SDSense: An Agile and Flexible SDN-Based Framework for Wireless Sensor Networks

Israat Haque, *Senior Member, IEEE*, Mohammad Nurujjaman*, Member, IEEE*, Janelle Harms*, Senior Member, IEEE*, and Nael Abu-Ghazaleh, *Senior Member, IEEE*

***Abstract*—Software-defined networking (SDN) is a new approach to designing networks. SDN decouples and migrates network control from the hardware, which enables innovative and efficient network design and operation. The SDN-based design is well adopted in data centers and enterprises. In this paper, we state that SDN-based design is beneficial in the multihop wireless networks like wireless sensor network (WSN) and propose an architecture called SDSense. We propose a novel principled approach of SDN-based WSN design, where we decompose network functions as slow (e.g., topology control) and fast (e.g., congestion control) changing components. Furthermore, we derive a network utility maximization framework for better resource allocation. We implement SDNSense, where software enabled sensors are dynamically reconfigured to adapt to current network conditions and demonstrate that SDSense can significantly improve the network performance compared to its counterparts.**

***Index Terms*—Reliability, resource allocation, software-defined networking, wireless sensor networks.**

I. Introduction

SOFTWARE-defined networking (SDN) [1], [2] is a new approach to designing networks, where network control functions are decoupled from elements like routers and switches to make them simple packet forwarder. The SDN architecture comprises management, control, and data planes. The management plane defines the network policies, which are translated into configuration rules to be installed in the data plane elements by the control plane (controller). The management and control layers communicate through a Northbound API; whereas, Southbound (SB) API is used for the control and data plane communication. The controller can dynamically configure data plane elements using its global network view to meet the applications' demand as well as to avoid the potentially sub-optimal or unstable emergent behavior of traditional distributed protocols. SDNs have been developed and commercialized in the context of data centers and enterprise networks, and are starting to get explored in other networking environments.

In this paper, we study the advantages of using the software-defined network design in Wireless Sensor Networks (WSNs). A WSN consists of a set of resource-constrained sensor devices that are connected through a wireless medium. WSN continues to attract significant research and commercial interest due to their potential to revolutionize sensing across many application domains including environmental monitoring, surveillance and tracking, health monitoring, manufacturing, and smart homes [3], [4]. Moreover, WSNs can form the sensing component of distributed cyber-physical and autonomous systems and therefore have applications within this emerging domain [5].

Due to the shared nature of wireless networks, resource allocation is difficult, and WSNs are prone to inefficiency and unfairness as nearby sensors compete for access to the complex wireless medium. SDNs can offer significant advantages to WSNs by allowing the network to be configured to an efficient state. However, it also introduces some new challenges due to the dynamic nature of the resources. We develop a reliable, adaptive and efficient software-defined WSN architecture called *SDSense* in a principled way. SDSense monitors the network state and dynamically configure software-enabled sensors to meet application's demand for on-time critical event delivery to an observation center. For example, a healthcare monitoring application for assisted living may require a health critical event data, such as a notification of a patient falling as well as video of their current location to reach health providers and first responders reliably and in near real-time.

Some works attempted to design software-defined WSN networks. For instance, SDN-WISE [6], [7] proposed a stateful software-defined architecture for WSN, i.e., sensors maintain local state. The proposed architecture is evaluated for geographic routing and security. The distributed nature of WSN demands alternative software-defined architecture, where control functions can judicially be delegated to sensors to support the agility. SDN-WISE delegated neighbor list management task to sensors. However, there must be a principled approach of delegating control task to data plane elements to balance between efficiency and agility, which is missing in SDN-WISE. Also, SDN-WISE does not guarantee reliable critical data forwarding in the presence of link failure. SDIoT [8] implemented

Manuscript received December 3, 2018; accepted December 3, 2018. Date of publication December 19, 2018; date of current version February 12, 2019. This work was supported by the National Sciences and Engineering Research Council of Canada (NSERC). The review of this paper was coordinated by Dr. Z. Fadlullah. *(Corresponding author: Israat Haque.)*

I. Haque is with the Department of Computer Science, Dalhousie University, Halifax, NS B3H 4R2, Canada (e-mail: israat@dal.ca).

M. Nurujjaman is with the Department of Global Network, Cloud and Datacenter Services Tata Communications, Matawan, NJ 07747 USA (e-mail: mohammad.nurujjaman@tatacommunications.com).

J. Harms is with the Department of Computing Science, University of Alberta, Edmonton, AB T6G 2E9, Canada (e-mail: janelleh@ualberta.ca).

N. Abu-Ghazaleh is with the Department of Computer Science and Engineering,University of California, Riverside, Riverside, CA 92521-0429 USA (e-mail: nael@cs.ucr.edu).

Digital Object Identifier 10.1109/TVT.2018.2888622

interference mitigation without considering a complete architecture to support reliability, agility, and efficiency. SDSense bridges these gaps and proposes a principled approach of delegating control tasks to data plane elements. It furthermore guarantees reliable on time-critical data delivery by constructing Gabriel [9] based edge-disjoint routing topology.

In SDSense, we separate the management of static or slow changing phenomena in the network from those more dynamic phenomena that can change rapidly. For example, in a static WSN, topology may not change significantly over large time scales, whereas congestion at the sensors may change over small timescales. Static or slow changing phenomena should be managed directly using SDN principles while developing a more agile approach to lead dynamic phenomena.

The protocol stack of SDSense consists of interference management, reliable topology control, routing, and congestion management. We reduce contention but preserve reliability by deriving a new reliable routing topology that provides alternative paths to deliver event information when a primary route fails. To effectively use the bandwidth, we complement the topology with the derivation of a Time Division Multiple Access (TDMA) transmission schedules that separate interfering transmissions in time. Both of these components of the design are slowly changing and managed by the logically centralized controller with a global network view. The congestion can arise due to the unpredictable nature of event data, which is a fast-changing phenomenon. Thus, SDSense explores distributed congestion management approaches implemented directly by the sensors and using locally observed congestion to select forwarding direction.

Finally, we consider that even fast moving phenomena may have an underlying structure that changes more slowly. Employing just a distributed solution may result in increasingly inefficient or unstable configurations. Thus, we also implement a centralized Network Utility Maximization (NUM) algorithm to optimize the bandwidth allocation if the congestion problems persist, providing a balance between centralized nearly optimal solutions and the need for rapid reaction to transient phenomena. The resulting model is agile, in that the distributed component can react quickly to transient phenomena, but also efficient in that persistent problems lead to a centralized reassignment of resources.

In SDSense architecture, the logically centralized controller is homed at the sink but can generally be implemented anywhere in the network, and even across multiple sensors for redundancy and load balancing. The software-enabled sensors are configured using the control instructions from the controller. The controller collects information from the network to build and maintain a global network view, allowing it to carry out effective resource allocation. The topology control, scheduling, and routing modules reside on the controller. Thus, adding or removing a module can be controlled through the northbound API from the users' applications. Each sensor has a local controller to communicate with the centralized one, but also to manage any distributed functionality delegated to the sensor. We implement SDSense and compare its performance with non-SDN based design to demonstrate that SDSense can significantly improve the performance in terms of successfully delivering critical event data in the presence of interference, link failures, and unpredictable congestion.

TABLE I
LIST OF ACRONYMS USED IN THIS WORK

Acronyms	Elaboration
SDN	Software Defined Networking
WSN	Wireless Sensor Network
NUM	Network Utility Maximization
SDN-WISE	SDN-WIreless SEnsor network
SDIoT	Software Defined Internet of Things
SDWN	Software Defined Wireless Network
SDWSN	Software Defined Wireless Sensor Network
RPL	Routing Protocol for Low-Power and Lossy Networks
SOF	Sensor OpenFlow
WNOS	Wireless Network Operating System
MST2	2-edge connected Minimum Spanning Tree
SPT2	2-edge connected Shortest Path Tree
DCSPT2	2-edge connected Degree Constrained SPT
Gabriel2	2-edge connected Gabriel topology
DCGB	2-edge connected DCSPT and Gabriel topology
MSGB	2-edge connected MST and Gabriel topology

Contribution: We introduce a principled approach of designing a software-defined WSN and propose an agile architecture called, SDSense, to offer reliable and efficient on time critical event data delivery. The reliability is achieved through the construction of an edge-disjoint topology. The efficiency and agility are realized by delegating control tasks among local and global controllers in a principled approach. SDSense controller monitors network states to dynamically allocates resources and orchestrate the distributed operations at the sensors. Furthermore, we propose an adaptive resource allocation model to manage the congestion within WSNs, where the controller reallocates the available bandwidth in areas where congestion occurs. We formulate the problem as a NUM problem and evaluate it on representative scenarios. The evaluation results confirm that SDSense can enhance the average network throughput by 80% compared to its counterparts while using the proposed demand based rate allocation. Table I presents a list of acronym used in this work.

The remaining paper is organized as follows. Section II presents and reviews state-of-the-art software-defined WSN work related to SDSense. The optimization model for joint routing and rate allocation is presented in Section III. We present the design of each functional module of SDSense architecture in Section IV. The next section defined the evaluation environment for SDSense. Section VI discusses on the evaluation results following the concluding remarks in Section VII.

II. RELATED WORK

This paper argues by construction that SDN is an effective framework for developing protocols for WSNs. The decoupled data and logically centralized control plane along with the network programmability provide a flexible framework for implementing network control strategies that are both efficient (e.g., implemented centrally) and agile (e.g., supported by distributed control on the sensors). We believe that this is one of the first systematic efforts to explore the use of SDNs in WSNs [10],

[11]. In the following, we outline SDN-based WSNs architectures and protocols that are closely related to SDSense.

SDN-WISE [6], [7] proposed a stateful software-defined architecture for WSN, i.e., sensors maintain the state information of themselves and their neighbors. Sensors use this state information to construct routes to convey control traffic towards the control plane. SDN-WISE protocol stack consists of forwarding, in-networking processing, and topology discovery layers. The proposed architecture implements geographic routing and security without guaranteeing reliable data transfer in the presence of link failures. In addition, it does not provide any principled approach of control tasks distributions.

SDWN [12] proposed a stateless architecture similar to SDN-WISE for data aggregation and routing. However, the work in [13] and SDN-TAP [14] extended SDN-WISE to support congestion control and load balancing. In the former design, packets are dropped according to their flow priority to realize congestion control and load balancing. In SDN-TAP, controller-defined alternate routes are followed by the traffic after experiencing congestion. In [15] load balancing problem is formulated as an optimization problem which cannot be solved in polynomial time; thus, a heuristic is proposed. Baddeley *et al.* deployed RPL (Routing Protocol for Low-Power and Lossy Networks) [16] protocol for control traffic communication in software-defined IoT (SDIoT) [8] network design. The controller dynamically updates flow rules to allow high-priority flows to follow alternate routes in the presence of interference and congestions. In SDSense, congestion control and load balancing are accomplished through designing an adaptive and optimal rate allocation model at the controller. In addition, the load is distributed among sensors through multiple edge-disjoint routes. These routes are further used to separate control and data traffic with associated priority queues.

A set of work considered energy minimization while designed software-defined WSN. For example, Smart [17] migrates the topology control, routing, QoS, mobility management, and localization in the controller, which makes sensors as simple forwarders and reduces their energy consumption. In [18] controller selectively broadcasts control packets to a subset of sensors to reduce energy consumptions. An extension of SDWN controls transmission power of sensors with the help of a controller to save sensors' energy [19]. A minimum-energy sensor activation problem is formulated in [20] as an optimization problem without supportive evaluation. SDSense by constructing Gabriel-based topology guarantees optimal communication energy consumption. In addition, edge-disjoint routes in SDSense can be used to support reliability, congestion control, and load distribution.

Friedman *et al.* [21] proposed an extension of the OpenFlow protocol and open virtual switch (OVS) to implement data aggregation. The aggregated traffic uses conventional sensor channel, whereas the control traffic between sensors and the controller uses OpenFlow. SensorOpenFlow (SOF) [22] also extended OpenFlow to support coexisting applications, data aggregation, and in-band control traffic management. The authors in [23] suggested using separate flow tables for routing and in-network processing. These solutions do not support reliability, control task delegation principle, and optimization model. In addition, they have little or no evaluation of their proposed design.

In WNOS [24] distributed operations of the data plane is abstracted to applications to design a software-defined WSN Operating System. It exploited the centralized view of the network at the controller to automatically generate the distributed cross-layer control program, which runs on sensors with an abstract representation of radio hardware. In WNOS, control tasks delegation is not done in a principled approach as in SDSense. In addition, WNOS focused on the physical layer functions; thus, may use SDSense to deploy additional layers. A survey on software-defined WSNs can be found in [25], [26] and a survey on software-defined wireless networks is presented in [10].

III. A Model for Optimal Resource Allocation

In WSNs, distributed routing and congestion control usually do not consider the entire network state, but rather rely on local network state. Such local decisions may lead to inefficient decisions, and oscillation may occur as decisions lead to congestion moving from one location to another. Moreover, selecting routes without considering available link capacity may have a direct impact on the congestion. Finally, congestion may persist and impact on time-critical data delivery if adaptive network operation is missing. These design challenges can efficiently be mitigated using software-defined WSN design as proposed in SDSense. However, using a globally centralized controller may not support the distributed network operations of WSNs. The distributed operations can be optimized and orchestrated by the logically centralized controller; whereas local controllers from the sensors can implement that optimal but distributed operation. The proposed SDSense is based on this design philosophy, where network functions associated with slowly changing states are defined and optimized in the control module; whereas functions that are associated with the fast-changing state are designed using a NUM problem [27]. Thus, in this section, we define a model of joint routing and rate allocation as a NUM problem.

We define a multihop wireless network as a set V consisting of N sensors placed in a two-dimensional Euclidean space. Each node is equipped with a single radio that can either receive from or transmit to a neighbouring node within the radio range. Let d_{ij} be the Euclidean distance between two nodes i and j. There exists a radio link and a corresponding edge between nodes i and j if they are in range with each other. More precisely, $l = (i, j) \in E$ iff $d_{ij} \leq R$. We further assume a fixed transmission rate (or link capacity) c_l. Let $G(V, E)$ be the graph representation of the network with N nodes and E edges, respectively. The edges are chosen based on the radio range constraint. If we consider *protocol interference model* [28], then a conflict graph G_{cg} of $G(V, E)$ can be defined as $G_{cg}(V_{cg}, E_{cg})$, where V_{cg} corresponds to the links from G and E_{cg} are the edges among them. Two nodes from V_{cg} are connected by an edge if they share a common node (*primary interference*).

To control interference, we implement TDMA scheduling where the transmission time is divided into globally synchronized slots. Each link is allocated one or more slots to meet the traffic demand. Multiple links can be active in the same time slot if they meet the constraints of the interference model: in other words, they share no edges in G_{cg}. Note that alternative interference models can be supported by identifying the legally concurrent edges based on the interference model rules. We call a set of active links assigned to a particular slot a *configuration* u. The union of configurations is the set U of all link sets that can be active simultaneously. Let x_l^u and $x_{l'}^u$ be two binary variables that take a value 1 if corresponding link is active in a configuration u and 0 otherwise. The single radio constraint for these two links per the protocol interference model is:

$$x_l^u + x_{l'}^u \leq 1 + a_{ll'}^u \ \forall (l, l') \in E \ l \neq l' \tag{1}$$

where the binary variable $a_{ll'}^u$ is defined as:

$$a_{ll'}^u = \begin{cases} 1 & \text{if links } l \text{ and } l' \text{ have no common nodes} \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

For a configuration, u, there will be a set of active links with corresponding rate assignments such that a particular link l's entry, r_l^u, is defined as:

$$r_l^u = \begin{cases} x_l^u \times c_l & \text{if } l \in u \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

The feasible rate region r_m at the link layer can be defined as:

$$r_m = \sum_{u \in U} r_l^u \ \ \forall l \in E \tag{4}$$

Next, let us consider f_m which is a particular flow generated from a sensor m, $m \in S$, such that:

$$f_m = \begin{cases} r_m & \text{if } m \in S \\ -r_m & \text{if } m \in D \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

Let f_l^m be the flow at a node i, $i \in N - (S, D)$, through a link $l \in E$ for sensor m. Let $w^+(i)$ and $w^-(i)$ be the set of all outgoing and incoming links from and to that node i, respectively. The flow conservation constraints can be stated as:

$$\sum_{l \in w^+(i): i \in N} f_l^m - \sum_{l \in w^-(i): i \in N} f_l^m = 0 \ \forall i \in \{N - S, D\} \tag{6}$$

$$\sum_{l \in w^+(i): i \in N} f_l^m - \sum_{l \in w^-(i): i \in N} f_l^m = r_m \ \forall i \in \{S\} \tag{7}$$

$$\sum_{l \in w^+(i): i \in N} f_l^m - \sum_{l \in w^-(i): i \in N} f_l^m = -r_m \ \forall i \in \{D\} \tag{8}$$

Let a configuration u be active for the time λ^u such that $\lambda^u \geq 0$ if u is scheduled, otherwise $\lambda^u = 0$. The link capacity constrain then be defined as:

$$\sum_m f_l^m \leq \sum_{u \in U} \lambda^u \times x_l^u \times c_l \ \ \forall l \in E \tag{9}$$

where

$$\sum_{u \in U} \lambda^u \leq T \tag{10}$$

$$f_l^m \geq 0$$

Next, we define the network utility function for our joint optimization. Let Q_m be the size of the queue of a sensor m and corresponding rate be r_m. Let b_m be the minimum rate at a sensor m that is defined as:

$$b_m = Q_m \log(r_m) \tag{11}$$

The optimization problem allocates link bandwidth in such a way that the perceived network utility is maximized:

$$\text{maximize} \ \sum_m Q_m \log(r_m) \tag{12}$$

subject to : Equation 6 to 9

The objective function is a convex function that balances the following two requirements: (1) it favors solutions where the rate at each link is proportional to corresponding sensor demand. We define the demand of each sensor as a function of the available data at the sensors, and (2) the function rewards fair allocation of the available rate among the sensors. The routing constraints (Eq 6 to 8) preserve the flow constraint while choosing routes from sources to destinations. The scheduling constraint states which links can be active at the same time without interfering with each other. We use a standard scheduling approach: first, we compute the *conflict graph* for a given routing topology. Next, we define the feasible schedule region based on the derived conflict graph. For evaluation purposes, we consider the most widely adopted *Protocol Interference Model* [29] to define the conflict graph.

IV. SDSense Architecture

In this section, we define SDSense architecture that accommodates a logically centralized controller along with a local controller associated with the software-enabled sensors. The centralized control allows intentional control of the network to place it in a state that may be difficult to achieve as emergent behavior from distributed protocols. In addition, the functionality of the network, which is encapsulated in the control plane is centralized and can be modified, providing the ability to evolve protocols easily, possibly without reprogramming the sensors [30]. Resulting reconfiguration of the sensors is then isolated to changes to the data plane configuration which is supported by the SDN operation.

In SDSense, the data plane is composed of software-enabled sensors associated with a *local controller*. The control plane or controller manages the network functions and resides in the sink, as well as distributed control agents that operate on the sensors. We compose various network functions as individual modules as part of the controller architecture. In addition, the controller allocates tasks like dynamic radio parameter setting to the local controllers. This decomposition reduces the control burden from the global controller, as well as improves the response time for the adaptation of the radio parameters where the

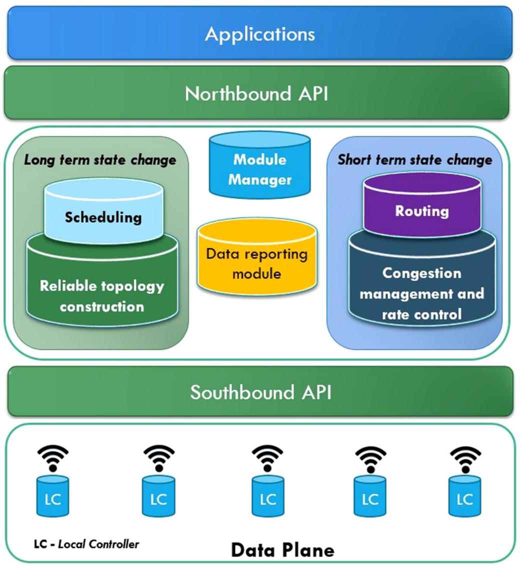


Fig. 1. Architecture of SDSense.

local controller can take effective decisions based on local information. In this design, we emphasize exposing mechanisms at the local controller but leave policy specification at the centralized controller, leaving the local controller essentially to be part of the data plane. The decomposition also reduces the control traffic flow between the sensors and the controller.

The architecture (Figure 1) supports essential modules such as interference management and scheduling, topology control, routing, and congestion control. However, any required functions can be added to the controller as a new module without modifying the existing architecture or reprogramming the sensors. Several active modules execute in the controller; their decisions must be combined to configure the network. We support this functionality using a *controller manager* module that is aware of all active modules and interfaces with them to inform them of the relevant network state. In the following, we present the design of individual modules in details.

Topology control module: This module incorporates with the local controllers to construct and control reliable routing topology. The local controller maintains and shares neighbor list with the global controller, which on the other hand constructs and maintains the entire network topology. Once the neighbor lists from all the local controllers are gathered, the reliable routing topology construction is conducted. This topology construction is briefly outlined below, which is presented in detail in our previous work [31].

The software-enabled sensors from the data plane can be represented as a graph *G(V,E)* consisting of V sensor nodes placed in a 2-dimensional Euclidean space. We assume that the transmission range of all sensors is R. Two nodes can communicate with each other if and only if their Euclidean distance is at most R. The ability to communicate is represented by an edge, in E, between the corresponding nodes. The resulting graph, *G(V, E)*, is the *physical topology* of the network. G varies over time due to the presence or absence of the links among nodes. The topology construction problem is to define a subgraph of a given

Algorithm 1: $k-$ Edge Connected Topology $D_k(G)$.

```
1: Extract $E_1$, the edges of a component-connected
   spanning subgraph of $G$.
2: $D_1(G) \leftarrow E_1$.
3: for $i \leftarrow 1$ to $k-1$ do
4:   Remove edges of $D_i(G)$ and get the connected
     components $D_{i1}(G), D_{i2}(G), \ldots, D_{il}(G)$
5:   for $j \leftarrow 1$ to $l$ do
6:     Generate component-connected spanning subgraph
       $D_{ij}(G)$.
7:   end for
8:   Merge $D_i(G), D_{i1}(G), D_{i2}(G), \ldots, D_{il}(G)$ to obtain
     $D_{k-1}(G)$
9: end for
```

physical topology, where packet routing will be performed; we call this subgraph the *routing topology*.

We use a spanning structure as the basis of the construction of the routing topology. We consider several spanning structures: Minimum Spanning Tree (MST) [32], Shortest Path Tree (SPT) and Gabriel Graph (GG) [9]. The detail definition of these spanning trees can be found in [31]. The reliable topology control in SDSense uses GG, which can be defined as follows; Assume that $disk(u, v)$ is denoted by a circle containing u and v with the diameter equal to the Euclidean distance between them. Then edge (u, v) of G belongs to GG if and only if no other nodes $w \in V$ are located within or on the periphery of $disk(u, v)$ [9]. Gabriel graphs have relatively short links and therefore good energy-use properties. The algorithm to design k-edge connected reliable routing topology is presented in Algorithm 1.

Thus, if we construct a $k = 2$ edge-connected GG-based routing topology, there will be 2-edge disjoint routes between every pair of sensors. We call that 2-edge connected routing topology Gabriel2. Similarly, we use the terms MST2, SPT2, and DCSPT2 to represent MST, SPT, and DCSPT (Degree-Constrained SPT) based 2-edge connected routing topology, respectively. In addition, Gabriel is combined with the other topologies as follows: DCSPT-Gabriel (DCGB) topology is a subgraph of G which first constructs the subgraph following definition of DCSPT, then uses the Gabriel graph construction on the remaining components. MST-Gabriel (MSGB) combines the MST construction (first) followed by the Gabriel construction.

Complexity analysis: Finding the DCSPT, SPT, and MST is $O(V^2)$ if no special data structures are used. Whereas, for Gabriel, if a node's degree is d, then the algorithm requires d^2 operations to find its edges. Therefore, in the worst case, each node takes $O(V^2)$ to obtain the edges of the routing topology. However, each node can find the edges locally, in parallel, if implemented in a distributed way which takes $O(V^2)$. In the subsequent steps, we repeat the topology construction on the edges that are not chosen in step 1, where the number of nodes remain unchanged. Thus the complexity depends on k, i.e., $O(kV^2)$. In the following, we show that GG topology can optimize the communication energy consumption, which is the routing topology in SDSense.

Definition 1: Let the distance between two vertices $u, v \in V$ in $G(V, E)$ be the total weight of the shortest path between u and v and be denoted by $d_{u,v}$. A subgraph $G'(V, E')$, where $E' \subseteq E$, is a t-spanner of G if for every $u, v \in V$, $d'_{u,v} \leq t \cdot d_{u,v}$, where $d'_{u,v}$ is the shortest path between u and v in G'. The value of t is called the spanning ratio, i.e., the ratio of actual path over the shortest path. If we replace the weight with the communication energy, then corresponding spanning ratio is called *energy spanning ratio* [33], [34].

The communication energy consumption can be defined using simple propagation model, i.e., $d^{\alpha}_{u,v} + C_k$, where α is the path loss exponent and C_k is a constant. Thus, if a topology guarantees energy spanning ratio of 1, then it is called a *energy spanner*. It is proved in [34], [35] that GGis an energy spanner.

Theorem 1: The Gabriel based k-edge connected topology $D_k(G)$ for $k = 2$ is an energy spanner for the path loss exponent $\alpha \geq 2$.

Proof: Construct a Gabriel based $D_1(G)$ rooted at r on a given connected graph $G(V, E)$. By definition $D_1(G)$ is an energy spanner for $\alpha \geq 2$. Now consider $G(V, (E \setminus \tilde{E}_1))$ to extract the Gabriel edges E_2 and add these newly generated edges to $D_1(G)$ to obtain $D_2(G)$. Assume that adding E_2 to $D_1(G)$ changes its spanning property. This can only happen if the set of edges $\tilde{E}_1$ are altered while choosing E_2. However, E_2 are chosen from $(E \setminus \tilde{E}_1)$ leaving $\tilde{E}_1$ and the spanning properties of $G(V, \tilde{E}_1)$ unchanged. Thus, the Gabriel based $D_k(G)$ for $k = 2$ preserves the energy spanning property and is an energy spanner. ■

Once the reliable edge-disjoint routing topology is constructed at the global controller, it is shared among the local controllers from the data plane elements. The Gabriel based reliable topology can locally be constructed at the sensors without controller's intervention. Local controller can furthermore deploy a distributed scheduling. This can reduce control overhead and improve network resource utilization.

Scheduling Module: SDSense deploys a collision-free TDMA [36] scheduling algorithm in this module to arbitrate the shared bandwidth by assigning links to slots to control the packet delivery. Our assisted living applications demand low latency. Thus, we focus on TDMA based scheduling, which is energy efficient and guarantees low delay compared to the CSMA based solutions [36]. In particular, the sink computes a *conflict graph* [36] that reflects which groups of links mutually interfere and hence cannot be active simultaneously. This graph is then colored using a heuristic (highest degree node is colored first) to ensure that interfering links of the original routing topology will not be assigned to the same slot. This module collects the current routing topology from the topology module and derives a TDMA based schedule. The scheduled slots are then disseminated to the local controllers to configure packet forwarding.

Routing Module: This module has the task of determining how the traffic will be routed through the network given the available topology, sensor demands, and TDMA slot assignments. In the distributed sensor network design, the forwarding decisions are made locally, alternating between the two shortest path parents. However, in SDSense the traffic will be routed based on the solution of the NUM problem; a flow table entry is configured by the *routing module* based on the NUM solution. When a sensor receives a packet from a particular flow, it first looks for an entry in the flow table. The packet is sent to the port mentioned in the flow table entry. In the case of a packet to a destination without a flow entry, the sensor requests a flow table entry from the routing module. This is a control message and will be routed using the default strategy in Algorithm 2. Algorithm 2 states that initially each sensor chooses their two best parents, $W_1(v_i)$ and $W_2(v_i)$, along the optimal and the next optimal routes. During initial packet forwarding a sensor sends a packet to $W_1(v_i)$ and waits for the ACK that includes the current queue status. In the subsequent forwarding, it checks whether the queue utilization of $W_1(v_i)$ reaches a threshold γ, which is a configurable fraction of the queue capacity that can be used to adjust the sensitivity of the algorithm. In that case, the packet is forwarded to $W_2(v_i)$. A sensor maintains a constant number of neighbors with whom it communicates. Thus, the communication and computation cost at a sensor is $O(1)$; thus the complexity of Algorithm 2 is $O(V)$ for a network of V sensors.

Algorithm 2: Protocol to Forward Control Traffic.

```
 1: Initialization:
 2: for $i \leftarrow 1$ to $|V| - 1$ do
 3:     set two best parents $W_1(v_i) \in N(v_i)$ and
        $W_2(v_i) \in N(v_i)$ for each sensor node $v_i$.
 4: end for
 5: Forwarding at node $i$:
 6: if $Queue(W_1(v_i)) \geq \gamma \times MaxSize$ then
 7:     $NHop(v_i) \leftarrow W_2(v_i)$
 8: else
 9: $NHop(v_i) \leftarrow W_1(v_i)$
10: end if
```

Upon receiving a new configuration request, the routing module configures the flow table of the requested sensor based on the global network view and the NUM solution. However, the local controller has the flexibility to control the global configuration to improve the efficiency. For example, the local controller is configured to choose the available path in case one of the two paths becomes unavailable. The local controller is also configured in such a way that it attempts to send each packet until the retry limit is reached before dropping it. We maintain two separate queues $Queue_d$ and $Queue_c$ for data and control messages, and their corresponding weights are W_d and W_c, respectively, where $W_c > W_d$.

Congestion control module: This module is designed to dynamically identify and react on the congestion based on the optimal rate allocation model proposed in Section III. Sensors gather event-based as well as periodic sensed data [37]: event-based data is unpredictable and can be bursty in nature. Moreover, event-data is often critical and must reach the sink as quickly and reliably as possible [38]. One option in such a case is to increase the rate allocated to sensors with such data.

The local controllers are configured in such a way that in the case of unexpected queue build-up, it reports this event to the global controller. The global controller learns the congestion events through different local controllers. In the case of a number of such congestions events, the global controller can reassign (reschedule) the rate for the sensors based on their current demand. Once the queue returns to a lower size, the local controllers again report this event to the global controller. Effectively, the global controller is learning and reacting to the network state accordingly. In the case of non-SDN based queue based congestion management mechanism [38], the local choice of an upstream sensor based on its queue-status can lead to route oscillation if intermediate sensors on the way to the sink are also congested.

Data reporting module: The local controller on each sensor selects which data to forward based on data reporting configuration set by the centralized controller module. The local controller also takes into account the availability of transmission bandwidth. For instance, we can have both periodic and event-based bursty data. When such an event occurs, the local controller may want to increase the rate to send the event data to the sink as quickly as possible. If needed, the local controller sends a control message to request the data reporting module to reconfigure its current settings.

V. Evaluation Setup

In this section, we define the simulation environment and the scenarios used in the performance evaluation. We implement and evaluate the performance of SDSense and its functional modules based on the design presented in Section IV. The controller is implemented at the sink, which controls a set of software-enabled sensors with associated local controllers. We implement the controllers and API in our simulation environment in detail, modeling all control traffic, along with the data traffic, within the same network.

We consider scenarios with N sensors that are placed according to a uniform distribution in a square 100 m $\times$ 100 m area. The transmission range of each sensor is fixed at 25 m. We vary the number of sensors N ($N \in 50, 75, 100, 125, 150$) to control the density of the network. The logically centralized controller collects the neighbor lists from local controllers. It then builds the $k-$ edge connected topology, finds the conflict-free TDMA schedule, and forwards this information to the local controllers. We assume the presence of a sink and a set of 20 sources that generate traffic at the beginning of each scheduling period. In our evaluation, sources generate low to high volume of traffic with probabilities ranging from 0.2 (low) to 1.0 (high). This generated traffic are then routed towards the sink based on the route configured by the logically centralized controller.

Given a physical topology, we first extract a 2-edge connected routing topology. Any sensors from these routing topologies can serve as a sink while offering 2-edge connected routes to rest of the sensors. Thus, we choose the sensor with minimum scheduling length as the sink; the idea is to help the packets to reach a destination with minimum possible delay. Interestingly, minimizing the schedule length leads to different locations of the sink (according to this optimality criteria) for different topologies. All results are averaged over 100 experiments with different locations of sensors which were sufficient to bound the 95% confidence intervals to be with 1% of the shown average.

The local controllers measure the congestion level at the sensors as the occupied percentage of packets in the associated queue. The sensors are considered congested if this percentage exceeds a threshold triggering a notification to the centralized controller. In the current evaluation, this threshold is 60% of the queue size set based on an empirical evaluation. The local controller sends a control message to the centralized one to inform events such as the presence of congestion or the presence of new route configuration requests. In the case of congestion, the centralized controller monitors the number of such congested sensors and the number of periods a sensor remains congested. If a sensor remains congested for several consecutive periods or a set of sensors report a congestion event, the global controller invokes and solves the rate allocation problem to reallocate the available network bandwidth. Generally, this leads to raising the capacity allocated to congested sensors. Over the time the sensors will get out of the congestion, and release the extra capacity demand to be used by other sensors in a future invocation of the reallocation problem. The evaluated results are compared to traditional distributed solutions.

VI. Performance Evaluation and Discussion

In this section, we first present a set of results that are conducted to select an appropriate reliable routing topology to be adopted in our SDSense framework. Then, we will illustrate that traditional distributed routing and congestion management will not be appropriate to support on time critical data delivery where sensors' resource demand and network congestion level changes. We will then present the performance of SDSense to show that software-defined network monitoring and management along with adaptive resource allocation can efficiently support the above-mentioned critical data delivery applications.

A. Reliable Topology Selection

We first measure the degree of reliability (the ability to provide alternative paths in the presence of broken links) of different 2-edge connected routing topologies. We then evaluate their performance in terms of throughput, path length (delay), and communication energy usage in various traffic conditions. The results lead us to select an appropriate routing topology to be deployed in SDSense. We define the *degree of reliability* as the resilience of a routing topology to broken links estimated as follows. We randomly choose a subset of links to remove from each topology and measure the corresponding path length. This result is shown in Figure 2 with different percentages of randomly chosen failed or down links. MST2 has the worst performance due to its longer hop paths. The Gabriel-based topologies have the best performance due to more available alternative paths.

The next set of experiments is conducted with varying network traffic density by changing the packet generation probability at the traffic generators. In these experiments, we introduce a link failure probability which is fixed for all links. Figures 3 and 4 present the average packet delivery ratio, path length,

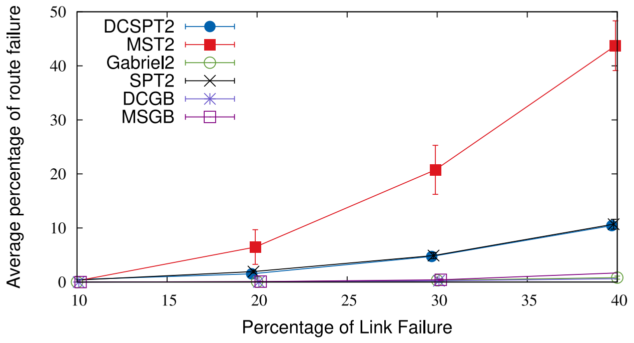


Fig. 2. Reliability measurement in the presence of link failures.

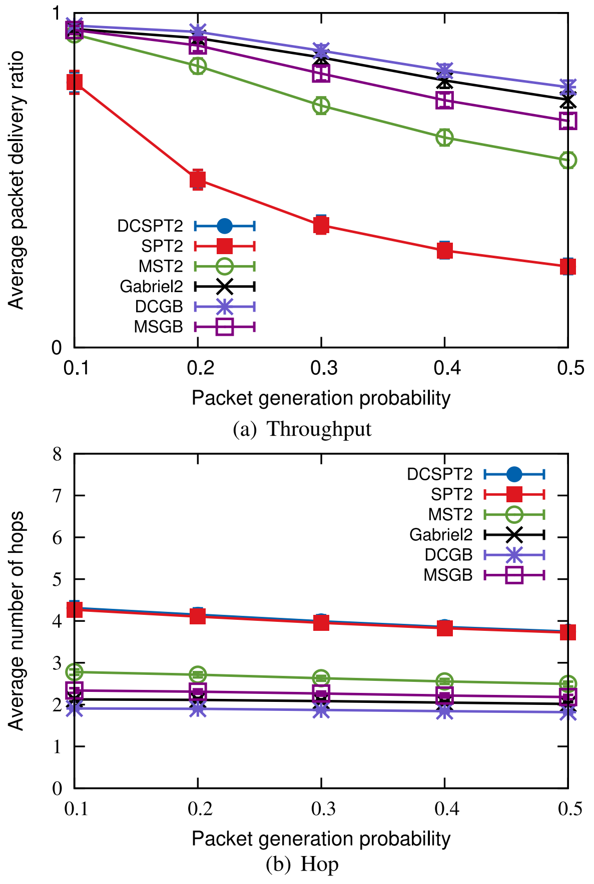


Fig. 3. The average packet delivery ratio and number of hops with different packet generation probability.

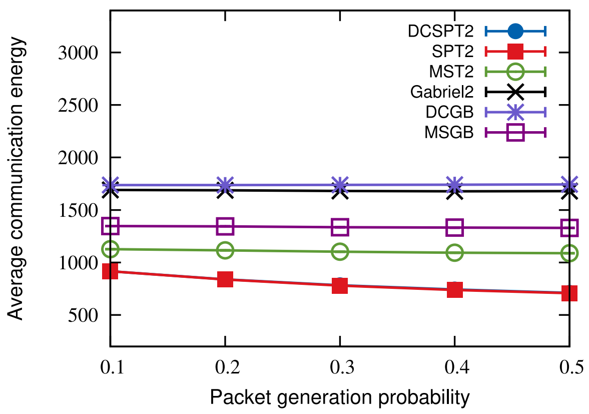


Fig. 4. The average communication energy uses with different packet generation probability.

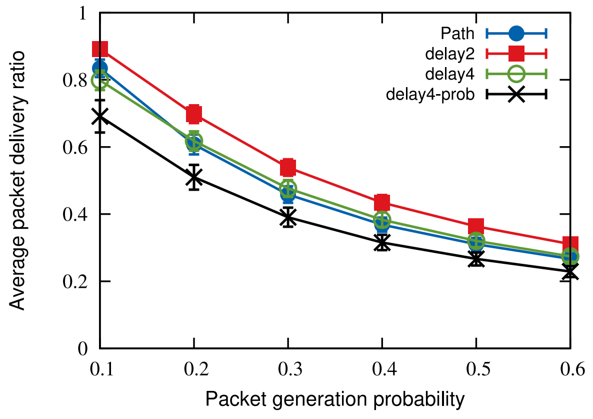
and energy consumption. The interesting outcome of these experiments is the impact of sink locations on the performance. The sink location with optimal schedule period is different for different topologies; this leads to a substantial difference in their performance. For example, the sinks are mostly located at the edge of the network for DCSPT2 and SPT2, which leads to higher average path lengths. The location of the sink was selected during topology construction to minimize the average transmission schedule: for DCSPT2 and SPT2 this results in favoring edge nodes given the high contention in the middle of the topology. Longer paths are more likely to break with link failure, leading to a lower packet delivery ratio. In contrast, in the Gabriel-based topologies, the sink locations that provide the smallest schedule length are mostly centrally located. This results shorter path lengths that eventually help packets to quickly reach destinations, despite the average static path length being slightly longer in Gabriel graphs. It is likely that the other topologies could benefit from a more centrally located sink even if it does not minimize the schedule length, but we did not consider such heuristics in sink selection.

We also measure the communication energy used by different topologies. We consider the energy for both transmissions and any retransmissions in case of unsuccessful attempts. For this reason, we expect the energy consumption of the topologies (shown in Figure 4) to be similar to their delivery ratio. The downward trend of the graphs in Figure 3 and 4 can be explained by the lower packet delivery ratio at higher loads, leading to more frequent delivery failures due to contention, particularly for longer paths. We conclude that Gabriel-based routing topologies are more robust than other topologies. These topologies also have the best communication energy consumption with a moderate path length. Thus, we deploy Gabriel based routing topology in the proposed SDSense architecture.

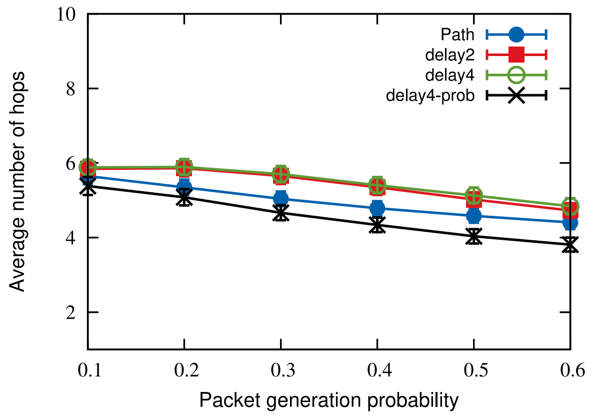
B. *Performance of Distributed Protocols*

In this set of experiments, we evaluate the performance of distributed congestion-aware routing on Gabriel-based topology. We consider a set of deterministic and non-deterministic protocols to show that distributed protocol may not be a good candidate to support on time critical data delivery.

Figure 5 compares the four distributed routing protocols on Gabriel-based topology. The *path* strategy alternates between the shortest and second shortest routes. We have defined three variants of *path*; namely, *delay2*, *delay4*, and *delay4-prob*. The former two and *path* are deterministic schemes; whereas, the latter one is a probabilistic approach. The *delay2* and *delay4* select the less congested next-hop parent out of two and four best choices, respectively. In the case of *delay4-prob*, four best next-hop candidate neighbors are assigned weight relative to their queue utilization. Then, a less congested neighbor is chosen probabilistically. The simulation results show that considering congestion helps improve the throughput of *delay2*. The other



(a) Throughput



(b) Hop

Fig. 5. The average packet delivery ratio and number of hops in Gabriel based topology.

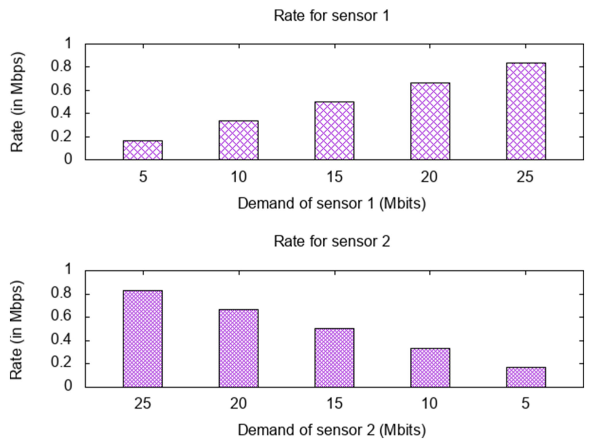


Fig. 6. Dynamic rate sharing between two sensors.

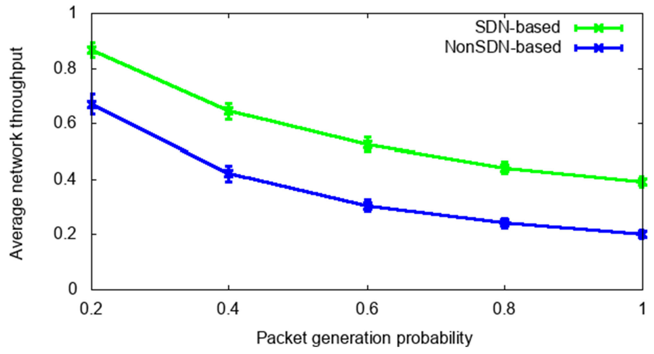


Fig. 7. The average network throughput with varying traffic load.

two variants that consider more parents and potentially longer paths take a longer time to reach the sink. Thus, having at least one high-quality forwarding alternative helps performance, but having more does not lead to additional improvements [39]. Figure 5 shows that *delay4-prob* experiences short delays on a successful packet delivery, which indicates this approach may be useful only for closely located source-destination pairs. *Path* uses slightly longer paths with a higher throughput compared to the probabilistic approaches. Thus, we may conclude that deterministic distributed packet forwarding is useful compared to the probabilistic approach. However, the lack of global network view may lead these deterministic solutions to a local optimal and may create oscillation across the network. In the following evaluation, we will show how the global network view and the network reprogramming capability of SDSense help it to achieve better network performance.

C. Performance Improvement Using SDSense

In this section, we first present a numerical analysis of our NUM model to illustrate its demand-based fair rate allocation. For this purpose, we consider a simple scenario where two sensors share radio resources while propagating sensed data to a sink. Each communication link has a capacity of 2 Mbps, where the demand for these two sensors changes over time. According to our model rate allocation must be demand-based, which is reflected in the rate allocation shown in Figure 6. In addition, the figure shows that the model ensures fair rate allocation, which allows sensors with low demand to continue to forward their traffic.

Next, we perform an extensive performance evaluation of the proposed SDSense framework in terms of throughput, path length, and communication energy consumption and compare the results with traditional distributed solutions presented in [31]. For this purpose, we consider 100 sensors classified as 20 sources, one sink (where the controller is implemented), with the remaining sensors relaying traffic if necessary. We also vary the traffic load from light to heavy to observe the rate adaptation performance of SDSense. In addition, we create link failure to capture the reliability performance of SDSense. To the best of our knowledge, SDSense is the first software-defined WSN framework that supports reliability in addition to performance and scalability (in terms of supporting varying loads).

The average network throughput of SDSense and traditional distributed protocols is presented in Figure 7. The demand-based rate allocation through programmable sensor network significantly improves the performance. In particular, we consider low to high traffic density and observe that in the low traffic density, the SDN-based dynamic resource allocation improves the overall network throughput by 30%. The non-SDN based scheme does not have dynamic rate control but locally chooses a less congested neighbor on the way towards the sink. Nonetheless, this congestion-aware neighbor selection is not enough under

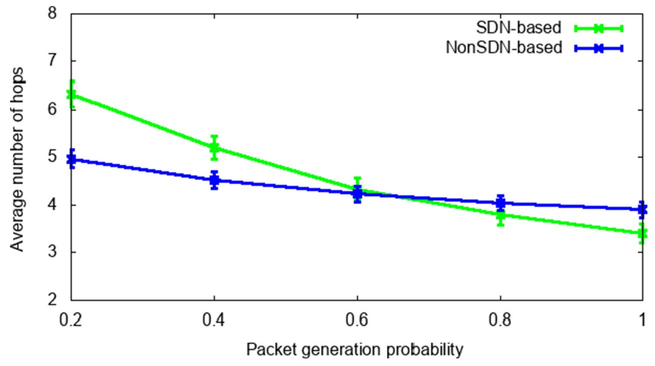


Fig. 8. The average path length with varying traffic load.

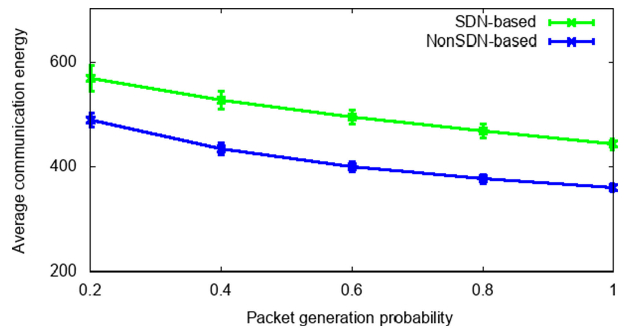


Fig. 9. The average communication energy consumptions.

heavy traffic density. As the traffic increases, the demand-based resource allocation enhances the performance by 80% compared to the distributed heuristic based congestion management.

The average path length in terms of number of hops is shown in Figure 8. The non-SDN based path length is smaller compared to the SDN-based one under low to moderate traffic load. However, the scenario is flipped under heavy traffic conditions. This behavior suggests that the demand based rate allocation can better manage the heavy traffic condition to offer high throughput leaving the network less congested. However, this may happen for the source-destination pairs that are not separated by a long path, so that the packets can reach the destination quickly. This is observed in the Figure 8, where the path lengths get shorter with the increasing traffic loads.

Furthermore, we evaluate the average communication energy consumptions of the successfully delivered packets for the SDN and non-SDN based schemes. For this purpose, we consider both the transmission and retransmission attempts of the successfully delivered packets. Thus, it is expected that the energy consumption behavior would follow the throughput trend. The SDN-based scheme delivers more packets compared to the standard solution, which is reflected through the higher communication energy consumptions presented in the Figure 9. We would like to mention that the performance trend in terms of the communication energy consumption with or without software-defined design is similar. This is because communication energy consumption depends on the link (topology structure) between a sender and a receiver. However, we show this result to indicate that the SDSense retains the communication energy consumption trend, which is proportional to the number of transmissions and retransmission for successful packet delivery.

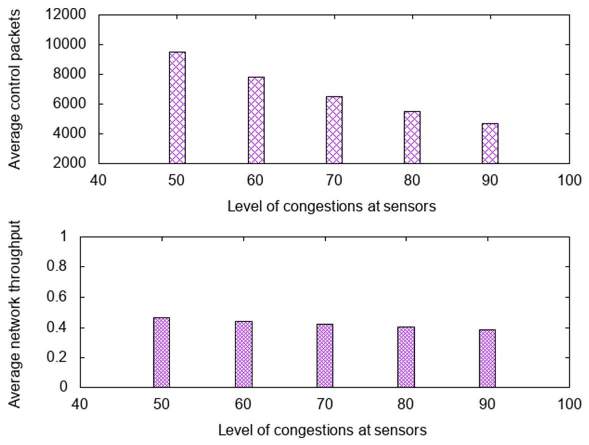


Fig. 10. The average number of control packets and corresponding throughput.

D. *The Impact of the Control Overhead*

We are able to more directly configure the network in reaction to the observed traffic conditions using the proposed SDSense. However, this requires additional communications to relay the network state from the data plane elements, and from the controller to reconfigure these elements. In contrast, the traditional local forwarding heuristics do not introduce any such control overhead. In this section, we characterize the observed overhead of the SDSense operation, which most of the existing solutions for software-defined WSNs overlooked.

Figure 10 shows the impact of the congestion alert threshold on the amount of control traffic and corresponding network throughput as the threshold is varied from 50% to 90% (in terms of queue utilization). The lower the threshold, the more reactive the network is, but the higher the overhead. The average number of control packets decreases 51% as the threshold is increased from 50% to 90%, which is shown in the top sub-figure. However, in the bottom sub-figure, it is shown that the overall network throughput is 17% lower due to the slower reaction to the presence of congestion. Thus, we can select an appropriate threshold value to balance between the control overhead and the achieved throughput.

VII. Conclusions

In this work, we have proposed an SDN-based event-driven WSN framework called SDSense that delegates control tasks among logically centralized and local controllers based on a principled design approach. Our principled approach decomposes the design solutions into two components: (1) A topology-control, scheduling and baseline rate allocation component that target the fixed or slowly changing components of the network; and (2) A congestion avoidance and rate reallocation component that react to the dynamic behavior of the network in a principled way, based on a NUM framework. For topology control, we have proposed a generalized algorithm to construct k-edge connected routing topology for WSNs, which supports any spanning structures. We have evaluated a set of k-edge connected topologies in terms of their reliability in the presence of link failure. We have evaluated the performance of different topologies with respect to the quality of the available paths and resilience to failure. Once we have identified the topology, we have carried out a baseline capacity allocation. We assume a TDMA network to reduce collisions and provide a more predictable performance.

The second component of the framework is targeted towards managing dynamic and emergent behavior. To manage congestion, we have first explored local heuristics that spread the traffic at congested nodes across multiple paths. However, these simple local strategies are not guaranteed to find the best paths through the network and may be susceptible to oscillation. In addition, given that the TDMA schedule requires a centralized solution, they cannot reallocate the available bandwidth or the routing topology. As a result, we have defined a global NUM problem that jointly optimized rate allocation and routing considering the demand of sensors. This optimal rate allocation is then implemented using the SDSense framework. In SDSense, network functions like routing, topology and congestion control are deployed in a logically centralized controller as interacting but isolated modules. The architecture also incorporated local controllers at the sensors to collect local information and implement any distributed functionality. We have evaluated the proposed framework under a number of scenarios. We show that it provides substantial improvement to network operations, especially when congestion arises.

REFERENCES

[1] M. Casado, M. Freedman, J. Pettit, J. Luo, N. McKeown, and S. Shenker, “Ethane: Taking control of the enterprise,” *SIGCOMM Comput. Commun. Rev.*, vol. 37, no. 4, pp. 1–12, Aug. 2007.

[2] M. Casado *et al.*, “Rethinking enterprise network control,” *IEEE/ACM Trans. Netw.*, vol. 17, no. 4, pp. 1270–1283, Aug. 2009.

[3] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, “Wireless sensor networks: A survey,” *Comput. Netw.*, vol. 38, no. 4, pp. 393–422, Mar. 2002.

[4] A. Mainwaring, D. Culler, J. Polastre, R. Szewczyk, and J. Anderson, “Wireless sensor networks for habitat monitoring,” in *Proc. 1st ACM Int. Workshop Wireless Sensor Netw. Appl.*, 2002, pp. 88–97.

[5] K. D. Kim and P. R. Kumar, “An overview and some challenges in cyber-physical systems,” *J. Indian Inst. Sci.*, vol. 93, no. 3, pp. 341–352, Jun. 2013.

[6] L. Galluccio, S. Milardo, G. Morabito, and S. Palazzo, “SDN-WISE: Design, prototyping and experimentation of a stateful SDN solution for wireless sensor networks,” in *Proc. IEEE Int. Conf. Comput. Commun.*, Apr. 2015, pp. 513–521.

[7] L. Galluccio, S. Milardo, G. Morabito, and S. Palazzo, “SDN-WISE: Design prototyping and experimentation of a stateful SDN solution for wireless sensor networks,” in *Proc. IEEE Int. Conf. Comp. Commun. (INFOCOM)*, Hong Kong, Apr. 2015.

[8] M. Baddeley, R. Nejabati, G. Oikonomou, M. Sooriyabandara, and D. Simeonidou, “Evolving SDN for low-power IoT networks,” in *Proc. 4th IEEE Int. Conf. Netw. Softwarization*, 2018, pp. 71–79.

[9] K. Gabriel and R. Sokal, “A new statistical approach to geographic variation analysis,” *Systematic Zoology*, vol. 18, pp. 259–278, 1969.

[10] I. Haque and N. Abu-Ghazaleh, “Wireless software defined networking: A survey and taxonomy,” *IEEE Commun. Surv. Tut.*, vol. 18, no. 4, pp. 2713–2737, Oct.–Dec. 2016.

[11] D. Kreutz, F. M. V. Ramos, P. E. Verssimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, “Software defined networking: A comprehensive survey,” *Proc. IEEE*, vol. 103, no. 1, pp. 14–76, Jan. 2015.

[12] S. Costanzo, L. Galluccio, G. Morabito, and S. Palazzo, “Software defined wireless networks: Unbridling SDNs,” in *Proc. Eur. Workshop Softw. Defined Netw.*, Oct. 2012, pp. 1–6.

[13] P. Dio *et al.*, “Exploiting state information to support QoS in software-defined WSNs,” in *Proc. Mediterranean Ad Hoc Netw. Workshop*, 2016, pp. 1–7.

[14] H. Fotouhi, M. Vahabi, A. Ray, and M. Björkman, “SDN-TAP: An SDN-based traffic aware protocol for wireless sensor networks,” in *Proc. 18th Int. Conf. e-Health Netw., Appl. Services*, Sep. 2016.

[15] G. Li, S. Guo, and Y. Yang, “Traffic load minimization in software defined wireless sensor networks,” *IEEE Internet Things J.*, vol. 5, no. 3, pp. 1370–1378, Jan. 2018.

[16] A. Brandt *et al.*, “RPL: IPv6 routing protocol for low-power and lossy networks,” IETF RFC 6550, 2012.

[17] A. D. Gante, M. Aslan, and A. Matrawy, “Smart wireless sensor network management based on software-defined networking,” in *Proc. 27th Biennial Symp. Commun.*, Jun. 2014, pp. 71–75.

[18] J. Wang *et al.*, “A software defined network routing in wireless multihop network,” *J. Netw. Comput. Appl.*, vol. 85, no. C, pp. 76–83, May 2017.

[19] S. Tomovic and I. Radusinovic, “Performance analysis of a new SDN-based WSN architecture,” in *Proc. 23rd Telecommun. Forum Telfor*, Nov. 2015, pp. 99–102.

[20] D. Zeng, P. Li, S. Guo, T. Miyazaki, J. Hu, and Y. Xiang, “Energy minimization in multi-task software-defined sensor networks,” *IEEE Trans. Comput.*, vol. 64, no. 11, pp. 3128–3139, Jul. 2015.

[21] R. Friedman *et al.*, “An architecture for SDN based sensor networks,” in *Proc. 18th Int. Conf. Distrib. Comput. Netw.*, 2017, pp. 20:1–20:10.

[22] T. Luo, H. Tan, and T. Quek, “Sensor OpenFlow: Enabling software-defined wireless sensor networks,” *IEEE Commun. Lett.*, vol. 16, no. 11, pp. 1896–1899, Nov. 2012.

[23] Y. Choi, Y. Choi, and Y.-G. Hong, “Study on coupling of software-defined networking and wireless sensor networks,” in *Proc. 8th Int. Conf. Ubiquitous Future Netw.*, 2016, pp. 900–902.

[24] Z. Guan, L. Bertizzolo, E. Demirors, and T. Melodia, “WNOS: An optimization-based wireless network operating system,” in *Proc. IEEE 19th Int. Symp. Mobile Ad Hoc Netw. Comput.*, 2018.

[25] H. I. Kobo, A. M. Abu-Mahfouz, and G. P. Hancke, “A survey on software-defined wireless sensor networks: Challenges and design requirements,” *IEEE Access*, vol. 5, pp. 1872–1899, Feb. 2017.

[26] A. Zahmatkesh and T. Kunz, “Software defined multihop wireless networks: Promises and challenges,” *J. Commun. Netw.*, vol. 19, no. 6, pp. 546–554, Dec. 2017.

[27] M. Chiang, S. H. Low, A. R. Calderbank, and J. C. Doyle, “Layering as optimization decomposition: A mathematical theory of network architectures,” *Proc. IEEE*, vol. 95, no. 1, pp. 255–312, Jan. 2007.

[28] P. Gupta and P. R. Kumar, “The capacity of wireless networks,” *IEEE Trans. Inf. Theory*, vol. 46, no. 2, pp. 388–404, Mar. 2000.

[29] L. Chen, S. H. Low, M. Chiang, and J. C. Doyle, “Cross-layer congestion control, routing and scheduling design in ad hoc wireless networks,” in *Proc. 25th IEEE Int. Conf. Comput. Commun.*, Apr. 2006, pp. 1–13.

[30] S. S. Kulkarni and L. Wang, “MNP: Multihop network reprogramming service for sensor networks,” in *Proc. 25th IEEE Int. Conf. Distrib. Comput. Syst.*, 2005, pp. 7–16.

[31] I. Haque, M. Islam, and J. Harms, “On selecting a reliable topology in wireless sensor networks,” in *Proc. IEEE Int. Conf. Commun.*, 2015, pp. 6376–6382.

[32] N. Li, J. C. Hou, and L. Sha, “Design and analysis of an MST-based topology control algorithm,” in *Proc. 22nd Annu. Joint Conf. IEEE Comput. Commun. Soc.*, Mar. 2003, pp. 1702–1712.

[33] K. Alzoubi, X. Li, Y. Wang, P. Wan, and O. Frieder, “Geometric spanners for wireless ad hoc networks,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 14, no. 4, pp. 408–421, Apr. 2003.

[34] R. Rajaraman, “Topology control and routing in ad hoc networks: A survey,” *ACM SIGACT News*, vol. 33, no. 2, pp. 60–73, 2002.

[35] I. Stojmenovic and X. Lin, “Power-aware localized routing in wireless networks,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 12, no. 11, pp. 1122–1133, Nov. 2001.

[36] S. Ergen and V. Pravin, “TDMA scheduling algorithms for wireless sensor networks,” *Wireless Net.*, vol. 16, no. 4, pp. 985–997, May 2010.

[37] S. Tilak, N. B. Abu-Ghazaleh, and W. Heinzelman, “A taxonomy of wireless micro-sensor network models,” *ACM SIGMOBILE Mobile Comput. Commun. Rev.*, vol. 6, no. 2, pp. 28–36, 2002.

[38] Y. Sankarasubramaniam, O. Akan, and I. Akyildiz, “ESRT: Event-to-sink reliable transport in wireless sensor networks,” in *Proc. 4th ACM Int. Symp. Mobile Ad Hoc Netw. Comput.*, 2003, pp. 177–188.

[39] M. Mitzenmacher, “The power of two choices in randomized load balancing,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 12, no. 10, pp. 1094–1104, Oct. 2001.

Authors’ photographs and biographies not available at the time of publication.