1. **Introduction**

Wireless Sensor Networks (WSNs) are generally composed of one or more sinks and tens or thousands of sensor nodes scattered in a physical space. The purpose of WSN congestion control is to improve the network throughput, reduce the time of data transmitted delay. There are multiple cluster heads are available in WSN. Each of the cluster heads collects the data from respective cluster’s nodes and forwards the aggregated data to base station. A major challenge in WSNs is to select appropriate cluster heads. We have used an alternative way to select the cluster heads in WSN. In a densely deployed network, by varying the buffer size of the cluster heads we can improve the congestion. In addition to that we will use ‘traffic node’ which will manage the data packets coming from cluster heads.

Buffers size of the packet plays an important role in network communications. Buffer size is essential to ensure communication effectiveness. The packets are queued unnecessarily causing delay in communication. It also decreases throughput. By varying the buffer size in multipath multihop WSN can help us to improve congestion. We will use LEACH algorithm to lower the energy consumption required to create and maintain clusters in order to improve the life time of a wireless sensor network.

1. **Literature Survey**

Congestion is one of the major issues which adversely affect all the QOS parameters. Congestion is considered to be occurred when any of the node or link gets the data more than its processing capacity.[1] Congestion occurs due to high reporting rate, node density as large number of packets thumbed into the network. Congestion not only causes packet loss, but also leads to excessive energy consumption as well as delay. Therefore, in order to prolong network lifetime and improve fairness and provide better quality of service, developing a novel solution for congestion estimation and control is important to be considered. Congestion can be reduced by two means either by congestion detection or congestion avoidance. Congestion can be detected at node level by measuring buffer occupancy or at link level by measuring links busyness.

There are two types of congestion. 1. Node-Level Congestion 2. Link-Level Congestion. Congestion can be detected at node level by measuring buffer occupancy or at link level by measuring links busyness. In node-level congestion, the node-level congestion that is common in conventional networks. It is caused by buffer overflow in the node and can result in packet loss, and increased queuing delay. In Link-Level Congestion, a particular area has severe collisions could occur when multiple active sensor nodes within range of one another attempt to transmit at the same time. Packets that leave the buffer might fail to reach the next hop as a result of collision. This type of congestion decreases both link utilization and overall throughput, while increasing both packet delay and energy waste.

## Congestion Control Protocols

Congestion control concerns controlling traffic entry into a [telecommunications network](https://en.wikipedia.org/wiki/Telecommunications_network), so as to avoid [congestive collapse](https://en.wikipedia.org/wiki/Congestive_collapse) by attempting to avoid oversubscription of any of the processing or [link](https://en.wikipedia.org/wiki/Data_link) capabilities of the intermediate nodes and networks and taking resource reducing steps, such as reducing the rate of sending [packets](https://en.wikipedia.org/wiki/Packet_(information_technology)). [2] It should not be confused with [flow control](https://en.wikipedia.org/wiki/Flow_control_(data)), which prevents the sender from overwhelming the receiver. Two general approaches to control congestion are

1. Resource Management

2. Traffic control

Congestion Avoidance and Detection (CODA)

CODA is energy efficient congestion control mechanism designed for WSNs. CODA, detects the congestion by observing the buffer size of sensor nodes and the load of the wireless channel. If these two characteristic exceed from a pre-defined threshold, a sensor node informs its neighbour to decrease the transmission rate. [3] Before transmitting a packet, a sensor node divides the channel to fixed periods. If it found the channel busier than pre-defined times, it adjusts a control bit to inform the base station of the congestion.

Congestion Control and Fairness (CCF)

CCF detects congestion based on packet service time at MAC layer and control congestion based on hop-by hop manner with simple fairness. CCF uses packets service time to deduce the available service rate and detect the congestion in each intermediate node. When the congestion is experienced, it informs the downstream nodes to reduce their data transmission rate and vice versa.

Adaptive Rate Control (ARC)

ARC monitors the injection of packets into the traffic stream as well as route-through traffic. Each node estimates the number of upstream nodes and the bandwidth is split proportionally between route-through and locally generated traffic, with preference given to the former. The resulting bandwidth allocated to each node is thus approximately fair. Also, reduction in transmission rate of route-through traffic has a backpressure effect on upstream nodes, which in turn can reduce their transmission rates.

SenTCP

SenTCP is an open-loop hop-by-hop congestion control protocol with two special features: 1) It jointly uses average local packet service time and average local packet inter-arrival time in order to estimate current local congestion degree in each intermediate sensor node. The use of packet arrival time and service time not only precisely calculates congestion degree, but effectively helps to differentiate the reason of packet loss occurrence in wireless environments, since arrival time ( or service time) may become small (or large) if congestion occurs. 2) It uses hop-by-hop congestion control. In SenTCP, each intermediate sensor node will issues feedback signal backward and hop-by-hop. The feedback signal, which carries local congestion degree and the buffer occupancy ratio, is used for the neighboring sensor nodes to adjust their sending rate in the transport layer. The use of hop- by-hop feedback control can remove congestion quickly and reduce packet dropping, which in turn conserves energy. SenTCP realizes higher throughput and good energy-efficiency since it obviously reduces packet dropping; however, SenTCP copes with only congestion and guarantees no reliability.

Fairness Aware Congestion Control (FACC)

FACC is a congestion control mechanism, which controls the congestion and achieves fair bandwidth allocation for each flow of data. FACC detects the congestion based on packet drop rate at the sink node. In FACC nodes are divided in to two categories near sink node and near source node based on their location in WSNs. When a packet is lost, then the near sink nodes send a Warning Message (WM) to the near source node. [4] After receiving WM the near source nodes send a Control Message (CM) to the source node. The source nodes adjust their sending rate based on the current traffic on the channel and the current sending rate. After receiving CM, flow rate would be adjusted based on newly calculated sending rate.

Fusion

In Fusion hop by hop flow control mechanism is used for congestion detection as well as congestion mitigation. Congestion is detected through queue occupancy and channel sampling technique at each intermediate node. Congestion notification (CN) bit will set in the header of every outgoing packet when the node detects congestion. Once the CN bit is set, neighboring node can overhear it and stop forwarding packet to the congested node.

Priority Based Congestion Control Protocol (PCCP)

PCCP is a congestion control mechanism based on node priority index that is introduced to reflect the importance of each sensor node. Nodes are assigned a priority based on the function they perform and its location. Nodes near the sink have a higher priority. The congestion is detected based on the ratio of sending rate to the packet arrival rate. If the sending rate is lower, it implies that congestion has occurred. The congestion information is piggybacked in data packet header along with the priority index. Nodes adjust their sending rate depending on the congestion at the node itself. PCCP tries to reduce packet loss in congestion state while achieving the weighted fairness transmission for single-path and multipath routing.

Trickle

Trickle, an algorithm for propagating and maintaining code updates in wireless sensor networks. [5] Trickle's basic primitive is simple: every so often, a mote transmits code metadata if it has not heard a few other motes transmit the same thing. This allows Trickle to scale to thousand-fold variations in network density, quickly propagate updates, distribute transmission load evenly, be robust to transient disconnections, handle network repopulations, and impose maintenance overhead on the order of a few packets per hour per mote. Trickle sends all messages to the local broadcast address. There are two possible results to a Trickle broadcast: either every mote that hears the message is up to date or a recipient detects the need for an update. Detection can be the result of either an out-of-date mote hearing someone has new code, or an updated mote hearing someone has old code. As long as every mote communicates somehow - either receives or transmits - the need for an update will be detected. For example, if mote A broadcasts that it has code φ, but B has code φ+1, then B knows that A needs an update. Similarly, if B broadcasts that it has φ+1, A knows that it needs an update. If B broadcasts updates, then all of its neighbors can receive them without having to advertise their need. Some of these recipients might not even have heard A's transmission. Trickle uses "polite gossip" to exchange code metadata with nearby network neighbors. It breaks time into intervals, and at a random point in each interval, it considers broadcasting its code metadata. If Trickle has already heard several other motes gossip the same metadata in this interval, it politely stays quiet: repeating what someone else has said is rude.

Siphon

Siphon aims at controlling congestion as well as handling funneling effect. Funneling effect is where events generated under various work load moves quickly towards one or more sink nodes, which increases traffic at sink which leads to packet loss. Virtual sinks are randomly distributed across the sensor network which takes the traffic load off the already loaded sensor node. In siphon initially VS discovery is done. Virtual sink discovery is initiated by the physical sink by as explained in . Node initiated congestion detection is based on past and present channel condition and buffer occupancy as in CODA. After congestion detection traffic is redirected from overloaded physical sink to virtual sinks. It is done by setting redirection bit in network layer header.

Prioritized Heterogeneous Traffic-oriented Congestion Control Protocol (PHTCCP)

PHTCCP is an efficient congestion control protocol for handling diverse data with different priorities within a single node motivates. PHTCCP module works interacting with the MAC layer to perform congestion control function. In this protocol, we focus on efficient mechanism so that congestion could be controlled by ensuring adjustment transmission rates for different type of data that generated by the sensors have various priorities. We assume that the sink node assigns individual priority for each type of sensed data and each node has n number of equal sized priority queues for n types of sensed data Heterogeneous applications can reflect the number of queues in a node. In congestion detection method , congestion level at each sensor node presented by packet service ratio.

r ( i )= Rsi / Rschi , Rsi is the ratio of average packet service rate and Rschi is the packet scheduling rate in each sensor node.

## Congestion Avoidance Protocols

Congestion avoidance techniques monitor network traffic loads in an effort to anticipate and avoid congestion at common network bottlenecks. Congestion avoidance is achieved through packet dropping. Among the more commonly used congestion avoidance mechanisms is Random Early Detection (RED), which is optimum for high-speed transit networks.

Random Early Detection

One solution is to use [random early detection](https://en.wikipedia.org/wiki/Random_early_detection) (RED) on the network equipment's [port queue buffer](https://en.wikipedia.org/wiki/Computer_port_(software)). On [network equipment](https://en.wikipedia.org/wiki/Computer_networking_device) ports with more than one queue buffer, [weighted random early detection](https://en.wikipedia.org/wiki/Weighted_random_early_detection) (WRED) could be used if available. RED indirectly signals to sender and receiver by deleting some packets, e.g. when the average queue buffer lengths are more than e.g. 50% (lower threshold) filled and deletes [linearly more](https://en.wikipedia.org/wiki/Linear) or (better according to paper) [cubical more](https://en.wikipedia.org/wiki/Cubic_function) packets, up to e.g. 100% (higher threshold). The average queue buffer lengths are computed over 1 second at a time.

TCP/IP congestion avoidance

The [TCP congestion avoidance algorithm](https://en.wikipedia.org/wiki/TCP_congestion_avoidance_algorithm) is the primary basis for [congestion control](https://en.wikipedia.org/wiki/Congestion_control) in the Internet. Problems occur when many concurrent TCP flows are experiencing [port queue buffer](https://en.wikipedia.org/wiki/Computer_port_(software)) [tail-drops](https://en.wikipedia.org/wiki/Tail-drop). Then TCP's automatic congestion avoidance is not enough. All flows that experience port queue buffer tail-drop will begin a TCP retrain at the same moment – this is called [TCP global synchronization](https://en.wikipedia.org/wiki/TCP_global_synchronization).

TCP Tahoe and Reno

To avoid [congestion collapse](https://en.wikipedia.org/wiki/Congestive_collapse), TCP uses a multi-faceted congestion-control strategy. For each connection, TCP maintains a [congestion window](https://en.wikipedia.org/wiki/Congestion_window), limiting the total number of unacknowledged packets that may be in transit end-to-end. This is somewhat analogous to TCP's [sliding window](https://en.wikipedia.org/wiki/Sliding_Window_Protocol) used for [flow control](https://en.wikipedia.org/wiki/Transmission_Control_Protocol#Flow_control). TCP uses a mechanism called [slow start](https://en.wikipedia.org/wiki/Slow-start) to increase the congestion window after a connection is initialized and after a timeout. It starts with a window of two times the [maximum segment size](https://en.wikipedia.org/wiki/Maximum_segment_size) (MSS). Although the initial rate is low, the rate of increase is very rapid: for every packet acknowledged, the congestion window increases by 1 MSS so that the congestion window effectively doubles for every [round-trip time](https://en.wikipedia.org/wiki/Round-trip_delay_time) (RTT). When the congestion window exceeds the ssthresh threshold, the algorithm enters a new state, called [congestion avoidance](https://en.wikipedia.org/wiki/Congestion_avoidance). In some implementations (e.g., Linux), the initial ssthresh is large, and so the first slow start usually ends after a loss. However, ssthresh is updated at the end of each slow start, and will often affect subsequent slow starts.

As long as non-duplicate ACKs are received, the congestion window is additively increased by one MSS every round trip time. When a packet is lost, the likelihood of duplicate ACKs being received is very high (it's possible though unlikely that the stream just underwent extreme packet reordering, which would also prompt duplicate ACKs). The behavior of Tahoe and Reno differ in how they detect and react to packet loss:

* Tahoe: Common Tahoe implementations detect congestion only by setting a timer for receiving a related ACK. Tahoe sets the slow start threshold to half of the current congestion window, reduces the congestion window to 1 MSS, and resets to slow-start state.
* Reno: If three duplicate ACKs are received (i.e., four ACKs acknowledging the same packet, which are not piggybacked on data, and do not change the receiver's advertised window), Reno will halve the congestion window (instead of setting it to 1 MSS like Tahoe), set the slow start threshold equal to the new congestion window, perform a fast retransmit, and enter a phase called [Fast Recovery](https://en.wikipedia.org/wiki/Slow-start#Fast_recovery). If an ACK times out, slow start is used as it is with Tahoe.

The two main differences between Tahoe and Reno are:

1. Tahoe only uses a timeout for detecting congestion, while Reno uses timeout and Fast-Retransmit
2. Tahoe sets the congestion window to 1 after packet loss, while Reno sets it to half of the latest congestion window.

#### Robust random early detection (RRED)

The [Robust Random Early Detection](https://en.wikipedia.org/wiki/Robust_random_early_detection) (RRED) algorithm was proposed to improve the TCP throughput against denial-of-service (DoS) attacks, particularly low-rate denial-of-service (LDoS) attacks. Experiments have confirmed that the existing RED-like algorithms are notably vulnerable under Low-rate Denial-of-Service (LDoS) attacks due to the oscillating TCP queue size caused by the attacks.

Weighted Random Early Detection (WRED)

WRED is a queuing discipline for a [network scheduler](https://en.wikipedia.org/wiki/Network_scheduler) suited for [congestion avoidance](https://en.wikipedia.org/wiki/Network_congestion_avoidance). It is an extension to [random early detection](https://en.wikipedia.org/wiki/Random_early_detection) (RED) where a single queue may have several different queue thresholds. Each queue threshold is associated to a particular [traffic class](https://en.wikipedia.org/wiki/Traffic_shaping#Traffic_classification). For example, a queue may have lower thresholds for lower priority packet. A queue buildup will cause the lower priority packets to be dropped, hence protecting the higher priority [packets](https://en.wikipedia.org/wiki/Network_packet) in the same queue. In this way [quality of service](https://en.wikipedia.org/wiki/Quality_of_service) prioritization is made possible for important packets from a pool of packets using the same buffer. It is more likely that standard traffic will be dropped instead of higher prioritized traffic.

Calculation of average queue size: avg = o\*(1-2-n) +c\*(2-n)

where n is the user-configurable exponential weight factor, o is the old average and c is the current queue length. The previous average will be more important for high values of n. Peaks and Lows in queue length will be smoothed by a high value. Low values of n allow the average queue size to stay close to the current queue size.

### Adaptive Random Early Detection (ARED)

The adaptive RED or active RED (ARED) algorithm infers whether to make RED more or less aggressive based on the observation of the average queue length. If the average queue length oscillates around min threshold then early detection is too aggressive. On the other hand if the average queue length oscillates around max threshold then early detection is being too conservative. The algorithm changes the probability according to how aggressively it senses it has been discarding traffic.

Buffer size is crucial for better performance in WSN such as reduction of congestion and delay. Various buffer management policies can be designed depending upon the optimum buffer size of the nodes in the network. The analysis of queue buffer size for real time network protocols made is found to be significant.

Low Energy Adaptive Clustering Hierarchy ("LEACH") is a [TDMA](https://en.wikipedia.org/wiki/Time_division_multiple_access)-based [MAC](https://en.wikipedia.org/wiki/Media_access_control) protocol which is integrated with clustering and a simple routing protocol in [wireless sensor networks](https://en.wikipedia.org/wiki/Wireless_sensor_network) (WSNs). The goal of LEACH is to lower the energy consumption required to create and maintain clusters in order to improve the life time of a wireless sensor network. LEACH is a hierarchical protocol in which most nodes transmit to cluster heads, and the cluster heads aggregate and compress the data and forward it to the base station (sink). Each node uses a stochastic [algorithm](https://en.wikipedia.org/wiki/Algorithm) at each round to determine whether it will become a cluster head in this round. LEACH assumes that each node has a radio powerful enough to directly reach the base station or the nearest cluster head, but that using this radio at full power all the time would waste energy.

In flooding, each sensor node receiving a data or a control packet repeats it by broadcasting, flooding has several shortcomings such as implosion (duplicated messages are sent to the same node), overlap (neighbor nodes receive duplicated messages) and resource blind-ness (does not take into account the energy resources, it is not energy aware). In gossiping, nodes do not broad-cast but send the incoming messages to randomly selected neighbors. A sensor node randomly select s one of its neighbors to send the data. Once the neighbor node receives the data, it randomly selects another sensor node. Gossiping avoids the implosion problem but still is not energy aware, and it takes a long time to propagate the message to reach the sink.

Solar panels are frequently used in wireless sensor nodes because they can theoretically provide quite a bit of harvested energy. However, they are not a reliable, consistent source of energy because of the Sun’s cycles and the ever changing weather conditions.

1. **Design & Architecture**

**3.1 Hierarchical Architecture**

Data-centric network topologies are not suitable for large-scale sensor networks. Covering a large area without performance degradation is not possible with data-centric architecture. Moreover, in data-centric architectures, the reporting latency increases with the size of the network. The data-centric approach also causes significant power inefficiencies as the network grows. The network scalability issue is addressed in hierarchical routing. The hierarchical routing’s main goal is to efficiently maintain network power consumption even in large-scale networks. In other words, hierarchical routing allows the network to scale in a number of sensor nodes. Most hierarchical architectures consist of sensor nodes grouped into cluster heads. Cluster heads build intra-cluster communication with other nodes within the same cluster, but they also build inter-cluster communicate with other cluster heads. Cluster heads aggregate data obtained individual sensors and then transfer the same information mostly in a multi-hop approach to the base. Figure 3.1 is showing the design of the proposed algorithm.

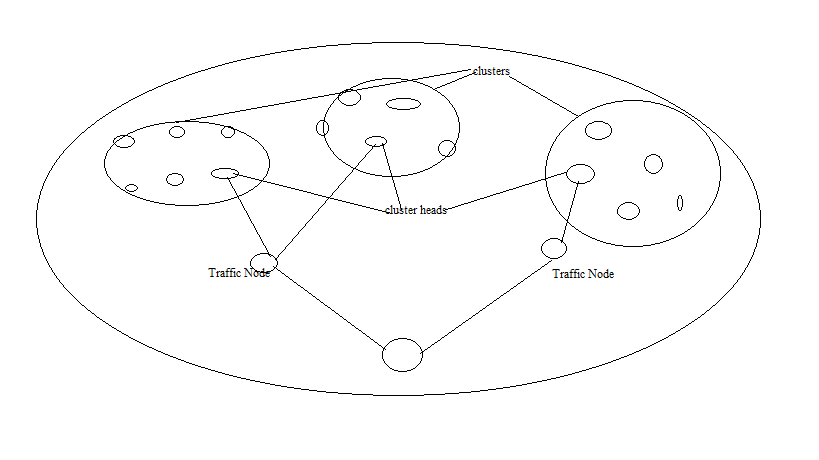


Figure 3.1 Design

Low-Energy Adaptive Clustering Hierarchy (LEACH) is one of the most popular hierarchical architectures. LEACH utilizes a randomized rotation of local cluster heads to evenly distribute the energy load among sensors in the network. It also minimizes the overall energy consumption by allowing each sensor node to determine which cluster it wants to join by choosing the cluster head that requires the minimum communication energy (typically, the cluster head closest to the sensor). However, LEACH architecture determines the percentage of cluster heads in the network and cluster switching frequency a priori. This approach may lead to a less than optimal number of cluster heads in the network at any point of time. It also leads to unnecessary overuse of cluster head switching and a waste of network power capacity associated with the switching overhead.

1. **Module Description**

* The project consists of following modules

|  |
| --- |
| 1. Node Selection |
| 1. Cluster Formation and Cluster Head Selection |
| 1. Algorithms used in WSN |
| 1. Node Configuration |

**4.1 Node Selection**

Node Selection module consists identification of different node to assign different characteristics. Initially we will define the traffic nodes then the LEACH algorithm will be applied to the rest of the nodes that can define the clusters, cluster heads and sink nodes.

**4.2 Cluster Formation and Cluster Head Selection**

Every round begins with a CH selection; each node in the network decides whether to become the CH for the current round or not. Depending on the required percentage of cluster heads for the network and the number of times the node has been a cluster head. For any node *n,* the threshold equation for CH selection is given by,

T(n) if n € G

T(n) = 0, otherwise

where, P is the desired percentage of CH, r is the current round and G is the set of nodes that have not been CH in the last 1/P rounds. Every node in G chooses a random number between 0 and 1 and if the number is less than the threshold value, it is selected as the CH for the current round.

1. Cluster Formation: Each node that has elected itself as a cluster-head for the current

round broadcasts an advertisement (ADV) message to its one hop neighbour with the

following information:

2. Budget allocated to the neighbour (*B*).

3. Minimum number of hops to reach cluster head through it (*hmin*).

4. Average energy of the path (*Eavg*).

5. Traffic information through that node (*t*).

During the cluster formation every node that receives ADV messages with/without a budget allotted to it. If the node has budget allotted to it in the ADV message, it checks if there already exists a shortest path to the CH in the routing table. If there is no such entry, it adds this path as the shortest path else adds it as the alternate path in the routing table. If the node has no budget allotted to it in the ADV message, it adds the path as an alternate path in its routing table. Hence it will join the cluster and forwards the ADV message to the rest of its neighbors. Initially, the CH node A is assigned a budget of 9 and it consumes 1 for itself and divides the remaining budget amongst its one hop neighbors and sends out an ADV message. Node B hears the ADV message from A first and adds this route as the shortest path. It then forwards the ADV message to the rest of its neighbors. After sometime, Node B hears another ADV message from E and since its shortest path has already been established, it adds this path as an alternate path. This process continues until the budget exhausts.

**4.3** **Algorithms used in WSN**

**4.3.1 TDMA (Time Division Multiple Access)**

TDMA is a type of [time-division multiplexing](https://en.wikipedia.org/wiki/Time-division_multiplexing), with the special point that instead of having one [transmitter](https://en.wikipedia.org/wiki/Transmitter) connected to one [receiver](https://en.wikipedia.org/wiki/Receiver_(radio)), there are multiple transmitters. In the case of the [uplink](https://en.wikipedia.org/wiki/Uplink) from a [mobile phone](https://en.wikipedia.org/wiki/Mobile_phone) to a [base station](https://en.wikipedia.org/wiki/Base_station) this becomes particularly difficult because the mobile phone can move around and vary the timing advance required to make its transmission match the gap in transmission from its peers.

**4.3.1.1 TDMA Characteristics**

* Shares single carrier frequency with multiple users
* Non-continuous transmission makes handoff simpler
* Slots can be assigned on demand in dynamic TDMA
* Less stringent power control than [CDMA](https://en.wikipedia.org/wiki/Code_division_multiple_access) due to reduced intra cell interference
* Higher synchronization overhead than CDMA
* Advanced [equalization](https://en.wikipedia.org/wiki/Equalization_(communications)) may be necessary for high data rates if the channel is "frequency selective" and creates [Inter Symbol Interference](https://en.wikipedia.org/wiki/Intersymbol_interference)(ISI)
* [Cell breathing](https://en.wikipedia.org/wiki/Cell_breathing_(telephony)) (borrowing resources from adjacent cells) is more complicated than in CDMA
* Frequency/slot allocation complexity
* Pulsating power envelope: [interference](https://en.wikipedia.org/wiki/Electromagnetic_interference) with other devices

**4.3.2 Low Energy Adaptive Clustering Hierarchy (LEACH)**

Low Energy Adaptive Clustering Hierarchy ("LEACH") is a [TDMA](https://en.wikipedia.org/wiki/Time_division_multiple_access)-based [MAC](https://en.wikipedia.org/wiki/Media_access_control) protocol which is integrated with clustering and a simple routing protocol in [wireless sensor networks](https://en.wikipedia.org/wiki/Wireless_sensor_network) (WSNs). The goal of LEACH is to lower the energy consumption required to create and maintain clusters in order to improve the life time of a wireless sensor network. LEACH is a hierarchical protocol in which most nodes transmit to cluster heads, and the cluster heads aggregate and compress the data and forward it to the base station (sink). Each node uses a stochastic [algorithm](https://en.wikipedia.org/wiki/Algorithm) at each round to determine whether it will become a cluster head in this round. LEACH assumes that each node has a radio powerful enough to directly reach the base station or the nearest cluster head, but that using this radio at full power all the time would waste energy. Nodes that have been cluster heads cannot become cluster heads again for *P* rounds, where *P* is the desired percentage of cluster heads. Thereafter, each node has a 1/*P* probability of becoming a cluster head again. At the end of each round, each node that is not a cluster head selects the closest cluster head and joins that cluster. The cluster head then creates a schedule for each node in its cluster to transmit its data. All nodes that are not cluster heads only communicate with the cluster head in a [TDMA](https://en.wikipedia.org/wiki/Time_Division_Multiple_Access) fashion, according to the schedule created by the cluster head. They do so using the minimum energy needed to reach the cluster head, and only need to keep their radios on during their time slot. LEACH also uses [CDMA](https://en.wikipedia.org/wiki/Code_Division_Multiple_Access) so that each cluster uses a different set of CDMA codes, to minimize interference between clusters.

**4.3.2.1 Properties of LEACH**

* Cluster based
* Random cluster head selection each round with rotation. Or cluster head selection based on sensor having highest energy
* Cluster membership adaptive
* Data aggregation at cluster head
* Cluster head communicate directly with sink or user
* Communication done with cluster head via TDMA
* Threshold value

**4.4 Node Configuration**

**4.4.1 Traffic Nodes**

Traffic Node will work as an intermediate node between cluster heads and sink node. It will manage the routes between the nodes and sink node. It will advertise the cluster heads to assign the time for the data packets using TDMA.

**4.4.2 Sink Node**

Sensor network comprises of scattered sensor nodes with limited computational capabilities and battery power. The existing security solutions for traditional wireless networks cannot be used because of the constraints associated with sensor network. We present secure sink node architecture as two-tiered scheme for sensor network security. The architecture protects the sink node from unauthorized access by surrounding it with two protection layers. Sink nodes listen to only inner layer nodes and inner nodes are allowed to communicate with only outer layer nodes. These protection layers are formed in an intelligent manner without violating constraints specific to sensor network. In order to enhance security, protection layers are re-adjusted in case of an attack.

**4.4.3 Energy Harvesting Nodes**

Energy Harvesting Nodes provide a very small amount of power for low-energy. Energy harvesting nodes are with the capability of extracting energy from the surrounding environment. Energy harvesting can exploit diﬀerent sources of energy, such as solar power, wind, mechanical vibrations, temperature variations, magnetic ﬁelds, etc.

1. **Use Case Diagram**

* The UML provides the use case diagram notation to illustrate the name of the use case actors and relationship between them.

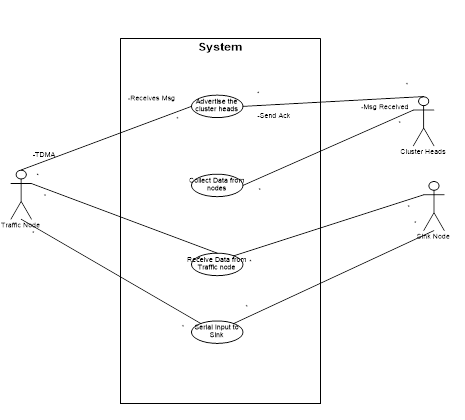


Figure 5.1 Use Case Diagram

1. **Class Diagram**

* The UML include the class diagram, to illustrate and their association. They are used for static object modelling.

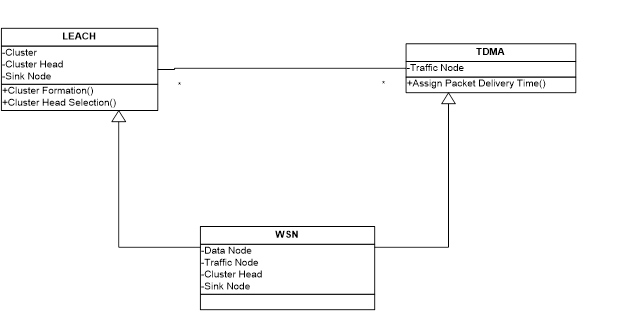


Figure 6.1 Class Diagram

1. **Sequence Diagram**

* A sequence diagram illustrates in a kind of format in which each object interact via messages. It is generalization between two or more specification diagram.

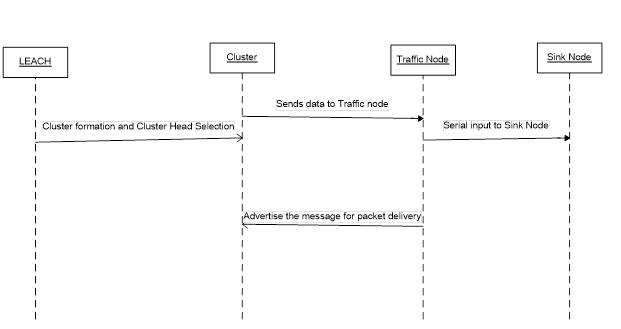


Figure 7.1 Sequence Diagram

1. **Activity Diagram**

* Activity diagram shows sequential and parallel activities in a process. They are useful for modeling business, workflows, the data flows and complex algorithm.

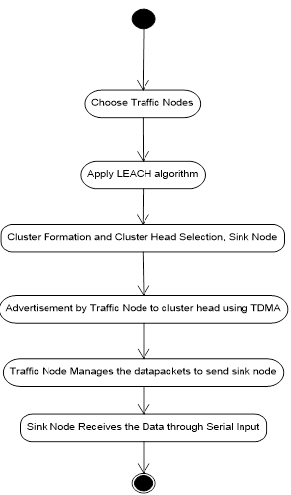


Figure 8.1 Activity Diagram

1. **Conclusion**

During Survey, congestion found to be a sensitive problem for WSN. As the path length increased and varied buffer size may improve the existing congestion problem. As congestion is a major issue that can be also affect the throughput, energy efficiency and all other QoS.

**References**

[1] Swati Kadu, Vivek Deshpade,“Optimaization Of Selective Throughput In Wsn Using Priority Scheduling”, Cyber Times International Journal Of Technology And Management Vol 7 Issue 1,Oct 13- March 14.

[2] Swati Kadu, Vivek Deshpande, “Handeling Throughput In Wireless Sensor Network”, IEEE International Conference On Computational Intelligence And Computing Research, 2012.

[3] Yi Huang and Yingbo Hua, “On Energy for progressive and Consensus Estimation in Multihop Sensor Network,”IEEE TRANSACTIONS ON SIGNAL PROCESSING, VOL. 59, NO. 8, pp 3863-3875, AUGUST 2011.

[4] Gajendra S. Vyas, Vivek S. Deshpande, “Performance Analysis of Congestion in Wireless Sensor Networks”, 3rd IEEE International Advance Computing Conference (IACC),Feb 2013.

[5] Swati Kadu, Vivek Deshpande, “Characterization Of Throughput In Wireless Sensor Network For Mac And Routing Protocol”, International Conference On Cloud & Ubiquitous Computing & Emerging Technologies 2013

[6] Vivek Deshpande, Prachi Sarode, Sambhaji Sarode, “Root Cause Analysis of Congestion in Wireless Sensor network Wireless Sensor Networks”, International Journal of Computer Application, Vol. 1, No.18, pp. 27-30, Feb 2010.

[7] Vivek S. Deshpande, Pratibha P. Chavan, Vijay M. Wadhi, Jagdish B. Helonde, “Congestion Control in Wireless Sensor Networks by using Differed Reporting Rate”, IEEE Conference 2012 World Congress on Information and Communication Technologies, pp 209-213, 2012

[8] Meera A. Thorat, Vivek S. Deshpande, “ Haldling Fairness in Wireless Sensor Network”, IEEE Conference 2012 World Congress on Information and Communication Technologies, pp 922-924, 2012.

[9] Charalabos Sergio, Vasos Vassilliou and Aristodemos Paphitis,“Hierarchical Tree Alternative Path (HTAP) algorithm for congestion control in wireless sensor nertworks”, Elsevier Ad Hoc Networks,Volume.11, Issue.1, pp 257-272, 2013.

[10] Xiaoyan Yin, Xingshe Zhou, Rongsheng Huang, Yuguag Fang and Shining Li, “ A Fairness-Aware Congestion Control Scheme in Wireless Sensor Networks”, IEEE Transactions on Vehicular Technology, Volume.58, Number. 9,pp 5225-5234,2009

[11] Almir Davis, Hwa Chang, “A SURVEY OF WIRELESS SENSOR NETWORK ARCHITECTURES” International Journal of Computer Science & Engineering Survey (IJCSES) Vol.3, No.6, December 2012