Journal of Electronics and Sensors
Review Article

Fault Tolerant Reliable Protocol (FTRP) Performance Evaluation in Wireless Sensor Networks: An Extensitive Study

Moursy IA[1*], ElDerini MN[1], Ahmed MA[2]

[1]Computer and Systems Engineering Department, Faculty of Engineering, Alexandria University, Egypt

[2]Vice Dean for Graduate Studies and Research, Faculty of Engineering, Alexandria University, Egypt

Received: 07 October, 2019**; Accepted:** 31 October, 2019**; Published:** 08 November, 2019

***Corresponding authors:** Islam Ahmed Moursy, Computer and Systems Engineering Departme -nt Faculty of Engineering, Alexandria University, Egypt. **Email:** islam.moursy@alexu.edu.eg

Abstract

Fault Tolerant Reliable Protocol (FTRP) is proposed as a novel routing protocol designed for Wireless Sensor Networks (WSNs). FTRP offers fault tolerance reliability for packet exchange and support for dynamic network changes. The key concept used is the use of node logical clustering. The protocol delegates the routing ownership to the cluster heads where fault tolerance functionality is implemented. FTRP utilizes cluster head nodes along with cluster head groups to store packets in transient. In addition, FTRP utilizes broadcast, which reduces the message overhead as compared to classical flooding mechanisms. FTRP manipulates Time to Live values for the various routing messages to control message broadcast. FTRP utilizes jitter in messages transmission to reduce the effect of synchronized node states, which in turn reduces collisions.

FTRP performance has been extensively through simulations against Ad-hoc On-demand Distance Vector (AODV) and Optimized Link State (OLSR) routing protocols. Packet Delivery Ratio (PDR), Aggregate Throughput and End-to-End delay (E-2-E) had been used as performance metrics. In terms of PDR and aggregate throughput, it is found that FTRP is an excellent performer in all mobility scenarios whether the network is sparse or dense. In stationary scenarios, FTRP performed well in sparse network; however, in dense network FTRP's performance had degraded yet in an acceptable range. This degradation is attributed to synchronized nodes states. Reliably delivering a message comes to a cost, as in terms of E-2-E. results show that FTRP is considered a good performer in all mobility scenarios where the network is sparse. In sparse stationary

scenario, FTRP is considered good performer, however in dense stationary scenarios FTRP's E-2-E is not acceptable.

There are times when receiving a network message is more important than other costs such as energy or delay. That makes FTRP suitable for wide range of WSNs applications, such as military applications by monitoring soldiers' biological data and supplies while in battlefield and battle damage assessment. FTRP can also be used in health applications in addition to wide range of geo-fencing, environmental monitoring, resource monitoring, production lines monitoring, agriculture and animals tracking. FTRP should be avoided in dense stationary deployments such as, but not limited to, scenarios where high application response is critical and life endangering such as biohazards detection or within intensive care units.

Keywords: FTRP; Fault Tolerance; Proactive Routing; Wireless Sensor Networks; NS 3; OLSR; ADV

Introduction

Although WSNs application domains are diverse, the software and hardware architectures used in those deployments tend to be very similar. WSNs are designed to support a large number of nodes [1], deployed in high density. A WSN can feature one or more sink nodes. Sink collects the data acquired by the sensor nodes and might function as gateways to external networks.

The typical hardware used are mote platforms featuring a limited computational power, locally available memory, power supply capability [2,3], a low power radio device and circuitry supporting low power standby modes to prolong the battery's lifetime.

WSNs are designed such that they operate unattended [4] for a prolonged time. This impose the below challenges, which had to be resolved for continuous operation. [4,5].

1. Limited power supply
2. Limited memory
3. Self-organization
4. Lossy wireless communication

WSNs architecture

WSNs are dynamic and can consist of various types of sensor nodes. The environment is heterogeneous in terms of both hardware and software. The sensor node construction focuses on reducing cost, increasing flexibility, providing fault tolerance and energy conservation.

The structure of sensor node consists of sensing unit (sensor and analog to digital converter (ADC)), processing unit (processor and storage), communication unit (transceive r), and power supply unit [2]. The building blocks for a typical sensor node are [5].

1. Sensing unit
2. Processing unit
3. Power unit
4. Communication unit

The communication architecture of a WSN is slightly different from the conventional computer communication and computer network. The major entities that build up the communication architecture are mentioned in [5,6]. WSN does not adhere as closely to the layered architecture of Open System Interconnection (OSI) model of conventional network. Nevertheless, the layered model is useful in WSNs for categorizing protocols. In contrast to the traditional seven layers in an OSI stack the WSN layers are reduced to the five in a TCP/IP stack, which includes the physical layer, data link layer, network layer, transport layer and application layer. Orthogonal to the five layers, Akyildiz et al. [2] defined three management planes named power, mobility and task management. These planes are responsible for monitoring the power, movement and task distribution among the

sensor nodes. These management planes help the sensor nodes to coordinate sensor tasks and minimize the overall power consumption.

WSNs applications

WSNs are designed to be the infrastructure for a broad spectrum of diverse sensing applications. Traditionally, WSNs have been used in the high-end applications such as radiation and nuclear-threat detection systems, "over-the-horizon" weapon sensors for ships, biomedical applications, habitat sensing, and seismic monitoring. Initially, applications focused on networked biological and chemical sensors for security [7].

Properties of WSN

Wireless Sensor Networks can be employed in many different contexts, with different kinds of devices and configurations. In this section, some properties of WSNs and their importance are explained.

Node configuration

The choice of nodes can heavily influence the overall network. Nodes come in many variations that make choosing one type is always a trade-off. Node prices range from low to high, depending on battery life, processing power, and sensing accuracy. Nodes vary in size from large boxes to very small particles. Such variation in size influences the available computing, storage, and transmission capacity resources. All those variations influence how long the network can be left unattended (sensing interval), also affects how often the nodes communicate, and how many nodes can be used to cover an area [8], and so on.

Deployment type

There are two types of WSN deployment: ad-hoc or planned [9]. Ad-hoc deployment is random, for example, sensors might be dropped by airplanes to cover remote areas. A planned deployment usually involves sensors placed at fixed predefined locations, such as in a factory or a building. Deployment type affects the expected node density, location, regular location patterns, and the expected degree of network dynamics.

Node mobility

Nodes may change their location after the initial deployment. A certain degree of mobility can result from the environment, such as from wind or water or from animal interference. Nodes may also be attached in a controlled manner to mobile platforms or robots, or to animals.

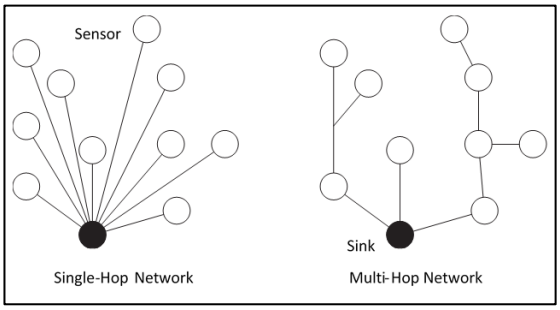
Mobility can be an accidental or desired network property. It can be intended, such as robots moving towards an area of interest, or unintended, such as wild animals moving network sensors. The degree of mobility relates to the quantity of moving sensors in the network; how often they move, and how long the distance they dislocate is.

Mobility influences networking protocols and distributed algorithms design. Routing protocols, transmission tables, and sensor accuracy are all affected by mobility.

Network topology

Network diameter is the maximum number of hops between any two nodes. It is considered one of the most important attributes of a sensor network. (Figure 1) depicts a general layout of a single-hop and a multi-hop network.

Figure (1) WSNs deployment showing the single-hop (left) and multi-hop network (right) [10].



A sensor network that forms a single-hop network is considered the simplest form. An infrastructure-based network with a single

base station forms a star network with a diameter of two. A multi-hop network may form an arbitrary graph, but often an overlay network with a simpler structure can be constructed, such as a tree or a set of connected stars.

Topology affects latency, robustness, and capacity, as well as the complexity of data routing and processing.

Processing architecture

WSN processing architecture refers to where data is processed in the network. Processing architecture can be centralized in a node with high processing power and with better power supply, usually the Sink. Processing architecture can be distributed among the nodes or, hybrid, in which pre-processing occurs in the nodes, and the heavy-duty processing is centralized.

Area coverage

Describes how much of the area of interest the sensor nodes cover. Coverage can be sparse or dense. With sparse coverage, the area of interest is partially covered; the nodes may not detect some events of interest, or may take longer to detect them. With dense coverage, the area is completely sensed, and therefore sensors are more likely to detect significant events. Degree of coverage influences data processing. Robust system design requires high coverage, yet consequently also produces a larger data set to analyze.

Node connectivity

A network can be fully connected; intermittent when nodes are sleeping; or sporadic when nodes lose communication for long periods. Connectivity influences the networking protocols and data-acquiring methods.

Network size

Network connectivity and coverage requirements, as well as the area of interest's size, determine the number of nodes participating in a sensor network. Network size in turn determines protocol and algorithm scalability requirements.

WSN Communication technologies

In the domain of WSNs, there are two main communication protocols: IPv6 over Low-power Wireless Personal Area Network (6LowPAN) [11,12]. Both protocols are built on top of the PHY layer and MAC defined in IEEE standard Lr-WPAN [13].

6LowPAN, released in 2007 by Internet Engineering Task Force (IETF), is an open standard communication protocol on how to use IPv6 on top of low power, low data rate, low cost personal area networks; it works on top of physical and MAC layers, defining how IPv6 datagrams are transmitted using Lr-WPAN [13] frames.

ZigBee is arguably as popular for a low-cost, low-power advanced communication protocol for small devices; it is widely used in Body Sensor Networks (BSNs). BSNs comprise a sensor or group of sensors attached to a patient and a coordinator for collecting raw data from the sensors. This data will be sent, analyzed, and processed by control devices through the network.

The main difference between ZigBee and 6LowPAN is the IP interoperability across different types of networks of the latter [13]. A ZigBee device requires an open 802.15.4/IP gateway to interact with an IP network while a 6LowPAN device communicates with other IP-enabled devices. Which one is chosen? That depends highly on the application [1].

Fault Tolerance in WSNs

Before fault tolerance techniques can be applied in WSN, it is necessary to understand the failure characteristics of WSNs. WSNs are inherently subject to failures besides it combines many of the difficulties of traditional embedded systems with those of traditional distributed systems, including the need for proper synchronization and fault tolerance. This poses a challenge to ambitious application goals, such as long-term, multiple-year deployments, large-scale networks of thousands of nodes, and highly reliable data

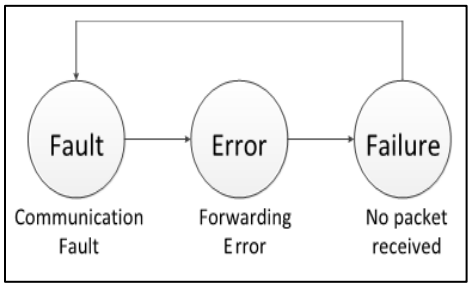
delivery. As the WSN field matures, strategies for detecting (and possibly correcting) the anomalies that are inherent to their physically coupled low-end system design will only grow in importance [14].

Tolerating faults, or maintaining fault resilience, involves prevention, detection, and recovery. Prevention involves taking actions before the faults' occurrence to reduce the probability of the network breaking. Detection describes actively monitoring the wireless network system at network, device, or data-layers. Finally, recovery amounts to overcoming a system/component failure, which can be done by replicating a system component, discarding corrupted data, or even informing an administrator of the failure's occurrence [15,16].

Fault, Error and Failure

According [17], a fault is a defect or incorrect state of the hardware or software that can occur in any part of the system. This fault is usually untraceable until an error has occurred. An error corresponds to an incorrect and unknown state of the system that can lead to failure. A failure may occur if the error is not detected and rectified. When a failure occurs, the system may deviate from its specification and unable to deliver its intended functionality. An unattended failure may later lead to further faults and errors of the system forming a chain reaction as depicted in (Figure 2). An error is the manifestation of a fault in the system, and a failure is the result of such error on the service [17].

Figure (2) Formation and manifestation mechanisms of faults, errors, and failures [17]



Type of failure in WSNs

Network-related failures

They occur when there is a communication problem among network nodes [14]. Several factors emerge to ensure proper WSN node connectivity. Nodes can have some degree of mobility, even if small, such as deploying sensors on plants, that can alter their radio link range. Radio interference can also occur, especially in urban areas. Furthermore, when a network is too dense, message collision can occur if several spatially close nodes transmit bursts at the same time.

Node-related failures

It can range from erratically behaving software, to failure of hardware, to external factors destroying the node. When deploying in a wild area, nodes can be destroyed or removed by animals and bad weather conditions. When left unattended, batteries may run out. Software bugs can also lead to node failures, such as memory leaks leading to application crashes, infinite loops, and high CPU usage, leading to quickly depletion of battery [14].

Sink-related failures

As defined [16], Sink failures occur on a higher network level. Sinks are devices that collect all the data in a certain region. Failures in these devices are especially significant, since they bridge the sensing nodes and the backend system, and are deployed in much smaller numbers. Sinks are usually high-capacity devices, with a permanent power supply or a more efficient supply.

Data-related failures

Date related failures occur when there are discrepancies in data collected by the sensing nodes or received by the sink. These failures can occur due to node hardware failure, bad sensor calibration, or, in case of reception, by transmission interference. Data failures can be perceived by establishing the relation

between nodes, and comparing in time or space their data range [14].

Detecting each kind of failure depends on how they present, i.e. what is the measurable property affected by its occurrence. Detection also depends on how quickly one needs to find the fault, or at each network layer, it needs to be detected.

Fault prevention

Fault prevention in WSNs aims to prevent failure by ensuring the network has enough sensors to provide redundancy, while continuing to cover the desired area. In other words, it aims for full coverage and connectivity. Prevention usually takes place during the design and deployment phases.

Different routing algorithms exist to suit different network topologies, such as flat, hierarchical and location-based protocols [18]. All these algorithms aim to extend the sensor network's lifetime by reducing battery usage in transmission, while not compromising data delivery.

Fault detection

To detect faults, it is needed to understand how they occur. The works of [14-16] provide an overview of faults types that occur in WSN, their common sources, and their detection mechanisms. Faults can be grouped into categories according to which network layer experiences failure when they occur. Commonly, they are grouped into network, node/device, sink, and data related faults.

Fault recovery

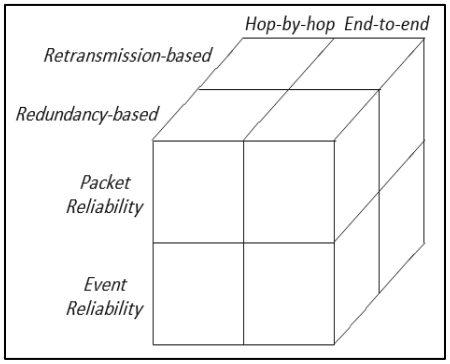
Although there are many sources for WSN system faults, recovery is mainly based on replication [16]. Divide recovery techniques into two categories: active replication and passive replication. In active replication, the replicated nodes or tasks are active before failure, and in passive replication, the replicated node or task is activated only when an instance fails. Active replication is applied in scenarios in which all or many nodes provide the same functionality. Generally, WSNs are restructured in such a way that failures or faulty nodes do not have further impact on network performance.

Reliability

Reliability in WSNs had been the focus of many existing researches. This section iterates through literature work that is related to this paper, mainly with similar schemes used in the proposed protocol which are retransmission based, connection oriented (end-to-end) and packet level reliability.

Introduced a novel three-dimensional reference model for research in WSN reliability [15].

Figure (3): A three-dimensional (3D) referen ce model for research in WSN [15].



The model categorizes WSN protocols into one of two techniques, which are retransmission or redundancy. Reliability is ensured within those techniques either by using a hop-by-hop or an end-to-end method to recover the lost data while maintaining either packet or event level reliability.

Classifies the Fault tolerance techniques according to the time at which the fault tolerance is triggered (before or after the fault occurrence) [16]. According to this, these techniques are classified as preventive (before the fault occurrence) and curative (after the fault occurrence). Hence, the proposed protocol is classified as retransmission based, connection oriented (end-to-end) and packet level reliability that utilizes a curative fault tolerant technique.

Retransmission-based reliability

Retransmission is the traditional way of ensuring reliability [15]. This is achieved by allowing the sender node to wait for ACK for its previously sent packets. In case no ACK is received, the packet is considered as lost and retransmission takes place to ensure reliability.

End-to-End (connection-oriented) Reliability

End-to-End method can be termed as connection-oriented scheme for achieving reliability in which only the two communicating end nodes (source and destination) are responsible for ensuring reliability while the intermediate nodes relay the packets between the source and the destination.

Packet level reliability

Packet level reliability ensures that all the packets carrying sensed data from all the related nodes are reliably transported to their destinations.

WSN routing protocols

Routing is the process of enabling data transfer over a network from a source to a destination machine. This is achieved in a two-step process [18]:

First, paths have to be established in the network along which data traffic can then be forwarded. Once the paths have been selected, data traffic is forwarded from one endpoint of the transmission via intermediate nodes to the other endpoint.

Routing algorithms are used to determine the paths in which the data will be transferred. The routes should be chosen such that data reaches its destination depending on the application requirements. For example, one widely used metric is using the route with the lowest end-to-end delay, or the highest throughput, while other ones could be to use the route with the least hop distance, the best link quality, or least energy consumption. More on routing metrics for WSNs can be found in [19].

An array of routing protocols for WSNs exists [15,16,20]. Flooding the data through the network is considered the most straightforward way [21]. Flooding means that every node that receives new information will forward it to all its neighbors until it reaches its destination. To prevent broadcast storms, several mechanisms are available: nodes check for duplicates, i.e. messages that have been already received, and packets may contain information on how many times they are allowed to be retransmitted. While being easy to implement, flooding causes several drawbacks: amongst others, nodes may receive duplicated messages and a large amount of energy is wasted, as there is no mechanism to include energy constraints.

Routing protocols are classified into two main categories, proactive and reactive [22], based on how and when they acquire routes in the network. Research also is conducted in developing hybrid protocols, attempting to incorporate the benefits of proactivity and reactivity.

Reactive Protocols

In reactive protocols, routes are acquired by nodes on demand when a packet needs to be forwarded and no path to the destination is currently known. The node triggers a route discovery process by issuing a route request packet through the network and then waiting for a response from the destination node. This response might take time to arrive, causing the packet delivery to be delayed. In reactive protocols, the overhead of control traffic is depending on the data traffic in the network. By acquiring routes on demand, a node has only a partial knowledge about the network, as routes are computed only for destinations to which data traffic has to be forwarded. This might be advantageous in terms of state, as reactive protocols do not require each node to store routes for the entire network. The Ad-hoc On-Demand Distance Vector (AODV) [23] is an example of a reactive protocol.

Proactive Protocols

In Proactive routing protocols, nodes regularly compute routing tables of the complete network, thus pre-provisioning all possible paths for the entire network topology. In this way, data traffic can be sent out to its destination immediately, without the delay imposed by route acquisition in reactive protocols where certain amount of control traffic is needed to keep routing tables up to date and consistent over the whole network. This control traffic is always present, independently of data traffic on the network. Amongst proactive routing protocols, Optimized Link State Routing (OLSR) [24] is a prominent example as it is used in real world deployments.

Discussion

Reactive routing protocols such as AODV [25] have the ability to discover the route when required. AODV uses the flooding mechanism to broadcast the route request to determine for new route during failure and can be very expensive to perform. AODV does not distinguish failure. It relies on the link layer feedback and the distance traverse by the packet to determine whether to broadcast for new route or drop the packet. As WSNs are prone to different failures with different durations caused by neighboring nodes, external radio devices, moving object and operating environments, nodes can suffer from transient, intermittent and permanent failure. Combinations of these failures may occur and may produce a complex unpredictable behavior that cannot be addressed with a single protocol. For example, transient failures may trigger the link layer to notify failure to the AODV and result in route discovery. When the next-hop neighbor experiencing the transient failure recovers, it will respond to the request while other nodes propagate the route request to all its local nodes. This will create a ripple effect that may congest the network. It is necessary to provide a reliable mechanism for the nodes to change their routing strategy according to the current network topology in order to re-establish the network connection. This leads to a motivation to investigate the potential of a new fault tolerant routing protocol.

Protocols such as Sensor Protocols for Information via Negotiation, (SPIN)[26] were developed to allow querying the WSN for data without being able to address particular nodes and to implement energy savings at the same time. SPIN follows an interest advertisement-request strategy in which information is described by meta-data which initially is exchanged between the nodes. Nodes, which acquired new data, advertise it via its meta-data classification. Neighboring nodes, which have an interest in that kind of data, reply with a request, on which the advertising node transmits the data to the requesting node. After receiving the new data, the requesting node advertises it to its neighbors. SPIN achieves a high-energy efficiency compared to flooding, as only requested information is transported in the network [21]. However, there is no standard meta-data format, as this is supposed to be application specific. In addition, the delivery of data is not guaranteed by SPIN's advertisement mechanism, as the nodes interested in a specific class of data might be distant from the node acquiring this data. If intermediate nodes are not interested in the given class of data, the interested node will never receive it.

To address the issues of scalability and energy preservation in a different way, the notion of hierarchy was introduced in several WSN routing protocols with the goal of avoiding an overload of sink nodes by too many received messages, as well as reducing the amount of overall message transmissions. To achieve this, nodes are grouped into clusters, which feature a node designated as cluster head. Information is relayed to this cluster head, which aggregates data to bundle the information and reduce the number of messages, which are sent to the sink nodes. With this strategy, communication is forced into a multi-hop manner, relaying information over neighboring nodes, which in turn preserves energy as the energy cost of radio communication increases with the

distance. Low Energy Adaptive Clustering Hierarchy (LEACH) [27] is one of the first routing protocols applying this strategy.

Proposed Sensor Transmission Control Protocol (STCP), an end-to-end reliability protocol with congestion control mechanism that is sink-centric [28]. STCP dynamically controls the application data flow by utilizing a controlled variable reliability mechanism where application type controls the throughput. Reliability is maintained by using ACK or Negative Acknowledgement (NACK) as an end-to-end retransmission mechanism. Packets are cached locally in each node until an ACK is received from the sink.

Whenever the sink receives information about congested paths, sink directs the downstream-congested nodes to select alternative paths. Reliability in STCP is achieved through connection-oriented explicit ACKs, which involves only the end nodes. STCP is considered scalable for large number of nodes with high hop counts from a source node to the sink. STCP nodes are prune to huge end-to-end delay time, which results in high latency and cache overflow.

Proposed a Distributed Transport for Sensor Networks (DTSN) [29]. DTSN is non-sink centric, end-to-end and an energy-oriented packet reliability protocol. DTSN is based on two mechanisms, full and differential reliability mechanisms. Full reliability is achieved via retransmission based explicit ACKs while differential reliability is performed independently. In the full reliability mechanism, the source node keeps transmitting the packets until the number of transmitted packets equals the size of the acknowledgement window. An explicit acknowledgement request is issued from the source node to the destination to confirm message delivery. If the sequence of the packets is in order, an ACK is sent. These packets are then removed from the buffer of the source node. If a NACK is received then retransmission of the missing sequence of packets is performed. The key contribution of DTSN is the integration of mechanisms involved in achieving reliability such as partial buffering at the source and intermediate nodes and erasure coding [30]. However, DTSN does not provide details about how the reliability level is maintained while network conditions change.

Proposed Fault Tolerant Reliable Protocol (FTRP)

FTRP [31] operates as a table-driven proactive protocol [22]. FTRP exchanges topology information with selected nodes of the network regularly. Initially, nodes are in learning mode and broadcast a status of not being in a sensor domain in preparation to join one. If no answer is received, node stays in that state until an answer is received. If an answer was received, the node evaluates the answer depending on its source and its included attributes. A cluster then begins to be formed according to the proposed protocol.

After cluster formation, a node sends to its designated cluster head (CH). The CH in turn decides how many copies of the message to be retained until an acknowledgment (ACK) is received from the destination. CH stores that message in one of the cluster head group (CHG) according to the protocol-defined parameters. To achieve fault tolerance, the proposed protocol utilizes retransmission-based reliability, end-to-end (connection-oriented) and packet level reliability.

Protocol overview

FTRP operations utilize a simple messaging system to communicate different protocol status through the participating nodes. This messaging system is used to transition the node from one state to another in order to form a logical grouping of nodes referenced later as a cluster. FTRP tries to overcome the issues in STCP [28] by utilizing a distributed cache rather than keeping it at the sender node. This approach allows the cluster head to control the amount of cache allocated and where to store the data packet. FTRP introduces a retry count for locally cached entries. Whenever a packet entry

reaches its max retry count, it is flushed out of the cache to overcome cache overflow. In fact, FTRP is well suited for changing environment, where its messages update the network paths and handle node failure well.

FTRP communicates using a unified packet format for all data related to the protocol. This provides an easy way to combine different messages in a single packet transmission. These packets are encapsulated into UDP [32] datagrams. On the other side, FTRP messages contain a sequence number, which is incremented for each message. In such case, the recipient of a control message is able to identify which information is more recent and to ignore those older unprocessed messages.

A common mechanism is used for populating the neighboring node information base. This is achieved by periodically exchanging of HELLO messages at predefined hello interval attribute.

FTRP Attributes

FTRP utilizes a group of attributes to facilitate cluster formation and node status updates across the WSN domain. Such attributes are summarized below:

1. FT Queue Retry count: The number of times a node tries to send its locally stored message in fault tolerance queue. Default is: 9.
2. PORT_NUMBER: Port number used for exchanging FTRP messages. Default is: 8888.
3. MAX_MSGS: The maximum number of messages a FTRP packet can carry. Default is: 64.
4. HELLO_INTERVAL: The interval for keep alive. Default is: 5.
5. MAXJITTER: The maximum value for jitter. Default is: HELLO_INTERVAL / 4.
6. MAX_SEQ_NUM: The maximum value for a sequence number. Default is: 65535.
7. DEFAULT_DOMAIN_ID: The default ID for a WSN domain in FTRP. Default is: 65 (i.e. ASCII for 'A')
8. DEFAULT_NULL_DOMAIN_ID: The default ID for a non-domain member in FTRP. Default is: 63 (i.e. ASCII for '?').
9. DEFAULT_MAX_CHG_MEMBERS: The ma ximu-m number of allowed nodes to join cluster head group. Default is: 2
10. DEFAULT_MAX_CM_MEMBERS: The max imum number of nodes in a cluster excluding the cluster head and cluster head group nodes. Default is: 3
11. DEFAULT_REPLICA_COUNT: The number of replicas of messages to be stored in cluster head groups for fault tolerance functionality. Default is: 2
12. JITTER: The jitter value used to randomize message sending. Default is: uniform Random Variable-between 0 and MAXJITTER.
13. Vtime: The time through which a message is considered valid. Default is: 2 * FTRP_HELLO_INTERVAL

FTRP operations

FTRP retransmission-based reliability

Retransmission is the traditional way of ensuring reliability [15]. This is achieved by allowing the sender node to wait for ACK for its previously sent packets. In case no ACK is received, the packet is considered as lost and retransmission takes place to ensure reliability. FTRP implementation relieves the responsibility of packet storage and retransmission to higher entity nodes (CHs, CHGs or Sinks) as it will be elaborated in the following sections.

FTRP End to-End (connection-oriented)

It is a connection-oriented scheme for achieving reliability in which only the two communicating end nodes (source and destination) are responsible for ensuring reliability. FTRP implementation expands the end-to-end reliability by relieving the source node from this task, and transfers it to the CH. The CH determines, according to the replica's parameters, which CHGs to be used as storage. Whenever the destination node

receives the packets, it broadcasts a message only processed by CHs or CHGs to release their locally stored corresponding replicas.

FTRP packet level reliability

FTRP packet level reliability ensures that all the packets carrying sensed data from all the related nodes are reliably transported to their destinations.

Definitions of main nodes status

Sink

The Sink is the central node of the network having info about all nodes. Usually, it is connected to a wired network and it has access to the wireless sensor domain.

Cluster Head (CH)

The Cluster Head can be regarded as a Sink, but for a subset of nodes. It is responsible for relaying all information from and to the nodes controlled under its domain.

Cluster Head Group (CHG)

CHGs are normal nodes selected by the CH as per the protocol parameters to act as local cluster storage for messages in transient.

Cluster Member (CM)

As the name imposes, CMs are normal nodes composing the cluster and are managed by the respective CH.

Cluster Bridge Head (CBH)

If the CH is far away from the sink, the CBH is the node within another cluster that links the cluster with the nearest CH.

Learning

Initially, all nodes other than the sink are not in a cluster or it does not know route to a sink.

Swarm

When a learning node identifies another node that is not in its domain and it has knowledge of other nodes (nonsink).

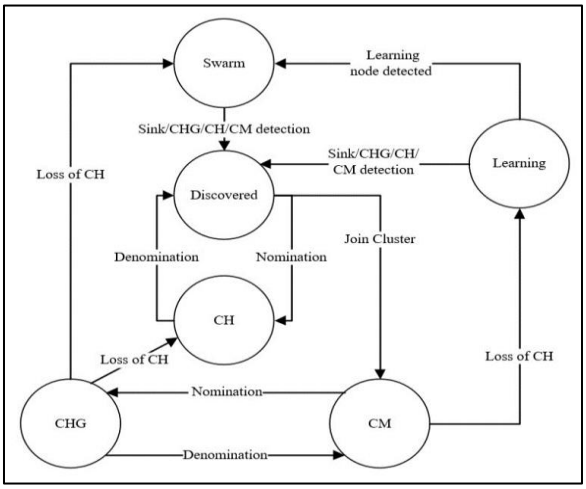
Discovered

A node is discovered from either a sink or another cluster.

FTRP state transition diagram

Node life cycle begins with a node in a Learning state. Few nodes whom have knowledge of their respective existence can form a swarm. Few swarm nodes can then transition to a discovered state upon sensing a nearby Sink. Sink can nominate a discovered node to be a CH. CH can request nearby node for association as CMs. Few CMs can then be nominated as CHGs as per the predefined configuration parameters of the protocol. (Figure 4) depicts the state transitio n for nodes in FTRP.

Figure (4): FTRP State Transition Diagram [3 3].



FTRP messaging system

Packet structure

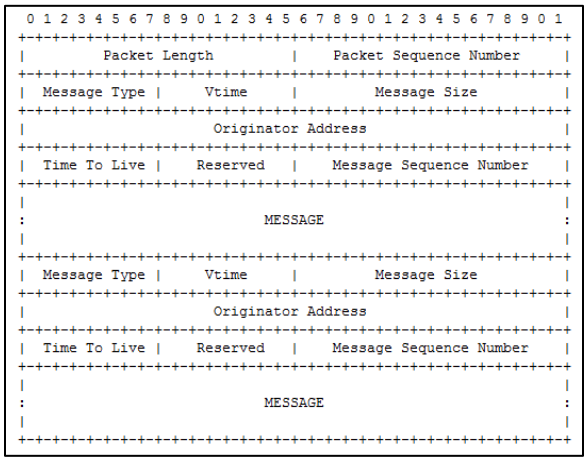
FTRP inherits the general packet structure of OLSR [24] where a packet header is appended to multiple FTRP messages and

has a sequence number. The choice of OLSR packet format was made to benefit from piggybacking multiple types of messages in the same packet.

Packet Header

The basic layout of any packet in FTRP is depicted in (Figure 5), omitting IP and UDP headers.

Figure (5): Generic FTRP packet structure



Packet Length

This field contains the full size of the packet in bytes.

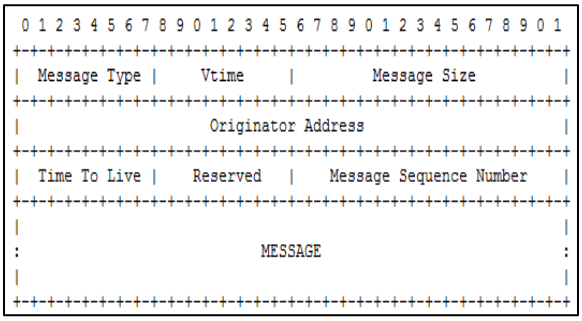
Packet Sequence Number (PSN)

PSN field is incremented by one each time a new FTRP packet is transmitted. A separate Packet Sequence Number is maintained for each interface such that packets are transmitted over an interface is sequentially enumerated. The IP address of the interface, over which a packet was transmitted, is obtainable from the IP header of the packet. If the packet contains no messages (i.e., the Packet Length is less than or equal to the size of the packet header), the packet is silently discarded.

Message header structure

A single FTRP packet can contain multiple routing messages. Messages share a common structure as well. (Figure 6) depicts the generic structure of message header structure.

Figure (6): Generic FTRP message header str ucture



Message Type

This field indicates the message type enclosed in the message. Message ranges from zero to 127, only three are used, the rest are reserved for future extensions. The three types of messages used are:

1. Hello messages
2. Association messages
3. Control messages

Validity time (Vtime)

This field indicates for how long time after reception a node MUST consider the information contained in the message as valid, unless a more recent update to the information is received.

The validity time is represented by its mantissa (the highest four bits of Vtime field) and by its exponent (the lowest four bits of Vtime field).

Message Size

This field gives the size of this message, counted in bytes and measured from the beginning of the "Message Type" field and until the beginning of the next

"Message Type" field. If there are no following messages, it is measured until the end of the packet.

Originator Address

This field contains the main address of the node, which has originally generated this message. This field SHOULD NOT be confused with the source address from the IP header, which is changed each time to the address of the intermediate interface that had re-transmitting this message. The Originator Address field is not changed in retransmissions.

Time to Live (TTL) [34]

Similar to Internet Protocol v4 (IPv4), this field contains the maximum number of hops a message will be transmitted. Before a message is retransmitted, the TTL is decremented by one. When a node receives a message with a TTL equal to zero or one, the message is retransmitted under any circumstances. Normally, a node would not receive a message with a TTL of zero. Thus, by setting this field, the originator of a message limits the flooding radius.

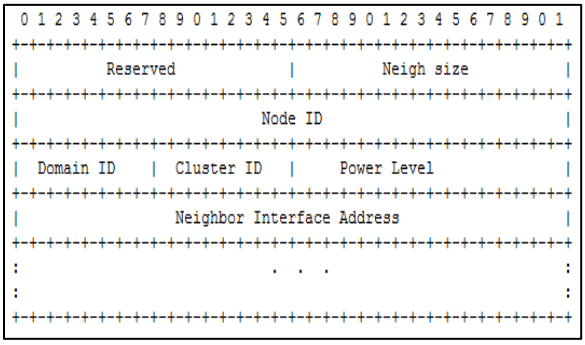
Message Sequence Number (MSN)

While generating a message, the "originator" node will assign a unique identification number to each message. This number is inserted into the Sequence Number field of the message. The sequence number is increased by one for each message originating from the node. A Wrap-around is handled as described in 0. Message sequence numbers are used to ensure that a given message is not retransmitted more than once by any node.

Hello message (HELLO)

A nonsink node lifecycle begins in a Learning state where it periodically broadcasts a hello message exposing its status and other parameters. Hello messages have its TTL value set to one in order to not flood the whole network. Hello message is populated with sending node known attributes, and it is known existing members if any. Hello message is broadcasted as keep alive periodically. The behavior of each node is different upon receiving Hello message according to the receiving node status. A Sink node receiving Hello message checks if the incoming node has not yet joined a domain, and if it is not a member of any other cluster. In that case, Sink sends an association request. If the node had already been identified in a domain yet not had joined any cluster, Sink will not take any action. This mechanism is adopted in order to control the allocation of CHs and to allow the network clustering formation to converge. Convergence is achieved by favoring the node to join a cluster than to promote it to a new CH. Sink will ignore any HELLOs from other sinks and will update the information received from any other CH. (Figure 7) depicts a generic structure for a HELLO message.

Figure (7): Generic Hello message structure



HELLO Message Structure

(Table 1) depicts the definitions of various HELLO message fields that were shown in (Figure 7).

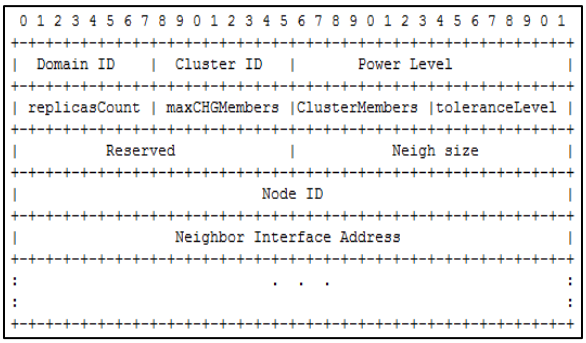
Table (1): HELLO message field's definition

Field	Description
Neighbor size	This field contains the number of neighbor interface addresses embedded in the message. This is used as reference count to ease the serialization and deserialization of the message.
Node ID	This field contains the node identifier number for the sender
Domain ID	This field contains the sender node's domain identifier, i.e. the domain ID for which the sending node is member of. It ranges from 0 to 255. None sink node initially sets its domain ID to a reserved ID equals to 63. Sink's default domain ID is 65. Network operator can set any arbitrary value for multiple sinks deployment
Cluster ID	This field contains the node's cluster identifier, i.e. the cluster ID for which the sending node is member of. It ranges from 0 to 255. Initially a node that is not yet a member of any cluster sets its cluster ID to 0. A Sink node always has a cluster ID of 255
Power Level	This filed contains the current power level of the sending node.
Neighbor Interface Addresses	This field contains a list of known interface addresses to the sending node.

Association message (ASC)

ASCs are used to instruct nodes to join a cluster or domain. Only sink and CHs are allowed to send association to other nodes. The general structure of an ASC message is depicted in (Figure 8). ASC messages have two classes, directed association and Broadcasted association.

Figure (8) Generic Association message structure



Directed Association

It is an ASC having its TTL field set to one, so that association does not flood the network.

Broadcasted association (ASCb)

Having its TTL, value set to 255 in order to a CH can be nominated when it has no direct link to the sink. It uses its nearest CBH to reach the sink through the distress Save-Our-Ship (SOS) mechanism which is illustrated in more details in section 53.7.3.2.5.

A node populates the ASC message with its members. Having that, members of a sink are the CHs known to that Sink and members of a CH are those nodes under the CH control as fault tolerance domain.

Association Message Structure

ASC messages share some field's definitions from HELLO message. Those fields are "Domain ID", "Cluster ID", "Power Level", "Neigh Size", "Node ID" and "Neighbor Interface Addresses". The remaining fields are depicted in (Table 2).

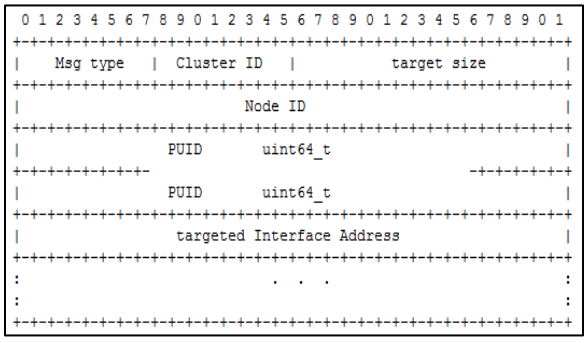
Table (2): ASC message field's definition

Field	Description
replicas Count	This field specifies the domain preconfigured parameter of how many replicas a CH has to keep in the CHG for each message it forwards
Max CHG Members	This field specifies the domain preconfigured parameter of the maximum needed nodes to join a CHG
cluster Members	This field specifies the domain preconfigured parameter of the maximum number of nodes to be a member in the same cluster, this includes the CHG nodes s
tolerance level	A reserved filed to be used upon extending the protocol to support event level reliability

Control message (CTL)

CTLs are used as decision-making mechanism and out of band, status updates of different protocol aspects. The general structure of a CTL message is depicted in (Figure 9).

Figure (9): Generic Control message structure



CTL Message has six subclasses:

1. Reject CH promotion
2. Members check
3. Release swarm members
4. Swarm release notify
5. Swarm SOS
6. Fault Tolerant message release (FT_ Release)

Control Message Structure

CTL messages share some field's definitions from HELLO message. Those fields are "Cluster ID" and "Node ID". The remaining fields are depicted in (Table 3).

Table (3): CTL message field's definition

Field	Description
Msg Type	This filed specifies the CTL sub class message. It has a range from 0 to 255 only 6 are used
target size	This field contains the number of targeted interface addresses embedded in the message. This is used as reference count to ease the serialization and deserialization of the message
PUID	This field contains a packet unique identifier provided by the NS-3 simulator. Other mechanisms such as hashing the node ID , MAC address and PSN can be used in real environment
Targeted interface addresses	This field contains a list of desired IPs that are targeted by this CTL message

Control message types

Reject CH promotion

Reject CH promotion is issued in the case when a Sink at some point in time decided to promote a CM to CH. This CM was earlier reached by another CH. In that case, rejecting the CH promotion is favored so that the CH ID pool is not depleted too fast. In return, the CM issues Reject CH Promotion control message to notify the Sink to release the allocated CH ID.

Members check

Upon node nomination to be CM or CH, it was used to be a swarm in later stage i.e. it knows of the existence of some other nodes that along they had used to form a swarm. This swarm must be checked against high entity node (Sink in case of the node is CH or CH in case the node was CM). The receiving node (Sink or CH) checks the incoming member list for local existence in its data structures, and then replies the sender node with a "Release swarm members" message for those members where higher entity does not know about.

Release swarm members

When this message is received, the node drops the sending node from its local base as swarm, and sends them swarm release notify control message.

Swarm release notify

It is processed by swarm to drop the sender from its local base.

Swarm SOS

Whenever the swarm is about to drop its last member, it issues swarm SOS to the sender of the release notify so that the sender is treated as bridgehead and relays the SOS to the sink. Sink then will send an ASCb, with its TTL value set to 255, to this swarm node to be nominated as new CH.

Fault Tolerant message release (FT_Release)

Whenever a node successfully receives its data packet, it sends this message in broadcast mode, i.e., TTL value set to 255, to notify CHs and CHGs to release the local copies of the messages considered for fault tolerance.

Message emission and jitter

To avoid synchronization of messages, jitter is introduced to allow protocol messages to be emitted such that they avoid synchronization.

Emission of protocol messages from neighboring nodes may, for various reasons (mainly timer interactions with packet processing), become synchronized such that several neighbor nodes attempt to transmit messages simultaneously. This may or may not lead to collisions and hence message loss of several subsequent messages.

To avoid synchronizations of messages, the following strategy is utilized. A node adds an amount of jitter to the interval at which messages are generated. The jitter is a random value for each message generated. Thus, for a node utilizing jitter:

Final message interval = original message interval - jitter

Where jitter is a uniform random value selected from the interval [0, original message interval/4].

Jitter is also used when a message is to be forwarded by a node. The message is kept in the node during a short time period:

Keep message period = jitter

This scheme increases the opportunit y to piggyback other messages in the same routing packet and contributes to the reduction of the overall number of packet transmissions.

Wrap-around

FTRP utilizes sequence number in PSNs and MSNs to be able to discard messages that are repeated or are received out of order. The limited number of bits (16 bit) for representing sequence numbers can cause repeated values to be present which is called a wrap-around (i.e. sequence number is incremented from the maximum possible value to zero). To be able to distinguish which sequence number is more recent. This

recovery technique was inherited from OLSR [24] by defining the following:

Sequence Number = (Sequence Number + 1) mod (MAXVALUE + 1)

Where:

MAXVALUE is the maximum value that can be held in the number of bits defined. Thus, even in the presence of wrap-around, it is possible to determine which message contains the most recent information.

Routing function and fault tolerance

The default routing or forwarding scheme for a node is to direct the outgoing packets to its master (CH in case of a node, and a Sink in case of a CH is sending). The scheme below also applies in case the CH or Sink is initiating a packet send. Upon the reception of a forward request, the routing function checks local parameters for replica count and then stores the message in the CHGs accordingly. Then finally, the packet is forwarded normally.

CHs and CHGs are using a timed queue to store the packets. The receiving node upon successful reception of a packet generates FT_Release message having the packet unique identification. Each receiving CHG, CH or Sink accepts this message and removes the requested messaged (if exists) from its local queue. Upon the expiry of queue timer, the local fault tolerance queue is checked, for packets that had not exceed their retry time, and resend those packets again. Packets, having their retry time exceeded, are removed from the queue and are considered undeliverable due to unreachable destination.

Simulation And Performance Evaluation

NS-3 overview

NS-3 is a discrete event-driven network simulator for Internet systems. It is targeted primarily for research and educational community [35]. It helps to study Internet protocols and network systems in a controlled simulation environment. NS-3 has the following features for performing simulations [36].

Performance metrics

The following performance metrics are used to analyze the behavior of FTRP.

Aggregate throughput

Aggregate throughput is defined as the sum of the throughputs in the uplink and the downlink.

Packet Delivery Ratio (PDR)

PDR is defined as the number of successfully delivered packets divided by the total number of transmitted packets during the lifetime of an experiment.

End-to-End delay (E-2-E)

E-2-E is defined as the sum of time taken for packets transmitted from sources to destinations divided by the total number of received packets.

Model validation

Had implemented and analyzed a model for AODV protocol using NS-3 simulator [37]. They performed extensive simulations for the calculation of Throughput, PDR and Control Overhead (COH) , where Control Overhead is calculated by dividing the total number of routing packets sent (includes forwarded routing packets as well) by the total number of data packets received.

A simulation model identical to that created by [37] was utilized. The experiment was repeated several times in attempt to validate the results of the AODV implementation paper with the proposed model. Same simulation scenarios were carried out.

(Figure 13) through (18) represents the various performance metrics for the proposed model versus the reference model proposed by [37]. The original reference model is represented by keywords "O2m/s" and "O1m/s" for a speed of 2 m/s and 1 m/s respectively and is represented graphically in dashed gray and black lines while the proposed model is represented by keywords "2m/s" and "1m/s" and is represented graphically in straight gray and black lines.

Scenario I – sparse network

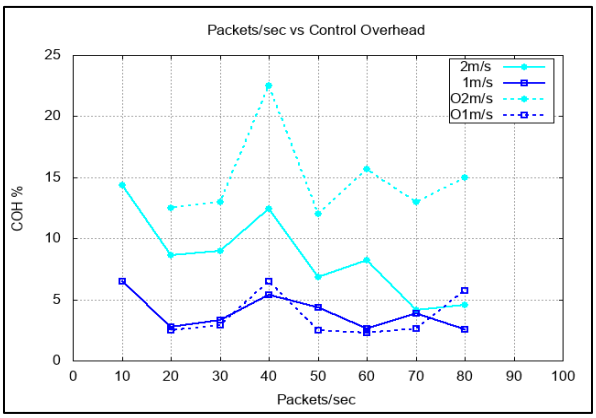
To analyze the performance of AODV in terms of throughput, PDR and COH in a sparse network comprising of 20 nodes. Simulation is performed by varying the number of data packets sent per second, while maintaining a constant number of flows, packet size and system load. Number of packets varied per flow ranges from 20 packets/sec to 80 packets/sec. Simulation was repeated using no mobility model, 1 m/sec and 2 m/sec walking models.

COH

(Figure 10) depicts the reference paper results [37] for control overhead represented in dashed lines versus the replicated simulation model represented in straight continuous lines.

Generally, it is found that obtained results have lower bounds than the reference paper results [37] for both mobility scenarios. In more details, in the mobility scenario of 2 m/s there are a considerable amount of data shifts while in lower mobility scenario of 1 m/s the data shift is minor and intersects with the reference results in some point. This can be attributed to the control mechanisms of AODV (Route Reply, Route Request and Route Error), which had many bugs at the time of the reference paper [37] implementation-on and were corrected in the used simulator version. Correction of such bugs increase the overhead of AODV messaging thus explains the lower results obtained.

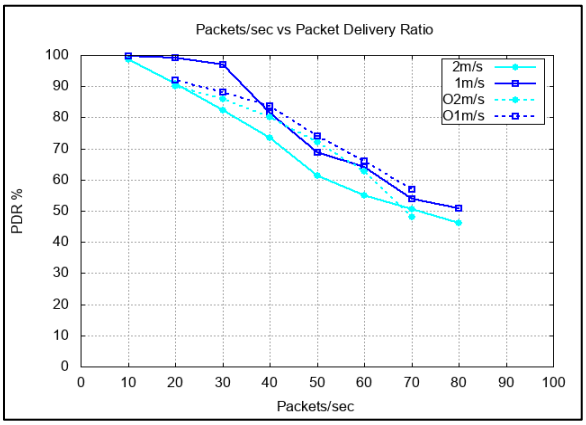
Figure (10): Model Validation Control overhead in sparse network



PDR

(Figure 11) depicts the reference paper results [37] for PDR represented in dashed lines versus the replicated simulation model represented in straight continuous lines. Overall simulated results lies below the reference paper results [37], which indicates lower PDR rate.

Figure (11): Model Validation: PDR in sparse network



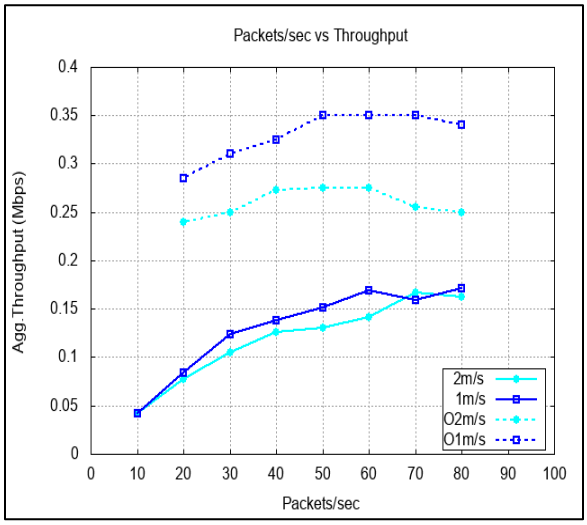
This can be attributed to the correction of bugs' number 772 and 966, which were presented at the time of the

reference paper [37] implementation and were corrected in the used simulator version as per the discussion in section. Correction of such bugs decreases the PDR ratio.

Aggregate throughput

(Figure 12) depicts the reference paper results [37] for aggregate throughput represented in dashed lines versus the replicated simulation model represented in straight continuous lines. Replicated results lies far below reference paper results [37] which can be directly related to the correction of bug number 1042, which by its correction a rate limitation is imposed for none found routes, which directly affects the overall throughput.

Figure (12): Model Validation: Aggregate throughput in sparse network



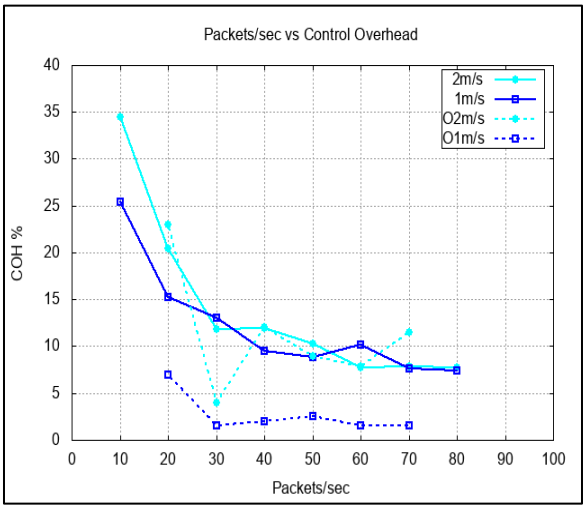
Scenario II – dense network

To analyze the performance of AODV in terms of throughput, PDR and control overhead in a dense network comprising of 40 nodes. Simulation is performed by varying the number of data packets sent per second, while maintaining a constant number of flows and system load. Number of packets varied per flow ranges from 20 packets/sec to 80 packets/sec. Simulation was repeated using no mobility model, 1 m/sec and 2 m/sec walking models.

COH

(Figure 13) depicts the reference paper results [37] for control represented in dashed lines versus the replicated simulation model represented in straight continuous lines. It is found that obtained results have higher bounds than the reference paper results [37] for both mobility scenarios. The increase in control overhead can be attributed to the synchronized nodes states where nodes can send or receive at the same time, which increases the probability of collisions and lost packets.

Figure (13): Model Validation: Control overhead in dense network

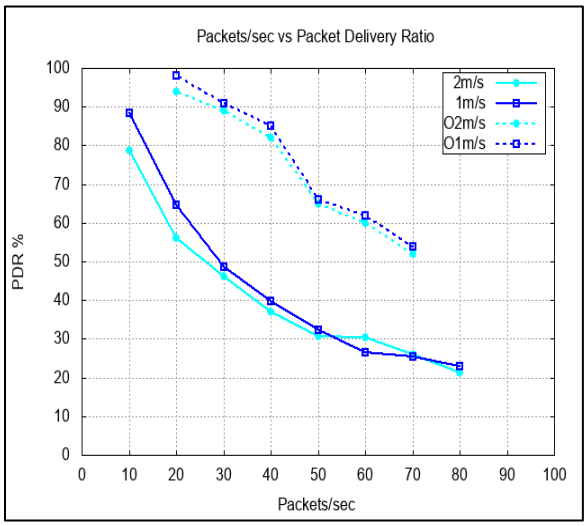


In addition, this can also be attributed to the correction of bugs related to AODV control mechanisms (Route Reply, Route Request and Route Error), which were corrected in the used simulator version. Correction of such bugs increases the overhead of AODV messaging thus explains the results obtained.

PDR

(Figure 14) depicts the reference paper results [37] for PDR represented in straight continuous lines versus the replicated simulation model represented in dashed lines. Simulated results lies below the reference paper results [37], which indicates lower PDR rate. This can be attributed to the correction of bugs' number 772 and 966, which were presented at the time of the reference paper [37] implementation and were corrected in the used simulator version. Correction of such bugs decreases the PDR ratio.

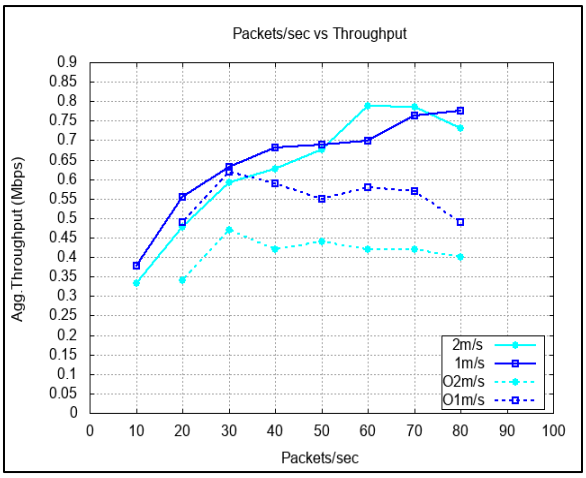
Figure (14): Model Validation: PDR in dense network



Aggregate throughput

(Figure 15) depicts the reference paper results [37] for aggregate throughput represented in dashed lines versus the replicated simulation model represented in straight continuous lines. Replicated results lies higher than the reference paper results [37], which can be directly related to the correction of bug number 1194 which directly improve the throughput of AODV in dense networks.

Figure (15): Model Validation: Aggregate throughput in dense network



Simulation assumptions

The simulation model is based on the following assumptions

1. Sink has infinite power source, while nodes have not.
2. Each node can behave as both a client and a router.
3. Each node has a single interface running FTRP protocol on that interface.
4. Nodes have same capabilities (i.e., same coverage area, same antenna).
5. Nodes are randomly placed.
6. Nodes follow a 2d-walk mobility pattern in mobility scenarios and follow constant position model for stationary simulations.
7. Nodes can either receive or transmit at a time.
8. There is no turn-around time between transmitting and receiving, nodes can switch between transmit and receive instantly.
9. Mobility is uncorrelated among the nodes and links fail independently.

Table (4): Parameters for simulation model

Simulation Parameter	Value
Simulator	NS-3 (version 3.25)
Operating system	Linux (Ubuntu 14.04)
Simulation time	50 secs
Simulation Area	100 m x 100 m
Number of nodes	20 for sparse, 40 for dense
Node transmission range	50 meters
Movement model (for mobility tests)	Random Walk 2d Mobility Model
Stationary model (for no mobility tests)	Constant Position Mobility Model
Nodes Position allocator	Random Disc Position Allocator
Speed of mobile nodes	1m/sec and 2m/sec
Traffic type	CBR
Data payload	512 bytes
Packet rates	20 p/sec to 80 p/sec
MAC Layer	802.11 DCF with RTS/CTS
Radio Frequency	2.4 GHz
Radio Channel rate	2Mbps
Propagation loss model	Friis Propagation Loss Model
Propagation Delay Model	Constant speed propagation delay model

Simulation environment

FTRP routing model is built using NS-3 network Simulator [35] on top of IEEE 802.11 MAC model of NS-3. Due to simulator limitation, model parameters had been tuned to match the 802.15.4 MAC layer. Random 2d-walk model [36] is adopted for driving mobile clients. In Random 2d-walk mobility model each instance moves with a speed and direction chosen randomly until either a fixed distance has been walked or until a fixed amount of time. If a node hits one of the boundaries (specified by a rectangle) of the model, it rebounds on the boundary with a reflexive angle and speed. This model is often identified as a Brownian motion model. The speed is varied from no mobility using constant position model, 1m/sec to 2 m/sec. (Table 4) depicts the parameters set for the simulation model that is common for all simulations.

FTRP is simulated using various networking scenarios with the help of NS-3 simulator. The scenarios and results along with detailed analysis are presented in the following sections.

Results and analysis of native FTRP protocol

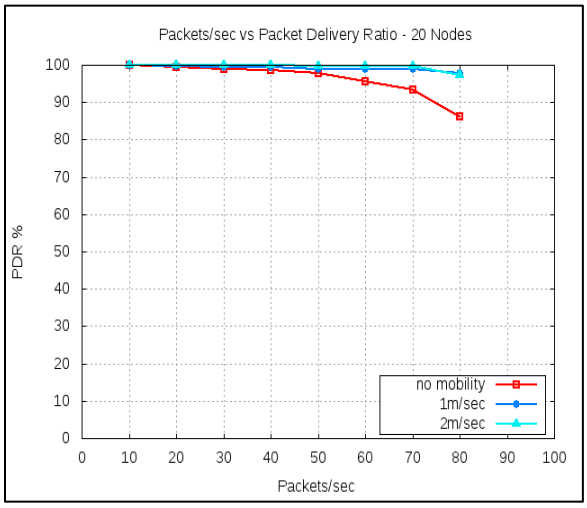
Scenario I – sparse network

To analyze the performance of FTRP in terms of throughput, PDR and E-2-E delay in a sparse network comprising of 20 nodes. Simulation is performed by varying the number of data packets sent per second, while maintaining a constant number of flows and system load. Number of packets varied per flow ranges from 20 packets/sec to 80 packets/sec. Simulation was repeated using no mobility model, 1m/sec and 2 m/sec walking models. Other parameters considered for simulations are the same as referred in (Table 4).

PDR

(Figure 16) depicts PDR against increasing traffic load in sparse network. It is observed that increasing the data rate beyond 280 kb/s (70 packets/sec) PDR begins to drop although not significant. As per the defined simulation parameters, a data rate of 240 kb/s corresponds to 60 packets/sec and a data rate of 280 kb/s corresponds to 70 packets/sec. Mobile nodes achieve a good PDR with regard to the maximum data rate supported by Lr-WPAN [13] standard, which are 250 kb/s (approximately 63 packets/sec). While nodes are stationary, the obtained PDR results fall to above 94% at the target data rate of 60 packets/sec, which are acceptable.

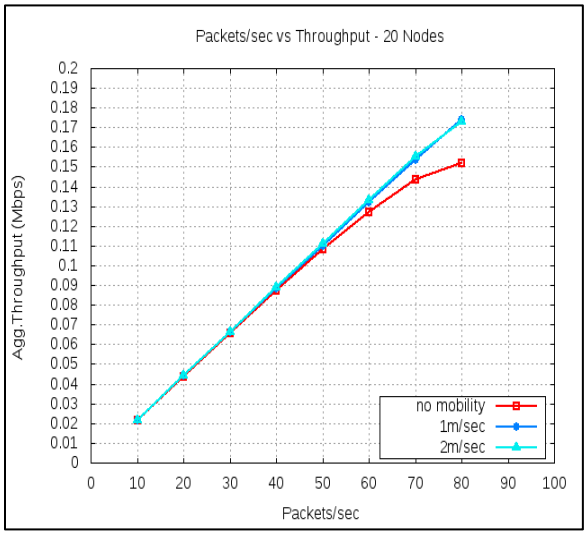
Figure (16): FTRP PDR in sparse network



Aggregate throughput

(Figure 17) depicts aggregate through put against increasing traffic load in sparse network. Overall, it is observed that throughput increases as the data rate increases. Both low and high mobility scenarios achieve good throughput as data rate increases even for data rates above the targeted 250 kb/s (approximately 63 packets/sec). Stationary nodes performance is lower than mobile ones, which can be attributed to the nodes synchronized states.

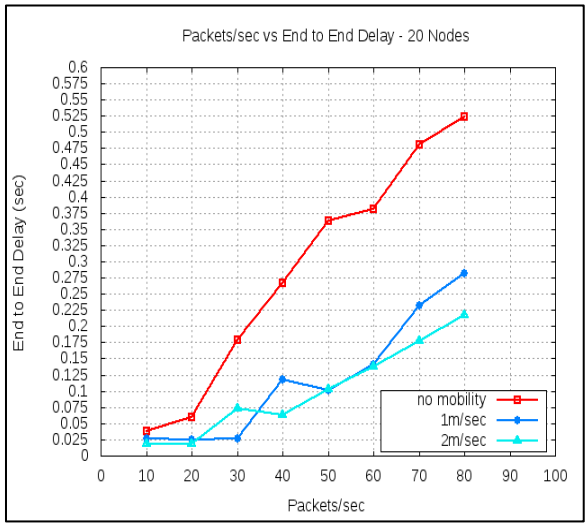
Figure (17): FTRP Aggregate Throughput in sparse network



End-to-End Delay (E-2-E)

(Figure 18) depicts E-2-E delay against increasing traffic load in sparse network. It is observed that as the data rate increases the E-2-E delay increases significantly in stationary scenario. E-2-E delay increases within acceptable range for mobile scenarios. The increase in E-2-E delay is expected due to the introduction of fault tolerance mechanism, which uses store and forward. In stationary scenario, the increase is significant and can be justified due to the nature of FTRP being too communicative. In stationary scenario, the collision rate of packets can increase, while mobility adds to decreasing collision. This can be attributed to the variations of node states. This variation reduces messages exchanged and accordingly reduces collisions and maintains good E-2-E delay.

Figure (18): FTRP End to End Delay in sparse network



Scenario II – dense network

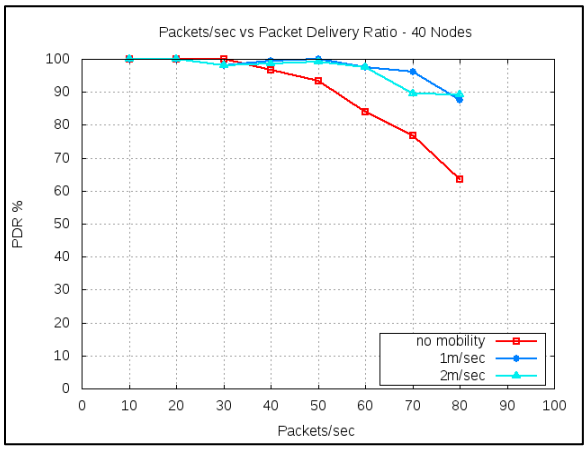
To analyze the performance of FTRP in terms of throughput, PDR and E-2-E delay in a dense network comprising of 40 nodes. Simulation is performed by varying the number of data packets sent per second, while maintaining a constant number of flows and system load. Number of packets varied per flow ranges from 20 packets/sec to 80 packets/sec. Simulation was repeated using

no mobility model, 1m/sec and 2m/sec walking models. Other parameters considered for simulations are the same as referred in (Table 4). Scenario II results emphasis the results of scenario I. It is found that in dense network with no mobility, PDR drops, aggregate throughput tends to saturate early and E-2-E delay increases significantly. In mobility scenarios, PDR is within acceptable ranges at 70 packets/sec rate, the aggregate throughput increases and E-2-E delay is within acceptable ranges.

PDR

(Figure 19) depicts PDR against increasing traffic load in dense network. It is observed that while nodes are mobile PDR is almost the same however for data rates higher than 260 kb/s (65 packets/sec) higher mobility nodes PDR tends to saturate while for less mobile nodes PDR tends to decrease. Stationary nodes are the worst performer, which can be attributed to synchronized nodes states.

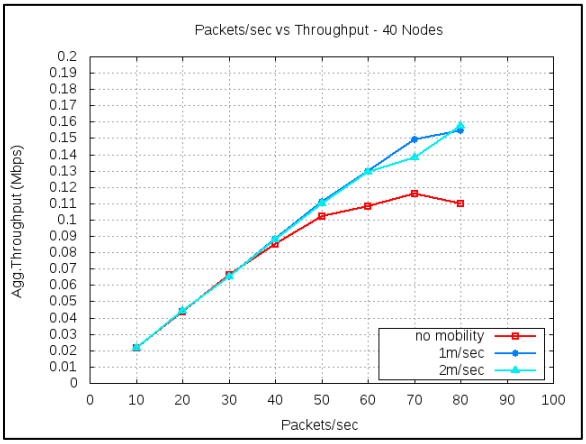
Figure (19): FTRP PDR in dense network



Aggregate throughput

(Figure 20) depicts aggregate throughput against increasing traffic load in dense network. It is observed that while nodes are mobile throughput is almost the same however, for data rates higher than 250 kb/s (approximately 63 packets/sec) higher mobility nodes' throughput tends to increase while for less mobile nodes throughput tends to saturate. Stationary nodes are the worst performer, which can be attributed to synchronized nodes states.

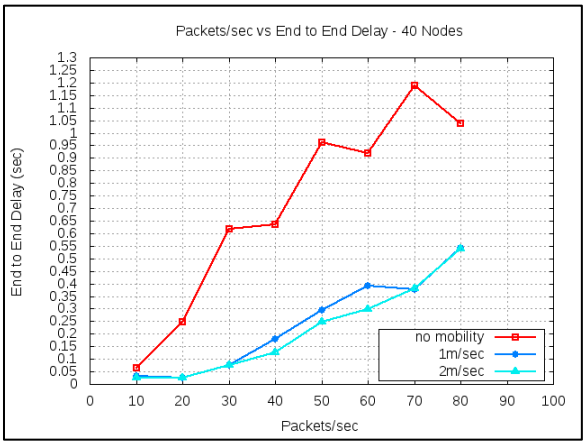
Figure (20): FTRP Aggregate Throughput in dense network



End-to-End Delay (E-2-E)

(Figure 21) depicts E-2-E delay against increasing traffic load in dense network. It is observed that while nodes are mobile E-2-E is almost the same and for data rates higher than 250 kb/s (approximately 63 packets/sec) all mobile nodes' E-2-E tends to increase. Stationary nodes are the worst performer, which is directly linked to the fault tolerance function, which for every sent packet an ACK for reception is needed to consider a packet is delivered.

Figure (21): FTRP End to End Delay in dense network



This increases the time at which a packet is considered successfully delivered. The ACK packet as well might get lost due to network collisions and synchronized nodes states, which in turn will cause the source node to resend the packet and wait for another ACK. Which significantly affects the E-2-E delay for FTRP.

Results and analysis of FTRP compared to AODV and OLSR

Stationary (no mobility)

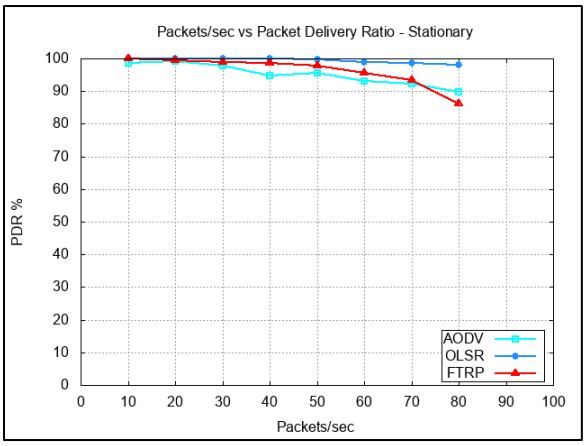
Scenario I – sparse network

To analyze the performance of FTRP against AODV and OLSR protocols in terms of throughput, PDR and E-2-E delay in a sparse network comprising of 20 nodes. Simulation is performed by varying the number of data packets sent per second, while maintaining a constant number of flows and system load. Number of packets varied per flow ranges from 20 packets/sec to 80 packets/sec. Simulation was performed using no mobility model (stationary). Other parameters considered for simulations are the same as referred in (Table 4).

PDR

(Figure 22) depicts PDR against increasing traffic load for AODV, FTRP and OLSR in sparse network with stationary nodes. It is observed that PDR performance for FTRP is average to that of AODV and OLSR for data rates lower than 250 kb/s (approximately 63 packets/sec). Data rates below 70 packets/sec are within acceptable values yet closer to AODV than OLSR. However, for data rates above 70 packets/sec FTRP PDR starts to drop. This is attributed to nodes synchronized states in FTRP when it needs to keep the topology updated at the keep alive intervals. In stationary scenarios, this does not occur in AODV as the routes are still in the nodes cache nor occur in OLSR, as its link-sensing algorithm does not receive any link drops.

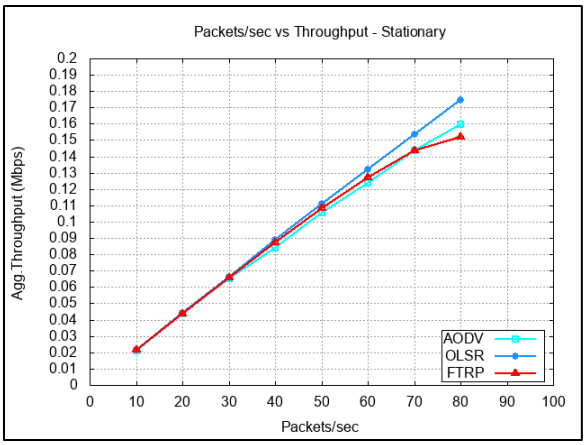
Figure (22): AODV, FTRP and OLSR - PDR in sparse network with stationary nodes



Aggregate throughput

(Figure 23) depicts aggregate through put against increasing traffic load for AODV, FTRP and OLSR in sparse network with stationary nodes.

Figure (23): AODV, FTRP and OLSR-Aggregat e throughput in sparse network with stationary nodes

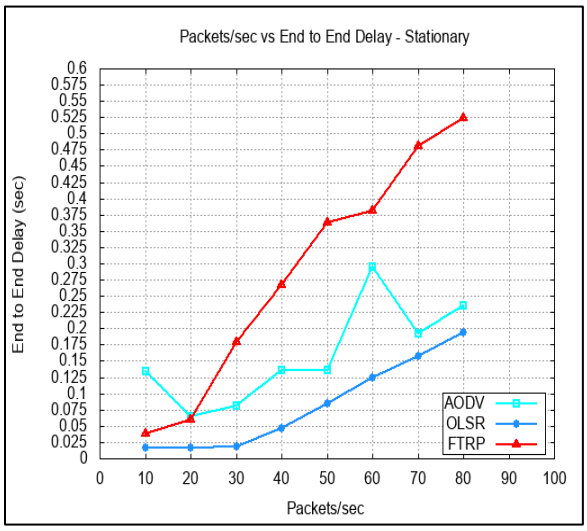


It is observed that for data rates lower than 250 kb/s (approximately 63 packets/sec), aggregate throughput performance for FTRP is closer to the of AODV and both are lower than that of OLSR. FTRP throughput tends to divert below the AODV for data rates higher than 70 packets/sec. this is attributed to both the fault tolerance function where increased checks on packet delivery take place.

End-to-End Delay (E-2-E)

(Figure 24) depicts E-2-E against increasing traffic load for AODV, FTRP and OLSR in sparse network with stationary nodes. It is observed that E-2-E performance for FTRP a worst performer compared to that of AODV and OLSR, which is directly linked to the fault tolerance function, which for every sent packet an ACK for reception is needed to consider a packet is delivered. This increases the time at which a packet is considered successfully delivered. The ACK packet as well might get lost due to network collisions, which in turn will cause the source node to resend the packet and wait for another ACK. Which significantly affects the E-2-E delay for FTRP.

Figure (24): AODV, FTRP and OLSR - End-to-End Delay (E-2-E) in sparse network with stationary nodes



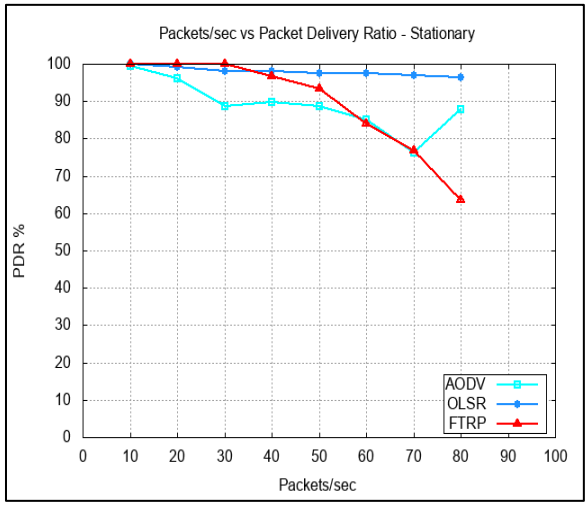
Scenario II – dense network

To analyze the performance of FTRP against AODV and OLSR protocols in terms of throughput, PDR and E-2-E delay in a dense network comprising of 40 nodes. Simulation is performed by varying the number of data packets sent per second, while maintaining a constant number of flows and system load. Number of packets varied per flow ranges from 20 packets/sec to 80 packets/sec. Simulation was performed using no mobility model (stationary). Other parameters considered for simulations are the same as referred in (Table 4).

PDR

(Figure 25) depicts PDR against increasing traffic load for AODV, FTRP and OLSR in dense network with stationary nodes. It is observed that for PDR performance for FTRP tends to decrease in a dense network for date rates higher than 200 kb/s (50 packets/sec) while nodes are stationary. Unlike AODV, which oscillates between 75% and 90%, and OLSR, which looks to be a good performer in that domain. Dense networks maximize the effect of synchronized nodes states, which significantly increase collisions and accordingly cause packet loss.

Figure (25): AODV, FTRP and OLSR - PDR in dense network with stationary nodes

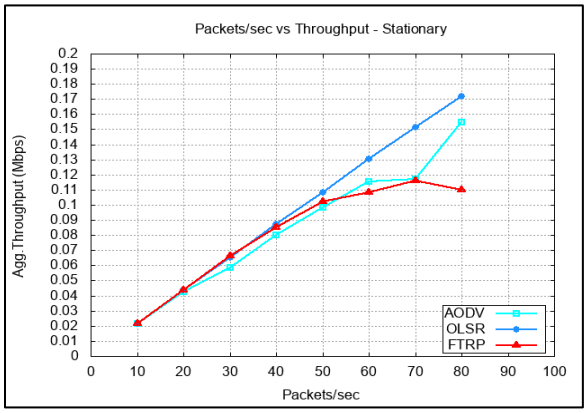


Aggregate throughput

(Figure 26) depicts aggregate throughput against increasing traffic load for AODV, FTRP and OLSR in dense network with stationary nodes. It is observed that aggregate throughput performance for FTRP is average to AODV's and OLSR's for data rates lower than 200 kb/s (50 packets/sec). For data rates between 240 kb/s (60 packets/sec) and 280 kb/s (70 packets/sec) FTRP performance is lower than that of AODV and OLSR. For higher data rates, FTRP performance tends to decrease. This is

attributed to dense network where synchronized nodes states maximize the probability of packets collisions and accordingly cause packet loss, which in turn contributes to the reduction of overall throughput.

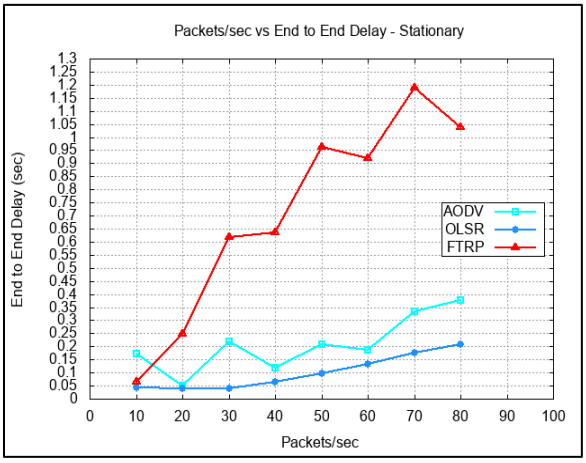
Figure (26): AODV, FTRP and OLSR - Aggregate throughput in dense network with stationary nodes



End-to-End Delay (E-2-E)

(Figure 27) depicts E-2-E against increasing traffic load for AODV, FTRP and OLSR in dense network with stationary nodes. It is observed that FTRP is a worst performer. FTRP low performance in E-2-E delay can be linked to two factors. First, it can be linked to dense network where synchronized nodes states maximize the probability of packets collisions and accordingly cause packet loss, which in turn increase retransmissions causing higher E-2-E values. Second factor is due to the fault tolerance function, which for every sent packet an ACK for reception is needed to consider a packet is delivered. This increases the time at which a packet is considered successfully delivered. The ACK packet as well might get lost due to network collisions, which in turn will cause the source node to resend the packet and wait for another ACK. Which significantly affects the E-2-E delay for FTRP. FTRP has the edge of fault tolerance in contrast to ADOV and OLSR, which they lack. Fault tolerance comes to a price of increasing E-2-E delay on favoring reliable packet delivery.

Figure (27): AODV, FTRP and OLSR - End-to-End Delay (E-2-E) in dense network with stationary nodes



As per results depicted in (Table 5), in stationary scenarios, it is evident that FTRP performed well in sparse network with regard to PDR and aggregate throughput. However, in dense network FTRP's performance had degraded yet in an acceptable range. This degradation is attributed to highly synchronized nodes states. The cost of reliably delivering a message manifested in increasing the E-2-E delay. FTRP E-2-E delay scored 2.9 times worse than best performer protocol (OLSR) which is still considered good performance. In dense stationary scenarios, FTRP E-2-E delay is not acceptable; however, it is highly dependable on the application, where some times the need to obtain every message is more important than delivering it quite a bit late.

Table (5): FTRP performance summary

	No Mobility		1m/s Mobility		2m/s Mobility	
	Sparse Network	**Dense Network**	**Sparse Network**	**Dense Network**	**Sparse Network**	**Dense Network**
PDR	Excellent (94%)	Very good (83%)	Excellent (99%)	Excellent (97%)	Excellent (100%)	Excellent (96%)
Aggregate Throughput	Excellent (94%)	Very good (81%)	Excellent (100%)	Excellent (96%)	Excellent (100%)	Excellent (94%)
End to End Delay factor	Good (2.9)	Poor (6.3)	Acceptable (1.2)	Good (3.1)	Acceptable (1.2)	Good (2.6)

Mobility of 1m/sec

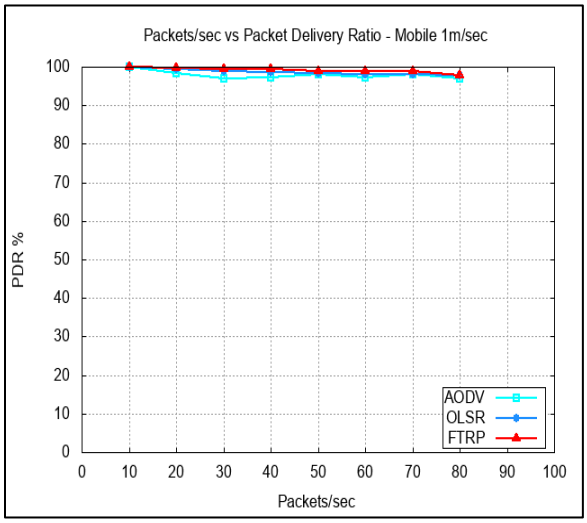
Scenario I – sparse network

To analyze the performance of FTRP against AODV and OLSR protocols in terms of throughput, PDR and E-2-E delay in a sparse network comprising of 20 nodes. Simulation is performed by varying the number of data packets sent per second, while maintaining a constant number of flows and system load. Number of packets varied per flow ranges from 20 packets/sec to 80 packets/sec. Simulation was performed using 1m/sec walking model. Other parameters considered for simulations are the same as referred in (Table 4).

PDR

(Figure 28) depicts PDR against increasing traffic load for AODV, FTRP and OLSR in sparse network while nodes are mobile with a speed of 1m/sec. it is observed that PDR performance for FTRP matches that of AODV and OLSR. It is observed that the mobility factor had contributed towards resolving the synchronized nodes states and that significantly improved the FTRP PDR performance.

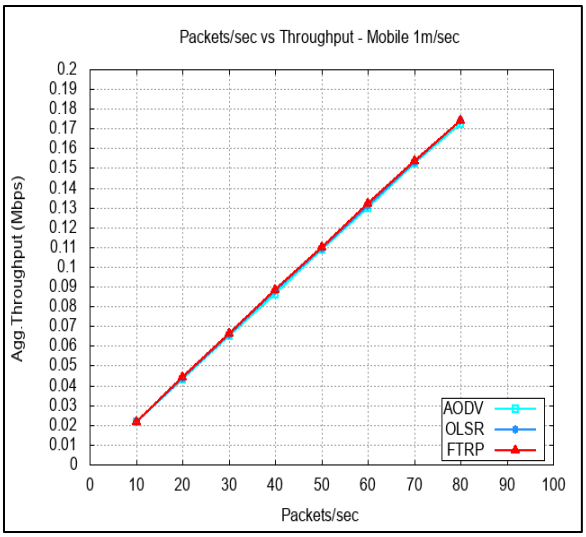
Figure (28): AODV, FTRP and OLSR - PDR in sparse network with node mobility of 1m/sec



Aggregate throughput

(Figure 29) depicts aggregate through put against increasing traffic load for AODV, FTRP and OLSR in sparse network while nodes are mobile with a speed of 1m/sec. it is observed that aggregate throughput performance for FTRP matches that of AODV and OLSR. It is observed that the mobility factor had contributed towards resolving the synchronized nodes states and that significantly improved the FTRP overall throughput performance.

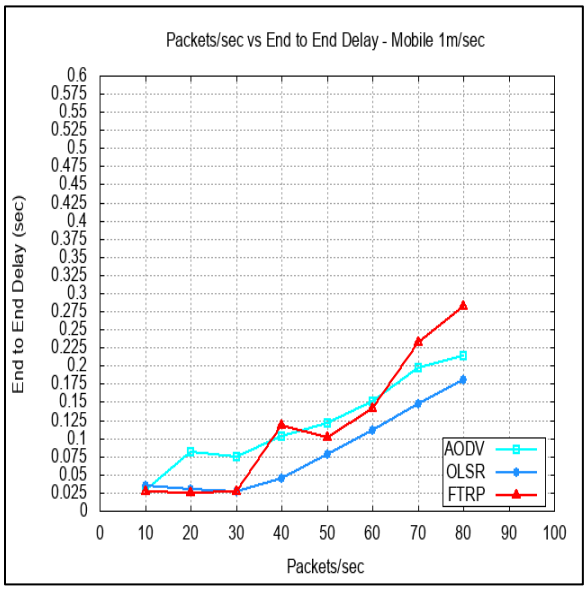
Figure (29): AODV, FTRP and OLSR-Aggregat e throughput in sparse network with node mobility of 1m/sec



End-to-End Delay (E-2-E)

(Figure 30) depicts E-2-E against increasing traffic load for AODV, FTRP and OLSR in sparse network while nodes are mobile with a speed of 1m/sec. it is observed that for data rates lower than 250 kb/s (approximately 63 packets/sec), E-2-E performance for FTRP averages that of AODV and OLSR. For higher data rates, FTRP E-2-E tends to increase. It is observed that the mobility factor had significantly contributed towards resolving the synchronized nodes states and that significantly improved the FTRP E-2-E performance compared to stationary scenario. The FTRP E-2-E results are now quite comparable to that of AODV and OLSR, being quite higher is the price for the fault tolerance introduction in FTRP, which is absent in AODV and OLSR.

Figure (30): AODV, FTRP and OLSR - End-to-End Delay (E-2-E) in sparse network with node mobility of 1m/sec



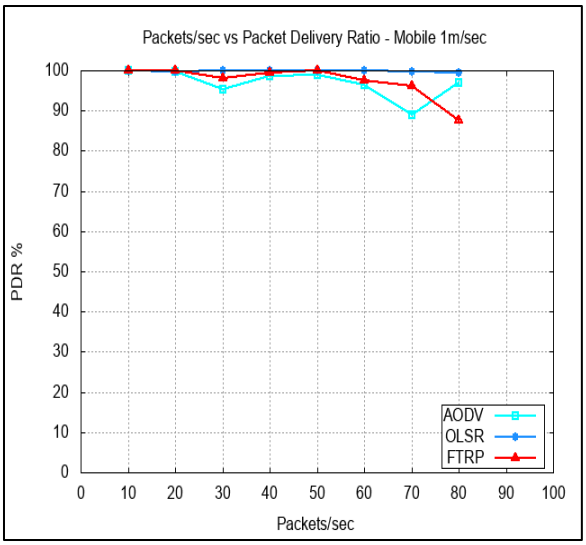
Scenario II – dense network

To analyze the performance of FTRP against AODV and OLSR protocols in terms of throughput, PDR and E-2-E delay in a dense network comprising of 40 nodes. Simulation is performed by varying the number of data packets sent per second, while maintaining a constant number of flows and system load. Number of packets varied per flow ranges from 20 packets/sec to 80 packets/sec. Simulation was performed using 1m/sec walking model. Other parameters considered for simulations are the same as referred in (Table 4).

PDR

(Figure 31) depicts PDR against increasing traffic load for AODV, FTRP and OLSR in dense network while nodes are mobile with a speed of 1m/sec. it is observed that PDR performance for FTRP matches that of AODV and OLSR for data rates lower than 280kb/s (70 packets/sec).

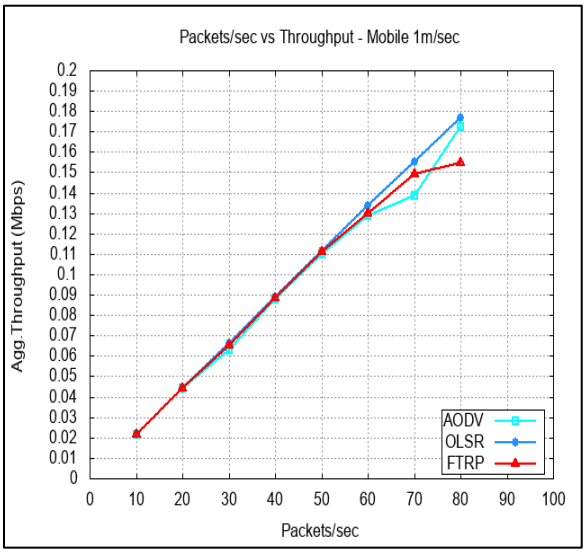
Figure (31): AODV, FTRP and OLSR - PDR in dense network with node mobility of 1m/sec



Aggregate throughput

(Figure 32) depicts aggregate through put against increasing traffic load for AODV, FTRP and OLSR in dense network while nodes are mobile with a speed of 1m/sec. it is observed that aggregate throughput performance for FTRP averages that of AODV and OLSR for data rates below 280 kb/s (70 packets/sec).

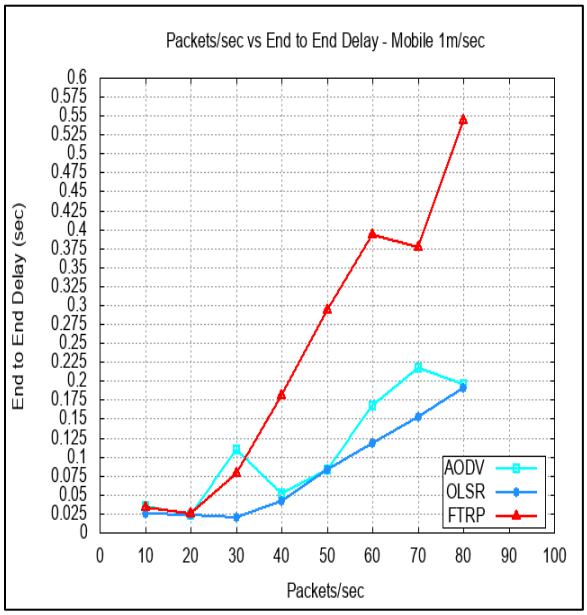
Figure (32): AODV, FTRP and OLSR - Aggregate throughput in dense network with node mobility of 1m/sec



End-to-End Delay (E-2-E)

(Figure 33) depicts E-2-E against increasing traffic load for AODV, FTRP and OLSR in dense network while nodes are mobile with a speed of 1m/sec. it is observed that E-2-E performance for FTRP is higher than that of AODV and OLSR. This can be attributed to the effect of fault tolerance function in FTRP, which for every sent packet an ACK for reception is needed to consider a packet is delivered. This increases the time at which a packet is considered successfully delivered. The ACK packet as well might get lost due to network collisions, which in turn will cause the source node to resend the packet and wait for another ACK. Which significantly affects the E-2-E delay for FTRP.

Figure (33): AODV, FTRP and OLSR - End-to-End Delay (E-2-E) in dense network with node mobility of 1m/sec



In low mobility scenarios (1m/s) and as per results depicted in (Table 5), it is evident that FTRP is an excellent performer in terms of PDR and aggregate throughput. The cost of reliably delivering a message manifested in increasing the E-2-E delay. FTRP E-2-E delay scored 1.2 times worse than best performer protocol (OLSR) which is still considered an acceptable performance. In dense stationary scenarios, FTRP E-2-E delay scored 3.1 times worse than best performer protocol (OLSR) which is still considered good performance.

Mobility of 2m/sec

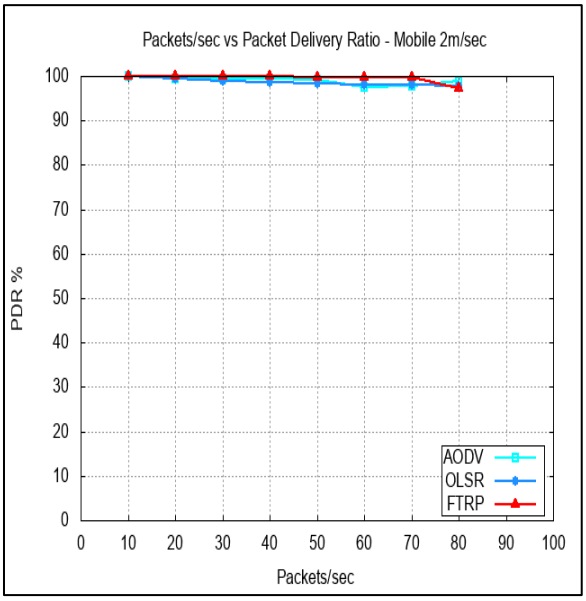
Scenario I – sparse network

To analyze the performance of FTRP against AODV and OLSR protocols in terms of throughput, PDR and E-2-E delay in a sparse network comprising of 20 nodes. Simulation is performed by varying the number of data packets sent per second, while maintaining a constant number of flows and system load. Number of packets varied per flow ranges from 20 packets/sec to 80 packets/sec. Simulation was performed using 2m/sec walking model. Other parameters considered for simulations are the same as referred in (Table 4).

PDR

(Figure 34) depicts PDR against increasing traffic load for AODV, FTRP and OLSR in sparse network while nodes are mobile with a speed of 2m/sec. it is observed that PDR performance for FTRP matches that of AODV and OLSR.

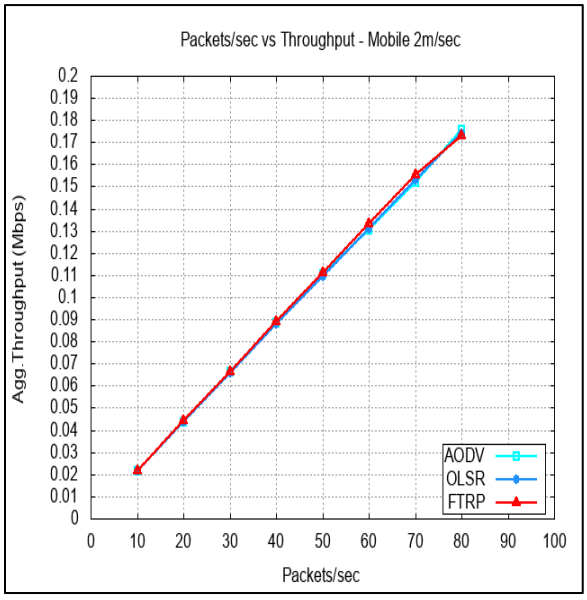
Figure (34): AODV, FTRP and OLSR - PDR in sparse network with node mobility of 2m/sec



Aggregate throughput

(Figure 35) depicts aggregate throughput against increasing traffic load for AODV, FTRP and OLSR in sparse network while nodes are mobile with a speed of 2m/sec. it is observed that aggregate throughput performance for FTRP matches that of AODV and OLSR.

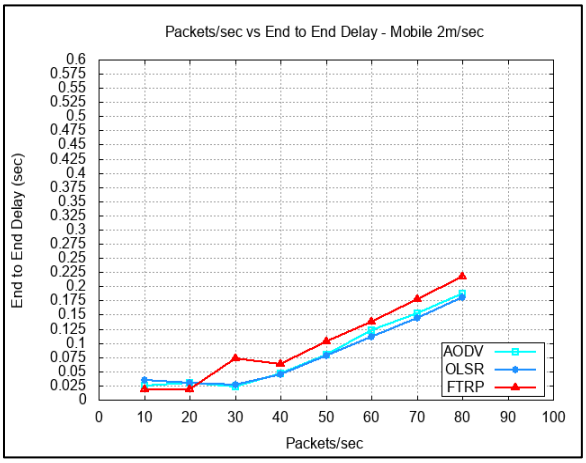
Figure (35): AODV, FTRP and OLSR-Aggregat e throughput in sparse network with node mobility of 2m/sec



End-to-End Delay (E-2-E)

(Figure 36) depicts E-2-E against increasing traffic load for AODV, FTRP and OLSR in sparse network while nodes are mobile with a speed of 2m/sec. It is observed that E-2-E performance for FTRP is slightly above that of AODV and OLSR. This can be attributed to the effect of fault tolerance function in FTRP. It is observed that the mobility factor had significantly contributed towards resolving the synchronized nodes states and that significantly improved the FTRP E-2-E performance compared to stationary scenario. The FTRP E-2-E results are now quite comparable to that of AODV and OLSR, being quite higher is the price for the fault tolerance introduction in FTRP, which is absent in AODV and OLSR.

Figure (36): AODV, FTRP and OLSR - End-to-End Delay (E-2-E) in sparse network with node mobility of 2m/sec



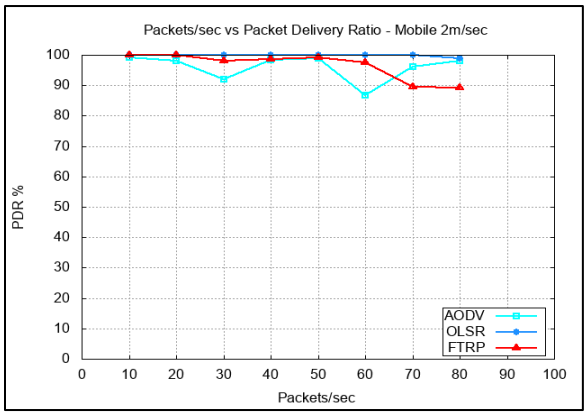
Scenario II – dense network

To analyze the performance of FTRP against AODV and OLSR protocols in terms of throughput, PDR and E-2-E delay in a dense network comprising of 40 nodes. Simulation is performed by varying the number of data packets sent per second, while maintaining a constant number of flows and system load. Number of packets varied per flow ranges from 20 packets/sec to 80 packets/sec. Simulation was performed using 2m/sec walking model. Other parameters considered for simulations are the same as referred in (Table 4).

PDR

(Figure 37) depicts PDR against increasing traffic load for AODV, FTRP and OLSR in dense network while nodes are mobile with a speed of 2m/sec. it is observed that PDR performance for FTRP averages that of AODV and OLSR for data rates lower than 250 kb/s (approximately 63 packets/sec). For higher data rates, FTRP achieves lower values than AODV and OLSR. This can be attributed to dense network, which increases the exchanged messages among nodes in addition to increased data collisions. It is also observed that the drop-in performance is still at an acceptable value (around 90%).

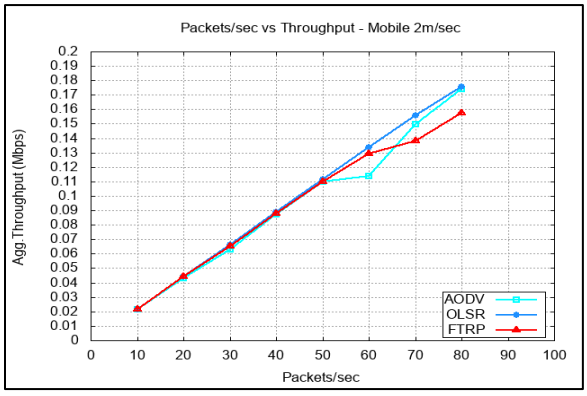
Figure (37): AODV, FTRP and OLSR - PDR in dense network with node mobility of 2m/sec



Aggregate throughput

(Figure 38) depicts aggregate throughput against increasing traffic load for AODV, FTRP and OLSR in dense network while nodes are mobile with a speed of 2m/sec. It is observed that aggregate throughput performance for FTRP averages that of AODV and OLSR for data rates lower than 240 kb/s (60 packets/sec). For higher data rates, FTRP aggregate throughput performance is slightly lower than that of AODV and OLSR. As throughput is linked to PDR, this can also be attributed to dense network, which increases the exchanged messages among nodes in addition to increased data collisions. It is also observed that the slight lag in performance is still at an acceptable value and it tends to be sustained.

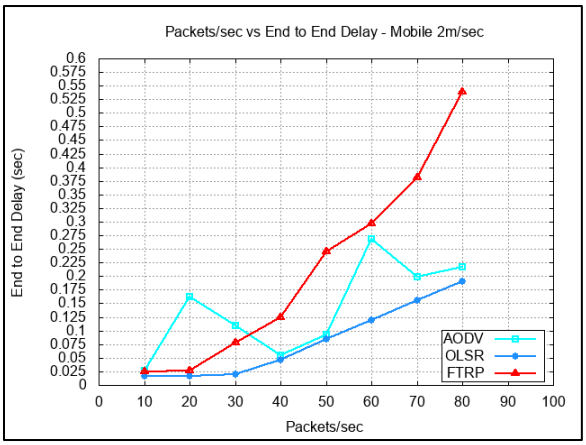
Figure (40): AODV, FTRP and OLSR - Aggregate throughput in dense network with node mobility of 2m/sec



End-to-End Delay (E-2-E)

(Figure 39) depicts E-2-E against increasing traffic load for AODV, FTRP and OLSR in dense network while nodes are mobile with a speed of 2m/sec. It is observed that E-2-E performance for FTRP is slightly above that of AODV and OLSR but better than the 1 m/sec scenario discussed in subsection 4.7.2.2.3. This can be attributed to the effect of fault tolerance function in FTRP. In addition to the increased mobility attributes which contributes to reduction of the effect of synchronized nodes states.

Figure (39): AODV, FTRP and OLSR - End-to-End Delay (E-2-E) in dense network with node mobility of 2m/sec



In high mobility scenarios (2m/s) and as per results depicted in (Table 5), it is evident that FTRP is an excellent performer in terms of PDR and aggregate throughput. The cost of reliably delivering a message manifested in increasing the E-2-E delay. FTRP E-2-E delay scored 1.2 times worse than best performer protocol (OLSR) which is still considered an acceptable performance. In dense stationary scenarios, FTRP E-2-E delay scored 2.1 times worse than best performer protocol (OLSR) which is still considered good performance. It is also clear that FTRP performance in terms of E-2-E delay is better in high mobility scenarios than in low mobility scenarios.

Discussion

FTRP performance had been analyzed in comparison to AODV and OLSR protocols. AODV and OLSR were chosen for being widely supported by industry leading mote manufactures such as ZigBee [12] and open source WSN operating system such as TinyOS [38]. Moreover, for AODV being a prominent example for reactive protocol and OLSR being a prominent example for proactive protocols in WSNs.

FTRP, AODV and OLSR performance has been evaluated through extensive simulation using NS-3. PDR, aggregate throughput and end-to-end delay had been used as performance metrics.

Throughout simulation results iterated in this section, FTRP had exhibited some good and weak features compared to AODV and OLSR. Moreover, FTRP surpasses AODV and OLSR by implementing fault tolerance feature, which AODV and OLSR lack. Table) depicts the performance summary for FTRP where PDR percentages are directly extracted from the plots, Aggregate throughput score represents how FTRP is compared to the best performer and E-2-E delay factor represents how many times FTRP is worse than the best performer. E-2-E scores ranged from 1 to 2.1 times worse are graded good, scores ranged from 2.1 to 3.2 times worse are graded acceptable, scores ranged from 3.2 to 4.3 times worse are graded fair, scores ranged greater than 4.3 worse are graded poor. Performance was scored with regard to PDR and aggregate throughput, values that ranged from 90% to100% are graded Excellent, values ranged from 80% to 89% are graded Very good, values ranged from 70% to 79% are graded Good, values ranged from 60% to 69% are graded Accepted.

In terms of PDR and aggregate throughput, it is found that FTRP is an excellent performer in all mobility scenarios whether the network is sparse or dense. In stationary scenarios, FTRP performed well in sparse network; however, in dense network FTRP's performance had degraded yet in an acceptable range. This degradation is attributed to highly synchronized nodes states.

As reliably, delivering a message comes to a cost. It is evident that by implementing the fault tolerance feature in FTRP paid its price by increasing the E-2-E delay. As per the E-2-E delay results summary depicted in (Table 5), FTRP is considered a good performer in all mobility scenarios whether the network is sparse or dense as it had scored from 1.2 times worse than the best performer protocol (OLSR) to 3.1 times. In sparse stationary scenario, FTRP E-2-E delay scored 2.9 times worse than best performer protocol (OLSR) which is still considered good performance, however in dense stationary scenarios FTRP E-2-E delay is not acceptable.

FTRP E-2-E delay results can be attributed to the way fault tolerance is handled in FTRP. In FTRP a node, sending messages directs it to its CH as its default gateway; the CH then decides the route to destination after storing the message in one of the CHGs as per the FTRP parameters. Recipient node, upon successfully receiving the message sends a broadcast FT_Release message, which is caught by the CHGs and sender node. CHGs and sender node use FT_Release to release its locally stored message. In other words, a packet is not considered successfully delivered until an FT_Release is received by the sender. There are situations when actually, the packet is well delivered but the FT_Release message is lost due to collisions.

That makes FTRP suitable for wide range of WSNs application domains, such as military applications by monitoring soldiers' biological data and supplies while in battlefield, battle damage assessment. FTRP can also be used in health applications by tracking and monitoring doctors and patients inside a hospital and elderly assistance. In addition to wide range of geo-fencing, environmental monitoring, resource monitoring, production lines monitoring, agriculture and animals tracking.

FTRP should be avoided in dense stationary deployments such as, but not limited to, scenarios where high application response is critical and life endangering such as biohazards detection or within intensive care units.

Conclusion and Future Work

Conclusion

This paper introduced a novel reliable fault tolerant routing protocol, FTRP, for wireless sensors network. FTRP creates a communication path between source and destination nodes and forward packets on that path. FTRP offers fault tolerance reliability for packet exchange as well as adaptation for dynamic network changes. The key concept used in this protocol is the use of node logical clustering. The protocol delegates the routing ownership to the cluster heads where fault tolerance functionality is implemented. FTRP utilizes cluster head nodes along with cluster head groups as intermediate storage for packets in transient. In addition, FTRP utilizes broadcast in its routing messages communication. This technique noticeably reduces the message overhead as compared to classical flooding mechanisms. FTRP manipulates TTL values for the various routing messages as well as utilizing jitters in messages transmission.

FTRP performance has been evaluated through extensive simulations using NS-3 against AODV and OLSR protocols. Aggregate throughput, PDR and E-2-E delay have been used as performance metrics to compare and analyze performance of FTRP routing protocol. Throughout simulation results, FTRP had exhibited some good and weak features compared to AODV and OLSR. Moreover, FTRP surpasses AODV and OLSR by implementing fault tolerance feature, which both AODV and OLSR lack. In terms of packet delivery ratio and aggregate throughput, it is found that FTRP is an excellent performer in all mobility scenarios as simulated by Random 2d-walk model whether the network is sparse or dense. In stationary scenarios, FTRP performed well in sparse network; however, in dense network FTRP's performance had degraded yet in an acceptable range. Reliably delivering a message comes to a cost. It is evident that by implementing the fault tolerance feature in FTRP paid its price by increasing the end-to-end delay. In terms of end-to-end delay, results show that FTRP is considered a good performer in all mobility scenarios where the network is sparse. In sparse stationary scenario, FTRP is considered good performer, however in dense stationary scenarios FTRP performance termed as worst-case behavior, which can be attributed to synchronized nodes states that occurs when nodes send similar messages at the same time.

There are times when properly receiving a network message carrying crucial information is more important than other costs such as, but not limited to, energy or delay. That makes FTRP suitable for wide range of WSNs application domains, such as military applications by monitoring soldiers' biological data and supplies while in battlefield, battle damage assessment. FTRP can also be used in health applications by tracking and monitoring doctors and patients inside a hospital and elderly assistance, in addition to wide range of geo-fencing, environmental monitoring, resource monitoring, production lines monitoring, agriculture and animals tracking. FTRP should be avoided in dense stationary deployments such as, but not limited to, scenarios where high application response is critical and life endangering such as biohazards detection or within intensive care units.

Future Work

As future work, it is planned to improve the performance of FTRP in stationary scenarios. Furthermore, the energy efficiency has to be evaluated for various FTRP operations.

FTRP performance was evaluated through simulations. Extending FTRP implementation in a WSN operating system to compare the complexity of a real system

against the simulation results should be investigated.

Techniques for packet storage optimization in forward queues used for fault tolerance function should be investigated.

The effect of varying the number of attempts to retransmit a non-delivered packet (max retry count) should be investigated.

For the time being, FTRP supports only IPv4. Future directions should adapt IPv6. Moreover, the good performance achieved by FTRP in mobility scenarios open the doors to adapt the proposed protocol to Mobile Ad-hoc Networks (MANET).

Last but not least, FTRP protocol can be tailored to support event-based reliability in the addition to the already implemented packet level reliability.

References

1. Buratti C, Conti A, Dardari D, et al. (2009) An Overview on Wireless Sensor Networks Technology and Evolution. Sensors; 9(9): 6869-6896.
2. Akyildiz IF, Su W, Sankarasubramaniam Y (2002) A Survey On Sensor Networks. IEEE Commun Mag; 40(8): 102-114.
3. Mottola L, Picco GP (2012) Middleware For Wireless Sensor Networks: An Outlook. J Med Internet Res; 3(1): 31-39.
4. Yick J, Mukherjee B, Ghosal D (2008) Wireless Sensor Network Survey, Computer Networks. The International Journal of Computer and Telecommunications Networking; 52(12): 2292-2330.
5. Ahmed M, Huang X, Sharma D, et al. (2012) Wireless Sensor Network: Cherecterestics and Architectures. International Journal of Electrical, Computer, Energetic, Electronic and Communication Engineering; 6(12): 660–663.
6. Tilak S, Abu-Ghazaleh N, Heinzelman W (2002) A Taxonomy Of Wireless Micro-Sensor Network Models. SIGMOBILE Mob. Comput. Commun; 6(2): 28–36.
7. Sohraby K, Minoli D, Znati T (2007) Wireless Sensor Networks: Technology, Protocols, and Applications. Wiley.
8. Beutel J, Romer K, Ringwald M, et al. (2009) Deployment Techniques for Wireless Sensor Networks, in Sensor Networks: Where Theory Meets Practice. Springer; 219–248.
9. Cecilio J, Furtado P (2014) Wireless Sensors in Heterogeneous Networked Systems. Springer.
10. Dargie W, Poellabauer C (2010) Fundamentals of Wireless Sensor Networks: Theory and Practice. Wiley.
11. "IPv6 over Low power WPAN 6lowpan)," IEFT. [Accessed 15 05 2017].
12. "ZigBee," ZigBee, [Online]. Available: http://www.zigbee.org/. [Accessed 15 5 2017].
13. "IEEE Standard for Low-Rate Wireless Networks," 2016. [Online]. Available: http://standards.ieee.org/getieee802/download/802.15.4-2015.pdf. [Accessed 7 7 2017].
14. Jurdak R, Wang XR, Obst O, et al. (2011) Wireless Sensor Network: Anomalies Diagnosis and Detection Strategies. Intelligence-Based Systems Engineering; 309-325.
15. Mahmood MA, Seah WK, Welch I (2015) Reliability In Wireless Sensor Networks: A Survey And Challenges. IJCCN; 79: 166-187.
16. Chouikhi S, ElKorbi I, Ghamri-Doudanec Y, et al. (2015) A Survey On Fault Tolerance In Small And Large Scale Wireless Sensor Networks. The International Journal for the Computer

and Telecommunications Industry; 69: 22-37.

17. Avizienis A, Laprie JC (1986) Dependable Computing: From Concepts To Design Diversity. Proceedings of the IEEE; 74(5): 629-638.
18. Misra S, Woungang I, Misra SC (2009) Guide to Wireless Sensor Networks. Springer.
19. Vasseur J, Systems C, Kim M, et al. "Routing Metrics Used for Path Calculation in Low-Power and Lossy Networks," IETF, December 2010. [Online]. Available: https://tools.ietf.org/html/draft-ietf-roll-routing-metrics-14. [Accessed 11 5 2017].
20. Ajith Kumar SA, Ovsthus K, Kristense LM (2014) An Industrial Perspective on Wireless Sensor Networks - A Survey of Requirements, Protocols and Challenges. IEEE Communications Surveys & Tutorials; 16(3): 1391-1412.
21. Akkaya K, Younis M (2005) A Survey On Routing Protocols For Wireless Sensor Networks. Ad Hoc Networks; 3(3): 325–349.
22. Shivahare BD, Wahi C, Shivhar S (2012) Comparison Of Proactive And Reactive Routing Protocols In Mobile Adhoc Network Using Routing Protocol Property. International Journal of Emerging Technology and Advanced Engineering; 2(3): 356-359.
23. "RFC 3561 - Ad hoc On-Demand Distance Vector (AODV) Routing," IETF. [Accessed 11 5 2017].
24. "RFC 3626 - Optimized Link State Routing Protocol," IETF. [Accessed 11 5 2017].
25. Perkins CE, Royer EM (1999) Ad-hoc On-Demand Distance Vector Routing. in WMCSA.
26. Kulik J, Rabiner W, Balakrishnan H (1999) Adaptive Protocols For Information Dissemination In Wireless Sensor Networks. in the 5th annual ACM/IEEE international conference on Mobile computing and networking; 174-185.
27. Handy MJ, Haase M, Timmermann D (2002) Low Energy Adaptive Clustering Hierarchy With Deterministic Cluster-Head Selection. in 4th International Workshop on Mobile and Wireless Communications Network; 368-372.
28. Iyer Y, Gandham S, Venkatesan S (2005) STCP: A Generic Transport Layer Protocol for Wireless Sensor Networks. in Computer Communications and Networks; 35-42.
29. Marchil B, Grilo A, Nunes M (2007) DTSN: Distributed Transport for Sensor Networks. in 12th IEEE Symposium on Computers and Communications.
30. Albano M, Chessa S (2015) Replication vs. Erasure Coding In Data Centric Storage For Wireless Sensor Networks. The International Journal of Computer and Telecommunications Networking; 77: 42-55.
31. Moursy A, ElDerini MN, Ahmed MAE (2017) FTRP: A Fault Tolerant Reliable Protocol for Wireless Sensor Networks. in The Eleventh International Conference on Sensor Technologies and Applications; 24-30.
32. Moursy A, ElDerini MN, Abd-Elazim MA (2017) A Novel Routing Fault Tolerant Reliable Protocol for Wireless Sensor Networks. Sensors and Transducers; 218(12): 10-18.
33. "RFC 791 - Internet Protocol," IETF. [Accessed 07 07 2017].
34. Henderson T (2017) "NS-3 Overview," [Accessed 11 5 2017].

35. "NS-3 Manual,". [Accessed 11 5 2017].
36. Paul AB, Shantanu Konwar UG, Chakraborty A, et al. (2010) Implementation and Performance Evaluation of AODV in Wireless Mesh Networks using NS-3. in 2nd International Conference on Education Technology and Computer (ICETC).
37. "TinyOS Documentation,". [Accessed 28 6 2017].
38. Alcaraz C, Najera P, Lopez J, et al. (2010) Wireless Sensor Networks and the Internet of Things Do We Need a Complete Integration. in International Workshop on the Security of the Internet of Things.
39. "RFC 768 - User Datagram Protocol - IETF," [Accessed 11 5 2017].
40. Bokare M, Ralegaonkar A (2012) Wireless Sensor Network: A Promising Approach for Distributed Sensing Tasks. Excel Journal of Engineering Technology and Management Science; 1(1): 1-9.
41. Romer, Zurich E, Mattern F (2004) The Design Space Of Wireless Sensor Networks. IEEE Wirel Commun; 11(6): 54-61.
42. Borges Margi C, Lu X,Zhang G, et al. (2006) Meerkats: A Power-Aware, Self-Managing Wireless Camera Network For Wide Area Monitoring. in Workshop on Distributed Smart Cameras; 20-26.
43. Alcaraz C, Najera P, Lopez J, et al. (2010) Wireless Sensor Networks and the Internet of Things Do We Need a Complete Integration. in International Workshop on the Security of the Internet of Things; 32-37.
44. Boukerche A, Pazzi RWN, Araujo RB (2006) Fault-Tolerant Wireless Sensor Network Routing Protocols for the Supervision of Contex-Aware Physical Environments. J Parallel Distrib Comput; 66(4): 586-599.
45. Chipara O, He Z, Xing G, et al. (2006) Real-time Power-Aware Routing in Wireless Sensor Networks. in 14th IEEE Workshop Quality of Service (IWQoS'06); 83–92.
46. Zeng Y, Sreenan CJ, Sitanayah L, et al. (2011) An Emergency-Adaptive Routing Scheme for Wireless Sensor Networks for Building Fire Hazard Monitoring. Sensors (Basel); 11(3): 2899-2919.
47. Ye W, Heidemann J, Estrin D (2002) An Energy-Efficient MAC Protocol for Wireless Sensor Networks. in 21st Ann. Joint Conf. IEEE Computer and Communications Societies (INFOCOM'02); 1567-1576.
48. Levis P, Brewer E, Culler D, et al. (2008) The Emergence Of A Networking Primitive In Wireless Sensor Networks. Communications of the ACM; 51(7): 99-106.
49. "NS-3 Change Log," [Online]. Available: https://www.nsnam.org/~pdbarnes/docs-related/changes.html. [Accessed 21 6 2017].
50. Moursy A, ElDerini MN, Abd-Elazim MA (2017) FTRP: A Fault Tolerant Reliable Protocol for Wireless Sensor Networks. in The Eleventh International Conference on Sensor Technologies and Applications (SENSORCOMM 2017).