once cell is identified, each sensor associates itself with a transmission sub-frame. A Transmission Frame (TF) is a sequence of time-slots. It is repeated again and again over time. A TF is divided into Sub Transmission Frames(STFs). Sensor finds its STF based on the ID of the cell to whom it is associated with.

Using Latin Square Matrix (LSM) technique, sensor of every cell selects slots for transmission and reception. Every sensor broadcasts its ID, cell ID and STF in its two hop neighborhood.

| Paper | Objective |
|---|---|
| DRAND (Rhee [2006]) | Scheduling only |
| DD-TDMA (Wang [2007]) | Reduce frequency of On-off switching |
| DCAS (Saifullah [2014]) | Reduce schedule Length |
| GLASS (C.Lin [2011]) | Reduce Control Overhead |

Table c: Summary of Miscellaneous Algorithms

Thus every sensor knows about details of sensors present in its cell. Every sensor builds a LSM. The rows in LSM are sensors and columns are slots. From LSM, transmission and reception slots are selected such that no collision occurs between interfering nodes i.e. nodes in two hop neighborhood. We have not explained LSM in detail. For details, the reader may refer the full paper.

2.3.1 *Discussion.* The importance of DRAND(Rhee [2006]) is that it is one of the pioneering paper in the field of distributed scheduling. It is the first distributed algorithm evaluated on real test-bed of sensor nodes. It results in better performance in terms of schedule length and control overhead compared to other approaches of that time.

The objective of DD-TDMA(Wang [2007]) is to reduce energy consumption by reducing frequency of on-off switching of nodes. The DCAS (Saifullah [2014]) uses multiple channels to reduce the total slots used to schedule the network. As a result, packets would reach to sink faster. Every node will get transmission turns more quickly. Finally, GLASS (C.Lin [2011]) is aimed to reduce control overhead generated during time-slot selection. As a result, energy of nodes would be saved.

It can be deduced that the papers summarized in this sub-section propose different methods of reducing schedule length or reduce energy consumption. The cause of energy consumption may be either on-off switching, control overhead or data transmission. These techniques may be plugged into scheduling algorithms described in the previous two sub-sections with suitable modifications.

## 3. ALGORITHMS FOR FAULT TOLERANCE

Sensor nodes have limited energy. It is possible that some nodes die because of lack of energy. When a node dies, its children in the tree need to select new parent. As entire subtree rooted at orphan child is shifted to new parent, workload of new parent may increase. It may need to select extra time slots. More specifically all the nodes from new parent to sink may require extra slots. Thus change in topology not only requires tree repairing but also slots adjustment.

Often new nodes are added into network after initial deployment. This is to maintain connectivity. Each newly added node joins the tree i.e. selects some existing node as parent. This will increase workload of parent. Parent will require more time-slots.

In many applications, for e.g. environment monitoring, nodes are deployed in large quantity in a region of interest. They are continuously sensing the environment and periodically sending the data to sink. When some critical event occurs, some nodes, not necessarily all, may start sensing at higher rate. The data that a node senses during this critical period may not be temporarily correlated and thus may not be aggregated. As a result, node may require more time-slots to send additional packets. As such all nodes from given node to sink require more time-slots. Common mechanism can be used to handle topology and workload variation.

In Zao [2013], slots assignment in the case of dynamic traffic pattern is considered. Most of the scheduling algorithms consider static traffic pattern. It is considered that every node has fix number of packets available at the beginning of a TDMA frame. But as explained in Zao [2013], sometimes traffic pattern does not remain fix. Due to temporal and spatial correlation, a

node does not send packet in every frame. When there is a considerable difference in previously sent reading and current reading, packet is sent. Some applications require conditional reporting. When certain condition is satisfied, reading is to be transmitted to sink.

But slots allocation is done considering full traffic load. Thus often many slots remain unused. Nodes waste their energy in idle listening during slots assigned for receiving. This problem is addressed in Zao [2013]. It is proposed that parent transmits packets only after it receives packets from all children. Thus transmission slots of parent come after transmission slots of children. Whenever children of the given node have no data to send, it will transmit its own packet in very first transmission slot. In remaining slots, it will not transmit anything. When parent of given node finds first reception slot empty, it considers that node has no more data to transmit. So it remains in sleep mode during those slots. Thus energy is not consumed in idle listening.

In L.Zang [2012], fault tolerant scheduling is considered. It may happen that some node dies because of lack of energy. When a node dies, all its children should select new parent. As parent changes, old transmission slots may become invalid and new slots may be required. It is suggested that every node should compute a set of backup parents during tree formation. Also backup slots should also be computed. Backup slot means the slot using which communication will take place with backup parent. When parent node fails, its every child switches to new parent. As backup slots are pre-computed, delay of finding new slots is not present. But at the same time total schedule length increases. This increases end to end packet delay.

In Chaktraborty [2013], convergecast tree management from arbitrary node failure is proposed. Only tree formation is addressed, not scheduling. Otherwise the idea is same as in L.Zang [2012]. Distributed algorithm for tree construction is suggested. During tree construction, every node computes its alternate parent. Whenever a node fails, its children switch to alternate parent. Then alternate parent is considered as main parent. So tree repairing is done without much delay. If main parent and alternate parent both fail at the same time, node has to find new parent at the time when failure is detected. Thus failure of single parent is handled pro-actively whereas failure of multiple parents is handled reactively.

In Chaktraborty [2014], an improvement of the work done in Chaktraborty [2013] is presented. In the previous work, tree maintenance was done in either proactive or reactive manner. Failure of single node was handled proactively i.e. alternate parent was pre-computed. But when multiple nodes fail, reactive recovery is used. Recovery mechanism proposed in Chaktraborty [2014] results in smaller route repair delay compared to that of Chaktraborty [2013].

It is proposed that every node is either active or redundant. Active nodes are part of the tree. Redundant nodes are not part of the tree. During tree formation itself, every node decides whether it is active or redundant. Around each node few redundant nodes are present. Whenever an active node fails, a redundant node around failed node becomes active. Now data passes through that node. One more important feature presented in the paper is load balancing. The non-leaf nodes do not differ much in the count of children nodes. If any one node has large number of children, it will spend more energy in receiving packets and so it will die very quickly.

In Lee [2008] also fault tolerance is considered. Node has four types of slots: FTS (Fault Tolerant Slot), MFS (Multi-Function Slot), Transmission and Reception slots. FTS is used for fault tolerance. MFS is used to ensure time synchronization with children and sharing slots information with children. Whenever a node does not listen from its parent for two consecutive MFS (Multi-Functional Slot), parent is considered dead. During next FTS period, node will broadcast distress signal. All its neighbors will reply.

Node will select the neighbor having shortest path to sink as new parent. It will inform the decision to new parent. New parent will select a data transmission slot for child and assigns the same. Parent will also select a slot for transmitting data coming from child. Every node on the path to sink will select additional slot and inform to its parent. Whenever a node claims new slot, it has to propagate this information two its one hop and two-hop neighbors. This is to prevent collision of packets between interfering nodes. Node broadcasts slots information during MFS

| Paper | Approach |
|---|---|
| Zao [2013] | Node goes to sleep in unused slots |
| L.Zang [2012] | Compute backup parent and slot in advance |
| Chaktraborty [2013] | Compute backup parent in advance |
| Chaktraborty [2014] | Use of redundant nodes |
| FlexiTP (Lee [2008]) | If nothing is heard from parent for two MFS, parent may be died |

Table d: Summary of Fault Tolerant Algorithms

and also during FTS.

### 3.1 Discussion

In the Table d, different approaches of fault tolerance are summarized. The papers address following two issues: (i) When there is a reduction in traffic, nodes should not waste energy in idle listening (Zao [2013]) (ii) When some node dies, its children should quickly switch to new parent and slot (L.Zang [2012],Chaktraborty [2013] and Chaktraborty [2014]).

## 4. CONCLUSION AND OPEN ISSUES

We have reviewed different papers related to distributed scheduling algorithms, classified them and talked about their strengths and weaknesses. From discussion about scheduling algorithms and fault tolerance methods, it can be said that joint scheduling algorithms are preferable compared to sequential approach. That is, scheduling and parent selection should be done at the same time. In addition, the algorithms should also aim to reduce the control overhead generated during slot and parent selection process. There should also be a provision to quickly recover from node failures.

To summarize, a list of papers reviewed is given Table e. The papers are grouped as per the type of problem addressed i.e. aggregated convergecast, raw convergecast, generalized approaches and fault tolerance. The papers in each group are listed in order of year of publication. Against each paper, its objective is also mentioned. Still there is scope of further research in this domain. The possible research issues are summarized in next few paragraphs.

No joint scheduling & tree formation algorithm for raw convergecast is designed yet. In Bagga [2015], joint scheduling & tree formation for aggregation convergecast scheduling is proposed. It is bottom-up in nature. As explained earlier, it is desirable to have hybrid approach in raw convergecast. That is, every node should select slot to send its own packet at earliest (i.e. top-down, from root to leaves). But, the slots required to forward children's packets should be selected later (i.e. bottom-up, from leaves to root). It would be an interesting task to design hybrid joint scheduling & tree formation for raw convergecast.

In fault tolerance, mainly following two issues are considered: (i) When traffic is less, how to prevent idle listening in unused slots. (ii) Select new parent/slot when current parent dies. The problem of sudden rise in traffic at a node is not handled yet. It is possible that some sensors may start sensing at higher rate when some event occurs.

For example, temperature sensors in a jungle may increase rate of sensing when fire takes place. They need more time slots to quickly forward the data. It is required that these sensors should be temporarily given more time slots. When they come back to normal sensing rate, additional slots may be relinquished. Thus scheduling mechanism should be elastic in nature i.e. slots should be granted and revoked as and when required.

When some important event occurs, sensors should be able to send the related readings without much delay. In a small network, the schedule length would be small. But schedule length would

| Year | Paper | Type | Objective |
|---|---|---|---|
| 2009 | Distributed Wakeup Scheduling (Wu [2009]) | Aggr. Conv. | Minimize Energy Consumption |
| 2012 | MOSS (Yu [2012]) | Aggr. Conv. | Minimize Energy Consumption |
| 2013 | Multipath DICA ([Bagga 2013a]) | Aggr. Conv. | Minimize Schedule Length |
| 2013 | Multipath Multi-channel DICA ([Bagga 2013b]) | Aggr. Conv. | Minimize Schedule Length |
| 2015 | DICA ([Bagga 2015]) | Aggr. Conv. | Minimize Schedule Length |
| 2008 | FlexiTP ([Lee 2008]) | Raw Conv. | Minimize Buffering of Packets |
| 2014 | Collaboration based (B.Zeng [2014]) | Raw Conv | Minimize Energy Consumption |
| 2014 | Joint Channel & Slot Assignment (R.Soua [2014]) | Raw Conv | Minimize Schedule Length |
| 2006 | DRAND (Rhee [2006]) | Generalized | Collision-free Schedule Formation |
| 2007 | DD-TDMA (Wang [2007]) | Generalized | Reduce frequency of On-off switching |
| 2011 | GLASS (C.Lin [2011]) | Generalized | Reduce Control Overhead |
| 2014 | DCAS (Saifullah [2014]) | Generalized | Reduce schedule Length |
| 2012 | L.Zang [2012] | Fault Tolerance | Fast Tree Repairing |
| 2013 | Zao [2013] | Fault Tolerance | Minimize Energy Consumption |
| 2013 | Chaktraborty [2013] | Fault Tolerance | Fast Tree Repairing |
| 2014 | Chaktraborty [2014] | Fault Tolerance | Fast Tree Repairing |

Table e: Summary of Papers Reviewed

be large in a large network. If an event occurs in some part of the network, the nodes in a large network will have to wait for longer duration to get transmission turn. Some method may be developed to support priority-based data transmission. That is, normally nodes transmit in their assigned slot. But if some high-priority data is to be sent, it should be sent without much delay.

## REFERENCES


Bagga, M. 2013a. Efficient multi-path data aggregation scheduling in wireless sensor networks. *IEEE International Conference on Communications*.

Bagga, M. 2013b. Multi-path multi-channel data aggregation scheduling in wireless sensor networks. *IEEE Wireless Days International Conference*.

Bagga, M. 2015. Distributed low latency data aggregation scheduling in wireless sensor networks. *ACM Transactions on Sensor Networks 11,3*.

Barrenetxea, G. 2008. Sensorscope: Out-of-the-box environmental monitoring. *USENIX OSDI*.

B.Zeng. 2014. A collaboration based distributed tdma scheduling algorithm for data collection in wireless sensor network. *Journal of Networks, Academy Publishers 9,9*.

Chaktraborty, S. 2013. Convergecast tree management from arbitrary node failure in sensor network. *Ad Hoc Networks Journal, Elsevier Publications 11,6*.

Chaktraborty, S. 2014. Topology management ensuring reliability in delay sensitive sensor networks with arbitrary node failure. *International Journal of Wireless Inf. Networks, Springer Publications 21,4*.

C.Lin. 2011. A distributed and scalable time slot allocation protocol for wireless sensor networks. *IEEE Transactions on Mobile Computing 10,4*.

Ghosh, A. 2010. Bounded degree minimum radius spanning trees for fast data collection in wireless sensor networks. *IEEE INFOCOM*.

Ghosh, A. 2011. Scheduling algorithms for tree-based data collection in wireless sensor networks. *Theoretical Aspects of Distributed Computing in Sensor Networks, Springer*.

Hartung, C. 2006. Firewxnet: A multi-tiered portable wireless system for monitoring weather conditions in wildland fire environments. *ACM MobiSys*.

Hia, M. 2011. W-mac: A workload-aware mac protocol for heterogeneous convergecast in wireless sensor networks. *MDPI Sensors Journal 17,2*.

Kim, Y. 2008. Nawms: Nonintrusive autonomous water monitoring system. *ACM SenSys*.

Lee, W. 2008. Flexitp: A flexible schedule based tdma protocol for fault tolerant and energy-efficient wireless sensor networks. *IEEE Transactions on Parallel and Distributed Systems 19,6*.

L.Zang. 2012. Fault tolerant scheduling for data collection in wireless sensor networks. *IEEE GLOBECOM*.

Malhotra, B. 2011. Aggregation convergecast scheduling in wireless sensor networks. *Springer Journal of Wireless Networks 17,2*.

Mamun, Q. 2012. A qualitative comparison of different logical topologies for wireless sensor networks. *MDPI Journal of Sensors*.

M.Bagga. 2014. Data aggregation scheduling algorithms in wireless sensor networks: Solutions and challenges. *IEEE Communication Surveys & Tutorials 16,3*.

McGrath, M. 2014. Sensor network topologies and design considerations. *Sensor Technologies Healthcare, Wellness and Environmental Applications, Springer*.

O.D.Incel. 2012. Fast data collection in tree based wireless sensor networks. *IEEE Transactions on Mobile Computing 11*.

Pan, M. 2008. Quick convergecast in zigbee beacon enabled wireless sensor networks. *ACM Journal of Computer Communications*.

Ren, F. 2012. Attribute-aware data aggregation using potential-based dynamic routing in wireless sensor networks. *IEEE Transactions on Parallel and Distributed Systems 24,5*.

Rhee, I. 2006. Drand: Distributed randomized tdma scheduling for wireless ad hoc networks. *IEEE Transactions on Mobile Computing 8,6*.

R.Soua. 2014. A distributed joint channel and time slot assignment for convergecast in wireless sensor networks. *6th International Conference on New Technology, Mobility and Security*.

Saifullah, A. 2014. Distributed channel allocation protocols for wireless sensor networks. *IEEE Transactions on Parallel and Distributed Systems 25,9*.

Selavo, L. 2007. Luster: Wireless sensor network for environmental research. *ACM SenSys*.

Soua, R. 2013. Musika: A multi-channel multiple sinks data gathering algorithm for wireless sensor networks. *IEEE International Wireless Communications and Mobile Computing Conference (IWCMC)*.

Wang, Y. 2007. A deterministic distributed tdma scheduling algorithm for wireless sensor networks. *International Conference on Wireless Communications, Networking and Mobile Computing*.

WernerAllen, G. 2006. Fidelity and yield in a volcano monitoring sensor network. *USENIX OSDI*.

Wu, F.-J. 2009. Distributed wake up scheduling for data collection in tree based wireless sensor networks. *IEEE Communication Letters 13,3*.

W.Z.Song. 2009. Air-dropped sensor network for real-time high-fidelity volcano mon- itoring. *ACM MobiSys*.

Yu, C. 2012. Many to one communication protocol for wireless sensor networks. *International Journal of Sensor Networks, Inderscience Publications 12,3*.

Zao, W. 2013. Scheduling data collection with dynamic traffic patterns in wireless sensor networks. *IEEE Transactions on Parallel and Distributed Systems 24,4*.

**Dr. Tejas Vasavada** is an Assistant Professor of Information Technology at Lukhdhirji Engineering College (Run by Government of Gujarat), Morbi, India. He has received PhD degree from Dhirubhai Ambani Institute of Information and Communication Technology, Gandhinagar, India in January, 2019. His research interests include ad hoc & sensor networks, distributed algorithm design and simulation & modeling.

**Dr. Sanjay Srivastava** is a professor at Dhirubhai Ambani Institute of Information and Communication Technology (DA-IICT), Gandhinagar, India. He has received his PhD from University of California, Los Angeles. He works in the area of sensor networks and network protocol design and analysis.