International Journal of Grid and Distributed Computing
Vol. 11, No. 10 (2018), pp.75-92
http://dx.doi.org/10.14257/ijgdc.2018.11.10.07

Supporting High Data Rate Sensing Applications in Internet of Things (IoT)

Raees Khan[1] and Adnan Noor Mian[2]

[1]Department of CS, FAST-National University of Computing and Emerging0020Sciences, Lahore
[2]Department of CS, Information Technology University (ITU), Lahore
[1]raees.khan@nu.edu.pk, [2]adnan.noor@itu.edu.pk

Abstract

Wireless sensor networks are main enabling resource for realization of Internet of Things (IoT) to access the physical world information. A number of dynamic IoT applications with dense sensor deployments generate continuous and high rate data streams. Such a sensor network pose congestion and data loss challenges due to contention near the sink and therefore needs transport layer services. In this work, we have proposed integrated congestion control scheme with reliability module at transport layer for high data rate sensor networks used in IoT environments. The proposed modules of the protocol are mainly implemented at the resourceful sink/gateway node and perform end-to-end reliable data delivery alongside proactive congestion avoidance for data collection in high data rate IoT applications. The proposed modules have been evaluated through extensive ns-2 simulations and protocol results are compared with three other well-known transport layer protocols i.e., NewReno, Westwood and ELFN. Comparative analysis of simulation results reveals that proposed protocol outperform its competitors on all selected performance measures. The end-to-end delay performance of proposed protocol is about 3 times better and its normalized control load (NCL) is less than 0.005 in comparison to 1 NCL by other sensor networks enhanced variants. Moreover, it effectively sends data packets generated by the sensing nodes with 99.85% packet delivery ratio (PDR) whereas competing protocols have 95% PDR in comparison.

Keywords: *Transport Layer; Congestion Avoidance; Reliability; High Data Rate; Sensing Applications*

1. Introduction

The term Internet of Things (IoT) is used to refer to billions of internet connected physical devices, including industrial manufacturing sensors, environmental and strategic surveillance, smart systems, structural monitoring and complex medical and mechanical machines. These also include non-internet enabled IoT devices like those in sensor network mode that connect through a gateway acting as a bridge. Such a Wireless Sensor Network (WSN) is a valuable resource for realization of Internet of Things (IoT) vision by providing a virtual layer for accessing the physical world information.

IoT sensing applications are formed by densely deployed, interconnected small devices having abilities to obtain, process, and transmit data cooperatively to a gateway node. These sensor networks are more likely to be installed in environments with inconsistent and extremely weak channel conditions called challenging environments [1-2].

In IoT based sensing applications, many challenges are faced due to harsh environments, for example the need for reliable delivery of packets containing important data or control information. Similarly, excessive data traffic can cause congestion when

Received (June 15, 2018), Review Result (September 10, 2018), Accepted (October 4, 2018)

ISSN: 2005-4262 IJGDC

the available resources fall short of current injected data into the network. It results in buffers overflow of IoT devices, due to which packets have to be dropped and queuing delays increase. Such issue can be a setback for energy constrained IoT devices which are mostly battery-operated. Packet losses result in retransmissions for reliability ensuring and are immense cause of energy wastage. In these circumstances, the nodes close to the gateway node, in particular, are under severe pressure. In commonly used WSN architectures this kind of pressure are rooted in contention near a single sink due to multi-path data forwarding to it [3]. This phenomenon is also commonly referred as energy-wave problem as shown in Figure 1. Node in red color are under extreme pressure and its residual energy can quickly deplete due excessive forwarding on behalf of other nodes. These nodes are critical and solutions are needed to prolong its lifetime by avoiding unnecessary forwarding as otherwise the network will soon be disconnected.

In order to extend the lifetime of IoT network, congestion must be actively avoided and efficiently controlled because it has direct impact on energy efficiency as well as conventional QoS parameters like throughput, packet delay, packet loss ratio and fairness.

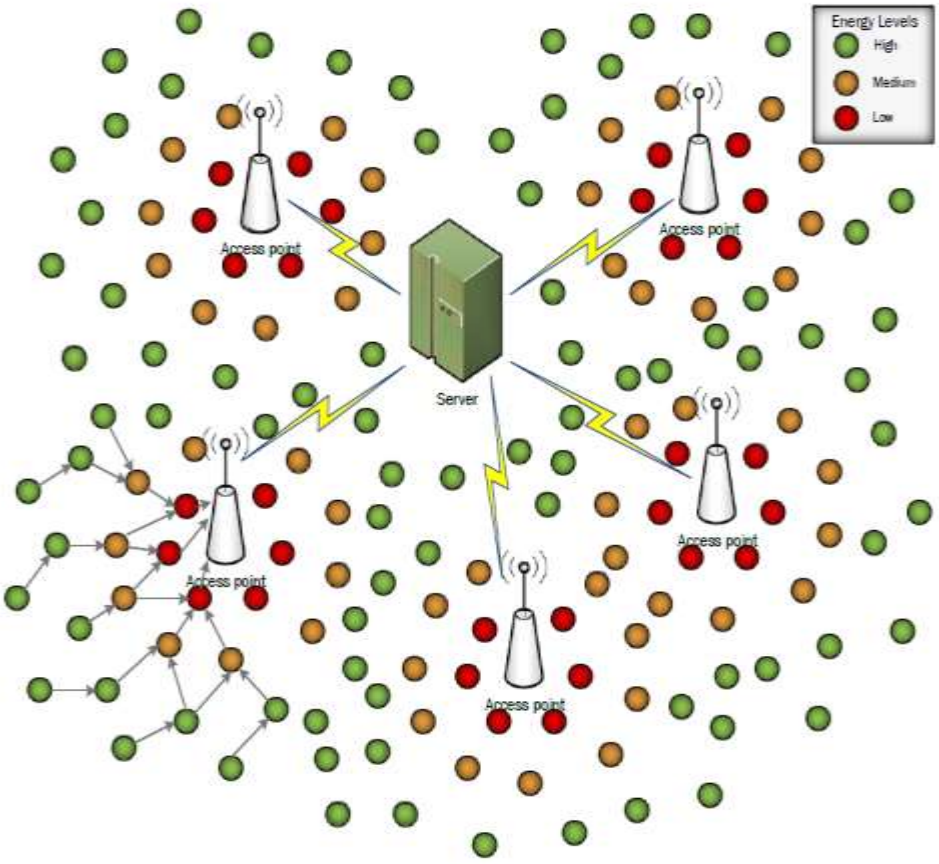


Figure 1. Illustration of Energy-Wave Problem in Sensor Network Environment

Therefore, IoT deployments in challenging environments pose congestion and data loss issues in high data rate communication applications and thus need transport layer services.

Data in IoT sensing applications flowing upstream from nodes, sending the sensed data to sink or gateway node is in many-to-one form. Congestion mostly occurs in sensing applications which generate data continuously and at high rates. There are two main approaches used to control congestion in sensor networks; end-to-end and hop-by-hop. Traditional TCP like methods which are end-to-end, are mainly based on Round-Trip

Times (RTT). RTT based methods can cause unfairness among IoT nodes which are at different distances from the sink called RTT Bias [4]. These methods are simple to implement, but are comparatively slow in response to the congestion occurred on intermediate nodes in dense networks. On the other hand, hop-by-hop congestion mechanisms are usually of complex nature, due to need of implementations at intermediate/forwarding IoT nodes but can give faster response to the point of congestion occurrence. Although link layer protocols provide link to link reliability but are unable to recover packet losses caused due to overflow of network queues. Therefore, a transport layer mechanism is required to re-transmit such dropped packets.

Transport layer protocols are meant to provide end-to-end delivery of data, with services of reliable delivery (in form of packet retransmissions if lost) and controlling congestion if it occurs due to excessive data traffic in the network. Due to implementation for specific diverse nature of sensing applications, traditionally most of the well-known transport layer protocols in sensor network domain are providing techniques for either, ensuring reliable delivery of data packets or solving issues of congestion control but very few are providing both the services [5-8]. This trend has got its own benefits like flexibility to choose different techniques for different services suitable for an application. On the other hand it creates problems like integration complexity of multiple techniques in order to achieve both the functionalities. So there is need for alternate single solution protocols of generic nature applicability in IoT sensing applications, which provide services of reliability, congestion control and traditional features of QoS metrics. Such a protocol enable users to eliminate integration complexity of separate solutions for services, eventually resulting in simplifying the design for resource constrained sensor network platforms and relieving the efforts for using it in bigger IoT (Internet of Things) paradigm.

In this work we have proposed combination of transport layer modules for high data rate IoT sensing applications; providing mechanisms both for end-to-end reliability and congestion control. Proposed work uses handshaking and connection establishment which are for continuous and dynamic high data rate IoT applications performing data collection. It uses data rate-based control and packet-window to perform data traffic injection close to available network resources. Most of the proposed functionalities have been implemented at the resourceful sink node, thus relieving the resource constrained IoT sensing motes and resulting in energy efficient working. Mechanism of congestion avoidance has been employed based on actively observing the trends in trip time delays of receiving packets at sink. Explicit congestion notification is sent and exact rate adjustment of senders is done according to the delivery rate value, received in the special congestion packet from sink.

Exact-rate adjustment has been taken up by some cross-layer protocols and considered to be very useful in traditional sensor network paradigm but their way of computing these rates are combination of complex mechanisms. The proposed congestion module adjusts exact sending rates through a simple transport layer mechanism by determining receive-rate at sink and providing it to data senders for rate control. For reliable delivery of data, the reliability module uses packet sequence numbering and efficient control overhead mechanism SNACK (Selective Negative Acknowledgments) for lost packets retransmissions. IoT sender mote uses packet-window based aggressive start phase to rapidly obtain effective sending rate and on receiving any control packet from sink, it switches to rate-based controlled phase, with window status staying active along for better matching of traffic and available resources.

The contributions of this work is twofold. First, we proposed a transport layer protocol by combination of novel delay based working modules, which achieves exact rate adjustment and fair sharing of bandwidth in high data rate IoT sensing applications. Second, we implemented the proposed modules in network simulator (ns-2) and showed

that these modules achieve significant performance improvement in comparison to existing protocols.

The rest of the paper is organized as follows: In Section 2, we present the related work Section 3 contains the detailed working and design of the proposed modules. Comparisons and results in simulations are discussed in Section 4 and Section 5 concludes the proposed work.

2. Related Work

Most of the existing well known transport layer protocols for sensor networks applications are providing techniques for either, ensuring reliable delivery of data packets or congestion control and very few are providing both the services.

CODA [9] is a hop-by-hop congestion control protocol which works in open loop back-pressure way. It detects congestion based on thresholds/levels of current buffer occupancy and channel load. When congestion is detected on a sensing node, it sends rate decrease notification (by setting "regulation" bit) to its up-stream neighboring motes which in turn forward it to their neighbours. If sink receives a packet with rate "regulation" bit ON, it instructs sensor nodes to decrease rates via "regulation" bit in ACK message. CODA does not have reliability mechanism, hence reliability of received data is an issue, especially under high data rate and sparse sources scenarios. The congestion module in proposed work in contrast has end-to-end congestion control mechanism based on delay trend detection and exact rate adjustment. It also has a module with functionality for reliable delivery of data packets.

RMST [10] protocol has been proposed to provide reliability of data delivery from sensing nodes to sink. It uses Selective NACKs for lost packets recovery which results in small control traffic and energy usage. RMST can be configured for in-network caching and the detection of lost packets is done through timer-driven approach. Retransmissions requests are originated from the packet loss detecting node and are forwarded on the path towards the source node. Packets get re-transmitted from the node in the path wherever found cached in-network. RMST is used with the routing paths discovered by Directed Diffusion protocol [11]. RMST has no mechanism for congestion control whereas the proposed work in this paper is providing it in congestion module. Hop-by-hop mechanism of RMST is different from end-to-end approach of the proposed work but both uses less control overhead and energy conserving SNACK mechanism.

Y. G. Iyer *et al.*, proposed sink based STCP [12], an end-to-end protocol providing both the functionalities. When connection is established, sender informs about variable reliability required, transmission rate, type and number of flows. For reliability service, in event-driven flows it uses ACKs mechanism whereas NACKs are used for continuous flows. Congestion detection is based on node's queue filling length and a single bit is set ON in data packet's header for congestion notification. Upon receiving such packets, sink will set the bit ON in ACKs to notify congestion to the senders. Its uses TCP AIMD rate control mechanism which is not efficient for sensor network environment. Also large number of event driven flows can be another source of energy wastage due to use of ACK for each received packet.

TCP is generally considered inefficient in case of wireless networks such as sensor networks used in IoT, as its congestion detection is based on packet losses. It cannot distinguish between the losses due to congestion and those dropped due to bit errors or link failures, which are fairly common in these environments. There are proposals which modify and enhance TCP, used to avoid its problems in wireless ad-hoc environments [13].

ELFN [14] was proposed to remove the problem to distinguish packet losses due to path break and network congestion in wireless ad-hoc environment. In case of link/router failure, the sender switches to standby mode during which retransmission timers are

disabled. To find out if route has been re-established, periodic probe packets are sent. When acknowledgment of any probe packet is received, the sender assumes repair of link and switches to normal mode.

New Reno [15] enhances retransmission of lost packets. During fast recovery phase for keeping the transmit window full, New Reno sends a new packet from the end of congestion window for every duplicate ACK that is received to it. Under conditions of network congestion, the congestion window is halved after a timeout or three duplicate acknowledgments. Westwood [16], which has modified the sender side of Reno, sets slow start threshold limit and congestion window according to the estimated channel bandwidth. Westwood performs monitoring of the delays of data packets and ACKs stream to estimate the available channel bandwidth which is then used for refinement of window control and back-off process. It improves efficiency under intermittent or random packet losses. The congestion detection mechanism of our proposed work has similarity to Westwood in terms of using the observation of delays but Westwood uses packet losses for congestion detection.

3. Proposed Work

In this paper, we have proposed a protocol with combination of transport layer modules for high data rate IoT sensing applications, which provide reliable end to end delivery and also perform congestion control efficiently.

3.1. Proposed Architecture

Figure 2 shows the typical view of node's internal structure at transport layer. IoT sensing nodes are constrained in resources, therefore most of the functionalities in the proposed modules have been implemented at the IoT network local sink node, which is usually rich in energy, memory and processing capabilities. A lightweight version of the same protocol is developed for resource constrained IoT devices.

In the proposed Reliability and Delay based Congestion Avoidance protocol, two-way handshaking is initiated when IoT sensing application creates and forwards the data packets to its buffer. Connection Request packet is sent from sender for connection initiation and on receiving this packet, the IoT sink replies with Connection Reply packet. Sink also computes the trip time (TT) by the difference of send-time contained in packet header and current time, then multiplying it by 2 to estimate initial RTT (Round Trip Time). It inserts the value of RTT in Connection Reply packet. Upon delivery of Connection Reply, sender node starts its first phase of data sending and packet-window timer having an initial fixed-interval. On periodic expiration of timer, packet scheduling on sender is done if packet-window control allows sending. It means if the difference of node's current sequence number SN_{curr} and last acknowledged sequence number SN_{last} is less than packet-window value W_{pkt}. Following inequality about packets show this relation:

$$\left(SN_{curr} - SN_{last}\right) < W_{pkt}$$

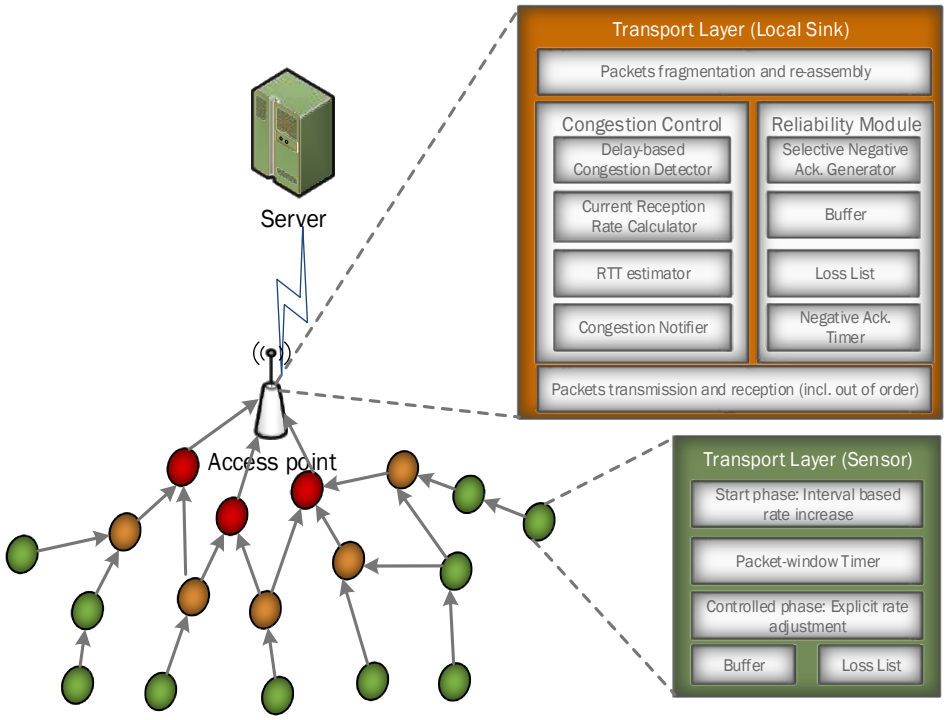


Figure 2. Node's Internal View at Transport Layer with Proposed Protocol Modules for Local Sink/Gateway and Sensor Nodes

When data has to be sent by the sensing application, payload is taken from application buffer and a new packet is generated and sent. The unacknowledged packets are kept in Sender Queue for possible retransmission request due to losses. When SNACK is received having sequence numbers of loss packets, proposed module enters these sequence numbers into its Loss List. When packet-window control allows sending, the sender first consults its Loss List and schedules the retransmission packets kept in its queue. New packets are only scheduled from application buffer when Loss List is empty. This way prosed module gives preference in scheduling to the packets requested in SNACK.

On receiving of each data packet at IoT sink, its trip time is calculated and stored in an array used to determine delay trend in packets receiving. If increase trend is observed continuously, then it is time to notify the sender in order to avoid congestion and packet losses. If sink receives a packet which has sequence number higher than the expected packet, it assumes possible packet losses. Now the sequence numbers of packets in between this and previous received packet are inserted to the Sink Loss List. A NACK timer is used and upon its expiration, retransmission of packets listed in Sink Loss List are requested by sending SNACK.

3.2. Types and Format of Packets

The proposed work uses two main types of packets shown in figure 3 and discussed below. Sizes of packets are varying but all have headers of size 32 bits divided into bit slots. Bit 0 has been kept for the flag to indicate the packet type.

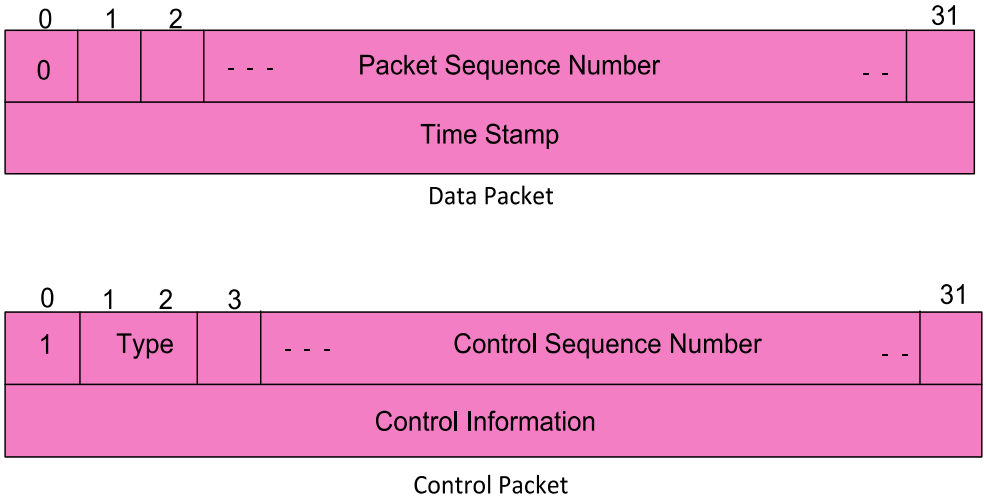


Figure 3. Format of Data and Control Packets

3.2.1. Data Packets: This type of packet is used to send data and has maximum size of 300 bytes, out of which 32 bits are used for header. Header contains flag (with value set to 0), Packet Sequence Number and Time Stamp. If data size from application layer is larger than 296 bytes, fragmentation has to be done. The proposed work supports handling the fragmentation by using sequence numbers in the packets.

3.2.2. Control Packets: The flag of control packet is set to 1 and its type value is combination of bit no 1 and 2 of header. Type value of control packets ranges from 1 to 4. The use of control packets is done for explicit congestion notification, retransmission requests (SNACK) and two-way handshake. Sizes of control packets vary depending upon its contained control information. Following four types of control packets are used. This type of packet is used to send data and has maximum size of 300 bytes.

- **Connection Request:** Initiation of connection establishment or two-way handshake is done by sender IoT node. It sends 24 bytes Connection Request packet to the sink node and then waits for the reply. It also has a timer attached and re-sends request if timeout occurs.
- **Connection Reply:** When sink node receives Connection Request from any sender node, it replies with this Connection Reply packet which is also 24 bytes.
- **Selective Negative Acknowledgment:** To ensure reliable delivery of data packets, retransmission request for lost packets is sent by this type of control packets. It carries a list containing IDs of lost packets. It also contains current average RTT value and receive-rate at sink node. SNACK packet size is varying depending upon the total number of packets sequence numbers in its list.
- **Congestion Control Packet:** This 32 bytes sized explicit congestion control packet is sent by sink for congestion avoidance. It too contains current receive-rate and values of average RTT at sink for rate adjustment of sender IoT node.

3.3. Communication Phases

The proposed architecture operates in the following two data sending phases.

3.3.1. Start Phase: For any sender IoT node, start phase is the initial mode of rapid increase in sending rate, to quickly reach the available capacity of the network paths. This phase is only driven by packet-window variable, which is number of in-transit packets at

the moment. Sender puts as many packets into the network as permissible by this window. Initial value for window is set to 2 packets and an increase of 1 packet is done after fixed intervals each equal to 0.4 second. Fixed rate increase intervals in the proposed module are learned from underlying topology and delays by senders reporting from event areas. In our used scenarios, average large trip times were 0.2 second. We did trials by using 0.2 and its multiples as fixed intervals and got better performance when using delay time 0.4 second as rate increase interval. That is why upper round trip times average (0.4 in our case) is chosen as increase interval in the start phase. The first phase is ended by sending node when Congestion Notification SNACK packet is received, but node does not adjust values of sending rate and packet-window, assuming packet losses may not be due to congestion.

3.3.2. Controlled Phase: Sending node switches to this mode at the end of Start Phase by receiving first Congestion Packet or SNACK. Congestion packet also carries sink's receive-rate (RR) and average RTT values. Node's sending rate is now adjusted equal to this receive-rate and adjustment of packet-window is done equal to RR * RTT to fill the perceived available bandwidth at the moment. In Controlled Phase, small increase in sending rate is done equal to 1 byte after small fixed intervals each of 0.01 second. Similar to start phase, small rate increase interval of controlled phase was chosen as small factor of upper delays average (0.01) gave better results in experiment trials. SNACK packets received during on-going controlled phase also cause adjustments of packet-window and sending rate values in similar way. If new packet losses are reported (via SNACK) during an interval, then increase in sending rate at the end of that interval is not done to avoid the probable congestion occurrence.

3.4. Reliability Module

The main cause of packet losses in the absence of channel errors is finite queues. In experiments we have employed DropTail Priority queue for nodes and its default size 50 packets. When congestion occurs, queues at some nodes in the network paths overflow and have to drop the packets forwarded to them. These are called congestion losses and transport layer in sensor networks is meant to recover these packets.

In the proposed module, reliability in form of retransmissions for lost packets is provided through SNACK. It is a less overhead mechanism as compared to Acknowledgment (ACK), Selective Acknowledgment (SACK) and Negative Acknowledgment (NACK). IoT sensor motes are constrained in resources especially energy, therefore control packets overhead should be kept very low. ACK or acknowledgment is used to tell the sender about the receipt of each data packet individually as is done in TCP. List-based acknowledgment SACK efficiently aggregates multiple acknowledgments to one packet, for the purpose of less control overhead. The use of NACK to request single lost packet, is used for efficient loss recovery especially in multi-hop wireless network environments [17]. Therefore, use of SNACK, aggregating sequence numbers of lost packets in a list, supports further reduction in total count of the control packets. Hence it results in less energy cost and is beneficial if used for reliability in IoT sensing application environment.

If a packet of higher sequence number than the next expected one is received at IoT sink, it enlists the in-between packets in its loss list. To enable probable out-of-order receiving, retransmission requests for these packets is not sent immediately but after a time duration equal to half of current average RTT value. If one or more packets in loss list have not actually lost, then this delay is reasonable enough for their receiving and placement at their respective positions in data for application. Sequence numbers of the packets received at sink before expiration of this delay timer are removed from Sink Loss List, while remaining packets are requested to be re-transmitted in SNACK.

Any sent packet is considered acknowledged (successfully delivered) if duration equivalent to average RTT or higher has been passed and no request has been received for its retransmission. But to be on the safe side due to possibility of its loss and resultant retransmission, packet is kept in sender's buffer upto 1.5 RTT interval, before its removal for freeing space for new data packets. On receiving SNACK packet from sink, IoT sensing node inserts sequence numbers of requested packets to a list maintained for retransmissions named as Sender Loss List. Scheduling of the retransmission packets takes priority in comparison to new data packets. When re-transmitted packet reaches at sink node, it is inserted at its proper position in received data and its sequence number is removed from Loss List.

Principally in wireless networks, the link layer is responsible to deal with bits corruption occurred due to channel errors on each link of the network path. If packet is not correctable then reliability mechanism correctly delivers it through retransmissions mechanism. In case of channel errors if SNACK is not received properly by the data sender, the sink will not receive the requested lost packets identified in SNACK packet. The sink will then retransmit the SNACK packet to the sender. When SNACK packet is received at sink after passing through the above mentioned mechanisms, it is considered reliable for loss recovery.

3.5. Congestion Control Module

The proposed module employs delay-based detection and end-to-end exact rate adjustment congestion control. These are described below.

3.5.1. Delay Based Congestion Detection: Delay based mechanisms have already been used in efficient high speed networks protocols like Hybrid congestion control, FAST TCP and Ilinois [18]. In the proposed protocol the delay trends are determined by a window of packet delays and takes into account variance of packet delays. Thus random packet loss or delay increase has virtually no effect. In such a protocol the congestion control mechanism starts playing its role very early at the time when packet delays are incurred and starts adjusting packet rates thus avoiding filling up of queues and packet drops.

An alternate to delay-based is loss-based congestion control. This is not suitable in IoT sensing applications because this might result in more energy consumption, described as follows. When congestion occurs the packet queues starts to fill up and eventually starts dropping the packets. At this stage the congestion is detected and congestion control mechanism triggers in. Thus packet loss becomes necessary before the congestion is detected resulting in energy wastage. Moreover, in case of random packet loss due to channel conditions, the loss-based mechanism will misinterpret it as a congestion state resulting in even more slowing of data rates.

Please note that the proposed module uses separate delay windows at sink for each sender and delay trend is determined for each sending IoT node based on its own distance and conditions. There is a possibility that packets may arrive at the sink from different paths resulting in different delays. Since the protocol use variance of delays of multiple packets, in such a case the effect would be determined by the delays of majority of packets received at sink.

Algorithm 1 describes the congestion detection of the proposed protocol. On receiving of each data packet at IoT sink, its trip time is calculated and stored in an array (line 3) used to determine if delay trend increases or not. Packet trip time is also used to update average RTT value by using its double in weighted average formula giving it 0.3 and 0.7 weight to existing average RTT. Delay trend is continuously observed and is determined on each packet receiving to sink. Delay trend is based on delay variance (line 5) in Trip-Times of 16 previous data packets received. Increased trend is considered to be the result of IoT sensing data traffic being heavier for available resources in network paths causing

more buffer delays. Hence it is taken as a signal to expected congestion occurrence and packet losses in near future. Therefore, to avoid congestion, Explicit-Congestion-Notification packet is sent to the sender pro-actively, putting current RTT average and receive-rate values in it. This congestion avoidance mechanism based on delay-trend (line 6) instead of packet losses, makes the proposed protocol suitable for high data rate IoT sensing applications environment. It is a proactive approach which tries to avoid congestion before it occurs which results in fewer losses of packets in comparison loss-based, reactive congestion detection mechanisms resulting more losses.

Another array called Time Array is maintained at IoT sink to record in-between intervals of arriving of data packets. When congestion packet or SNACK is to be sent towards the sending IoT node, the previous sixteen intervals stored in this array are sorted and random huge or small values are excluded by applying median filter. Values greater than (2.5 median) and less than (median/2.5) are dropped and rest of the intervals (at least 8 or more in number) are taken to get the current average receive interval. Its reciprocal is sent as the current receive-rate (line 9) in the packet header of control packet (line 11) towards the sender.

Algorithm 1: Congestion Detection algorithm on IoT Sink node s

1 **Upon reception of** $data$ **packet** i
2 $t_{trip} \leftarrow t_{clock} - t_i$
3 $trip\_times[\texttt{++x} \mod 16] \leftarrow t_{trip}$
4 $RTT \leftarrow RTT \times 0.7 + 2 \times t_{trip} \times 0.3$
5 **calculate** $Variance\ (trip\_times[..])$
6 bool $delay\_trend(Variance)$
7 $t_{last} \leftarrow t_{clock} - t_{ctrl}$
8 **if** $delay\_trend = increase$ ***and*** $t_{last} > 4 \times t_{trip}$ **then**
9 $\quad cong.recv\_rate \leftarrow \frac{1}{recv\_interval}$
10 $\quad cong.rtt \leftarrow RTT$
11 $\quad$ **send** $cong$ packet
12 $\quad t_{ctrl} \leftarrow t_{clock}$
13 **else**
14 $\quad$ **terminate** algorithm

3.5.2. Explicit Congestion Notification: The proposed module uses special control packet for notification of congestion explicitly. Values of current RTT and receive-rate at sink are inserted into this Congestion Packet, which are to be used by the sending IoT node to adjust its rates and packet-window variable accordingly. For optimization purposes, next congestion packet is sent only if time equal to 2 RTT has passed since the previous such packet was sent towards a sender.

3.5.3. Exact Rate Adjustment and Additive Increase: Proposed delay based mechanism supports fairness in comparison to the RTT-bias inherent in TCP AIMD algorithm [4]. The algorithm does small rate increase for all the nodes reporting events after fixed intervals thus supporting approximate fair sharing of the bandwidth, irrespective of IoT nodes' location. On the other hand TCP RTT-bias is relative to distance of nodes to the sink. Sender's location here means its relative distance to the IoT sink. For example, the direct neighbors to sink with small RTT are the nearest nodes as compared to the nodes at 10 hops distance resulting in longer RTT. The TCP algorithm increases sending rate of a node after a duration equal to RTT of its packets. Therefore, nearby nodes get frequent

increases due to their small RTT as compared to nodes at a far distance, resulting in unfair bandwidth distribution.

Algorithm 2: Rate Adjustment and Increase algorithm on Sender IoT node n

1 **Upon reception of** $congestion$ **packet** $cong$
2 **terminate** $start\_phase$
3 $send\_rate \leftarrow cong.recv\_rate$
4 $W_{pkt} \leftarrow cong.rtt \times cong.recv\_rate$
5 **set** $t_{inc}(0.01)$
6 **upon** t_{inc} **expire{**
7 $send\_rate(bytes) \leftarrow send\_rate + 1$
8 $t_{send} \leftarrow \frac{1}{send\_rate}$
9 **set** $t_{inc}(0.01)$
10 **}**

Algorithm 2 describes the rate adjustment and increase on a sender IoT node. When a sender node receives a Congestion Packet (line 1), it tunes its sending interval in order to adjust its sending rate according to receive-rate of sink (line 3). Sender also makes packet-window equal to RR * RTT (line 4). After exact rate adjustment, the main driving factor Sending Rate (line 7) is additively increased resulting in decreased Sending Interval (line 8). This increase is done periodically by a very small value 1 byte after passing of small fixed intervals 0.01 second.

4. Experiments and Results

In this section we evaluate and compare the performance of the proposed architecture with different enhanced variants using ns-2 [19]. We have used New Reno, Westwood and ELFN variants which are enhanced and suitable for IoT sensor networks like wireless ad-hoc network environments.

In experiments, IoT sensor network comprising of 100 nodes, spread on an area 450*200 m^2 has been used. We used uniform random distribution topologies for IoT nodes in simulations. Using random topology generator, we generated 5 different random deployments with uniform distribution. For each topology we did 10 runs of an experiment set, resulting the values averaged on 50 repetitions in total. Congestion usually occurs in event based, high data rate IoT sensing applications when some events occur in the network. We have simulated such scenarios when IoT nodes near to any event start sending large observation data to effectively report it to IoT gateway/ sink whereas other nodes of the network participate as forwarding hops. There could be another IoT sensing application scenario in which all the nodes continuously send small data packets at a stable rate after long send intervals, usually such kind of scenarios do not cause congestion. We experimented with traffic increasing using certain number of senders which are in proximity with the event. Such a scenario would result in traffic flows using shared paths and hence congestion occurs. The scenarios used in our experiments contain an event occurring few seconds after simulation start and nearby sensor nodes start sending sensed data to the sink. The said event is continuously monitored upto the end of simulations. All the experiments have been run for 100 seconds duration. If sending data rate of a single node increases or number of sending nodes increases, then chances of congestion occurrence in network paths rises.

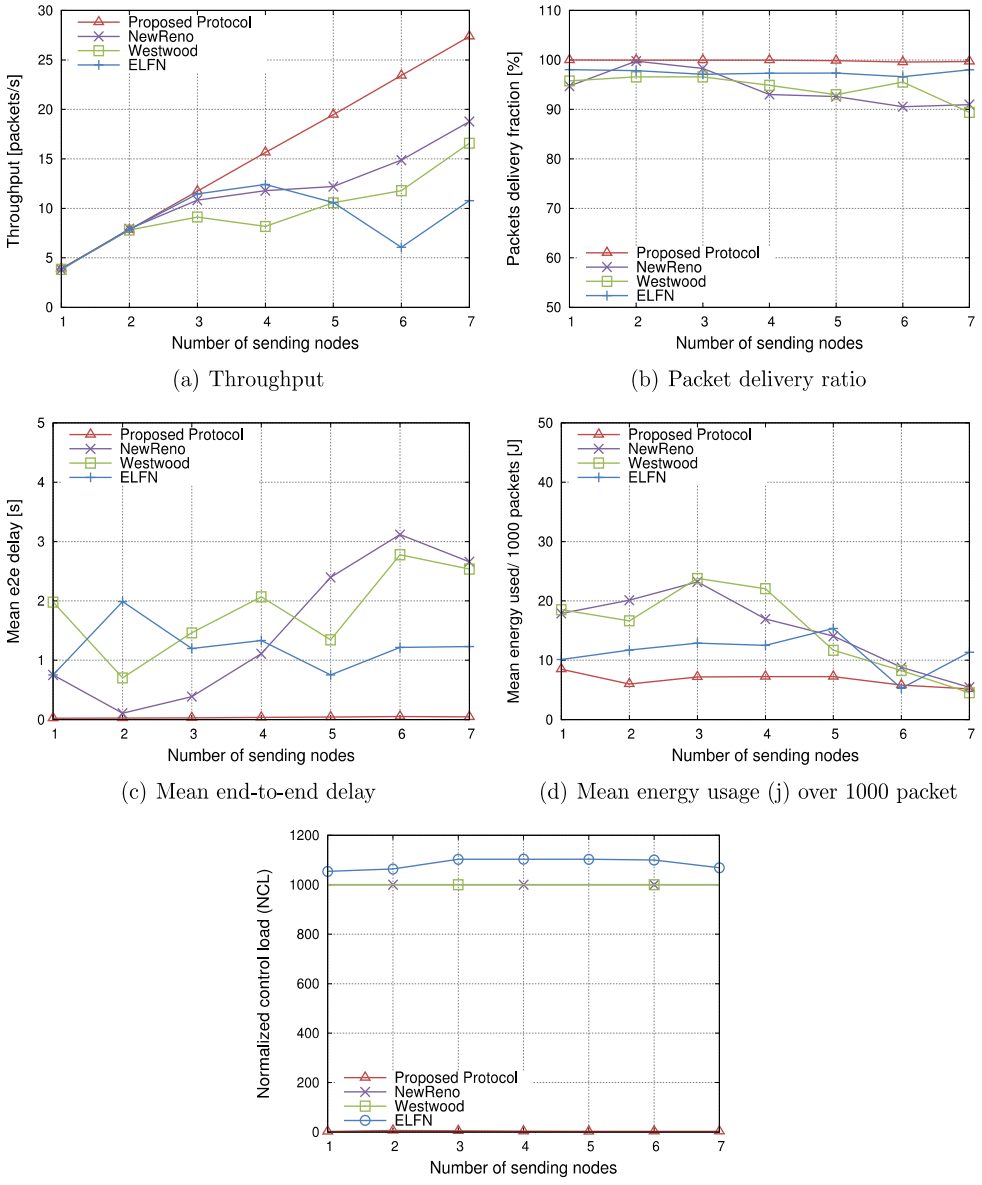
We have used AODV protocol on the routing layer and typical IEEE 802.11 MAC protocol which uses CSMA/CA channel access mechanism. Its underlying access technique uses DCF (Distributed Coordination Function) similar to its high speed extensions (a/b/g). Nodes use contention based carrier sense multiple access with collision avoidance (CSMA/CA) and binary exponential backoff algorithm. IEEE 802.15.4 standard was designed for low power, low rate wireless networks but it poses challenges for large scale IoT sensing applications requiring high reliability and scalability [20]. Despite high power consumption, 802.11 is considered suitable due to its ubiquitous coverage and providing many different rates. Therefore, IEEE is working on the low power Wi-Fi standard 802.11ah for embedded connectivity which communicates in the sub Gigahertz ISM bands giving about a kilometer range. Hence we decided to use 802.11 MAC for our experiments.

Table 1. Simulation Parameters used for Results Collection

Parameter	Value/Range
Nodes count	100
Packet Size	300 Bytes
Node Queue Type	Drop Tail Priority Queue
Node Queue Size	50 packets
Node Initial Energy Level	1000 Joules
Data sending rate (KBps per node)	4, 8, 12, 16, 20, 24, 28, 32, 36, 40 KB/sec
Number of sending nodes	1, 2, 3, 4, 5, 6, 7
Application Layer	Constant Bit Rate (CBR) Application protocol
Transport Layer	Proposed Protocol, NewReno, Westwood, ELFN
Routing Layer	AODV Protocol
MAC Layer	IEEE802.11 MAC (CSMA/CA)
Radio Propagation Model	Two Ray Ground
Communication Range	100m
Area Size	450m x 200m
Mobility Type	No Mobility
Simulation Time	100 sec

The results of experiments have been obtained using well defined approximation models of true shared wireless channel conditions. We have employed Wireless Physical network interface layer to access the shared channel, which is subject to collisions, attenuation and the radio propagation model. We have used Two Ray Ground radio propagation model which considers the direct path plus a ground reflection path and predicts the received power as a deterministic function of the distance. The Wireless Channel module simulates the shared medium and supports the medium access mechanisms of the MAC to carry out carrier sense, contention, backoff algorithm, transmission and collision avoidance. We employed ns-2 default Energy Model and accordingly used the default value of 281.8 mW for transmit power (txPower) and receive power (rxPower). For our experiments, we set initial energy of each IoT node equal to 1000 Joules. In this model, node consumes energy for every packet it transmits according to (txPower * txTime) and for packet it receives equal to (rxPower * rxTime). Energy Model logs remaining energy of node in trace file for every event of the packet send/receive. We used these values to calculate average consumed energies. Table 1 presents a summary of configuration and various parameters used in these simulation results.

4.1. Results with Increasing Number of Sender Nodes



(a) Throughput

(b) Packet delivery ratio

(c) Mean end-to-end delay

(d) Mean energy usage (j) over 1000 packet

(e) Normalised control packets load

Figure 4. Results with Increasing Number of Sending Nodes

In this subsection, performance of the proposed protocol has been done to other IoT variants on increasing number of sending IoT nodes (observing an event) ranging from 1 to 7. Figure 4(a) is showing the throughput comparison of the proposed protocol against others. We can see that proposed architecture shows high scalability and gives about linearly increasing throughput with increasing number of sending nodes, thus outperforming competing variants. Competing variants show unstable, varying performance and are unable to deliver the increased amount of generated IoT sensing data.

In Figure 4(b), the comparison of packet delivery ratio (PDR) in percentage has been shown. The PDR is calculated as the number of packets received (without loss) over the total number of packets sent. We can see that the proposed architecture is able to deliver

the increasing number of data packets with very less number of packet drops, thus it saves energy. Proposed protocol shows stable performance, delivering more than 99.85 % of generated data packets in all 7 cases, with number of dropped packets only 0.15 on average overall, whereas competing versions are dropping more packets resulting on average 95 % PDR.

We can see the comparisons of the proposed work and other versions for mean end-to-end delays in Figure 4(c). In these experiments based on increasing number of sending nodes, our protocol maintains less than 0.1 second delay, whereas competing versions remains above 1 second delays and goes up-to as large as 3 seconds. The other enhanced versions suffer due to the imposition that congestion losses do that congestion losses do occur first and congestion control is done after detection based on these losses. Whereas our architecture is able to achieve low delays due to its proactive congestion avoidance mechanism and it effectively adjusts the rates when delay trend increases before congestion losses. It uses delay as well as packet-window for controlling sending data rate. This results in sensing traffic injection equal to the available IoT network resources enabling it to achieve sustained low delays.

Table 2. Statistical Summary of Throughput (packets/sec) Results with Varying Number of Sending Nodes

Statistics	Protocol	Number of Sending Nodes			
		1	3	5	7
Mean	Proposed Protocol	3.91	11.74	19.47	27.37
	NewReno	3.82	10.83	12.20	18.78
	Westwood	3.83	9.13	10.56	16.58
	ELFN	3.86	11.46	10.58	10.78
Std. dev	Proposed Protocol	0.05	0.14	0.26	0.35
	NewReno	0.09	0.27	0.39	0.48
	Westwood	0.11	0.31	0.44	0.52
	ELFN	0.15	0.33	0.48	0.56
Min.	Proposed Protocol	3.71	10.06	14.27	17.57
	NewReno	3.46	7.86	7.52	9.66
	Westwood	3.39	6.34	5.72	7.74
	ELFN	3.26	7.50	5.30	4.62
Max.	Proposed Protocol	4.11	13.42	24.67	37.17
	NewReno	4.18	13.80	16.88	27.90
	Westwood	4.27	11.92	15.40	25.42
	ELFN	4.46	15.42	15.86	16.94

Figure 4(d) shows the energy usage values of the proposed work against competing versions, averaged for 1000 packets. It is clearly evident in this result that the proposed protocol is stable and does efficient usage of energy resources. The competing versions are using more energy for delivering the same number of packets, because of their use of acknowledgments for each data packet. The proposed reliability module with its SNACK mechanism uses very less control overhead, making it energy efficient even with continuously increasing data generation by more senders for an event.

In Figure 4(e), the comparison about Normalized Control Load (NCL) of the proposed architecture has been shown against the other versions. We can see that control packets used by proposed protocol are much less in number. For delivering huge amount of data, it still shows stable energy-efficient nature regarding control packets, whereas competing versions show NCL values 1 and above, which are due to acknowledgments sent in response of each received packet.

Table 3. Statistical Summary of end-to-end Delay (sec) Results with varying Number of Sending Nodes

Statistics	Protocol	Number of Sending Nodes			
		1	3	5	7
Mean	Proposed Protocol	0.03	0.03	0.04	0.05
	NewReno	0.75	0.39	2.40	2.66
	Westwood	1.98	1.46	1.34	2.54
	ELFN	0.75	1.20	0.75	1.23
Std. dev	Proposed Protocol	0.02	0.02	0.02	0.03
	NewReno	0.13	0.18	0.35	0.49
	Westwood	0.19	0.28	0.39	0.46
	ELFN	0.12	0.23	0.30	0.38
Min.	Proposed Protocol	0.01	0.01	0.02	0.02
	NewReno	0.62	0.21	1.35	1.19
	Westwood	1.60	0.90	0.95	1.16
	ELFN	0.63	0.97	0.45	0.85
Max.	Proposed Protocol	0.04	0.05	0.07	0.07
	NewReno	0.88	0.57	3.45	4.13
	Westwood	2.36	2.02	1.73	3.92
	ELFN	0.87	1.43	1.05	1.61

Table 2 present the statistical summary of throughput results with varying number of sending nodes. Likewise, statistical summary of end-to-end delay results with varying number of sending nodes is given in Table 3.

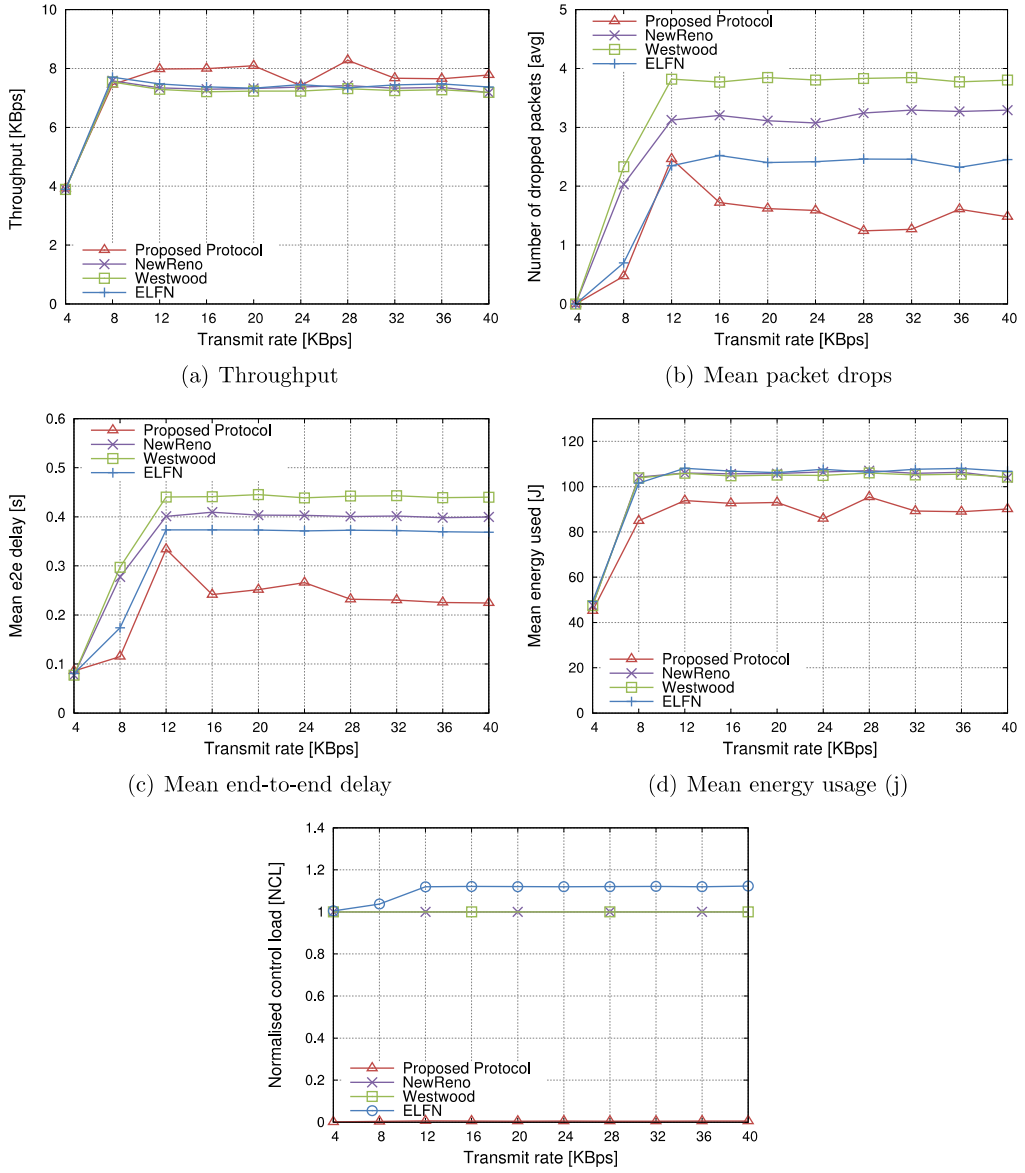
4.1. Results with Increasing Rate

In this subsection, performance of proposed work is compared against other sensor network versions, on different sending data rates of a single IoT node, ranging from 4 KBps to 40 KBps.

Figure 5(a) shows the throughput performance of the proposed architecture against other versions. The proposed work shows stable performance and higher throughput than competing versions for most of the different sending rates. Looking at this graph, we can say that network paths are able to support data rates less or above 8 KBps. Proposed protocol is delivering about this much data nearly in all the varying high data rate experiments, whereas other variants throughput performance is less than that of proposed architecture.

In Figure 5(b), we have shown the average number of packets dropped by the proposed work and competing protocols. In this result, we see that proposed protocol has dropped less number of packets on average (less than 2) than TCP versions (as large as 4) on all increasing values of transmitting data rates, thus saving much of nodes' energy.

Figure 5(c) is for showing the mean end-to-end delay performance of the proposed work against other versions. In these experiments using increasing sending rate, our architecture remains on 0.25 second delay, whereas competing versions have 0.4 second delays on average. We can observe that end-to-end delay values of competing protocols are higher than that of the proposed work, which means that it is able to deliver the high data rate sensing data efficiently in time constraints.



(a) Throughput

(b) Mean packet drops

(c) Mean end-to-end delay

(d) Mean energy usage (j)

(e) Normalised control packets load

Figure 5. Results with Increasing Data Rate

We can see the comparison of the proposed protocol and other versions on mean energy consumption during high data rate sending in Figure 5(d). Here the results show stable and efficient usage of energy by the proposed architecture on all the sending rates. The other sensor network variants use more energy for different sending rates in all experiments and has near equal consumption (above 100 joules) for all competing versions. We know that this increase in energy consumption is due to acknowledgments in response to all data packets.

In Figure 5(e), we have shown the Normalized Control Load (NCL) performance of the proposed protocol and other versions. The proposed architecture uses less than 1 packet on average for delivering 100 packets in these experiments involving increasing sending data rates by an IoT node. It is due to the SNACK mechanism usage for reliable data delivery/retransmission that the proposed module shows stable and efficient energy usage. Whereas TCP versions show NCL values 1 (means 1 control packet for each 1 data

packet) and above, which are due to acknowledgments sent in response of each received packet.

5. Conclusion

For high data rate IoT sensing applications in challenging wireless environments, we need to integrate sensor network protocols designed specifically for reliability or congestion control. So for high data rate IoT sensor networks, there is need of scalable and energy efficient generic transport mechanisms having both the functionalities. We proposed delay based IoT transport protocol which is providing both the services; reliability and congestion control in energy efficient manner. The proposed work uses SNACK mechanism for re-transmissions which enables it to keep its control overhead very small and thus resulting in efficiency. The protocol mechanisms are residing at resourceful IoT gateway/ sink node, its active congestion control module based on delay and exact rate adjustment is efficient, supporting fairness and scalability. The proposed module allows receiving of out-of-order packets, which usually happens in environments like IoT sensor networks. Its mechanism of rate increasing after fixed short intervals is the source for supporting fairness among IoT nodes at different distances to the sink and this does not result biasing due to RTT. It also controls on-fly packets number through its packet-window and results in less queue delays, overflows and IFQ losses.

We evaluated the proposed architecture through extensive ns-2 simulations against existing protocols. Itachieved3timesbetter end-to-end delay and less than 0.005 normalized packets load (NCL) in comparison to 1 NCL by other variants. Moreover, it effectively sends data packets generated by the sensing nodes with 99.85 % packet delivery ratio (PDR) whereas competing sensor network variants have 95 % PDR in comparison. Therefore, for IoT sensing applications with high data generation rates, our proposed end-to-end delay based generic transport architecture using exact rate adjustment and fair bandwidth sharing emerges to be more efficient, scalable and fair protocol.

References

[1] M. Ehrlich, L. Wisniewski and J. Jasperneite., “State of the art and future applications of industrial wireless sensor networks”, In Kommunikation und Bildverarbeitung in der Automation, Springer, **(2018)**, pp. 28-39.
[2] U. I. Minhas, I. H. Naqvi, S. Qaisar, K. Ali, S. Shahid and M. A. Aslam, “A WSN for monitoring and event reporting in underground mine environments”, IEEE Systems Journal, **(2017)**.
[3] B. Silva and G. P. Hancke, “IR-UWB-based non-line-of-sight identification in harsh environments: Principles and challenges”, IEEE Transactions on Industrial Informatics, vol. 12, **(2016)**, pp. 1188-1195.
[4] K. Tan, J. Song, Q. Zhang and M. Sridharan, “A compound TCP approach for high-speed and long distance networks”, Proceedings-IEEE INFOCOM, **(2006)**.
[5] X. Chen, Y. Xu and A. Liu, “Cross layer design for optimizing transmission reliability, energy efficiency, and lifetime in body sensor networks”, Sensors, vol. 17, **(2017)**, pp. 900.
[6] M. A. Mahmood, W. K. Seah and I. Welch, “Reliability in Wireless Sensor Networks: A Survey and Challenges Ahead”, Computer Networks, **(2015)**.
[7] A. Ghaffari, “Congestion control mechanisms in wireless sensor networks: A survey”, Journal of Network and Computer Applications, vol. 52, **(2015)**, pp. 101-115.
[8] M. Kafi, D. Djenouri, J. Ben-Othman and N. Badache, “Congestion Control Protocols in Wireless Sensor Networks: A Survey”, Communications Surveys Tutorials, IEEE, vol. 16, **(2014)**, pp. 1369-1390.
[9] J. Zhao, L. Wang, S. Li, X. Liu, Z. Yuan and Z. Gao, “A survey of congestion control mechanisms in wireless sensor networks”, Intelligent Information Hiding and Multimedia Signal Processing (IIH-MSP), (2010) Sixth International Conference on IEEE, **(2010)**, pp. 719-722.
[10] P. R. Pereira, A. Grilo, F. Rocha, M. S. Nunes, A. Casaca, C. Chaudet, P. Almström and M. Johansson, “End-to-end reliability in wireless sensor networks: Survey and research challenges”, EuroFGI Workshop on IP QoS and Traffic Control. Citeseer, vol. 54, **(2007)**, pp. 67-74.
[11] H. Karl and A. Willig, “Protocols and architectures for wireless sensor networks”, John Wiley & Sons, **(2007)**.

[12] Y. Iyer, S. Gandham and S, Venkatesan, "STCP: a generic transport layer protocol for wireless sensor networks", Computer Communications and Networks, ICCCN 2005. Proceedings. 14th International Conference, **(2005)**, pp. 449-454.
[13] I. F. Akyildiz, T. Melodia and K. R. Chowdhury, "A Survey on Wireless Multimedia Sensor Networks", Computer. Networks, vol. 51, **(2007)**, pp. 921-960.
[14] A. M. Al-Jubari, M. Othman, B. M. Ali and N. A. W. A. Hamid, "TCP performance in multi-hop wireless ad hoc networks: challenges and solution", EURASIP Journal on Wireless Communications and Networking, **(2011)**, pp. 198.
[15] Tcp New Reno, https://tools.ietf.org/html/rfc3782, **(2004)**.
[16] S. Henna, "A throughput analysis of TCP variants in mobile wireless networks", Next Generation Mobile Applications, Services and Technologies, NGMAST'09, Third International Conference on. IEEE, **(2009)**, pp. 279-284.
[17] N. Tiglao and A. Grilo, "On the optimization and comparative evaluation of a reliable and efficient caching-based WSN transport protocol", Design of Reliable Communication Networks (DRCN), 9th International Conference, **(2013)**, pp. 226-233.
[18] W. Xu, Z. Zhou, D. T. Pham, C. Ji, M. Yang and Q. Liu, "Hybrid congestion control for high-speed networks", Journal of Network and Computer Applications, vol. 34, **(2011)**, pp. 1416-1428.
[19] Network Simulator ns-2. http://www.isi.edu/nsnam/ns.
[20] G. Anastasi, M. Conti and M. Di Francesco, "A comprehensive analysis of the MAC unreliability problem in IEEE 802.15. 4 wireless sensor networks", IEEE Transactions on Industrial Informatics, vol. 7, **(2011)**, pp. 52-65.

