Improving network formation in 6TiSCH networks

Carlo Vallati, Simone Brienza, Giuseppe Anastasi, Sajal K. Das

Abstract—The industrial Internet of Things (IIoT) is expected to revolutionize the current industry. The capillary introduction of sensors and actuators for real-time monitoring and remote control and their seamless integration into existing information systems will represent a technological breakthrough that will help reshaping the industrial processes. To this aim, the definition of wireless communication standards will play a crucial role in reducing deployment costs and minimizing the time for installation. In this context, the new IPv6 over the TSCH mode of IEEE802.15.4e communication stack, 6TiSCH, represents the current leading standardization effort that aims at achieving both reliable and timed wireless communication and integration within IPv6 communication networks for industrial systems. In this paper, the network formation dynamics of 6TiSCH networks are assessed, considering the current guidelines for the so-called minimal configuration, a static initial configuration pre-configured to guarantee control communication during network bootstrap. It is shown that the minimal configuration might lead to long network formation and suboptimal performance of the routing algorithm which may result into a disconnected network. In order to overcome this issue, a dynamic resource management algorithm to be executed during network bootstrap is proposed. Simulation and experimental results show that the proposed solution allows to minimize the network formation time and also helps in optimizing routing operations leading to the discovery of better routes.

Index Terms—6TiSCH, IEEE 802.15.4, RPL, Network Formation

1 Introduction

The Industrial Internet of Things (IIoT) is expected to trigger the next industrial revolution. Although sensors and actuators are already deployed on industrial deployments for telemetry and remote control, their integration into a single horizontal information system is desired to enable a novel set of services and applications so as to exploit the convergence of data collected from heterogeneous domains and the possibility to interact with devices from different industrial processes [1]. In this context, wireless communication technologies represent a crucial enabler to foster capillary diffusion of sensors and actuators. Although popular in a wide range of contexts, the adoption of wireless communications has been always refrained in industrial deployments in favor of wired communication technologies that could guarantee reliable and timed data delivery. Recently, the dramatic growth of connected devices that is expected for industrial environments is pushing towards the adoption of wireless solutions also for industrial deployments to minimize costs and reduce time to deployment. However, the most popular standard for sensors and actuators networks, the IEEE 802.15.4 standard, is unfit to offer the Quality of Service (QoS) that is required for industrial applications since it does not include support for reliable and timed delivery [2].

In order to overcome such limitation, recently the IEEE 802.15.4e amendment has been standardized aimed at improving the communication reliability and at introducing explicit QoS support in the original communication protocol. Among the new proposed MAC protocols, the Time-Slotted Channel Hopping (TSCH) has been designed to guarantee reliable data delivery with bounded latency. To foster interoperability of TSCH networks, IETF created the 6TiSCH working group (WG) that aims at defining an open communication stack to integrate industrial networks into the existing IPv6 infrastructure. Specifically, the WG has defined the 6TiSCH architecture (IPv6 over the TSCH mode of IEEE802.15.4e) in which low-power and wireless devices can form a multi-hop low-power and lossy network using the IEEE 802.15.4e TSCH MAC, which is connected into the Internet through one or more border routers [3]. Since the IEEE 802.15.4e standard defines only the mechanisms for communication and leaves out of scope how transmission opportunities are managed, the 6TiSCH WG is currently standardizing the mechanisms to allocate the resources according to the QoS communication requirements. In addition, the WG is defining the mechanisms to integrate the RPL routing protocol, the standard routing protocol for low-power and lossy networks, to optimally operate on top of the TSCH layer, thus enabling multi-hop communication.

In addition to the mechanisms that are executed during regular network operations for the allocation of transmission opportunities for data delivery, 6TiSCH also defines how resources are allocated during bootstrap. Every 6TiSCH network is characterized by a *network formation* phase in which nodes progressively first join the TSCH network and then execute the RPL operations to collect topology information and compute the optimal routes for multi-hop data delivery. In order to specify the configuration of the network and the allocation of transmission opportunities at bootstrap, 6TiSCH is currently defining a standard called *minimal configuration* [4], which exploits a

Carlo Vallati, Simone Brienza, Giuseppe Anastasi are with the Department of Information Engineering, University of Pisa, Largo L. Lazzarino, Pisa, Italy. Giuseppe Anastasi is also affiliated with Smart Cities & Communities National Lab, CINI, Italy. Sajal K. Das is with the Department of Computer Science, Missouri University of Science & Technology, Rolla, MO USA.

C. Vallati, S. Brienza, G. Anastasi and S. K. Das, "Improving Network Formation in 6TiSCH Networks," in IEEE Transactions on Mobile Computing, vol. 18, no. 1, pp. 98-110, 1 Jan. 2019. doi: 10.1109/TMC.2018.2828835

static resource allocation, implemented by all the nodes, for the transmission of control messages at network formation.

Although different policies for the allocation of resources in 6TiSCH networks have been proposed in literature, e.g. [5][6], all of them focus on the management of resources for data transmission, after the initial formation phase. In addition, all these works start from the assumption that the network if fully operational, i.e., all the nodes are connected to the TSCH network and RPL has successfully found the optimal routes for data forwarding. However, a reliable and fast network formation cannot always be assumed. The TSCH network formation and the initial topology discovery operations of RPL can be unsuccessful or inefficient when a timely transmission of control messages is not guaranteed, as highlighted in [7] and [8]. For this reason, the initial resource allocation influences significantly the efficiency of the network formation: especially when large topologies are considered, the allocation of an insufficient number of transmission opportunities for control messages can lead to congestion and, consequently, to long convergence times of the TSCH and routing initial operations.

In this work, the performance of the 6TiSCH network at bootstrap is assessed. At the best of authors' knowledge, this is the first work investigating the performance of the 6TiSCH network during formation. Initially, it is shown that the allocation proposed in the 6TiSCH minimal configuration, can delay significantly the network formation and impair the performance of the routing algorithm thus resulting in disconnected networks. Specifically, first an analytical model capturing the dynamics of uninitialized nodes (i.e. nodes that are still waiting to join the network) is presented showing how the minimal configuration can result in long periods to join the network. Then, results from simulations are presented, showing that, the overall network formation phase can be long and that, the inefficient transmission of control messages can result in poor routes selection. Consequently, a dynamic algorithm to manage the allocation of resources during network bootstrap is proposed to ensure proper diffusion of control messages thus guaranteeing fast join to TSCH network and proper diffusion of routing information. The proposed approach is assessed by means of both simulations and real experiments.

The reminder of the paper has the following structure. In Section 2, an introduction of the network formation dynamics that characterize 6TiSCH networks is offered. Section 3 provides an overview of the related work through a critical analysis of the literature. Section 4 presents an evaluation of the 6TiSCH network formation, by analysis and simulation. Section 5 presents the proposed dynamic resource allocation algorithm. Section 6 presents a performance evaluation of the proposed methodology by means of simulation while Section 7 presents the results from real experiments. Finally conclusions are drawn in Section 8. For readers who are unaware of the technical details of the 6TiSCH architecture and the RPL protocol, a short introduction is offered in Appendix A, while a detailed presentation can be found in [2].

2 6TiSCH Formation Dynamics

In this section, an overall description of the network formation process of 6TiSCH networks is presented. For the sake of brevity, this section does not cover the technical details of the 6TiSCH architecture but focuses on the basic concepts of the TSCH MAC protocol and the 6TiSCH formation process. For a description of the technical background, i.e. an overview of the 6TiSCH architecture and the RPL routing protocol, the reader can refer to Appendix A.

The TSCH MAC has been specifically designed for WSNs that require high resiliency to interference and deterministic latency in data delivery. Unlike the traditional CSMA/CA contention based MAC, TSCH adopts timeslotted channel access in which all nodes are time synchronized. Time is divided into chunks of fixed length, called *timeslots*, that are grouped into a *timeslotframe or slotframe*. Each slotframe has a fixed length and its allocation repeats over time. Each timeslot is scheduled for communication: they can be assigned as *dedicated timeslots* to one node for communicating with a neighbor or they can be allocated as *shared timeslots* for broadcast communication. Dedicated timeslots are ensured to be contention free, so they are accessed without contention; on the other hand, channel access in shared timeslots is contended and requires the execution of a CSMA/CA like protocol.

In order to exploit frequency diversity and combat multi-path fading and interference, TSCH adopts a channel hopping technique. Specifically, a pre-defined channel sequence is shared among all the nodes and it is used to select a different operating frequency for each transmission. Concurrent transmissions on the same timeslot can be performed applying an offset to the channel hopping sequence, thus resulting in the adoption of different frequencies. Every node can follow the schedule to know when and on which frequency can transmit or data is expected.

The formation of 6TiSCH networks is a dynamic process composed of two different phases: the *formation of the TSCH network* and the *bootstrap of the RPL routing protocol*. Between the two phases there is no clear distinction as they are executed autonomously by each node. Specifically, a node, before being fully operational, has first to join the TSCH network to enable single-hop communication with its neighbors and then it has to join the logical topology of the routing protocol to populate its routing table and enable multi-hop data forwarding. The result of the operations performed by each node is a dynamic formation in which nodes progressively joins the TSCH network and the routing topology, one by one, until all of them are fully operational.

Both TSCH and RPL bootstrap operations are coordinated by a *central node* that has two distinct roles: the *TSCH coordinator* and the *RPL root*. The TSCH coordinator is in charge of broadcasting the Enhanced Beacons (EBs), the messages that advertise the presence of the network. Each EB specifies the network parameters that allow new nodes to initialize their TSCH instance. In addition, the periodic emission of EBs allows nodes to maintain time synchronization through the network. The RPL root node, instead,

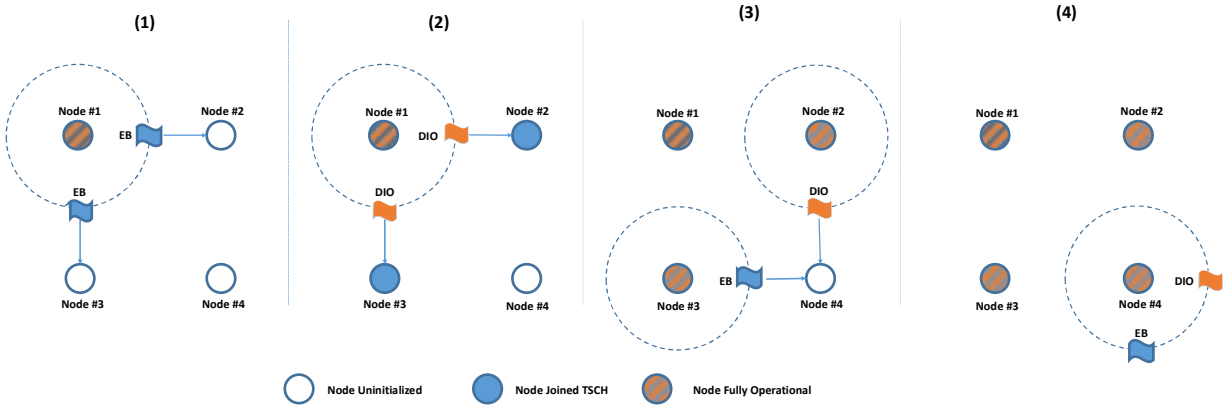

Figure 1. 6TiSCH Network Formation, example with 4 nodes.

has a different role as it represents the origin of the routing topology. Specifically, RPL builds a logical topology for data forwarding that is a Destination Oriented Acyclic Graph, DODAG for short. The center of this topology is the root node, usually implemented by the sink or concentrator of the network to which all (or the majority) of data traffic is directed. In addition of being the center of the routing topology, the root node is also responsible to trigger the RPL operations by emitting DODAG Information Object messages (DIO). Such messages, originated by the root node, are forwarded node by node through the network, thus allowing each node to discover the network topology. Each DIO message includes not only the routing information, required to reconstruct the network topology, but also the RPL settings necessary to initialize RPL functions.

In order to better illustrate the formation process, let us introduce a simple network formation example, pictured in Figure 1. Initially all the nodes are *uninitialized*, except for Node #1 that starts behaving both as TSCH coordinator and RPL node. Specifically, it broadcasts both EBs and DIO messages, the former on a periodic basis, the latter according to the Trickle algorithm, an adaptive algorithm that regulates the emission of control messages in RPL as specified in [9]. Uninitialized nodes wait for the reception of an EB. Since TSCH MAC exploits the channel hopping technique, through which transmissions are performed on different frequencies over time, based on a pre-defined channel hopping sequence, uninitialized nodes can listen for EBs scanning over different frequencies or on one fixed frequency. After the successful reception of an EB, the node can join the TSCH network, synchronizing with the neighbors, e.g. in Figure 1 (1) Nodes #2 and #3 receives an EB from Node #1. As a node joins the TSCH network it enables the reception of messages other than EBs and wait for the reception of the messages of the routing protocol. In particular, the node waits for the reception of a DIO message to initialize the RPL protocol, e.g. in Figure 1 (2) Nodes #2 and #3 receives a DIO from Node #1. When at least one DIO message is received, the node can initialize the local RPL instance and start forwarding data for multihop delivery. Even though the topology surrounding the node is not discovered, at least one neighbor is known and can be selected as preferred parent, i.e., the designated neighbor for data forwarding towards the root node. The selection of the preferred parent and the consequent route towards the root node are refined over time as more DIO messages are received and the network topology is unveiled. From this point on, the node is completely operational and starts broadcasting both EBs and DIO messages, e.g. , e.g. in Figure 1 (3) Node #2 and #3 broadcast both EBs and DIOs as they are fully operational. This allows other nodes that are not in direct communication with the central node to join the network progressively following the same procedure, e.g. in Figure 1 (4) Node #4 becomes fully operational after receiving an EB and a DIO from Nodes #2 and #3.

The broadcast of control messages is performed by nodes within shared timeslots. The allocation of such transmission opportunities during the network formation is defined in a document called 6TiSCH minimal configuration [4].

3 Related Work

The dynamics of the network formation for TSCH networks and the bootstrap of the RPL protocol have been studied separately in different research works. In [7] for instance, the TSCH formation phase has been modeled in order to derive guidelines for the allocation of shared timeslots for EBs broadcast in order to minimize the network formation process. In [10], instead, a simple random-based advertisement algorithm is proposed. In order to reduce the number of collisions, the transmission probability is tuned according to the number of neighbors. The efficiency of the RPL bootstrap has been also studied for wireless networks that adopt the classic CSMA/CA as MAC layer. In [8] it is shown how RPL, under certain assumptions, converges slowly in discovering the network topology, and consequently all the routes available, due to the trickle algorithm that can suppress the broadcast of DIO messages, [11].

Although the 6TiSCH network architecture is still under standardization, several research efforts have been considered different aspects. The majority of them focuses on the problem of network resource allocation in order to guarantee timed data delivery. For instance, in [5] a distributed dynamic scheduling algorithm is proposed. The algorithm

Figure 2. RPL dynamics over TSCH: (a) DIO Generation According to Trickle, (b) Actual DIO Transmission in TSCH

allows nodes to reserve timeslots on the fly in order to accommodate data traffic variations over time. In [12] a localized scheduling algorithm aimed at minimizing the end-to-end delay of traffic is proposed. The proposed approach exploits RPL information to allow the allocation of timeslots for data transmission during the RPL network formation. The resulting schedule guarantees that a packet can be delivered within one single slotframe, thus minimizing the overall end-to-end delay. The proposed scheduling algorithm, however, focus on managing the transmission opportunities for data transmission while they do not consider the allocation for the transmission of control messages. The only work considering both the network formation and allocation of timeslots for control messages is [6] in which *Orchestra*, a distributed scheduling algorithm for TSCH networks, is proposed. Orchestra allows nodes to autonomously schedule data and control timeslots at network formation using only local information. The proposed solution does not rely on inter-node schedule negotiation or on demand path reservation as it allocates timeslots in function of the sender's and receiver's identifiers (e.g. MAC address or unique network node ID). Three different types of timeslots are allocated in three different schedules (potentially with different periods): TSCH beacon timeslots, RPL traffic timeslots and application data timeslots. The resulting schedule, however, uses shared timeslots for control packets (e.g. EBs and RPL messages). Such timeslots are allocated statically at network formation and remains fixed regardless of the number of messages transmitted at the moment in the network. Although the work in [6] proposes an allocation of timeslots different from the minimal configuration, the authors do not specifically study the performance of the network at formation but analyze the performance of Orchestra only at the steady state.

At the best of our knowledge, this is the first work studying the performance of 6TiSCH networks at bootstrap and proposing a solution to allocate shared timeslots during network formation to both improve the time required by nodes to join the TSCH network and the performance of the routing algorithm. All the research works on 6TiSCH resource allocation has adopted the minimal configuration as initial schedule, however, without evaluating its effectiveness in guaranteeing a reliable network formation.

4 Network Formation Analysis on Minimal Configuration

The generation of control messages, EBs and DIOs, is performed by nodes that have already joined the network: EBs are generated on a periodic basis, while DIOs are generated according to the trickle algorithm. Figure 2 (a) illustrate an example of trickle dynamics: time is divided into subsequent generation intervals whose length is doubled every time starting from an initial value I_0. Within each generation interval, a DIO message is generated at a random instant. In case an inconsistent routing information is received the interval length is reset to I_0. This behavior allows, on one side, to minimize the energy consumption; on the other, to reduce the generation interval when fresh routing information is received to facilitate its diffusion.

Such control messages are generated by each node independently from the TSCH allocation and their transmission is performed on shared timeslots. Considering that EBs have precedence over the other messages, their transmission is performed in the next shared timeslot after their generation. Instead, DIOs are buffered and transmitted in the first transmission opportunity in which the node does not have an EB to transmit (assuming that no other traffic is generated during network formation).

The standard strategy for the allocation of timeslots to control messages is the 6TiSCH minimal configuration [4]. Such strategy statically allocates only one shared TSCH timeslot for the transmission of control messages. Specifically, the first timeslot in the slotframe at channel offset zero is allocated for the transmission of EBs and the broadcast of RPL messages. Although, such configuration might be optimal at the network steady state as it minimizes the number of timeslots allocated for control messages, it might increase significantly the number of collisions at network bootstrap, considering that in such initial phase, several DIOs are generated in a short interval.

An example of the resulting behavior is illustrated in Figure 2 (b), which shows the actual transmission of DIO messages with TSCH minimal configuration. Considering that the initial trickle interval adopted is usually significantly shorter than the slotframe duration (e.g. the minimal configuration suggests to adopt the default value from the RPL RFC [13] that is 8ms), during this phase several DIO messages are generated before the first shared timeslot, as shown in the left part of Figure 2 (a). Such messages, are buffered and transmitted over the following shared timeslots, as shown in Figure 2 (b).

In this section, we analyze the network formation process when the minimal configuration is adopted. First an analytical model is presented to evaluate the time required by a single node to join the network, then we assess the performance of the overall network formation by means of simulations.

4.1 Analytical Model

In this section, we model the behavior of a node that wants to join a 6TiSCH network under the minimal configuration. Our goal is to provide an analytical model for the time required by a single node to join the network. Our aim is not to offer a highly accurate model. Instead, we trade complexity for tractability and hence we introduce some simplifying assumptions. Our aim is to show that the time required by a node to join TSCH and RPL might be significant, as an effect of the current definition of the minimal configuration.

The behavior of a node that is joining the network is modeled through a Discrete Time Markov Chain. Specifically, three states are defined, as shown in Figure 3: the initial state N/N that represents uninitialized nodes, the transitional state J/N that represents nodes that joined the TSCH network but are still waiting to join the RPL topology, and the absorbing state J/J that represents nodes that are fully operational. Each node moves from N/N state to the J/N state with probability P_{tsch}, which is the probability for a node to join the TSCH network, while it moves from J/N to J/J with probability P_{rpl}, which is the probability for a node to join the RPL network. The probability matrix of the Markov Chain is the following:

$$P = \begin{bmatrix} 1 - P_{tsch} & P_{tsch} & 0 \\ 0 & 1 - P_{rpl} & P_{rpl} \\ 0 & 0 & 1 \end{bmatrix}$$

As can be seen the matrix has an absorbing state, so the average number of steps before entering in the absorbing state can be evaluated as:

$$t = \frac{1}{P_{tsch}} + \frac{1}{P_{rpl}}$$

which represents the average number of slotframes required for a node to join the network.

In order to express P_{tsch} and P_{rpl}, we consider that a node joins TSCH and RPL when it receives an EB and a DIO, respectively. To this aim, in the following, we assume that each node can buffer only one EB and one DIO at a time. Considering that 6TiSCH specifications ensure strict priority to the transmission of EBs over other types of messages, at each slotframe a node emits an EB if it has an EB buffered, while it emits a DIO if it has a DIO buffered *and* it does not have an EB buffered. With this in mind, let us define the following: P_{eb} as the probability that a node emits an EB; P_{dio} as the probability that a node emits a DIO; P_{msg} as the probability that a node emits any message, i.e. an EB or a DIO.

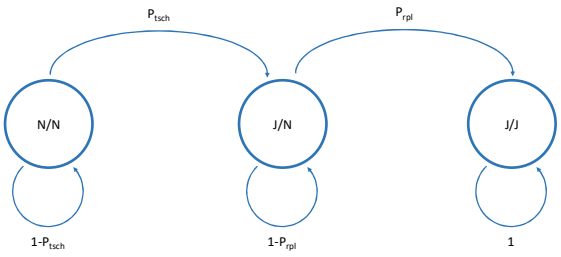

Figure 3. Discrete Markov Chain, graphical representation.

Consequently, P_{tsch} can be expressed as the probability for a node to correctly receive an EB from any of its N neighbors. Specifically, P_{tsch} can be expressed as the probability that any node, among the N nodes that have already joined the network, emits an EB and such message does not collide with other transmissions or it is not corrupted, i.e. the other $N-1$ nodes do not emit any message and the packet is successfully received. Specifically, P_{tsch} can be expressed as follows:

$$P_{tsch} = N\, P_{eb} \left(1 - P_{msg}\right)^{N-1} P_{eb-r}$$

where P_{eb-rx} is the probability that the EB is correctly received by the joining node. In this case P_{eb-r} can be expressed as follows:

$$P_{eb-r} = (1 - P_{loss}) \frac{1}{N_c}$$

as the joint probability that the message is not corrupted, assuming a lossy channel with constant loss probability P_{loss}, and the message is transmitted on the same channel on which the joining node is listening, considering a channel hopping sequence composed of N_c channels. The latter is required since the node joining the TSCH network is not synchronized with the channel hopping sequence.

Similarly, we can derive P_{rpl} that can be expressed as the joint probability that one node emits a DIO, while the other $N-1$ nodes do not emit any message. A lossy wireless medium with constant packet loss is modeled by conditioning on the probability that the packet is correctly received. Considering that after joining TSCH the node is synchronized with the channel hopping sequence, P_{rpl} can be expressed as follows:

$$P_{rpl} = N\, P_{dio} \left(1 - P_{msg}\right)^{N-1} (1 - P_{loss})$$

It is important to highlight that a node starts transmitting EBs and DIOs only after it joined the TSCH network. In the TSCH network, all the nodes are synchronized in time and move across different frequencies following a shared channel hopping sequence, in order to ensure that sender and transmitter are always on the same channel. For this reason this expression does not include the probability that sender and receiver are on the same channel.

If we assume that during network bootstrap only EBs and DIOs are transmitted, the probabilities P_{eb}, P_{dio} and P_{msg} can be calculated as follows. Considering that EBs have strict priority over the other messages, P_{eb} can be expressed as follows:

$$P_{eb} = P_{eb-buffere}$$

where $P_{eb-buffere}$ is the probability that a node has an EB buffered. The probability P_{dio} can be expressed as follows:

$$P_{dio} = \left(1 - P_{eb-buffered}\right) P_{dio-buffered}$$

where $P_{dio-buffered}$ is the probability that a node has a DIO message buffered waiting for transmission. The probability that a node emits a message, instead, either an EB or a DIO, can be expressed as the probability that a node transmits an EB or transmits a DIO:

$$P_{msg} = P_{eb} + P_{dio} = P_{eb-buffered} + \left(1 - P_{eb-buffered}\right) P_{dio-buffere}$$

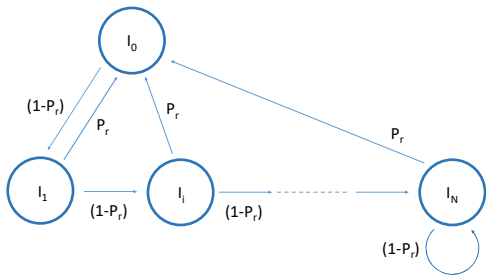

Figure 4. Embedded Discrete Time Markov Chain, graphical representation.

In order to evaluate P_{tsch}, P_{rpl} and P_{msg} we derive an expression for $P_{eb-buffere}$ and $P_{dio-buffered}$. In order to derive $P_{eb-buffered}$ we assume that each node produce an EB within a period I_{eb} selecting a random instant in $[0, I_{eb}]$. I_{eb} is assumed to be greater than the slotframe length L. This assumption is reasonable considering that a value $I_{eb} < L$ would result in transmitting only EBs since the transmission of EBs has strict priority over the transmission of DIOs. Following this assumption, at the beginning of each slotframe a node has an EB buffered with the following probability:

$$P_{eb-buffere} = \frac{L}{I_{eb}}$$

The characterization of $P_{dio-buffered}$, instead, is not trivial as the generation of DIOs is regulated by the trickle algorithm, which doubles the transmission interval I every time, until a maximum number of doublings N_D is performed. The value of I can also be reset if an event, called inconsistency, occurs, triggering the reset of the interval to the initial value I_0. In addition to this, a suppression mechanism is also included, to suppress the transmission of DIOs when a certain number of messages have been already overheard in an interval. In order to simplify the analysis and make it tractable, we assume that the suppression mechanism is disabled and that the probability of resetting the interval I is fixed and equal to P_r.

In order to obtain the average value of $P_{dio-buffered}$, we model the trickle behavior as a Semi-Markov Process (SMP) that consists of two components: (i) a discrete-time Markov chain (the Embedded Markov Chain) $\{X(n), n = 0,1, \dots N_D\}$ in which each state represents one of the possible trickle interval length I_i; (ii) the values $T_i = 2^i I_0$ that describe the time spent on each state from the moment the process entered in that state. The state representation is shown in Figure 4. From each state the state is reset to I_0 with probability P_r. Instead with probability $1 - P_r$, the state jumps to I_{i+1}, until the state I_{N_D} is reached. The probability matrix is the following:

$$P = \begin{bmatrix} P_r & (1-P_r) & \dots & \dots & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots \\ P_r & 0 & \dots & (1-P_r) & \dots & 0 \\ \dots & \dots & \dots & \dots & 0 & (1-P_r) \\ P_r & 0 & \dots & \dots & 0 & (1-P_r) \end{bmatrix}$$

The stationary distribution π of the Semi-Markov Process, can be evaluated from the stationary distribution of the embedded Markov chain, i.e. $\phi = \phi P$ and $|\phi|_1 = 1$, and the sojourn time in each state T_i as follows, [14] Ch. 9:

$$\pi_i = \frac{\phi_i T_i}{\sum_{j=0}^{N} \phi_j T_j}$$

which, in our case, results in the following:

$$\pi_0 = \frac{P_r}{P_r + 2^{N_D}(1-P_r)^{N_D} + \sum_{j=1}^{N_D-1} P_r\, 2^j (1-P_r)^j}$$

$$i \epsilon [1, N_D - 1]$$

$$\pi_i = \frac{P_r\, 2^i (1-P_r)^i}{P_r + 2^{N_D}(1-P_r)^{N_D} + \sum_{j=1}^{N_D-1} P_r\, 2^j (1-P_r)^j}$$

$$\pi_{N_D} = \frac{2^{N_D}(1-P_r)^{N_D}}{P_r + 2^{N_D}(1-P_r)^{N_D} + \sum_{j=1}^{N_D-1} P_r\, 2^j (1-P_r)^j}$$

Similarly to the generation of EBs, the probability of having a DIO buffered in each state $\mathrm{P}(\mathrm{P}_{dio-buffered} \mid I_i)$ can be expressed as follows:

$$\mathrm{P}(\mathrm{P}_{dio-buffered} \mid I_i) = \begin{cases} 1 & if\ L \geq I_i \\ \dfrac{L}{I_i} & if\ L < I_i \end{cases}$$

Hence the average probability of having a DIO buffered $P_{dio-buffered}$ can be expressed as follows:

$$P_{dio-buffered} = \sum_{i=0}^{N_D} \pi_i\, \mathrm{P}(\mathrm{P}_{dio-buffered} \mid I_i) =$$

$$\frac{2^{N_D}(1-P_r)^{N_D} \max\left(\frac{L}{2^{N_D} I_0}, 1\right) + \sum_{i=0}^{N_D-1} P_r\, 2^i (1-P_r)^i \max\left(\frac{L}{2^i I_0}, 1\right)}{P_r + 2^{N_D}(1-P_r)^{N_D} + \sum_{j=1}^{N_D-1} P_r\, 2^j (1-P_r)^j}$$

Eventually, a closed form for both P_{tsch} and P_{rpl} can be expressed as follows:

$$P_{tsch} = \frac{N\, L}{N_c\, I_{eb}} \left(1 - \frac{L}{I_{eb}}\right)^{N-1} \left(1 - P_{dio-buffered}\right)^{N-1} (1 - P_{loss})$$

$$P_{rpl} = N\, P_{dio-buffere} \left(1 - \frac{L}{I_{eb}}\right)^{N} \left(1 - P_{dio-buffere}\right)^{N-1} (1 - P_{loss})$$

Using this model, the average time for a node to join, as a function of the number of neighborhood that have already joined, can be evaluated. Figure 5 shows the results obtained with the following parameters: N_{ch}=16; $I_{eb} = 4s$; $L = 1.9s$; $I_{min} = 32ms$; $N_D = 10$; $P_r = 0.2$. Two different values of P_{loss} are considered, 0 and 0.2, respectively. As the results show, when the number of nodes that already have joined the network grows, the time required for a new node to join increases up to hundreds of seconds. In order to validate the results obtained with the analytical model, a simple ad-hoc event simulator that simulates a node joining a network of N nodes has been written in Python. For each scenario 2000 independent replications are run. The average node joining time is reported along with the 95% confidence intervals. As can be seen the analytical model approximates simulation results with an acceptable level of accuracy.

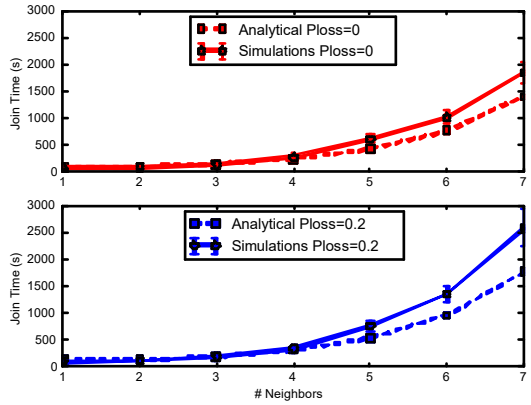

Figure 5. Average node joining time, analytical model vs simulations.

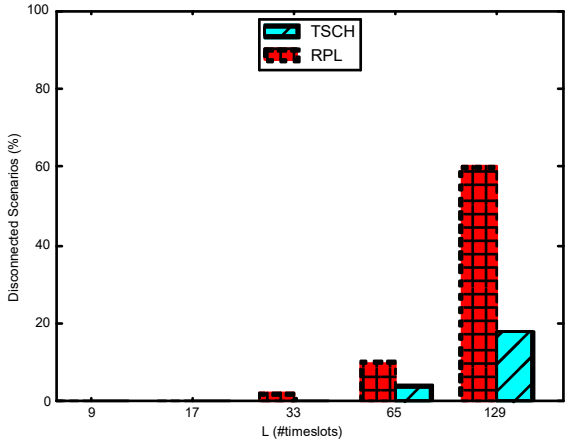

Figure 6. Disconnected scenario ratio.

4.2 Evaluation of the 6TiSCH Minimal Configuration

In order to assess the overall performance of the formation of the whole network without the assumptions performed to derive the analytical model, we run some preliminary simulations. The TSCH implementation available in the Contiki OS[1], a popular operating system for sensor nodes, is adopted. Simulations are run exploiting the Cooja tool [15], which allows to reproduce the behavior of wireless sensor networks. In our simulations, Cooja motes are adopted in order to avoid RAM and ROM limitations. To simulate a realistic channel, we adopted the Multipath Ray-tracer Medium (MRM) model, a propagation model that implements ray-tracing techniques with various propagation effects, e.g., multi-path, refraction, diffraction, etc. The TSCH MAC layer is configured according the 6TiSCH minimal configuration. The slotframe size L is varied from 9 to 127 timeslots, in order to test the performance with different configurations. The EB broadcast period is set to 4s. The overall simulation parameters are summarized in Table 1.

The simulated network topology includes 49 nodes placed in a 7x7 grid. Power and channel model parameters are set to allow each node to communicate with its close neighbors without packet loss. Each simulation represents approximately 2 hours of operations (8000 seconds). In order to obtain statistically sound results, 20 independent replications with different seeds are run for each scenario.

Figure 6 summarizes the outcome of each scenario reporting the number of simulations in which the network resulted in being disconnected or partitioned because at least one node could not successfully join the TSCH network or the RPL DODAG. As can be seen, when the slotframe is configured to be longer than 100 timeslots in approximately 20% of the considered scenarios the TSCH network resulted in being only partially formed, while in more than half of the scenarios (around 60%) the routing topology that resulted from routing operations was disconnected.

This can be explained considering the results in Figure 7 that shows the cumulative distribution function (CDF) of the *joining time* of nodes, defined as the time between the beginning of the simulation and the moment in which a node is fully operational, i.e. it is connected to the TSCH network and it has discovered at least one neighbor to join the RPL DODAG. As can be seen, the large number of nodes that could not join the network is the result of the long delay experienced by the nodes in receiving at least one EB and one DIO message from nodes that have already joined the network. It is important to highlight that the time required for joining the network is strictly dependent on the slotframe length and it is the reason why the number of scenarios in which there is at least one node not connected is higher with longer slotframes.

Long overhead in network bootstrap can be explained

Table 1. Simulation parameters

TSCH timeslot duration	15ms
RPL Redundancy Threshold	9
RPL DIO Minimum Interval	32ms
RPL DIO Interval Doublings	20 times
RPL Objective Function	MRHOF - ETX
EB generation interval	4s
Number of Channels	16

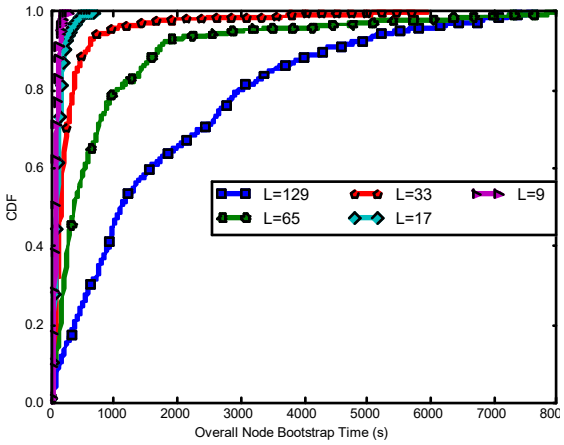

Figure 7. Node joining time distribution with minimal configuration.

1 Contiki OS, http://www.contiki-os.org

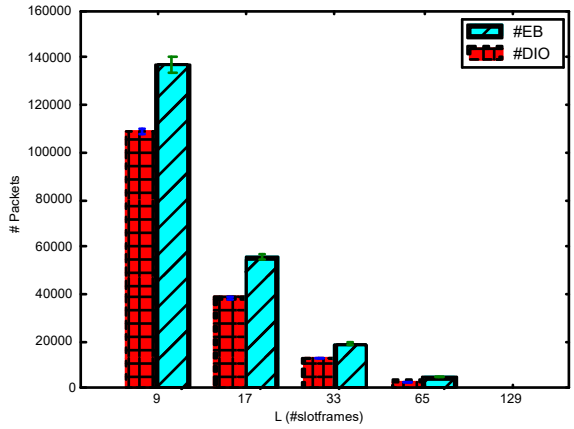

Figure 8. Overall number of received packets.

with the reduced diffusion of control messages for both TSCH and RPL operations. Figure 8 reports the overall number of messages received by the nodes in the network for all the configurations. Both the number of EBs and DIOs are reported. As can be observed, the more the slotframe length increases, the more the number of received messages (dramatically) decreases, due to the reducing number of timeslots that can be used to broadcast such type of messages and the increasing likelihood of collision.

5 6TiSCH Shared Timeslot Allocation

As shown in the previous sections, the allocation of shared timeslots as in the minimal configuration can lead to poor performance. Although increasing the number of shared timeslots can be beneficial for both TSCH network formation and routing, it would result in a waste of resources at the steady state, when the network is formed and the amount of control messages is reduced significantly. This is specifically obvious considering the trickle algorithm, that dynamically adjusts the transmission window in order to broadcast DIO messages at higher rate during the network bootstrap, whereas it scales down to a few messages per hour when the routing operations are completed and the topology does not change.

As shown in the performance evaluation, the static allocation proposed in the minimal configuration leads to delay in the diffusion of routing information since packets are buffered waiting for the next shared timeslot. Such delay does not only postpone the completion of the formation but also can lead to the instability of the routing process,

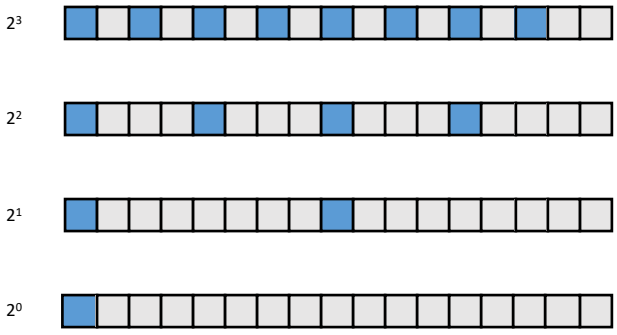

Figure 9. Shared slots allocation example.

as outdated routing information is also broadcast in some occasions. On the other hand, when the network is at the steady state, the trickle algorithm increases significantly the interval length, thus reducing the number of messages. In this case, a minimal allocation might be sufficient to send messages that are, instead, sporadic, as changes in the topology rarely occur. A fixed allocation does not master the tradeoff between network formation performance and the resource utilization. Hence, in this paper we propose a dynamic allocation strategy for shared TSCH timeslots that can adapt to the variation in the number of control messages sent in the network over time. The rationale behind the proposed algorithm is to have each node adapting the allocation of shared timeslots.

In order to optimize the association of new nodes to the TSCH network and reduce the waiting time for RPL control messages, shared timeslots are allocated following the guidelines provided in [7]. Specifically, timeslots are allocated equally spaced within the slotframe. In order to ease the dynamic allocation/deallocation of timeslots over time, the number of shared timeslots is set as 2^m, according to a parameter m, called the *allocation exponent*. Setting the number of allocated timeslots as a power of 2 ensures that timeslots can be kept equally spaced without the need for relocation. Figure 9 shows an example of how the allocation can be performed and updated over time: as m increases/decreases, more/less timeslots are allocated/deallocated easily still keeping their schedule equally spaced without the need for relocating them. Such configuration still allows the schedule to be reduced down to the minimal configuration: setting the allocation exponent to 0, thus allocating only the first timeslot.

The allocation of shared timeslot is performed periodically, every τ slotframes. The number of shared timeslots allocated by each node is based on an estimation of the rate of the control messages (both EBs and RPL messages) transmitted within a neighborhood. Essentially, each node estimates the number of control messages generated by itself and its neighbors. To this aim, it is assumed that control messages, both DIOs and EBs, include a transmission sequence number kept on a per node basis. The sequence number is introduced to allow the evaluation of the number of control messages produced by each node. Although such estimate could be obtained also from the number of messages transmitted by each neighbor, it might not be accurate as control messages can be produced but not transmitted, e.g. because the local buffer is full or the reception failed due to collision. Every time a control packet is received by a node, the node updates s_i, i.e., the overall number of packets generated by node i since the last allocation, by evaluating the difference between the sequence number received and the one from the last reception.

When the allocation is performed, 2^m shared timeslots are allocated using the following criteria:

$$\min m \mid 2^m \geq \frac{\sum_i s_i}{\tau}$$

where m, the *allocation exponent*, is selected as the minimum exponent to ensure that all the nodes in the neighborhood can transmit successfully their control messages in the next

Algorithm 1. Shared Slot Allocation Procedure.

```
1 ReceiveControlMessage():
2     sn ← Message sequence number
3     lsn_i ←Last sequence number received from node i
4     m_i ← Allocation exponent advertised in the message

5     n ← sn - lsn_i
6     s_i ← s_i + n

7 AllocateSharedSlots():
8     for i in neighbors
9         s ← s_i + s
10        s_i ← 0
11    s ← s / τ

12    m ← ⌈ log2 (s) ⌉
13    m ← min( m, M )
14    AllocateTimeslots(m)

15    m̂ ← max( m_i )
16    if m̂ > m
17        AllocateTimeslotsListenOnly( m̂ – m )
```

period, assuming a steady generation rate. The sum also includes the s_i calculated for the node itself. In order to limit the allocation of timeslots, a maximum allocation exponent M is set. This will limit to 2^M the number of timeslots that can be allocated. This maximum limit on the number of shared timeslots has a twofold purpose: it sets an upper-bound limit to the amount of energy consumed by the nodes for the bootstrap phase and keeps free a certain amount of timeslots for the initial allocation of slots for data forwarding. As soon as the network bootstrap phase is completed, the number of shared timeslots allocated decreases with the amount of routing messages. In order to release definitively some timeslots for the allocation of data transmission, the maximum allocation exponent can be reduced as the network bootstrap ends.

The allocation policy proposed assigns the number of shared timeslots according to the information obtained by neighbors. However, since the neighborhood of each node is different, the allocation performed by nodes can differ even if they are neighbors. In order to ensure proper transmission and reception of messages, every node i advertises its current allocation exponent m_i inside its control messages. When the allocation process is run, every node also evaluates the following:

$$\hat{m} = \max_i m_i,$$

which is the maximum exponent received from its neighbors. If $\hat{m} > m$ holds, an additional number of shared timeslots is allocated as listen-only timeslots. Basically, a total number of $2^{\hat{m}}$ timeslots are allocated, but only the first 2^m timeslots are allocated for both transmission and reception, while the others $2^{\hat{m}-m}$ timeslots are allocated in listen-only, thus ensuring the reception of messages from all the neighbors.

The overall pseudo-code of the algorithm is illustrated in Algorithm 1. The steps between lines 1-6 are performed every time a control message is received: first, the number of messages generated by the neighbor i is assessed (line 5), then the message counter is updated (line 6). Between lines 7-17 the steps performed periodically for allocating shared timeslots are presented: first, the overall number of messages produced during the last period is computed (lines 8-10), then the average number of message per timeslotframe is evaluated (line 11) and finally the allocation exponent is computed and the slots allocated (lines 12-14). Eventually, if required, additional listen-only timeslots are allocated (lines 15-17).

6 Performance Evaluation

The proposed dynamic allocation algorithm is evaluated by means of simulations. In order to provide a term of comparison, its performance is compared against Orchestra. Although Orchestra does not focus on the network formation specifically, it is the only algorithm that includes a schema for allocating shared slots for both EBs and RPL control messages [6]. In order to make the two strategies comparable, the allocation period of Orchestra broadcast slotframe and the receiver-based slotframe have been set equal to L. The proposed dynamic solution has been configured in each scenario with the allocation period τ set as the minimum number of slotframes that ensures a period of at least one second. The same simulation scenarios and settings presented in Section 3.2 are adopted.

A preliminary analysis of all the experiments with different values of L highlighted that both Orchestra and the proposed dynamic allocation schema allow to avoid scenarios in which nodes are disconnected. This is obtained by allocating more shared timeslots for the transmission of EBs and RPL messages, thus allowing all the nodes to join both the TSCH network and the RPL DODAG within the 2 hours of simulation.

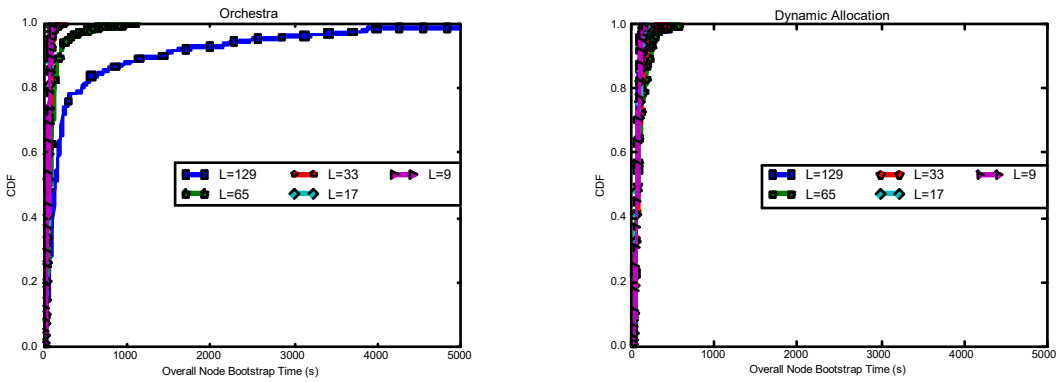

Figure 10. Node Joining time distribution. Orchestra (left) vs dynamic allocation (right)

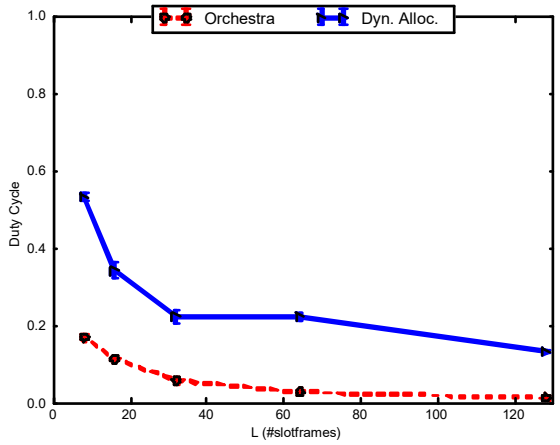

Figure 11. Average node duty cycle during bootstrap.

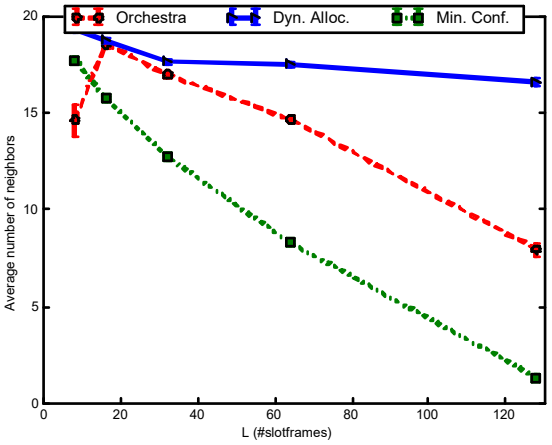

Figure 12. Average number of neighbors per node.

An insight on the overall time required by nodes to be fully functional is shown in Figure 10 which shows the CDF of the joining time experienced by nodes with Orchestra (the figure on the left) and with the proposed dynamic solution (the figure on the right). Compared with the results obtained with the minimal configuration Figure 7, both Orchestra and the proposed solution help in reducing the overall joining delay. However, as can be seen, Orchestra still results in a long joining time for some nodes when large slotframes are adopted. A detailed analysis of such delay reveals that these nodes join almost immediately the TSCH network and, instead, wait a long period to receive the first DIO message from a neighbor thus delaying significantly the initialization of the routing protocol. This can be explained considering the fixed allocation for RPL messages performed by Orchestra, which results a large number of collisions during the initial convergence of the routing protocol.

In order to evaluate the cost of the initial allocation in Figure 11 the average node duty cycle during bootstrap is reported. The duty cycle is defined as the ratio between the number of the scheduled slotframes in which the node is active (shared slotframes or dedicated slotframes for transmission/reception) and the overall timeslotframe length. This metric evaluates indirectly the energy consumption of the nodes. As expected the proposed scheduling results in a higher energy consumption as it allocates more timeslots for control message delivery. However, right after the end of the bootstrap phase, the maximum number of shared timeslots M can be reduced, thus minimizing the energy consumption and releasing more transmission opportunities for data delivery.

A larger amount of RPL messages successfully sent and delivered thought the network helps every node to gain a more advanced view of the network topology. Specifically, each node can discover a larger number of neighbors and potentially more routes for data delivery. Figure 12 reports the average number of neighbors discovered by each node with different allocation strategies. As expected, both Orchestra and the proposed allocation strategy helps in discovering a higher number of neighbors, as a larger number of messages are broadcast compared with the minimal configuration. Also in this case, the proposed strategy helps in discovering more neighbors than Orchestra, due to the fact that more DIO messages are transmitted.

Discovering more neighbors also results in acquiring more routes for data forwarding. This increases the decisional space for the routing protocol that should be able to converge quickly in finding better routes for the communication towards the root node. In Figure 13 we show the distribution of the node stretch at the end of the simulation considering different allocation schema. The node stretch is a metric that measures the quality of the route selected by a node for upward data forwarding. Specifically, it is defined as the difference between the cost of the route selected by the node (in terms of the end-to-end average number of transmissions, i.e. the expected transmission count ETX [16]) and the cost of the shortest path [8]. As can be seen both Orchestra and the proposed dynamic allocation allow nodes to discover better routes than the minimal configuration. The larger number of neighbors discovered through the proposed allocation mechanism, however, does not produce a significant advantage in terms of stretch, which is even higher for some of the nodes compared with orchestra.

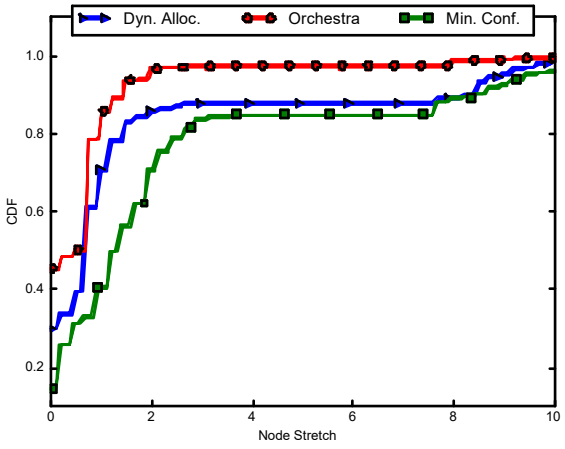

Figure 13. Node stretch distribution, scenario with L=65.

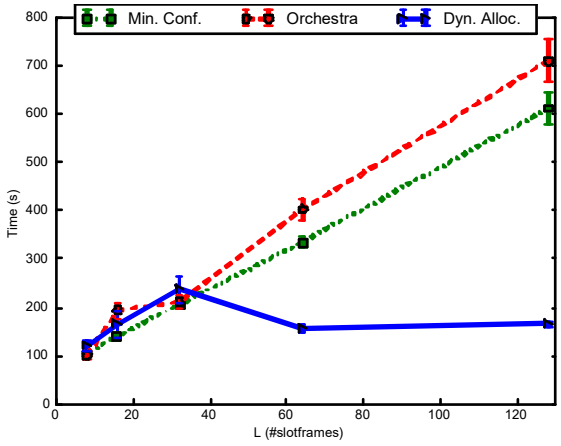

Figure 14. Average time on a preferred parent, scenario with L=65.

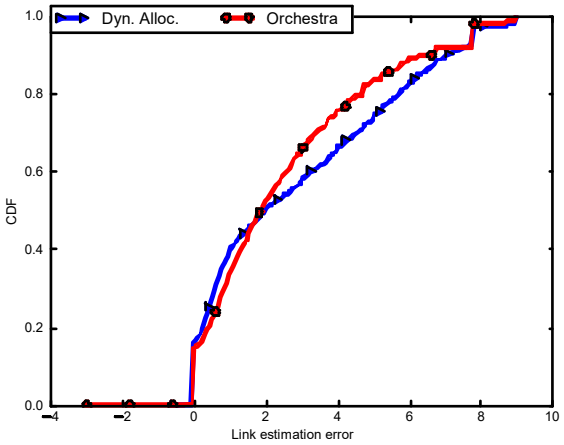

Figure 15. Link error estimation distribution, scenario with L=65

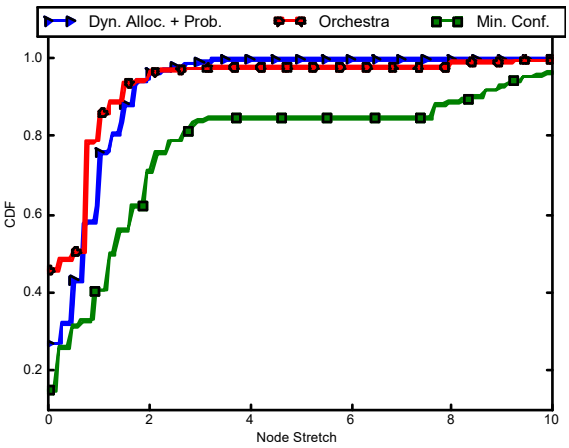

Figure 16. Node stretch distribution with active probing.

This behavior can be explained considering how the default link quality estimation in RPL works. Since the RPL protocol does not specify a mechanism to estimate the quality of links towards neighbors, different approaches have been proposed in literature [17]. The default strategy adopted in Contiki OS RPL implementation estimates the link quality using a passive monitoring approach in which packets transmitted to the preferred parent, data packets or other packets like in this case TSCH keep-alive packets, are exploited to derive the link quality. Such approach, however, measures the quality only towards the current preferred parent. In order to obtain an almost exhaustive evaluation of all the links towards the other neighbors, a policy that opportunistically favors the selection of recent discovered neighbors as preferred parent is implemented. Specifically, when a link to a new neighbor is discovered, the link is by default assumed to be good. By doing this, the new neighbor is likely selected as the new preferred parent and a link quality estimate is obtained by passively monitoring the data traffic. Although simple, such policy facilitates rotation among next-hop candidates but it results in routing instability during the network bootstrap. Moreover, the accuracy of the link quality estimation is significantly influenced by both the pattern of traffic data, i.e. a sporadic data transmission results in less accurate link estimation, and the frequency of the discovery of new neighbors, i.e. as new neighbors are discovered they are likely selected as new preferred parent thus resulting in varying period for link quality assessment.

In order to explain the degradation of the routes experienced with the proposed schema in comparison with Orchestra, in Figure 14 the average time each node spends on a preferred parent is reported. As can be seen, since the proposed approach allows more neighbors to be discovered, the amount of time each node spends with one specific preferred parent is dramatically reduced, since every time a new neighbor is discovered it is selected as new preferred parent. The reduction of the time with each preferred parent leads to a less accurate link quality estimation as confirmed in Figure 15 that reports the distribution of the error in the estimation of the quality of each link, defined as the difference between the theoretical ETX of a link and the actual ETX estimated by a node. As can be seen, the reduced time spent in assessing the quality of links increases the likelihood of underestimating their quality. The latter can eventually lead to wrong routing decisions as initially reported in Figure 13.

In order to overcome this issue, an active monitoring strategy has been adopted [18]. With this strategy, special control messages, called probe packets, are sent over each link to derive link quality information. In order to obtain a conservative selection of the preferred parent and avoid the selection of a new neighbor every time a new preferred parent is discovered, the protocol is configured to assume that neighbors that are recently discovered have bad link quality. This configuration avoids route instability, as a new neighbor can be selected as the new preferred parent only after an accurate estimation of its link quality. Such solution could be effective, especially with the proposed dynamic allocation scheme, as the shared slots allocated for the transmission of control messages could be exploited also for the transmission of probing messages.

In order to test the performance of the proposed solution when active probing is employed an additional set of simulations is run. Specifically, the active probe imple-

mented in Contiki RPL is enabled. The active probe of Contiki RPL is implemented to probe neighbors, other than the preferred parent, every 120s on a round robin fashion. The default link quality in terms of ETX is configured to 5, instead of the default value 2. Figure 16 shows the distribution of the node stretch. As can be seen the stretch of all the node in the network significantly improves as result of a more accurate link quality estimation that ensure the selection of routes based on correct information.

7 Testbed Experiments

In order to confirm the results obtained with simulations, a set of experiments is run on a real testbed. To this aim, the IoT testbed described in [18] is exploited. The testbed is composed of 23 wireless sensors deployed in office spaces, student labs, and corridors of two floors in the Department of Information Engineering of the University of Pisa. Each sensor node is a TelosB mote, equipped with an MSP430 micro-controller that can run the Contiki operating system. IEEE802.15.4 connectivity is provided through the cc2420 wireless chip equipped with an external 5dBi antenna. The same Contiki code used for Cooja simulations is loaded on the testbed to assess the performance of the minimal configuration, orchestra and the proposed dynamic allocation.

In order to assess the performance with different network densities, two scenarios with different transmission powers are considered, i.e. 0 dBm and -7 dBm respectively. Experiments are run with the same parameters shown in Table 1, while the slotframe size is fixed to L=129. Each experiment is executed for 1hr. Results shown that only the proposed dynamic allocation succeeded in having all the nodes to join correctly the network within one hour in all the scenarios. Orchestra and the minimal configuration, instead, resulted in having only 93% and 97% of nodes, respectively, joined correctly. Figure 17 shows the CDF of the overall node bootstrap time of the nodes that successfully joined the network in both the scenarios with 0 dBm and -7 dBm. Consistently with simulations, the results show that both orchestra and the proposed dynamic allocation help in reducing the time required by nodes to join in comparison with the minimal configuration that is confirmed to lead to higher bootstrap times in the order of hundreds of seconds. The proposed approach is confirmed to further reduce the overall node bootstrap time compared with orchestra. Such advantage, although reduced, is also confirmed in the scenario with 0 dBm transmission power. As expected, in comparison with the results obtained with -7 dBm, a higher transmission power reduces the overall node joining time with all the configurations. This can be explained considering that a higher transmission power reduces the size of the network, thus minimizing the overall number of messages required to have all the nodes joining the network.

8 Conclusion

In this paper, an insight on the network bootstrap phase of 6TiSCH networks is offered. First a performance evaluation of the minimal configuration is carried out showing how a static allocation of transmission opportunities can result in poor performance when large deployments are considered. Based on these results, a dynamic scheduling algorithm is proposed to drive the allocation of shared timeslots during network formation. The proposed schema is designed to allow the transmission of EBs and RPL control messages to ensure proper TSCH network formation and execution of routing functionalities. The proposed schema is evaluated by means of simulations and real-world experiments against Orchestra. Performance evaluation results showed that the proposed algorithm can significantly enhance both reliability and efficiency of the network formation.

Acknowledgment

We thank the anonymous reviewers for their many insightful comments and suggestions that helped to improve significantly the quality of the paper.

References

[1] T. Watteyne et al., "Industrial Wireless IP-Based Cyber –Physical Systems," in Proceedings of the IEEE, vol. 104, no. 5, pp. 1025-1038, May 2016.

[2] Domenico De Guglielmo, Simone Brienza, Giuseppe Anastasi, IEEE 802.15.4e: A survey, Computer Communications, Volume 88, 15 August 2016, Pages 1-24.

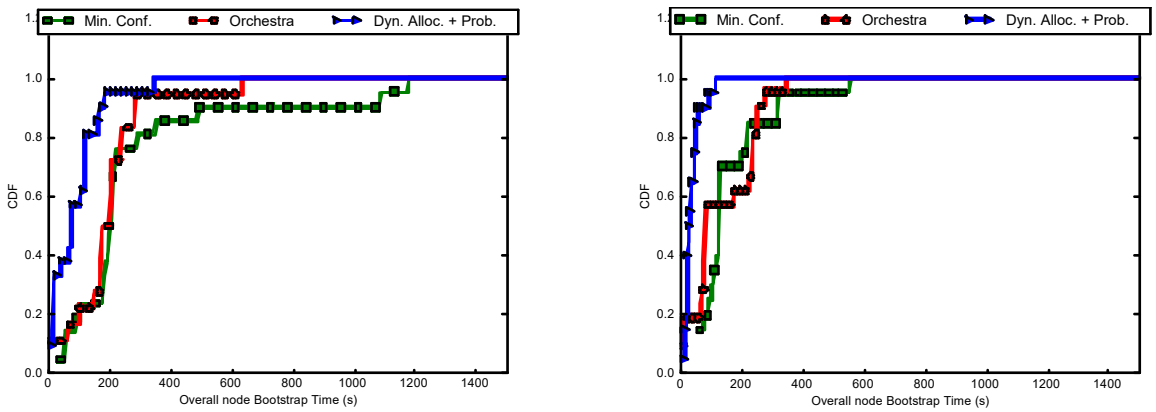

Figure 17. Node Joining time distribution in testbed experiments, -7dBm (left) and 0dBm (right) transmission power.

[3] M. R. Palattella et al., "Standardized Protocol Stack for the Internet of (Important) Things," in IEEE Communications Surveys & Tutorials, vol. 15, no. 3, pp. 1389-1406, Third Quarter 2013.

[4] Vilajosana, X., K. Pister and T. Watteyne. "Minimal 6TiSCH Configuration-draft-ietf-6tisch-minimal-21." IETF: Fremont, CA, USA (2017).

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

[6] Simon Duquennoy, Beshr Al Nahas, Olaf Landsiedel, and Thomas Watteyne. 2015. Orchestra: Robust Mesh Networks Through Autonomously Scheduled TSCH. In Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems (SenSys '15). ACM, New York, NY, USA, 337-350.

[7] D. De Guglielmo, S. Brienza and G. Anastasi, "A Model-based Beacon Scheduling algorithm for IEEE 802.15.4e TSCH networks," 2016 IEEE 17th International Symposium on A World of Wireless, Mobile and Multimedia Networks (WoWMoM), Coimbra, 2016, pp. 1-9.

[8] C. Vallati and E. Mingozzi, "Trickle-F: Fair broadcast suppression to improve energy-efficient route formation with the RPL routing protocol," 2013 Sustainable Internet and ICT for Sustainability (SustainIT), Palermo, 2013, pp. 1-9.

[9] Levis, P., Patel, N., Culler, D., & Shenker, S. (2004, March). Trickle: A self-regulating algorithm for code propagation and maintenance in wireless sensor networks. In Proc. of the 1st USENIX/ACM Symp. on Networked Systems Design and Implementation.

[10] D. De Guglielmo, A. Seghetti, G. Anastasi, and M. Conti, "A performance analysis of the network formation process in IEEE 802.15.4e TSCH wireless sensor/actuator networks", in Proceedings of 2014 IEEE Symposium on Computers and Communication (ISCC), 23-26 June 2014.

[11] T. Clausen, U. Herberg and M. Philipp, "A critical evaluation of the IPv6 Routing Protocol for Low Power and Lossy Networks (RPL)," 2011 IEEE 7th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob), Wuhan, 2011, pp. 365-372.

[12] I. Hosni, F. Theoleyre, N. Hamdi, "Localized Scheduling for End-to-End Delay Constrained Low Power Lossy Networks with 6tisch". IEEE Symposium on Computers and Communications (ISCC), Messine, Italy, June 2016

[13] T. Winter, P. Thubert, A. Brandt, J. Hui, R. Kelsey,P. Levis, K. Pister, R. Struik, J. Vasseur, and R. Alexander, "RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks," RFC 6550, Mar. 2012.

[14] Stewart, William J. Probability, Markov chains, queues, and simulation: the mathematical basis of performance modeling. Princeton University Press, 2009.

[15] Osterlind, F., Dunkels, A., Eriksson, J., Finne, N., & Voigt, T. (2006, November). Cross-level sensor network simulation with cooja. In Local computer networks, proceedings 2006 31st IEEE conference on (pp. 641-648). IEEE.

[16] De Couto, D. S., Aguayo, D., Bicket, J., & Morris, R. (2005). A high-throughput path metric for multi-hop wireless routing. Wireless Networks, 11(4), 419-434.

[17] E. Ancillotti, R. Bruno, M. Conti, E. Mingozzi and C. Vallati, "Trickle-L2: Lightweight link quality estimation through Trickle in RPL networks," Proceeding of IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks 2014, Sydney, NSW, 2014, pp. 1-9.

[18] C. Vallati, E. Ancillotti, R. Bruno, E. Mingozzi, G. Anastasi, Interplay of Link Quality Estimation and RPL performance: an Experimental Study, Proceedings of the 13th ACM International Symposium on Performance Evaluation of Wireless Ad Hoc, Sensor, and Ubiquitous Networks (ACM PE-WASUN 2016) - co-located with ACM MSWiM 2016, Valletta, Malta, November 13-17, 2016.

[19] Thubert, P., Watteyne, T., Struik, R., & Richardson, M.. "An Architecture for IPv6 over the TSCH mode of IEEE 802.15. 4". draft-ietf-6tisch-architecture-11. (2016) IETF Draft.

[20] IEEE802.15.4e: IEEE standard for local and metropolitan area networks. Part 15.4: Low-Rate Wireless Personal Area Networks (LRWPANs) Amend- ment 1: MAC Sublayer, Institute of Electrical and Electronics Engineers Std., Apr. 2012.

Carlo Vallati is Assistant Professor at the Department of Information Engineering of the University of Pisa. He received a Master's Degree (magna cum laude) and a PhD in Computer Systems Engineering in 2008 and 2012, respectively, from the University of Pisa. In 2010, he visited the Computer Science department of the University of California at Davis. He is co-author of +40 peer-reviewed papers in international journals and conference proceedings. He has been involved in the project BETaaS, Building the Environment for the Things as a Service, funded by the European Union under the 7th Framework Program and in several research projects supported by private industries. He has served as a program committee member for more than 20 international conferences and workshops and as Workshop Chair for the 2017 edition of the IEEE IoT-SoS workshop. He is currently serving as TPC Co-Chair for the IEEE SmartSys 2018 workshop co-located with IEEE SMARTCOMP and on the editorial board of two international journals, the "Journal of Reliable Intelligent Environments", Springer and "Applied Sciences", MDPI.

Simone Brienza. Is a Post-Doctoral researcher at the Dept. of Information Engineering at the University of Pisa, Italy. He received a Master's Degree and a PhD in Computer Systems Engineering in 2012 and 2016, respectively, from the University of Pisa.

Giuseppe Anastasi is Full Professor and Head at Dept. of Information Engineering at the University of Pisa, Italy. He is also the Director of the CINI National Smart Cities Lab. His research interests include pervasive computing, sensor networks, sustainable computing, and ICT or smart cities. He has contributed to many research programs funded by both national and international institutions. He has co-edited two books and published more than 140 papers in international journal and conferences. Dr. Anastasi is serving as General Co-chair of IEEE SMARTCOMP 2018. He has served Associate/Area Editor of Pervasive and Mobile Computing, Computer Communication and Sustainable Computing.

Sajal K. Das is a professor of Computer Science and Daniel St. Clair Endowed Chair at Missouri University of Science and Technology. During 2008-2011, he served the NSF as a Program Director in the Computer Networks and Systems division. His research interests include wireless sensor networks, cyber-physical systems and IoT, smart environments (smart city, smart grid and smart health care), cloud computing, cyber security, and social networks. He has published over 700 papers, holds 5 US patents and coauthored 4 books. His h-index is 80 with more than 26,000 citations. He is a recipient of 10 Best Paper Awards at conferences like ACM MobiCom and IEEE PerCom, and IEEE Computer Society's Technical Achievement Award for pioneering contributions to sensor networks and mobile computing. He serves as founding Editor-in-Chief of Pervasive and Mobile Computing (PMC) Journal, and as Associate Editor of several journals including IEEE Transactions of Mobile Computing and ACM Transactions on Sensor Networks.