Topology Management and TSCH Scheduling for Low-Latency Convergecast in In-Vehicle WSNs

Rasool Tavakoli, *Student Member, IEEE,* Majid Nabi, *Member, IEEE,* Twan Basten, *Senior Member, IEEE,* and Kees Goossens, *Member, IEEE*

***Abstract*—Wireless Sensor Networks (WSNs) are considered as a promising solution in intra-vehicle networking to reduce wiring and production costs. This application requires reliable and real-time data delivery, while the network is very dense. The Time-Slotted Channel Hopping (TSCH) mode of the IEEE 802.15.4 standard provides a reliable solution for low-power networks through guaranteed medium access and channel diversity. However, satisfying the stringent requirements of in-vehicle networks is challenging and demands for special consideration in network formation and TSCH scheduling. This paper targets convergecast in dense in-vehicle WSNs in which all nodes can potentially directly reach the sink node. A cross-layer Low-Latency Topology management and TSCH scheduling (LLTT) technique is proposed that provides a very high timeslot utilization for the TSCH schedule and minimizes communication latency. It first picks a topology for the network that increases the potential of parallel TSCH communications. Then, by using an optimized graph isomorphism algorithm, it extracts a proper match in the physical connectivity graph of the network for the selected topology. This network topology is used by a light-weight TSCH schedule generator to provide low data-delivery latency. Two techniques, namely grouped retransmission and periodic aggregation, are exploited to increase the performance of the TSCH communications. The experimental results show that LLTT reduces the end-to-end communication latency compared to other approaches, while keeping the communications reliable by using dedicated links and grouped retransmissions.**

***Index Terms*—Intra-vehicle networks, Wireless Sensor Networks, Topology Management, Time-Slotted Channel Hopping, TSCH, Scheduling, Low-Latency, Industrial WSNs.**

I. Introduction

Industrial Wireless Sensor Networks (WSNs) are usually used for supervisory control, in which tens to hundreds wireless nodes are distributed over a platform to send data samples to a central node (sink node). For small size platforms such as vehicles, all wireless sensor nodes are usually in one-hop distance of each other, and are able to communicate directly with the sink node. In such a platform, a star topology may be used to form the required convergecast network. Due to the high density of nodes in these WSNs, using contention-based Medium Access Control (MAC) protocols may lead to considerable internal interference though. The IEEE 802.15.4 Time-Slotted Channel Hopping (TSCH) protocol [1] provides Time-Division Multiple Access (TDMA)-based communications for low-power WSNs to prevent such internal interference. This technique has lower communication and power overhead compared to the contention-based IEEE 802.15.4 MAC, due to use of guaranteed communications. It divides time into fixed time periods called *timeslots* that each is enough to transmit a single packet and its acknowledgement. A number of timeslots are grouped into a *slotframe* that repeats over time. TSCH also enables parallel communications using multiple *Channel Offsets* (and multiple slotframes) through a channel hopping technique. A Channel Offset determines the physical channels that should be used for transmissions in a given timeslot. Assignment of timeslots and channel offsets to the links and extracting a communication schedule, called scheduling, is left to the upper layers in the protocol stack (i.e., a sublayer between network and MAC layers).

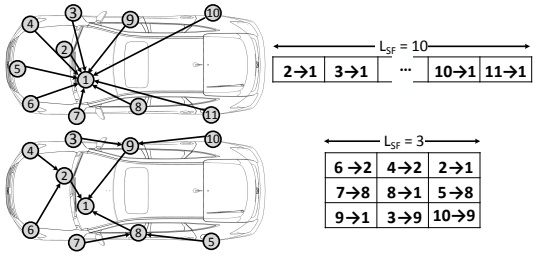


Fig. 1. TSCH schedule for a star topology compared to a tree topology (L_{SF} is the slotframe length).

The communication schedule has a high impact on the performance of the WSN. There are several TSCH scheduling algorithms such as [2], [3], [4], [5], [6], [7] that aim at improving reliability and/or latency. These scheduling algorithms usually assume the network communication topology (that is determined by the network layer) as their input, and their output differs for different input topologies. When a WSN is small and dense, a wide range of topologies can be used for it. A suboptimal topology may lead to larger slotframes that

Manuscript received February 2, 2018; revised April 30, 2018; accepted June 12, 2018. Date of publication xxxx x, xxxx; date of current version July 5, 2018. This work was partially supported by the SCOTT and ENABLE-S3 European projects, that have received funding from the ECSEL Joint Undertaking under grant agreements no. 737422 and 692455-2, respectively. (Corresponding author: Rasool Tavakoli)

R. Tavakoli and K. Goossens are with the Department of Electrical Engineering, Eindhoven University of Technology, P.O. Box 513, 5600 MB Eindhoven, the Netherlands (e-mail: r.tavakoli@tue.nl; k.g.w.goossens@tue.nl).

M. Nabi is with the Department of Electrical Engineering, Eindhoven University of Technology, P.O. Box 513, 5600 MB Eindhoven, the Netherlands and Department of Electrical and Computer Engineering, Isfahan University of Technology, Isfahan 84156-83111, Iran (email: m.nabi@tue.nl)

T. Basten is with the Department of Electrical Engineering, Eindhoven University of Technology, P.O. Box 513, 5600 MB Eindhoven, the Netherlands and ESI, TNO, P.O. Box 6235, 5600 HE Eindhoven, the Netherlands (email: a.a.basten@tue.nl)

Digital Object Identifier 10.1109/TII.2018.xxxxxxx

causes higher end-to-end communication latency, or unnecessary multi-hop communications that lead to lower reliability. As an example (see Fig. 1), using the TSCH MAC together with a star topology leads to a long slotframe and high packet delivery latency when the number of nodes is increased. On the other hand, a tree topology can reduce the slotframe length and accordingly the latency of communications by enabling parallel communications. This makes the topology management and link scheduling challenging for dense WSNs.

In order to take advantage of parallel communications of TSCH to reduce the data delivery latency, we propose the Low-Latency Topology management and TSCH scheduling (LLTT) technique. It is a cross-layer design which picks a proper network topology at the network layer to maximize the TSCH schedule utilization for the MAC layer. To match the extracted topology with the physical connectivity graph of the network, an optimized sub-graph isomorphism [8] algorithm is proposed. In the matching process, the quality of links and power plane of nodes are taken into account. The resulting network topology graph is used as the input of a light-weight TSCH schedule generator that minimizes the multi-hop communication latency. We moreover use particular shared timeslots for retransmission, called grouped shared timeslots, in the schedule to increase the reliability of the multi-hop communications. Furthermore, a periodic aggregation technique is exploited to efficiently use the available bandwidth of multi-hop links. LLTT has polynomial complexity; it makes the TSCH schedule and accordingly behavior of the WSN predictable, targeting applications that require a prior knowledge of the network behavior. In conclusion, the goal of this paper is to minimize the average latency of guaranteed data convergecast in a small size and dense TSCH network. The proposed technique targets industrial automation applications, such as use cases in automotive, or factory automation. Wireless in-vehicle networks are taken as a representative case study in this paper. Experimental results show that LLTT achieves higher reliability compared to a star topology when latency bounds are defined for data validity. For high data generation rates, it provides lower average end-to-end data delivery latency, even compared to the TSCH minimal schedule [9].

The paper is organized as follows. The related work about topology management and TSCH scheduling is overviewed in the next section. The two main building blocks of LLTT, namely topology management and low-latency scheduling, are introduced in Sections III and IV. Section V introduces the exploited aggregation technique and gives an analysis of latency in the proposed mechanisms. Implementations and experimental setups are presented in Section VI. The achieved results are discussed in Section VII. Section VIII concludes.

II. Related Work

Considering the impact of in-vehicle WSNs on reducing the weight and manufacturing costs of vehicles, the design of efficient and reliable wireless in-vehicle networks is getting more and more research attention. In [10], authors show that under high traffic loads, congestion plays an important role in the performance of single channel WSNs. Authors propose to use multiple base stations to mitigate this problem. The authors of [11] show that using multi-hop networking for data convergecast in a vehicle improves the performance compared to single-hop convergecast. However, as a single channel and contention-based MAC protocol is used, multi-hop communications together with low transmission powers reduce the local contention in the network. This is the main reason for the observed performance gain. On the other hand, multi-hop communications increase the average transmission count, which leads to higher energy consumption. In [12], authors use extensive real-world experiments and show that an in-vehicle WSN can be separated into different zones, considering the behavior of wireless channels and link qualities. This can be used to improve in-vehicle communication protocols. In [13], Volvo group trucks technology presents a practical design of an in-vehicle WSN. They use the IEEE 802.15.4 TSCH protocol [1] as the MAC protocol for their experiments and show that this protocol is sufficiently robust to host low-to-medium time criticality. This robustness is due to the use of guaranteed communications and channel hopping. This work uses a network with only 10 nodes, while vehicles have the potential to use a much higher number of wireless sensors. For instance, a high end truck can have around 150 sensors and assuming that 20% of this number can be migrated to short range wireless links, we would have a WSN with a node population of around 30 sensor nodes [14]. Compared to single channel protocols, TSCH is a better candidate for dense networks. This is because TSCH increases the network throughput by enabling parallel communications on multiple channels. This requires a network topology with links with different sources and destinations. This can be reached by defining multiple paths in the network that are at least two hops.

The TSCH scheduling task is an NP-hard problem [15]. Thus, all the available scheduling algorithms use suboptimal solutions, targeting specific performance parameters. For instance, AMUS [5] is a centralized scheduling technique that reserves sequential timeslots for the set of links along an end-to-end route to reduce latency of multi-hop communications. This increases the scheduling complexity for dense networks in which the number of neighbors of each node is very high. [16] proposes a latency-optimal scheduling algorithm for convergecast in TSCH-based WSNs. The authors show that the network topology affects the scheduling result and the end-to-end communication latency.

The distributed scheduling techniques such as Orchestra [3] and DeTAS [6] mainly focus on scalability and cannot guarantee a latency bounded schedule, because global network information is unavailable. However, some distributed scheduling techniques target low latency communications. For instance, Wave [17] aims to minimize latency by reducing the schedule length, using a repeated conflict-free schedule called wave.

All the scheduling algorithms that are proposed for the IEEE 802.15.4 TSCH protocol (e.g., [3], [4], [5], [6], [7], [17], [18]) require the network topology as an input and have no control on it. Thus, they cannot guarantee a predefined bound for the communication latency.

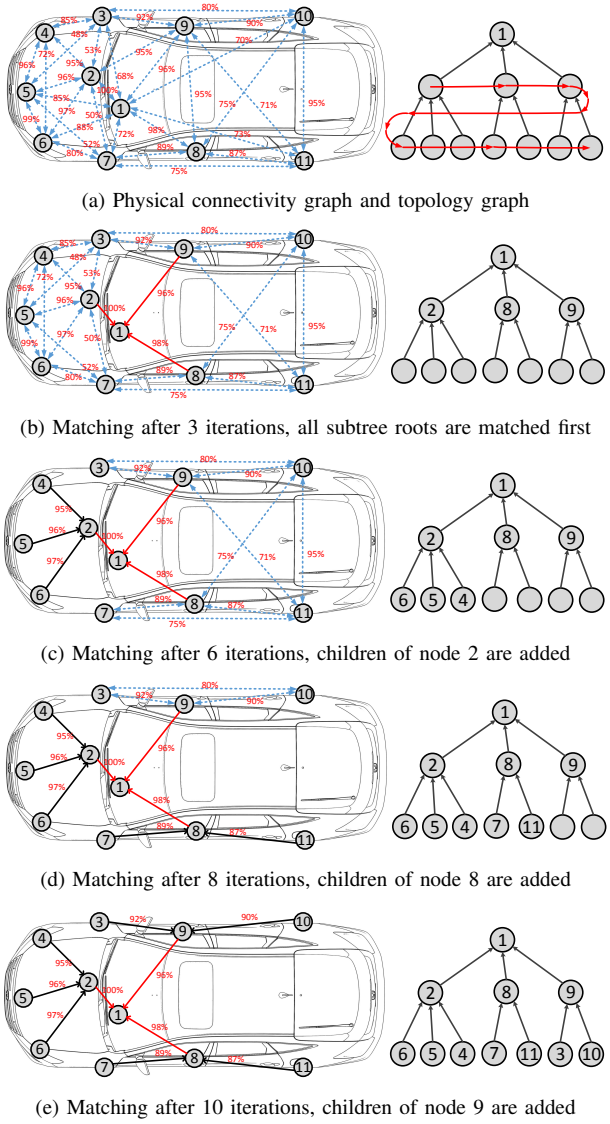
Network topology management is considered a task of the network layer. Topology management techniques typically aim to find a set of links to construct the network in order to provide energy efficiency [19], [20], [21], [22], delay bounds [23], and/or handling node failure [24]. To the best of our knowledge, there is no topology management technique that takes TSCH parallel communications into account, in order to increase the TSCH schedule utilization and reduce communication delay. As a cross-layer technique, LLTT manages network topology based on the number of nodes in the WSN and is able to provide a prior estimation of the resultant schedule and communication latency. Energy efficiency is not playing a role in our work, as in many automotive applications, energy consumption of wireless nodes is of limited importance.

Using data aggregation for data collection in WSNs may follow a tree-based or cluster-based approach [25]. In both approaches, an aggregation point (e.g., cluster head) normally waits until it receives data from a set of active sources and then aggregates the data and forwards them to the sink node [26]. DICA [7] is a distributed data aggregation technique for single channel networks that uses data aggregation to employ the available bandwidth efficiently. This technique intertwines the network tree formation to reduce latency for aggregation. In this work, we use a periodic aggregation technique in which a period equal to a TSCH slotframe is used to aggregate data in a subtree root. This technique manages the packet generation rate based on the available bandwidth in the TSCH MAC layer.

Allocating shared timeslots in a TSCH schedule for retransmissions to improve communication reliability is proposed in [27]. This technique also reduces the forwarding delay in a WSN. In the current work we exploit shared retransmission timeslots and dedicate each one to a group of links in the network that have the same destination. This technique reduces the contention on accessing these timeslots by reducing the number of potential transmitters in each of them. Moreover, it enables allocating them in parallel over multiple channels.

III. LLTT Topology Management

In this section we introduce the topology management part of the LLTT technique. Suppose that $V = \{n_i : 1 \leq i \leq N\}$ is the set of N wireless sensor nodes deployed in an in-vehicle convergecast WSN. Sensor nodes are either battery-powered or ambient-powered [14]]. E.g., all parts in the car that do electro-mechanical tasks need the power line anyway. Ambient-powered nodes include the IEEE 802.15.4-enabled control units of the vehicle that have a wired connection (e.g., CAN-bus) with the central unit of the vehicle. LLTT topology management consists of two steps. LLTT first selects a tree-topology structure T based on the number of available nodes in the network (right side of Fig. 2a). This fixed tree structure is designed to provide high parallelism for the TSCH schedule, which leads to short slotframes. Moreover, it reduces the complexity of the TSCH scheduling algorithm, as the topology is predictable. As a second step, LLTT topology management maps each node of V to a vertex in T (right side of Fig. 2a-e). In this process, link quality and the power condition of each node are taken into account. After selecting the best topology



(a) Physical connectivity graph and topology graph

(b) Matching after 3 iterations, all subtree roots are matched first

(c) Matching after 6 iterations, children of node 2 are added

(d) Matching after 8 iterations, children of node 8 are added

(e) Matching after 10 iterations, children of node 9 are added

Fig. 2. An example of matching in a small in-vehicle network with 11 nodes. Matching is done based on the quality of physical links in the network to build the network topology with high communication reliability. For simplicity of the example, all links are assumed to be symmetric.

for the network, the scheduling part of LLTT (presented in the next section) schedules the defined links in the TSCH context.

Two links $n_i \rightarrow n_j$ and $n_m \rightarrow n_k$ are independent iff $\{i, j\} \cap \{m, k\} = \emptyset$. This is because each node can only participate in one transmission or reception at a time. Two independent links can perform communications at the same time on different channels, while two dependent links should be serialized. Assuming a star in-vehicle network, the sink node is one side of all communications and all links will be dependent. Thus, parallel communication is not possible. To be able to maximally take advantage of TSCH parallel communications to reduce latency, the number of independent links in the network topology should be maximized. This requires the convergecast network to follow a multi-hop tree topology, rooted at the sink node. Accordingly, each link in a subtree is independent of all the links in other subtrees of the network (e.g., in Fig. 2e all links towards node 2 are independent of links towards nodes 8 and 9).

While multi-hop communication provides more potential parallelism in TSCH scheduling, it increases the average transmission count in the network. This is because of packet forwarding that leads to higher energy consumption. Moreover, multi-hop communication may cause higher end-to-end packet error rates due to persistent interference. This requires retransmissions at each hop which comes with extra required timeslots per slotframe and thus higher data delivery latency. To lower these costs, LLTT limits the number of hops to two. LLTT constructs a two-level tree topology that each nodes in the first level works as a router to forward the received data from its children to the sink node. Limiting the number of hops to two is in line with the fact that almost all wireless sensor nodes in an in-vehicle network are able to directly reach each other and a star network can satisfy the connection requirements.

Assume that $ST^i \subset V, 1 \leq i \leq N_{ST}$ is the set of $|ST^i|$ sensor nodes of the i^{th} subtree, including the root of that subtree. N_{ST} is the total number of subtrees in the network. All the links in a subtree and the link from that subtree heading towards the sink node are dependent. Thus for ST^i, there are $|ST^i|$ links that should be scheduled in serial timeslots. This is equal to the degree of the root of ST^i. also the N_{ST} links towards the sink node are dependent, which makes it necessary to allocate them into N_{ST} serial timeslots. Considering one slotframe to be allocated for links of each subtree, a higher number of nodes in each subtree leads to a longer slotframe required for it. This increases the time between data generation at end nodes and packet delivery at the sink node. A smaller N_{ST} leads to higher average $|ST^i|$, requiring longer slotframes and vise versa. Thus, for all $1 \leq i \leq N_{ST}$, $|ST^i|$ should be as close as possible to the number of subtrees N_{ST}. This can be reached through a balanced complete k-ary tree. A balanced complete k-ary tree [28] is a rooted tree in which each node has k children, except the last level which can be incomplete as long as the distribution of leaf nodes is balanced.

We aim to find $k = N_{ST}$ based on the number of nodes in the network to build a balanced complete k-ary tree with height two. As a balanced complete k-ary tree is a subgraph of a perfect k-ary tree with $k(k+1)+1$ nodes, the maximum number of nodes in a perfect k-ary tree with height two is $k(k+1)+1$. The lower bound for the number of nodes in a balanced complete k-ary tree is equal to the number of vertices of the perfect (k-1)-ary tree.

$$k \times (k-1) + 1 < N \leq k \times (k+1) + 1 \tag{1}$$

Having N as the input, k can be calculated as

$$k = \lceil \frac{\sqrt{(4N-3)} - 1}{2} \rceil. \tag{2}$$

To assign independent links to parallel slotframes of a TSCH schedule, the number of available channels in the Hopping Sequence List ($|HSL|$) is the upper bound for k. If the number of nodes is higher than what a k-ary tree can support, we build the network topology with $|HSL|$ subtrees and divide the leaf nodes to all the subtrees equally. In the remainder, let T be the tree-topology structure derived in this way for a given problem instance.

ALGORITHM 1: Network subgraph isomorphism

Input:
T: tree-topology structure
LQ: $N \times N$ matrix that contains quality of the links between each pair of nodes in the network
P: vector of length N, contains power condition of each node
Output:
NT: network topology graph

```
M = ∅;
Initialize G from LQ;
FindMatch(G, LQ, T, P, M, n_1);
NT = ToTopology(T, M);
/* Merge the tree-topology structure and the
   assigned node IDs to each vertex of it into a
   topology graph                                  */
```

```
FindMatch(G, LQ, T, M, v)
    if |M| = N then
        return ;
    else
        v := NextVertex(T, v);
        H[] := ∅ ;
        /* The history of nodes that are matched
           to v in this iteration                  */
        foreach u := NextMatch(G, LQ, P, M, H, v) do
            add u to H[];
            if IsFeasible(G, v, u) then
                AddPairedSet(G, M, v, u);
                FindMatch(G, LQ, T, P, M, v);
                RemovePairedSet(v, u, G, M);
            end
        end
    end
end
```

```
NextMatch(G, LQ, P, M, H, v)
    u' := M(parent(v ∈ T));
    /* The matched vertex of G to the parent of v
       in T                                        */
    find u ∈ G such that
        1) (u ∉ M) ∧ (u ∉ H) ;
        /* u is not already matched                */
        2) G(u, u') = available_link;
        /* There is a physical link from u toward
           its potential parent                    */
        3) degree(u) ≥ degree(v);
        /* The candidate u is at least connected
           with the same number of nodes that is
           required for connections of v in T      */
        4) if(u' = sink) then P(u) = 1 is preferred;
        /* Ambient-powered nodes are preferred to
           be used as first level nodes            */
        5) W(LQ(u, u'), degree(u), P(u), ...) has the
        maximum value;
    end
    return u;
end
```

After picking a proper tree-topology structure for the WSN, LLTT maps each node in the WSN to a vertex in that structure. This mapping is based on the physical connectivity graph of the network that captures link qualities. *Sub-graph isomorphism* [8] is used to perform the mapping. We enhanced the VF2 subgraph isomorphism algorithm [29] to reduce the memory usage. Considering the high connectivity between nodes in a dense WSN, this algorithms finds a match in polynomial time.

Algorithm 1 shows the high level functionality of LLTT topology management. Besides topology structure T, this algorithm gets the extracted link quality of each link in the network (LQ) and the power condition of each node (P) as its inputs. Each element of P holds a value between 0 and 1 based on the battery percentage of the node (1 shows an ambient-powered node). The algorithm maintains a connectivity matrix G. Each element of G gets a value of {*no_link*, *available_link*, *used_link*, and *blocked_link*}. It is initially filled with no_link and available_link values by applying a threshold to the LQ values. The algorithm further maintains a vector M of network nodes matched to T. It then starts from the root of T, which is assigned to the sink node of the network (node n_1 in Fig. 2a). Then it goes through each vertex v of T, connected to already matched vertices, and finds a match in G.

Fig. 2 shows some iterations of the matching technique for a simple network with $N = 11$ sensor nodes deployed in a vehicle. The physical connectivity graph (after applying a threshold of 50% on the link qualities) is depicted on the left side of Fig. 2a. For simplicity, the link quality of the two links between each couple of nodes is considered to be the same in this example. This figure also shows the balanced 3-ary tree (using Eqn. 2 and $N = 11$) extracted for the depicted network, rooted at node n_1. The sequence of vertices for matching, which is defined by subroutine `NextVertex()`, is shown in Fig. 2a as red arrows in the tree topology structure. This subroutine selects the subtree roots for matching first and then goes throughout leaf nodes in each subtree. This is because the link between each subtree root and the sink node is also used for forwarding data from leaf nodes towards the sink node and has a high impact on the reliability of the multi-hop network. Using a history vector H, the algorithm keeps track of the candidates that failed to be matched in the current iteration. As shown in Algorithm 1, a set of rules is defined to select a candidate match for a given vertex of $v \in T$. The first three rules guarantee that u is a possible candidate to be matched to v. The fourth rule guarantees the use of ambient-powered nodes in the first level of the topology graph, because they forward data from nodes in the second level and consume more power than the second level nodes. Any IEEE 802.15.4-enabled sensor node with wired connection (e.g., CAN-bus) towards the central unit gets the highest priority to be selected as the first level node. The last rule selects the best possible candidate by assigning a weight to each of them. The weighting function **W** determines the suitability of a node to be matched to a vertex and can be calculated as

$$\begin{cases} (\alpha\, LQ(u,u') + \beta\, degree(u)) \times P(u)^2 & if\ degree(v) > 1 \\ \alpha\, LQ(u,u') / (\beta\, degree(u) \times P(u)^2) & if\ degree(v) = 1 \end{cases} \tag{3}$$

where coefficients α and β are to be determined by the user based on the link quality range and platform requirements. For instance, if the application requires very reliable links, $LQ(u,u')$ should be given a higher weight than the other parameter. If the number of ambient-powered sensors in a network is less than N_{ST}, to satisfy rule 4, some of the first level nodes should be selected from the battery-powered sensors. The function **W** gives a high weight to the nodes with higher power levels to be matched to the first level nodes. By running this algorithm every once in a while (e.g., when the vehicle is going to be turned on), battery-powered sensors that are used as the first level nodes will change if there is another sensor available with higher battery charge. Re-running the algorithm also handles the long-term changes in the link qualities that are due to changes in the positioning of nodes, obstacles in the environment and the transmission power of nodes. Since TSCH is a multichannel protocol, short-term changes in link qualities are handled by TSCH interference mitigation techniques such as ATSCH [30], ETSCH [31], and MABO-TSCH [32].

At each iteration, for the current vertex v and the selected matching candidate u, subroutine `IsFeasible()` checks whether by removing this node from G, and accordingly removing all its incoming links and its parent (if the parent has all its children matched), the graph G still remains connected. Otherwise, u is not an actual match for v in this iteration. If u is qualified to be matched to v, `AddPairedSet()` adds pair (v,u) to M and applies the required changes to G. These changes include: (1) changing the link between u and its matched parent from *available_link* to *used_link*; (2) if v is a leaf in T ($degree(v) = 1$), change all the *available_link* links to and from u in G to *blocked_link*; (3) if the degree of $Parent(v)$ in T is equal to the number of *used_link* links towards $Parent(u)$ in G, change all the *available_link* links to and from $Parent(u)$ in G to *blocked_link*. After these changes, the connectivity graph G only keeps the links that still can be used for matching. Fig. 2b shows the matching process after 3 iterations, when all parent nodes are matched. Fig. 2c-e show the matching process after 6, 8, and 10 iterations where all children of each subtree root are matched. As depicted, after matching children of each subtree root, all the *blocked_link*s are removed from the possible candidate links for scheduling.

The algorithm proceeds to match the remaining vertices of T to the remaining vertices of G by recursively calling `FindMatch()`. By returning from each iteration of `FindMatch()` due to a matching failure, all the applied changes to the graph G in that iteration are restored by calling `RemovePairedSet()`. The algorithm finishes when a complete match ($|M| = |T|$) is found in the N^{th} recursion. Because this algorithm uses graph G (with four states for each link) to keep track of changes in each iteration, it requires a low amount of memory to execute. The match of T in G is used as the network graph *NT*, as input of the low-latency scheduling part of LLTT.

IV. LLTT Scheduling

The output of the topology management part of LLTT is a two-hop tree network. In this section, we present the low-latency timeslot allocation for such a tree topology in detail. Algorithm 2 shows the process of low-latency scheduling. This algorithm allocates all the links in each subtree and the link from that subtree to the sink node to one slotframe with a unique channel offset for all timeslots. This is because the links in each subtree are dependent, while they are independent of the links in other subtrees. The links from subtree roots

ALGORITHM 2: Low-latency schedule generator

Input:
L_{SF}: slotframe size given by (4)
NT: network topology graph
N_{ReTx}: number of required retransmissions for each subtree
Output:
$Schedule[slotframe][timeslot]$: TSCH scheduling matrix

```
/* Allocate shared timeslots towards the sink
   node at the end of first slotframe          */
for i ← (L_SF − N_ReTx + 1) to L_SF do
    Schedule[1][i] := [SH, sink_node];
end
for sf ← 1 to N_ST do
    /* The timeslot for the link from the current
       subtree root to the sink node            */
    slot := L_SF − N_ReTx − sf + 1;
    /* Allocate a dedicated timeslot from the
       subtree root towards the sink node       */
    Schedule[sf ][slot ] := [root_of(ST^sf), sink_node];
    /* Allocate shared timeslots towards the
       subtree root right before the current
       timeslot                                 */
    for i ← 1 to N_ReTx do
        slot := slot − 1 ;
        if slot = 0 then
            slot := L_SF − N_ReTx ;
        end
        Schedule[sf ][slot] := [SH, root_of(ST^sf)];
    end
    /* Allocate dedicated timeslots for the
       subtree right before the shared timeslots
       */
    foreach n_i ∈ ST^sf do
        slot := slot − 1 ;
        if slot = 0 then
            slot := L_SF − N_ReTx;
        end
        Schedule[sf ][slot] := [n_i, root_of(ST^sf)];
    end
end
```

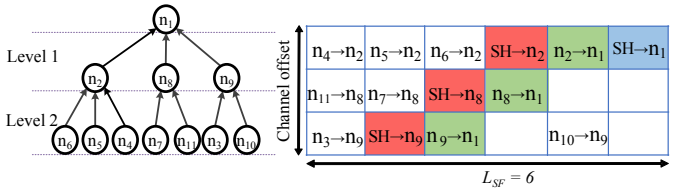


Fig. 3. The TSCH schedule of an example network that is generated by Algorithm 2 (SH indicates shared slots).

towards the sink node, are also dependent and should not be parallelized. Furthermore, a link from a subtree root towards the sink node is dependent on the links in that subtree. Accordingly, multiple parallel slotframes can be used for links of different subtrees.

The required L_{SF} is given to the algorithm as an input. It can be calculated as:

$$L_{SF} = Max[degree(v)|v \in NT] + (2 \times N_{ReTx}) \qquad (4)$$

where N_{ReTx} is the number of timeslots required for grouped retransmission in each subtree. It may be defined by the user or be extracted based on the communication statistics of the network. Fig. 3 shows the schedule that is generated by this algorithm for an example network and $N_{ReTx} = 1$.

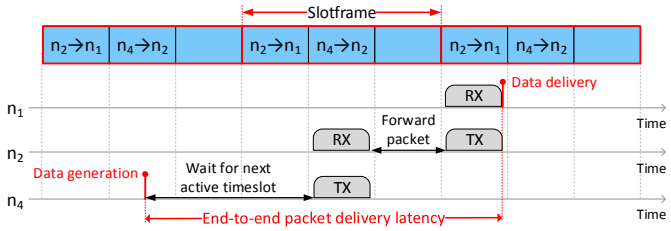


Fig. 4. TSCH timeline of one packet transmission in a two-hop route from n_4 to n_1.

We define a *grouped retransmission timeslot* as a shared TSCH timeslot that is dedicated only for packet retransmissions of all links with the same destination. Assigning grouped retransmission timeslots in LLTT is based on the subtrees and is done by assigning a number of shared timeslots for possible retransmissions of all links in a subtree. Each grouped retransmission timeslot can only be used to transmit packets that are not acknowledged in their dedicated timeslots. The scheduling algorithm starts with allocating the last timeslots of the first slotframe for the grouped retransmission timeslots towards the sink node (the blue timeslot in Fig. 3). Because all subtree roots are possible transmitters in these timeslots, these timeslots cannot be used in other slotframes. Then the algorithm creates a slotframe for each subtree, using the already started slotframe for the first subtree. For each subtree, it allocates a timeslot for the link from that subtree root towards the sink node, together with the links in that subtree. Because these links are all dependent they are allocated to the same slotframe. Because of the dependency between the links in the first level of the network, our algorithm uses different timeslots for the links from each subtree root towards the sink node (green timeslots in Fig. 3).

Considering the example network in Fig. 3, there is a two-hop path from n_4 to n_1 (sink node). This path requires two sequential TSCH timeslots to establish the end-to-end connection. As shown in the timeline diagram of Fig. 4, the end-to-end latency is equal to sum of the buffering time at the source node (n_4) and the intermediate node (n_2). The data buffering time at n_4 may vary from 0 to L_{SF}, while The buffering time at n_2 depends on the placement of the allocated timeslots for the link in the second level and the following link in the first level within the TSCH slotframes. To reduce the buffering time at the intermediate nodes (subtree roots), the timeslots allocated for the links in a subtree should be before the timeslot allocated for the link from the root of that subtree towards the sink node.

To reduce the latency also for the failed transmissions during dedicated timeslots, grouped retransmission timeslots for the links in each subtree (red timeslots in Fig. 3) are allocated between the dedicated timeslots for the second and first levels. Then the algorithm continues with allocating dedicated timeslots for all the links in that subtree, before the shared timeslots. Since this scheduling algorithm is based on a predefined topology structure, it has a low complexity compared to other central scheduling algorithms for the TSCH networks.

The LLTT schedule provides guaranteed access to the medium for all links, targeting reliable and low-latency com-

munications. This is done through parallel TSCH communications and a reduced slotframe length that increase the frequency of allocated timeslots to each link, supporting high data rates. A node in the second level of the network that has a lower data generation rate than the allocated bandwidth, can wake up and transmit in its allocated timeslots only if there is a packet ready for transmission in its MAC buffer. Thus, there is no power overhead for these nodes and low latency is still ensured. However, not all in-vehicle sensor nodes require low latency or high data rate communications. For those sensor nodes, a super-schedule can be used with a size of multiple LLTT slotframe sizes. The multiplication factor can be selected based on the data generation rate of the sensor node and the maximum allowed latency. This technique minimizes power consumption of the second level nodes and reduces the power consumption of the forwarding nodes at the first level, as they only listen to the allocated timeslot every multiple slotframes.

V. Data Aggregation and Latency Analysis

LLTT uses topology management and scheduling to provide a collision-free and low-latency TSCH schedule. Disabling retransmissions, the worst-case packet forwarding delay in an intermediate node can be expressed as the maximum time difference between the links in the second level of the two-hop tree and the following links in the first level (the time between Rx and Tx at n_2 in the timeline diagram of Fig. 4). This delay, in terms of timeslots, is equal to L_{SF}. Since the timeslot duration is a constant in our work, we consider latency in terms of timeslots and do not convert to time.

As sensory data is often very short, aggregation can be used at the intermediate nodes in the multi-hop paths to efficiently use the available bandwidth. When aggregation is used, all the data packets received from the second level nodes are aggregated with the subtree root's own data to be sent to the sink node. Aggregation reduces the number of timeslots in each slotframe needed by the nodes for forwarding data items received from others. This leads to more efficient use of available bandwidth of each link. Furthermore, it reduces the average communication latency due to reducing the size of the slotframe. We use a periodic aggregation at the network layer in periods of one slotframe. Shorter periods may cause overloading the MAC which increases the latency due to packet buffering after aggregation. On the other hand, longer periods lead to higher latencies due to longer buffering time for packets before aggregation. The aggregation ideally should be done exactly before the allocated timeslot to the link from the subtree root towards the sink node, in order to lower the forwarding latency at the subtree roots. This requires a notion of TSCH MAC timing at upper layers (i.e., network layer), which is not practical. This periodic aggregation may lead to a delay of maximum L_{SF}, when a data packet is received exactly after the aggregation at the network layer. In this case, the data should wait for one slotframe to be aggregated with the next packet. Thus, in total the maximum packet forwarding latency in an intermediate node is $2 \times L_{SF}$.

Considering the data buffering time at the source node (see Section IV) and forwarding delay at intermediate nodes, the maximum packet delivery latency is $3 \times L_{SF}$. Using (2) and (4), the Worst Case Packet Delivery Latency (WCPDL) (when retransmissions are disabled) is:

$$\begin{aligned} \text{WCPDL}_{No\_ReTx} &= 3 \times L_{SF} \\ &= 3 \times Max[degree(n)|n \in NT] \\ &\leq 3 \times (\lceil \frac{\sqrt{(4N-3)}-1}{2} \rceil + 1). \end{aligned} \tag{5}$$

Using retransmission timeslots to increase the reliability of the network leads to higher data delivery latency in a network. This higher delay has two reasons: 1) longer slotframes due to retransmission timeslots and, 2) the timeslot gap between the dedicated and retransmission timeslots for the links in the first level of the network. The retransmission timeslots in the second level are allocated in such a way that they cause an extra latency of only one timeslot, due to the longer slotframes. This delay is therefore covered in the first point.

The dedicated timeslot for the link from the root of each subtree towards the sink node is the last point of data delivery when there is no retransmission. This is while with retransmission, the last chance for data delivery at the sink node is the allocated retransmission timeslot. This costs a latency equal to the gap between the dedicated and retransmission timeslots for the links towards the sink node which is at most equal to $L_{SF} - 1$. Accordingly, the WCPDL when the maximum number of retransmissions is one can be extracted by (6).

$$\text{WCPDL}_{ReTx} = 4 \times L_{SF} - 1 \tag{6}$$

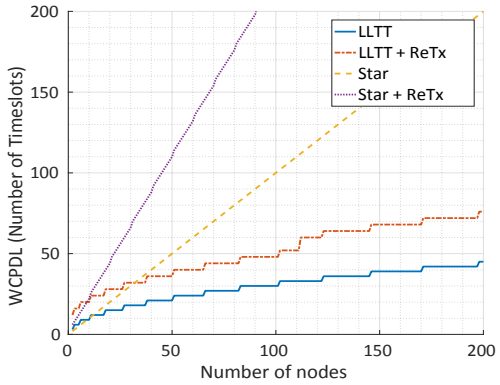


Fig. 5. Analytical worst case packet delivery latency vs the number of nodes in the network for TSCH schedules of star and LLTT topologies.

Fig. 5 shows the analytical WCPDL for star and LLTT networks for different numbers of nodes. As shown, WCPDL in a star TSCH network increases linearly with the number of nodes in the network, because all the links in the network are dependent. Using a two-hop network in LLTT, the WCPDL is proportional to the square root of the number of nodes in the network. This plot also shows the WCPDL when retransmissions are enabled, and 10% of the slotframe bandwidth is allocated for a maximum retransmission count of one. For a star network, due to the gap between a dedicated and shared timeslot for transmissions of a link (which is at maximum equal to the $L_{SF} - 1$), retransmissions lead to a double latency compared to star with no retransmissions. LLTT with retransmissions can still provide a better latency

Fig. 6. Experimental testbed with 30 JN5168 [33] wireless sensor nodes distributed over a car.

bound compared to star with no retransmission when the number of nodes in the network exceeds 32.

VI. Experimental Setups

We set up an experimental testbed of 31 NXP JN5168 dongles [33] in a car for evaluation (Fig. 6). These dongles include a wireless microcontroller which integrates a 32-bit RISC processor and a 2.4GHz IEEE 802.15.4 compliant transceiver. The transmission power of nodes is set to 0 dBm. Each dongle runs the Contiki OS with TSCH as the MAC protocol. The topology management and scheduling parts of LLTT are done centrally using Matlab on a host computer that is connected to the sink node (n_1) via a UART interface. The TSCH schedule is distributed by the sink node through an extra timeslot at the beginning of the first slotframe that is allocated for downstream data. The sink node initiates the network as the PAN coordinator and all nodes periodically generate data packets to be delivered to the sink.

To provide the LQ matrix for LLTT, initially, the network goes through an identification phase. In this phase, each node gets a dedicated timeslot and periodically broadcasts random packets to all other nodes. Accordingly, each node can extract quality of all its possible incoming links. We use Packet Reception Ratio (PRR) as the quality metric (LQI of incoming packets can be used as well). After a predefined number of transmissions by each node, the sink node collects the results from all nodes and reports them to the host computer. Because this step is done once and only at the initialization of the network, the overhead is negligible.

For the used network with 30 sensor nodes and one sink node, the network topology is a complete 5-ary tree ($N_{ST} = 5$). We consider one third of the nodes as ambient-powered sensors that are placed in different parts of the car. The rest are considered as battery-powered. The schedule generated by LLTT is broadcast to all nodes by the sink node. Then the main phase of the experiment starts, in which each node generates and sends periodic data packets.

As LLTT is a cross-layer design performing topology management as well as scheduling, we use some combinations of known networking and scheduling techniques for our performance comparison. We use a pure star network with a schedule that dedicates one unique timeslot to each node. In addition, we built two setups with combinations of RPL [34] and two TSCH scheduling mechanisms, namely the minimal schedule [9] and Orchestra [3]. For the TSCH minimal schedule, we use $L_{SF} = 1$ timeslot (slotted ALOHA), which leads to 100% duty cycle for all nodes. For Orchestra, we used the sender-based schedule, which was reported to have the best performance for convergecast [3], with $L_{SF} = 31$ and Enhanced Beacon (EB) slotframe size 53.

TABLE I
Experimental setups and their slotframe length

Network	L_{SF}	Comment
LLTT	7	Schedule size of 6, using (4) + One timeslot for EB advertisement
LLTT+ReTx	7+2	2 timeslots for retransmissions
Star	30+1	One dedicated timeslot for each node + One timeslot for EB advertisement
Star+ReTx	31+6	6 timeslots for retransmissions
Minimal	1	All timeslots active for Tx/Rx
Orchestra	31	sender-based slotframe

We perform experiments with retransmissions enabled and disabled for all the setups. We consider 20% retransmission bandwidth for each link, and accordingly decide on the number of shared retransmission timeslots for LLTT. For the minimal schedule, we conduct multiple experiments with different maximum number of link layer retransmissions from 1 to 3, and report the best results in terms of latency and reliability. To reduce contention on shared timeslots in the star network, we assign each shared timeslot to a group of 5 nodes, based on the retransmission bandwidth. This increases the size of the star slotframe by 6 when the network consists of 30 sensor nodes and one sink node. As Orchestra uses a hash of the nodes' MAC address to assign timeslots to nodes, there is a chance of allocating the same timeslot for multiple links. Because all nodes are in the range of each other and have a high data generation rate, this leads to interfered communications within the network and thus packet failures. The experimental results showed that only few nodes were able to deliver their data to the sink node. Thus, the results for RPL in combination with Orchestra were ultimately not considered in the performance analysis discussed in the next section. Orchestra is better suited for larger and lower density networks than the ones considered in this work. The network setups and their slotframe sizes are summarized in Table I.

VII. Performance Analysis

We investigate two metrics to evaluate the performance of LLTT. The end-to-end Data Delivery Ratio (DDR) is the ratio between the received data samples at the sink node and the total number of data samples generated by the source node. The end-to-end latency is the time between data sample generation by the source node and its reception at the sink node. In the following, first we analyze the data delivery performance of LLTT, when different latency bounds are considered for validity of data by the application. Moreover, to evaluate the effect of shared retransmission timeslots, we performed a set of experiments in presence of controlled interference. Second, we investigate the performance of LLTT for different data generation rates.

A. Data Delivery vs Latency

Fig. 7 shows the average DDR for different protocols and latency bounds, when the network is in an interference-free

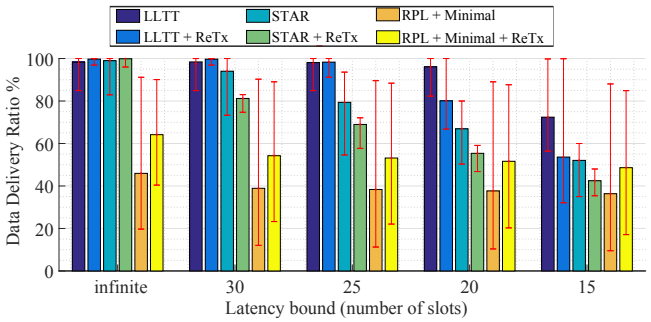


Fig. 7. Average DDR for different latency bounds in an interference-free environment. Red error bars show the distribution of DDR for all nodes in the network.

environment. A homogeneous data generation rate of 2Hz is used for all the nodes. Red error bars show the worst and best DDR for all 30 non-sink nodes in the network. When the applied latency bound is infinite so latency has no effect on data validity, both LLTT and star networks have a DDR higher than 98%. By using the retransmission timeslots, DDR is increased to 99.7% for both, while in the best case, RPL+Minimal has an average DDR of 64% with maximum retransmissions of one. For higher retransmissions, RPL+Minimal faces high traffic load of retransmitted packets, leading to worse DDR and latency.

We use the results of the unbounded latency experiment and apply four latency bounds of 30, 25, 20, and 15 timeslots duration to it, i.e., 300, 250, 200, and 150 ms for a TSCH timeslot duration of 10 ms. Each bar in Fig. 7 shows the percentage of valid data that is delivered to the sink node within the specified latency bound. As shown, by reducing the latency bound, there are less valid data packets delivered, especially for the star network. This is mostly because data has to wait in the MAC buffer of the source node for the next active timeslot to be transferred. This waiting time varies from 0 to L_{SF}. Because the star network has the longest slotframe, it causes higher average latency compared to others and thus it is affected more by the latency bounds. This effect is more visible when retransmission slots are used, caused by the longer slotframes. LLTT reduces the buffering time at the source nodes by using very short slotframes. Thus, even for a latency bound of 25 timeslots, both versions of LLTT still deliver the same percentages of data packets as in the unbounded scenario.

Fig. 7 shows that DDR of both versions of RPL+Minimal is almost the same for all latency bounds from 30 down to 15 timeslots. This is because all the timeslots are active for transmissions of all nodes and a generated packet at application layer can be transmitted at the same slot in MAC layer. The DDR drop observed when going from unbounded to the 30 slot latency bound is due to the use of the back-off mechanism after transmission failures, which cause longer latencies that improve DDR in the unbounded scenario but invalidate data under latency bounds.

Fig. 8 shows the latency distribution of each node for LLTT and star networks. The results confirm the analytical WCPDL (Section V), considering one extra timeslot at each slotframe for network advertisement. This means that for a

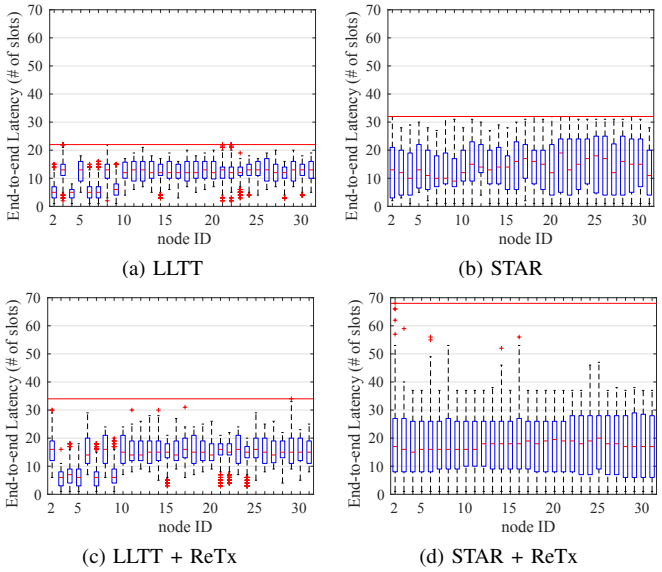


Fig. 8. End-to-end latency distribution of each node's communication towards the sink node. X-axes show the node number and the red lines show the maximum observed packet delivery latency for all nodes.

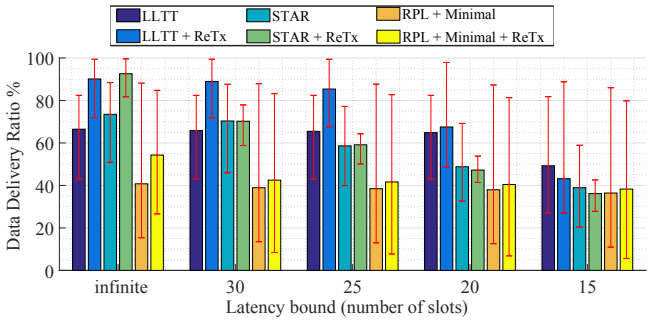


Fig. 9. Average DDR for different latency bounds, in presence of controlled interference on 4 out of 16 channels.

higher number of nodes than what is used in these experiments, LLTT even performs better than the star network. Another observation of Fig. 8 is that five nodes in both LLTT experiments have lower latencies than other nodes. These are the subtree roots which have one-hop communications towards the sink node, while other nodes have two-hop communications. Accordingly, the in-vehicle network designer may decide to select the nodes with more stringent latency requirements as the subtree roots.

Retransmission timeslots are effective when physical layer links are not 100% reliable, otherwise they only lead to longer slotframes and higher latencies (Fig. 8c and 8d). To evaluate the effect of the shared retransmission timeslots on communication reliability and latency, we conduct a set of experiments in presence of wireless interference. We used four interference generators to block 4 out of 16 frequency channels used for TSCH channel hopping by continuously generating dummy packets. Fig. 9 shows the DDR results of these experiments for different latency bounds. Since 75% of the channels are not interfered, 75% of the links are expected to have successful communications. This is visible for the star experiment (with infinite latency bound), since all the communications are single hop. Due to multi-hop communications in

LLTT and RPL+Minimal, the average DDR is lower. The high variation in DDR results of each experiment (red error bars) is because of difference in reliability level of one-hop and two-hop communications.

For all the experiments with retransmissions, we set the maximum retransmission count to one. As observed in Fig. 9, grouped retransmission timeslots improve the average DDR about 24% and 20% for LLTT and star network, respectively. For RPL+Minimal, this improvement is less than 15%. This is because retransmissions impose more contention in the Minimal schedule.

A star network with retransmissions experiences a high drop in DDR when latency bounds apply. This implies higher average latency. This happens because failed packets should wait for shared retransmission slots. Longer slotframes due to addition of retransmission slots is another reason. Retransmission timeslots in LLTT are allocated in such a way that they impose the lowest possible latency increase for single and multi-hop communications. For RPL+Minimal, the latency overhead of retransmissions is much lower, because all timeslots for transmissions and retransmission can take place after one back-off period. For very low latency bounds, the performance of the LLTT and star networks, which use dedicated timeslots, gets closer to the performance of RPL+Minimal with shared timeslots.

B. Effect of Data Generation Rate

To evaluate the performance of LLTT under different data generation rates, we picked five data generation rates (1, 2, 3, 5, and 10 Hz), and performed experiments for each network type with each data rate. No latency bound is considered for data validity. We performed these experiments in presence of interference generators. Since the slotframe size of both versions of LLTT is less than 10 timeslots (for 30 non-sink nodes) and aggregation is used by subtree roots, it can handle a data generation rate of 10 Hz (one packet every 10 timeslots). On the other hand, star uses a slotframe size of 31 timeslots and if more than one packet is injected to the TSCH MAC every 31 timeslots, latency increases dramatically. In order to prevent this, we use periodic local data aggregation in star networks with a period of one slotframe. Higher packet generation rates for RPL+Minimal lead to higher contention rates and lower performance. This decreases DDR when retransmissions are disabled, and increases latency when retransmissions are enabled. We use the periodic aggregation technique for RPL+Minimal with a period of 3Hz. This reduced contention helps more packets to be delivered per time unit, but with a higher average latency.

Fig. 10 shows the average achieved DDR and latency in each network under different data rates. This figure reveals that both versions of LLTT have almost the same average end-to-end latency for different data rates. For RPL+Minimal, the achieved latency increases when increasing the data generation rate and, at the same time, average DDR goes down. This is due to higher contention which increases failure rate as well as the waiting time for packet (re)transmissions. While aggregation only affects the packet size and has no effect on

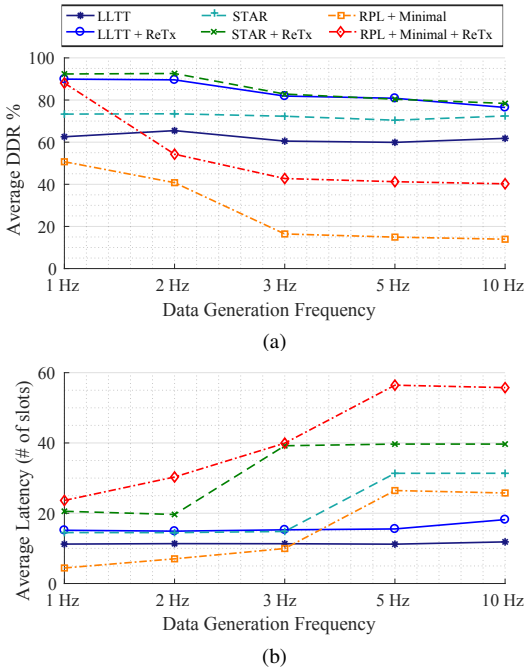


Fig. 10. a) average DDR and b) average data delivery latency for different networks, for different data generation rates.

the contention rate, for data rates higher than 3Hz, the achieved DDR is almost the same for both RPL+Minimal setups (with and without retransmission). In contrast, periodic aggregation increases the average latency with about half the aggregation period, i.e., 16 timeslots, for data rates higher than 3Hz. The same happens for the star setup when the data generation period is lower than L_{SF}. For star without retransmission timeslots, this happens for data rates higher than 3Hz with a latency increase of around 16 timeslots ($L_{SF} = 31$). This is while for star with retransmission timeslots, this effect is observed for data rates higher than 2Hz with about 19 timeslots latency increase ($L_{SF} = 37$).

As all the communications of LLTT and star use dedicated links, they show almost constant DDR for all data rates. When shared retransmission slots are exploited, lower data rates lead to higher DDR. This is because of the lower contention on shared retransmission timeslots that increases the chance of packet delivery for failed transmissions on dedicated timeslots. For low data generation rates, LLTT with retransmission timeslots provides similar DDR to the star and RPL+Minimal networks with retransmission, while it has over 25% lower average latency. Furthermore, for high data generation rates, LLTT outperforms star with almost 60% and RPL+Minimal with 70% lower average latency. It shows the ability of LLTT to provide reliable and low latency data communications for a dense network with high communication loads.

VIII. Conclusions

This paper proposes LLTT, a cross-layer design for reliable and low-latency data convergecast in dense TSCH-based industrial wireless sensor networks. Based on the number of nodes in the network, LLTT picks a balanced two-level complete k-ary tree as the topology structure of the network. This tree-topology structure provides a high number of independent links, which paves the way for a high degree of

parallelism in TSCH schedules. Employing an optimized graph isomorphism algorithm, LLTT extracts a proper match for the selected tree structure in the connectivity graph of the network. The extracted network topology is used by a light-weight TSCH schedule generator to find a highly parallelized TSCH schedule. This schedule generator allocates grouped shared retransmission timeslots among dedicated timeslots to improve communication reliability. Periodic data aggregation is used to improve bandwidth utilization.

Experimental results in an in-vehicle testbed with 31 sensors nodes show that LLTT is able to provide low-latency data delivery. Experiments show that for high data generation rates, the periodic aggregation technique of LLTT keeps the average communication latency low. A star network and a network with RPL routing and minimal schedule face latency increase under high data generation rates.

It is an interesting future direction of research to monitor link-qualities at run-time and update the used links in the network, based on the long-term changes in the link qualities and/or application requirements.

REFERENCES

[1] "IEEE Standard for Low-Rate Wireless Networks," *IEEE Std 802.15.4-2015 (Revision of IEEE Std 802.15.4-2011)*, pp. 1–709, April 2016.

[2] M. R. Palattella, T. Watteyne, Q. Wang, K. Muraoka, N. Accettura, D. Dujovne, L. A. Grieco, and T. Engel, "On-the-Fly Bandwidth Reservation for 6TiSCH Wireless Industrial Networks," *IEEE Sensors Journal*, vol. 16, no. 2, pp. 550–560, Jan 2016.

[3] S. Duquennoy, B. Al Nahas, O. Landsiedel, and T. Watteyne, "Orchestra: Robust Mesh Networks Through Autonomously Scheduled TSCH," in *Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems*, ser. SenSys '15. New York, NY, USA: ACM, 2015, pp. 337–350. [Online]. Available: http://doi.acm.org/10.1145/2809695.2809714

[4] M. R. Palattella, N. Accettura, M. Dohler, L. A. Grieco, and G. Boggia, "Traffic Aware Scheduling Algorithm for reliable low-power multi-hop IEEE 802.15. 4e networks," in *Personal Indoor and Mobile Radio Communications (PIMRC), 2012 IEEE 23rd International Symposium on*. IEEE, 2012, pp. 327–332.

[5] Y. Jin, P. Kulkarni, J. Wilcox, and M. Sooriyabandara, "A centralized scheduling algorithm for IEEE 802.15.4e TSCH based industrial low power wireless networks," in *2016 IEEE Wireless Communications and Networking Conference*, April 2016, pp. 1–6.

[6] N. Accettura, M. R. Palattella, G. Boggia, L. A. Grieco, and M. Dohler, "Decentralized traffic aware scheduling for multi-hop low power lossy networks in the internet of things," in *World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2013 IEEE 14th International Symposium and Workshops on a*. IEEE, 2013, pp. 1–6.

[7] M. Bagaa, M. Younis, D. Djenouri, A. Derhab, and N. Badache, "Distributed Low-Latency Data Aggregation Scheduling in Wireless Sensor Networks," *ACM Transaction on Sensor Networks.*, vol. 11, no. 3, pp. 49:1–49:36, Apr. 2015. [Online]. Available: http://doi.acm.org/10.1145/2744198

[8] D. Eppstein, "Subgraph isomorphism in planar graphs and related problems," in *SODA*, vol. 95, 1995, pp. 632–640.

[9] X. Vilajosana and K. Pister, "Minimal 6TiSCH Configuration draft-ietf-6tisch-minimal-06 (work in progress)," IETF, Internet Draft, Jan. 2015.[Online]. Available: http://tools.ietf.org/html/draft-ietf-6tisch-minimal-06, Tech. Rep., 2015.

[10] M. A. Rahman, "Design of wireless sensor network for intra-vehicular communications," in *Wired/Wireless Internet Communications*, A. Mellouk, S. Fowler, S. Hoceini, and B. Daachi, Eds. Cham: Springer International Publishing, 2014, pp. 29–40.

[11] M. Hashemi, W. Si, M. Laifenfeld, D. Starobinski, and A. Trachtenberg, "Intra-car multihop wireless sensor networking: a case study," vol. 52, no. 12, December 2014, pp. 183–191.

[12] S. Reis, D. Pesch, B.-L. Wenning, and M. Kuhn, "Intra-vehicle wireless sensor network communication quality assessment via packet delivery ratio measurements," in *Mobile Networks and Management*, R. Agüero, Y. Zaki, B.-L. Wenning, A. Förster, and A. Timm-Giel, Eds. Cham: Springer International Publishing, 2017, pp. 88–101.

[13] D. Parthasarathy, R. Whiton, J. Hagerskans, and T. Gustafsson, "An in-vehicle wireless sensor network for heavy vehicles," in *2016 IEEE 21st International Conference on Emerging Technologies and Factory Automation (ETFA)*, Sept 2016, pp. 1–8.

[14] D. Parthasarathy, E. Wermstrm, and R. Whiton, "Automotive architecture and integration requirements," European project DEWI: Dependable Embedded Wireless Infrastructure, Tech. Rep. D308.001, September 2014. [Online]. Available: http://www.dewiproject.eu/media/deliverables/?wpdmc=deliverables

[15] S. Gandham, M. Dawande, and R. Prakash, "Link scheduling in wireless sensor networks: distributed edge-coloring revisited," *Journal of Parallel and Distributed Computing*, vol. 68, no. 8, pp. 1122–1134, 2008.

[16] H. Zhang, P. Soldati, and M. Johansson, "Performance Bounds and Latency-Optimal Scheduling for Convergecast in WirelessHART Networks," *IEEE Transactions on Wireless Communications*, vol. 12, no. 6, pp. 2688–2696, June 2013.

[17] R. Soua, P. Minet, and E. Livolant, "Wave: a distributed scheduling algorithm for convergecast in IEEE 802.15. 4e TSCH networks," *Transactions on Emerging Telecommunications Technologies*, vol. 27, no. 4, pp. 557–575, 2016.

[18] G. Gaillard, D. Barthel, F. Theoleyre, and F. Valois, "High-reliability scheduling in deterministic wireless multi-hop networks," in *2016 IEEE 27th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*, Sept 2016, pp. 1–6.

[19] C. Wang, Z. Zhao, L. Zhu, and H. Yao, "An energy efficient routing protocol for in-vehicle wireless sensor networks," in *Data Science*, B. Zou, Q. Han, G. Sun, W. Jing, X. Peng, and Z. Lu, Eds. Singapore: Springer Singapore, 2017, pp. 161–170.

[20] T.-Y. Huang, C.-J. Chang, C. W. Lin, S. Roy, and T.-Y. Ho, "Intra-vehicle network routing algorithm for wiring weight and wireless transmit power minimization," in *The 20th Asia and South Pacific Design Automation Conference*, Jan 2015, pp. 273–278.

[21] S. Din, A. Paul, A. Ahmad, and J. H. Kim, "Energy efficient topology management scheme based on clustering technique for software defined wireless sensor network," *Peer-to-Peer Networking and Applications*, Oct 2017. [Online]. Available: https://doi.org/10.1007/s12083-017-0607-z

[22] J. Luo, J. Hu, D. Wu, and R. Li, "Opportunistic routing algorithm for relay node selection in wireless sensor networks," *IEEE Transactions on Industrial Informatics*, vol. 11, no. 1, pp. 112–121, Feb 2015.

[23] L. LoBello and E. Toscano, "An adaptive approach to topology management in large and dense real-time wireless sensor networks," *IEEE Transactions on Industrial Informatics*, vol. 5, no. 3, pp. 314–324, Aug 2009.

[24] M. Younis, I. F. Senturk, K. Akkaya, S. Lee, and F. Senel, "Topology management techniques for tolerating node failures in wireless sensor networks: A survey," *Computer Networks*, vol. 58, pp. 254 – 283, 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1389128613002879

[25] E. Fasolo, M. Rossi, J. Widmer, and M. Zorzi, "In-network aggregation techniques for wireless sensor networks: a survey," *IEEE Wireless Communications*, vol. 14, no. 2, pp. 70–87, April 2007.

[26] R. Velmani and B. Kaarthick, "An efficient cluster-tree based data collection scheme for large mobile wireless sensor networks," *IEEE Sensors Journal*, vol. 15, no. 4, pp. 2377–2390, April 2015.

[27] M. Hashimoto, N. Wakamiya, M. Murata, Y. Kawamoto, and K. Fukui, "End-to-end reliability- and delay-aware scheduling with slot sharing for wireless sensor networks," in *8th International Conference on Communication Systems and Networks (COMSNETS)*, 2016, pp. 1–8.

[28] J. A. Storer, *An introduction to data structures and algorithms*. Springer Science & Business Media, 2012.

[29] L. P. Cordella, P. Foggia, C. Sansone, and M. Vento, "A (sub)graph isomorphism algorithm for matching large graphs," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 10, pp. 1367–1372, Oct 2004.

[30] P. Du and G. Roussos, "Adaptive time slotted channel hopping for wireless sensor networks," in *Computer Science and Electronic Engineering Conference (CEEC), 2012 4th*, Sept 2012, pp. 29–34.

[31] R. Tavakoli, M. Nabi, T. Basten, and K. Goossens, "Dependable Interference-Aware Time-Slotted Channel Hopping for Wireless Sensor Networks," *ACM Trans. Sen. Netw.*, vol. 14, no. 1, pp. 3:1–3:35, Jan. 2018. [Online]. Available: http://doi.acm.org/10.1145/3158231

[32] P. H. Gomes, T. Watteyne, and B. Krishnamachari, "Mabo-tsch: Multi-hop and blacklist-based optimized time synchronized channel hopping," *Trans. Emerg. Telecommun. Technol.*, pp. e3223–n/a, 2017.
[33] NXP Semiconductors, "JN516x IEEE802.15.4 Wireless Microcontroller," accessed: Jan. 2018. [Online]. Available: http://www.nxp.com/docs/en/data-sheet/JN516X.pdf
[34] T. Winter and RPL Author Team, "RPL: IPv6 routing protocol for low-power and lossy networks," 2012.

Rasool Tavakoli received the B.S. and M.Sc. degrees in computer engineering from the Isfahan University of Technology, Isfahan, Iran in 2010 and 2013, respectively. Since 2014, he is a PhD candidate in the Electronic Systems group at the Department of Electrical Engineering of Eindhoven University of Technology, the Netherlands. His research interests include dependable and low-power wireless sensor networks, real-time networks, automotive networks, optimization of embedded sensors, and vehicular ad-hoc networks.

Majid Nabi (M08) received the M.Sc. degrees in computer engineering from Tehran University, and the Ph.D. degree in electrical and computer engineering from Eindhoven University of Technology (TU/e), Eindhoven, the Netherlands. He is currently an assistant professor with the Department of Electrical Engineering at TU/e, and Isfahan University of Technology. His research interests include efficient and reliable networked embedded systems, wireless sensor networks, and internet-of-things.

Twan Basten (M98-SM06) received the M.Sc. and Ph.D. degrees in computing science from Eindhoven University of Technology (TU/e), Eindhoven, the Netherlands. He is currently a Professor with the Department of Electrical Engineering, TU/e, where he chairs the Electronic Systems group. He is also a Senior Research Fellow with ESI, TNO, Eindhoven. His current research interests include the design of embedded and cyber-physical systems, dependable computing, and computational models.

Kees Goossens has a PhD from the University of Edinburgh in 1993 on hardware verification using embeddings of formal semantics of hardware description languages in proof systems. He worked for Philips/NXP from 1995 to 2010 on real-time networks on chip for consumer electronics. He was part-time full professor at Delft university from 2007 to 2010, and has been full professor at the Eindhoven University of Technology, researching composable, predictable, low-power embedded systems, supporting multiple models of computation since then. He published 4 books, 175+ papers, and 24 patents.