DeSSR: a Decentralized, Broadcast-based Scalable Scheduling Reservation Protocol for 6TiSCH Networks

Kaushal Kumar, and Mario Kolberg, Senior *Member, IEEE*

***Abstract*—The emergence of IPv6 (Internet Protocol Version 6) for low-power wireless communication is considered a breakthrough allowing a densely populated multi-hop network of Internet of Things (IoT) devices to be used for data gathering over a range of 1-2 kilometer (km). However, the communication between the devices has suffered from external interferences and multi-path fading challenge. The Internet Engineering Task Force (IETF) and Institute of Electrical and Electronics Engineers (IEEE) jointly proposed The IPv6 over IEEE 802.15.4 TSCH mode (6TiSCH) to deal with existing challenges and improve network performance to meet key requirements of industrial applications. The 6Top layer integrates TSCH (Time Slotted Channel Hopping)-MAC over IEEE 802.15.4 with the rest of the IPv6 stack where the schedule allocation is performed by scheduling function (SF). However, network scalability remains an open challenge. Specifically, the 6TiSCH Working Group (WG) do not define rules towards optimal schedule allocation over Time Slotted Channel Hopping (TSCH) mode of IEEE 802.15.4. In this paper, we propose Decentralized, and Broadcast-based Scalable Scheduling Reservation Protocol for 6TiSCH Networks (DeSSR). The experimental performance analysis demonstrates strong performance under steady and bursty traffic when compared with current SFs. This makes DeSSR a strong proposal contributing towards improving scalability in large-scale 6TiSCH networks.**

***Index Terms*— Internet of Things (IoT), IEEE 802.15.4 Networks, 6LoWPAN, 6TiSCH Architecture, TSCH Scheduling.**

I. INTRODUCTION

IoT is a network of interconnected resource-constrained devices exchanging data simultaneously over the internet. Today, billions of IoT devices are connected to the Internet world-wide [1]. Consequently, new standards and technologies for low power and lightweight communication have emerged to match the wired-like connectivity for industrial coverage [2]. The 6TiSCH is a wireless communication standard, introduced by IETF 6TiSCH WGs in 2013, with the objective to enhance IPv6 operation using IEEE 802.15.4 specification [3]. It uses a compressed IPv6-enabled 6LoWPAN stack [4] where the TSCH mode [5] has been already added to IEEE 802.15.4 for countering the impact of external interferences, and to deal with the multi-path fading issue using its channel-hopping capability.

The 6Top layer [6] holds the central position integrating the rest of the IPv6 stack to the TSCH-MAC over IEEE 802.15.4. This led to improved reliability. However, 6TiSCH installations using a densely populated multi-hop Destination Oriented Direct Acyclic Graph (DoDAG) topology [7] have suffered from poor scalability. The actual bottleneck is the collisions in TSCH slotframe [5].

Centralized SFs suffer from scalability limitations due to high signaling overheads and their use has sharply declined after the introduction of distributed scheduling as per the literature [8].

Distributed scheduling allows both *negotiation-based* and *autonomous* scheduling operation [8]. Under negotiation-led scheduling, it incurs a cost of reoccurring negotiations between nodes and their neighbors to adapt to changing traffic conditions. Furthermore, the risk of collisions cannot be overlooked in a densely occupied slotframe.

Autonomous scheduling avoids the requirement to negotiate. Instead, it assigns the all-time active cells in a static manner. However, autonomous SFs perform poorly in changing traffic conditions in dynamic topologies causing a temporary peak of traffic. Clearly, the use of negotiation-led designs cannot be ruled out.

Currently, the negation-based scheduling has evolved over time from reactive to on-demand reservation-based bandwidth allocation, albeit there is no guidance provided by the 6TiSCH WG. The key algorithms under this category are documented in existing literatures [8]-[13].

Because the reoccurring 6Top transactions are lengthy and charge-consuming and the adaptive negotiation-based SFs use a significant amount of control overheads. The use of the *piggyback* technique to ensure an overhead-free operation is proposed by the adaptive autonomous SFs [14]. That is, no separate Enhanced Beacon (EB) for negotiation. However, there is no evidence of such proposals to be beneficial in terms of scalability. Furthermore, the combined use of autonomous and negotiation-based scheduling [9] does not guarantee scalability for larger networks due to poor propagation and inefficient adaption.

Decentralized, Broadcast-based Scheduling (DeBraS) [15] is an efficient design that is allowing more than one broadcast cells to improve propagation of information in the network, hence promoting seamless connectivity of nodes, and avoiding network bottlenecks caused by collisions. The selection of how many cells will be sufficient is left unto the implementer [15]. The design of DeBraS has led to significant

Manuscript received February 13, 2023; accepted November 17, 2023. Date of publication xx xx, xxxx; date of current version xx xx, xxxx.

Kaushal Kumar is with Computing Science and Mathematics – Division, University of Stirling, Stirling FK9 4LA, U.K. (e-mail: kaushal.kumar@stir.ac.uk).

Mario Kolberg is with Computing Science and Mathematics – Division, University of Stirling, Stirling FK9 4LA, U.K. (e-mail: mario.kolberg@ stir.ac.uk).

improvements during the network bootstrap period and the scale of operation, where the key trade-off has been with the charge consumption [15]. Previously, we developed Smart Scheduling Reservation (SSR), which is a distributed solution allowing the 6Top layer to hold a central position while the sensor devices are negotiating with their neighbors for bandwidth. It offered good scalability and achieved strong distribution [12], However, due to limited proliferation of information, channel-hopping became slower, which led to decline in the performance beyond 70 nodes under heavy traffic conditions. Here, we aim to exploit a decentralized, broadcast-based scheduling design to improve scalability.

A. Contribution

In this article, we review SSR's key strategies for DeBraS-led operation. The outcomes of the review include the need for enhancement of Dynamic Traffic Strategy (DTS) and Network Depth Strategy (NDS). We also replaced the Cell Selection Strategy (CSS) with a randomized selection. The following contributions are reported in this paper:

- We propose DeSSR, which enhances the NDS to increase participation by nodes and reassess their distance from the root in a flexible manner. This was aimed to promote competition among nodes to access excess TSCH cells.
- We enhance DTS using PDR (Packet Delivery Ratio) and that is necessary to choose the best eligible node for allocation. This process does not introduce additional overheads.
- DeSSR is evaluated using a densely populated large-scale network. In particular the impact of *steady*, and *bursty* traffic experiments on network performance over time is investigated.
- We test the scalability of DeSSR considering increased coverage, n*etwork size*, *buffer size*, and *packet generation periods.*

The remainder of this paper is divided into five main sections: Section I provides an introduction of the proposed study. This is followed by a review of related work, which also discusses SSR in a detailed manner in Section II. Section III presents DeSSR. The evaluation is conducted using the 6TiSCH simulator in Section IV where results are analyzed using a variety of scenarios. Finally, Section V concludes our work and outlines future directions.

II. Related work

6TiSCH scheduling is a popular topic that links to the evolution of lightweight 6TiSCH architecture for industrial networks using heterogeneous sensor devices [16]. This section reviews the related work in the context of decentralized, and broadcast-based scheduling, and discusses our previously defined approach, SSR, in a detailed manner.

A. Literature Review

This section reviews the key scheduling algorithms, dedicated towards the scalability of 6TiSCH networks. Municio et al. [15] proposed a lock-based decentralized approach, which floods the network with the information about the reservation of the locked cells to avoid conflict of interest with the other nodes in the network. However, it overlooks collisions. In addition, it causes high convergence delays due to trade-off with cell consumption, and it does not screen unrealistic links.

Accettura et al. [17] proposed DeTAS (Decentralized Traffic Aware Scheduling), which is an extension to TASA (Traffic-Aware Scheduling Approach). The approach forms a common schedule where nodes use multipath reservation to improve reliability on a shared medium. However, it causes a high volume of overheads.

Palattella et al. [18] proposed a fixed threshold-based scheduling, which allowed adding and deleting the portion of cells on demand basis and based on the custom-defined limit. However, it under-or-over estimates the demand of cells per node since the nodes frequently change their position in the network hierarchy.

Soua et al. [19] fragmented the slotframe into waves where each time a sender node transmits a packet, a new pattern of wave with unique channel id is made available to control collisions. Here, it remains a valid solution as long as the nodes do not transmit packets simultaneously and the packets are not dropped due to the scarcity of *Tx* cells similar to unicast-based TSCH scheduling operation.

Raza et al. [20] proposed a decentralized, adaptive multi-hop scheduling protocol for 6TiSCH wireless network. It uses a data-centric query flooding the network to ensure that the traffic is forwarded in advance. Evaluation confirms that it is reliable and efficient for mobile-friendly SF, However, the query process to reach the nodes located far away from the root is subject to high overheads.

Kralevska et al. [21] proposed a decentralized, multipath schedule reservation protocol using the graph theory. However, the author ignores bandwidth allocation to traffic variation. In the extension to LV (Local Voting) [22], the author used the fixed threshold of 10 cells. However, neither one is scalable as the pre-estimated load sharing provides no benefit for dynamic topology as nodes appear and disappear frequently. Additionally, it witnesses high *rank-churn* and poor *parent-change,* and uses unreliable, and poor links, which may be present in the routing topology [8].

Duaquennoy et al. propose *Orchestra* [23], which is the first autonomous SF using all-time active slots configured on a per node basis using *receiver-based* and *sender-based* allocation modes. Orchestra allocates one cell per node per slotframe. Unfortunately, the approach does not adjust well to variable traffic conditions. Further, shortcomings include a comparably high latency and low scalability.

Autonomous Link-based Cell Scheduling (ALICE) [24] is considered an enhanced version of Orchestra, which replaces Orchestra's node-based allocation with a link-based scheduling approach. Hence, it offers a volume of cells to a node depending on the number of RPL (Routing Protocol for Low Power and Lossy Networks) neighbors. However, it too adopts a static allocation of the autonomous cells, which is not optimal considering the unpredictability of a node's movement in the topology [14]. Consequently, ALICE suffers from packet loss under heavy traffic conditions and for network sizes beyond *64* nodes as is discussed in [14]. Thus, it lacks the scalability required for industrial applications where hundreds of nodes are deployed under a single root node.

ALICE-FP [14] is an extension of ALICE that allows nodes to exchange their *frame pending* bit to allocate more cells to the corresponding node that carries more traffic from the sending node. The proposal employed *piggybacked* technique and is overhead-free. However, it lacks adaption despite assisting nodes which experience peaks of traffic.

The *piggybacked* technique is further exploited by Traffic-Aware Elastic Slotframe Adjustment (TESLA) [25]. It uses the *receiver-based* mode of Orchestra and uses adjustments in the slotframe based on incoming traffic. However, this is not an overhead-free approach. Recently, autonomous scheduling has been utilized alongside the negotiation-based approaches using 6top layer or 6P protocol [26].

Chang et al. [27] proposed Minimal Scheduling Function (MSF), which uses Orchestra for maintaining network dynamics and On the Fly bandwidth allocation (OTF) for data traffic adaption [28]. OTF's integration with the MSF has made it vulnerable due to fixed threshold-based allocation, which under-or-over estimates the demand and leads to bandwidth wastage [13]. OTF [18] has been extended by numerous SFs and inherited key drawbacks of fixed threshold-based overprovisioning. Righetti et al. [29] proposed an extension of OTF called E-OTF, in which the author used a similar representation of bandwidth allocation policy like OTF except it uses signaling to measure slot occupancy and provide congestion bonus as real-time queue occupation threshold. Wang et al. [30] highlighted key drawbacks of E-OTF, as poorly defined controls towards optimal scheduling for bursty traffic, and insufficient measurement of occupancy threshold. Furthermore, E-OTF's performance under large-scale network is currently unknown [30].

SSR [12] proposed a unique *cake-slicing* based distribution. It outperformed a range of popular SFs including MSF with consideration to traffic adaption. However, SSR experienced a decline in performance beyond *70* nodes under heavy traffic conditions (packet period of *1s*). The next section introduces SSR and its key concepts.

B. Smart Scheduling Reservation

This SSR [12] is a distributed scheduling solution, designed to improve scalability of 6TiSCH industrial networks using an analytical technique called '*cake-slicing'* along with the four peer strategies: NDS, DTS, CSS, and Queue Optimization Strategy (QoS) or Packet Aggregation Strategy (PAS). This section briefly discussed these capabilities of SSR using examples.

S	D		Hop 1	Hop 2	Hop 3	Hop 4	Hop 5	Hop 6	Hop 7	Hop 8	Hop 9
100	2	=	75	25	X	X	X	X	X	X	X
100	3	=	67	22	11	X	X	X	X	X	X
100	4	=	63	21	10	6	X	X	X	X	X
100	5	=	60	20	10	6	4	X	X	X	X
100	6	=	59	19	10	6	4	2	X	X	X
100	7	=	58	19	9	5	4	3	2	X	X
100	8	=	56	19	9	6	4	3	2	1	X
100	9	=	56	18	9	5	4	3	2	2	1

Fig. 1. Example of the cake-slicing algorithm

Figure 1 demonstrates a distribution of slotframe (S= 100) against varying depth identifiers (D= 2,3,4,5,6,7,8,9) using the cake-slicing technique of SSR. The output is presented on a per row basis, which corresponds to D. The columns represent a forecast of slotframe distribution for a multi-hop hierarchical network topology, which is divided into a number of hops based on the distance to root (controller). According to the example in Figure 1, a non-linear pattern is observed with a difference in values between hop 1 and hop 2 that is almost 3 times that of values at hop 2 and it is true for the rest of the 9 cases presented in this example. This translates into the throughput capability of nodes depending on the distance to root. The cross (X) sign in Figure 1 is used as a filler in the example showing D is less than 9 hops while S and N are non-zero. A fuller discussion on the cake-slicing algorithm is provided in [12].

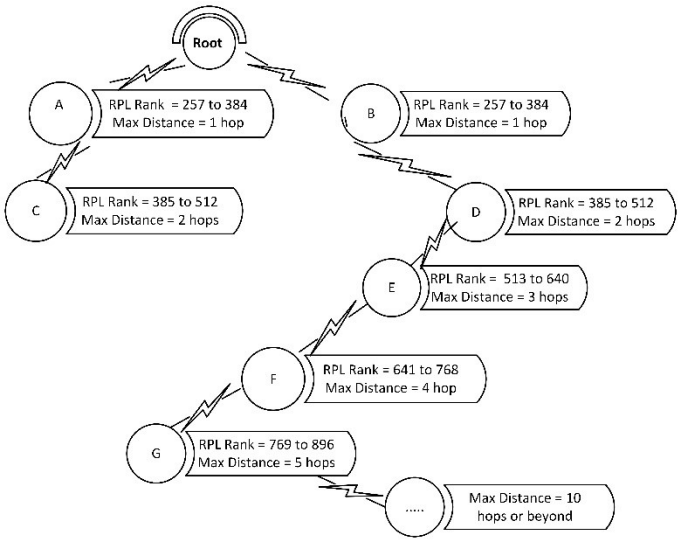

Fig. 2. Example of SSR's Network Depth Strategy.

Figure 2 captures the view of how NDS functions with an example. Here, the nodes are numbered alphabetically and the rank identifier of RPL [7] is used to calculate distance to root, which eventually translates into a number of hops. SSR uses a fixed interval of 127, which is added to the corresponding rank value to reorganize the topology. In Figure 2, Node A and B are located at closest distance to root at hop 1. This means, any node whose rank coordinates fall between 257 and 384 will be considered one-hop away from the root. Nodes C and D are considered to be two-hops away. Similarly, node E, F, and G follow hop 3, 4, and 5 respectively. However, where the *rank* exceeds the 10 hops, a random hop id is assigned.

Figure 3 presents an example of DTS using a representative cake-slicing scenario from Figure 1, and the example of NDS in Figure 2, which also corresponds to Figure 1. Here, the example of DTS uses a particular row from Figure 1 where S= 100 and N=D= 5 and Figure 2 provides hop id of nodes in the network topology. In Figure 3, the root node is located at the bottom. The nodes that are used for data gathering are aligned on the left-hand side, and the computation of DTS threshold (T_{sd}) takes place on the right-hand side. To compute T_{sd}, DTS uses a bitwise operator (>>) between the slice (value) chosen based on the hop distance and the constant of 3. Thus, Node

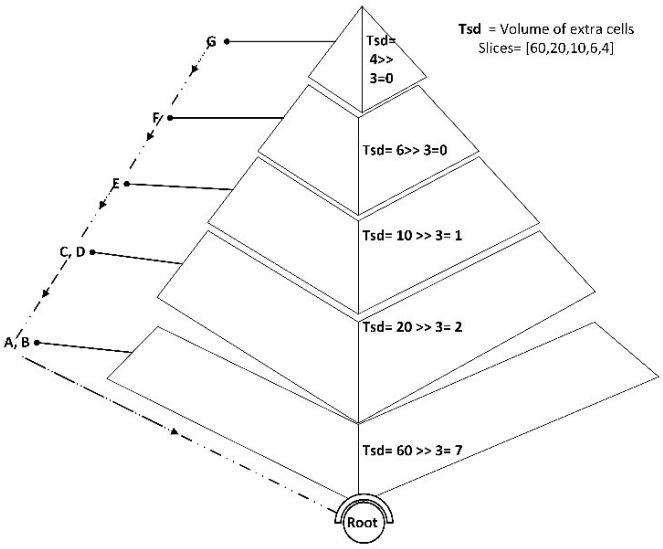

Fig. 3. Example of SSR's DTS using 5-hop RPL topology.

A, and B get T_{sd}= 7, which is the highest estimate. Node C and D get T_{sd}= 2 each, that is about 3.5 times less than the value of T_{sd} at hop 1 and is as per the distribution shown in Fig. 1. Finally, Node E, F, and G at hop 3 get T_{sd}= 1, T_{sd} = 0, T_{sd} = 0 respectively. With these heuristics being supplied on the run-time, SSR replaces static distribution of excess cells in an on-the-fly manner.

The original representation of DTS is available in [12]. For the influence of DTS on network performance, refer to performance evaluation of SSR, which is provided in [12].

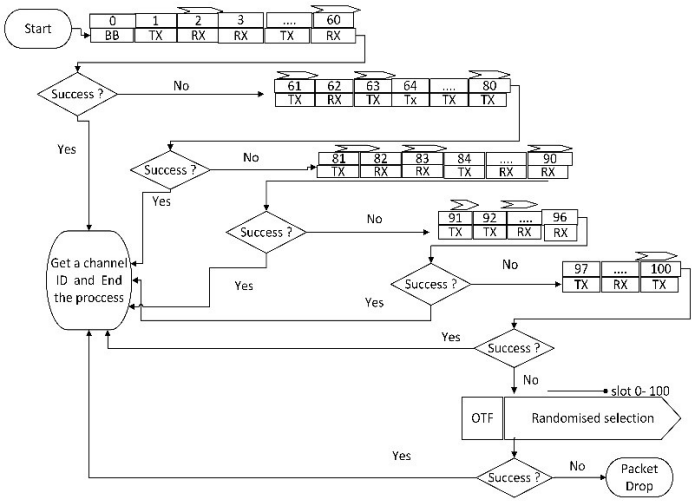

Fig. 4. Example of SSR's CSS using 5 hop RPL topology.

Cell Selection using SSR's CSS closely follows the cake-slicing distribution. In Figure 4, it is shown using Figure 1 where S= 100, and D= 5. CSS treats each corresponding slice as a slotframe, which is then scanned thoroughly by nodes for free cells until the demand of requested cells is met.

For example, the 1st slice contains 60 timeslots, Hence, it contains 0- 60 slots where the 0th slot is a broadcast schedule, used for bootstrapping.

The next slice is 20 slots long as per the distribution shown in Figure 1, hence, the 2nd slotframe starts from 61 and ends with 80. That is, exactly 20 slots.

The rest are shown in Figure 4 where the scheduler allocates most slots to the left in the slotframe under normal traffic conditions. However, if the occupancy of a slotframe reaches 100%, the scheduler chronologically uses the next available slotframe and this process continues until the demand is met. However, if all slotframes are scanned and the occupancy is high, SSR opts for a randomized selection and drops packets upon failure to locate the sufficient volume of free cells.

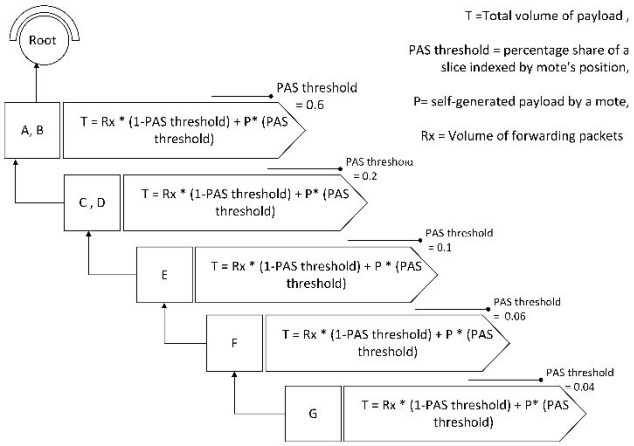

Fig. 5. Example of SSR's PAS.

Figure 5 Demonstrates an example of PAS using slicing hysteresis from Figure 1, where we retain a list of values from a particular row (60, 20, 10, 6 and 4) as slicing hysteresis.

SSR uses this hysteresis to distinguish self-generated traffic from forwarding traffic. The PAS further uses an example to show how it functions. Here, *P* indicates the node's self-generated payload, and *Rx* indicates the forwarding payload. *PAS* threshold is computed as the percentage of corresponding slice indexed by the node's position (hop id).

In the example given in Figure 5, each tab is divided into two parts where the left-hand side of it shows node ID as prefix, hierarchically organized in the topology in the same order as Figure 2, and the remaining part of the tab contains finalized transmission payload (T).

In the 1st tab, the T is calculated for Node A and B where both nodes take 40% of forwarding packets, which is calculated using *(Rx * (1- PAS threshold))* provided *PAS* threshold is a value between 0.00 and 1.00. The node's self-generated payload is collected to 60%, which is inversely proportional to share of the forwarding payload to be transmitted. Node C and D choose 80% of forwarding streams and 20% of self-generated streams. Node E chooses 90% of forwarding traffic and 10% of its own. Node F takes 94% of forwarding traffic and 6% of its own. Finally, Node G, which is located at the bottom of the hierarchy, chooses to take 96% of forwarding packets and 4% of its own. In the next section, we enhanced SSR to improve the scalability using a Decentralized, Broadcast-based scheduler.

III. DECENTRALIZED, AND BROADCAST-BASED SCALABLE SCHEDULING RESERVATION PROTOCOL

SSR follows a negotiation-led scheduling approach in which nodes allocate and deallocate cells using 6Top commands [6]. SSR follows a defined configuration where each and every node exchange EBs using *minimal cell* as per the RFC 8180 [31]. That is, in the large-scale network headed by a single-sink node, the probability of collision remains a key concern, not only in the *minimal cell,* which is also called *hard* cell (only configured once) but also in the *soft cells* that are allocated and deallocated on demand-basis [31]. The collision in *minimal cell* is handled using back-off mechanism provided in IEEE 802.15.4 and collided soft cells are relocated using 6Top commands [18]. Furthermore, with the high volume of soft cells collided, it can even lead to network collapse [18]. The triggering point is the conflict of interest with other nodes since the advertisement of real-time changes in the slotframe is expensive.

In this paper, we propose the enhancement of SSR using a decentralized, broadcast-based scheduling design, first introduced by Tinka et al. [32], and extended by Municio et al. [15]. According to this design, nodes advertise their reserved cells in the network through more than one broadcast cell. This improves propagation to lower the probability of collisions. The approach selects a minimum of 3 broadcast cells randomly.

As far as the topology formation is concerned, once the root is configured, it triggers transmission of EBs, and DoDAG information objects (DIO) messages to allow new nodes to join. A node, when switched on, initiates listening at a randomly chosen channel and receives EBs from nearby nodes, thereby discovering nodes in its surrounding. The joining node selects one of the neighbors as Join Proxy (JP) to which it synchronizes its clock to and starts the joining process [33]. A Join Proxy is an existing RPL neighbor sharing connectivity metrics with joining node [31]. Further information on how connectivity metrics are calculated by RPL is available in RFC 6550 [7]. The joining process then requires the node to follow the *secure join* operation [33]. Here, the joining node exchanges unicast messages with security information, carried within EBs. Once the node is authenticated, it sends DIS (DoDAG Information Solicitation) messages [31] to solicit DIOs from RPL neighbors. The node then deciphers the DIOs to acquire a preferred parent and rank information [7]. Lastly, the joined node must add at least 1 *Tx* cell with the corresponding parent using 6top unicast messages [43], which are also carried by EBs. Hence, network formation incurs a significant number of overheads.

The proposed solution follows the design principles of RPL RFC 6550 where *non-storing* mode is default mode of operation [7]. Here, multiple nodes send data to the root node via a preferred parent. The root node populates path information to the outgoing packet's header information leading to a downward trajectory to reach the corresponding destination in DoDAG topology, i.e., source routing. Hence, there is no requirement for a node in DoDAG to store an entire set of routing entries. Instead, a single entry to the corresponding Direct Acyclic Graph (DAG) parent is sufficient. This is important as the nodes are memory-constrained devices. The full set of routing entries is only stored by the root, which is also responsible for computing the shortest path to the destination. This mechanism is supported by Destination Advertisement Object (DAO) and Destination Advertisement Object-Acknowledgement (DAO-ACK) unicast messages, sent periodically by non-root nodes in the topology. The DAO messages contain downward routing entries and are marked valid upon a timely receipt of DAO-ACK [7]. If the destination is not found, then the packet is dropped. This mechanism is discussed in more detail in [7]. The following scheduling restrictions are imposed by DeSSR:

- *All source nodes send data to the root node via preferred parent selection and follow RPL's non-storing routing mode of operation [7].*
- *A sensor node can either transmit or receive the packet at a time hence it follows a half-duplex communication.*
- *Parents and children can transmit and receive a packet using the same TSCH cell.*
- *A Child node from a common parent can either transmit or receive packets simultaneously using the same time slot and channel id (cell).*

In this section, we propose DeSSR, and review the key strategies of SSR towards the decentralized, broadcast-based scheduling operation.

A. SSR, and Decentralized, Broadcast-based Scheduling

SSR anticipates requirements to monitor queue occupation and adapts traffic conditions on nodes (motes) dynamically. However, it allocates too many cells to nodes based on the NDS distribution and ignores poor links (nodes) on the way to root. Hence, further screening of nodes is necessary to avoid bandwidth wastage. Apart from that, SSR uses the fixed value of 127, which separates nodes from one another based on the hop-distance. The potential problem with this constant is that when nodes are allowed to use shared cells rapidly, the current limit of 127 undermines the node's participation.

SSR's CSS is not suitable for decentralized, and broadcast-based scheduling as the strategy demands adjacent *Tx* cells in the slotframe thereby, triggering a high volume of collided cells. Hence, the proposed solution must avoid using CSS. One option to use is the random selection method, However, it too does not promise a collision-free operation. As far as the packet aggregation is concerned, it uses a unique *strategy* based on the dynamicity of the *cake-slicing* heuristics. That is, the aggregation follows dynamic selection of payload in the node's buffer. Hence, it will be implemented without any modification. Similarly, the cake-slicing method.

B. Design principles of DeSSR

DeSSR is a distributed SF, which, on one hand, promotes increased negotiation between nodes using enhanced NDS, while eliminating poorly performing nodes to have extra cells on the other hand. It filters the nodes based on PDR: a value between 0.00 to 1.00. The proposed solution does not introduce additional overheads while integrating PDR-based screening on top of DTS. DeSSR allocates cells when the available cells are not enough for a node to complete the transmission of payload. The algorithm is shown in Figure 6.

DeSSR Bandwidth Allocation Algorithm

```
DeSSR Allocation (Now_cells, Req_cells)
Now_cells← Number of cells in node buffer
Req_cells← Number of cells needed to match traffic
T← threshold to calculated, S← slotframe length
rank ← RPL rank of a node, ND← network depth
Interval← [256,512,368….N.]
pdr← packet delivery ratio of a node.
hop ← calculate_rank (rank, Interval)
T← int ((get Slicer(S, hop) >> number of parents) * pdr)
 if  Req_cells> Now_cells AND Now_cells==0 then
     if Req_cells > 0 AND  (pdr/1.5)>0.5  then
          Now_cells= Req_cells-Now_cells + (T+1)/2
     end if
 else if Req_cells < (Now_cells - T)    then
     Now_cells = Now_cells- Req_cells- (T+1)/2
 else
     Now_cells= 0
 end if
End
```

Fig. 6. DeSSR's bandwidth allocation algorithm.

DeSSR-led scheduling indicates that only the deserving nodes are given excess cells ensuring the estimated PDR of eligible nodes is greater than 0.5. As far as the role of NDS is concerned, it provides a context to each node where the value of the *interval* is reset from 127 to 256 out of many available choices. This means, the higher the NDS interval, the higher the participation.

The algorithm begins with the *get Slicer* (cake-slicing) function, which generates a list of slices, indexed by node's hop id. The resultant value is processed using a *bitwise right-shift* operator against the *number of parents.*

Currently, the number of parents that a node can have been limited to 3 per DAG. But in mesh topology, this number can be even greater than 3. Here a greater value will get a lower dynamic threshold limit (*T*), and a lower number of children will get a high *T* value.

The end results are multiplied with *pdr* and that is how *T* is calculated for overprovisioning in DeSSR. The rest of the algorithm follows the same rules for ADD and REMOVE operation as SSR except an additional PDR-based screening while adding new cells.

IV. PERFORMANCE EVALUATION

DeSSR's performance is evaluated using the 6TiSCH simulator [33], which is a discreate event-driven simulator. It can be used to deploy and test large-scale networks and can predict network behavior accurately and realistically compared to mathematical models [33].

The simulator uses the algorithm provided by Pister et al. [34] for collision-detection. The energy-consumption is based on a realistic energy model, introduced by Vilajosana et al. [35] for calculating charge consumed during various radio activities. In their paper the authors provided the measurements used for *transmission*, *reception*, and *idle-listening.* The battery capacity is limited to *2200* milliampere-hours (mAh).

The TSCH slotframe is configured using 101 slots with the maximum duration of 10 milliseconds (ms) each and 16 channels. Hence each slotframe cycle lasts *1010*ms.

The broadcast probability of EB and DIO is set to 0.1, and 0.22 respectively. The routing beacons including DIOs and DAOs are sent per *1s* and *60s* respectively.

The experiment uses multiple simulation runs and for every run, it generates a new topology. The nodes are positioned randomly, and each one is connected to at least *3* RPL neighbors whose PDR (Packet Delivery Ratio) is expected to be about 0.5 or higher.

PDR is calculated based on *Received Signal Strength Indicator* (RSSI) metrics [36]. The qualifying RSSI threshold to allow packet reception is -97 *Decibel Milliwatts* (dBm).

Figure 7 captures a view of topology using the 6TiSCH simulator [33].

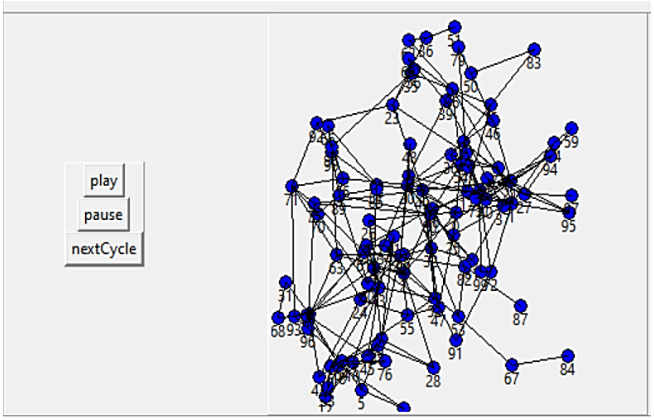

Fig. 7. A view of topology in 6TiSCH simulator.

This experiment is divided based on the *steady*, and *bursty* traffic pattern [21]. In the *steady* traffic scenario, the nodes will experience a continuous flow of packets generated periodically. In the *bursty* traffic scenario, a sample of burst consisting of a stream of packets will be injected given timestamps. In either scenario, packet's destination is the root alone. OTF [18], and LV [21] are selected for comparison based on the literature review. The remaining set of parameters are given in the corresponding tables.

A. Steady Traffic Experiment

The experiment uses the 6TiSCH simulator, which considers a single IPv6 subnet in which data is gathered continuously at a sample rate of 60 packets per minute. The payload is generated soon after the network is configured.

In the 6TiSCH network, nodes take time to join the network and the sooner a node is assigned rank, the sooner it starts transmitting path information to the root through DAOs. The data packets are transmitted using a multipath scenario depending on the transmission schedule that nodes have reserved with their preferred parent. The configuration parameters for this experiment are given in Table 1.

TABLE I
CONFIGURATION PARAMETERS

Parameters	**Value**
Nodes	[100]
Area	Square, [1*1] km
Housekeeping Period	[5]s
Packet Generation Interval	[1]s
Slot Duration	[10]ms
Channel Density	[16]
Slotframe Length	[101]
Buffer Size	[100] packets
NDS interval	[256]
Radio Sensitivity	[-97] dBm
Simulation cycles	[100]
Simulation runs	[100]
Confidence Interval	[95] percent
Number of broadcast cells	[3] cells
Number of RPL children	[3]
Broadcast probability of EBs	[0.1]
Broadcast probability of DIOs	[0.33]
DIO period	[1]s
DAO Period	[60]s

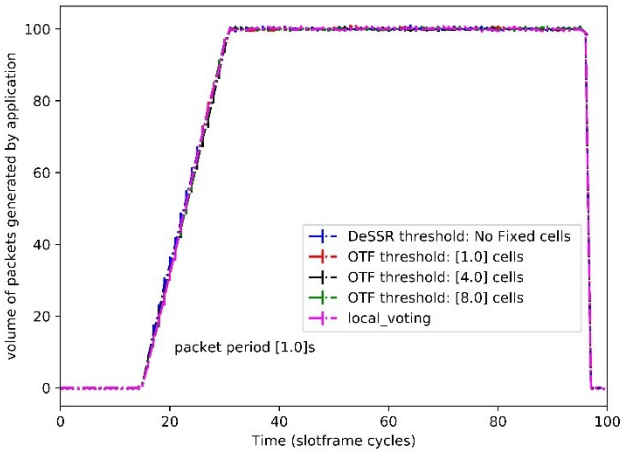

Fig. 8. Application-generated packets over time.

Figure 8 shows a steady packet generation scenario as soon as the *joining* is complete, just slightly ahead of slotframe cycle 20. After this, all SFs maintain a steady payload portion of 100 packets over time. as all nodes have joined the network (post configuration time).

Figure 9 depicts a stream of packets being sent to the root where number of packets are shown on the Y-axis and timestamps in slotframe cycles are given on the X-axis. The presented results in Figure 9 shows the throughput in volume of packets unstreamed by LV, OTF and DeSSR. Here, LV is showing variations as a difference between lower and upper mean values, computed at a 95% confidence interval. The variations are triggered due to scheduling incompetence against changing traffic conditions in dynamic topology. LV distributes the payload equally between nodes and disregards the fact that nodes closer to the root are responsible to send more packets than those at a farther distance from the root.

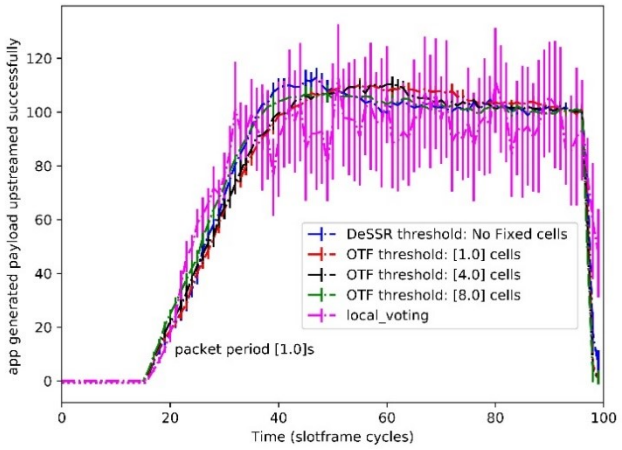

Fig. 9. Application payload up streamed successfully.

The rest of the SFs show progress with the evidence of DeSSR sending slightly more packets than OTF in the beginning.

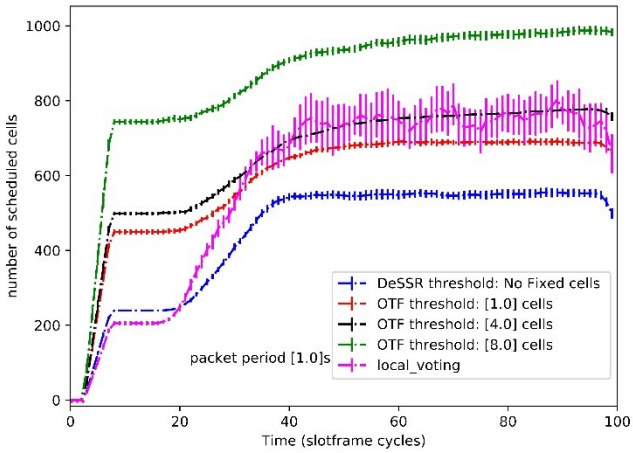

Fig. 10. Cell consumption in steady traffic flow over time.

Sensor nodes depend on the slotted medium to dispatch the payload. Figure 10 shows that OTF reserves more cells in advance where a higher threshold limit (8 cells) translates into the highest number of cells being scheduled. LV doesn't use overprovisioning, hence, the volume of scheduled cells by LV is lower than OTF. Albeit, both SFs scheduled significantly higher volumes of cells compared to DeSSR.

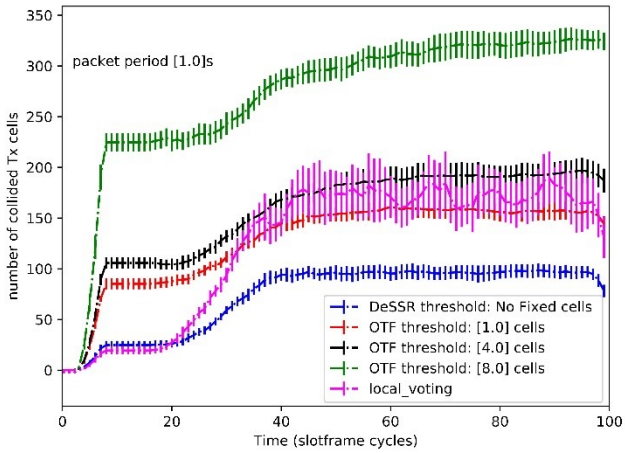

Fig. 11. Volume of collided cells over time.

In broadcast-based scheduling, collisions can be expected where the key to control is reduced cell consumption. It is evident through Figure 10 and Figure 11. Figure 11 reflects that high cell consumption causes an increased volume of collided cells and eventually triggers higher charge Consumption. Since DeSSR's consumption was the lowest, hence the collided cells triggered over time are the lowest too.

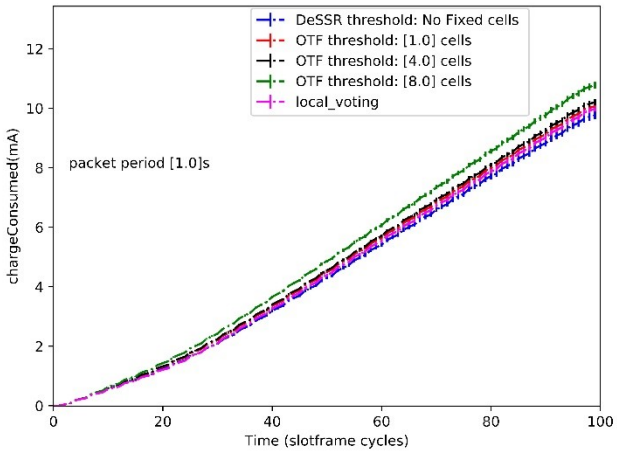

Fig. 12. Lowest charge consumption in mA over time.

Figure 12 shows that charge consumption by DeSSR is roughly the same as the others and this is due to the DeBraS scheduling itself, where the negotiations take place frequently, and beacons are dispatched frequently allowing more and more nodes to participate in negotiations with other nodes.

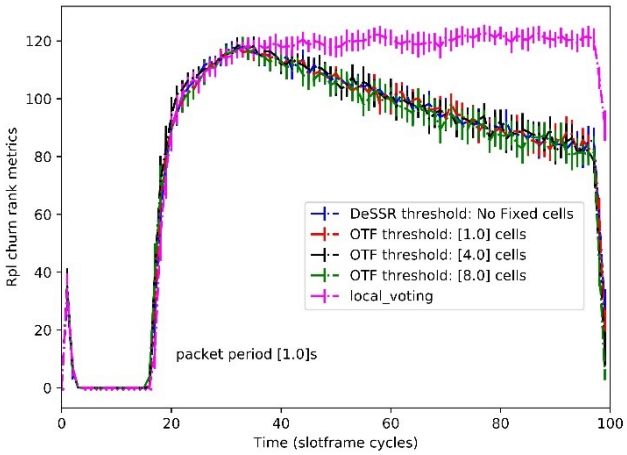

Fig. 13. RPL rank-churn over time.

In the dynamic topology, the node's rank increases for two reasons: (1) downgrading link quality, and (2) interference. The *rank* is a weight assigned by IETF RPL [7] for forming routing topology where nodes advertise their rank frequently in the network to probe the shortest path to the root. This process is called *preferred parent change* [7]. While the rank fluctuates, it triggers the additional number of 6Top cycles. Thus, a lower-bound in *rank-churn* is preferred to a stable network. In Figure 13, LV maintains a fairly steady portion *rank-churn* over time, which contrasts *rank-churn* patterns of OTF and DeSSR. The reason of rank oscillation by LV is poor bandwidth allocation, which also leads to many inconsistencies including poor Tx-buffer utilization.

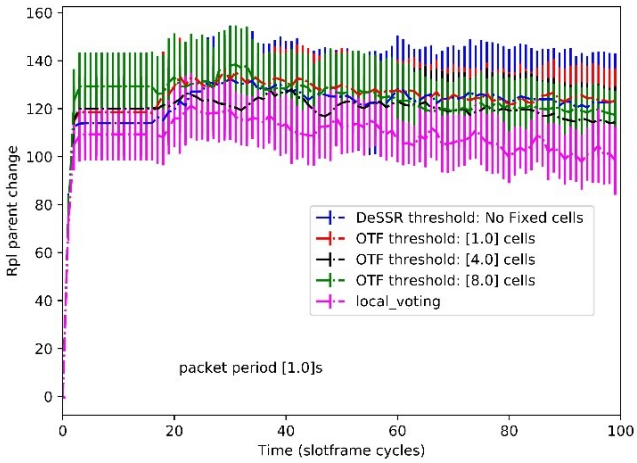

Fig. 14. Node-churn over time upper-bound is preferred.

Contrary to the *rank-churn*, the *node-churn* is the process that influences network performance positively if an upper-bound is followed. In DeBraS-led operation, the nodes from an uncommon parent can transmit or receive packets simultaneously. When high *node-churn* is evident, nodes are flexible to progress to the shortest path to root, which cuts down latency, improves utilization of cell, and balances the traffic load. However, if there are not enough cells provided to probe the shortest path, the *node-churn* becomes counterproductive. This is evident in Figure 14 where a lower-bound *node-churn* is followed by LV while OTF and DeSSR both exhibit a high *node-churn* over time.

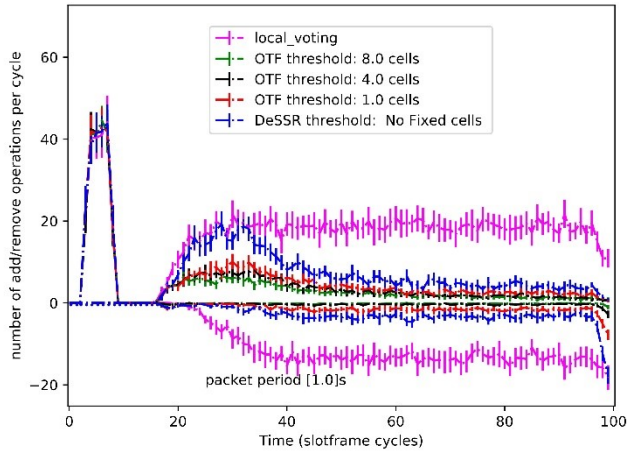

Fig. 15. 6Top add/remove operations per slotframe cycle.

In general, the SFs add and delete cells to nodes as a result of the changing traffic conditions. Figure 15 shows a volume of ADD/DELETE transactions per slotframe cycle (time), where ADD transactions are observed above the X-axis and DELETE transactions are observed below the X-axis. The results show that LV triggers ADD and DELETE transactions most frequently compared to OTF and DeSSR because it lacks overprovisioning. OTF, in order to suppress the recurrent transactions, allocates a fixed number of cells. However, OTF does not match the real-time demand, and that way, it under-or-overestimates the actual demand. This also means, while it underestimates the demand, a slightly more cycles of ADD and DELETE are scheduled. DeSSR adapts the demand based

on the slicing heuristics where nodes closer to the root are allowed to maintain a high throughput. Figure 15 shows that DeSSR triggers a balanced number of overheads, which is roughly the same as OTF despite the lowest consumption observed by DeSSR. That is necessary to balance the recurrent 6Top overheads for the improved network performance.

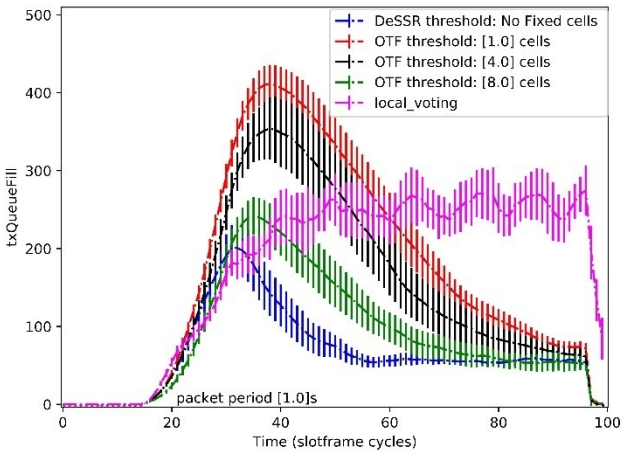

Fig. 16. Transmission buffer utilization over time.

Figure 16 shows how the Tx-buffer is managed by SFs over time based on the traffic adaption strategies. In this regard, LV maintains a roughly steady portion with a high volume of packets remains in the queue throughout the time. DeSSR and OTF follow a non-linear pattern where DeSSR keeps the lowest volume of packets in the queue to control congestion.

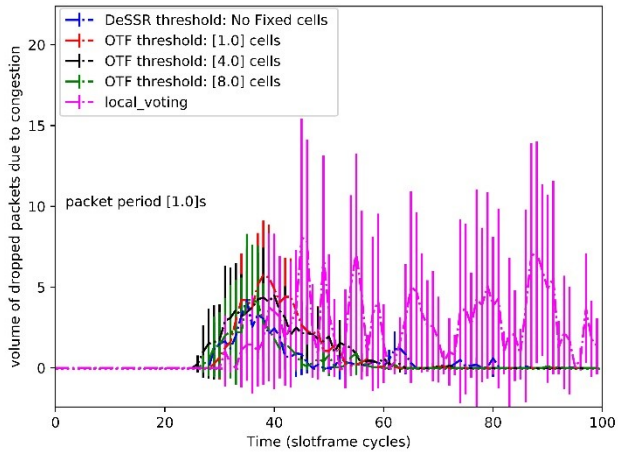

Fig. 17. Showing packet loss due to congestion over time.

Figure 17 depicts that LV triggers congestion throughout the time and incurs packet loss. DeSSR and OTF trigger congestion for a shorter time where DeSSR registers the lowest estimate of congestion to Tx-buffer ratio.

In Figure 18, DeSSR's latency is settling to the lowest over time, while OTF maintains three peaks depending on the multiple threshold limits. It shows that higher threshold causes reduced latency. Unlike OTF, DeSSR settles latency to *1s* over the time, which is optimized to the point that it is free from the trade-off with cell consumption. LV's latency remains largely between *2s* and *3s* and this kind of delay is not ideal for real-time operation given there are other technologies that are only rejected because of the high delay.

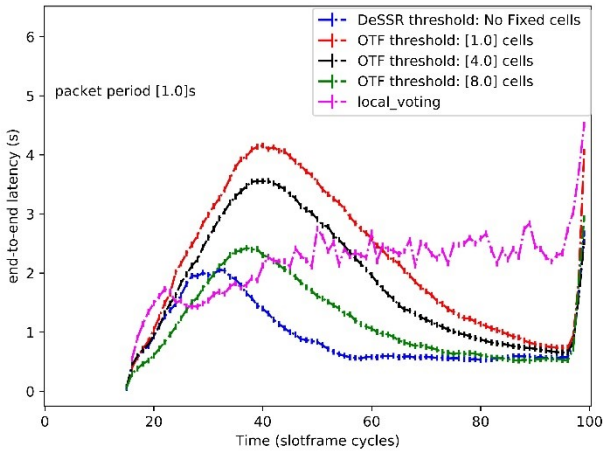

Fig. 18. Latency maintains a steady lower-bound.

B. Discussion

The *steady* experiment simulates the industrial deployment using the 6TiSCH standard where the performance of DeSSR was compared with LV and OTF using several key indicators. The results confirmed that DeSSR offers high throughput using the lowest volume of cells compared to other SFs. The charge consumption is roughly the same as others because of decentralized, and broadcast-based scheduling-led operations. Notably, DeSSR showed strong performance over time compared to the LV and OTF, and achieved lowest volume of collided cells, lowest latency, lowest congestion in queue, improved node-churn, and optimized Tx-buffer. However, it triggered a slightly high 6top transactions as these were necessary to adapt rapidly changing network dynamics.

C. Bursty Traffic Experiment

The *bursty* traffic experiment draws significance from the real-world industrial scenarios such as leak detection [37]. According to this, nodes experience a sudden gust of traffic. The experiment injects a sample of 25 packets per burst per node in a network of 100 nodes at fixed timestamps of 20 and 60 respectively in slotframe cycles. For analysis, the results are benchmarked with the OTF using multiple thresholds. The queue length for all nodes is 100 packets. The remaining configuration parameters are given in Table 2.

TABLE II
CONFIGURATION PARAMETERS

Parameters	**Value**
Nodes	[100]
Area	Square, [1* 1]km
Housekeeping Period	[5]s
Packet Generation Interval	[1]s
Slot Duration	[10]ms
Channel Density	[16]
Slotframe Length	[101]
Buffer Size	[100] packets
NDS interval	[256]
Radio Sensitivity	[-97] dBm
Simulation cycles	[100]
Simulation runs	[100]

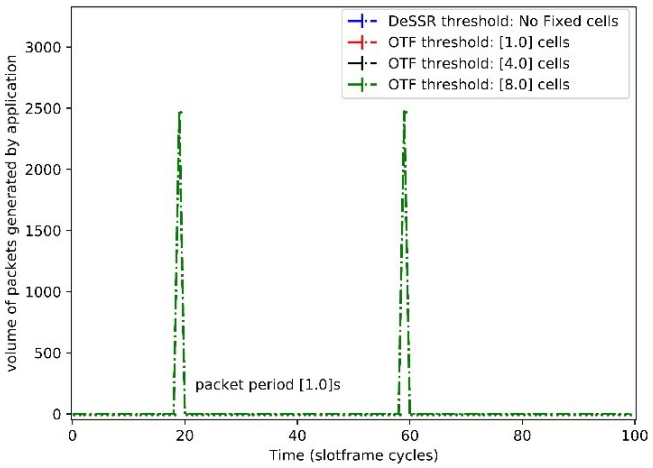

Fig. 19. Packets generated during traffic burst over time.

Figure 19 depicts the volume of traffic generated as per the bursty traffic conditions where each node generates roughly the 25 packets over the given time and this process repeats twice per slotframe run. Because a TSCH slotframe repeats itself over time hence for each repetition (run), two traffic bursts are supplied per run. Both SFs consistently generate 2475 packets precisely at slotframe cycle 20 and 60 respectively.

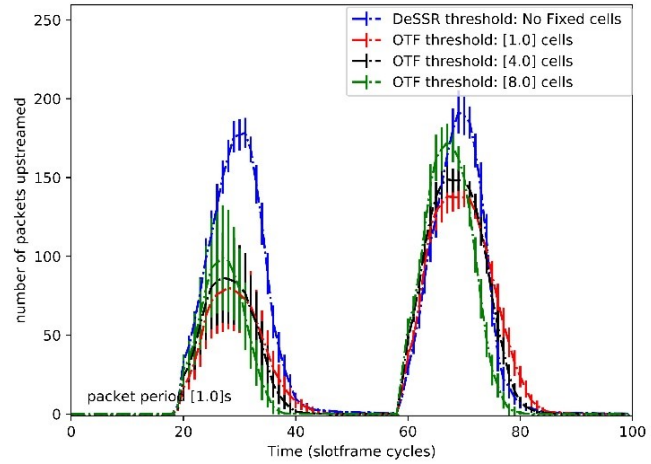

Fig. 20. Total number of packets up streamed to root.

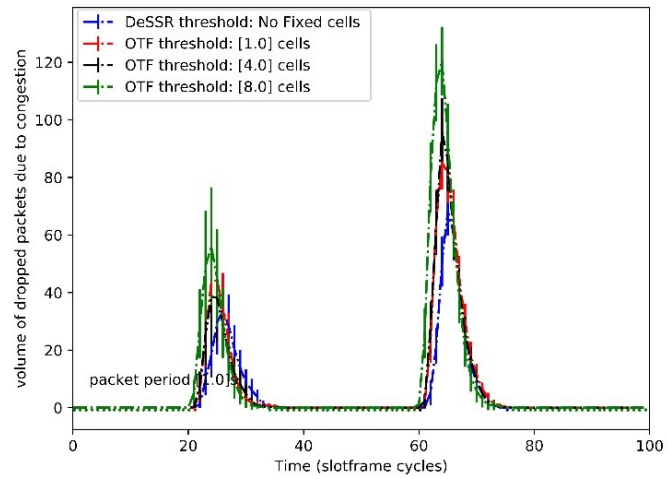

Fig. 21. Packet loss due to congestion in the queue.

Figure 20 shows packet transmission capability where OTF drops more packets during the 1st peak compared to 2nd peak. The reason is poor assessment of demand. On the other hand, DeSSR sends more packets than all three variants of OTF throughout time and it is evident in Figure 20.

In a sudden gust of heavy traffic, congestion can occur due to fixed queue capacity. Figure 21 shows that each peak incurs packet loss due to congestion where DeSSR's loss is the lowest considering a toll of 30 packets during 1st peak and 70 packets during the 2nd peak. OTF suffers the worst congestion with the highest volume of cells.

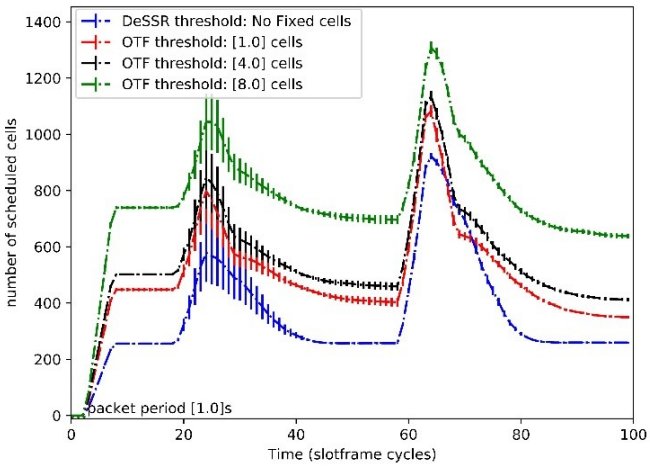

Fig. 22. Cell consumption in a bursty traffic scenario.

Figure 22 demonstrates the volume of scheduled cells by both SFs over the sudden arrival of a heavy payload. According to this, DeSSR schedules the lowest portion of cells on average and during both events despite there being no packets to send. Conversely, OTF follows a fixed distribution where the higher threshold limit translates into the high consumption of cells and eventually causes an increased collision.

The presented results in Figure 22 shows that DeSSR scheduled the lowest volume of cells where it uses scheduled cells efficiently to ensure reliable operation that is free from performance trade-offs.

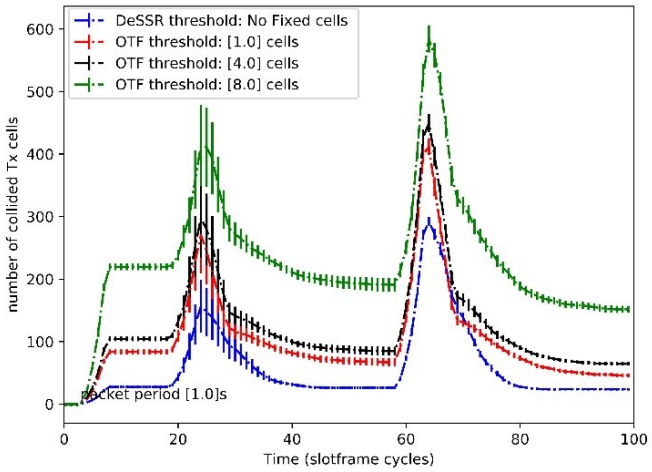

Fig. 23. Collided Tx cells as a function of time considering the bursty traffic scenario.

Figure 23 shows the volume of collided cells triggered by both SFs over time. Here, both SFs randomly select cells from the slotframe, and it is possible that two or more nodes are using the same Tx cells for transmission simultaneously under common parent. So, the collision can be expected. Apart from that, a higher consumption of cells can also influence collision to increase. In Figure 22, DeSSR observes the lowest cell consumption, hence the volume of collided cells triggered by DeSSR is the lowest too as shown in Figure 23, and this is true for both events, and for remaining times when there is no activity (idle times).

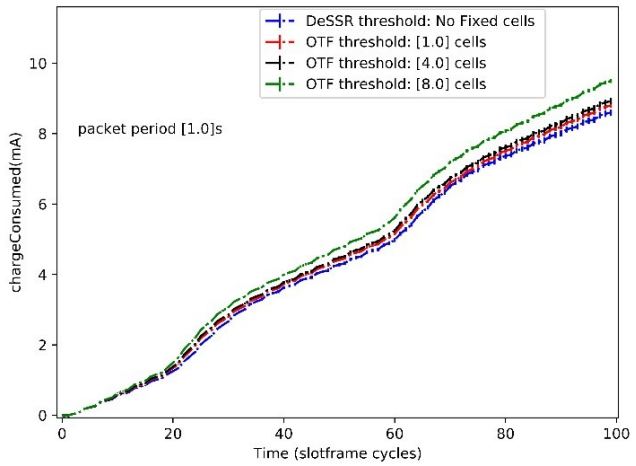

Fig. 24. Charge consumption(mA) over time in bursty traffic.

Figure 24 shows a charge consumption over time by OTF, and DeSSR. Here, both SFs consumed roughly the same amount of battery charge whereas the DeSSR consumed slightly less charge, however, the margin seems very thin. The key reason is the decentralized, broadcast-based scheduling itself where enhanced beacons take a toll of most charge consumed by both SFs. Charge consumption is further discussed in detail under Section IV (F) (1).

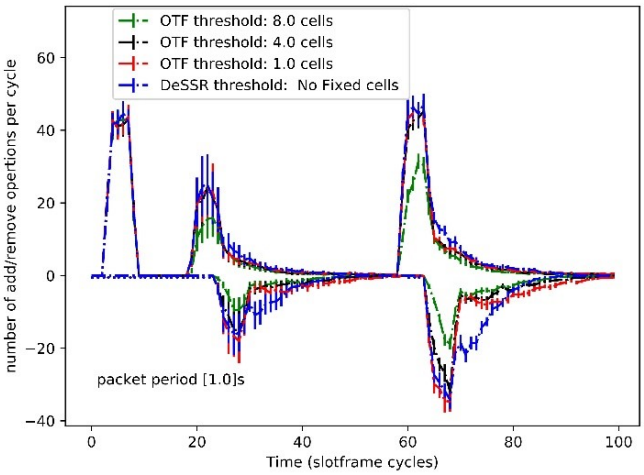

Fig. 25. Add/Remove 6Top operation per slotframe cycle.

Figure 25 shows 6Top ADD/DELETE activities over time where ADD activities are the positive number, shown above the X-axis and DELETE activities are the negative number, shown below the X-axis. In the beginning, the scheduler unanimously adds 40-45 cells to nodes towards bootstrapping. This process is the same for both *steady* and *bursty* traffic scenarios as can be compared with Figure 15. As the first event unfolds at slotframe cycle 20 in Figure 25, Both SFs generate marginally the same amounts of overheads except for the OTF with the highest threshold limit. The reason is obvious that each node is given 8 cells already by OTF and when nodes change position, OTF does not have to allocate or deallocate cells more often. On the contrary, DeSSR sends substantially more packets than OTF's high threshold using the lowest volume of cells and when the nodes change their parent frequently in search of closer parent to the root, it releases occupied cells and allocates new ones in an adaptive manner. The key benefit is that released cells are available for other nodes to use and this is why it causes additional ADD/DELETE transactions. The 2nd event at slotframe cycle 60 indicates the toll of 6Top ADD/DELETE transactions is higher than the 1st event. The key reason is that more cells are scheduled at this time by both SFs, as is shown in Figure 22.

D. Discussion

In the bursty traffic experiment, DeSSR outperformed OTF considering high throughput, lowest number of packet loss due to congestion, lowest scheduled cells, reduced collisions, lower charge consumption, and a fairly-balanced 6Top cycles. With these key achievements, DeSSR outperforms currently popular decentralized, broadcast-based scheduling functions and it achieved this without monitoring queue occupation. However, further experimentation is ideal to test scalability of DeSSR knowing that its packet transmission capability is higher than the most SFs.

E. Scalability Analysis

This experiment tests the scalability of DeSSR, and it is divided into three parts. In the first part, we test the reliability in medium-sized networks against extreme traffic load. In the 2nd part, the network size is increased to 100 nodes. Finally, in the 3rd part, we increased the buffer size. We have followed the steady pattern depending on the *packet period intervals*. The results are compared with multiple variants of the OTF. The configuration parameters are given in Table 3.

TABLE III
CONFIGURATION PARAMETERS

Parameters	**Value**
Nodes	[50,100]
Area	Square**, [**2*2]km
Housekeeping Period	[5]s
Packet Generation Interval	[1.0, 0.5, and 0.1] s
Slot Duration	[10]ms
Keep Alive Period	[10]ms
Channel Density	[16]
Slotframe Length	[101]
Buffer Size	[10,100] packets
NDS interval	[256]
Radio Sensitivity	[-97] dBm
Simulation cycles	[100]
Simulation runs	[100]

1) Scalability Analysis of Medium-sized Network
In this experiment, a total of 50 nodes are deployed randomly where queue length of all nodes is 10 packets [22]. The experiment produces a number of subplots reflecting the impact of varying traffic load over time where buffer size is explicitly mentioned.

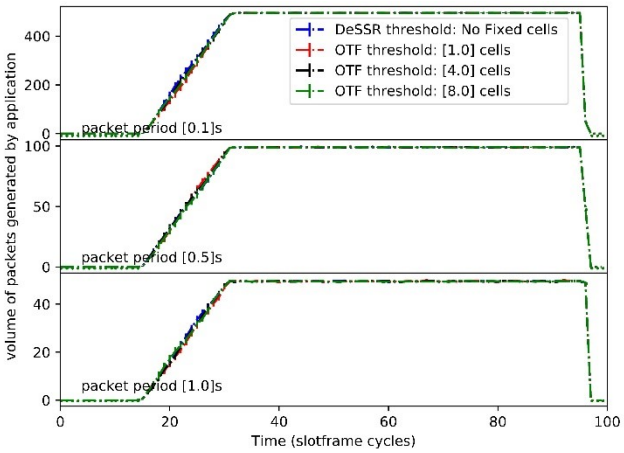

Fig. 26. Payload generation over time.

Figure 26 is divided into three subplots depending on the traffic load scenarios where each subplot observes steady distribution of traffic over time. That is, about 50 packets are generated per cycle under *packet period* 1s, 100 packets under *packet period* 0.5s, and 500 packets under *packet period* 0.1s. These estimates correspond to the total number of packets to be generated as per the given traffic rate.

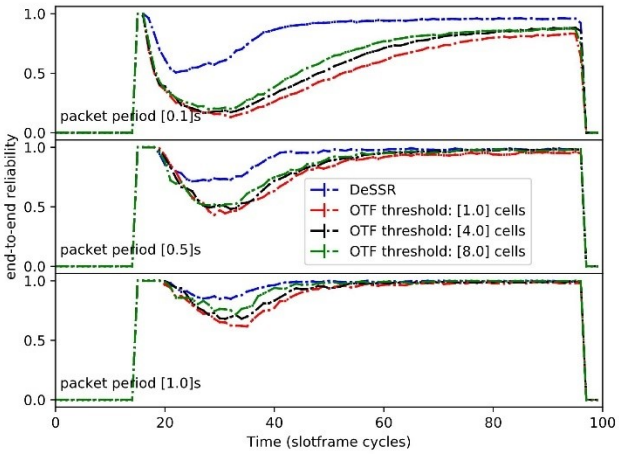

Fig. 27. Reliability for medium-range deployment.

Figure 27 shows that both OTF and DeSSR take a hit and the packet loss is evident for a shorter time between slotframe cycle 20 and 40 except that DeSSR's loss is less pronounced.
In the 3rd subplot of Figure 27 particularly, we observe intense load where DeSSR is capable of delivering a high volume of traffic load while OTF is challenged due to high consumption to collisions ratio. On the contrary, DeSSR drops packets due to constrained buffer size.

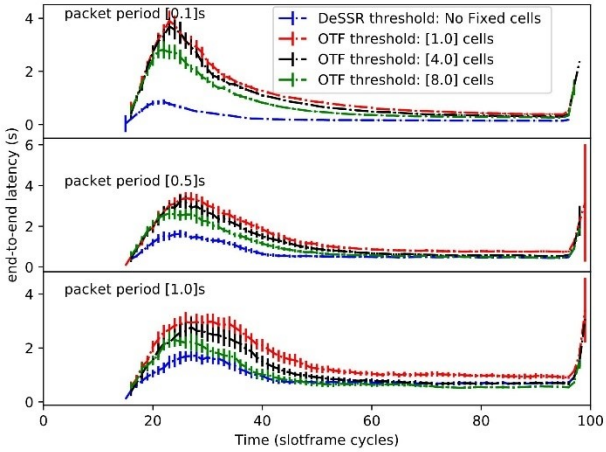

Fig. 28. Latency for medium-range deployment.

Figure 28 shows latency over time where DeSSR, at slotframe cycle 20 and 40, triggers the lowest delay of 2s provided not all packets are delivered at this stage. Thereafter, it remains fairly steady around 1s, and it is the same for OTF too. At this stage, most packets are up streamed successfully.
The 3rd subplot of Figure 28 tallies the gap between trajectories of OTF and DeSSR, at slotframe cycle 20 and 40, and the delay is yet the lowest among all trajectories of DeSSR, which is concerning. The reason is too many cells were given to nodes, which the nodes utilized to progress to shortest path and delivered a fairly less amount of payload.

2) Scalability Analysis of Large-scale Network
The experiment is used for testing scalability and robustness of DeSSR using 100 nodes. This time, the network is twice as dense and queue length is 10 packets for all nodes.

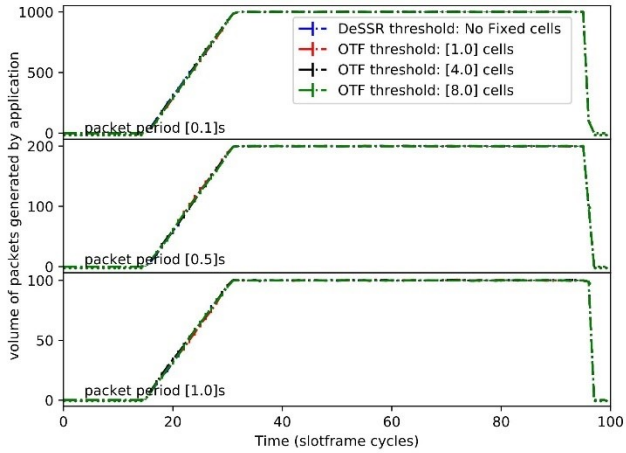

Fig. 29. Payload generation over time in large networks.

Figure 29 depicts application-generated payload over time where *100*, *200*, *1000* packets are generated for most of the time following the period interval 1s, 0.5s, and 0.1s respectively.

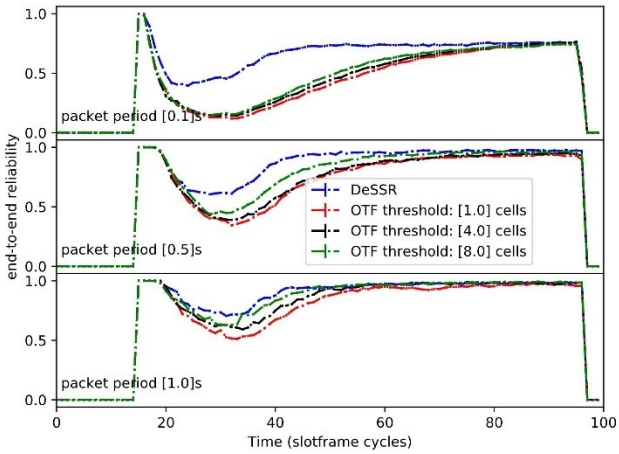

Fig. 30. End-to-End reliability as a function of time.

Figure 30 shows three consecutive subplots where reliability is shown on the Y-axis and time is given on the X-axis. Here, both SFs drop packets between slotframe cycle 20 and 60, due to congestion. However, a leading gap is observed, which monitors sharp declines in reliability of DeSSR and OTF as the load increases proportionately. Here, DeSSR still sends more packets than OTF and is more efficient in recovering from the packet loss compared to OTF.

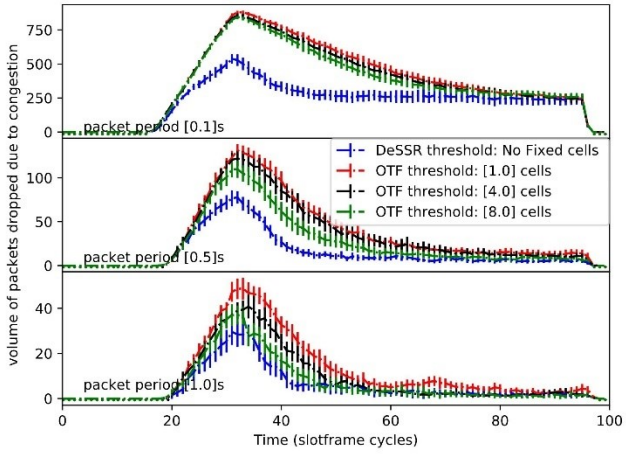

Fig. 31. Packet loss due to congestion in the buffer.

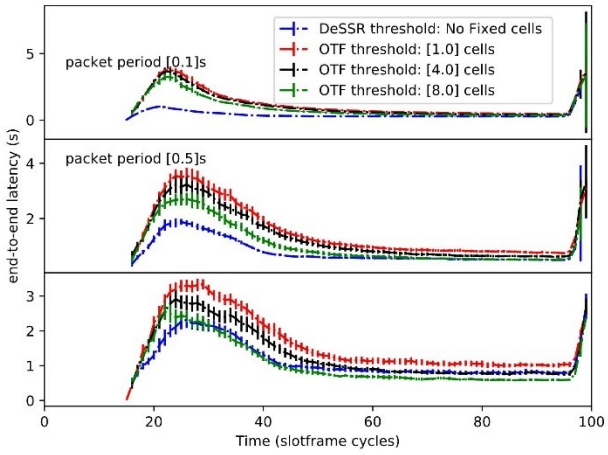

Fig. 32. Latency as a function of time, 100 nodes.

Figure 31 analyzes a backlog of packets that are lost due to congestion. The results show that congestion is the factor behind the loss incurred in Figure 30.
Figure 32 shows latency over time and the results are more or less the same as shown in Figure 28. Hence, it will not be accurate to predict the actual behavior.

3) Large-scale Network with Increased Queue Length
This experiment increases the queue length to 100 packets and uses a *packet period* of 0.5s. The presented results show the impact of congestion in the queue, end-to-end reliability, and latency. With previous approaches including OTF, congestion caused poor performance with OTF's performance significantly degrading during the temporary peak of traffic between slotframe timestamp 20 and 60. The results presented are averaged at a 95% confidence interval using 100 slotframe runs. This drop in performance is indicated by an increase in dropped packets leading to a reliability drop to about 75% and a latency of between 6 and 8s for OTF. DeSSR performs significantly better with reliability staying above 90% over time and latency not exceeding 4s.

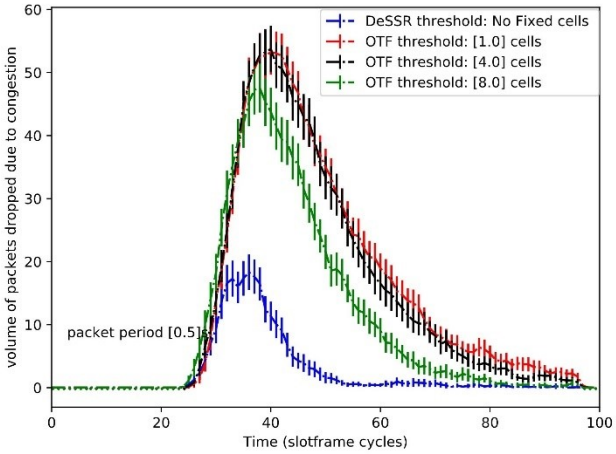

Fig. 33. Congestion in the extended buffer over time.

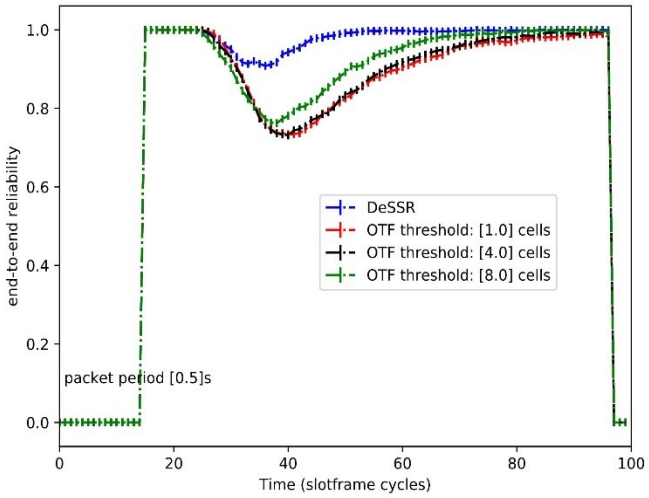

Fig. 34. Reliability over time and extended size of buffer.

Figure 33 shows that at an early stage (slotframe cycle 20 and 40), congestion cannot be ruled out despite an

increase in buffer size. Here, both SFs drop packets due to congestion, except DeSSR's loss is 75% less than OTF and it observes quick recovery over time.

Figure 34 shows that the extension to buffer size is rewarding for both SFs. However, both SFs drop packets in the beginning due to congestion, which negatively impacts the end-to-end reliability as is shown in Figure 34. Here, DeSSR observes the lowest decline in reliability and provides stability to the network fairly quickly compared to the OTF.

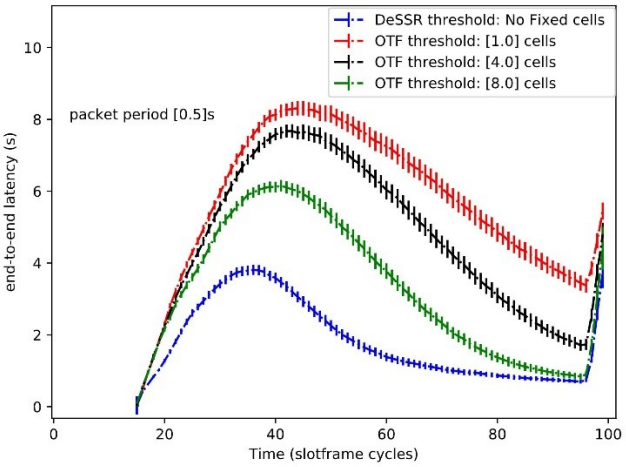

Fig. 35. Latency over using extended buffer scenarios.

Figure 35 shows end-to-end latency over time. Here, OTF registers a maximum delay depending on the threshold limits. That is, the higher the threshold limit, the lower the delay. That is because nodes have enough cells to probe the shortest path to the root. Results show that OTF's trajectories take longer to reach optimal latency of 1s while the DeSSR achieves the target in a fairly shorter time, and yet observes the lowest delay comparatively.

F. Discussion

This section discusses the performance of DeSSR in terms of side-effects considering charge-consumption, complexity, and scalability.

1) Charge Consumption

In 6TiSCH network, charge consumption is impacted by a number of aspects including the cost of network formation, dynamics of SF, and management of costs incurred due to propagation and control overheads.

Network bootstrap is an expensive period in low-power and lossy networks, involving frequent EBs carrying broadcast and unicast traffic. In a fully configured DoDAG topology, nodes periodically generate EBs ensuring nodes are synchronized [33]. Hence, an optimal broadcast strategy is useful to reduce energy consumption. Vucinic et al. [38] studied various broadcast strategies to minimize the delay in network formation by setting an optimal point to control portions of EBs without ignoring the convergence delay and collision. Therefore, all broadcast messages are carried in a form of *slotted aloha* [31]. The author also proposed an optimal threshold (0.1 and 0.33 for EB and DIO), which is representative of the lowest network formation time in the network of 45 nodes considering reduction in the volume of EBs [33]. Municio et al. [33] showed the delay in network formation increases in a steady-linear fashion as the network size increases using the same hysteresis. DeSSR uses the same value as described as optimal in [38] and demonstrates a shorter bootstrap period using more than one broadcast cell from the TSCH slotframe. The remaining portion of the slotframe is left unused for the SFs to implement.

In 6TiSCH network, nodes compute their radio duty-cycle per slot depending on how long it takes to finish the scheduled task. A variation in charge-consumption is likely as different SFs take different approaches adapting and managing the traffic in the network. Daneel et al. [13] highlighted the role of SF carries to prevent recurrent wastage of charge, triggered by poor overprovisioning and static allocation of cells, i.e., a mismatch between the actual traffic and predicted traffic. No defined mechanism so far has been proven optimal. Hence, charge consumption is dependent on the priorities of different SFs [18] (refer to Section III for review of SFs).

In a dynamic topology, nodes change parents frequently. This has a considerable impact on the underlying TSCH links [15]. That is, each parent-changing node must relocate its resources from one parent to another parent. Hence charge consumption varies from the point of how and to what extent the movement of non-root nodes is controlled. For this reason, many recently introduced SFs do not allow leaf nodes to have a Tx cell. However, this negatively affects the optimal path formation in RPL routing. Thus, constraining parent-change is not only a greedy setup but also has repercussions on overall scalability of the network. Our proposal inherits some of the key drawbacks of using multiple broadcast cells for advertisement [15]. However, it permits non-leaf nodes to have access to Tx cells. To analyze the impact of broadcast cells on charge consumption, a separate experiment is conducted using the same configuration parameters as used in Table 2, except the range of broadcast is set to be 1- 8 cells.

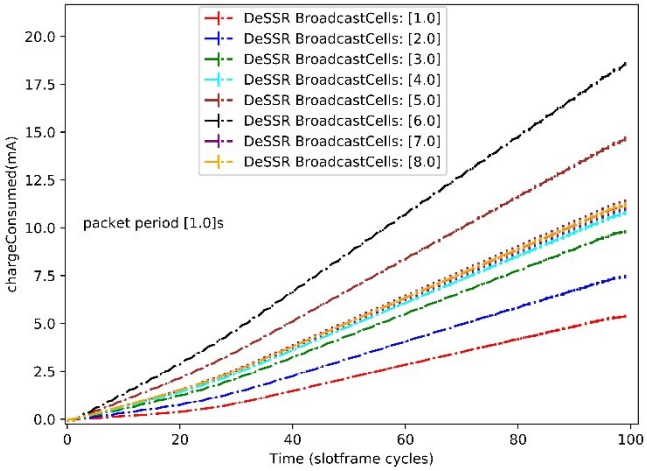

Fig. 36. Charge consumed by the varying broadcast cells.

Figure 36 illustrates the cost of managing scheduling dynamics indicating that the lowest number of broadcast cells consume the lowest amount of charge. However, this is not true with the highest portion comprising 7 – 8 cells with the presented results showing a moderate charge-consumption that increases over time and is about the same level as with 3- 4 cells. The highest amount of charge is consumed with 6 and 5 broadcast cells. Municio et al.[15] argues the key reason for this behavior being the increased waiting time for contention-access in large-scale networks. Hence, the charge-consumption decreases in larger networks. DeSSR retains the same number of broadcast cells as used by other algorithms.

2) Complexity

The complexity of DeSSR is analyzed in terms of control overheads (6Top transactions). This is further illustrated in Figure 15 where key SFs are shown to have added and deleted negotiated cells over time. The exchange is facilitated using 6Top unicast transactions, which take a longer time to execute and are resource intensive. Hence, the fewer number of cells are used the better. DeSSR manages complexity of control overheads by allowing extra cells to be reserved for the nodes closer to the sink and ensuring a strong PDR. However, this is not an absolute allocation. For nodes which change parent, the resources attached are only diverted to the new parent depending on the criteria provided by DeSSR. Hence the requirement is managed dynamically. This is evident in the analysis presented in Section IV(A), and (C).

The complexity of EBs overshadows the complexity due to control overheads. This is shown under Figure 12 where no significant variations are present in charge consumption by DeSSR, LV and OTF, with DeSSR observing the lowest cell consumption, lowest collision, and roughly same complexity level as OTF. This suggests the complexity of DeSSR in terms of propagation is comparable to the other SFs.

DeSSR does not add new overheads, instead, it allows a high availability of collision-free cells without causing trade-offs with latency and reliability. It achieves improved scalability allowing hundreds of nodes to use spare cells under a single DoDAG tree.

The flow of RPL control messages is unaffected by DeSSR's scheduling approach.

3) Scalability

We studied performance degradation factors in large-scale networks using multiple simulation-based experiments, and benchmarked results of DeSSR against other SFs in Section IV (E).

The selection of SFs was made as per the literature review presented in Section II (A). A number of recently published SFs such ALICE, and MSF, do not qualify for this comparison due to poor propagation, and limited throughput [14].

The significance of the contribution by DeSSR is presented using the following points:

- Improved availability of Tx cells at both steady and bursty traffic conditions. (Figure 10 and Figure 22 in Section IV).
- Improved throughput despite using the lowest portion of cells. (Figure 20, and 27).
- Reduced collision among Tx cells so that extra cells can be allocated to nodes experiencing a temporary peak. (Figure 11 and Figure 12).
- DeSSR's complexity in terms of 6Top overheads is comparable with OTF and is significantly lower than LV. (Figure 15).
- There are no trade-offs involved with latency and charge consumption. (Figure 9, Figure 16 and Figure 18).

Further experiments are carried out to test scalability using several hundred nodes. E-OTF's performance, as per the literature review, has been unknown for larger networks. Hence, DeSSR and E-OTF are included in the experiment to test performance considering the scalability being the prominent concern. The E-OTF is configured with no extra cells and the rest of the configuration parameters are the same as shown in Table 2 except the network density is increased to several hundred nodes.

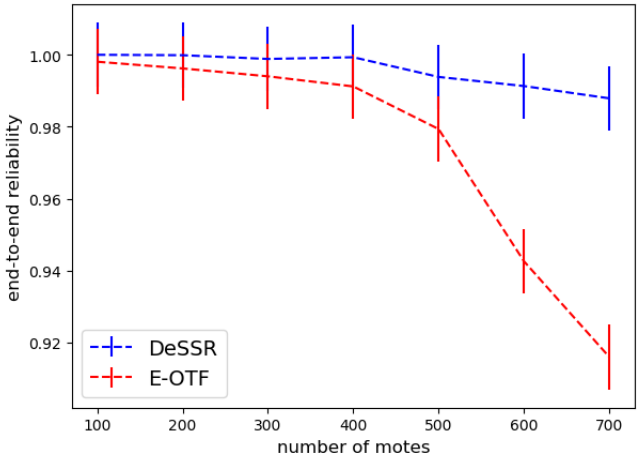

Fig. 37. Scalability of DeSSR over several hundred nodes using 60 packets per minutes.

Figure 37 highlights the contribution DeSSR makes for large-scale networks of several hundreds of nodes in size. Figure 37 shows DeSSR's reliability to be at least 99% even for networks up to 700 nodes (95% confidence interval). The results indicate a drop of performance of E-OTF beyond 400 nodes. This makes DeSSR a strong candidate for managing large-scale networks without ignoring the fundamental need of reliable and scalable operation.

V. Conclusion

This article presents DeSSR following the aim to improve scalability of IEEE802.15.4e networks. The work of DeSSR implemented TCSH-MAC mode in a decentralized and broadcast-based operation. The key contribution is provided through PDR-based DTS and using flexible NDS assessment.

The performance of DeSSR was extensively tested using *steady* and *bursty* traffic experiments in a large-scale network of 100 nodes under packet generation period of *1s*. Results showed that DeSSR outperformed both LV & OTF and achieved best results in steady traffic experiment. In a bursty traffic experiment, DeSSR's performance was superior to multiple versions of OTF thresholds. Towards scalability, DeSSR got tested with varying buffer sizes and varying traffic conditions. Results showed, in both medium-scale and large-scale networks, DeSSR maintains a significant lead over OTF. Finally, the scalability test was conducted using large networks up to 700 nodes. The results show that DeSSR significantly outperforms E-OTF for larger networks.

REFERENCES

[1] Ericsson, "Connected Industries," September 2020. [Online]. Available: https://www.ericsson.com/assets/local/internet-of-things/docs/connected-industries-a-guide-to-enterprise-digital-transformation-success.pdf.

[2] M. Kuzlu, M. Pipattanasomporn and S. Rahman, "Review of communication technologies for smart homes/building applications," 2015 IEEE Innovative Smart Grid Technologies - Asia (ISGT ASIA), 2015, pp. 1-6, doi: 10.1109/ISGT-Asia.2015.7437036.

[3] D. Dujovne, T. Watteyne, X. Vilajosana and P. Thubert, "6TiSCH: deterministic IP-ena-bled industrial internet (of things)," IEEE Comm. Mag., vol. 52, no. 12, pp. 36-41, 2014.

[4] E. Kim, D. Kaspar and J. Vasseur, "Design and Application Spaces for IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs)," Document RFC 6568, Internet Engineering Task Force, 2012.

[5] T. Watteyne, M. R. Palattella, L.A. Grieco. Using IEEE802.15.4e Time-Slotted Channel Hopping((TSCH) in the Internet of Things (IoT): Problem Statement. Internet Engineering Task Force, IETF, 2015.

[6] Q. Wang, X. Vilajosana and T. Watteyne, "6TiSCH Operation Sublayer (6top)," 04 06 2014. [Online]. Available: https://tools.ietf.org/html/draft-wang-6tisch-6top-sublayer-01. [Accessed 07 03 2020].

[7] P. Thubert, "ROLL: Internet Draft," IETF ROLL, 11 06 2012. [Online]. Available: https://tools.ietf.org/html/draft-thubert-roll-asymlink-02. [Accessed 1 10 2019].

[8] R. T. Hermeto, G. Antoine and T. Fabrice, "Scheduling for IEEE802.15.4-TSCH and slow channel hopping MAC in low power industrial wireless networks: A survey," Computer Communication, vol. 114, pp. 84-105, 2017.

[9] F. Righetti, C. Vallati, S. K. Das and G. Anastasi, "Analysis of Distributed and Autonomous Scheduling Functions for 6TiSCH," IEEE Access, vol. 8, pp. 158243-158262, 2020

[10] A. R. Urke, Ø. Kure and K. Øvsthus, "A Survey of 802.15.4 TSCH Schedulers for a Standardized Industrial Internet of Things," Sensors, vol. 22, no. 1, 2022.

[11] S. Hammoudi, A. Bentaleb, S. Harous and Z. Aliouat, "Scheduling in IEEE 802.15.4e Time Slotted Channel Hopping: A Survey," 2020 11th IEEE Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON), 2020, pp. 0331-0336, doi: 10.1109/UEMCON51285.2020.9298043.

[12] K. Kumar and M. Kolberg, "Improving Scalability of 6TiSCH Networks using Smart Scheduling Reservation", in IEEE Sensor journal, Under review, 2022.

[13] G. Daneel, B. Spinnewyn, S. Latre and J. Famaey, "ReSF: recurrent Low-Latency scheduling in IEEE 802.15.4e TSCH networks," Ad-Hoc Networks, vol. 69, pp. 80-88, 2018.

[14] F. Righetti, C. Vallati, A. Gavioli and G. Anastasi, "Performance Evaluation of Adaptive Autonomous Scheduling Functions for 6TiSCH Networks," in *IEEE Access*, vol. 9, pp. 127576-127594, 2021, doi: 10.1109/ACCESS.2021.3112266.

[15] E. Municio and S. Latre, "Decentralized broadcast-based scheduling for dense multihop TSCH networks," in MobiArch '16 Proceedings of the Workshop on Mobility in the Evolving Internet Architecture, NY, USA, 2016.

[16] M. Vucinic, M. Pejanovic-Djurisic and T. Watteyne, "SODA: 6TiSCH Open Data Action," 2018 IEEE Workshop on Benchmarking Cyber-Physical Networks and Systems (CPSBench), 2018, pp. 42-46, doi: 10.1109/CPSBench.2018.00014

[17] N. Accettura, E. Vogli, M. R. Palattella, L. A. Grieco, G. Boggia and M. Dohler, "Decentralized Traffic Aware Scheduling in 6TiSCH Networks: Design and Experimental Evaluation," in IEEE Internet of Things Journal, vol. 2, no. 6, pp. 455-470, Dec. 2015, doi: 10.1109/JIOT.2015.2476915.

[18] M. R. Palattella et al., "On-the-Fly Bandwidth Reservation for 6TiSCH Wireless Industrial Networks," in IEEE Sensors Journal, vol. 16, no. 2, pp. 550-560, Jan.15, 2016, doi: 10.1109/JSEN.2015.2480886.

[19] R. Soua, E. Livolant and P. Minet, "DiSCA: A distributed scheduling for convergecast in multichannel wireless sensor networks," in 2015 IFIP/IEEE International Symposium on Integrated Network Management (IM), Ottawa, ON, Canada, 2015.

[20] A. Aijaz and U. Raza, "DeAMON: a decentralized adaptive multi-hop scheduling protoc-ol for 6Tisch wireless networks," IEEE Sensors Journal, vol. 17, no. 20, pp. 1-10, 2017.

[21] K. Kralevska, D. J. Vergados, Y. Jiang and A.

Michalas, "A Load Balancing Algorithm for Resource Allocation in IEEE 802.15.4e Networks," 2018 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops), 2018, pp. 675-680, doi: 10.1109/PERCOMW.2018.8480306.

[22] J. D. Vergados , K. Kralevska, Y. Jiang and A. Michalas, "Local voting: A new distributed bandwidth reservation algorithm for 6TiSCH networks," Computer Networks, vol. 180, no. 107384, 2020.

[23] S. Duaquennoy, B. A. Nahas, T. Watteyne and O. Landsiedel, “Orchestra: Robust Mesh Networks Through Autonomously Scheduled TSCH,” in ACM Conference on Embedded Networked Sensor Systems (Sensys), Seoul, South Korea, 2015.

[24] S. Kim, H. -S. Kim and C. Kim, "ALICE: Autonomous Link-based Cell Scheduling for TSCH," 2019 18th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN), Montreal, QC, Canada, 2019, pp. 121-132, doi: 10.1145/3302506.3310394.

[25] S. Jeong, J. Paek, H. S. Kim and S. Bahk, “TESLA: Traffic-Aware Elastic Slotframe Adjustment in TSCH Networks,” IEEE Access, vol. 7, pp. 130468 - 130483, 2019.

[26] Q. Wang, X. Vilajosana, and T. Watteyne, 6top Protocol (6P), Internet Engineering Task Force Std. RFC8480, August 2018.

[27] T. Chang, X. Vilajosana, M. Vucinic, S. Duaquennoy and D. Dujovne, “6TiSCH Minimal Scheduling Function (MSF),” 2 July 2019.[Online].Available:https://tools.ietf.org/html/draft-ietf-6tisch-msf-04. [Accessed 5 August 2019]

[28] Y. Tanaka, P. Minet, M. Vucinic, X. Vilajosana and T. Watteyne, "YSF: a 6TiSCH Scheduling Function Minimizing Latency of Data Gathering in IIoT," IEEE Internet of Things, pp. 2327-4662, 2021.

[29] F. Righetti, C. Vallati, G. Anastasi and S. K. Das, "Analysis and improvement of the on-the-fly bandwidth reservation algorithm for 6tisch", IEEE WoWMoM, 2018.

[30] H. Wang and A. O. Fapojuwo, "Design and Performance Evaluation of a Hysteresis-Free On-the-Fly Scheduling Function for 6TiSCH," in IEEE Internet of Things Journal, vol. 8, no. 13, pp. 10499-10508, 1 July1, 2021, doi: 10.1109/JIOT.2021.3049218

[31] V. Xavier, T. Watteyne, T. Chang, M. Vucinic, S. Doquennoy and P. Thubert, “IETF 6TiSCH: A Tutorial,” IEEE, vol. 22, no. 1, pp. 595-615, 2019.

[32] A. Tinka, T. Watteye, K. S. Pister and A. M. Bayen, “A decentralised scheduling algorithm for time synchronised channel hopping,” Mobi. Comm., vol. 11, no. 1, pp. 1-16, 2011.

[33] E. Municio, G. Daneels, M. Vucinic, S. Latre, J. Famaey, Y. Tanaka, K. Brun, k. Muraoka, X. Vilajosana and T. Watteyne, “Simulating 6TiSCH network,” Telecommuni-cation Journal, pp. 1-17, 2018.

[34] K. S. J. Pister, T. Watteyne, A. Nicola, X. Vilajosana and M. Kazushi, “Simple Distributed Scheduling with Collision Detection,” IEEE sensors, vol. 16, no. 15, p. 5848–5849, 2016.

[35] X. Vilajosana, Q. Wang, F. Chraim, T. Watteyne, T. Chang and K. S. J. Pister, “A realistic Energy Consumption Model for TSCH Networks,” IEEE Sensors, vol. 14, no. 2, pp. 482-489, 2014.

[36] M. Vučinić, T. Chang, B. Škrbić, E. Kočan, M. Pejanović-Djurišić and T. Watteyne,“Key Performance Indicators of the Reference 6TiSCH Implementation in Internet-of-Things Scenarios,” IEEE Access, vol. 8, pp. 79147 - 79157, 2020.

[37] T. G. van Kessel et al., "Methane Leak Detection and Localization Using Wireless Sensor Networks for Remote Oil and Gas Operations," 2018 IEEE SENSORS, 2018, pp. 1-4, doi: 10.1109/ICSENS.2018.8589585.

[38] Vučinić Mališa, Watteyne Thomas, Vilajosana Xavier. Broadcasting Strategies in 6TiSCH Networks. Internet Technology Letters. 2017.

Kaushal Kumar was born in Nagla Bihari Village, Sakaraya, Mathura, Uttar Pradesh, India in 1983. He received the MSc in computer science from the University of Stirling, UK, in 2008, and the MBA from Edinburgh Napier University, UK, in 2013. He is a tutor in the computer science department, and a student who is currently pursuing a Ph.D. degree in computer science at the University of Stirling, Stirling, UK.

Mario Kolberg is a Senior Lecturer of computing science and Associate Dean at the University of Stirling. His research interests include Peer-to-Peer overlay networks, Wireless Sensor Networks, and Internet of Things. Mario is editor of the bi-annually published Consumer Communication Networking Series within the IEEE Communications Magazine. Mario is also on the editorial Board of the Springer Journal ‘Peer-to-Peer Networking and Applications’ and has a long-standing involvement with the IEEE CCNC conference series. Dr. Kolberg has published more than 60 papers in leading journals and conferences. He is a member of a number of international conferences and program committees on networking and communications. He is a Senior Member of the IEEE and holds a PhD from the University of Strathclyde, UK.