IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received May 6, 2021, accepted May 28, 2021, date of publication June 4, 2021, date of current version June 15, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3086460

IPLQueeN: Integrity Preserving Low-Overhead Query Handling Over NDN-Based WSN

TANUSREE CHATTERJEE[1], SOMNATH KARMAKAR[2], AND SIPRA DAS BIT[1], (Senior Member, IEEE)

[1]Department of Computer Science, Indian Institute of Engineering Science and Technology, Shibpur, Howrah 711103, India
[2]Research and Innovation Lab, Tata Consultancy Service, Kolkata 700160, India

Corresponding authors: Tanusree Chatterjee (tnsr.chatterjee@gmail.com) and Sipra Das Bit (sdasbit@yahoo.co.in)

ABSTRACT Recently there has been a new emerging trend in using the technology of Named Data Networking (NDN) on wireless sensor networks (WSNs) to improve data-centric communications over WSNs. The usage of the NDN notion in WSN for such communications can be advantageous in many respects ranging from ensuring secure data collection without using the nodes' location and several nodes to possible delivery of sensory data by utilizing the cached data in intermediate components of the network. On the other hand, WSN being a resource-constrained, energy-starved network, the data communication mechanism must be low-overhead in terms of computation and communication overheads. Thus, we propose integrity preserving low overhead query handling over NDN-based WSN. We make the scheme low overhead by judicious use of the NDN data structure Content Store (CS), Pending Interest Table (PIT), Forwarding Information Base (FIB) in a few strategically placed nodes. We also make the data collection secure by applying an existing Light-weight One-way Cryptographic Hash Algorithm (LOCHA) on the collected data. The performance of the scheme is analyzed theoretically in terms of communication, computation, storage overheads. We also simulate the entire query processing scheme including request and response forwarding in Cooja, the Contiki network simulator. Simulation results show that our scheme does not compromise with network performance such as packet loss rate, network lifetime, end-to-end delay, etc. while achieving the design goal of making the scheme energy saving which establishes that the scheme is readily implementable in real life mote e.g., Tmote Sky. Finally, all the results are compared with three competing schemes and the results confirm our scheme's supremacy in terms of both design performance as well as network performance.

INDEX TERMS Data integrity, hierarchical clustering, named data networking, query processing, wireless sensor network.

I. INTRODUCTION

Data-centric communication [1], [2] plays an important role in wireless sensor networks (WSNs) by enabling sensory content delivery without revealing the identity of the sensors. For example, in developing query-driven applications in WSN [3], attribute-based naming is necessary to specify the properties of data in a query. However, there is no standard naming scheme for various complex applications in the literature, which limits the deployment of the data-centric approaches in practice for WSNs [4]–[9]. In the meantime, named data networking (NDN) has emerged as a promising field to cope with the usage of today's internet which follows data-centric communication [10], [11]. This receiver-based model in the NDN fits naturally into data-centric WSNs [12].

Unlike IP networks, NDN uses hierarchical data names instead of IP addresses for data delivery. The hierarchical naming structure of NDN makes routing and forwarding easier for data retrieval. Here, communication is initiated from the receiver end, called a *consumer*. A node in the network which can serve the requested data is called a *producer*.

The data transfer takes place by exchanging requests (*interest*) and response packets (*data*) between consumers and producers. Each of the nodes maintains three data structures namely CS, PIT, and FIB. The nodes store some of the recently received data in their CS, which is called

The associate editor coordinating the review of this manuscript and approving it for publication was Amjad Ali.

in-network caching. Also, before forwarding any interest towards producers, each node checks for the interest in their PIT. The usage of these features with the help of such data structures reduces communication overheads thereby reducing energy consumption, which finds its applicability in the energy-constrained network like WSN. Also, NDN uses digital signatures on data packets to preserve data integrity thereby enhancing security [12], [13].

Due to the data-centric nature of WSN applications, a lot of research has been done on querying and tasking sensor nodes to answer both real-time and non-real-time queries [2], [9]. Also, the nodes in WSN are limited in computational capability. But a recent trend is observed in using the technology NDN on WSN to improve data-centric communications over WSNs [12], [14]. In most applications, the nodes are deployed in the open area thereby remaining unattended and prone to attack. Thus, the need of securing data during communication in such an environment is of utmost importance. This motivates us to develop a low-overhead secure query processing scheme over NDN-based WSN.

We consider a hierarchical clustered architecture of WSN over which query-based application is running. We propose an integrity **p**reserving and **low**-overhead **que**ry handling scheme applying NDN techniques on hierarchical cluster-based WS**N** (IPLQueeN). We make the scheme low overhead by judicious use of the NDN data structure in a few strategically placed nodes. We also make the data collection secure by applying an existing Light-weight One-way Cryptographic Hash Algorithm (LOCHA) [15] to the collected data.

The main contributions of our work are as follows:

- Developing a query processing scheme over the hierarchical clustered architecture of WSN.
- The scheme is made low overhead by judicious use of the NDN data structure in a few strategically placed nodes in the network.
- The data transfer is made secure by applying a low-overhead hash algorithm LOCHA.
- Evaluating the performance of our scheme using Cooja, the Contiki network simulator.

The rest of the paper is organized as follows. Section II discusses a few related works on WSN, NDN and, NDN-based WSN. The system model and some background studies are presented in Section III. Section IV explains the proposed IPLQueeN. Performance analysis of the scheme is done in Section V. We conclude the work in Section VI highlighting the future scope.

II. RELATED WORK

We firstly review a couple of works on low overhead query processing over WSN.

In work [2], a query processing technique over wireless sensor networks (WSN) is proposed which improves energy efficiency and storage space optimization in the network. The user's input queries in the server in simple SQL-like language which describes how the data to be collected and how users combine and summarize it. The declarative queries provide an easy-to-interface and energy-efficient execution approach. The simulation of the scheme shows that it works well in a controlled environment. Authors in the other works [3], [4] propose secure query processing that preserves basic security properties like authentication, data integrity, privacy. The works consider a query-driven application platform where the sink forwards query messages towards member nodes (MNs) via cluster heads (CHs) or aggregation nodes (ANs). Upon receiving the responses, each CH/AN aggregates the data and forwards it to the sink. While forwarding, the data from different MNs are aggregated and encrypted resulting in reduced data transmission and data protection, respectively. The performance results show the schemes are low overhead in terms of storage, communication, and computation. Also, the works in [5], [6] propose energy-efficient clustering schemes that reduce the transmission delay and energy consumption in the network. The schemes either adopt the conventional methods and/or algorithms or hybridize the concepts of conventional algorithms. The methods also compare the risk probability, data security with other state-of-the-art schemes. The comparative analysis shows an improvement of the proposals over other conventional artworks in terms of all the overheads.

In another work [7], the authors propose a reliable and energy-efficient query-driven routing protocol that routes the request and response packets with low overheads. Here, the Base Station (BS) can choose the optimal path towards the target sensor node as it has a global view of the entire network topology consisting of the Euclidean Distances of neighboring nodes. While forwarding, instead of attaching the path with the request packet, a special mechanism is proposed. The mechanism provides a single integer value which is considered as the route's summary. Each intermediate node applies a specific function on this value to infer its successor until reaching the targeted sensor node. To reach a node, a single route is not always selected, instead, the routing task is distributed depending on the number of times the sensor nodes are involved. The simulation results and the comparison with other state-of-the-art methods have shown that our routing protocol balances the routing load, increases the network lifetime, and reduces energy consumption. Authors in [8] introduce a query processing method to improve the amount of valid data transmitted in the network. Firstly, they run some verifications using network connectivity and synchronization (among nodes) before injecting any query in the network. Then they propose a query processing model which adds new clauses for execution deadlines and ensures data validity for each query and response, respectively. To do the same, the method considers the query processing time, query response time and response transmission time. Then the authors also develop a data validity control algorithm that verifies the data validity time defined in the corresponding query.

Like WSN, NDN also works based on named data regardless of the identity of the node in the network. Thus, a few works on query processing over NDN are also reviewed.

The authors in work [16] state that though the patricia trie is widely used in the implementation of forwarding table for its memory efficiency, the choice of trie granularity is important. They compare the three different trie granularities - bit level, byte/character level, and component level. They evaluate the results based on a collection of datasets and performance metrics. To do this, the authors also introduce a new tool, called NameGen, which uses a Markov-based name learning model and generates pseudo-real datasets with different tunable name characteristics. The results show that bit-level trie can be considered as the best choice in terms of low memory requirement. Otherwise for fast look-up and update operations such as insertion, deletion, either of the character or component level trie is preferred. In a couple of works [17], [18], the authors introduce a special type of patricia trie data structures such as speculative and fingerprint-based patricia that can scale variable-length name forwarding to a large volume of prefixes. The structures make query searching and data forwarding faster and easier. The experimental results show that with the use of such a compact data structure, FIBs with a few million entries can fit in SRAM resulting in reduced searching time during query processing. The works in [19], [20] address the challenges of high storage, look-up, and communication overheads in the NDN routing method. To do the same, instead of storing the Link-State DataBase (LSDB) in all the routers, it is kept in some selective special nodes. The non-special nodes store a partial database only. The concept of the dominating set is used to select the special nodes. Algorithms are proposed for Data forwarding and update over simple network structures like grid networks as well as for real-world networks. The performance of the schemes is evaluated in terms of the overheads and the results show significant improvement over conventional schemes of NDN.

In another work [21], a simple protocol namely Real-time Data Retrieval (RDR) is proposed which can efficiently retrieve real-time data with minimal delays. To do the same, RDR provides the necessary information to consumers with the use of metadata packets. The information includes the ''most recent'' frame number, the number of interests that need to be pipelined to achieve timely retrieval of all frames, and frame segments that are produced in real-time. A real-time producer publishes metadata about its data production periodically or in response to interests. In interest packets, RDR makes use of the flags like ''FreshnessPeriod'' and ''MustBeFresh'' to let the consumers retrieve the fresh metadata. It limits the time the valid responses can stay in router caches as they cross the network, and it can also be effectively used to defend against overloads such as receiving redundant requests from the same node within a specified time.

Finally, we review a few works on query processing over NDN-based WSN utilizing the advantages of NDN notion in ad-hoc networks such as WSN.

In [22], a lightweight variant of CCN (Content-Centric Networking) protocol is proposed for WSNs namely CCN-WSN. Many aspects of the protocol are revised to cope with the memory, computation constraints, and communication pattern of WSN such as follows. The message format is redesigned. Messages are received and sent through Faces which are either a network interface or an interface to a local application. Then, a flexible naming strategy is proposed which extends the functionality of content names to add a small amount of data in interest messages. The evaluation results show that this lightweight variant is suitable for usage in WSNs. The authors claim that it is much more intuitive than other related approaches due to the stringent usage of interest, content, and implicit routing within the CCN domain. Using CCN on top of IEEE 802.15.4 in the future can provide an efficient solution with fewer overheads. In [23] the authors consider a caching method during query processing for wireless NDN-IoT networks namely pCASTING (probabilistic caching strategy for the internet of things). The method works with the freshness of data and considers the storage, energy constraints of WSN nodes. The evaluation results show that it outperforms the traditional NDN caching strategy in terms of data retrieval and network energy efficiency.

In [24], a Dataset Synchronization protocol for WSN (DSSN) is designed. Here, sensor nodes are divided into groups, and each group has a shared dataset. The DSSN ensures to keep the group's latest dataset always accessible from its active sensors through the dataset synchronization within each group. It enables dataset state synchronization utilizing the state vector that combines sensor names and data sequence numbers to represent all data produced by one sensor node and it also utilizes NDN's Interest aggregation and Data caching to optimize energy consumption. In another set of works [25]–[28], the authors present a query processing/Interest forwarding scheme in WSN with the notion of the basic NDN forwarding strategy. The schemes aim to reduce communication overheads in the network thereby reducing energy consumption and increase network lifetime. To achieve the same, the schemes avoid unnecessary flooding and reduce hop count to forward the interest towards the producer. The work in [25] is inspired by the data-centric directed diffusion routing technique of WSN whereas in dual-mode switching [26] flexibly switching from one mode to another depends on whether interest is found in Consumer FIB or not. The schemes in [27], [28] propose to use geographical location to retrieve the data. The simulation results show that the schemes outperform the state-of-the-art works in WSN by using NDN's forwarding strategy.

Summarily, the works from [2]–[8] discuss query-processing techniques in WSN. Though they try to lower down the overheads in terms of energy, they use the broadcast nature of communication for query forwarding, which incurs a decent amount of energy. Besides, the nodes in conventional schemes require high energy consumption in search of the best path. Now the works [16]–[21] in NDN address several issues such as complex routing and forwarding, high look-up, update, storage, and communication overhead during query processing. Out of these, a couple of works [16]–[18] discuss

TABLE 1. Summary of related works.

Category	Related Work	Communication overhead	Computation overhead	Storage	Energy overhead	Delay	Packet loss	Network Lifetime	Data Security
WSN	[2]	Low	-	Low	Low	-	-	-	-
	[3]	Low	Low	Low	Low	-	-	-	High
	[4]	Low	Low	Low	Low	Low	Low	High	High
	[5]	Low	Medium	Low	Low	Low	Low	-	High
	[6]	-	-	-	Low	Low	-	-	High
	[7]	Low	-	-	Low	-	-	High	-
	[8]	Medium	-	-	Low	Low	-	-	-
NDN	[16]	-	-	Low	-	-	-	-	-
	[17]	-	-	Low	-	-	-	-	Medium
	[18]	Low	-	Low	-	-	-	-	-
	[19]	Low	-	Low	-	-	-	-	-
	[20]	-	Low	Low	-	-	-	-	-
	[21]	-	-	High	-	Low	-	-	High
NDN-WSN	[22]	Low	-	Low	-	-	-	-	Low
	[23]	Low	-	-	Low	Low	-	High	-
	[24]	Low	-	-	Low	-	Medium	-	-
	[25]	Low	-	-	Medium	-	-	-	-
	[26]	Low	-	-	Low	-	-	High	-
	[27]	Low	-	-	Low	Low	-	-	-
	[28]	Low	-	-	High	-	-	Medium	-

special structures like patricia trie to reduce look-up, update, storage overheads but may produce incorrect query forwarding. Another couple of works [19], [20] lack in providing evaluation results with real-world networks using a real dataset. Whereas [21] discuss a real-time data retrieval protocol that reduces data retrieval delay and limits the caching time of the valid data. But the method does not provide any full cache management policy to decide the priorities of data removal. Further, the works [22]–[28] in NDN-based WSN reduce communication overheads, energy consumption, data retrieval delay in nodes by decreasing flooding overhead. Out of them, a few [22] lack lightweight security measures and a few [23] still follow a periodic broadcast mechanism to forward the generated query towards sensor nodes in the network. A few schemes also suffer from packet losses and decreased data availability due to packet collisions [24]. The directed diffusion-based technique [25] incurs significant energy due to its broadcast nature of Interest forwarding and the other method [26] is silent about the complexity of its switching technique for the decision of interest flooding. The works in [27], [28] are not discussed for complex scenarios or large-scale networks. The related studies on query processing over WSN, NDN and, NDN-based WSN are compared and summarized in TABLE 1.

III. SYSTEM MODEL

This section provides network architecture along with background studies on using NDN in WSN and the low overhead hash algorithms.

A. NETWORK ARCHITECTURE

We consider a hierarchical network that is divided into several regions with individual region heads (RHS). Each of the regions in turn is divided into several clusters with individual cluster heads (CHs). This architecture is the most widely used architecture for energy-efficient data communication. Moreover, considering such a multilevel hierarchical architecture, many nodes can be accommodated within the network [20]. Figure 1 shows such a system model. Here the entire area is divided into five regions and each of the regions has a varying number of clusters. The area covered by a region is shown by the dotted outer circle whereas the inner circle is shown in the figure representing the communication range of an RH. All the CHs of a region are within the communication range of the RH. Nodes are deployed throughout the network area and form a cluster in a self-organizing manner. Unlike other sensor nodes in the network, the sink is a powerful node in terms of storage, computation, and energy consumption. Users interact with the network by the sink. We assume the nodes are static and deployed randomly in an area.

B. BRIEF ON NDN

The main building block of NDN is the named content chunk. Any packet in the network whether it is interest or data must have a name. Instead of announcing IP prefixes, a node in NDN announces name prefixes for the content that the node can serve. The names of interest and corresponding data packets must be the same, and they are always hierarchically structured. For example, the temperature of today in the Haldia industry belt may be structured as - *temperature/haldia_industrial_zone/belt_001/today*, where ''/'' is not part of the name but specifies a boundary between the name components. The NDN nodes see the boundaries between the components; they do not know the meaning of a name [10], [11].

In NDN, each node/router stores the three data structures Content Store (CS), Pending Interest Table (PIT)

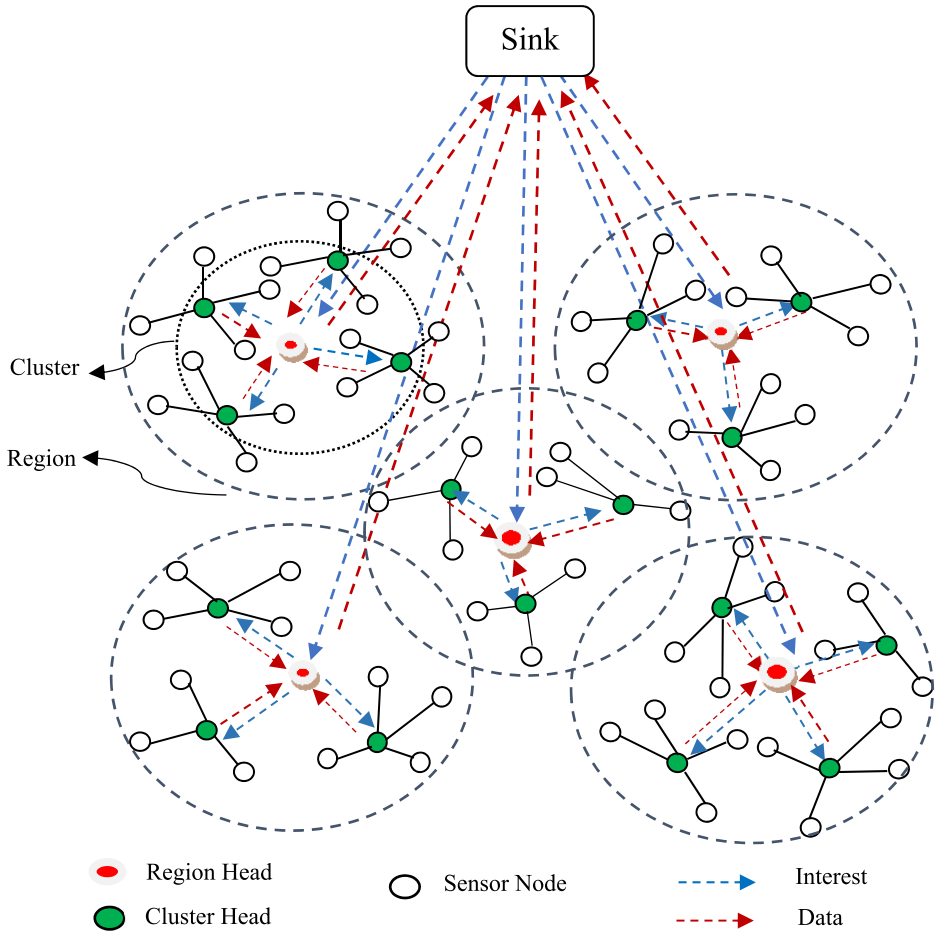


FIGURE 1. System model.

and, Forwarding Information Base (FIB) [13]. Here, each router/node has the CS to cache data packets passed by and, therefore, frequently requested content by consumers is cached at multiple routers in the network. NDN supports data distribution and data sharing with the help of such in-network content caching. This content caching at routers enables data delivery to consumers from the nearest location with minimal latency and thereby improves overall network performance. On the other hand, PIT keeps track of the past requests which are received but un-responded. This gives NDN a significant feature. It helps in forwarding decisions on how to handle an interest without the knowledge of the source or destination. Further, the FIB is a routing table that maps name components to interfaces. Whenever an interest arrives, an NDN router first checks its CS. If the corresponding data exists in the CS, the router returns the data packet on the interface from which the interest came. Otherwise, the router searches the name in its PIT, and if a matching entry exists, it only adds the incoming interface and does not forward the interest again. If there is no such PIT entry, a new entry is created. Then the router forwards the interest towards the data producer based on information in the FIB. On the contrary, data packets take the reverse path of interests. When a data packet reaches a router, the router searches and deletes the matching PIT entry and forwards the data to all downstream interfaces listed in that PIT entry. It then removes the PIT entry and caches the data in CS.

In the proposed scheme, we adopt the concept of the current NDN routing protocol NLSR (Link State Routing for Named data) [29] to incorporate the advantages of such a protocol in ad-hoc networks like WSN. After reviewing the routing protocols of several ad-hoc networks such as WSN and MANET [30]–[32], it has been noticed that to reduce the energy consumption in the routing of such networks, the in-built NDN features can be used.

The use of CS, PIT in a node can reduce the unnecessary communications in interest/query processing and thus the energy requirement. The hash digest creation of each data packet maintains the message integrity and thus provides data security in such networks. However, as NDN routing is not meant for such ad-hoc networks, we modify the NDN routing to make it implementable in WSN.

C. USING NDN IN WSN

As WSN is a resource-constrained network, we propose to use the NDN data structures only in selective nodes of the

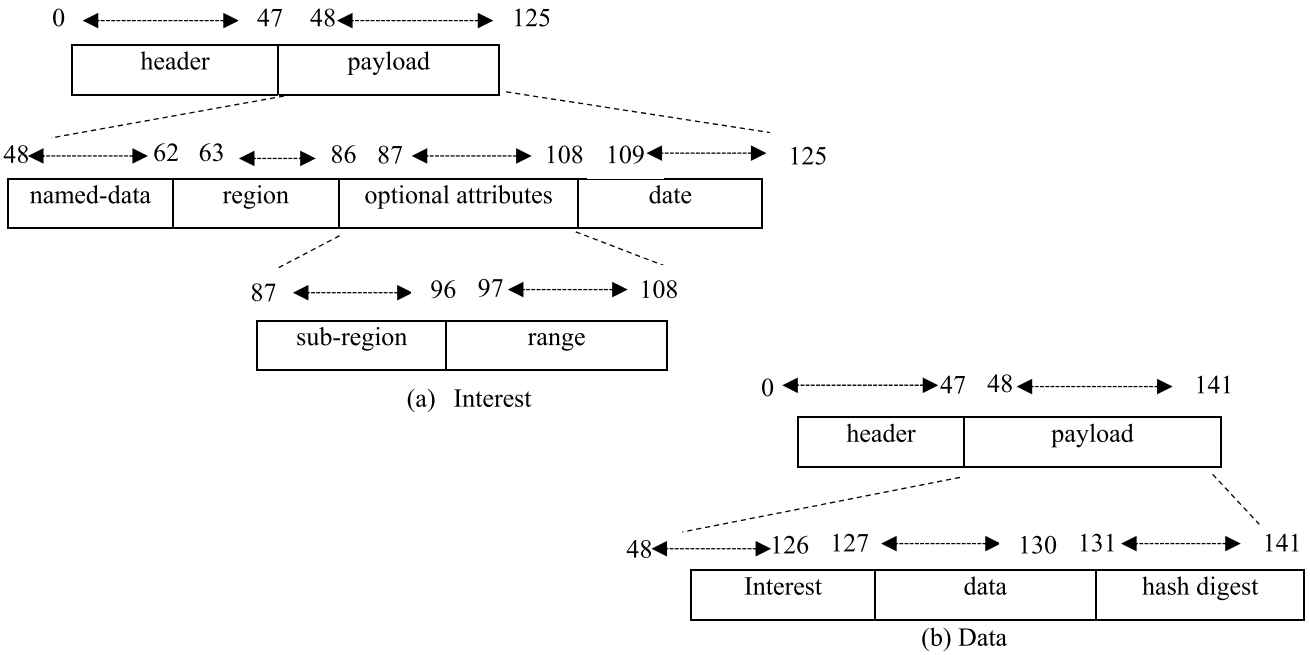


FIGURE 2. Interest and data packet format.

network. The sink node and the RH contain the **CS** which consists of (INTEREST, DATA) implying the interest name and the corresponding content, respectively. However, CHs do not maintain any CS considering these nodes are resource-constrained. The **PIT** entries are the same in the sink, RH, cluster head (CH), and these consist of (INTEREST, TIME_STAMP) where TIME_STAMP implies the time instance at which interest is forwarded towards the producer. The sink node has one more attribute CONSUMER implying the requesting consumer. The **FIB** entry of sink is consisting of a region or landmark name (REGION/L_MARK), location (RH_LOC/L_LOC), id (RH_ID/LOC_ID), radius (R_RADIUS) whereas the FIB of RH consists of CH location (CH_LOC), id (CH_ID), and radius (C_RADIUS) of the CHs under the RH. The sensor nodes within a CH do not need to maintain any data structures. It is assumed that each RH of the network knows the RH_LOC of all other RHs. Like NDN nodes, there is the existence of data structures like CS, PIT in WSN nodes. These structure-based features saves energy by reducing searching for producer's data in the network. Further, in NDN, each data is digitally signed while sending it back from the producer towards the consumer. In NDN, for the creation of the hash digest in the signature, a hash function like SHA-256 is used. But it is not a suitable hash function to be used in energy-starved networks like WSN. Thus, we use an existing lightweight hash (LOCHA) [15] in place of SHA-256, only to create a digest of each data.

D. BRIEF ON LOCHA

This section provides a brief overview of LOCHA which has been developed with a target to produce a hash-digest with a fixed and relatively small length for the energy-starved environment such as WSNs. It is a lightweight, one-way, cryptographic hash algorithm where the nodes can successfully run the algorithm with low energy. The hash algorithm LOCHA is made lightweight by using low overhead operations such as MOD, SWAP, etc. The scheme also uses two substitution tables to obscure the relationship between the plain text and the ciphertext. The algorithm also fulfills all the basic properties such as preimage resistance, collision resistance of a one-way unkeyed hash function. However, to avoid imposing high computation and communication overheads, instead of creating any signature, the digest is sent along with the original data to check message integrity.

E. PACKET FORMAT

We consider the packet formats shown in Figure 2 for our proposed query handling scheme. Figure 2 (a) shows the interest packet format which consists of the header and payload. The header size (according to the RIME protocol stack used in Contiki) is 48 Bytes (0-47) [37]. The remaining in the interest packet is the payload. The payload is the *interest name* consisting of the hierarchical structure - */named-data/region/sub-region/range/date/*. The *sub-region* and *range* are the optional attributes as shown in the figure. Thus, based on the queries the interest may have 3-5tuples. Though the scheme is not application-specific, here the push-based queries such as environment monitoring are considered where information on temperature, humidity, toxic gases, etc. are enquired and retrieved. So, the size of *named data* is considered to be a maximum of 15 Bytes (48-62). The query may ask for such information about a region. We consider that a region may have multiple sub-regions. For example, an industrial area may have several industrial belts. In our system model

(Section III.A), the *sub-region* in the interest packet implies a *cluster*. If a query intends to retrieve data from a specific *sub-region*, then the *sub-region* is specified in the query, and the data is retrieved from the corresponding cluster. Otherwise, the data is retrieved from the entire region. The sizes of *region* and *sub-region* are considered to be a maximum of 24 Bytes (63-86) and 10 Bytes (87-96) respectively. Further, a query may involve a *range* if the requested information is about a specified range. The size of the *range* is considered to be a maximum of 12 Bytes (97-108). Finally, the *date* attribute contains either the *current date* or *today* in a real-time query. However, the non-real-time query contains any previous date or range of dates. So, the size of the *date* is considered to be a maximum of 17 Bytes (109-125). Thus, the maximum size of the interest packet size is $48 + 15 + 24 + 10 + 12 + 17 = 126$ Bytes.

Figure 2 (b) shows the data packet format which consists of the header and the payload. The header size is 48 Bytes (0-47). The payload is a 3-tuple entry. It consists of the *interest/data name*, *data value* (retrieved response i.e., the content), and the *hash digest* of the data. As mentioned earlier, the maximum size of *interest name* is 78 Bytes (48-125). The size of the *data value* is considered to be 4 Bytes (126-129) as it should be float. The *hash digest* is obtained by applying the hash function LOCHA to the *data value*. The function produces 96 bits or 12 Bytes (130-141) digest. Thus, the maximum size of the data packet is $48 + 78 + 4 + 12 = 142$ Bytes.

IV. INTEGRITY PRESERVING LOW-OVERHEAD QUERY HANDLING OVER NDN BASED WSN

This section provides the proposed scheme **I**ntegrity **P**reserving **L**ow-overhead **Que**ry handling ov**e**r NDN-based WSN (IPLQueeN). It handles both real-time and non-real-time queries in hierarchical cluster-based WSN.

A. PRINCIPLE OF OPERATIONS

The entire query processing scheme has two phases:

- Query phase or Interest forwarding
- Response phase or Data forwarding

The activities in Interest forwarding, in turn, occur in three phases as follows:

- Sink-RH
- RH-CH
- CH-Sensor

Similarly, the activities in Data forwarding occur in three phases in reverse order of the activities in Interest forwarding.

- Sensor-CH
- CH-RH
- RH-Sink

1) INTEREST FORWARDING

Sink-RH: For a real-time query, whenever an interest reaches the sink, firstly it checks the CS and if the data is not found subsequently it checks PIT. If the same interest is not pending in PIT, it searches the FIB to know the corresponding RH, and accordingly, the interest is forwarded to the respective RH. The interest may also ask for retrieving data from multiple regions. Such interests specify a region and a range around the region (R_{query}) for the intended data. In such cases, the sink checks all RH_IDs to find which regions fall within the range specified in the interest. Then the interest is forwarded to all such respective RHs, for which the following condition (1) is true.

$$\sqrt{(x_i - x_i')^2 + (y_i - y_i')^2} < R\_RADIUS + R_{query} \quad (1)$$

where (x_i, y_i) is the RH_LOC specified in the query and (x_i', y_i') is the RH_LOC which needs to be checked for overlapping.

For a non-real-time query, firstly CS of the sink is checked for the intended data. If the same is not found, it is checked in PIT to find whether the same interest is already requested. If it is already pending, the interest is not forwarded again, only the incoming interface is added; otherwise, a new entry for the interest is created. Subsequently, FIB is checked to retrieve the RH_LOC(s) according to the region specified in interest. Then the interest is forwarded to the RH(s).

RH-CH: For a real-time query, when an interest reaches an RH, the PIT of the RH is searched. If the matching entry is not found, FIB is searched to find the corresponding CH_IDs. Accordingly, the interest is forwarded to the respective CHs for which the following condition (2) is true.

$$\sqrt{(x_i - x_{ij})^2 + (y_i - y_{ij})^2} < R_{query} \quad (2)$$

where (x_{ij}, y_{ij}) is CH_LOC under the RH with location (x_i, y_i). The R_RADIUS is the radius of the region for which the condition is being checked. Now, if the specified region in interest is also overlapping with multiple regions (checked by equation (1)), then all the clusters of those regions may or may not come within the range. The CH_IDs of the overlapped regions fall within the specified region, are found by condition (3). In this case, also, the RHs forward the interest to the respective CHs for which the following condition (3) is true.

$$\sqrt{(x_i - x_{ij}')^2 + (y_i - y_{ij}')^2} < C\_RADIUS + R_{query} \quad (3)$$

where the $\left(x_{ij}', y_{ij}'\right)$ is the location of the overlapped (other) region clusters and C_RADIUS is the cluster radius.

For a non-real-time query, the RH checks its CS to retrieve the requested data. If the data is not found in CS, the interest is discarded.

CH-Sensor: If a CH receives interest from its RH, the interest is forwarded to all the sensor nodes under the cluster.

An example scenario showing how the regions can be overlapped for a real-time query is presented in Figure 3.

2) DATA FORWARDING

Sensor-CH: Upon receiving the interest, a sensor node (within a cluster) applies LOCHA to the collected sensory

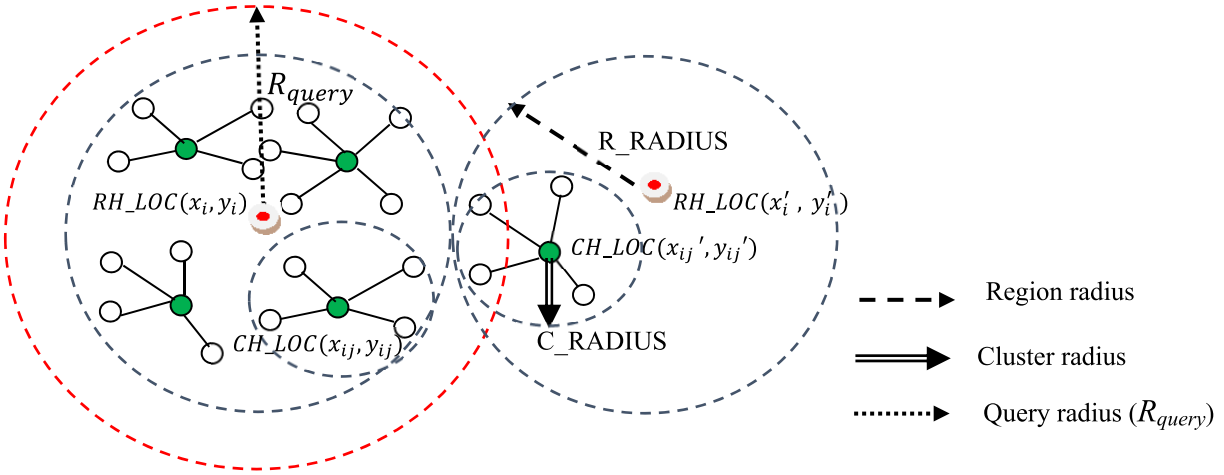


FIGURE 3. Overlapping of regions/clusters.

data and generates the digest. Then, it adds the digest with the data and sends the response to the CH.

CH-RH: The CH receives the signed data from multiple sensor nodes, verifies each data applying the same LOCHA. If the verification is successful, the CH aggregates the data received from the nodes and applies the LOCHA on the aggregated data. The response is forwarded toward the RH.

RH-Sink: Similar to the CHs, RH also aggregates data received from each CH under it after successful verification of each of the data using LOCHA. Next, the aggregated data and the generated hash digest are sent to the sink. The sink may receive the data from a single RH or multiple RHs. However, it finally verifies the data and upon successful verification, it sends back the data to the requesting consumer. While the intended data against interest is received by CHs and subsequently by RHs and sink, the corresponding interest entries in each PIT are deleted.

We observe from the above discussion that in our proposed scheme data is retrieved based on the region specified in the interest. It does not contradict the notion of NDN which considers data irrespective of who requests/delivers it implying location or the address of the producer is not important. Here, unlike IP networks, for the successful execution of the query processing, only the location coordinates of the nodes are used locally, instead of the IP addresses.

B. ALGORITHMS

The Interest forwarding and Data forwarding algorithms considering both real-time and non-real-time queries are running in different components of the network architecture such as sink, RH(s), CH(s), etc. Thus, the algorithm is presented from a global view mentioning the activities (in sequence) of each of the components involved during processing a query.

The notations presented in TABLE 2 are used in the algorithms.

1) ALGORITHM FOR INTEREST FORWARDING

In the interest/query forwarding, once a real-time query or interest reaches a sink, it is forwarded, based on the regions specified in the interests. Now the designated RHs under the sink receives the query. Then the RHs, in turn, forward the query toward their required CHs. The CHs finally, broadcast the query to the corresponding sensor nodes under them. Upon receiving the interest, the nodes within a cluster retrieve the corresponding data. So, the sink, some of the RHs, their CHs, and sensor nodes get involved in the query processing. But, due to the in-network caching mechanism of NDN, for a non-real-time query, the corresponding data can be found in CS of sink or RHs. So, in this case, CHs and nodes within a cluster need not participate in query processing. The algorithm for Interest forwarding from the sink to sensor nodes is as follows.

TABLE 2. Table of notations.

Notation	Meaning
I	Interest
D	Data
R_{query}	The range specified in the query(interest)
D_{digest}	Digest on D from one sensor node

2) ALGORITHM FOR DATA FORWARDING

During the query response phase, the hash function LOCHA is applied on each data transmitted from producer towards consumer i.e., from sensor nodes towards the sink. The data along with its digest are forwarded back to the sink following the exact reverse path of the interest. While traveling back, the function is applied in each node to verify the received data. Thus, the integrity of the data is preserved at each hop along the reverse path. Once the data at each node is received and verified, the corresponding PIT entry is deleted from the node. In this manner, whenever the RHs and sink receive it, they keep a copy of the data in their respective CS. Finally, the sink sends the data to the requesting consumer. The algorithm for Data forwarding from the sensor node to sink is as follows.

Algorithm for Interest Forwarding:

Begin

/* Action executed by Sink */

1. Receive I
2. **if** (Is_nonrealtime(I) == True)
   /**non-real-time query*/*
3. **if** (NonRealTimeQuery (I, *Consumer*) == True)
4. **exit**
5. **else**
6. Forwarding_decision(I)
7. **end if**
8. **else** /* *real-time query* */
9. Forwarding_decision (I)
10. **end if**

/* Action executed by RH */

11. Receive I
    /* *non-real-time data* */
12. **if** (Is_nonrealtime(I) == True)
13. **if** (NonRealTimeQuery(I, Sink) == False)
14. Send back D *not found*
15. **exit**
16. **end if**
    /* *real-timedata* */
17. **else**
18. **if** (Check_PIT(I) == True)
19. **if** (R_{query} exists in I)
20. **if** (same region)
    /**Checking overlapping of CHs within same region*/*
21. **for** each CH_ID of FIB
22. RealTimeQuery(I, C2, RH_ID,CH_ID)
    /**C2 = Condition (2)*/*
23. **end for**
24. **else** /**Checking overlapping of CHs of different region */*
25. **for** each CH_ID of FIB
26. RealTimeQuery(I, C3, RH_ID,CH_ID)
    /* C3 = Condition(3)*/
27. **end for**
28. **end if**
29. **else**
30. **for** each CH_ID of FIB
31. Forward I to CH_ID
32. **end for**
33. **end if**
34. **end if**
35. **end if**

/* Actions executed by CH */

36. Receive I
37. **if** (Check_PIT(I) == True)
38. Forward I to all Sensor nodes
39. **end if**

/*Action executed by Sensor node*/

40. Receive I
41. Retrieve D

/*Functions*/

42. Is_nonrealtime(I)
43. **if**((date != current date)||(date != today))
44. **return** True
45. **else**
46. **return** False
47. **end if**
48. NonRealTimeQuery(I, *Receiver*)
49. Check CS
50. **If** I found
51. Send D to *Receiver*
52. **return** True
53. **else**
54. D not found
55. **return** False
56. **end if**
57. Forwarding_decision(I)
58. **if**(Check_PIT(I) == True)
59. Retrieve RH_ID from FIB
60. Forward I to RH_ID
61. **if**(R_{query} exists in I)
62. **for** each RH_ID' of FIB
    /* *RH_ID' = RH id of region other than the region specified in query* */
63. RealTimeQuery(I, C1, RH_ID, RH_ID')
64. **end for**
65. **end if**
66. **end if**
67. Check_PIT(I)
68. **if** (I present in PIT)
69. **if**(Sink) /* *the function runs in Sink* */
70. Add requesting consumer
71. **return** True
72. **else** /* *the function runs in RH* */
73. Discard I
74. **return** False
75. **end if**
76. **else**
77. Create a new PIT entry
78. **return** True
79. **end if**
80. RealTimeQuery(I,Condition,REF_location, FIB_location)
81. Retrieve RADIUS from FIB /**R_RADIUS if at Sink and C_RADIUS if at RH */*
82. Check condition
83. **if** true
84. Forward I to FIB_location
85. **end if**

End

Algorithm for Data Forwarding:

Begin

/* Action executed by Sensor node */

1. Apply LOCHA on D to create D_{digest}
2. Send D and D_{digest} to CH

/* Action executed by CH */

3. **for** each D, D_{digest} from n_{sensor}
   /* n_{sensor} = *no. of sensor nodes within a Cluster* */
4. **if**(Verify(D, D_{digest}) == True)
5. Add in D_{array1} /* D_{array1} *is array of D from all sensor nodes* */
6. **end if**
7. **end for**
8. Remove I from PIT
9. Data_forward(D_{array1}, n_{sensor}, RH)

/* Action executed by RH */

10. **for** each D, D_{digest} from n_{CH}
    /* n_{CH} = *no. of CHs within a Region* */
11. **if**(Verify(D, D_{digest}) == True)
12. Add in D_{array2} /* D_{array2} *is array of D from all CHs* */
13. **end if**
14. **end for**
15. Remove I from PIT
16. Data_forward(D_{array2}, n_{CH}, Sink)

/* Action executed by Sink */

17. **for** each D, D_{digest} from n_{RH}
    /* n_{RH} = *no. of RHs* */
18. **if**(Verify(D, D_{digest}) == True)
19. Add in D_{array3} /* D_{array3} *is array of D from all RHs* */
20. **end if**
21. **end for**
22. Remove I from PIT
23. Data_forward(D_{array3}, n_{RH}, *Consumer*)

/*Functions*/

24. Verify(*D, digest1*)
25. Apply LOCHA on D to create *digest2*
26. **if**(*digest1* == *digest2*)
27. **return** True
28. **else**
29. **return** False
30. **end if**
31. Data_forward(D_{arr}, n_s, *Receiver*)
    /* n_s = *number of nodes* */
32. Create *data* = Aggregate $D_{arr}[i]$
    /* $i = 1, 2, \ldots, n_s$, D_{arr} *is array of Ds* */
33. **if** CS exists
    /* *if CS exists in the sender node* */
34. Add *data* in CS
35. **end if**
36. Apply LOCHA on *data* to create *digest*
37. Send *data* and *digest* to *Receiver*

End

C. ILLUSTRATIVE EXAMPLES

Let us consider a WSN with 84 nodes. We consider 4 RHs with 4 CHs each and 4 sensor nodes under each of the CHs. Thus, there are 4(RHs) + 16 (CHs) + 64 (sensor nodes) = 84 nodes. The values of R_RADIUS and C_RADIUS are taken as 50 meters and 20 meters respectively. The proposed IPLQueeN is illustrated with the help of two examples – one real-time query and one non-real-time query. Figure 4 presents the sequence diagram of the example.

1) EXAMPLE OF REAL-TIME QUERY

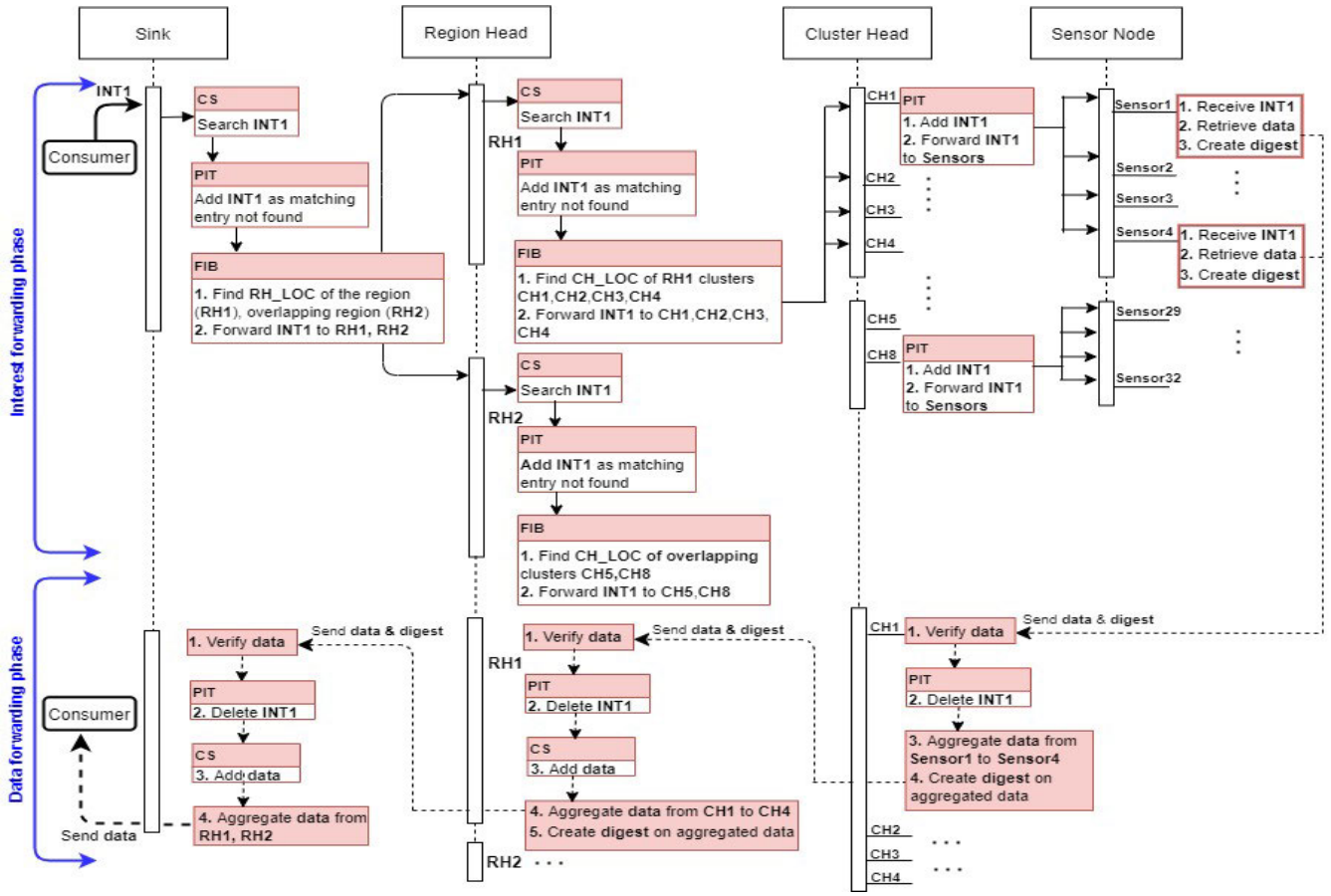
Let us consider an interest **INT1** = '***temperature/haldia_industrial_zone/around_50m/today***' retrieves the today's average temperature of the area around 50 meters of the head of region haldia_industrial_zone i.e., RH1. So, it asks for the data in real-time. Figure 4 (a) presents the Interest forwarding and Data forwarding phase for INT1. Now, following the algorithm (Section IV.B.1), the steps for forwarding the interest are as follows.

As the Date in the interest is 'today', and as there is no such data in CS and subsequently no pending interest in the PIT, the interest '*temperature/haldia_industrial_zone/around_50m/today*' is added to the PIT. Then, the location of RH1is found from the FIB of the sink. Then condition (1) is checked with $R_{query} = 50$ meters, $R\_RADIUS = 50$ meters, and the regionRH2is found to be overlapping with RH1. So, the interest is sent to RH1 and RH2. This is as per lines 7-10 of the Algorithm.

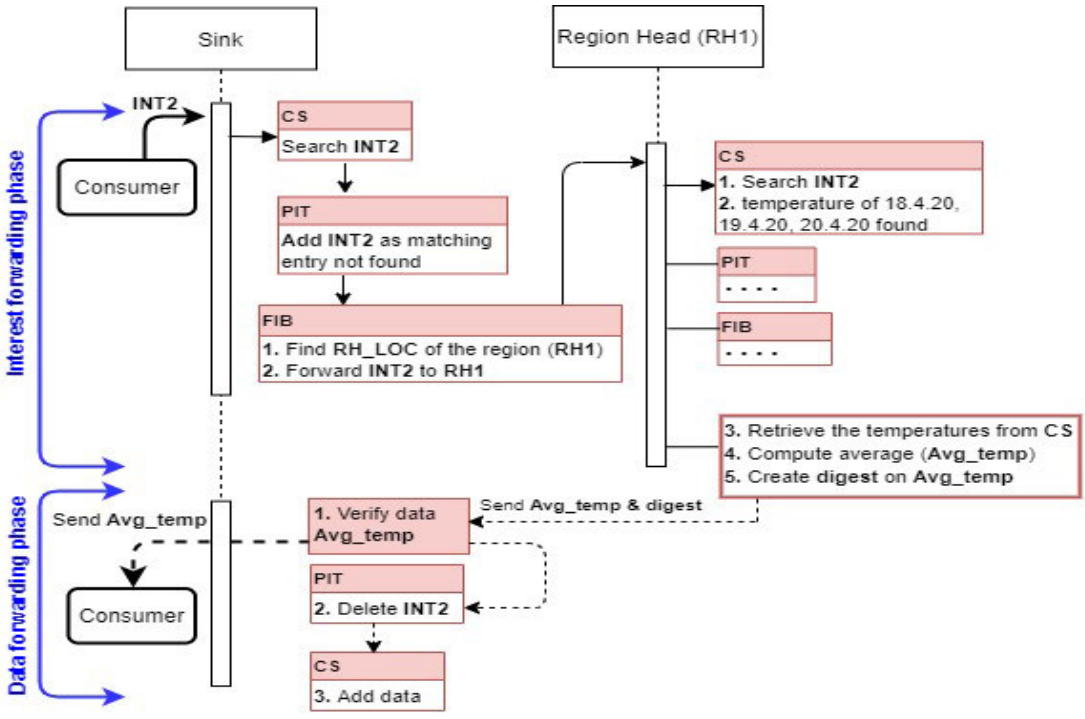
In RH1 (Algorithm line 11-35), as the data is not found in CS and subsequently in PIT, a new PIT entry is created. Then, condition (2) is checked, and it is found from the FIB of RH1 that clusters CH1, CH2, CH3, CH4 come under 50 meters range (of RH1 location). In RH2, after adding the PIT entry, condition (3) is checked with $C\_RADIUS = 20$ meters to find the overlapping clusters of that region. As a result, the clusters CH5 and CH8 of RH2 are found within 50 meters range of RH1. The interest is forwarded to the clusters within the specified range from the corresponding RH1 and RH2. Following lines 36-39 of the Algorithm, in CH1, the interest is added to the PIT as a new entry. Then, the interest is forwarded to all sensor nodes under it. The same operations are performed in CH2, CH3, CH4, CH5, and CH8 as shown in Figure 4(a). Now, the steps for forwarding the data are as follows.

According to the Algorithm (Section IV.B.2, Algorithm lines 1-2), each sensor node within a Cluster senses the temperature (the *data*) and applies LOCHA on it to create the *digest*. Then, each node sends the *data* and the *digest* to its CH.

Now, each of the CH1, CH2, CH3, CH4 under RH1 and CH5, CH8 (Algorithm lines 3-9) under RH2 verify the *data* with the help of LOCHA. The corresponding PIT entries are removed. Then each CH aggregates all the *data* received from the sensor nodes under them and creates a *digest* on the



(a) Real-time query



(b) Non-real-time query

FIGURE 4. Example of real-time and non-real-time query processing in IPLQueeN.

aggregated *data*. The aggregated *data* and the *digest* are then forwarded to the respective RHs. Figure 4(a) shows the Data forwarding phase by CH1 only.

Next, similar to the CHs, both the RH1 and RH2 (Algorithm lines 10-16) verify the received *data*. Then the corresponding PIT entries are removed, and *data* is added in CS. The RHs compute the aggregated *data* received from all the corresponding CHs. Next, they create the *digest* on the aggregated *data* and forward the *data* and the *digest* to the sink. Finally, the sink verifies (Algorithm lines 17-23) the *data* received from both the RHs and, upon successful verification, the *data* is added in the respective CSs and the corresponding PIT entries are removed. Then, the sink computes the average temperature which is sent back to the requesting Consumer.

2) EXAMPLE OF NON-REAL-TIME QUERY

Let us consider an interest **INT2** = '***temperature/haldia_industrial_zone*/18.4.20 − 20.4.20**' retrieves the temperature of the 3 days (***18.4-20.4***) from the *industrial_zone*region of *haldia*. So, it asks for the data in non-real-time. Figure 4(b) presents the Interest and Data forwarding phase for INT2. The data against this interest may be found in the CS of the sink or the RH. The steps for forwarding the interest are as follows.

As the interest is not in CS and subsequently not in PIT of the sink, the interest ***temperature /haldia_industrial_zone/* 18.4.20 − 20.4.20** is added to the PIT of sink following the Algorithm (Section IV.B.1; lines 1-10). The location of ***haldia_industrial_zone***i.e., RH1 is then found from the FIB and the interest is forwarded to RH1. Following the Algorithm (line 11-16), the ***temperature*** of RH1 for ***18.4-20.4*** is retrieved from the CS. Now, the steps for forwarding the data are as follows.

According to the Algorithm (Section IV.B.2; line 10-16), the RH1 computes the average temperature (*Avg_temp)* and the *digest*. Then the *data Avg_temp* and the *digest* are sent to the sink from the RH1. The sink verifies the *Avg_temp* (line 17-23), and upon successful verification, INT2 is deleted from PIT; subsequently, the *data* is added in CS. Then it is sent to the requesting Consumer.

D. JUSTIFICATION IN FAVOUR OF LOW OVERHEAD CLAIM

We claim that the proposed scheme **IPLQueeN** is low overhead. The reasons are stated as follows.

- Unlike flat architecture, the number of participating nodes in the proposed hierarchical structure in our scheme is quite low as no interest broadcast is required to receive data. It results in low communication overhead.
- For many of the queries, the data can be found from the CS of the sink or just from the next level of the hierarchy. So, searching time in data retrieval reduces. It results in low computation overhead.
- The proposed scheme implements the notion of NDN in WSN. But, unlike NDN, it does not need to store all data structure CS, PIT, FIB in all the resource-constrained nodes. As a result, the storage overhead is low in the network.
- We use the LOCHA hash function to create a digest of the data while sending back the data thus maintaining message integrity. LOCHA is a low-overhead hash function as it crates only 12-bit digest and requires only 2952 clock cycles which are lower compared to other hash functions. Moreover, it takes 1.21 years to break the basic cryptographic property like collision resistance of LOCHA [15]. Thus, our scheme **IPLQueeN** becomes low overhead without compromising with the basic security features.

V. PERFORMANCE ANALYSIS

This section evaluates the performance of IPLQueeN both theoretically and through simulation.

A. THEORETICAL ANALYSIS

The IPLQueeN is evaluated theoretically in terms of storage, computation, and communication overhead. In this analysis, we consider a network of N nodes and the total number of the sink, RH, CH, and sensor nodes are referred as n_{Sink}, n_{RH}, n_{CH} and n_{Sensor}, among which number of the participating sink, RHs, CHs, and sensor nodes are $n'_{Sink}, n'_{RH}, n'_{CH}$ and n'_{Sensor} respectively. We consider one sink, and this implies $n_{Sink} = n'_{Sink} = 1$. This analysis is done considering sensor nodes with the specification of Tmote sky [33]. Also, as mentioned in Section III.E, interest packet size is a maximum of 78 Bytes and the maximum size of the data packet is 94 Bytes.

1) STORAGE OVERHEAD

Assuming each of CS, PIT, FIB stores n number of entries, the storage overhead is estimated as follows, for the worst-case scenario i.e., considering the maximum size of the fields.

A sink stores 2-tuple (INTEREST, DATA) in CS. So, one entry of the CS stores 78(INTEREST) + 4(DATA) = 82 Bytes. Similarly, one entry of the PIT at the sink stores 3-tuple (INTEREST, CONSUMER, TIME_STAMP) implying 78(INTEREST) + 4(CONSUMER) + 4(TIME_STAMP) = 86 Bytes. Then, the FIB at the sink stores 4-tuple (REGION/L_MARK, RH_LOC/L_LOC, RH_ID, R_RADIUS) which are assumed to require a maximum of 25 + 8 + 4 + 4 = 41 Bytes for one entry. Further, the sink stores two substitution tables to generate a LOCHA [15] based hash digest for verifying the integrity of the received response and the authenticity of the sender. It requires storing (97 + 68) = 165 integers which need (165 × 4) = 660 Bytes for two such tables. Hence, the sink needs to store $\{n \times (82 + 86 + 41)\} + 660$ Bytes = $(209n + 660)$ Bytes.

Similar to the sink, an RH stores 82 Bytes for one entry in CS. The PIT stores 2-tuple (INTEREST, TIME_STAMP) which is 78(INTEREST) + 4(TIME_STAMP) = 82 Bytes for one entry. The FIB of one entry needs a 3-tuple (CH_LOC, C_RADIUS, CH_ID) requiring (8 + 4 + 4) = 16 Bytes.

Thus, one RH stores $[\{n \times (82 + 82 + 16)\} + 660]$ Bytes $= (180n + 660)$ Bytes including a hash digest of 660 Bytes.

Finally, similar to the RH, a CH stores 82 Bytes for one entry in PIT. So, one CH stores $\{(n \times 82) + 660\}$ Bytes $= (82n + 660)$ Bytes including a hash digest of 660 Bytes. Each sensor node (in the lowest level of the hierarchy) stores only the substitution tables required in the hash computation consisting of a total of 660 Bytes.

a: REAL-TIME QUERY

For real-time query, in the Interest forwarding phase, the interest travels from the sink towards sensor nodes. The sink transmits the interest to RH(s). Each of the concerned RHs receives such interest and transmits the same towards CHs. In the response or Data forwarding phase, the data travels back following the exact reverse path of the interest. Therefore, the storage overhead $(O_{storage})$ in the network can be expressed as follows:

$$O_{storage} = \{n'_{Sink} \times (209n + 660) + n'_{RH} \times (180n + 660) + n'_{CH} \times (82n + 660) + (n'_{Sensor} \times 660)\} \text{ Bytes.}$$

b: NON-REAL TIME QUERY

For non-real-time queries, in the worst case, the data for an interest may not be found in the sink. Instead, it may be found in the RH according to the specification in the query. So, the worst-case storage overhead $(O_{storage})$ for the network can be expressed as follows:

$$O_{storage} = \{n'_{Sink} \times (209n + 660) + n'_{RH} \times (180n + 660)\} \text{ Bytes.}$$

2) COMMUNICATION OVERHEAD

This section provides communication overheads for both real-time and non-real-time queries. This overhead is considered for handling one query.

a: REAL-TIME QUERY

In the Interest forwarding phase, the sink transmits interest of 78 Bytes. Each of the concerned RHs receives such interest and transmits the same towards CHs. So, the communication overhead for RHs is $(n'_{RH} \times 78 + n'_{CH} \times 78)$ Bytes. Next, similar to the RHs, the CHs receive the interest and broadcast towards the sensor nodes under them. Thus, the communication overhead for CHs is $(n'_{CH} \times 78 + n'_{CH} \times 78) = (n'_{CH} \times 156)$ Bytes. In this Interest forwarding phase, the sensor nodes receive only such interest of 78 Bytes. Hence, in this phase, the total communication overhead is $(n'_{RH} \times 78) + (n'_{RH} \times 78 + n'_{CH} \times 78) + (n'_{CH} \times 156) + (n'_{Sensor} \times 78) = 78 \times (2 \times n'_{RH} + 3 \times n'_{CH} + n'_{Sensor})$ Bytes.

In the response phase, the concerned sensor nodes (n'_{Sensor}) transmit data (4 Bytes), hash digest (12 Bytes) along with the interest (78 Bytes) which is equal to 94 Bytes. Thus, communication overhead for sensor nodes is $(n'_{Sensor} \times 94)$ Bytes. The CHs receive 94 Bytes from each of the nodes under it and aggregate data part keeping the received packet size 94 Bytes. After that, each CH sends the 94 Bytes response to the respected RH. So, the communication overhead for CHs is $(n'_{Sensor} \times 94 + n'_{CH} \times 94)$ Bytes. Similarly, the RHs receive 94 Bytes from each of the concerned CHs (n'_{CH}) under it and send the response to the sink after aggregating the data part keeping the packet size 94 Bytes. So, communication overhead for RHs is $(n'_{CH} \times 94 + n'_{RH} \times 94)$. Finally, the sink receives 94 Bytes from each of the concerned RHs (n'_{RH}). Hence communication overhead for the sink is $(n'_{RH} \times 94)$ Bytes. Thus, the total communication overhead in the response phase is $(n'_{Sensor} \times 94) + (n'_{Sensor} \times 94 + n'_{CH} \times 94) + (n'_{CH} \times 94 + n'_{RH} \times 94) + (n'_{RH} \times 94) = 94 \times (2 \times n'_{Sensor} + 2 \times n'_{CH} + 2 \times n'_{RH})$ Bytes.

According to the specification of the CC2420 transceiver of Tmote Sky, energy consumption for transmitting and receiving 1-byte of data is 0.00189 mJ and 0.00167 mJ respectively [34]. Thus, the communication overhead in terms of energy (E_{comm}) can be expressed as follows:

$$E_{comm} = \left[\left\{78 \times \left(n'_{RH} + 2 \times n'_{CH}\right) + 94 \times \left(n'_{Sensor} + n'_{CH} + n'_{RH}\right)\right\} \times 0.00189\right] + 78 \times \left(n'_{RH} + n'_{CH} + n'_{Sensor}\right) + 94 \times \left(n'_{Sensor} + n'_{CH} + n'_{RH}\right)\} \times 0.00167]mJ$$

b: NON-REAL-TIME QUERY

In the worst case, in the Interest forwarding phase, a sink transmits interest (78 Bytes) to the concerned RHs and the RHs receive the same (78 Bytes). So, the communication overhead in this phase is $(78 \times n'_{RH}) + (78 \times n'_{RH}) = 156 \times n'_{RH}$ Bytes. In the response phase, the n'_{RH} number of RHs transmit data (4), hash digest (12) along with the interest (78) which is equal to $(n'_{RH} \times 94)$ Bytes. The sink receives $(n'_{RH} \times 94)$ Bytes. Thus, the total communication overhead in terms of the number of Bytes transmitted and received are $\{78 \times n'_{RH} + 94 \times n'_{RH}\}$ and $\{78 \times n'_{RH} + 94 \times n'_{RH}\}$ Bytes respectively.

Thus, the communication overhead in terms of energy (E_{comm}) is as follows:

$$E_{comm} = \left[\{78 \times n'_{RH} + 94 \times n'_{RH}\} \times 0.00189\right] + \left[\{78 \times n'_{RH} + 94 \times n'_{RH}\} \times 0.00167\right] mJ$$

3) COMPUTATION OVERHEAD

This section provides computation overheads for both the real-time and non-real-time queries. This overhead is also considered for handling one query. We provide here the worst-case analysis of computation overhead.

a: REAL-TIME QUERY

Interest Forwarding Phase: In this phase computation overhead is incurred by the sink and the RHs for the following tasks:

Sink: PIT search + Decision making by consulting FIB
RH: PIT search + Decision making by consulting FIB

TABLE 3. Required number of cycles for query processing [33], [35], [36] (a) Real-time (b) Non-real-time.

(a) Real-time

Algorithm		Algorithmic construct	No. of cycles
Interest forwarding	Sink	5×IF-E LSE, MOV, 3×SEARCH, CMP, 9×ADD	5×10 + 4 + 3×69 + 6 + 9×4 = 303
	RH	6×IF-ELSE, MOV, 2×SEARCH, CMP, 9×ADD	6×10 + 4 + 2×69 + 6 + 9×4 = 244
Data forwarding	Sensor	LOCHA	2952
	CH	SEARCH, (IF-ELSE + LOCHA), LOCHA	69 + n'_{Sensor} × (10+2952) + 2952 = 3021 + 2962× n'_{Sensor} n'_{Sensor}: Number of concerned Sensor nodes
	RH	SEARCH, (IF-ELSE + LOCHA), LOCHA	69 + n'_{CH} × (10+2952) + 2952 = 3021 + 2962 × n'_{CH} n'_{CH}: Number of concerned Cluster Head nodes
	Sink	SEARCH, (IF-ELSE + LOCHA)	69 + n'_{RH} × (10+2952) = 69 + 2962×n'_{RH} n'_{RH}: Number of concerned Region Head nodes

(b) Non-real-time

Algorithm		Algorithmic construct	No. of cycles
Interest forwarding	Sink	5×IF-ELSE, MOV, 3×SEARCH, CMP, 9× ADD	6×10 + 4 + 3 × 69 + 6 + 9 × 4 = 313
	RH	4×IF-ELSE, SEARCH	4×10 + 69= 109
Data forwarding	RH	SEARCH, LOCHA	69 + 2952 = 3021
	Sink	SEARCH, (IF-ELSE + LOCHA)	69 + n'_{RH} × (10+2952) = 69 + 2962×n'_{RH} n'_{RH}: Number of concerned Region Head nodes

Data Forwarding Phase: In this phase computation overhead is incurred by the sink RHs, CHs, and the sensor nodes for the following tasks:

Sensor nodes: Making digest using LOCHA

CH: Aggregation + Digest (check + Apply)

RH: Aggregation + Digest (check + Apply)

Sink: Aggregation + Digest (check)

According to the Tmote sky specification, the clock of the MSP430 microcontroller works at 8 MHz [35], [36]. Thus, time per clock cycle = $\frac{1}{8\times10^6}$ = 0.125 microsecond. So, the energy needed to run one cycle = 3 × 1.8 × 0.125 = $0.675nJ$ [36]. Thus, the computation overhead in terms of energy (E_{comp}) can be expressed as follows:

$$E_{comp} = \big[\{303 + (n'_{RH} \times 244)\} + \{(2952 \times n'_{Sensor}) + n'_{CH} \times (3021 + 2962 \times n'_{Sensor}) + n'_{RH} \times (3021 + 2962 \times n'_{CH}) + (69 + 2962 \times n'_{RH})\}\big] \times 0.675nJ$$

TABLE 3(a) provides the required computation cycles of the algorithmic constructs for a real-time query as per the instruction set of Tmote sky [36], [37].

b: NON-REAL-TIME QUERY

Interest Forwarding Phase: In the worst case, the computation overhead is incurred by the sink and the RHs for the following tasks:

Sink: PIT search + Decision making consulting FIB

RH: PIT search

Data Forwarding Phase: In this phase computation overhead is incurred by the sink and RHs for the following tasks:

RH: CS search + Digest (Apply)

Sink: Digest (check)

The computation overhead in terms of energy (E_{comp}) can be expressed as follows:

$$E_{comp} = \big[\{313 + (n'_{RH} \times 109)\} + \{(n'_{RH} \times 3021) + (69 + 2962 \times n'_{RH})\}\big] \times 0.675nJ$$

TABLE 3(b) provides the details of computation cycles of the algorithmic constructs for a non-real-time query.

B. QUANTITATIVE ANALYSIS

This section provides the performance evaluation of the proposed IPLQueeN through simulation.

1) SIMULATION ENVIRONMENT

Simulation is carried out in Cooja, the Contiki [37] network Simulator. The Cooja simulator supports the simulation of networks consisting of wireless sensor nodes. We use sky mote for our simulation. We simulate the network with the number of nodes varying from 100 to 500 deployed across a 200m × 200m square grid. We take average results of 20 independent runs while plotting the simulation graphs.

TABLE 4. Simulation parameters [26].

Environment/Parameter		Specification/Value
Radio Medium		Unit disk graph with distance loss
Sink positioning		Centre
Mote start-up delay		1000ms
Transmission range		50m
Current consumption	Radio transmitting	19.5mA
	Radio receiving	21.8mA
	MCU on, Radio off	1800μA
	MCU idle, Radio off	54.5 μA
Initial energy of a node		1000 J
Query arrival rate		1/minute

The other important simulation parameter values are provided in TABLE 4.

2) SIMULATION METRICS

Performance evaluation of the proposed scheme is divided into two stages: measuring the extent of achieving major design goals and evaluation of network performance. By design metrics, we mean energy consumption, query processing delay, and the number of the packet flow to process a query. On the other hand, network performance metrics are packet loss rate and network lifetime.

a: DESIGN METRICS

Energy Consumption: We define energy by measuring the total energy consumption of the network to process a query starting from the interest to the response phase. During the simulation, we measure the energy using the Power trace application of the Cooja simulator [37]. The total energy consumption can be expressed as,

$$Energy\ consumption = \sum_{i=1}^{N_p} \times \frac{Energest\ value \times current \times voltage}{RTIMER\_SECOND}$$

Energest_Value is the difference between the numbers of ticks in two-time intervals (from sending interest to receive a response) which is found from the Power trace after simulation. At each i^{th} node, we find the different Energest value for CPU, LPM, TX, and RX where $i = 1$ to N_P, N_P is the number of participating nodes in interest forwarding and data forwarding phases. The value of *current* is found from Sky mote datasheet which is also different for CPU, LPM, TX, and RX. The *voltage* for Sky mote is 3V and *RTIMER_SECOND* is 32768.

Delay: We consider the delay in query processing by measuring the total time duration from generating a query to receive the response by the generating end (sink). Let t_s be the time a query (interest) packet transmitted from sink and t_r be the time the corresponding data packet is received at the sink [32]. So, the delay can be expressed as,

$$Delay = (t_r - t_s)$$

Message Flow: We define message flow as the total number of message exchanges to process a query starting from the generation of interest to reception of the data.

b: NETWORK PERFORMANCE METRICS

Packet Loss Rate (PLR): It is the fraction of packets that are transmitted within a time window, but not received [38]. Packet loss minimizes the Packet Delivery Ratio. The packet loss (%) can be expressed as,

$$PLR = (100 \times \frac{No.\ of\ packets\ lost}{No.\ of\ packets\ sent})\%$$

Network Lifetime: The lifetime of a network is often defined as the maximum time a certain task can be carried out without any node running out of energy. It can be derived from the initial energy and average consumed energy at each node, and it specifies the time a node can serve before draining out. The long lifetime of a node indicates a long network lifetime. It is calculated and expressed as follows assuming one query at each node/minute is served in the network [32], [39].

$$Network\ Lifetime = (\frac{Initial\ Energy\ at\ each\ node}{Energy\ consumption\ per\ node/minute})$$

End-to-End Delay (EED): This metric indicates the average time duration over all the packets that are transmitted from the source to the destination. This value includes all possible delays caused by buffering, queuing, retransmissions, propagation, and transfer through a channel. [32].

$$Average\ EED = \frac{1}{p}\sum_{j=1}^{p} Delay(j)$$

where p is the total number of packets and $Delay(j)$ denotes the total transmission delays of a packet.

Throughput: This metric is defined as the total number of bits successfully received at the server within a definite time duration. The throughput at the receiver can be calculated as follows:

$$Throughput = \frac{Total\ Bits\ Received}{t - t_f}$$

where t_f is the time of the first packet received and the t represents either the time of the last packet received if the session is complete or the simulation time if the session is incomplete [32].

3) RESULTS AND DISCUSSION

We conduct experiments to evaluate the comparative performance of IPLQueeN considering both real-time and non-real-time queries. We compare the performance of real-time

TABLE 5. Performance results with non-real-time queries.

Packet Size (Bytes)	No. of nodes	Energy consumption (mJ)	Delay (Sec)	Message Flow	Packet Loss Rate (%)	Network Lifetime (Days)	End to End Delay (mSec)	Throughput (Bytes/Sec)
90	100	9.23	0.23	2	0.00	138	116	30.22
	200	9.70	0.23	2	1.28	137	116	31.54
	300	10.00	0.24	4	1.55	204	117	32.60
	400	10.10	0.24	4	2.00	181	118	35.11
	500	10.20	0.24	4	2.50	227	117	35.73
100	100	10.10	0.24	4	0.90	121	117	32.76
	200	10.12	0.24	4	1.51	116	117	34.33
	300	10.20	0.25	4	2.32	171	118	34.81
	400	11.50	0.26	6	2.53	159	120	36.30
	500	12.20	0.26	4	2.75	201	121	38.50
110	100	11.50	0.25	2	1.12	116	118	35.46
	200	12.00	0.26	4	1.50	115	121	36.50
	300	12.20	0.27	4	2.80	171	125	38.22
	400	13.10	0.27	6	2.50	158	129	40.30
	500	13.80	0.28	4	2.85	198	132	42.65
120	100	12.00	0.27	4	1.34	114	118	38.75
	200	12.10	0.28	4	2.10	115	121	40.23
	300	12.20	0.28	4	2.62	171	124	41.50
	400	13.20	0.30	4	2.78	151	131	43.78
	500	14.00	0.31	6	3.06	190	131	44.32

queries with three state-of-the-art competitor schemes such as Q-LEACH (Query based low-energy adaptive clustering hierarchy) [3], pCASTING [23], and BSS (broadcast suppression scheme) [27]. The Q-LEACH is a query processing scheme over LEACH [40] based WSN whereas the pCASTING is a caching-based query processing scheme over NDN-IoT. The BSS is an interest broadcast suppression scheme that avoids broadcasting unnecessary copies of interest by forwarding the interest through potential forwarders only. The Q-LEACH, pCASTING, and BSS are developed as follows.

In Q-LEACH whenever a query comes to the sink, it broadcasts the query towards the cluster heads. Now, each of the cluster heads, in turn, broadcasts the query towards their member (sensor) nodes. Next, in the Data forwarding phase, the sensor nodes apply the lightweight LOCHA on sensory data and send responses to their respective cluster heads. The cluster heads then verify the digest. Upon successful verification, the cluster heads aggregate the data received from the members against the query and send the aggregated data to the sink. So, during the experiment, we make two modifications in the Q-LEACH. One such modification is, instead of the competitor's integrity scheme, LOCHA is used to make the securing mechanism more energy efficient. Another modification is instead of using TDMA, here CSMA/CA is used to support scalability and reduced delay in the query response.

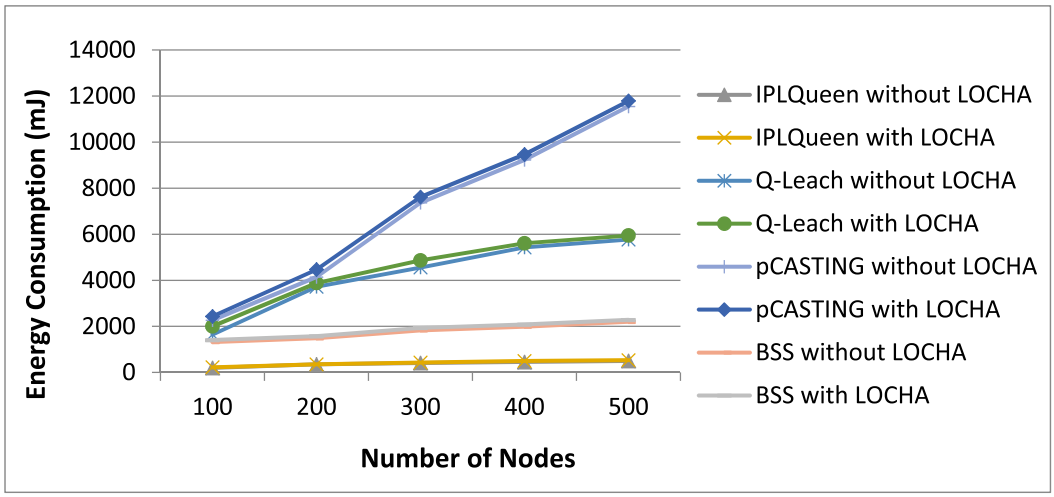
In pCASTING, 15% of total nodes in the network act as Consumers, one node acts as Access Point (AP), and the remaining nodes act as Forwarder Nodes (FNs). Here, a Consumer periodically broadcasts the generated query towards all the FNs. While receiving the query, an FN searches the data in CS, and if not found it checks in PIT. If a matching entry is found, the query is discarded; otherwise, a new PIT entry is created. Then the query is broadcast towards all the FNs. If any of the FN has the data is CS, it is forwarded back to the intended Consumer. If the query is not found in any of the FNs, it is retrieved from the AP. With each query (interest), a freshness value is also attached. If any data is found in FN which exceeds the specified freshness, it is not retrieved. Instead, the fresh data is retrieved from the AP. However, during the experiment, we make two modifications in the pCASTING. Firstly, to maintain message integrity during data transmission, the lightweight hash function LOCHA is used while data is forwarded back from any FN or AP. Secondly, the nodes are considered static to make the pCASTING comparable with the proposed scheme.

In BSS, an interest broadcast suppression scheme is proposed. Here, a consumer node (sink) first broadcasts the interest towards the neighbor nodes. But only the potential forwarder nodes forward the interest. The nodes are selected as potential forwarders based on the holding time for each unique interest containing specific content. This holding time for a potential node is less than the other nodes in the transmission range of the consumer or the previous forwarder nodes. The rest of the activities in the interest phase are taken from the notion of NDN. Forwarding data message also follows the similar set of rules as interest but uses different holding time. In the data message, however, to make the scheme comparable with us, we have used hash-based (LOCHA) verification instead of a signature.

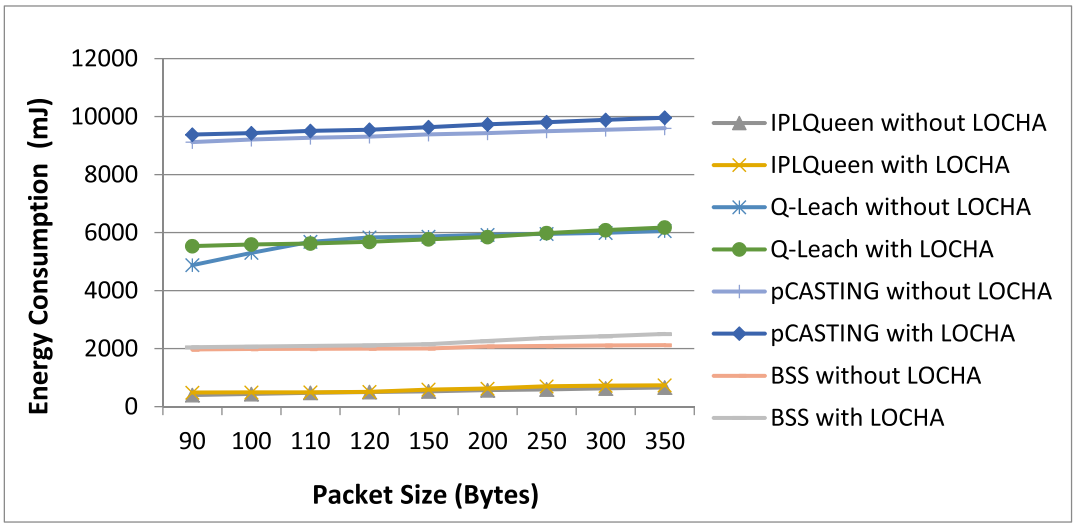
a: RESULTS WITH REAL-TIME QUERIES

We evaluate simulation results in terms of both the design metrics and network performance metrics for real-time queries.

Achieving Design Goal: Three sets of experiments are conducted for evaluating the performance of the present scheme by measuring the extent of attaining the design objective in



(a) Varying number of nodes



(a) Varying packet size

FIGURE 5. Energy consumption.

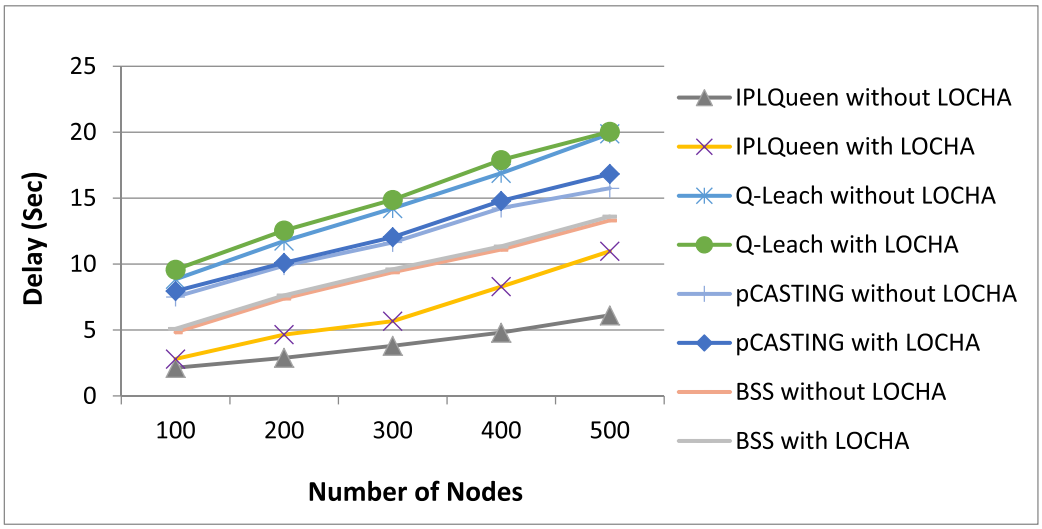
terms of energy consumption, query processing delay, and the number of the packet flow.

Energy Consumption: In the first set of experiments (Figure 5) we measure energy consumption for processing a query for all the competing schemes including IPLQueeN both with-LOCHA and without-LOCHA. Figure 5(a) plots the energy consumption results with a varying number of nodes. We primarily observe from the plot that, unlike all the competing schemes, in our scheme, IPLQueeN energy consumption remains the same for varying numbers of nodes. We also observe that the energy consumption for IPLQueeN is the lowest among all the competing schemes Q-LEACH, pCASTING, and BSS for both the with-LOCHA and without-LOCHA versions. Precisely, it is on average 91% and 93% and 78% lower as compared to Q-LEACH, pCASTING, and BSS, respectively.

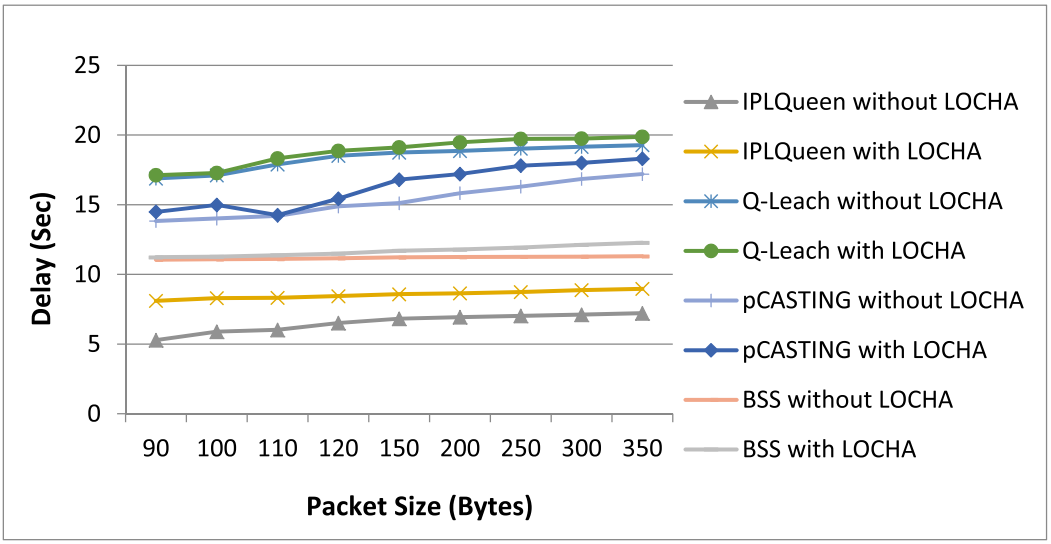
Figure 5(b) plots the energy consumption results with varying packet sizes. Here also we observe that, unlike all the competing schemes, in our scheme, IPLQueeN energy consumption remains the same for varying packet size.

We also observe that the energy consumption for IPLQueeN is the lowest among all the competing schemes for both the with-LOCHA and without-LOCHA versions. To be more specific, on average, the energy consumption of IPLQueeN is 90%, 94%, and 74% less than Q-LEACH, pCASTING, and BSS, respectively.

The reason for both sets of the above results is explained as follows. The IPLQueeN is based on multilevel hierarchical architecture resulting in a wide coverage of the application area and has a mechanism to identify the sub-area for query answering. So, instead of involving all the nodes in answering the query, a part of the hierarchy having a subset of the nodes participate in query answering. On the contrary, in Q-LEACH, pCASTING, and BSS, all the nodes participate in answering the query. So, the energy consumption increases at a much higher rate for all the schemes with varying



(a) Varying number of nodes



(b) Varying packet size

FIGURE 6. Delay in processing a query

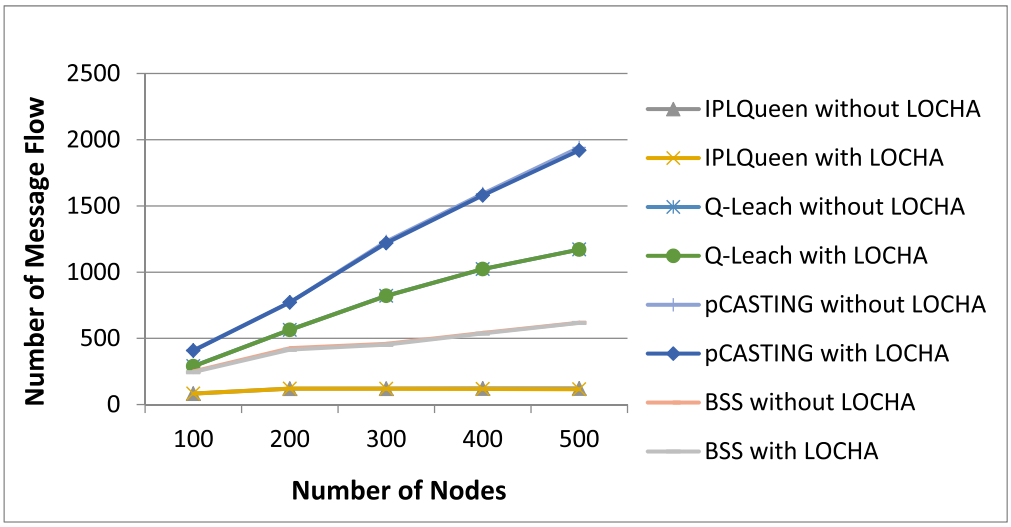
FIGURE 6. Delay in processing a query.

numbers of nodes (Figure 5(a)) and varying packet sizes (Figure 5(b)).

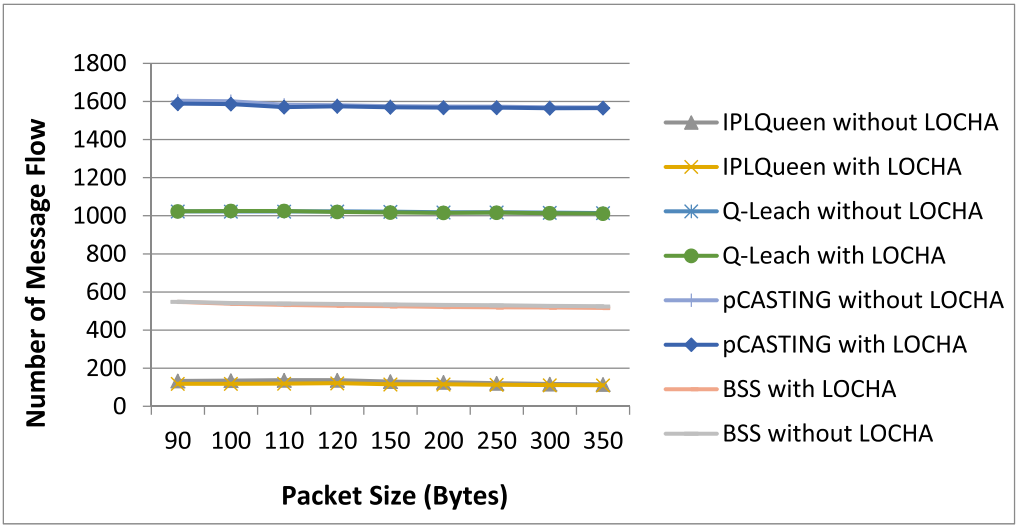
We further observe that in all the schemes including ours for both sets of experiments, the plots for with-LOCHA and without-LOCHA overlap. It implies that LOCHA being a lightweight digest incurs very little overhead in terms of energy.

Delay: In the second set of experiments (Figure 6) we measure query processing delay for all the competing schemes including ours for both with-LOCHA and without-LOCHA. Figure 6(a) plots the results with the varying number of nodes. We generally observe that delay increases with a varying number of nodes. However, it is the lowest in IPLQueeN among all the schemes. Precisely, it is 73% and 58% less than Q-LEACH for without-LOCHA and with-LOCHA, respectively. These values are 67%, 50% and 57%, 34% for pCASTING and BSS, respectively.

Figure 6(b) plots the results with varying packet sizes. We generally observe that delay for all the schemes does not vary much with varying packet sizes. However, it is the lowest for IPLQueeN both for with-LOCHA and without-LOCHA. Precisely, it is 54%, 64% less than Q-LEACH for with-LOCHA and without-LOCHA, respectively. These values are 47%, 57% and 26%, 42% less than pCASTING and BSS. Similar to the energy plot, the reason for both sets of the above results on delay is as follows. Unlike Q-LEACH, pCASTING, and BSS, instead of involving all the nodes in answering the query, a part of the hierarchy having a subset of the nodes participate in IPLQueeN. It results in a lowering of delay.



(a) Varying number of nodes



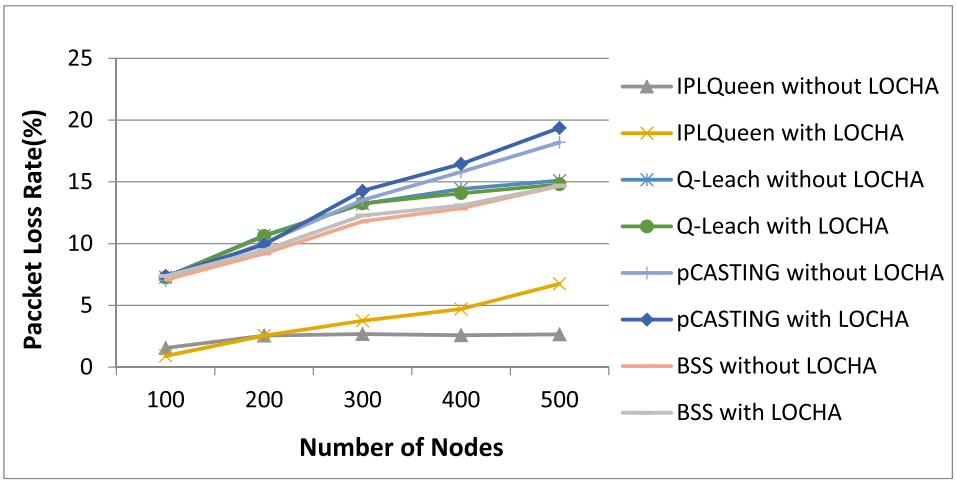
(b) Varying packet size

FIGURE 7. Message flow.

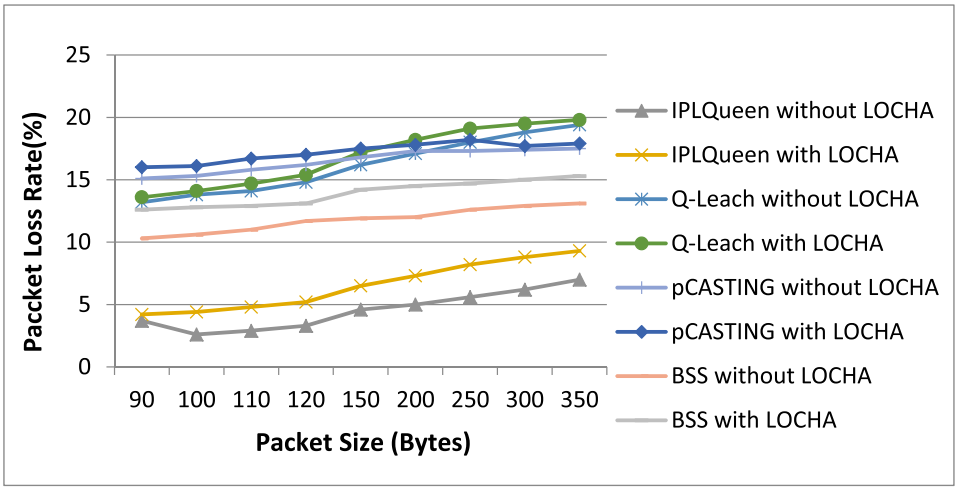
We further observe that in all the schemes including ours for both sets of experiments, there is a gap between the plots for with-LOCHA and without-LOCHA. It implies that although LOCHA is a lightweight digest, it incurs marginally higher delay in the with-LOCHA version than the without-LOCHA version.

Message Flow: In the third set of experiments (Figure 7) we measure message flow for all the competing schemes including ours for both with-LOCHA and without-LOCHA. Figure 7(a) plots the results with the varying number of nodes. We primarily observe that number of message flows increases with the number of nodes for all the competing schemes except our scheme IPLQueeN. In our scheme, it remains almost flat over a varying number of nodes. The reason is while receiving an interest, despite broadcasting it in the network, the intended receivers of the interest are found from the FIB. We also observe that the number of such flows is the lowest in our scheme for any number of nodes. Precisely, on average, the number of message flow in IPLQueeN is 83%, 88%, and 74% lower than Q-LEACH, pCASTING, and BSS, respectively. This is possible as our scheme is low-overhead compared to the others.

Figure 7(b) plots the results with varying packet sizes. We generally observe that the message flow for all the schemes does not vary much with varying packet sizes. As the number of message flow for processing a query does depend on the number of message exchange among the participating nodes, it intrinsically does not depend on packet size. However, it is the lowest for IPLQueeN both for with-LOCHA and without-LOCHA. Finally, we observe that the plots for with-LOCHA and without-LOCHA overlap for both the plots (Figure 7(a), 7(b)) which imply that the overhead of LOCHA does not have an impact on message flow either with the varying number of nodes or with varying packet size.



(a) Varying number of nodes



(b) Varying packet size

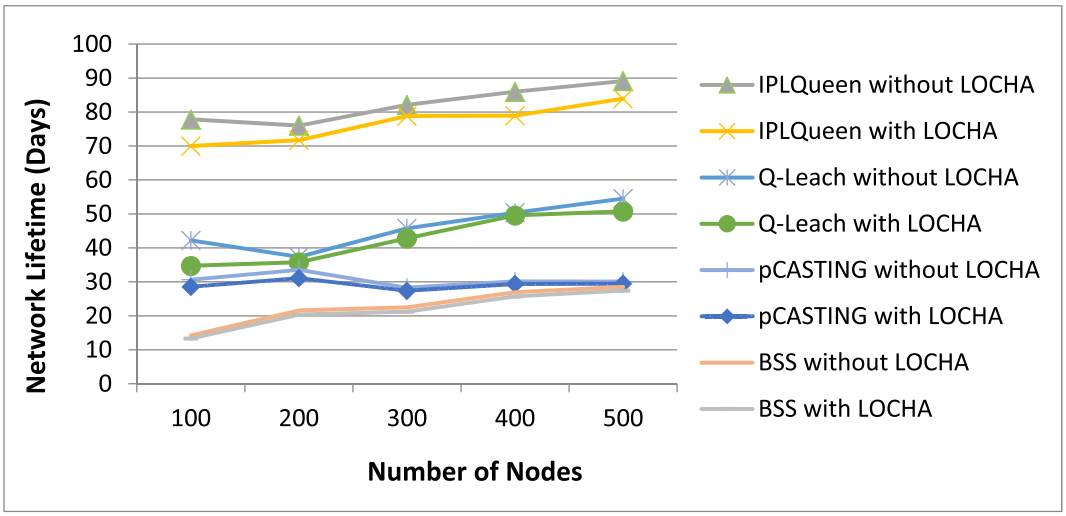
FIGURE 8. Packet loss rate.

Evaluation of Network Performance: Four sets of experiments in terms of packet loss rate, network lifetime, end-to-end delay, and throughput are conducted to observe the network performance of the present scheme while achieving the design goal.

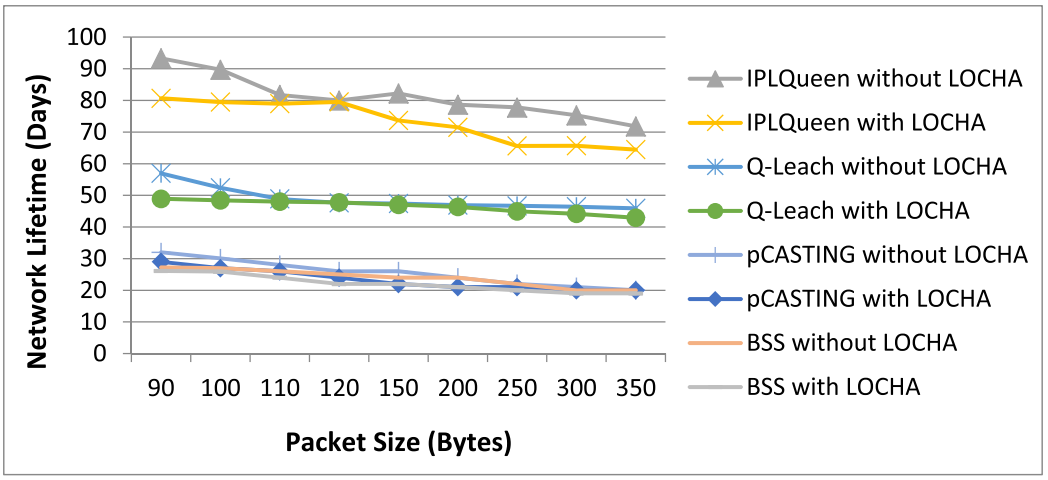
Packet Loss Rate: In the first set of experiments (Figure 8) we measure packet loss rate for all the competing schemes including ours for both with-LOCHA and without-LOCHA. Figure 8(a) plots the results with the varying number of nodes. We generally observe from Figure 8(a) that such a rate increases with the number of nodes for all the competing schemes including ours. However, the rate in IPLQueeN is significantly less compared to Q-LEACH, pCASTING, and BSS. We also observe that irrespective of the number of nodes the packet loss is the least in IPLQueeN. To be more specific, on average, it is 80% and 71% less in IPLQueeN compared to Q-LEACH without-LOCHA and with-LOCHA, respectively. These values are 80%, 74% and 78%, 70% compared to pCASTING and BSS, respectively.

Figure 8(b) plots the results with varying packet sizes. We observe from the plot that the packet loss rate does not vary much with varying packet size for all the competing schemes including IPLQueeN. However, such a rate is the least in IPLQueeN compared to all the competing schemes for all the packet size. Precisely, on average, it is 86% and 67% less in IPLQueeN compared to Q-LEACH without-LOCHA and with-LOCHA, respectively. These values are 83%, 71% and 68%, 52% compared to pCASTING and BSS, respectively.

We finally observe that in all the schemes for both sets of experiments, there is a gap between the plots for with-LOCHA and without-LOCHA. This gap is marginal in the plot with varying numbers of nodes; however, this is little more in the plot with varying packet size. It implies that although LOCHA is a lightweight digest, it incurs a marginally higher packet loss rate with varying packet size in the with-LOCHA version than the without-LOCHA version.



(a) Varying number of nodes



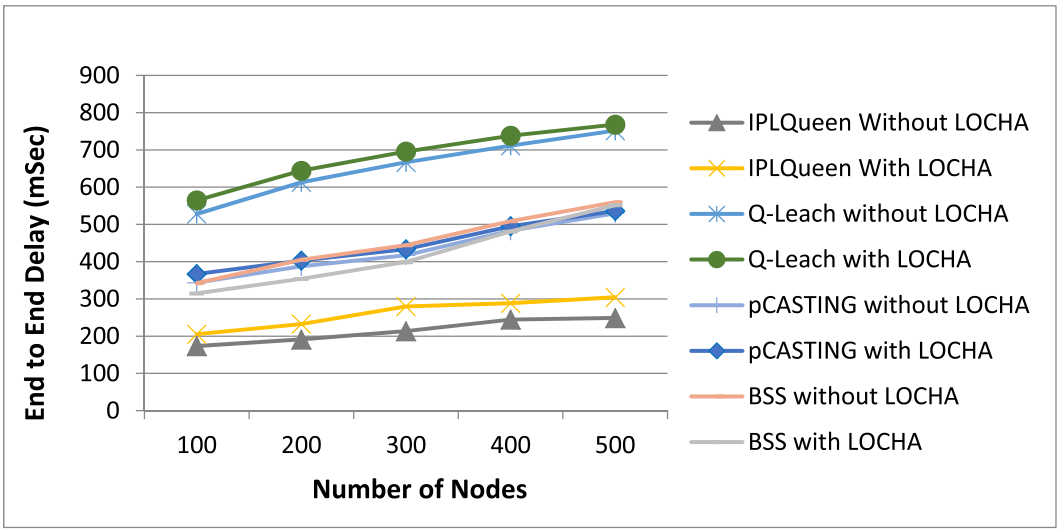
(b) Varying packet size

FIGURE 9. Network lifetime.

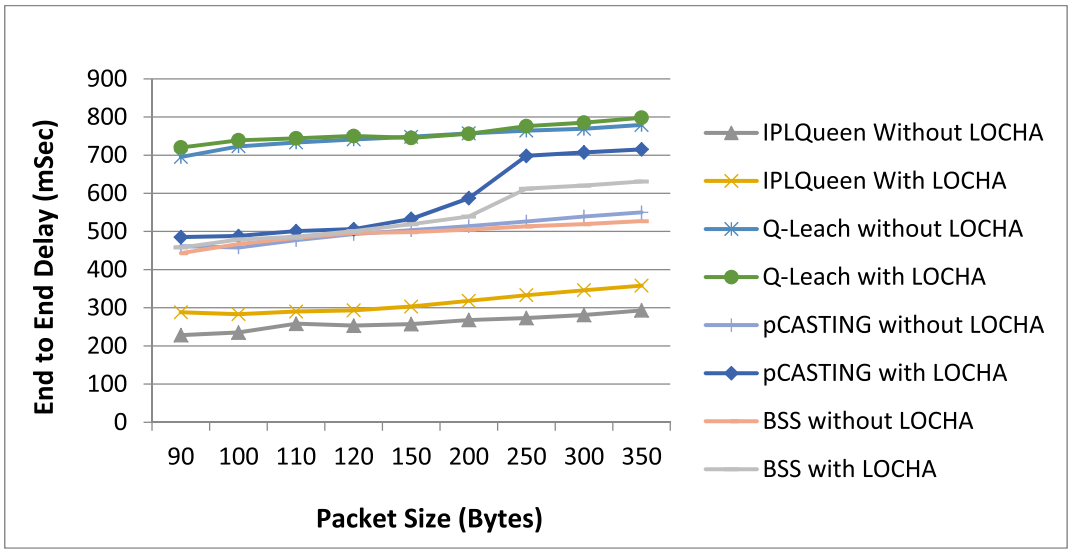
Network Lifetime: In the second set of experiments (Figure 9) under network performance evaluation, we measure the lifetime for all the competing schemes including ours for both with-LOCHA and without-LOCHA. Figure 9(a) plots the results with the varying number of nodes. We generally observe from Figure 9(a) that, as expected, the network lifetime increases, with the number of nodes for all the competing schemes including ours. Though the rate of increase is not very high, the lifetime remains the highest in IPLQueeN compared to others. To be more specific, on average, the lifetime in IPLQueeN is 45%, 62%, and 72% higher than the Q-LEACH, pCASTING, and BSS, respectively. As the number of nodes increases, participating nodes in processing a query in other competing schemes increases at a much higher rate compared to the IPLQueeN. Thus, the energy consumption of the network increases at a much higher rate in the competing schemes compared to IPLQueeN.

Figure 9(b) plots the results with varying packet sizes. We generally observe that network lifetime reduces with an increase in packet size. As packet size increases, the communication overhead of a node increases. This increases the probability of the node getting drained and as a result, the lifetime reduces. Although the rate of lifetime fall is marginally higher in IPLQueeN compared to the others, it remains the longest for all the packet sizes. In IPLQueeN, it is, on average, 37%, 68%, and 70% higher than the Q-LEACH, pCASTING, and BSS, respectively.

End to End Delay: In the third set of experiments (Figure 10), we measure end-to-end delay for all the competing schemes including ours both for with-LOCHA and without-LOCHA. Figure 10(a) plots the results with the varying number of nodes. We generally observe that end-to-end delay increases with the number of nodes for all the schemes. The reason is as the number of nodes increases it involves a greater number of nodes in responding a query, which incurs



(a) Varying number of nodes



(b) Varying packet size

FIGURE 10. End to end delay.

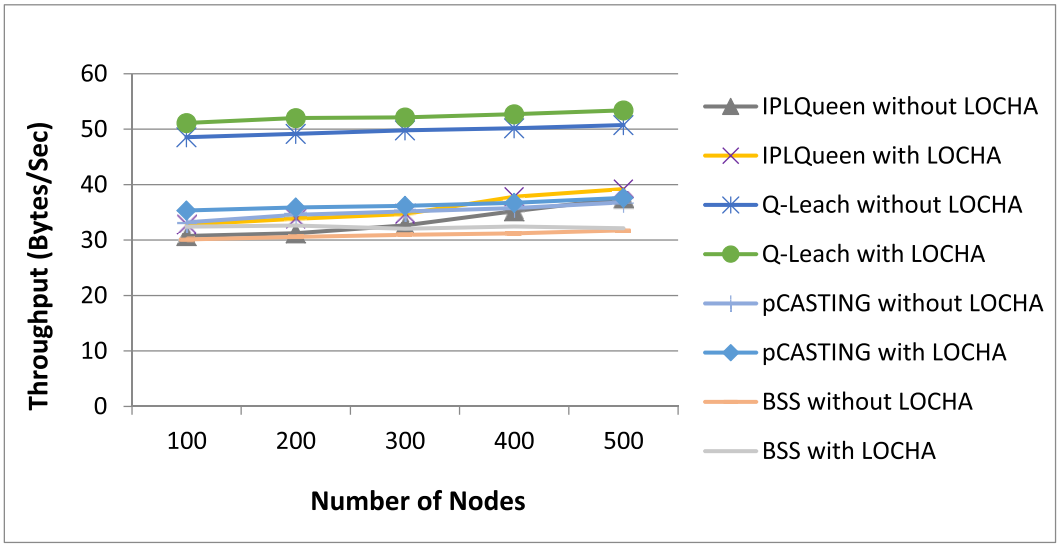
longer such delay. However, end-to-end delay is the lowest in IPLQueeN both for with-LOCHA and without-LOCHA version. Precisely, it is 67% and 62% lower compared to Q-LEACH without-LOCHA and with-LOCHA, respectively. These values are 50%, 41% and 52%, 36% compared to pCASTING and BSS, respectively.

Figure 10(b) plots the results with varying packet sizes. We observe from the plot that similar to the results with the varying number of nodes, end-to-end delay for all the schemes increases with an increase in packet size. However, such delay is the lowest in IPLQueeN. To be more specific, it is 57% and 65% less as compared to Q-LEACH for both with-LOCHA and without-LOCHA versions. These values are 45%, 48%, and 42%, 47% as compared to pCASTING and BSS, respectively.

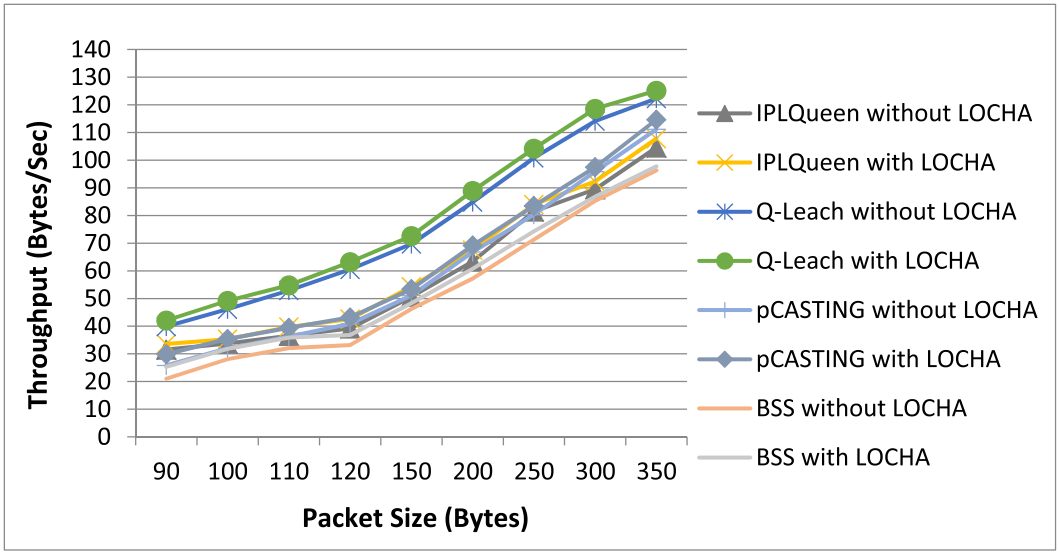
Throughput: In the fourth set of experiments (Figure 11), we measure throughputs for all the competing schemes including ours for both with-LOCHA and without-LOCHA. Figure 11(a) plots the results with the varying number of nodes. We generally observe from the plot that the throughput marginally increases with an increase in the number of nodes for all the schemes including ours. We also observe that the throughput in our scheme IPLQueeN is either at par or marginally higher compared to all the competing schemes except Q-LEACH. increases with packet size for all the schemes. Moreover, the results of all the schemes including ours almost overlap except Q-LEACH. In Q-LEACH it is marginally higher than all the schemes.

Figure 11(b) plots the results with varying packet sizes. We observe from the plot that as expected, throughput sharply.

Thus, from the above discussion on the performance results we claim that the IPLQueeN achieves all the design goals such as the least energy requirement, query processing delay



(a) Varying number of nodes



(b) Varying packet size

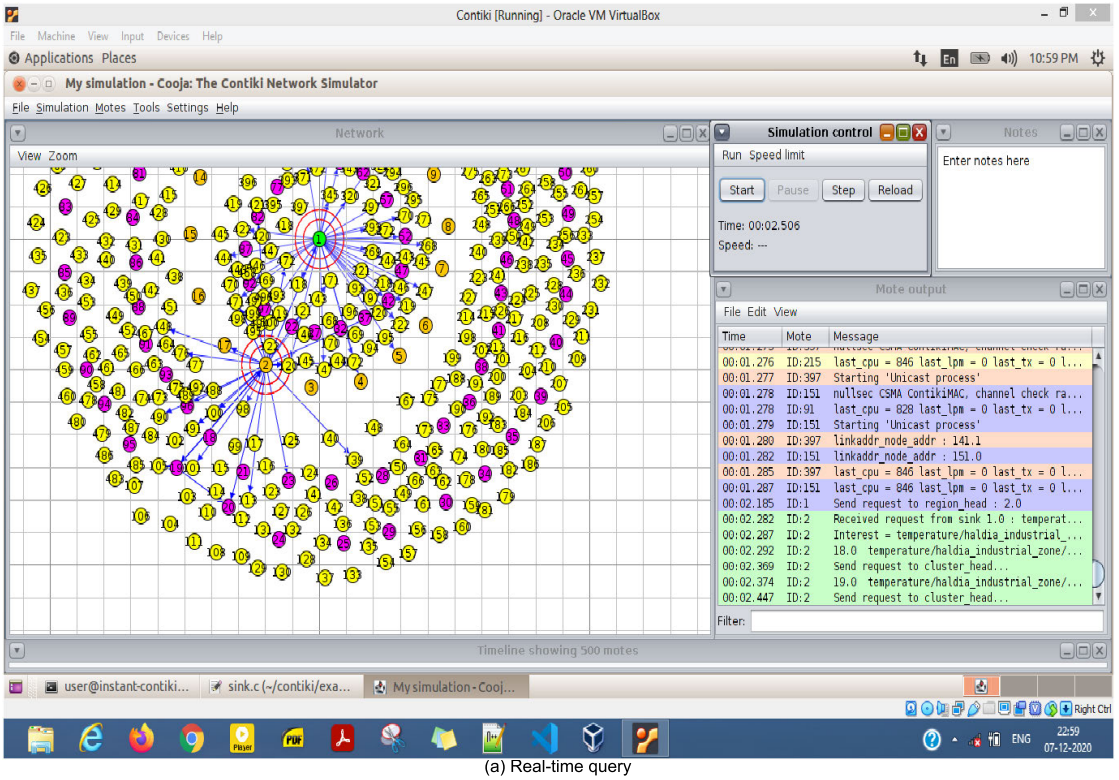
FIGURE 11. Throughput.

while maintaining the best network performance in terms of network lifetime, end-to-end delay at the cost of either at par or little less performance in throughput.

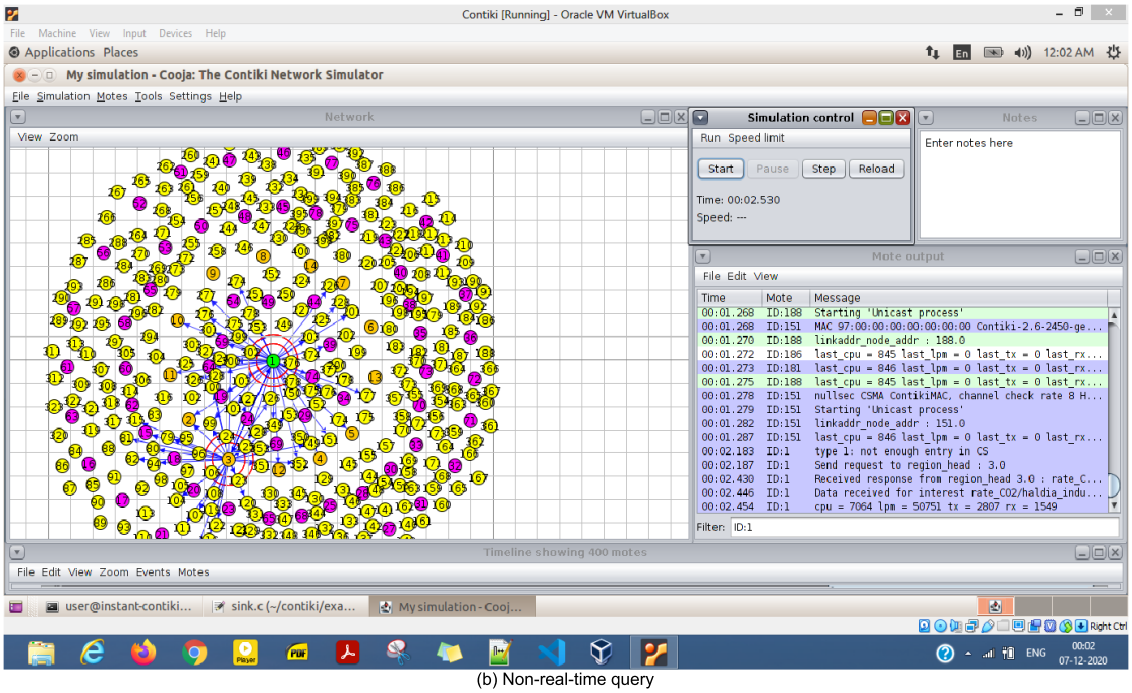
b: RESULTS WITH NON-REAL-TIME QUERIES

We evaluate simulation results in terms of both the design goals and network performance for non-real-time queries. TABLE 5 provides the comprehensive performance results of the with-LOCHA version of our scheme. We observe from the results that all the values of design metrics such as energy, delay, and packet flow are significantly less compared to the results of real-time queries. For example, here energy varies between 9.23-14 mJ whereas it is 196-530 mJ (Figure 5) for real-time queries. Similarly, query processing delay varies between 0.23-0.31 Sec and 2-11 Sec respectively in non-real-time and real-time queries (Figure 6).

We also observe that the end-to-end delay for non-real-time queries varies between 116-132 msec and the same for real-time queries varies between 173-304 mSec (Figure 10). These values for network lifetime are 137-190 days for non-real-time queries and 70-90 days for real-time queries. We further observe that the values of other network performance metrics such as packet loss rate and throughput do not vary much with the varying number of nodes and packet size. Precisely, in non-real-time queries, the packet loss rate, and throughput vary between 0-3% and 30-44 Bytes/Sec, respectively. For real-time queries, these values are 2-7% and 30-40 Bytes/Sec, respectively. The reason for such results is that, unlike real-time queries, the non-real-time queries are mostly responded from cache (CS) of the sink, RHs and the queries do not penetrate up to the bottom level of the hierarchy of the network.



(a) Real-time query



(b) Non-real-time query

FIGURE 12. **Snapshots of simulation at different instances for different types of query.**

The snapshots of the simulation for both the real-time and non-real-time queries in Cooja are shown in Figure 12.

4) COMPREHENSIVE COMPARISON

Finally, we provide a comprehensive overhead comparison, including theoretical and simulation results of IPLQueeN in TABLE 6. We compare both computation and communication overheads and resulting energy overheads. We also assess the approximate storage requirement theoretically by considering the required data structure for implementing the scheme. We take the results by varying N from 100 to 500 considering the number of entries $n = 20$

TABLE 6. Comprehensive overhead comparison.

No. of nodes	Computation overhead (mJ)		Communication (Tx+Rx) overhead (mJ)		Total Energy (mJ)		Storage overheads (Theoretical) KiloBytes
	Theoretical	Practical	Theoretical	Practical	Theoretical	Practical	
100	0.38	46.72	16.94	302.04	17.32	348.76	34.48
200	0.46	55.63	19.73	358.91	20.19	414.54	38.44
300	0.46	60.68	19.73	386.24	20.19	446.92	38.44
400	0.78	78.7	25.89	466.2	26.67	544.9	48
500	0.96	86.63	28.98	540.18	29.94	626.81	52.78

for sink and $n = 10$ for other nodes. The other parameter values are considered as $n'_{Sink} = 1$, $n'_{RH} = 2$, $n'_{CH} = 6 - 9$, $n'_{Sensor} = 24 - 25$.

All the theoretical values presented here are computed as per the derivations provided in Section V.A. We observe, as expected, theoretical and experimental results differ in all the cases. Possible reasons for less energy consumption in the theoretical analysis compared to practical are as follows:

- We have not considered quantization and channel coding phases in the theoretical analysis whereas during the simulation the real-time issues in Cooja are considered which are inbuilt.
- During the simulation, each node may not require the same energy to transmit or receive the interest/data packets as it depends on the distance a packet must travel. But, in the theoretical analysis of communication overhead, each node whether it is a sink, or any other normal sensor nodes have been considered to require equal energy while transmit/receive a packet.

VI. CONCLUSION

In this article a low overhead, integrity preserving query handling scheme for WSN is proposed which follows the notion of data-centric network NDN. It is developed for query-driven applications in WSN, and the communication takes place in the network through the exchange of interest and data packets. The scheme is made low overhead by judicious use of the NDN data structure CS, PIT, FIB in some of the nodes which are strategically placed in the multilevel hierarchical architecture of the network. The scheme ensures data integrity by applying a low-overhead hash LOCHA on each data transmitted in the response phase corresponding to an interest. Finally, the entire query processing scheme is analyzed theoretically and establishes its feasibility to apply on an energy-starved network like WSN. It is also evaluated through simulation and the results prove that our scheme achieves the design goals such as the least energy requirement, least query processing delay over all the competitors while maintaining the best network performance in terms of network lifetime, end-to-end delay at the cost of either at par or little less performance in throughput. Also, the energy-saving performance of the scheme establishes its potential to readily implement the same in real-life motes. As a future scope, the scheme may be appropriately modified to work for both the hierarchical and flat architecture in WSN applicable for any query-driven real-life applications.

REFERENCES

[1] B. S. Kim, H. Park, K. H. Kim, D. Godfrey, and K. I. Kim, "A survey on real-time communications in wireless sensor networks," *Wireless Commun. Mobile Comput.*, vol. 2017, Oct. 2017, Art. no. 1864847.
[2] J. Gehrke and S. Madden, "Query processing in sensor networks," *IEEE Pervasive Comput.*, vol. 3, no. 1, pp. 46–55, Jan./Mar. 2004.
[3] A. Ghosal, S. Halder, and S. DasBit, "A dynamic TDMA based scheme for securing query processing in WSN," *Wireless Netw.*, vol. 18, no. 2, pp. 165–184, 2012.
[4] R. Sinde, F. Begum, K. Njau, and S. Kaijage, "Refining network lifetime of wireless sensor network using energy-efficient clustering and DRL-based sleep scheduling," *Sensors*, vol. 20, no. 5, p. 1540, 2020.
[5] X. Liu, X. Zhang, J. Yu, and C. Fu, "Query privacy preserving for data aggregation in wireless sensor networks," *Wireless Commun. Mobile Comput.*, vol. 2020, Feb. 2020, Art. no. 9754973.
[6] T. A. Alghamdi, "Energy efficient protocol in wireless sensor network: Optimized cluster head selection model," *Telecommun. Syst.*, vol. 74, no. 3, pp. 331–345, Jul. 2020.
[7] M. Omar, S. Yahiaoui, and A. Bouabdallah, "Reliable and energy aware query-driven routing protocol for wireless sensor networks," *Ann. Telecommun.*, vol. 71, nos. 1–2, pp. 73–85, Feb. 2016.
[8] A. Belfkih, C. Duvallet, L. Amanton, and B. Sadeg, "A new query processing model for maintaining data temporal consistency in wireless sensor networks," in *Proc. IEEE 10th Int. Conf. Intell. Sensors, Sensor Netw. Inf. Process. (ISSNIP)*, Apr. 2015, pp. 1–6.
[9] N. Y. S. Al-Hoqani, "In-network database query processing for wireless sensor networks," M.S. thesis, Loughborough Univ., Loughborough, U.K., 2018. [Online]. Available: https://hdl.handle.net/2134/36226
[10] Z. Zhu and A. Afanasyev, "Let's chronoSync: Decentralized dataset state synchronization in named data networking," in *Proc. 21st IEEE Int. Conf. Netw. Protocols (ICNP)*, Goettingen, Germany. Feb. 2013, pp. 1–10.
[11] Z. Li, Y. Xu, B. Zhang, L. Yan, and K. Liu, "Packet forwarding in named data networking requirements and survey of solutions," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 2, pp. 1950–1987, 2nd Quart., 2019.
[12] K. Tetsuro and S. Tetsuya, "A study on implementation of NDN to WSN," in *Proc. IEEE 31st Int. Conf. Adv. Inf. Netw. Appl. (AINA)*, Taipei, Taiwan, Mar. 2017, pp. 392–398.
[13] T. Chatterjee, S. Ruj, and S. D. Bit, "Security issues in named data networks," *Computer*, vol. 51, no. 1, pp. 66–75, Jan. 2018.
[14] A. Tariq, R. A. Rehman, and B.-S. Kim, "Forwarding strategies in NDN-based wireless networks: A survey," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 1, pp. 68–95, 1st Quart., 2020.
[15] C. A. Roy, T. Chatterjee, and S. DasBit, "LOCHA: A light-weight one-way cryptographic hash algorithm for wireless sensor network," in *Proc. Int. Conf. Ambient Syst., Netw. Technol. (ANTS)*. Amsterdam, The Netherlands: Elsevier, 2014, pp. 1–8.
[16] C. Ghasemi, H. Yousefi, K. G. Shin, and B. Zhang, "On the granularity of trie-based data structures for name lookups and updates," *IEEE/ACM Trans. Netw.*, vol. 27, no. 2, pp. 777–789, Apr. 2019.
[17] T. Song, H. Yuan, P. Crowley, and B. Zhang, "Scalable name-based packet forwarding: From millions to billions," in *Proc. 2nd ACM Conf. Inf.-Centric Netw.*, Sep. 2015, pp. 19–28.
[18] H. Yuan, P. Crowley, and T. Song, "Enhancing scalable name-based forwarding," in *Proc. ACM/IEEE Symp. Archit. Netw. Commun. Syst. (ANCS)*, Beijing, China, May 2017, pp. 60–69.

[19] T. Chatterjee, S. Ruj, and S. DasBit, "Data forwarding and update propagation in grid network for NDN: A low-overhead approach," in *Proc. IEEE Int. Conf. Adv. Netw. Telecommun. Syst. (ANTS)*, Indore, India, Dec. 2018, pp. 1–6.
[20] T. Chatterjee, S. Ruj, and S. DasBit, "LowSHeP: Low-overhead forwarding and update solution in NDN with hexadecimal patricia trie," in *Proc. IEEE Int. Conf. Adv. Netw. Telecommun. Syst. (ANTS)*, Goa, India, Dec. 2019, pp. 1–6.
[21] S. Mastorakis, P. Gusev, A. Afanasyev, and L. Zhang, "Real-time data retrieval in named data networking," in *Proc. 1st IEEE Int. Conf. Hot Information-Centric Netw. (HotICN)*, Shenzhen, China, Aug. 2018, pp. 61–66.
[22] Z. Ren, M. A. Hail, and H. Hellbruck, "CCN-WSN—A lightweight, flexible content-centric networking protocol for wireless sensor networks," in *Proc. IEEE 8th Int. Conf. Intell. Sensors, Sensor Netw. Inf. Process.*, Melbourne, VIC, Australia, Apr. 2013, pp. 123–128.
[23] M. A. Hail, M. Amadeo, A. Molinaro, and S. Fischer, "Caching in named data networking for the wireless Internet of Things," in *Proc. Int. Conf. Recent Adv. Internet Things (RIoT)*, Singapore, Apr. 2015, pp. 1–6.
[24] X. Xu, H. Zhang, T. Li, and L. Zhang, "Achieving resilient data availability in wireless sensor networks," in *Proc. IEEE Int. Conf. Commun. Workshops (ICC Workshops)*, Kansas City, MO, USA, May 2018, pp. 1–6.
[25] M. Amadeo, C. Campolo, A. Molinaro, and N. Mitton, "Named data networking: A natural design for data collection in wireless sensor networks," in *Proc. IFIP Wireless Days (WD)*, Valencia, Spain, Nov. 2013, pp. 1–6.
[26] S. Gao, H. Zhang, and B. Zhang, "Energy efficient interest forwarding in NDN-based wireless sensor networks," *Mobile Inf. Syst.*, vol. 2016, Apr. 2016, Art. no. 3127029.
[27] S. H. Bouk, S. H. Ahmed, K.-J. Park, and Y. Eun, "Interest broadcast suppression scheme for named data wireless sensor networks," *IEEE Access*, vol. 7, pp. 51799–51809, 2019.
[28] A. Aboud and H. Touati, "Geographic interest forwarding in NDN-based wireless sensor networks," in *Proc. IEEE/ACS 13th Int. Conf. Comput. Syst. Appl. (AICCSA)*, Agadir, Morocco, Nov. 2016, pp. 1–8.
[29] V. Lehman, A. K. M. M. Hoque, Y. Yu, L. Wang, B. Zhang, and L. Zhang, "A secure link state routing protocol for NDN," NDN, Tech. Rep. NDN-0037, 2016, pp. 1–9. [Online]. Available: https://named-data.net/techreports.html
[30] W. A. Jabbar, M. Ismail, and R. Nordin, "Multi-criteria based multipath OLSR for battery and queue-aware routing in multi-hop ad hoc wireless networks," *Wireless Netw.*, vol. 21, no. 4, pp. 1309–1326, May 2015.
[31] W. A. Jabbar, M. Ismail, R. Nordin, and R. M. Ramli, "Traffic load-based analysis of MBQA-OLSR routing protocol in wireless ad hoc networks," in *Proc. IEEE Region 10 Conf. (TENCON)*, Penang, Malaysia, Nov. 2017, pp. 2677–2682.
[32] W. A. Jabbar, W. K. Saad, and M. A. Ismail, "MEQSA-OLSRv2: A multicriteria-based hybrid multipath protocol for energy-efficient and QoS-aware data routing in MANET-WSN convergence scenarios of IoT," *IEEE Access*, vol. 6, pp. 76546–76572, 2018.
[33] M. Ismail, R. Nordin, R. M. Ramli and W. A. Jabbar. (2006). *Tmote Sky Data Sheet*. [Online]. Available: https://insense.cs.st-andrews.ac.uk/files/2013/04/tmote-sky-datasheet.pdf
[34] S. Khriji, R. Cheour, M. Goetz, D. E. Houssaini, I. Kammoun, and O. Kanoun, "Measuring energy consumption of a wireless sensor node during transmission: PanStamp," in *Proc. IEEE 32nd Int. Conf. Adv. Inf. Netw. Appl. (AINA)*, Krakow, Poland, May 2018, pp. 274–280.
[35] M. Doddavenkatappa, M. C. Chan, and A. L. Ananda, "A dual-radio framework for MAC protocol implementation in wireless sensor networks," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Kyoto, Japan, Jun. 2011, pp. 1–6.
[36] MSP430 Family. *Instruction Set Summary*. [Online]. Available: https://www.ti.com/sc/docs/products/micro/msp430/userguid/as_5.pdf
[37] A. Velinov and A. Mileva, "Running and testing applications for Contiki OS using Cooja simulator," in *Proc. Int. Conf. Inf. Technol. Develop. Educ. (ITRO)*, Serbia, The Balkans, 2016, pp. 279–285.
[38] J. Zhao and R. Govindan, "Understanding packet delivery performance in dense wireless sensor networks," in *Proc. Int. Conf. Embedded Networked Sensor Syst.*, 2003, pp. 1–13.
[39] S. Farahani, "Battery life analysis," in *ZigBee Wireless Networks and Transceivers*, vol. 6. Burlington: Newnes, 2008, ch. 6, pp. 207–224, doi: 10.1016/B978-0-7506-8393-7.00006-6.
[40] G. R. Kaur, P. Chawla, and M. Sachdeva, "Study of LEACH routing protocol for wireless sensor networks," in *Proc. Int. Conf. Commun., Comput. Syst.*, 2014, pp. 196–198.

TANUSREE CHATTERJEE received the M.Tech. degree in computer science from the Maulana Abul Kalam University of Technology, Kolkata, India, in 2012. She is currently pursuing the Ph.D. degree with the Department of Computer Science Technology, Indian Institute of Engineering Science and Technology, Shibpur, India. She has published research articles in reputed peer-reviewed journals as well as in refereed international conference proceedings. Her research interests include wireless sensor networks, named data networks, the IoT, and network security.

SOMNATH KARMAKAR received the B.Tech. degree in computer science and engineering from the Government College of Engineering and Leather Technology, Kolkata, in 2018, and the M.Tech. degree in computer science and engineering from the Indian Institute of Engineering Science and Technology, Shibpur, in 2020. He is currently working as a Researcher with TCS Research and Innovation Lab. His research interests include the Internet of Things, wireless sensor networks, machine learning, and artificial intelligence.

SIPRA DAS BIT (Senior Member, IEEE) has served as a Visiting Professor for the Department of Information and Communication Technology, Asian Institute of Technology, Bangkok, in 2017. She is currently a Professor with the Department of Computer Science and Technology, Indian Institute of Engineering Science and Technology, Shibpur, India. A recipient of the Career Award for Young Teachers from the All-India Council of Technical Education (AICTE), she has more than 30 years of teaching and research experience. She has published many research articles in reputed journals and refereed international conference proceedings. She has three books to her credit. Her current research interests include the Internet of Things, named data networking, wireless sensor networks, delay tolerant networks, and network security.