energies

MDPI

Article

Autonomous Scheduling for Reliable Transmissions in Industrial Wireless Sensor Networks

Armaghan Darbandi and Myung-Kyun Kim *

Department of Electrical, Electronic and Computer Engineering, University of Ulsan, Ulsan 44610, Republic of Korea; hedye7@ulsan.ac.kr

* Correspondence: mkkim@ulsan.ac.kr

Abstract: Deploying Internet of Things (IoT) on low-power lossy wireless sensor/actuator networks (LLN) in harsh industrial environments presents challenges such as dynamic link qualities due to noise, signal attenuations and spurious interferences. However, the critical demand for industrial applications is reliability of data delivery on low-cost low-power sensor/actuator devices. To address these issues, this paper proposes a fully autonomous scheduling approach, called Auto-Sched, which ensures reliability of data delivery for both downlink and uplink traffic scheduling and enhances network robustness against node/link failures. To ensure reliability, Auto-Sched assigns retransmission time slots based on the reliability constraints of the communication link. To avoid collision issues, Auto-Sched creates an upward pipeline-like communication schedule for uplink end-to-end data delivery, and a downward pipeline-like communication schedule for downlink scheduling. For enhancing network robustness, we propose a simple algorithm for real-time autonomous schedule reconstruction, when node or link failures occur, with minimal influence on communication overhead. Performance evaluations quantified the performance of our proposed approaches under a variety of scenarios comparing them with existing approaches.

Keywords: IoT networks; LLN; autonomous scheduling; reliability; robustness; real-time

Citation: Darbandi, A.; Kim, M.-K. Autonomous Scheduling for Reliable Transmissions in Industrial Wireless Sensor Networks. *Energies* **2023**, *16*, 7039. https://doi.org/10.3390/en16207039

Academic Editors: Chun-Yen Chang, Cheng-Fu Yang, Charles Tijus, Teen-Hang Meen and Po-Lei Lee

Received: 1 September 2023
Revised: 27 September 2023
Accepted: 9 October 2023
Published: 11 October 2023

1. Introduction

The Industrial IoT (IIoT) wireless sensor actuator networks (WSANs) can be grouped into four areas: monitoring, control, optimization, and autonomy [1]. Examples include complex monitoring and control processes such as factory automation [2], distributed and process control [3,4] such as smart detection of liquid/gas leakage, and smart buildings. These applications require low-cost sensor or actuating devices that must operate unattended for years on modest batteries. This fact limits buffering capability, computational power, and communication range, and accordingly, the data rate is limited to 250 kbps in the 2.4 GHz band. Consequently, IIoT WSANs are prone to dynamic link quality issues and spurious interferences, whereas the critical demands in harsh industrial environments are reliability and resilience to node or link failure.

The networks mentioned above are summarized as low-power and lossy networks (LLN) exhibiting a set of challenging resource allocation problems. In the light of the characteristics of LLNs, the Routing Over Low power and Lossy networks Working Group (ROLL WG) within the Internet Engineering Task Force (IETF) [5] has designed and specified an IPv6 routing protocol for LLNs (RPL) [6]. RPL consists of a set of metrics and constraints suitable for routing over limited-memory and limited-energy LLNs. Specifically, the objective function (OF) in RPL defines a parent selection metric, which enables each node (or device) to select its preferred parent among the neighbor nodes and forward its packet toward the gateway. Minimum rank with hysteresis objective function (MRHOF) [7] is the most commonly used OF that leverages link reliability. In addition, contention-free IEEE 802.15.4e time-synchronized channel hopping (TSCH) [8] has been established as a

standard for highly reliable and deterministic time and channel scheduling in LLNs. TSCH reveals time-slotted frequency hopping potentials for offering stringent timing requirements. However, in order to meet application specific constraints, the scheduling algorithm is left to the vendor.

However, even when using time division multiple access (TDMA), transmission errors occur because of the signal-to-noise ratio (SNR) changes due to harsh and unstable industrial environments. To tackle this issue, a promising approach is packet retransmission, which arranges reservations of redundant time and frequency to fulfill constraints given by reliability and delay. This mechanism has already been built into most existing products, wireless network standards, and proposed approaches such as [9–18]. The schedulers can be categorized into centralized approaches [10–12], where a central entity provides optimality periodically, and decentralized and autonomous scheduling approaches [9,13–19] that enable each node to decide on a reliable schedule autonomously, with minimum control packet exchange. Our contribution is primarily motivated by multiple drawbacks of existing approaches. For instance, the exponential computational complexity issues of the centralized scheduling approaches may result in delay issues for periodic scheduling reconstructions or when faults are detected. In addition, for a majority of existing autonomous scheduling approaches, the delay issues are a result of the fact that each node can forward a single packet in each scheduling period, leading to long end-to-end response times and packet drops. Additionally, current autonomous approaches exhibit a lack of support for retransmission mechanisms. By omitting the impact of link reliability on successful transmissions, these approaches implicitly assume that all links have ideal and flawless quality, thereby posing risks of packet drops in practical scenarios.

To tackle the above issues, the primary objective of this study is to present an autonomous scheduling approach, denoted as Auto-Sched, that enhances reliability, efficiency, and resiliency against node and link failures. The main contributions of Auto-Sched can be summarized as follows.

- Auto-Sched enhances the reliability of data transmissions. To do so, it provides dedicated slots for autonomous transmission and retransmission scheduling. Each node autonomously computes its transmission and reception time slot scheduling based on hop counts to the gateway, its unique identifier (MAC address or a unique node ID [14]), the current link quality and the worst-case link quality constraints in the network. For uplink schedules, our methodology to ensure allocating collision-free time slots is to create parallel pipeline-like communication schedules for all nodes, where each pipeline in the scheduling table starts from the slot allocated to the source node and ends at the slot allocated to the gateway. Similarly, for downlink schedules, parallel downward pipeline-like communication schedules are constructed, starting from the slot allocated to the gateway and ending at the slot allocated to the corresponding destination actuator node. Our performance analysis in Section 5 demonstrates that Auto-Sched significantly enhances the packet delivery ratio (PDR) and mitigates end-to-end delays across varying network sizes and packet generation intervals, compared with widely adopted techniques in [13,15,17].
- Auto-Sched enhances robustness against network changes. We propose a simple algorithm that enables a node aimed at changing the parent (due to link or node failures) or joining the network, delivering its request to intermediate nodes through the new path to the gateway within, at most, two slotframes. To do so, Auto-Sched allocates a time slot for each individual node in the network, to receive collision-free join or parent change requests. Once the request is passed through the intermediate nodes, the time and frequency scheduling are computed autonomously by Auto-Sched, enhancing resiliency against faults and minimizing packet drops. This means that, a multi-path or multicast approach is no longer needed to overcome node or link failure issues. This feature of Auto-Sched enables each node to flawlessly change parent and construct new routes in the event of link or node failure, with minimal influence

on communications and computations. Our analysis shows superior performance in reducing delay and packet drop compared with approaches in [13,15].

The remainder of this paper is organized as follows. Section 2 describes the related works. Section 3 introduces the system model and the problem definition. Section 4 describes our proposed approach. Section 5 presents the evaluation results, with a conclusion following in Section 6.

2. Related Works

By leveraging network parameters and attributes maintained by all field devices, autonomous scheduling eliminates the need for dedicated communication between neighbors or reliance on a central entity, thereby mitigating energy and bandwidth consumption overhead. However, the inherent simplicity of the proposed approaches presents challenging issues that limit their applicability to restricted domains.

Collision issues in dedicated slots: Orchestra in [14], as the pioneering autonomous scheduling approach, allocates a dedicated time slot for each node based on its unique identifier. The sender-based policy of Orchestra allocates a single transmitting slot to each node, and the receiver-based policy dedicates a single receiving slot to each node. However, collision and hidden node problems remain prevalent in the receiver-based model, and delay and packet drop issues arise as inevitable challenges in the sender-based model. The main reason is that, in the receiver-based model, multiple child nodes may simultaneously transmit packets to the parent node, while in the sender-based model, each node is restricted to a single transmitting slot in each scheduling period.

Collision issues: The approaches in [17,18] enhance Orchestra's receiver-based scheduling policy by adapting the static schedule of Orchestra to high traffic load or traffic bursts. The approach in [17], called OrchEx in this paper, mainly focuses on the gateway-immediate child nodes that are responsible for forwarding network traffic loads to the gateway. When the buffer of a child node exceeds a given threshold, it alerts the gateway about potential congestion. The gateway responds by adding more reception time slots for that particular child node, by hash function. The quantity of additional allocated time slots is proportional to the size of the sub-tree rooted at the child node. This approach can lead to scalability issues, since it mitigates the collision issue solely at the gateway side, while the collision issue persists in the rest of the nodes in the network. In the approach introduced in [18], denoted as OSCAR, a super-slotframe consisting of multiple slotframes is defined. In the initial slotframe, the Orchestra receiver-based scheduling policy is implemented, while in the k'th slotframe, nodes with a rank of k are excluded from the scheduling allocation, resulting in higher energy efficiency for low traffic loads. In the RPL standard, rank depicts the distance of the node from the gateway. The number of reception time slots allocated for the nodes at each rank is fixed and does not change with traffic load or link reliability. Therefore, an insufficient number of time slots can be allocated to a node in case of high traffic load or unreliable links.

ALICE in [16] mitigates the collision and hidden node issues inherent in Orchestra, by allocating a unique cell for each directional link. The approach in [13] enhances the reliability of Orchestra, primarily by allocating excess dedicated transmitting slots for retransmissions by each node. Furthermore, the authors leverage graph routing as an alternative to the conventional RPL protocol, thereby introducing path diversity and improving robustness against node or link failures. This approach is called SchedEx in this paper. Despite the notable improvements in SchedEx, and as demonstrated by simulation results in Section 5, the challenges of delay and packet drop issues still persist. This fact stems from the limitation in assigning a single slot for a node to transmit a single packet in each scheduling period.

Delay and packet drop issues: Escalator in [15] targets the above delay and packet drop issues by sequential slot assignment along the transmission path, starting from the source node to the gateway. This strategy ensures that all nodes autonomously possess slots to forward the received packets, and each received packet is promptly forwarded in

the next immediate slot. To construct the reception/forwarding schedule of each child node in the sub-tree, the node uses only the unique identification of child nodes and its hop-count. Consequently, delay and packet drop issues are effectively resolved as packets are consistently and sequentially received and forwarded by intermediate nodes throughout the path to the gateway. This strategy leads to packet delivery to the gateway at the same scheduling interval as the packet is generated.

Reliability issues for data packets: In most of the above approaches, the retransmission opportunities for ensuring successful delivery of application data from sensor nodes to actuator nodes are not investigated. Omitting the impact of link loss implies that the link quality is assumed to be continuously ideal at each time instant. However, the connectivity between each pair of nodes can be impacted by external interference or due to path loss and multipath fading phenomena of wireless links in harsh industrial environments.

Long delay posed on RPL layer configurations: In addition to scheduling application data traffic, the majority of the aforementioned approaches also address scheduling the synchronization traffic (i.e., enhanced beacons or EB control packets) and routing update traffic (RPL control packets). To do so, a prioritized slotframe configuration is proposed, wherein a slotframe with highest priority is specified for scheduling EB packets, and its length corresponds to the period of EB packets. Additionally, a slotframe with next priority is specified for routing control packets, and its length aligns with the RPL update period. Finally, a data packet slotframe with its length configured to the application data period has the lowest priority. These slotframes run in parallel and can lead to collisions between EBs, RPL control packets and application data packets. Consequently, at a time slot where collisions occur, EB control packets can interrupt the transmission of both routing control packets and data packets, while routing control packets can interrupt the transmission of data packets.

Orchestra drops the lower-priority packet when a collision occurs. The autonomous approach in [13] autonomously defers all traffic with lower priority in conflict to a conflict-free slot within the slotframe, avoiding forced packet drops. Therefore, the length of the application data slotframe must be long enough to accommodate deferrals resulting from all types of collisions. Escalator in [15] addresses this challenge by deferring the colliding data packets to the subsequent slotframe. In addition, several restrictions are applied to the length of application data and routing slotframes to ensure that colliding data packets can be successfully transmitted in the next slotframe, with no collisions.

In all above approaches, despite the high priority of RPL control packets, a single shared slot within the routing slotframe is allocated for all sensor and actuator devices, to broadcast or unicast their routing update control packets. In this case, collisions between control packets increase significantly at network bootstrap or when any changes occur in the network, considering that in such phases, several control packets are generated to notify the neighbors about instabilities or joining requests. Such collisions lead to delays in packet delivery and ultimately cause high packet drops. In addition, an urgent RPL message from a node (for instance, notifications of congestion, link/node failure or join requests) may not be received by neighbor nodes, as both sender and receiver nodes select a random channel within this single time slot to send and listen, respectively. This problem becomes more significant, as this request must travel through the path toward the gateway using a shared slot in the RPL slotframe. Consequently, this issue leads to long delays in handling dynamic network changes or the scalability of the network to accommodate new nodes.

In contrast to the above approaches, Auto-Sched schedules retransmissions for unreliable links, thereby taking into account a comprehensive network model specifically designed for reliability-critical industrial WSANs. In addition, the slot dedicated to RPL control packets by Auto-Sched mitigates the issues related to collisions and delays, leading to a minimum in the required time for a node to successfully handle changes in the network. Our contribution to the handling of network changes is similar to the approach in [19], wherein a notification is propagated to all nodes responsible for handling the changes in the network, and then the responsible nodes generate a schedule for handling the disturbance.

However, in Auto-sched, the nodes autonomously handle the issue, and packet drop issues are avoided, while the approach in [19] employs a distributive carrier sense multiple access/collision avoidance (CSMA/CA)–based approach, leading to contention for network resources and packet drop issues. Auto-Sched achieves this aim with minimum delay, as shown in Section 5, an aspect that has not been addressed by prior works and could potentially lead to long delays when utilizing the single shared slot in a routing slotframe.

Discussions

An overview of all of the discussed schedulers is presented in Table 1. Readers interested in more details are referred to [20] and references therein. In the subsequent discussion, we shall examine the performance results achieved in corresponding manuscripts.

Table 1. Overview of the related works.

Scheduler	Strong Points	Limitations
Orchestra Receiver-based [14]	Simplicity, Low bandwidth, Low Energy consumption	Collision, hidden node, congestion and packet drop issues.
Orchestra Sender-based [14]		Delay, congestion and packet drop issues.
OrchEx [17]	Simplicity, Improved collision and packet drop issues compared to Orchestra Receiver-based.	Collision, delay, congestion and packet drop issues persists in intermediate nodes that do not constitute the child nodes of gateway.
OSCAR [18]	Simplicity, Lower bandwidth and energy consumption compared to Orchestra Receiver-based.	Fixed amount of time slots are allocated to each node. Thus, for unreliable links or for high traffic loads, it may define insufficient resource allocation, while for high link qualities or for low traffic load, it may define unnecessary resource allocation.
ALICE [16]	Improved collision and packet drop issues compared to Orchestra. Supports both downlink and uplink traffic.	A single time slot is allocated to each communication link. Thus, it can result in insufficient resource allocation.
SchedEx [13]	Improves collision and packet drop issues compared to Orchestra Sender-based. Higher reliability in routing layer.	Fixed bandwidth is allocated to each node, which cannot be adopted to low or higher link quality constraints. Not usable with RPL standard.
Escalator [15]	Sufficient amount of bandwidth is granted to each node to deliver all buffered packets to the gateway within a single slotframe.	The resource allocation does not deal with links reliability, and an ideal link quality is assumed. No support for downlink traffic.

The simulation results, as detailed in [17], show that both OrchEx and Orchestra achieve 100% PDR in a network consisting of 20 nodes, provided that the packet generation interval exceeds 12 s. However, when subjected to higher traffic load, OrchEx demonstrated nearly 10% higher performance. Notably, in comparison to OSCAR, where the packet generation interval is one packet every minute, OrchEx shows higher performance. Specifically, within networks encompassing fewer than 60 nodes, OrchEx outperforms OSCAR in terms of packet delivery delays. However, as network size expands from 80 to 100 nodes, it exhibits the same performance as that of OSCAR. The authors relate this to less congestion in intermediate nodes in OrchEx, when network size is small enough, leading to the outperformance by OrchEx.

ALICE performance was simulated against Orchestra, in a network comprising 68 nodes, when traffic load increased from one packet to six packets per minute. ALICE maintains higher PDR under a heavy traffic load, since it provides more transmission and less contention and collisions. It provides slightly longer latency than Orchestra under light traffic load due to a larger slotframe size. However, as the traffic load increases, ALICE incurs better latency. The resulting lower packet drops in ALICE enable RPL to

provide stable topology, resulting from less parent switches (high packet drops initiate parent switch) and lower routing control packet communications.

The conducted simulations in [13] assess the performance of SchedEx in comparison to Orchestra within a network comprising 150 nodes. The network involved 20 data flow sets, and the packet generation interval was set at 10 s. The results revealed that, on average, SchedEx achieves 16.3% higher PDR than Orchestra. This is mainly attributed to the provision of additional transmission slots assigned to each node, which serves to mitigate link unreliability. Furthermore, the utilization of a graph played a key role in enhancing the performance. In a network consisting of 36 nodes, Escalator shows 100% PDR with packet generation intervals of both 20 and 5 s. However Orchestra, with a slotframe containing 37 time slots, exhibited a significant packet drop, decreasing from 90% to 50%, respectively.

Overall, different protocols excel in different scenarios. For example, Orchestra, OrchEx and OSCAR achieve reasonable PDR ratios in low traffic settings or smaller networks. ALICE, SchedEx and Escalator demonstrate higher scalability to network size or traffic loads, as they offer enhanced reliability in more demanding networks. The simulation results in Section 5 shows that Auto-Sched results in higher performance compared to SchedEx and Escalator, since it is designed to ensure end-to-end delivery of each individual packet under a generalized system model that supports unreliable links.

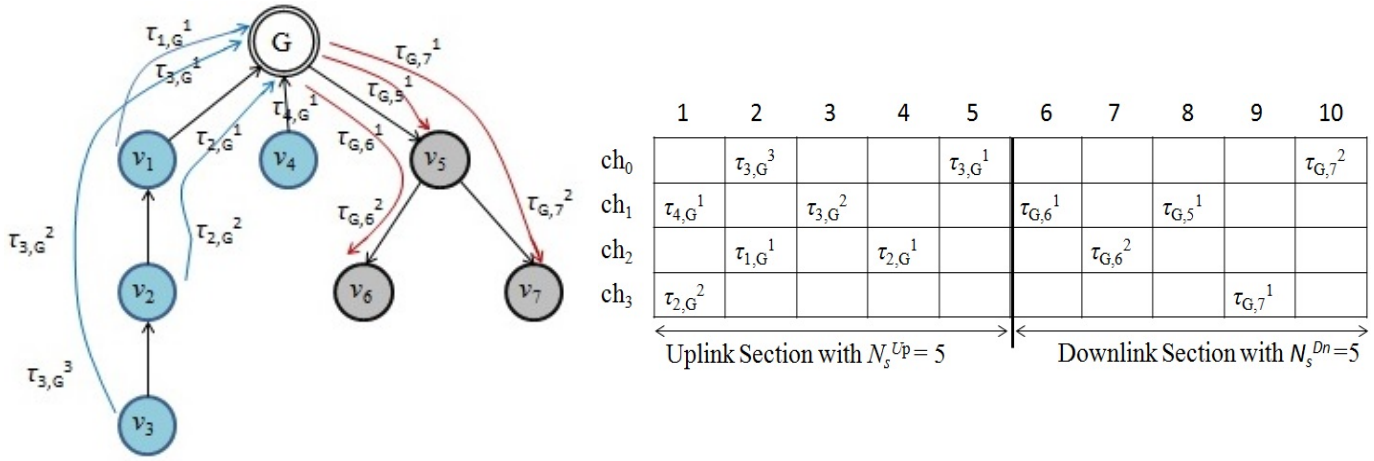
3. Preliminaries

3.1. System Model and Notations

In TSCH, the communication takes place identically in periodic cycles called slotframes. Each slotframe is divided into N_S number of 10 ms-sized time slots, and the total length of the slotframe is $L_S = N_S$. The total number of timeslots that have elapsed since the start of the network or an arbitrary start time determined by the Personal Area Network (PAN) coordinator is called the Absolute Slot Number (ASN). It increments globally in the network at the beginning of each time slot, and is used globally by devices as the slot counter. Each slot is long enough for the transmitter to send a maximum-length packet and for the receiver to send back an acknowledgment. Initially, $n_{ch} \leq 16$ different channels are available for communication. Each channel is identified by a channel-offset, which is an integer value in the range [0, 15]. Each field device is assumed to be equipped with a half-duplex radio transceiver that cannot transmit and receive concurrently. As each device supports communications on multiple channels, multiple node pairs communicate at the same time slots using different channel offsets, thereby increasing the network capacity.

Auto-Sched partitions the slotframe into two distinct sections, as seen in Figure 1b: the first section, called the uplink section, is allocated for uplink traffic scheduling, enabling convergecast communications from sensor nodes to the gateway (or controller). The length of this section is denoted by ${L_S}^{Up}$. Similarly, the second section, called the downlink section, is designated for downlink traffic scheduling, facilitating the distribution of the control data generated by the controller to the actuators¸ and the length of this section is denoted by ${L_S}^{Dn}$.

We adopt the system architecture of a typical WSAN modeled as a directed acyclic graph (DAG) $A = (V,E)$, where $V = \{G, v_1, v_2, \ldots, v_N\}$ is the set of all sensor and actuator devices, arcs in E are communication links, and G is the gateway node that acts as a controller, as seen in Figure 1a. In the rest of this paper, we will use the terms gateway and controller interchangeably. This DAG is constructed by the RPL protocol, the procedure of which is explained in Section 3.2. All sensor and actuator devices are wirelessly connected to a controller node directly or through multi-hop routing. Each link in E is identified by an ordered pair of nodes, e.g., v_iv_j, where v_i and v_j are the transmitting receiving nodes, respectively. The link v_iv_j has an error rate (or loss rate) $\varepsilon(v_i,v_j)$, which is the probability that a transmission over link v_iv_j does not succeed. Thus, $ETX(v_i, v_j) = \frac{1}{1-\varepsilon(v_i,v_j)}$ represents the number of expected transmission/retransmissions for successfully transmitting a packet between v_i and v_j. We assume that the maximum ETX value is given from the worst possible link quality between two nodes, and we denote it by $\omega = \max(ETX(v_i, v_j))$.



(**a**) (**b**)

Figure 1. (**a**) An example network graph with four sensor devices sending data to the gateway G and three actuator devices receiving controlling course of actions from G. The links v_1G and Gv_5 are reliable, while the rest of the links have $ETX(v_i,v_j) = 2$. (**b**) An example schedule in a TSCH slotframe consisting of ${N_S}^{Up}$ and ${N_S}^{Dn}$ time slots in uplink and downlink sections, respectively.

A sensor node periodically collects sensing information and sends the generated data to the central controller, an actuator device periodically receives the optimum course of actions from the controller, and both devices support multi-hop routing. Two links conflict with each other if they share a common sender or receiver. The conflicting links cannot share the same time slot. We shall denote by $CNF(v_i,v_j)$ the set of all links conflicting with v_iv_j, e.g., $CNF(v_1, G)=\{v_2v_1, v_4G, Gv_5\}$ in Figure 1a. Two links interfere with each other if the receiver or transmitter node of one link can overhear transmissions by the sender of the other link. The interfering links cannot share the same communication cell. We shall denote by $INF(v_i,v_j)$ the set of all links interfering with v_iv_j. The parent node of v_i is denoted by $Pr(v_i)$, and the set of child nodes in the sub-tree of the network rooted at node v_i is denoted by $SG\text{-}Tree(v_i)$.

Each source sensor node v_S with H_S hop-counts from the gateway periodically generates a packet τ_S. τ_S is periodically released at the beginning of each slotframe and travels through its designated uplink path to controller node G. The deadline for delivery of this packet is at the end of ${L_S}^{Up}$. The transmission of τ_S along the edge v_pv_q, where v_p is k hops away from G, is denoted by ${\tau_{S,G}}^k$. In Figure 1a, the transmissions of packets generated by sensor nodes are inserted in the corresponding links. Accordingly, the controller node (or gateway) periodically generates the controlling data packet τ_D destined for the actuator node v_D. This packet is periodically released at the beginning of ${L_S}^{Dn}$, and it travels through the downlink path toward v_D. The deadline for delivery of this packet is at the end of ${L_S}^{Dn}$. The transmission of $\tau_{D,}$ along the edge v_pv_q, where v_q is k hops away from G, is denoted by ${\tau_{G,D}}^k$.

3.2. Overview of RPL DODAG Construction

RPL is a distance vector protocol that constructs a destination-oriented DAG (DODAG) based on a metric called rank, defined in OF. Specifically, rank depicts the accumulative cost of each node toward the gateway. MRHOF defines link reliability as the metric for a node to select its preferred parent among the neighbor nodes. To do so, the gateway initiates the DODAG construction by broadcasting a DODAG information object (DIO) message consisting of rank = 0, periodically. Upon receiving a DIO message, a node selects a parent with highest link reliability among neighbors with lowest hop-count to gateway. Then, the node broadcasts its DIO with its own accumulated rank with that of its selected parent.

This process repeats at each node and continues until all of the nodes in the network join DODAG to form a tree-structured topology.

The DIO broadcast period is controlled by trickle timer [21] to maintain a balance between control message overhead and the freshness of routing information. A timer varying from I_{min} to I_{max} is used to control the interval between two consecutive DIO messages. Specifically, the trickle algorithm uses I_{min} as the first interval and then doubles the size of the interval until it reaches I_{max}. If a node detects a change of its parent, selects a new parent node, or receives a DODAG information solicitation message (DIS), it resets its trickle timer to I_{min} to quickly update its rank to its neighbors. When a new node joins the network and does not receive a DIO for a time out, it may send a DIS message to request a DIO and discover the network metrics and constraints.

Upon selecting the parent, the sensor or actuator node initiates the transmission of a destination advertisement object (DAO) message to its parent through the route to the gateway. When changing the preferred parent due to link variation, a node sends a DAO to the new preferred parent to set up a route and a no-path DAO to the old preferred parent to remove the old downward route [16]. The DAO generated by node v_i contains its unique identifier. In Auto-Sched, this fact enables each intermediate node to autonomously construct the reception and forward slots for packet τ_i. In Section 4.1, the construction of uplink schedules is detailed after reception of a DAO. Subsequently, Section 4.2 describes the procedure of constructing downlink schedules after receiving a DAO.

3.3. Problem Definition

The problem defined in this paper is to find a feasible and reliable schedule for all of the transmissions of each packet along their respective designated paths. This means that all packets must be scheduled before the end of the slotframe, and the hop-to-hop retransmissions must ensure successful packet reception. Given the TSCH network A, the length of slotframe L_S, the set of transmissions of each packet in U, and the set of ETX of the links in R, we would like to solve the following problem:

$$\arg min_{\Gamma} E(A, L_S,\ R,\ U, n_{ch}) \tag{1}$$

$$ASN\left(\tau_{vs,G}^{k+1}\right) < ASN\left(\tau_{vs,G}^{k}\right),\ \forall v_s \tag{2}$$

$$ASN\left(\tau_{G,vD}^{k}\right) < ASN\left(\tau_{G,vD}^{k+1}\right),\ \forall v_D \tag{3}$$

$$D_i \le L_S^{Up},\ \forall \tau_S,\ D_i \le L_S^{Dn},\ \forall \tau_D \tag{4}$$

$$X_{p,q}(t, ch) + X_{m,n}(t, ch) \le 1,\ \forall v_m v_n \in INF(v_p v_q) \tag{5}$$

$$Y_{p,q}(t) + Y_{m,n}(t) \le 1,\ \forall v_m v_n \in CNF(v_p v_q) \tag{6}$$

$$Z_{p,q}(\tau_i) \le ETX(v_p, v_q),\ \forall \tau_i,\ ETX(v_p, v_q) \in R \tag{7}$$

where Γ represents a schedule with maximum reliability. The constraints in Equations (2) and (3) state that the transmission ${\tau_{i,j}}^{k+1}$ must occur earlier than ${\tau_{i,j}}^{k}$ if it is uplink traffic, and the transmission ${\tau_{i,j}}^{k}$ must occur earlier than ${\tau_{i,j}}^{k+1}$ if it is downlink traffic. Equation (4) restricts the end-to-end delays to the end of slotframe. The constraints in Equation (5) and Equation (6) state that two interfering links cannot be scheduled in the same communication cell (t, ch) (i.e., t's time slot and ch's channel) in the TSCH slotframe, and two conflicting links cannot be scheduled in the same time slot t. The variable $X_{p,q}$(t,ch) is a binary decision variable that is set to 1 when the transmission on the link $v_p v_q$ is allocated to the communication cell (t, ch), and 0 otherwise. Similarly, $Y_{p,q}(t)$ is a binary decision variable that is set to 1 when the transmission on the link $v_p v_q$ is allocated to the time slot t, and 0 otherwise. The constraint in Equation (7) ensures the successful delivery of each packet τ_i on the link $v_p v_q$, where $Z_{p,q}(\tau_i)$ denotes the number of retransmissions on link $v_p v_q$.

4. Autonomous Scheduling for Control Systems

This section introduces the autonomous scheduling Auto-Sched to solve the problem defined in Section 3.3. Auto-Sched defines two variations: Auto-Sched^{U} and Auto-Sched^{D} for autonomous reliable transmissions and retransmissions for uplink and downlink schedules, respectively. In the uplink direction, each intermediate node v_i allocates 2ω timeslots for each source sensor node $v_S \in SG\text{-}Tree(v_i)$, where ω slots are allocated for receiving the packet generated by v_S and ω immediate next slots to forward it toward G. Similarly in the downlink direction, each intermediate node v_i allocates 2ω timeslots for scheduling the packet generated by controller node G, where the first ω slots are allocated for receiving the generated packet and ω immediate next slots to forward it toward v_D. In both scenarios, this consecutive allocation of reception and transmission slots results in a pipeline-shaped schedule for each packet along its path.

This pipeline orientation along with ω autonomously allocated slots for transmission and retransmissions minimizes collisions and contention for resources, as shown in next section, and enhances reliable data flow for critical applications where data accuracy and integrity are crucial for making correct decisions. More importantly, as described in Section 4.3, nodes can adjust their RPL paths based on change in the network, without changing the pipeline shape or its position in the slotframe, resulting in minimum communication, computation and route re-construction delay overhead.

4.1. Autonomous and Reliable Time Slot Allocation for Uplink Traffic

The notations used in this section are listed in Table 2.

Table 2. Definition of notations used for uplink scheduling.

Symbol	Meaning	Symbol	Meaning
$S(v_S)$	Set of time slots allocated to the source sensor node S, to transmit ${\tau_{S,G}}^1$, to receive join requests and to broadcast EBs.	$Rx^R\ (v_p, {\tau_{S,G}}^k)$	The set of time slots allocated to intermediate node v_p for receiving packet τ_S, due to the worst-case link quality.
$Tx^R\ (v_S)$	The subset of time slots in $S(v_S)$ reserved for re-transmitting ${\tau_{S,G}}^1$ to $pr(v_S)$, in the worst-case link quality.	$Rx(v_p, {\tau_{S,G}}^k)$	The subset of time slots in $Rx^R(v_p, {\tau_{S,G}}^k)$ allocated to intermediate node v_p for receiving packet τ_S, due to the current actual link quality.
$Tx\ (v_S)$	The subset of time slots in $Tx^R\ (v_S)$ used for re-transmitting ${\tau_{S,G}}^1$ to $pr(v_S)$, due to the current link quality.	$Fd^R\ (v_p, {\tau_{S,G}}^k)$	The set of time slots allocated to intermediate node v_p for forwarding packet τ_S, due to the worst-case link quality.
$J(v_S)$	The subset of time slots in $S(v_S)$ that are utilized to receive join requests such as DAO and DIS.	$Fd\ (v_p, {\tau_{S,G}}^k)$	The subset of time slots in $Fd^R\ (v_p, {\tau_{S,G}}^k)$ allocated to intermediate node v_p for forwarding the packet τ_S, due to the current actual link quality.
$B(v_S)$	A time slot in $S(v_S)$ that is utilized to broadcast EB control packets.	$Ch\_Tx(v_i)$	The channel offset allocated to each node v_i to transmit the buffered packets.
		$Ch\_Rx(v_i)$	The channel offset allocated to each node v_i to receive the data packets from child nodes.

As Figure 2 illustrates, the upward pipeline-like schedule starts from the slots assigned to source node v_S. Each source sensor node reserves $2\omega + 1$ consecutive time slots in the TSCH slotframe, where the first ω time slots for receiving unicast RPL control packets such as DAO, the next slot is reserved to broadcast enhanced beacon (EB) control packets, and the next ω time slots for transmission/retransmissions of the generated packet. For this

purpose, Equation (8) defines the set of slots autonomously allocated to node v_S, based on the node identifier (S), the hop-count of v_S to gateway (H_S) and the value of ω:

$$S(v_S) = \{(2\omega+1)\cdot S - H_S\cdot\omega + m_S \,|-(\omega+1) \le m_S \le \omega - 1\}. \quad (8)$$

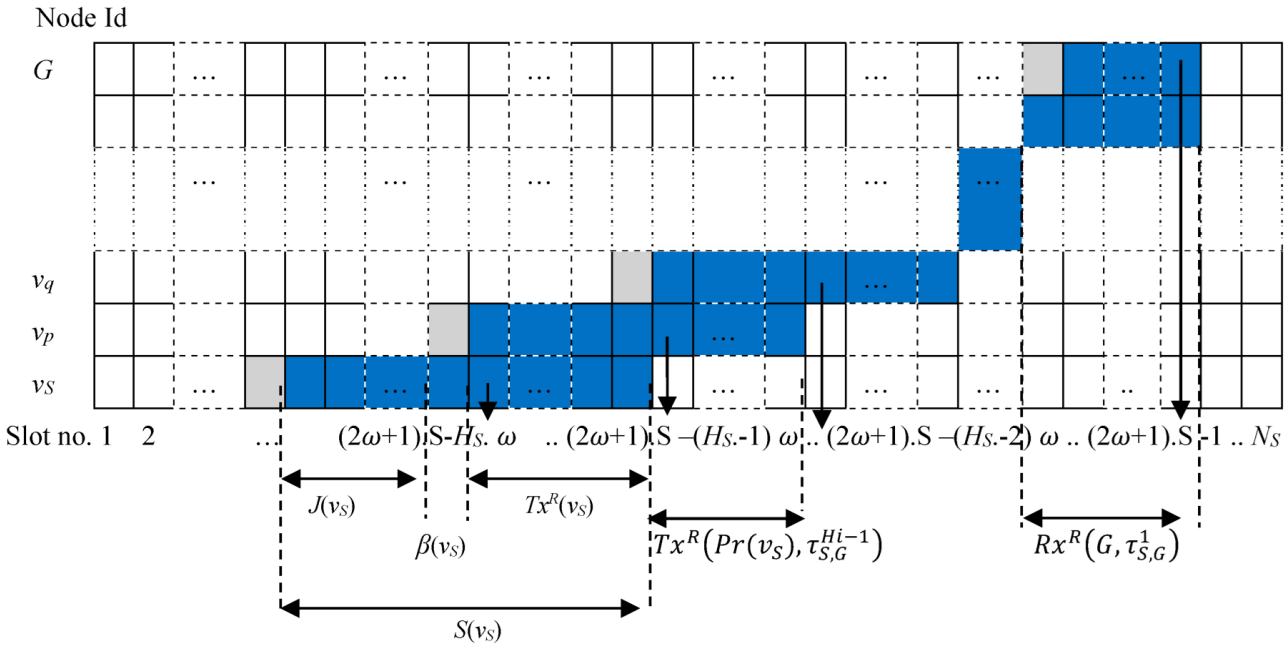


Figure 2. Timeline of Auto-Sched^U scheduling of node v_S and all intermediate nodes in its path toward the gateway, where $v_p = Pr(v_S)$, $v_q = Pr(v_p)$.

Within the set $S(v_S)$, the slots in the set $Tx^R(v_S)$ are allocated for the purpose of transmitting/re-transmitting the packet $\tau_{S,G}{}^1$ to $pr(v_S)$ in the worst-case link quality:

$$Tx^R(v_S) = \left\{(2\omega+1)\cdot S - H_S\cdot\omega + m_{Tx}^R(v_S)\middle| 0 \le m_{Tx}^R(v_S) \le \omega - 1\right\}, \quad (9)$$

$$Tx(v_S) = \{(2\omega+1)\cdot S - H_S\cdot\omega + m_{Tx}(v_S)|0 \le m_{Tx}(v_S) \le ETX(v_i, pr(v_i)) - 1\}, \quad (10)$$

while the slots in the set $Tx(v_S)$ are utilized due to the actual link quality. The first ω slots are utilized to receive join requests such as DAO and DIS, and the next slot for broadcasting EB control packets. These designated slots are given in the set $J(v_S)$ and $\beta(v_S)$, respectively, as follows:

$$J(v_S) = \{(2\omega+1)\cdot S - H_S\cdot\omega + m_{JR}(v_i)|-\omega \le m_{JR}(v_S) < -1\}, \quad (11)$$

$$\beta(v_S) = (2\omega+1)\cdot S - H_S\cdot\omega - 1\,. \quad (12)$$

The slots assigned for $J(v_S)$ enable reliability and determinism for DAO control packets, facilitating RPL operations in managing join or parent change requests. Section 4.3 elaborates on the utilization of $J(v_S)$ for reliable and deterministic parent change policy.

The gray colored slots in Figure 2 are the additional slots that each node reserves to ensure collision-free communication for $\beta(v_S)$. This fact is elaborated later in Theorem 3. Additionally, from Equations (11) and (12), it can be concluded that each node can calculate $J(v_j)$ and $\beta(v_j)$ of each neighbor node v_j, to receive and transmit EB and RPL control packets, respectively.

The set of transmission time slots defined by Equation (9) implies that $pr(v_S)$ must reserve all of the slots in $Tx^R(v_S)$ to receive $\tau_{S,G}{}^M$ from v_S, where $M = H_s$. But $pr(v_S)$ only turns on its receiver in the slots defined by $Tx(v_S)$, due to actual retransmissions given from link ETX. Consequently, $pr(v_S)$ reserves the next consecutive ω slots, in order

to forward $\tau_{S,G}{}^{M}$. In general, each intermediate node v_p, which is k hops away from v_S ($H_p = k$), autonomously allocates ω slots by Equations (13) and (14), in order to receive the transmission $\tau_{S,G}{}^{k+1}$ of packet τ_S:

$$Rx^R\left(v_P, \tau_{S,G}^{k+1}\right) = \left\{(2\omega+1)j - k{\cdot}\omega + m_{Rx}^R\left(v_P, \tau_{S,G}^{k+1}\right) \middle| -\omega \le m_{Rx}^R\left(v_P, \tau_{S,G}^{k+1}\right) \le -1\right\}, \tag{13}$$

$$Rx\left(v_P, \tau_{S,G}^{k+1}\right) = \left\{(2\omega+1)S - k{\cdot}\omega + m_{Rx}\left(v_P, \tau_{S,G}^{k}\right) \middle| -\omega \le m_{Rx}\left(v_P, \tau_{S,G}^{k}\right) \le -\omega + ETX(v_P, pr(v_P)) - 1\right\} \tag{14}$$

Therefore, the intermediate node v_p forwards τ_S to the next ω slots defined by the set $Fd(v_P, \tau_{S,G}{}^{k})$, as follows:

$$Fd^R\left(v_P, \tau_{S,G}^{k}\right) = \left\{(2\omega+1)S - k{\cdot}\omega + m_{Fd}^R\left(v_P, \tau_{S,G}^{k}\right) \middle| 0 \le m_{Fd}^R\left(v_P, \tau_{S,G}^{k}\right) \le \omega - 1\right\}, \tag{15}$$

$$Fd\left(v_P, \tau_{S,G}^{k}\right) = \left\{(2\omega+1)S - k\cdot\omega + m_{Fd}\left(v_P, \tau_{S,G}^{k}\right) \middle| 0 \le m_{Fd}\left(v_P, \tau_{S,G}^{k}\right) \le ETX(v_P,\ pr(v_P))\ - 1\right\}, \tag{16}$$

From the above discussion, it can be seen that each sensor node v_i autonomously allocates six types of slots: (i) ω slots for transmission of its own generated packet by Equation (9), (ii) $\omega - 1$ slots for receiving join requests by Equation (11), (iii) one slot for broadcasting EB control packets by Equation (12), (iv) ω slots for receiving a packet from a child sensor node in *SG-Tree*(v_i) by Equation (13), (v) ω slots for forwarding this packet toward the gateway by Equation (15), and (vi) ω slots for join request and broadcast slots of neighbor nodes by Equations (11) and (12). The following theorems analyze the interference and collision conditions caused by Auto-Sched^{U}.

Theorem 1. *Auto-Sched^{U} results in interference-free autonomous slot scheduling, if every two nodes with a distance of 2- hops away employ distinct channels for transmissions.*

Proof. Suppose the neighbor nodes v_i and v_j receive a packet from source nodes $v_p \in$ *SG-Tree*(v_i) and $v_q \in$ *SG-Tree*(v_j), respectively. The transmissions/reception slots allocated for nodes v_p and v_q in the intermediate nodes v_i and v_j interfere if one of the following cases occur:

Case. 1. The reception slots in v_i and v_j overlap. In this case, we have $(2\omega+1)\cdot p - k_i{\cdot}\omega + m_{Rx}^R(v_i, \tau_p) = (2\omega+1)\cdot q - k_j{\cdot}\omega + m_{Rx}^R(v_j, \tau_q)$. Hence, the following relationships between the hop counts of nodes v_i and v_j can be deduced:

$$((2\omega+1)|\,p-q| + M)/\omega = \left|\,H_j - H_i\right|, \tag{17}$$

where $0 \le\ M = \left|m_{Rx}^R(v_i, \tau_p) - m_{Rx}^R(v_j, \tau_q)\right| < \omega$. According to Equation (17), $H_j - H_i$ has minimum value when we have $p - q = 1$, which means that the nodes v_p and v_q have consecutive identifications (e.g., p = 2, q = 3), and $M = 0$, which implies that the first reception slots are overlapped. In this case, we have $Min\left(\left|\,H_j - H_i\right|\right) = 2$, which means that the node v_i must be 2 hops away from node v_j.

Case. 2. Applying the same justification as in Case 1, the reception slots of node v_i overlap with the transmission slots of node v_j, if we have $(2\omega+1)\cdot p - k_i{\cdot}\omega + m_{Rx}^R(v_i, \tau_p) = (2\omega+1)\cdot q - k_j{\cdot}\omega + m_{Fd}^R(v_j, \tau_q)$, where $k_i = H_i$ and $k_j = H_j$. Hence, the following relationships between the hop counts of nodes v_i and v_j can be deduced:

$$(2\omega+1)|\,p-q| + |H_p - H_q|\omega + M'/\omega = \left|\,H_j - H_i\right|, \tag{18}$$

where $0 \le M' = \left|m_{Rx}^R(v_i, \tau_p) - m_{Fd}^R(v_j, \tau_q)\right| < 2\omega$. In this case, $H_j - H_i$ has minimum value when $p - q = 1$, $H_p - H_q = 0$ and $M\prime = 0$. In this case, again we have $Min\left(\left|\,H_j - H_i\right|\right) = 2$. □

Thus to ensure interference-free slot scheduling, each channel $n \le 16$ is allocated for the transmission of all of the nodes positioned $2n + 1$ or $2n + 2$ hop counts from the gateway.

This means that channel 0 is allocated for transmission in the nodes 1 and 2 hop counts away from the gateway, while channel 1 is assigned for transmission in the nodes 3 and 4 hop counts away from the gateway, and so forth, in accordance with the allocation scheme. Thus, we apply the following equations for transmission, reception and broadcast slots for each node v_i:

$$Ch\_Tx(v_i) = \lfloor (H_i - 1) \ / \ 2 \rfloor \ , \tag{19}$$

$$Ch\_Rx(v_i) = \lfloor (H_i) \ / \ 2 \rfloor \ , \tag{20}$$

where $Ch\_Tx(v_i)$ denotes the channel assigned to node v_i for its transmissions, including transmitting its own generated packets and transmitting packets received from $v_j \in SG\text{-}Tree(v_i)$. $Ch\_Rx(v_i)$ denotes the channel assigned to node v_i for receiving packets from $v_j \in SG\text{-}Tree(v_i)$, for broadcasting its EB packets, and for receiving join requests. The allocation of channels by Equations (19) and (20) ensures that each node tunes its receiver to the channel utilized by its child node in the join request slots, reception slots and broadcast slots.

For example, Figure 3 presents the schedule of the packets generated by the network shown in Figure 1a, constructed by Auto-Sched^{U}. It can be seen that the node v_2 utilizes channel 0 for transmission, while channel 1 is used for reception. The following theorems demonstrate the collision-free characteristics of Auto-Sched.

Slot no. / Ch no.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
0	$J(v_1)$	$\beta(v_1)$	τ_1^1	τ_1^1		τ_2^2	τ_2^2	τ_2^1	τ_2^1		τ_3^2	τ_3^2	τ_3^1	τ_3^1		τ_2^2	τ_2^2	τ_2^1	τ_2^1
1				$J(v_2)$	$\beta(v_2)$		$J(v_3)$	$\beta(v_3)$	τ_3^1	τ_3^1									

Figure 3. Auto-Sched^{U} schedule constructed for the uplink packets in Figure 1a.

Theorem 2. *Auto-Sched^{U} results in collision-free autonomous slot scheduling.*

Proof. Collision occurs if at least one slot allocated to the links in the set $CNF(v_i,v_j)$ overlaps with the slot assigned to the link v_iv_j. Thus, one of the following cases occur:

Case 1. A reception slot of v_i, wherein v_i receives a packet from $v_p \in SG\text{-}Tree(v_i)$, overlaps with its transmission slot in link v_iv_j, wherein v_i forwards a packet from $v_q \in SG\text{-}Tree(v_i)$. In this case, we have $(2\omega+1)\cdot p - k_p\cdot\omega + m^R_{Rx}(v_i,\tau_p) = (2\omega+1)\cdot q - k_q\cdot\omega + m^R_{Fd}(v_i,\tau_q)$. Hence, the following relationships between the hop counts of nodes v_i and v_j can be deduced:

$$(2\omega+1)|\,p-q\,| + |\,H_p - H_q\,|\,\omega = \left| m^R_{Rx}(v_i,\tau_p) - m^R_{Fd}(v_i,\tau_q) \right|, \tag{21}$$

where $0 \le \left| m^R_{Rx}(v_i,\tau_p) - m^R_{Rx}(v_j,\tau_q) \right| \le 2\omega$. However, the minimum value of the lefthand side of the above equation is $2\omega+1$, when $p-q=1$, $H_p - H_q = 0$. This implies that the condition stated in Case 1 cannot occur. The same line of reasoning can be applied for reception and transmission slots of node v_j.

Case 2. A transmission slot of v_j, wherein v_j transmits a packet from $v_p \in SG\text{-}Tree(v_j)$, overlaps with a reception slot in node v_i, wherein v_i receives a packet from $v_q \in SG\text{-}Tree(v_j)$. Then, we have:

$$(2\omega+1)|\,p-q\,| + |\,H_p - H_q\,|\,\omega = \left| m^R_{Fd}(v_j,\tau_p) - m^R_{Rx}(v_i,\tau_q) \right| \tag{22}$$

where $1 \leq \left| m^R_{Fd}(v_j, \tau_p) - m^R_{Rx}(v_i, \tau_q) \right| \leq 2\omega$. However, the minimum value of the lefthand side of the above equation is $2\omega + 1$. This implies that the condition stated in Case 2 cannot occur. □

Theorem 3. *Auto-Sched^U results in collision-free broadcast slot.*

Proof. To prove conflict-free $B(v_S)$, we consider three cases as follows:

Case 1. $\beta(v_S)$ overlaps with the transmission/reception slots of a neighbor v_i with hop count $H_i = H_S + 1$. In this case, we have $(2\omega+1)\cdot p - (H_S+1)\cdot\omega + m^R_{Rx}(v_i, \tau_p) = (2\omega+1)\cdot S - H_S\cdot\omega - 1$, which means that $(2\omega+1)\cdot(p-S) + m^R_{Rx}(v_i, \tau_p) = \omega - 1$. However, this equality is invalid, since $-\omega \leq m^R_{Rx}(v_i, \tau_p) \leq -1$, even if $p - S = 1$ and $m^R_{Rx}(v_i, \tau_p) = -\omega$. This indicates that the neighbor nodes with next hop count would not collide with this slot.

Case 2. $\beta(v_S)$ overlaps with the transmission/reception slots of a neighbor v_i with identical hop count $H_i = H_S$. Thus we have $(2\omega+1)\cdot(p-S) + m^R_{Rx}(v_i, \tau_p) = 0$, which is invalid again. In fact, all of the nodes that utilize the channel $Ch\_Tx(v_i)$ have sequential schedules, each separated by a fixed interval of $2\omega + 1$ slots, as Figure 3 presents.

Case 3. $\beta(v_S)$ overlaps with the transmission/reception slots of a neighbor v_i with hop count $H_i = H_S - 1$. Thus, we have $(2\omega+1)\cdot(p-S) + m^R_{Rx}(v_i, \tau_p) = -\omega$ which is invalid again. Because the closest value of the lefthand side of the equation to $-\omega$ is when $p - S = -1$, and $m^R_{Rx}(v_i, \tau_p) = -1$, which results in the lefthand side being equal to -2ω. In fact, the intention behind resrving 2ω + 1 slots by Auto-Sched^U, while utilizing only 2ω slots by each node, is that the neighbors in Case 3 have no transmission/reception in the gray-colored slots in Figure 2. Then, they cannot interfere with broadcasts by v_S. For instance, in the 5th slot in the schedule shown in Figure 3, the node v_1 has no transmissions/reception to interfere with $\beta(v_2)$. □

The total number of slots required for scheduling uplink transmissions is $(2\omega+1)\cdot N_U$, where N_U denotes the number of sensor source nodes that have data packets to deliver to the gateway. The reason is that the gateway, which receives all of the uplink packets, reserves $2\omega + 1$ time slots for each packet.

4.2. Autonomous and Reliable Time Slot Allocation for Downlink Traffic

The notations used in this section are listed in Table 3.

As seen in Figure 4, Auto-Sched^D enables each actuator destination node v_D to autonomously allocate ω slots for receiving the controlling data generated by the gateway and destined for v_D, ω slots to receive join requests, and a single slot for broadcasting EB control packets. The reserved slots for reception and the slots actually used for packets destined for node v_D are given by:

$$Rx^R(v_D) = \left\{ (2\omega+1)\cdot D + H_D\cdot\omega + m^R_{Rx}(v_D) \middle| -\omega \leq m^R_{Rx}(D) \leq -1 \right\}, \tag{23}$$

$$Rx(v_D) = \{ (2\omega+1)\cdot D + H_D\cdot\omega + m_{Rx}(v_D) | -\omega \leq m_{Rx}(v_D) \leq -(ETX(v_D, pr(v_D) - 1) \}, \tag{24}$$

The ω slots to receive join requests and the slot for broadcasting EB are formulated as follows, respectively:

$$J(v_D) = \{ (2\omega+1)\cdot D + H_D\cdot\omega + m_{JR}(v_D) | 1 \leq m_{JR}(v_D) \leq \omega - 1 \} \tag{25}$$

$$\beta(v_D) = (2\omega+1)\cdot D + H_D\cdot\omega \tag{26}$$

Table 3. Definition of notations used for downlink scheduling.

Symbol	Meaning	Symbol	Meaning
$Rx^R(v_D)$	The set of time slots reserved for receiving the control packet generated by gateway and destined for destination actuator v_D, in the worst-case link quality.	$Rx^R(v_p, \tau_{S,G}{}^k)$	The set of time slots allocated to intermediate node v_p for receiving packet τ_S, due to worst-case link quality.
$Rx(v_D)$	The subset of time slots in $Rx^R(v_D)$, used for receiving the control packet generated by the gateway and destined for v_D, due to current actual link quality.	$Rx(v_p, \tau_{S,G}{}^k)$	The subset of time slots in $Rx^R(v_p, \tau_{S,G}{}^k)$ allocated to intermediate node v_p for receiving packet τ_S, due to current actual link quality.
$J(v_D)$	The set of time slots that node v_D utilizes for receiving join requests such as DAO and DIS.	$Fd^R(v_p, \tau_{S,G}{}^k)$	The set of time slots allocated to intermediate node v_p for forwarding the packet τ_S, due to the worst-case link quality.
$B(v_D)$	A time slot that that node v_D utilizes to broadcast EB control packets.	$Fd(v_p, \tau_{S,G}{}^k)$	The subset of time slots in $Fd^R(v_p, \tau_{S,G}{}^k)$ allocated to intermediate node v_p for forwarding the packet τ_S, due to the current actual link quality.

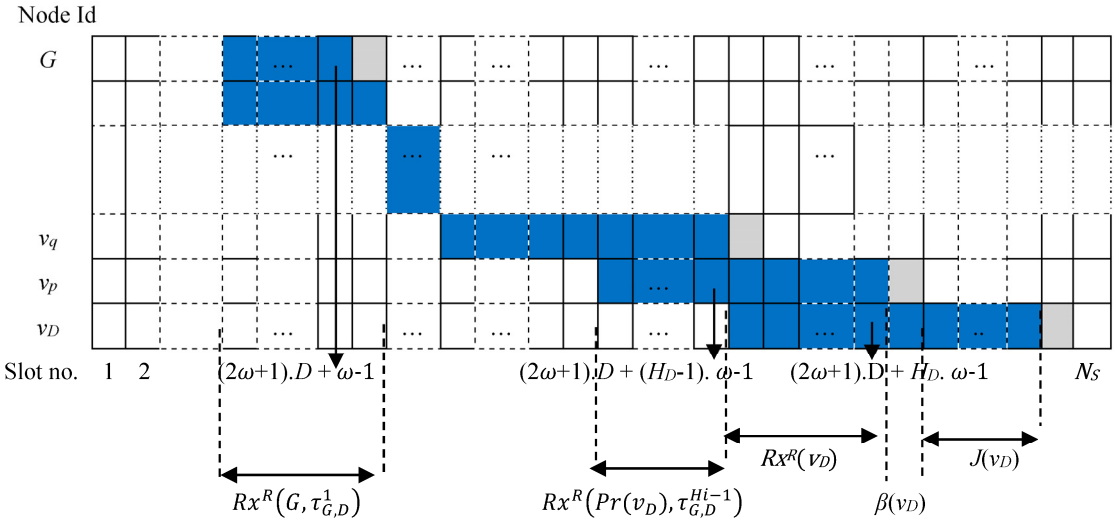


Figure 4. Timeline of Auto-Sched^D scheduling at node v_D and all intermediate nodes in its path from the gateway.

Auto-Sched^D enables each intermediate node to forward the controlling data generated by the controller node toward its destined actuator. Following the same line of reasoning in the previous section, we can derive the following equation for node v_i to forward the controlling data from the gateway toward the actuator node v_D:

$$Fd^R\left(v_i, \tau^{k+1}_{G,D}\right) = \left\{(2\omega+1)j + k{\cdot}\omega + m^R_{Fd}\left(v_i, \tau^{k+1}_{G,D}\right) \middle| 0 \le m^R_{Fd}\left(v_i, \tau^{k+1}_{G,D}\right) \le \omega - 1\right\}, \quad (27)$$

$$Fd\left(v_i, \tau^{k+1}_{G,D}\right) = \left\{(2\omega+1){\cdot}D + k{\cdot}\omega + m_{Fd}\left(v_i, \tau^{k+1}_{G,D}\right) \middle| 0 \le m_{Fd}\left(v_i, \tau^{k+1}_{G,D}\right) \le ETX(v_i, pr(v_i)) - 1\right\}, \quad (28)$$

where $\tau_{G,D}{}^{k+1}$ is the transmission of the packet $\tau_{G,D}$, which is $k+1 = H_i + 1$ hops away from the gateway. The following equation can be derived for node v_i to receive this controlling data packet from its parent node:

$$Rx^R\left(v_i, \tau^{k}_{G,D}\right) = \left\{(2\omega+1)j + k \cdot\omega + m^R_{Fd}\left(v_i, \tau^{k}_{G,D}\right) \middle| -\omega \le m^R_{Fd}\left(v_i, \tau^{k}_{G,D}\right) \le -1\right\}, \quad (29)$$

$$Rx\left(v_i, \tau^k_{G,D}\right) = \left\{(2\omega+1)j + k\cdot\omega + m_{Fd}\left(v_i, \tau^k_{G,D}\right) \middle| -\omega \le m_{Fd}\left(v_i, \tau^k_{G,D}\right) \le -(ETX(v_i, pr(v_i)) - 1)\right\} \quad (30)$$

An example of an uplink schedule is given in Figure 5. It is noteworthy that Theorems 1, 2 and 3 can be applied to both interference and collision conditions in Auto-Sched^{D}. As a result, Auto-Sched^{D} applies the Equations (19) and (20) to define the channel assignments for transmission and reception of each node.

Slot no. / Ch no.	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35
0	$\tau_{G,7}{}^1$	$\beta(v_7)$	$J(v_7)$		$\tau_{G,5}{}^1$	$\tau_{G,5}{}^1$	$\tau_{G,5}{}^2$	$\tau_{G,5}{}^2$		$\tau_{G,6}{}^1$	$\tau_{G,6}{}^1$	$\tau_{G,6}{}^2$	$\tau_{G,6}{}^2$		$\tau_{G,7}{}^1$
1									$\beta(v_5)$	$J(v_5)$				$\beta(v_6)$	$J(v_6)$

Figure 5. Auto-Sched^{D} schedule constructed for the downlink packets in Figure 1a.

4.3. Schedule Re-Construction after Link/Node Failure in Auto-Sched

Algorithm 1 describes the schedule reconstruction for the uplink schedule after a node fails to transmit/re-transmit to its parent. $NB(v_i)$ denotes the set of neighbor nodes of v_i.

Algorithm 1: Auto-Sched^{U} Schedule Reconstruction ($NB(v_i)$)

1: If (parent node failure), then
1.1: Find the best parent $v_j \in NB(v_i)$.
1.2: Send DAO to $J(v_j)$.
2: End If
3: If(received DAO in $J(v_i)$)
3.1: Delay the current packet to next slotframe.
3.2: Send the DAO within Auto-Sched^{U} data slot to gateway, instead.
3.3: Construct the new schedule for the new child, by Equations (11), (12), (15) and (16).
4: End If.
5: If(received DAO in Auto-Sched^{U} data slot)
5.1: Send the DAO within Auto-Sched^{U} data slot to gateway.
5.2: Construct the new schedule for the new child, by Equations (13)–(16).
6: End If.

Line 1 states that when node v_i selects node $v_j \in NB(v_i)$ as parent, then v_i uses a slot in $J(v_j)$ to send DAO to v_j. When v_i receives a DAO, it delays its generated message to next slotframe and utilizes the corresponding slots to send the DAO packet, line 3. Thus, the main idea is to delay the current generated packet to next slotframe and instead send the DAO to the gateway to construct the new schedule for the requesting node within one slotframe. When v_i receives a DAO in a Auto-Sched^{U} data slot, v_i then configures new slots for forwarding the packets generated by the requesting source node, by Equations (13)–(16).

This enables reliability and determinism for DAO control packets, facilitating RPL operations in managing join or parent change requests. Specifically, when a node changes its parent, the DAO packet is forwarded through the new path and constructs the new schedule before the end of the current slotframe, enhancing robustness of the network.

Algorithm 2 describes the schedule reconstruction for the downlink schedule after a node v_i sends DAO to its new parent. Since in the downlink graph, an actuator node does not have an uplink route toward the gateway, the main idea is to only use $J(v_j)$ of each intermediate node to forward the DAO. In this case, H_i slotframes are required to construct the new schedule.

Algorithm 2: Auto-Sched^{D} Schedule Re-Construction ($NB(v_i)$)

1: If (no packet received after a time out), then
    1.1: Find the best parent v_j.
    1.2: Send DAO to $J(v_j)$.
2: End If
3: If(received DAO in $J(v_i)$)
    3.1: Forward the DAO within $J(P(v_i))$.
    3.2: Construct the new schedule for the new child, by Equations (27)–(30).
4: End If.

In the best scenario, when the unique identifiers of the nodes in the downlink network are allocated sequentially from top to bottom of the network in descending order, the DAO packet will arrive to the gateway within one slotframe. This is due to the fact that the $J(P(v_j))$ of each intermediate node is allocated after $J(v_j)$. Thus, for applications that require real-time schedule re-construction in the downlink, it is preferable to apply ID assignment approaches. This problem is out of the scope of this paper and can be found in [22] and references therein.

5. Experimental Results

5.1. Simulation Parameters

This section investigates the performance of Auto-Sched compared to SchedEx in [13], OrchEx in [17] and Escalator in [15], in terms of delay bounds and reliability. The end-to-end delays are measured with respect to the average delay for end-to-end response times of generated packets within a packet generation time interval, while the reliability is measured in terms of PDR and the robustness of the approaches against failures. PDR is defined as the ratio between the number of packets delivered to the destination compared with the number of packets generated in the network within a packet generation interval.

To enhance reliability, SchedEx adopts graph routing, facilitating path diversity and enhancing robustness in the event of node/link failure. To do so, during the network initialization phase, each node selects two preferred parents to serve as forwarding nodes. When transmission to the primary parent fails, the node forwards the packet to the secondary parent. Each node, autonomously and based on its identifier, allocates two dedicated time slots to transmit/retransmit a single packet to the best parent, with an additional dedicated time slot to retransmit it to the second-best parent. In addition to SchedEx, within this section, we evaluate the efficiency of our refined autonomous scheduling derived from SchedEx, denoted as SchedEx-M. In the event of successful delivery of the packet in the first or second slot, SchedEx-M enables the node to utilize the remaining slots to transmit and retransmit the second or third packet.

OrchEx, as explained in Section 2, introduces adaptive mechanisms, in order to manage high traffic load or traffic bursts. By default, all of the nodes are allocated a reception time slot and a channel offset by a hash function. When the buffer of a child node exceeds a given threshold (5 packets is defined as threshold in reference [17]), it initiates a notification to the gateway. This notification is embedded within the data packet of the respective child node. In this case, the gateway node adds more reception time slots to the respective child node, with the number of added slots being proportional to the size of its sub-tree. In accordance with [17], when sub-tree size falls between 3 to 5 nodes, then the gateway adds two more reception time slots for that particular node, through a hash function. However, when the sub-tree size exceeds 5 nodes, then the gateway includes 3 additional reception time slots for that node.

Similarly to Auto-Sched, the scheduling approach by Escalator provides sequential packet transmission, but Escalator assumes a network with reliable links and thus does not provide retransmission opportunities to nodes. Additionally, Auto-Sched uses the joining slot, i.e., $J(v_i)$, for receiving DAO, as stated in Section 3. However, both Escalator and SchedEx utilize a single shared slot for all RPL control packets. This shared slot

is defined within a separate slotframe specified for RPL routing control packets. Such configuration might be optimal at the network steady state, as it minimizes the number of timeslots allocated for control messages [23]. But, as our experiments show, it might increase significantly the number of collisions under any network change scenario, as several DIO and DAO control packets are generated in a short interval.

In Escalator, when a collision occurs between the application data and the routing control packet, the data packet is delayed to the subsequent slotframe. However, SchedEx autonomously defers all data traffic at conflict to a conflict-free slot within the slotframe. However, OrchEx does not elaborate on this collision problem. Thus, similarly to Orchestra, we assume that OrchEx drops the data packet when collision occurs. Similarly to Escalator and SchedEx, Auto-Sched adopts the concept of this shared slot within the routing slotframe. However, in contrast to Escalator and SchedEx, where this single slot serves for various purposes in the RPL layer, Auto-Sched employs this shared slot exclusively for broadcasting DIO messages. In the event of a collision between the shared slot and the application data schedules, Auto-Sched applies the strategy defined by Escalator to delay the conflicting data packet to the subsequent slotframe.

The simulation was conducted using the Cooja simulator [24], a Java-based tool that operates in conjunction with the Contiki operating system, developed for resource-constrained IoT devices. In all of the experiments, Tmote sky devices, comprising a CC2420 radio transceiver with a data transfer rate of 250 kbit/s using IEEE 802.15.4 MAC and physical layer specification, were randomly deployed, with one device designated as the gateway. The nodes were assumed to employ RPL as the routing protocol for the Auto-Sched and Escalator protocols, while the approach in SchedEx utilized the graph routing protocol, as defined in [13]. For the link between each pair of neighbor nodes, we assigned a random packet reception rate in the range [25–100]%, based on the distance between them. In all routing protocol scenarios, the ETX reliability constraint of the candidate parent node was configured to a maximum threshold, permitting a maximum of three retransmissions.

To conduct fair experiments, the length of the slotframe for each approach was defined to ensure that the slotframe contained all of the slots required by each approach. For example for n nodes, slotframe lengths of $2n$, $3n$ and $7n$ slots were defined for Escalator, SchedEx and Auto-Sched, respectively. As for OrchEx, the size of the slotframe was defined by $n + 3n_c$, where n_c is the number of child nodes of the gateway. This was due to the consequence of the gateway's response to the high-traffic load scenarios, wherein it allocated three additional time slots to each of its immediate child nodes. In all experiments, the transmission range was 50 m, while the interference range was 60 m, and at most 20 hop counts were allowed. The length of the routing slotframe was set to 47 time slots, as specified by SchedEx. Unless otherwise noted, the reported results are averages of the results with 100 randomly generated networks.

5.2. Performance under Interference

Figure 6 illustrates the PDR as the number of field devices varied in the network from 50 to 250 nodes, with the packet generation interval in all nodes set to 5 s, 10 s and 20 s, respectively. In the context of the IEEE 802.15.4e TSCH standard, where each slot length is 10 ms, the aforementioned specified packet generation intervals corresponded to 500, 1000 and 2000 time slots, respectively. From the results in Figure 6a, we observe that Auto-Sched obtained 100% PDR in the networks with 50 nodes in all experiment iterations. This is due to the fact that 350 out of 500 available time slots were utilized for complete delivery of all packets. Therefore, before the next packet generation period, all of the current generated packets were delivered to the gateway. However, with a packet generation interval of 5 s, as the number of nodes increased, the efficiency of Auto-Sched diminished to 30% for a network with 250 nodes. This is mainly due to the insufficient number of available time slots to guarantee the reliable delivery of packets, which led to packet drop in intermediate nodes.

The poor efficiency exhibited by SchedEx shown in Figure 6 is attributed to its slot allocation strategy, where each node had only one opportunity to reliably forward a single packet in each slotframe. For 50 nodes, when the packet generation interval was 5 s, SchedEx provided a maximum of $500/150 \approx 3$ opportunities for each intermediate node to forward the packets. This fact restricted the immediate neighbor of the gateway to forwarding only three packets to the gateway. The PDR was about 25%, and as the number of nodes increased, there was a noticeable rise in packet drops. Therefore, despite leveraging multi-path routing in the RPL layer, the poor slot allocation restricted the reliability and scalability of the approach. However, SchedEx-M displayed higher PDR in each experiment, as the node utilized the allocated slots for transmitting second or third buffered packets.

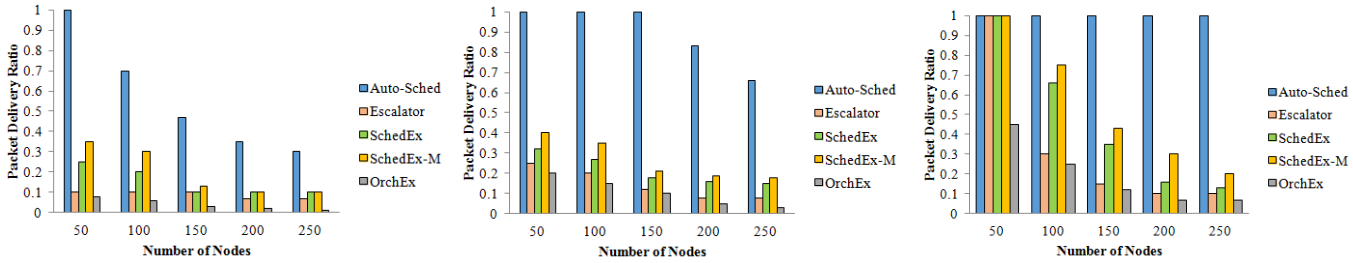


Figure 6. Packet delivery ratio by varying packet generation intervals and number of nodes in the network. The packet generation interval is (**a**) 5 s, (**b**) 10 s and (**c**) 20 s.

OrchEx demonstrated slightly lower PDRs in comparison to SchedEx. This is mainly because OrchEx adopted the receiver-based policy of Orchestra, which led to increasing collisions. Additionally, OrchEx utilized RPL for routing, while SchedEx leveraged graph routing and took advantage of rout diversity. Furthermore, in OrchEx, the additional reception time slots were only assigned to immediate child nodes of the gateway. Consequently, as the network scaled with the increasing number of nodes, the packet drop ratio worsened due to the inadequate number of time slots allocated to each individual node in the network. Another reason is that, in OrchEx, when a collision occurred between a data packet and the routing slotframe, the data packet was forced to be dropped. However, a slight improvement in OrchEx's performance was observed when packet generation intervals were extended from 5 s to 10 s and 20 s. For instance, for a network comprising 50 nodes, the PDRs demonstrates an increase, rising from 10% when employing a 5 s packet generation interval to approximately 50% when the packet generation interval was 20 s.

Despite the sequential forwarding strategy by Escalator, its efficiency fell slightly lower than that demonstrated by SchedEx. In scenarios where a node's link necessitates retransmissions, the sequential forwarding of the packets generated by children was disrupted and delayed for the subsequent slotframe. In the worst-case scenario, when all of the links through a path that consists of k hops need m retransmissions for guaranteeing reliability, the number of slotframes required to deliver a packet becomes $k.m$ slotframes. When the packet generation interval is less than $k.m$, the intermediate nodes experience buffer overload and ultimately packet drop.

As Figure 6c shows, with a packet generation interval increased to 20 s, the performance of all approaches increased strongly for less than 100 nodes in the network. However, for more nodes, these approaches attained low performance. SchedEx provided better performance, as it ensures reliable hop-by-hop transmission at each slotframe and takes advantage of multi-path routing. Regarding Auto-Sched, it achieved a flawless PDR of 100% across all network sizes. Furthermore, Auto-Sched consistently maintained a 100% PDR for a packet generation interval of 10 s within all networks containing fewer than 150 nodes.

Figure 7 illustrates the cumulative density function (CDF) of the average end-to-end delay of delivered packets in the network with 50 nodes, where the packet generation period in all nodes is 5 s, 10 s and 20 s, respectively. Escalator performance was eliminated

from this result, as it showed significantly higher end-to-end delays. This delay was mainly due to the long length of each slotframe, while retransmission slots were not allocated at each slotframe. From Figure 7a, it can be seen that when the packet generation interval was 5 s, SchedEx resulted in an average end-to-end delay of 5 s for almost 10% of packets, while SchedEx-M delivered around 20% of packets within 5 s. However, OrchEx delivered less than 1% of packets in 5 s. When the packet generation interval was 10 s, approximately 25% of the packets had an end-to-end delay of 10 s by SchedEx, while that it was 30% for SchedEx-M and 10% for OrchEx. For a packet generation interval of 20 s, SchedEx and SchedEx-M delivered 70% and almost 85% of packets, respectively, within 20 s, while OrchEx resulted in an average end-to-end delay of 20 s for almost 30% of the packets. Auto-Sched provided consistent, predictable average end-to-end delays, which were less than the packet generation intervals (or deadlines). This was mainly due to its deterministic scheduling to guarantee reliability constraints within each slotframe.

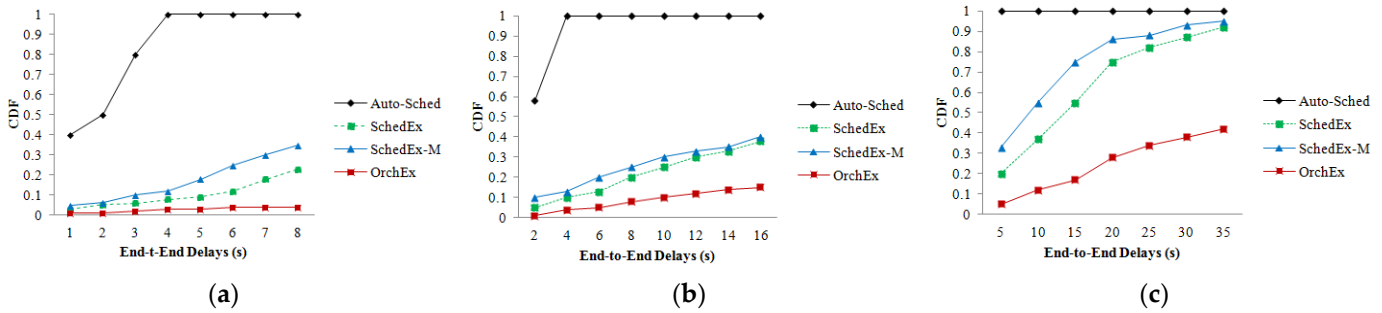
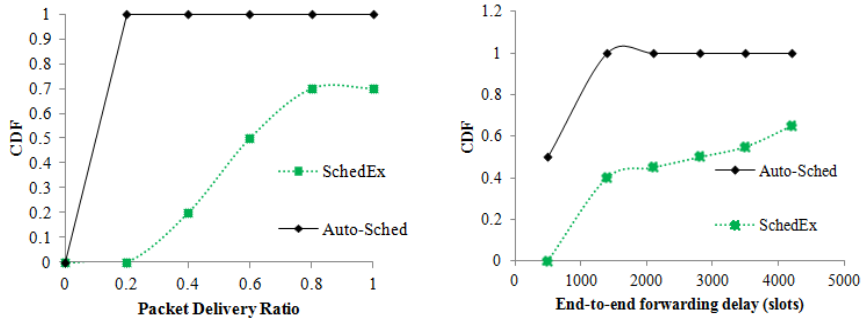


Figure 7. Average end-to-end delay of packets for network with 50 nodes, where packet generation interval is (**a**) 5 s, (**b**) 10 s and (**c**) 20 s.

5.3. Performance under Node Failure

Within the randomly generated networks consisting of 50 nodes operating under a packet generation interval of 20 s, in Figure 6c, 2–4 nodes located on randomly selected routing paths were turned off. We repeated the experiments 100 times with different sets of failing nodes. Figure 8 shows the CDF of the PDR and end-to-end delays for path reconstruction by each approach. Notably, a consistent 100% PDR was observed for Auto-Sched, due to its deterministic schedule provided for DAO packets. Furthermore, as mentioned in Section 3, Auto-Sched facilitates forwarding DAO throughout the new path and constructs a new schedule within a single slotframe. However, the approaches in SchedEx and Escalator provided a single shared slot for all control packets of RPL. In addition, in scenarios when a node sends a DAO to the preferred parent by utilizing a channel within the shared slot, the parent transceiver could be tuned to an alternate channel. Such latency issues are introduced for each intermediate node forwarding the DAO, causing increased packet drops and delays for nodes further from the gateway. In addition, collisions between DAO packets with data packets in SchedEx caused frequent schedule deferral by the nodes, which ultimately leading to increased packet drops.

Evidently, when a sensor or actuator node joins the network or when a node detects that a link to the parent is unreliable, our results as shown in Figure 8 imply that Auto-Sched handles it autonomously with minimum latency, packet drops and perfect PDR. This property of Auto-Sched shows a significant improvement over existing approaches, which incur data packet drops and delay issues due to utilizing a single shared slot in the routing slotframe. This key feature positions Auto-Sched as an empowering option for facilitating the scheduling of network changes in real-time applications.



(**a**) (**b**)

Figure 8. Distribution of average PDR and end-to-end delays in a network with 50 sensor nodes, where the packet generation interval is 20 s. (**a**) Distribution of average PDR. (**b**) Distribution of average end-to-end delays.

5.4. Discussion

In our simulations, we assumed that the end-to-end deadline of each packet equaled its generation period. This means that when a packet is generated, it must be delivered to the gateway prior to the generation of its next instance. The significance of this assumption is attributed to real-time applications, wherein end-to-end delays are constrained by upper bounds (i.e., deadlines), e.g., tens of milliseconds for discrete manufacturing, seconds for process control, and minutes for asset monitoring [25]. However, ensuring hop-by-hop schedule reliability introduces a trade-off wherein the pursuit of higher reliability results in longer delays due to the allocation of additional time slots.

For all simulated network configurations, the high performance of Auto-Sched was notable, as Auto-Sched effectively managed data reliability and real-time response times of varying network sizes and packet generation intervals. For instance, for a tight packet generation interval of 5 s, under Auto-Sched, a network comprising 50 nodes is schedulable (i.e., all packets meet the deadlines); for an intermediate packet generation interval of 10 s, a network comprising 150 nodes is schedulable; and for a large packet generation interval of 20 s, a network comprising 250 nodes is schedulable, since Auto-Sched achieves optimal PDR within the deadlines. This makes Auto-Sched a potentially attractive choice for enabling real-time applications to take advantage of the simplicity of autonomous scheduling while guaranteeing hop-by-hop reliability, end-to-end deadlines, and handling of all network dynamics autonomously with negligible communications or control overhead. Specifically, as demonstrated by our performance analysis, Auto-Sched has the capacity to manage large-scale networks with packet generation intervals of multiple seconds, which makes it an optimal option for managing process control and asset monitoring applications. In conclusion, in an industrial environment where timeliness and reliability are critical constraints, our proposed approach is a promising solution to facilitate the scheduling and management of packet deliveries and network changes.

6. Conclusions

This paper introduced an autonomous scheduling approach named Auto-Sched, which exhibit the following key features: First, Auto-Sched is designed to enhance autonomous and reliable data delivery. It achieves this by enabling each node to autonomously allocate transmission/retransmission time slots for all buffered packets. Second, our performance analysis demonstrated that Auto-Sched ensures real-time end-to-end data delivery, such that each packet is delivered to its destination within its generation period. Third, Auto-Sched is designed to enhance robustness against potential node or link failures. A simple yet efficient algorithm is proposed to facilitate the propagation of join requests through the new path. Fourth, Auto-Sched provides autonomous and reliable scheduling of both downlink and uplink schedules. Our simulation results demonstrate the reliability and real-time response times for networks with 250 nodes when the packet generation interval

is 20 s, for networks with 150 nodes when the packet generation interval is 10 s, and for networks with 50 nodes when the packet generation interval is 5 s. This demonstrates that Auto-Sched is an optimal option for managing process control and asset monitoring applications, where end-to-end delays are constrained by multiple tens of seconds.

Author Contributions: Investigation, A.D.; Project administration, M.-K.K. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by Institute of Information & communications Technology Planning & evaluation (IITP) grant funded by the Krea government (MSIT) (RS-2022-00155933, IoT-based Intelligent Smart Dust Collection Platfrom for Electric Dust Collector).

Data Availability Statement: Data is contained within the article.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Willig, A. Recent and emerging topics in wireless industrial communications: A selection. *IEEE Trans. Ind. Informat.* **2008**, *4*, 102–124. [CrossRef]
2. Miorandi, D.; Uhlemann, E.; Vitturi, S.; Willig, A. Guest editorial: Special section on wireless technologies in factory and industrial automation, Part I. *IEEE Trans. Ind. Informat.* **2008**, *3*, 95–98. [CrossRef]
3. Salvadori, F.; Gehrke, C.S.; de Oliveira, A.C.; de Campos, M.; Sausen, P.S. Smart grid infrastructure using a hybrid network architecture. *IEEE Tran. Smart Grid* **2013**, *4*, 1630–1639. [CrossRef]
4. Ghayvat, H.; Liu, J.; Mukhopadhyay, S.C.; Gui, X. Wellness sensor networks: A proposal and implementation for smart home for assisted living sign in or purchase. *IEEE Sens. J.* **2015**, *15*, 7341–7348. [CrossRef]
5. IETF 6TiSCH Working Group. Available online: https://datatracker.ietf.org/wg/6tisch/ (accessed on 25 August 2023).
6. Winter, T.; Thubert, P.; Brandt, A.; Hui, J.W.; Kelsey, R.; Levis, P.; Pister, K.; Struik, R.; Vasseur, J.P.; Alexander, R.K. RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks. Available online: https://tools.ietf.org/html/rfc6550 (accessed on 25 August 2023).
7. Gnawali, O.; Levis, P. The Minimum Rank with Hysteresis Objective Function. IETF 2012, RFC 6719. Available online: https://tools.ietf.org/html/rfc6719 (accessed on 25 August 2023).
8. *IEEE 802.15.4e*; IEEE Standard for Local and Metropolitan Area Networks–Part 15.4: Low-Rate Wireless Personal Area Networks (LRWPANs) Amendment 1: MAC Sublayer. IEEE 802.15.4e Task Group: Piscataway, NJ, USA, 2012.
9. Jung, J.; Kim, D.; Hong, J.; Kang, J.; Yi, Y. Parameterized slot scheduling for adaptive and autonomous TSCH networks. In Proceedings of the IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), Honolulu, HI, USA, 15–19 April 2018.
10. Yan, M.; Lam, K.-Y.; Han, S.; Chan, E.; Chen, Q.; Fan, P.; Chen, D.; Nixon, M. Hypergraph-based data link layer scheduling for reliable packet delivery in wireless sensing and control networks with end-to-end delay constraints. *Inf. Sci.* **2014**, *278*, 34–55. [CrossRef]
11. Hashimoto, M.; Wakamiya, N.; Murata, M.; Kawamoto, Y.; Fukui, K. End-to-end reliability- and delay-aware scheduling with slot sharing for wireless sensor networks. In Proceedings of the International Conference on Communication Systems and Networks (COMSNETS), Bangalore, India, 5–10 January 2016.
12. Yang, D.; Xu, Y.; Wang, H.; Zheng, T.; Zhang, H.; Zhang, H.; Gidlund, M. Assignment of segmented slots enabling reliable real-time transmission in industrial wireless sensor networks. *IEEE Trans. Indust. Elec.* **2015**, *62*, 3966–3977. [CrossRef]
13. Shi, J.; Sha, M.; Yang, Z. Distributed graph routing and scheduling for industrial wireless sensor-actuator networks. *IEEE/ACM Trans. Netw.* **2019**, *27*, 1669–1682. [CrossRef]
14. Duquennoy, S.; Al Nahas, B.; Landsiedel, O.; Watteyne, T. Orchestra: Robust mesh networks through autonomously scheduled TSCH. In Proceedings of the 13th ACM in Embedded Networked Sensor Systems, Seoul, Republic of Korea, 1–4 November 2015.
15. Oh, S.; Hwang, K.; Kim, K.H.; Kim, K. Escalator: An autonomous scheduling scheme for convergecast in TSCH. *J. Sens.* **2018**, *18*, 1209. [CrossRef] [PubMed]
16. Kim, S.; Kim, H.; Kim, C. ALICE: Autonomous link-based cell scheduling for TSCH. In Proceedings of the 18th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN), Montreal, QC, Canada, 16–18 April 2019.
17. Deac, D.; Teshoma, E.; Van Glabeek, R.; Doborota, V.; Breaken, A.; Steenhaut, K. Traffic aware scheduler for time-slotted channel-hopping-based IPv6 wireless sensor networks. *J. Sens.* **2022**, *22*, 6397. [CrossRef] [PubMed]
18. Osman, M.; Nabki, F. OSCAR: An optimized scheduling cell allocation algorithm for convergecast in IEEE 802.15.4e TSCH networks. *J. Sens.* **2021**, *21*, 2493. [CrossRef] [PubMed]
19. Zhang, T.; Gong, T.; Han, S.; Deng, Q.; Hu, X.S. Fully Distributed Packet Scheduling Framework for Handling Disturbances in Lossy Real-Time Wireless Networks. *IEEE Trans. Mob. Comput.* **2021**, *20*, 502–518. [CrossRef]
20. Elsts, A.; Kim, S.; Kim, H.S.; Kim, C. An emprical survey of autonomous scheduling methods for TSCH. *IEEE Access* **2020**, *8*, 67147–67165. [CrossRef]

21. Levis, P.; Patel, N.; Culler, D.; Shenker, S. Trickle: A selfregulating algorithm for code propagation and maintenance in wireless sensor networks. In Proceedings of the 1st USENIX/ACM Symposium on Networked Systems Design and Implementation, San Francisco, CA, USA, 29–31 March 2004.
22. Vall, E.O.A.; Blough, D.; Ferri, B.H.; Rilley, G. Distributed global ID assignment for wireless sensor networks. *Ad Hoc Netw.* **2009**, *7*, 1194–1216. [CrossRef]
23. Vallati, C.; Brienza, S.; Anastasi, G.; Das, S.K. Improving network formation in 6TiSCH networks. *IEEE Ttrans. Mob. Comput.* **2019**, *18*, 98–110. [CrossRef]
24. Osterlind, F.; Dunkels, A.; Eriksson, J.; Finne, N.; Voigt, T. Cross-level sensor network simulation with COOJA. In Proceedings of the 31st IEEE Conference on Local Computer Networks, Tampa, FL, USA, 14–16 November 2006.
25. Zurawski, R. Networked embedded systems: An overview. In *Networked Embedded Systems*; Zurawski, R., Ed.; CRC Press: Boca Raton, FL, USA, 2009; Chapter 1; pp. 1.11–1.16.