MABO-TSCH: Multi-hop And Blacklist-based Optimized Time Synchronized Channel Hopping

Article *in* Transactions on Emerging Telecommunications Technologies · August 2017

DOI: 10.1002/ett.3223

CITATIONS
41

READS
1,201

3 authors:

Pedro Henrique Gomes
Ericsson
42 PUBLICATIONS **822** CITATIONS
SEE PROFILE

Thomas Watteyne
National Institute for Research in Computer Science and Control
185 PUBLICATIONS **6,515** CITATIONS
SEE PROFILE

Bhaskar Krishnamachari
University of Southern California
507 PUBLICATIONS **23,413** CITATIONS
SEE PROFILE

TRANSACTIONS ON EMERGING TELECOMMUNICATIONS TECHNOLOGIES
Trans. Emerging Tel. Tech. 2017; **00**:1–15
DOI: 10.1002/ett

SPECIAL ISSUE ARTICLE

MABO-TSCH: Multi-hop And Blacklist-based Optimized Time Synchronized Channel Hopping

Pedro Henrique Gomes[1,*], Thomas Watteyne[2], Bhaskar Krishnamachari[1]

[1] University of Southern California, Los Angeles, USA. {pdasilva,bkrishna}@usc.edu
[2] Inria, Paris, France. thomas.watteyne@inria.fr

ABSTRACT

Emerging Industrial IoT applications, such as smart factories, require reliable communication and robustness against interference from co-located wireless systems. To address these challenges, frequency hopping spread spectrum (FHSS) has been used by different protocols, including IEEE802.15.4-2015 TSCH. FHSS can be improved with the aid of blacklists to avoid bad frequencies. The quality of channels in most environments shows significant spatial-temporal variation, which limits the effectiveness of simple blacklisting schemes. In this article, we propose an enhanced blacklisting solution to improve the TSCH protocol. The proposed algorithms work in a distributed fashion, where each pair of receiver/transmitter nodes negotiates a local blacklist, based on the estimation of packet delivery ratio. We model the channel quality estimation as a multi-armed bandit problem and show that it is possible to create blacklists that provide results close to optimal without any separate learning phase. The proposed algorithms are implemented in OpenWSN and evaluated through simulations in two different scenarios with about 40 motes, and experiments using an indoor testbed with 40 TelosB motes. Copyright © 2017 John Wiley & Sons, Ltd.

*Correspondence
3740 McClintock Avenue, EEB 300, University of Southern California, Los Angeles, CA 90089, USA.

1. INTRODUCTION

The Industrial Internet of Things (IIoT) is emerging as one of the biggest drivers for productivity growth in the next decade [1]. Many IIoT-based systems require wireless communication, which causes serious reliability problems due to the non-deterministic nature of wireless links. In the 2.4 GHz band – used by the IEEE802.15.4 standard – ensuring reliability is a major challenge because of the proliferation of networks that use this frequency band. Techniques that exploit both *space diversity* and *frequency diversity* are being employed by new protocols such as IEEE802.15.4-2015 TSCH [2] to improve reliability and throughput. These diversity techniques exploit the fact that external interference and fading are different across frequencies, and also vary in space and time.

Timeslotted Channel Hopping (TSCH) slices time into slots and employs multiple frequencies. In addition, it adopts Frequency Hopping Spread Spectrum (FHSS) to smooth the impact of multi-path fading and external interference. Frequency Hopping is a technique that requires network synchronization and changes the communication frequency at every time slot. Each time slot in a TSCH network has an associated *channel offset* that is converted into a *frequency* by means of a pseudo-random hopping function. The frequency that is used might be any one of the 16 available in the 2.4 GHz band defined in the IEEE802.15.4 standard.

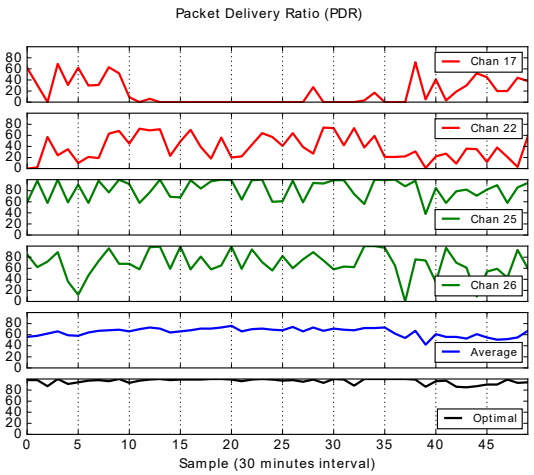

Figure 1. Packet delivery ratio of a particular link over a 24-hour period. The 2 top plots show the 2 worst channels; middle, the 2 best; and bottom, the average PDR over all 16 channels and the optimal PDR when selecting the best channel at every moment.

Prepared using ettauth.cls [Version: 2012/06/19 v2.10]

If a simple *blind hopping* function is used, all the frequencies are uniformly selected and transmissions suffer the average level of interference. In any network, link quality is coherent in the short term [3], which means that links with a poor quality are likely to remain in a bad state for a certain period, called "coherence time". Experiments in industrial environments show that coherence time can last from hundreds of milliseconds to seconds [4]. It is therefore preferable to avoid frequencies in which the last transmissions failed. This fact encourages the idea of *blacklisting* frequencies, and performing *selective hopping* to improve the performance of FHSS [5]. When blacklisting is employed, the frequencies with bad quality are temporarily excluded from the hopping list. The exclusion of a particular frequency should persist as long as the link quality at that frequency is poor, which requires an efficient method to regularly estimate link quality.

Although frequency blacklisting has already been used by other technologies, it is difficult to achieve an optimal implementation. Building a blacklist centrally is not trivial, since the quality of each frequency on all links need to be collected and combined by a central agent. Moreover, it is not effective, because link qualities are spatial-dependent, which means that a frequency that is bad for a particular link can be in a good state for others. Distributed blacklisting is more effective, but requires coordination and may even increase interference in networks that employ simultaneous transmissions.

In this article, we introduce the **M**ulti-hop **A**nd **B**lacklist-based **O**ptimized TSCH protocol (**MABO-TSCH**). *Our proposal employs a distributed blacklist for improving the performance of multi-hop wireless networks that have to cope with high levels of external interference and multi-path fading*.

In MABO-TSCH, the hopping sequence is locally built with information exchanged between each pair of communicating nodes. In addition, the hopping pattern that must be used in each link is optimally chosen so that, regardless of the neighbor's blacklists, two interfering links never use the same frequency. In this way, interference between neighboring links is avoided, and optimal TSCH schedules – with simultaneous transmissions at different links – can be executed. The challenging task of channel quality estimation is solved through Multi-Armed Bandit (MAB) optimization.

The main contributions of the article are threefold:

(i) *proposal of a solution for distributed blacklisting that is optimized for multi-hop networks and compliant with the IEEE802.15.4 TSCH standard*;

(ii) *a channel quality estimation algorithm based on MAB optimization*;

(iii) *implementation and empirical evaluation of MABO-TSCH protocol, both in simulation and on a 40-node testbed.*

The remainder of the article is organized as follows. Section 2 outlines the motivation and background behind our proposal. Section 3 lists the related literature that exploits blacklisting techniques and MAB-based spectrum sensing. Section 4 introduces MABO-TSCH, the proposal of this article. Section 5 evaluates the performance of the proposal through simulations considering two different scenarios; the first with dataset from Tutornet testbed with 40 nodes, and the second with dataset from Soda testbed with 46 nodes. Section 6 evaluates its performance experimentally on a 40-mote indoor testbed. Section 7 summarizes the lessons learned from the simulation and testbed results. Finally, Section. 8 concludes this article.

2. MOTIVATION AND BACKGROUND

The recent IEEE802.15.4-2015 TSCH [2] standard is built for critical low-power wireless applications. Several studies have demonstrated limitations of legacy IEEE802.15.4 because of its single-channel operation, which makes it susceptible to external narrow-band interference and multi-path fading [6, 7]. In real deployments, even on frequencies with no other technology present (such as channels 25 and 26*, which typically do not overlap with WiFi) the received signal strength can vary by tens of dB over time.

To illustrate our motivation, we measure the connectivity between 40 sensor nodes in an indoor testbed. In this experiment, every node broadcasts 100 packets with a length of 100 bytes. When one particular node is sending packets, all others listen and record whether the packets are received, and their corresponding RSSI. This sequence is repeated for all 16 channels. Every 30 min, the experiment is re-run. We execute 48 rounds, totaling a 24 h period. Fig. 1 shows a very small subset of that data. It shows the PDR (Packet Delivery Ratio) between two particular nodes located approximately 15 m apart. The 4 upper plots show the PDR of the 2 worst and 2 best channels. The bottom 2 plots show the average PDR over all 16 available channels, and the highest PDR across all frequencies.

Channels 17 and 22 have a low PDR and variable quality, resulting in connectivity problems at different times during the 24 h period. Even the best channels (25 and 26) undergo considerable degradation at certain times; the PDR goes as low as 20%. When using blind FHSS (hopping over all 16 channels), the PDR perceived by a network is equivalent to the average PDR over all channels. The "Average" plot shows that blind FHSS is sufficient to avoid connection failures. This is what standards such as WirelessHART, ISA100.11a or IEEE802.15.4 TSCH use.

We want to go one step further. The bottom-most plot in Fig. 1 shows that, in the ideal case where the frequency with the highest PDR is used for each packet, the PDR of the link jumps from approximately 60% to approximately 90%. The hard (impossible) part is picking the best frequency all the time, as there is no way for

*In this article the terms *frequency* and *channel* are interchangeably used when referring to the spectrum portion used for communication.

Trans. Emerging Tel. Tech. 2017; **00**:1–15 © 2017 John Wiley & Sons, Ltd.
DOI: 10.1002/ett
Prepared using ettauth.cls

nodes to know which frequency is best with infinite and instantaneous knowledge.

The middle-ground we propose to explore is to learn the subset of best frequencies, hop over them and blacklist the others. Blacklisting can be divided into two distinct phases. First, a pair of nodes – or a central coordinator – decides when to include a given channel in the blacklist. Since the quality of the channels varies over time, in a second phase, each channel in the blacklist is re-evaluated, and possibly removed from the blacklist. In this decision-making process, a channel is added to the blacklist when its quality falls below a certain threshold, and removed when its quality is above a certain level.

An accurate *link quality estimator (LQE)* must be implemented to determine the quality of a channel. The authors in [8] provide a comprehensive survey of available LQEs. Hardware-based LQEs only rely on the information available from the radio chip, such as RSSI, LQI, and SNR. Even though this information is part of the IEEE802.15.4 standard, and even though hardware-based LQEs do not require additional computation, their degree of accuracy is limited because they rely on parameters that need to be fine-tuned. Software-based LQEs require additional computation but achieve a better degree of accuracy and stability. Most software-based LQEs leverage data provided by hardware, such as RSSI and LQI, and improve the estimates by employing different processing methods.

The state-of-the-art proposals for enhancing FHSS rely on hardware-based link quality estimators that require fine-tuning parameters and may become inefficient in networks with a high variability of the link quality. The use of blacklists without a good link quality estimator can cause a deterioration in the network performance, since channels with good quality may be accidentally blacklisted.

We propose a link quality estimation based on the Multi-Armed Bandit problem. We employ an approximate solution based on ϵ-greedy strategy [9], and analyze how the algorithm adapts to dynamic scenarios. This stochastic optimization solution depends on neither hardware parameters nor sophisticated computation.

2.1. IEEE802.15.4 and Frequency Hopping

The TSCH operation is based on network synchronization and individual schedules that are followed by each node. The time in the TSCH network is sliced into slots, each of which of a sufficient duration (typically 10 ms) to accommodate a data packet of maximum size and an acknowledgment packet (ACK), as well as all the required guard times. It employs Time Division Multiple Access (TDMA) with multiple frequencies. TSCH networks allow more than one transmission to occur at the same time; as long as simultaneous transmissions are on different frequencies, collision-free operation is guaranteed.

In TSCH, every time slot is uniquely identified by its Absolute Sequence Number (ASN), a counter that increments at each time slot. Every node is aware of the current ASN in the network. The schedule consists

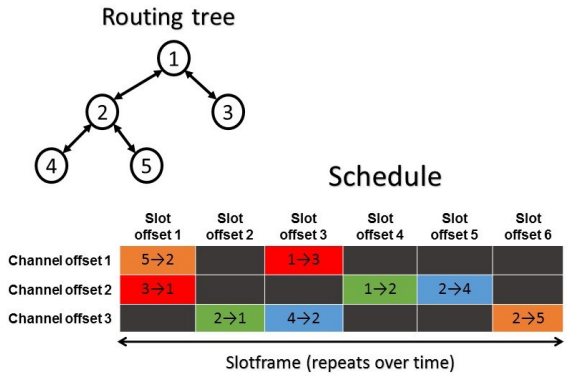

Figure 2. Example of a TSCH schedule with 6 time slots and 3 channel offsets.

of a sequence of atomic resource units (time-frequency allocations) that repeat over time. The group of atomic resource units (called *cells*) is denoted as a *slotframe*. Each cell can be shared (contention-based access is employed using CSMA/CA), or dedicated (contention-free access is guaranteed for a pair of nodes). Fig. 2 shows an example of network and its TSCH schedule. Each link has an associated cell, and each cell is uniquely identified by a tuple *(slot offset, channel offset)*. The slot offset sets the location of the cell in time from the beginning of the current slotframe; the channel offset is a "virtual channel" that is translated into an actual frequency that is going to be used. The translation is performed by the FHSS algorithm, that follows a pseudo-random pattern and spreads the packets across the 16 frequencies allocated in the 2.4 GHz band.

Even though the TSCH standard specifies how the hopping scheme should be implemented, the actual list of frequencies to be used and the hopping sequence is left out of scope. The standard states that the number of allowed hopping sequences is arbitrary and each sequence can cover either all or a subset of the available frequencies. In addition, the sequence of channels can be static or dynamically determined by an algorithm.

3. RELATED WORK

The authors in [5] conducted one of the first works that demonstrate that frequency hopping improves the reliability over single-channel operation. In their experiment, blind frequency hopping reduced the average ETX by 56% and reduced the network churn (changing parent in the routing tree) by 38%. The authors evaluate different sizes of blacklists and find that allowing the nodes to use the best 6 channels (and blacklist the other 10) was the best solution for their scenario. Their analysis was conducted with trace-based simulations.

A few recent works have proposed TSCH enhancements to improve FHSS. The authors in [10] propose an

Adaptive TSCH (ATSCH) protocol, which is a distributed blacklisting solution. It introduces two new features: (i) a new type of time slot used for Noise Floor (NF) estimation, and (ii) the blacklist information that is sent in every Enhanced Beacon (EB). The quality of channels is periodically estimated through RSSI measurements during NF time slots; a blacklist is locally calculated by each node and disseminated inside EBs. Results show that a blacklist of size 6 can improve the average ETX by 8.1% when compared with blind hopping. It also shows that blacklisting increases the average PDR of the whole network *and* reduces its dispersion, i.e. the links become stronger and more stable. The solution was implemented in OpenWSN and tested in a real testbed with WiFi and Bluetooth interference. Even though, to the best of our knowledge, the proposal is the first realization of an adaptive FHSS algorithm for TSCH networks, it is not fully standards-compliant. The negotiation method piggybacks the blacklist into EB, which is not efficient, since EBs are expected to be exchanged every tens of seconds, thus incurring a long delay for updating the blacklist. The solution also does not guarantee that neighbor nodes use the same blacklist, since EBs are broadcast packets and may not be received by all the neighbors.

As an improvement to the ATSCH solution, the authors of [11] introduce an Enhanced TSCH (ETSCH) variant, which has two main components: (i) a non-intrusive channel-quality estimation (NICE) procedure that measures energy during periods of silence in every time slot, and (ii) an enhanced beacon hopping sequence list (EBSL) that only makes use of the strongest channel for broadcasting EBs and, thus, improves the reliability of blacklist distribution. Results show that ETSCH provides a 24% higher PRR and 50% shorter length of burst packet losses, in scenarios with high level of interference, when compared to ATSCH. Even though the proposed link estimation procedure outperforms ATSCH, it is only executed at the sink and requires a maximum acceptable clock drift. It has not been evaluated, and does not seem to be feasible, in multi-hop scenarios where there are large clock drifts between the sink and leaf nodes. Using a smaller subset of stronger channels for broadcast EBs is an improvement for blacklist distribution, but still does not guarantee that all the nodes use the same blacklist, since there are no ACK packets for beacon transmissions. Finally, both ATSCH and ETSCH fail to take into account cases where simultaneous transmissions are scheduled in a multi-hop network, where the blacklist may cause internal interference.

The multi-armed bandit (MAB) problem is a classical paradigm in stochastic optimization where an automated agent seeks to maximize the total payoff obtained after a sequence of trials. In MAB problems, the agents have to choose a strategy that provides the best trade-off between exploring the unknown environment and exploiting current knowledge. We refer the reader to [9] for an overview of the MAB technique and examples of practical applications. Although MAB has been explored in theory for channel allocation problems such as for opportunistic spectrum access [12, 13, 14], there is few prior work demonstrating a practical application of MAB to communication systems.

It is clear from the state-of-the-art that there are still three open problems:

(i) *how to design an optimized hopping sequence that prevents interference between nodes that have been scheduled for simultaneous transmissions*;

(ii) *how to create a distributed blacklist and ensure that all the neighbors use the same*;

(iii) *how to implement a feasible channel estimation mechanism that does not depend on hardware resources and is adaptable to dynamic networks.*

The goal of our MABO-TSCH proposal is to solve these three open problems.

4. MABO-TSCH

Multi-Hop **A**nd **B**lacklist-based **O**ptimized TSCH protocol (**MABO-TSCH**) consists of three key algorithms.

The first algorithm (Sec. 4.1) assigns channel offsets to time slots to prevent interference. The adopted solution is based on a graph coloring heuristic that associates multiple orthogonal channel offsets to each non-leaf node, and allows the use of different frequencies in each time slot.

The second algorithm (Sec. 4.2) ensures proper blacklist negotiation between the nodes. Each pair of nodes (parent-child in the routing tree) negotiates a local blacklist by piggybacking blacklist information into the data or ACK frames.

The third algorithm (Sec. 4.3) measures and classifies the channels, and is responsible for building and maintaining the blacklists. The channel classification process is modeled as a multi-armed bandit problem with an approximate solution based on ϵ-greedy strategy.

4.1. Channel Offset Assignment

The channel offset assignment must be executed from the moment different pairs of neighbor nodes communicate in the same time slot offset (i.e. at the same time) to avoid intra-network interference. The channel offset allocation can be of three types: link-based, receiver-based or transmitter-based. In the link-based assignment, each active link is associated with a channel offset. In the receiver and transmitter-based types, the channel offset is assigned to the receiver or transmitter nodes, respectively, and all the time slots must be executed with the channel offset associated with the participating nodes.

In data collection applications, most unicast transmissions are directed towards the sink and most of the routing trees have a large number of leaf nodes. Hence, link-based and receiver-based channel offset assignments might be

Trans. Emerging Tel. Tech. 2017; **00**:1–15 © 2017 John Wiley & Sons, Ltd.
DOI: 10.1002/ett
Prepared using ettauth.cls

more appropriate for multi-hop tree-based data collection applications. Thus, MABO-TSCH uses a receiver-based channel offset assignment

The channel offset assignment is a graph coloring problem, which is known to be NP-hard. However, heuristics such as greedy degree-ordering known as Welsh-Powell [15] yield near-optimal results in most practical cases.

In our proposed algorithm, we extend the Welsh-Powell heuristic and associate multiple non-interfering channels to each node. All the nodes are sorted in non-increasing order according to their degrees in a graph that is constructed with nodes as vertices and interfering links as edges. The coloring problem is solved for the sorted array of nodes (from the highest to the lowest degree), considering all 16 available channel offsets as colors. After all the nodes are colored, the algorithm repeats in order to find multiple channel offsets for each node. The channel offset assignment is completed when no more colors can be assigned to any node.

Alg. 1 shows the algorithm for multiple channel offset assignments. It is centrally executed and its results are used by a Path Computation Element (PCE), that is responsible for computing and disseminating the TSCH schedule. The network graph should be obtained previously from the network with any simple data collection application that is able to gather PDR statistics. If it is built considering the time schedule it can disregard the edges between nodes that do not have time slots allocated at the same time, which will increase the number of assigned channel offsets.

Even though PDR statistics change significantly over time (as seen in Fig. 1), the channel assignment algorithm does not require the most updated statistics to work properly. If this algorithm is not executed regularly, the outcome of blacklisting algorithms may become degraded over time if optimal schedules are employed (with minimum number of times slots). However, if non-optimal schedules are used, less intra-network interference happens, and the coloring algorithm plays a less important role. In the extreme case of event-triggered application, where there is only sporadic data traffic and no simultaneous time slots are scheduled for two different nodes, all channel offsets can be associated to all nodes. In this case, Alg. 1 is not necessary, since there is no conflict of transmissions within the network to be solved.

Algorithm 1 Channel offset assignment

Input:
$G(V, E)$ - network graph with nodes as vertices and interfering links as edges
C - list of 16 channel offsets
Output:
Nodes colored with multiple channel offsets

```
1: Sort vertices v_1, v_2, ..., v_n in V in non-increasing degree order
2: colored ← true
3: while colored is true do
4:     colored ← false
5:     for all v_i in V do
6:         find c_i as the minimal color in C not assigned to any vertex v_j connected to v_i
7:         if c_i exists then
8:             colored ← true
9:             Add c_i to the list of channel offsets of v_i
10:        end if
11:    end for
12: end while
```

4.2. Distributed Blacklist Negotiation

The blacklist negotiation process must ensure that each pair of nodes uses the same blacklist. Were they to use a different blacklist, neighbor nodes would have their radio turned on at different frequencies, and not hear one another. Moreover, the information exchanged for blacklist negotiation should not incur a large overhead on the network.

The dissemination of blacklist information may be based on either broadcast [10, 11] or unicast messages. In the case of broadcast messages, there is no guarantee that the information is correctly exchanged between neighbors. Moreover, broadcast messages such as Enhanced Beacon (EB), tend to have transmission intervals much larger than the dynamics of the channels. Unicast messages provide the guarantee that the negotiation is successful, but may induce some overhead. When using unicast messages for node-to-node blacklist negotiation, the blacklist can be embedded in the data or ACK frame. When embedded in the data frame, the overhead incurred by blacklist information may affect the application performance, since it uses part of the application payload. When embedded in the ACK frame, no overhead is perceived by the application, since the time reserved for the ACK transmission is fixed and can accommodate a few extra bytes.

After the exchange of a new blacklist, both the transmitter and receiver must decide when to start using it. This is a key point in the blacklist negotiation process because if there is an information mismatch, a large number of packets might be lost. In the worst case scenario, if one of the involved nodes is a time synchronization source, part of the network may be disconnected.

Blacklist negotiation has been efficiently implemented in this work by adopting unicast messages and embedding the blacklist information into the ACK *or* Data frames, depending on the type of application that is employed. In addition, we use bidirectional negotiation: parent nodes are responsible for creating and disseminating blacklists to their children if ACK-based negotiation is used; the reverse occurs for Data-based negotiation. Blacklist information is only used in unicast communication; transmissions that use shared time slots (such as EBs) do not employ blacklisting.

ACK-based negotiation is used when constant traffic flows from children to parent, since the measurement is

performed by the parent node. The main advantage of such approach is that no overhead is perceived by the application.

Data-based negotiation is used when an event-triggered application (e.g. alarm systems) is the main focus. Since parents cannot predict when packets will arrive, the link quality estimation has to be executed by the children. In this case, a few bytes of data overhead are perceived by the application, since blacklist information has to be embedded into the data packets.

We mainly focus this article on data collection applications that employs ACK-based negotiation. Alg. 2 shows the pseudo-code executed at the parent. Alg. 3 shows the pseudo-code executed at the children. We evaluate Alg. 2 and Alg. 3 through simulations and real experiments.

We also propose similar algorithms for the Data-based negotiation, which is more appropriate for an event-triggered application. Alg. 4 shows the pseudo-code executed at the children. Alg. 5 shows the pseudo-code executed at the parent. We evaluate Alg. 4 and Alg. 5 exclusively through simulations.

In all 4 algorithms, both nodes keep a table with two rows: r_{using} and $r_{negotiating}$. Both rows have the Data Sequence Number (DSN) of the last packet and the blacklist information. Row r_{using} has the most recent negotiated blacklist and must be used at the beginning of each time slot. Row $r_{negotiating}$ has the blacklist information that is currently being negotiated, and has not yet been used.

Algorithm 2 Blacklist embedded in ACK frame (algorithm for the parent)

1: At the beginning of time slot, consider the blacklist information in r_{using}
2: **if** data frame was successfully received **then**
3: $FS \leftarrow$ DSN of received data frame
4: $BL \leftarrow$ Most recent local *blacklist information*
5: **if** $r_{negotiating}$ has DSN equal to FS **then**
6: **if** maximum number of retransmissions is reached **then**
7: $BL \leftarrow$ *blacklist information* from r_{using}
8: **end if**
9: Update $r_{negotiating}$ with BL
10: **else**
11: Replace r_{using} by $r_{negotiating}$
12: Update $r_{negotiating}$ with FS and BL
13: **end if**
14: Send ACK frame with FS, embedding BL
15: **end if**

At the beginning of every time slot, nodes use the blacklist information in r_{using} and based on the success/failure of packet exchange, the $r_{negotiating}$ may replace r_{using} and nodes can start a new blacklist negotiation. It is important that r_{using} is only replaced when $r_{negotiating}$ is consistent on both sides.

Algorithm 3 Blacklist embedded in ACK frame (algorithm for the child)

1: At the beginning of time slot, consider the blacklist information in r_{using}
2: $FS \leftarrow$ DSN of data frame to be sent
3: Send data frame with FS
4: **if** $r_{negotiating}$ has DSN different than FS **then**
5: Replace r_{using} by $r_{negotiating}$
6: Update $r_{negotiating}$ with FS and *blacklist information* from r_{using}
7: **end if**
8: **if** ACK frame was successfully received **then**
9: $BL \leftarrow$ *blacklist information* from received ACK frame
10: Update $r_{negotiating}$ with BL
11: **end if**

Algorithm 4 Blacklist embedded in data frame (algorithm for the child)

1: At the beginning of time slot, consider the blacklist information in r_{using}
2: $FS \leftarrow$ DSN of data frame to be sent
3: **if** $r_{negotiating}$ has DSN different than FS **then**
4: Replace r_{using} by $r_{negotiating}$
5: Update $r_{negotiating}$ with FS and *blacklist information* from r_{using}
6: **end if**
7: **if** Maximum number of retransmissions is reached **then**
8: $MaxFlag \leftarrow 1$
9: $BL \leftarrow$ *Blacklist information* from $r_{negotiating}$
10: **else**
11: $BL \leftarrow$ Most recent local *blacklist information*
12: **end if**
13: Send data frame with FS, embedding BL and $MaxFlag$
14: **if** ACK frame was successfully received **then**
15: $MaxFlag \leftarrow 0$
16: Update $r_{negotiating}$ with BL
17: **end if**

The IEEE802.15.4 standard specifies that the DSN is incremented only after an ACK frame is received and every ACK frame has its DSN copied from the Data frame that it is acknowledging. Thus, the nodes can guarantee that both ends have the same information after a packet with a different DSN has been received on both sides.

The blacklist information that should be used when transmitting packets with DSN_{n+2} is the one exchanged DSN less than or equal to DSN_n. Every packet has a maximum number of retransmissions at the link layer, which by default in our implementation is equal to 3 (a maximum of 4 trials). The proposed algorithms are designed for networks with link-layer ACK and a maximum number of retransmissions greater or equal to 1.

Trans. Emerging Tel. Tech. 2017; **00**:1–15 © 2017 John Wiley & Sons, Ltd.
DOI: 10.1002/ett
Prepared using ettauth.cls

Algorithm 5 Blacklist embedded in data frame (algorithm for the parent)

1: At the beginning of time slot, consider the blacklist information in r_{using}
2: **if** Data frame was successfully received **then**
3: $FS \leftarrow$ DSN of received data frame
4: $BL \leftarrow$ *blacklist information* from received data frame
5: $MaxFlag \leftarrow$ flag from received data frame
6: **if** Row *negotiating* has DSN equal to FS **then**
7: Update $r_{negotiating}$ with BL
8: **else**
9: **if** $MaxFlag$ is 1 **then**
10: Copy *blacklist information* from r_{using} to $r_{negotiating}$
11: **end if**
12: Replace r_{using} by $r_{negotiating}$
13: Update $r_{negotiating}$ with FS and BL
14: **end if**
15: Send ACK frame with FS
16: **end if**

The *blacklist information* that is negotiated in all 4 algorithms may be different depending on how this information is used for estimating the channel quality and optimizing the hopping sequence. In Section 4.3, we discuss which types of blacklist information we use in our proposal. It should be pointed out that the framework proposed so far can be employed for any type of decision-making and any type of blacklist information can be used.

4.3. Multi-armed Bandit Link Estimation

A multi-armed bandit can be formulated as a set of K probability distributions $B = \{R_1, R_2, ..., R_K\}$, each associated with the rewards delivered by one of the K arms (or levers). The distribution probabilities have expected reward value $\mu_1, \mu_2, ..., \mu_K$ and are *a priori* unknown to the player.

In an MAB problem, at each turn $t = \{1, 2, 3, ...\}$, an arm with index $i(t)$ is chosen and the player receives the reward $r(t) \sim R_{i(t)}$. Let $\mu^* = max_{i=1,2,...,k}\ \mu_i$, we define the total regret for a sequence of trials with duration T as:

$$R_T = T\mu^* - \sum_{t=1}^{T} r(t) \quad (1)$$

We can think of regret as the difference between the chosen strategy and an optimal strategy which always chooses the best arm. A common formulation of the MAB problem is the Bernoulli multi-armed bandit, where a reward of x is obtained with probability p and otherwise a reward of 0. Related work shows that simple approximate heuristics, such as ϵ-greedy algorithm, achieve results close to or better than sophisticated algorithms in most settings [9, 16] of MAB problems.

Our problem consists of estimating the quality of each of the 16 channels to ensure that the best ones can be employed in the blacklisting mechanism. In our implementation, the channel estimation problem is modeled as multi-armed bandit problem with Bernoulli reward equals to 100 for successes and 0 for failures. Each node is an autonomous agent with 16 arms corresponding to the 16 available channels.

We choose ϵ-greedy as the strategy for implementing our MAB-based channel quality estimation because the algorithm is tractable enough to be embedded in the sensor nodes. During each trial, the bandit selects the arm (channel) that has the highest mean reward with probability $1 - \epsilon$, and selects a random arm with probability ϵ. We define $\hat{\mu}_i(t)$ as the empirical mean reward of arm i after t trials. The average empirical reward for each channel ($\hat{\mu}_{ch}(t)$) is updated with exponential moving average so that the most recent reward values have more significance in the average reward calculation.

The MAB algorithm must be executed at one node for each pair child/parent, and the blacklist information must be embedded either in the data frames, or ACK frames, as described in Section 4.2 If the blacklist information is embedded in data frames, the MAB algorithm is executed at the child node, while if the blacklist information is embedded in ACK frames, it must be executed at the parent node. On the basis of the current $\hat{\mu}_{ch}(t)$ obtained by the ϵ-greedy algorithm, the node where the MAB algorithm is being executed has to create the most accurate channel quality estimation and create a *blacklist information*, which will then be sent to the neighbor node.

We propose two different types of *blacklist information*: a simple 2-byte blacklist bitmap, and an 8-byte rank list. In the case of the blacklist bitmap, the k channels with the highest average reward are not included in the blacklist (the corresponding bits are equal to 0), while $16 - k$ channels with the lowest values are included in the blacklist.

In the case of the rank list, all the 16 frequencies are sorted in non-decreasing order according to their current $\hat{\mu}_{ch}(t)$, and the rank of the channel must be equal to its position in the sorted array.

The overhead of both types of *blacklist information* are restricted to a small increase in energy consumption if ACK-based negotiation is used (with data collection application), since the *blacklist information* fits into the interval reserved for the ACK transmission within the time slots. But it may represent at least 10% of the data payload (plus the extra energy consumption) of common 6LoWPAN data frames if Data-based negotiation is employed. However, the event-triggered applications, such as alarm systems, usually do not require large payload and may easily support this overhead of up to 8 bytes.

We propose two different algorithms for employing each type of *blacklist information* and implementing the optimized frequency hopping. Alg. 6, called *First Good Arm MABO-TSCH*, uses the simple 2-byte blacklist

bitmap. Alg. 7, called *Best Arm MABO-TSCH*, uses the 8-byte rank list.

Algorithm 6 *First Good Arm MABO-TSCH* frequency selection

Input:
BL - 2-byte bitmapped blacklist
CL - list of available channels offsets
Output:
Frequency to be used

1: **for** all c_i in CL **do**
2: $freq \leftarrow c_i$ converted into actual frequency
3: **if** bit corresponding to $freq$ in BL is 0 **then**
4: return $freq$
5: **end if**
6: **end for**
7: return $freq$

Algorithm 7 *Best Arm MABO-TSCH* frequency selection

Input:
RL - 8-byte rank list
CL - list of available channels offsets
Output:
Actual frequency to be used

1: **for** all c_i in CL **do**
2: $freq \leftarrow c_i$ converted into actual frequency
3: $frequencies \leftarrow frequencies \bigcup freq$
4: **end for**
5: Sort $frequencies$ in non-decreasing order of rank according to RL
6: return $freq$ with highest rank in $frequencies$

In Alg. 6, each of the available *channel offsets* is translated into an actual frequency and the first frequency that is not blacklisted is used. In Alg. 7 all available *channel offsets* are translated into their corresponding frequencies and the frequency with the highest rank is used.

5. SIMULATION RESULTS

We first evaluate the performance of MABO-TSCH by simulation to assess its effectiveness, evaluate the impact of the different parameters, and compare its performance with an optimal solution.

We use a custom-made simulator written in C. It receives a set of connectivity traces as input, and calculates the routing tree and the optimized TSCH schedule[†]. On basis of the calculated TSCH schedule, and considering the set of time-sparse connectivity traces, the simulator runs the appropriate FHSS algorithm to calculate network statistics such as the number of received packets, packet drop rate, etc. The connectivity traces used consist of 16 PDR matrices ($PDR_{i,j}^{ch}$ is the PDR of link $i \rightarrow j$ at channel ch).

We consider the dataset obtained from two different deployments. The first dataset consists of 5 different traces, each with 32 connectivity snapshots of a 40-node network from the Tutornet testbed[‡], using TelosB motes. The snapshots are obtained every 15 min, for a total of 8 h for each trace, resulting in a 40-hour simulation (5 traces of 8 hours each). The transmission power is set to -15 dBm to obtain PDR statistics that lead to routes with more than 2 hops. We name this dataset as "Tutornet" throughout the paper.

The second dataset consists of one single trace with 17 connectivity snapshots from a 46-node network deployed in a UC Berkeley office space (50m x 50m)[§]. The nodes are also TelosB motes. Each connectivity snapshot was obtained at different times of the day with several hours separating them. To compare the results with the first scenario (using Tutornet), we scaled the time of this second dataset and considered that it was obtained continuously with intervals of 15 min, similarly to the first set of traces, which resulted in a 4.25-hour simulation. The power used in the second dataset was 0 dBm. We name this dataset as "Soda" throughout the paper.

We only have access to the testbed from which "Tutornet" dataset was obtained, thus we use it for both simulation and real implementation. "Soda" dataset is only used for simulation. The simulation with "Soda" dataset is mainly considered to confirm that the obtained results can be generalized to multiple environments.

The simulator builds a fixed routing tree and a fixed TSCH-compatible schedule considering the algorithms proposed by the MultiChannel Collection (MCC) protocol [17]. The only modification made in MCC is the replacement of its graph coloring heuristic by Alg. 1, in order to associate the time slots with multiple channel offsets. The modified MCC creates a TSCH schedule with multiple non-interfering channel offsets for each time slot, as required for MABO-TSCH.

5.1. Different Types of FHSS

We simulate 5 different types of FHSS to evaluate the effectiveness of MABO-TSCH. A TSCH variant is defined for each type of FHSS. The only difference between all 5 variants is the algorithm used for translating channel offsets into frequencies – in order words, the frequency hopping sequence.

1. ***Default TSCH***

[†] The source code of the simulator is available at `http://anrg.usc.edu/www/downloads/`.

[‡] The first dataset is available at `http://anrg.usc.edu/www/tutornet/`.

[§] The second dataset is available at http://wsn.berkeley.edu/connectivity/. The dataset used is "Soda".

Trans. Emerging Tel. Tech. 2017; **00**:1–15 © 2017 John Wiley & Sons, Ltd.
DOI: 10.1002/ett
Prepared using ettauth.cls

In *Default TSCH* the channel translation is carried out with a simple function that takes into account the channel offset and the current ASN, and does not employ any blacklisting technique. It follows (2):

$$freq = table[(channel\_offset + ASN)\%16] \quad (2)$$

where $table$ is a look-up table with all 16 frequencies randomly ordered.

2. ***Centrally Blacklisted TSCH***

A central agent creates a list with N channels that have more links with PDR below a given threshold, and then disseminates this information to all the nodes in the network. In the simulation, this blacklist is built by means of the link quality traces and the threshold set to 90%. The results obtained from the simulation represent an optimal centralized solution where the sink has a complete knowledge of all link qualities and can disseminate the best global blacklist to all the sensor nodes. This is not a realistic solution, since it is not possible to keep a real-time perfect link estimation at a centralized agent. All the nodes are supposed to translate the available channel offsets in every time slot using (2), and use the first frequency that is not in the blacklist.

3. ***Optimal TSCH***

It is assumed that all the nodes have knowledge of the channel quality for all the links and are able to pick the channel with the highest PDR. This solution can be thought as a distributed blacklist where each pair of nodes selects the best channel for each packet transmission, among all channels that are allowed to be used. In this ideal scenario, all nodes are supposed to translate *all* the available channel offsets in each time slot using (2) and pick the frequency with the highest PDR. This algorithm is an optimal distributed solution and would achieve the highest possible performance in the network.

4. ***First Good Arm MABO-TSCH***

The *First Good Arm MABO-TSCH* employs a multi-armed bandit problem using ϵ-greedy strategy, as described in Section 4.3. In selecting the first best arm, the parent constructs a simple 2-byte blacklist bitmap for each of its children. The blacklist is then shared with the children using the negotiation mechanism presented in Section 4.2. *First Good Arm MABO-TSCH* uses Alg. 6 for selecting the frequency that will be used in each time slot. The conversion from channel offsets to frequencies follows (2).

5. ***Best Arm MABO-TSCH***

The *Best Arm MABO-TSCH* uses an 8-byte ranking list to select the best channel, as estimated by the ϵ-greedy strategy. The 8-byte ranked list is also created for each child and negotiated with the algorithms from Section 4.2. It uses Alg. 7 and the channel offset conversion also follows (2).

It is important to note that both **First Good Arm MABO-TSCH** and **Best Arm MABO-TSCH** employ the whole framework proposed in this work, which consists of the three algorithms described in Section 4. *First Good Arm MABO-TSCH* uses a smaller *blacklist information* and, consequently requires less energy and overhead, but should obtain average performance. On the other hand, *Best Arm MABO-TSCH* uses a larger *blacklist information*, which incurs more energy consumption, but should be able to improve the performance even further. The objective of both proposals is simply to compare the trade-offs of them; obviously, the best solution that should be used, if possible, is *Best Arm MABO-TSCH*.

5.2. Simulation Setup

The simulations are executed considering the two different datasets, as described above. The first ("Tutornet") uses 32 connectivity snapshots (8-hour duration) from a 40-node network. And the second ("Soda") uses 17 connectivity snapshots (4.25-hour duration) from a 46-node network.

Simulations are repeated with different traces for "Tutornet" or with different seeds for "Soda". Results are presented with 95% confidence intervals. The schedule used in the simulation has a slotframe with 101 time slots, each time slot taking 15 ms. In each slotframe, every node has one reserved time slot to send one packet upwards to the sink node, and a sufficient number of reserved time slots to forward packets from their children.

5.3. Tuning the Algorithms

Before comparing the 5 different types of TSCH, we find their best parameters to obtain the best results. Simulations are divided into two groups. We first evaluate data collection applications with blacklist information embedded into ACK frames. We then evaluate event-triggered applications with blacklist information embedded into Data frames.

5.3.1. Evaluating Data Collection Application

We set out by examining the case of *data collection* application, where all sensor nodes transmit one packet toward the sink in every slotframe. In this case, since all the time slots are used for data transmission, the parent nodes can build and maintain the blacklist using ACK packets, following Alg. 2 and 3. It is important to note that even though we simulate a saturated network (with packets transmitted at every time slot), this is not a requirement for the proposed algorithms. The only requirement for Alg. 2 and 3 is that the parent nodes are able to predict which time slot will be used by children nodes to transmit data, since parents have to differentiate between an empty time slot and a time slot where there was a packet loss.

The overhead for data collection is minimal, just requiring the extra energy used by blacklist information transmission. The main purpose of this application is to maximize the network throughput (i.e. the total the number of received packets at the sink node over time).

All 5 types of TSCH are simulated to tune the optimal parameters in the algorithms. Initially, we consider *Centrally blacklisted TSCH* and vary the number of channels in the common blacklist (N). *The optimal value of N was equal to* 11 *for "Tutornet" and* 12 *for "Soda"*. This is similar to the results obtained in [5], where the optimal blacklist was found to be equal to 10.

We now aim to find the best parameters for ϵ-greedy strategy. We first consider *Best Arm MABO-TSCH* and vary ϵ. We initially considered fixed ϵ. *The optimal value of ϵ was found to be* 0.05 *for "Tutornet" and* 0.025 *for "Soda"*.

Following this, we consider the *First Good Arm MABO-TSCH* solution and vary both ϵ and k parameters. In this algorithm, the channels are sorted according to their empirical reward, and the k channels with the highest rewards may be employed in the hopping sequence. We simulate *First Good Arm MABO-TSCH* with fixed ϵ varying from 0.5 to 0.01 and k varying from 1 to 16. *The best ϵ was found to be* 0.03 *and the best k was* 6*, for "Tutornet". Finally, the best ϵ was equal to* 0.02 *and the best k was equal to* 5*, for "Soda"*.

We also investigate whether ϵ-greedy strategy with decreasing ϵ can achieve better performance. For both types of MABO-TSCH solutions (*First Good Arm MABO-TSCH* and *Best Arm MABO-TSCH*) we set ϵ to 0.05 and periodically reduced it by 0.001. The minimum value of ϵ is 0.01; after reaching this value, ϵ is reset to 0.05. In our simulations, we verify that a decreasing ϵ does not significantly affect the number of received packets at the sink and the cumulative regret was also very close. It can be concluded that, similar to what was verified by [16], a decreasing ϵ does not significantly improve the performance of MAB-based algorithms.

Now, we compare all 5 types of TSCH solutions: *Default TSCH*, *Centrally blacklisted TSCH*, *Optimal TSCH*, *First Good Arm MABO-TSCH*. *Best Arm MABO-TSCH*. For each, we use the best values of N, k and ϵ, as found previously.

Initially we focus on the results from "Tutornet". Fig. 3 shows the total number of packets received at the sink for all 5 different types of TSCH. Fig. 4 shows the average regret per time slot. The average regret is calculated as the total regret R_T (Eq. (1)) divided by the number of time slots. Fig. 5 shows the percentage of optimal channels used. For this last statistic, we compare the channel chosen by all non-optimal TSCH solutions in each time slot with the channel chosen by *Optimal TSCH*.

Fig. 3 shows that both types of MAB-based solutions (First Good Arm MABO-TSCH and Best Arm MABO-TSCH) outperform *Default TSCH* and *Centrally blacklisted TSCH*. *The average number of received packet is 43% higher when Best Arm MABO-TSCH is compared with Default TSCH. The performance of Best Arm MABO-TSCH is less than 10% lower than Optimal TSCH.*

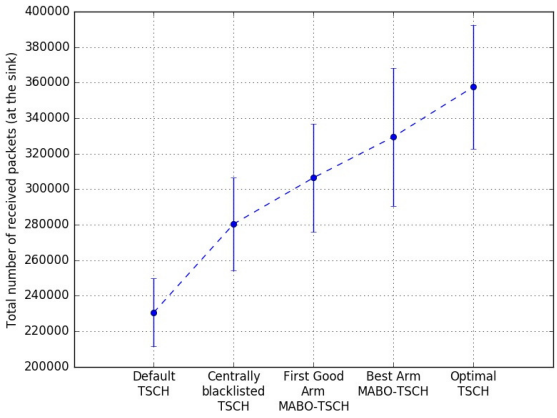

Figure 3. Total number of received packets at the sink (Tutornet connectivity trace).

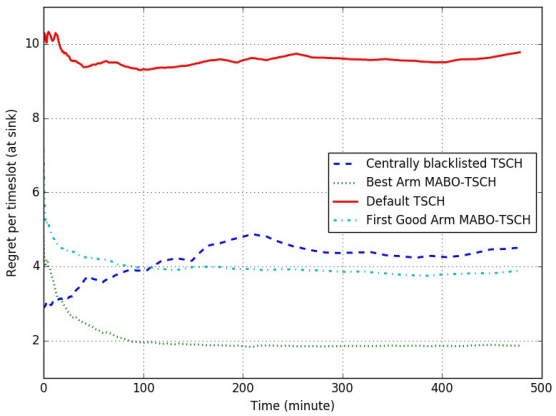

Figure 4. Average regret per time slot (Tutornet connectivity trace).

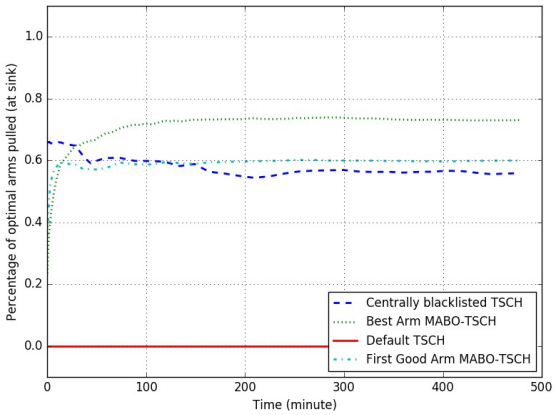

Figure 5. Percentage of optimal channels utilized (Tutornet connectivity trace).

Fig. 4 shows that *Best Arm MABO-TSCH* has the least regret, and that the average regret per time slot converges more quickly for this type of TSCH.

Trans. Emerging Tel. Tech. 2017; **00**:1–15 © 2017 John Wiley & Sons, Ltd.
DOI: 10.1002/ett
Prepared using ettauth.cls

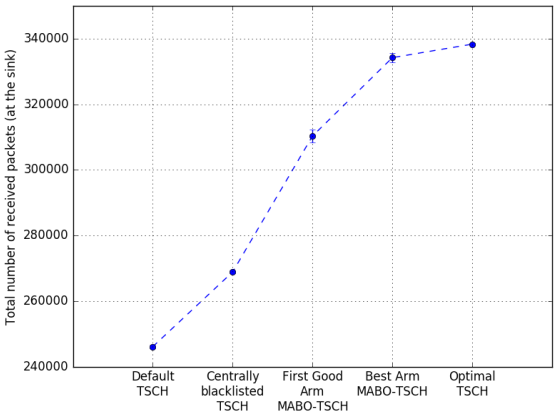

Figure 6. Total number of received packets at the sink (Soda connectivity trace).

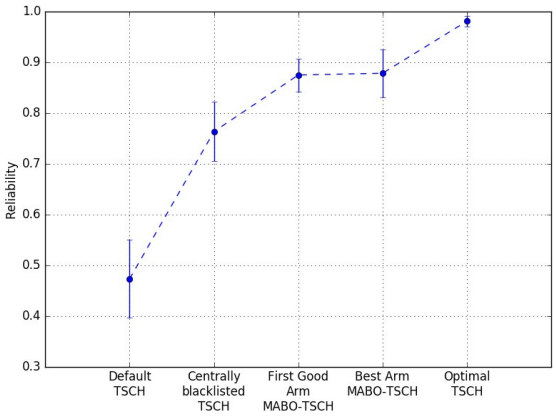

Figure 7. Reliability of packet transmissions (Tutornet connectivity trace).

In view of the channels that are employed during the network operation, *we conclude that Best Arm MABO-TSCH chooses the best channel in approximately 75% of the transmissions (Fig. 4), while Centrally blacklisted TSCH employs the best channel in approximately 60% of times*. Even small improvements in the decision-making process with regards to selecting the best channel can dramatically improve the performance of the network.

Following, we analyze the results from "Soda". Fig. 6 shows the total number of packets received at the sink. The graphs of regret per time slot and percentage of optimal channels show a behavior similar to the previous ones (from "Tutornet") and were not reproduced in the paper.

It can be concluded that the MABO-based solutions are able to improve the performance in both environments. Since when using "Soda" dataset the multiple runs were executed with the same sequence of PDR statistics, only changing the seed used in the simulation, the confidence interval of results are very small. Besides, due to the higher transmission power, a larger number of packets was received per minute in the second set of simulations.

We can also verify that the optimal parameters are environment-dependent, which may require that the tuning process be periodically repeated to keep up with changes in the network and be self-adaptive to each environment. This is discussed in detail below.

5.3.2. Evaluating event-triggered application

We now examine the case of an *event-triggered* application, where the sensor nodes have to forward a packet towards the sink node at random moments. This application is often found in alarm systems and the most important factors that must be optimized is reliability, which is the percentage of packets that successfully reach the sink, and energy consumption. It is important to note that, in any network, the reliability can be optimized to 100%, as long as a sufficient number of link-layer retransmissions are used on each hop. In the network considered for the simulations, it was not possible to obtain 100% of reliability, even with an optimal policy and 3 link-layer retransmissions. One reason that made us use traces with such weak links was to test extreme cases of poor network connectivity.

Since children nodes are those that are aware of which time slots will be used for data communication, they must be responsible for building/maintaining the blacklist, using the data packets to embed this information. Alg. 4 and 5 should be used in this case. The overhead, in this case, is perceived by the application because the available data payload is reduced to up to 8 bytes (if the rank list is used as blacklist information). In the experiment, each node randomly chooses to transmit one packet at the beginning of every slotframe with a probability p equal to 0.01.

We also simulate all 5 types of TSCH to find the optimal parameters in the algorithms, similarly to Section 5.3.1. Initially we consider "Tutornet" dataset. *The optimal value of N was equal to* 12. *The optimal value of ϵ was found to be* 0.125. *And the optimal k was* 4. Then, we consider "Soda" dataset. *In this case, the optimal value of N was equal to* 10. *The optimal value of ϵ was found to be* 0.06 *and the optimal k was* 5. The number of link-layer retransmissions was equal to 3 in all the simulations.

Fig. 7 shows the average reliability of 5 different types of TSCH. Fig. 8 shows the average number of retransmissions per packet. These two last figures considered the "Tutornet" dataset.

It can be seen in Fig. 7 that both MABO-based solutions improved the average reliability from about 50% for *Default TSCH* to almost 90%. They outperformed the centralized solution by more than 10% of improvement.

Fig. 7 shows that the improvement in reliability is due to the reduction in the number of retransmissions required. While a blind frequency hopping required that every packet is re-transmitted at least 1.4 times, blacklisting solutions (both centralized and MAB-based) reduce the average number of retransmissions to 0.8. Reducing the

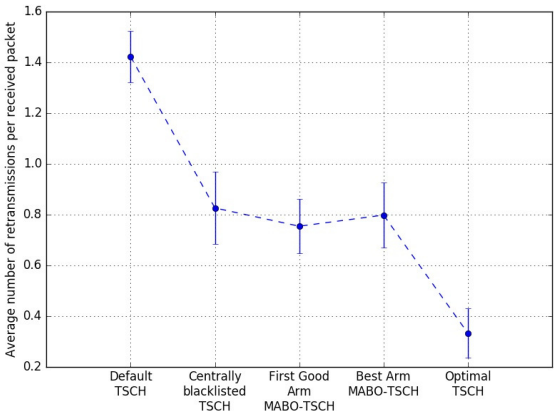

Figure 8. Average number of retransmissions per successfully received packet (Tutornet connectivity trace).

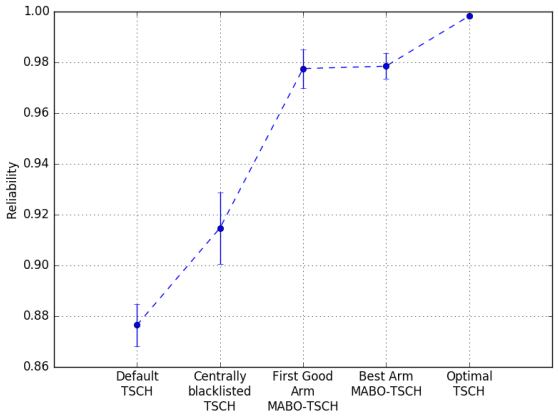

Figure 9. Reliability of packet transmissions (Soda connectivity trace).

number of retransmissions impact mainly the power consumption of sensor nodes and the overall delay for packets.

Finally, Fig. 9 shows the average reliability and Fig. 10 shows the average number of retransmissions per packet, both figures obtained from simulations with "Soda" dataset.

Since "Soda" dataset used a higher transmission power (0 dBm), we can notice that reliability was close to 88% even for *Default TSCH*. The MABO-based solutions increase the reliability from about 88% to 98%, and also outperform the centralized solution. The average number of retransmissions is also reduced from more than 1.0 to about 0.3.

The results involving two different datasets show that the algorithms adapt to different environments with distinct profiles of interference and multipath fading. As noticed previously, the environment changes the optimal parameters to be employed in the algorithms. However, the fine-tuning process should not incur major overhead on

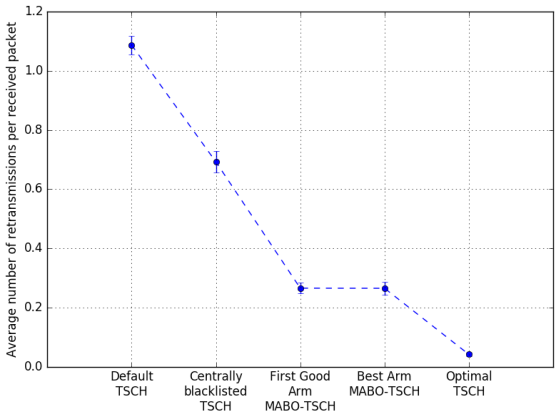

Figure 10. Average number of retransmissions per successfully received packet (Soda connectivity trace).

the network operation, since it can use PDR statistics from regular traffic and the offline fine-tuning process takes only a few seconds to be executed in a 40-node network.

6. EXPERIMENTAL RESULTS

To evaluate our solutions experimentally, we implement *First Good Arm MABO-TSCH* and *Best Arm MABO-TSCH* on OpenWSN [18][¶]. The default OpenWSN implementation is modified to disable the use of RPL protocol. As in the simulation, we use a fixed routing tree and static schedule based on MCC. The same set of 40 nodes is used as that from which the simulation traces were gathered. Node #1 is set as the sink, the other 39 nodes as sensors. All nodes are located on the same floor of the Tutornet testbed[‖].

Similarly to the simulations settings, the time slots size is set to 15 ms. The slotframe is formed of 101 time slots, and repeats approximately every 1.5 s. Within each slotframe, there were 39 reserved time slots used for unicast communication (in which the blacklisting techniques are employed), 5 shared time slots for beaconing, and 56 time slots used for serial communication for logging and turning the radio off.

Three solutions are examined in the experiments: *Default TSCH*, *First Good Arm MABO-TSCH* and *Best Arm MABO-TSCH*. For each setting, we execute 4 h experiments with 5 repetitions each. The repetitions are scattered throughout the day, including business and non-business hours. Experiments with different algorithms are repeated at similar times of the day to obtain close levels of external interference. An extra node is used to measure

[¶] As an online addition to this article, the source code is available at `http://anrg.usc.edu/www/Downloads/`.

[‖] See `http://anrg.usc.edu/www/tutornet/` for a map of the testbed deployment.

DOI: 10.1002/ett
Prepared using ettauth.cls

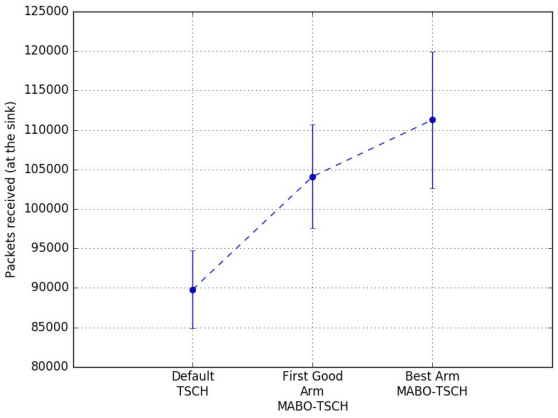

Figure 11. Total number of received packets at the sink in the testbed-based experiment.

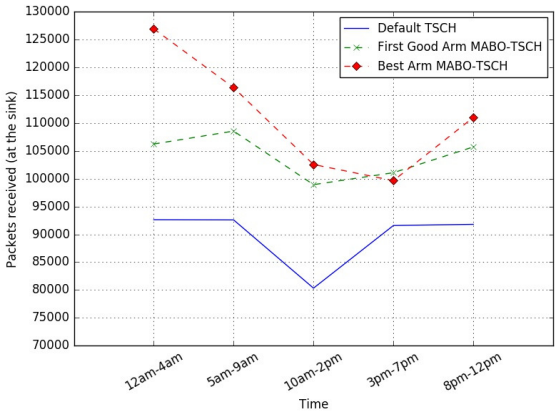

Figure 12. Total number of received packets at the sink over time.

external interference levels, so we can quantify how similar the interference is in different experiments.

ϵ is fixed at 0.05 for *Best Arm MABO-TSCH*. For *First Good Arm MABO-TSCH*, ϵ is fixed to 0.03 and k is set to 6.

Fig. 11 shows the total number of packets received at the sink. *It is possible to see an average improvement of about 23% for Best Arm MABO-TSCH, when compared to Default TSCH.* The larger confidence interval of *First Good Arm MABO-TSCH* and *Best Arm MABO-TSCH* is due to the fact that MAB algorithms need to adapt to the environment dynamics and may suffer larger variation at moments when the environment changes drastically (i.e. when external interference increases).

Fig. 12 shows the total number of packets received for each of the 5 repetitions of the experiment, and the time each repetition is executed. Both MAB-based algorithms outperform *Default TSCH* in all experiments. During non-business hours (12am-9am), the improvement is much higher than during business hours (9am-5pm), when external interference increases. Even though all experiments are repeated at similar periods of the day, the external interference perceived by each experiment is different, since we cannot control the use of WiFi in the building.

From the noise measurements, we calculate the correlation between the different experiments. We consider a moving average with length 40 to filter the measurements, and calculate the minimum correlation for all 16 channels. All experiments had a cross-correlation of at least 0.25 on all channels, except channels 15, 16, 17, 18, 19 and 22, where the correlation was close to 0 (meaning that the interference pattern was different at these channels). Even though external interference patterns differ, we did not identify any anomaly that could have drastically interfered in the results. Such differences in external interference may justify the slightly better performance of *First Good Arm MABO-TSCH* during the interval of 3pm-7pm over *Best Arm MABO-TSCH*.

Fig. 13 shows the channel usage in all 5 experiments for each type of TSCH. *Default TSCH* uses all channels equally, as expected, while *First Good Arm MABO-TSCH* and *Best Arm MABO-TSCH* employ more often channels with less interference, such as 20, 25 and 26. *First Good Arm MABO-TSCH* and *Best Arm MABO-TSCH* still select channels with high levels of interference because of the limited number of channel offsets available in the optimal schedule created by MCC. Hence, there is a trade-off between the size of the schedule (which leads to higher throughput) and the freedom to choose the best channel at every time slot. It is also clear that the optimization of the TSCH schedule influences the use of blacklists, since optimal schedules restrict the set of channels that can be used. For the purposes of maximizing throughput, reducing the size of the schedule may be preferred (even if it reduces the availability of channels). On the other hand, when improving reliability in event-triggered applications, the size of the schedule is less important, since being able to pick the best channels is the crucial factor to reduce the number of retransmissions per packet.

7. LESSONS LEARNED

From results both in simulation and real experiments, MABO-TSCH showed that online learning is a viable solution for the best channel selection problem. Below we pinpoint a few lessons learned from this work:

(i) MABO-TSCH is effective for both periodic and non-periodic data traffic;

(ii) If we want to optimize the overhead and utilize ACK packets to embed the blacklist information, periodic data traffic is required, since the parent has to know when packets are expected. In multi-hop scenarios, however, such configuration may lead to errors in channel quality estimation, since hops closer to the

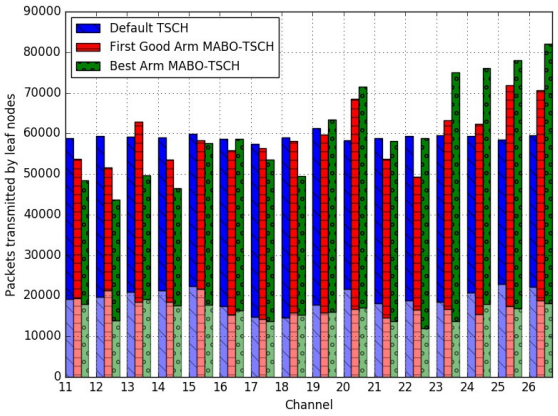

Figure 13. Channels used by all leaf nodes (light bars are failed transmissions and dark bars successful).

sink node will face more errors due to packets losses at links closer to the leaf nodes;

(iii) In simulation, MABO-TSCH outperforms an ideal centralized solution and gets results close to the optimal solution, which demonstrated the effectiveness of online learning;

(iv) In simulation, MABO-TSCH has been shown to require large intervals to converge, which may justify the worse results obtained in real experiments, and require fine adjustment to other network parameters. One solution is to dynamically adapt the moving average weight used for average reward calculation, such that measurements after intervals with high loss are more relevant;

(v) MABO-TSCH has been shown to improve the performance of event-triggered applications. However, in this type of traffic, the channel quality estimation is a major challenge, since the rate of packet transmissions is low and, hence, the estimation process is affected. One way of overcoming this issue is to leverage lower-layer packets, such as RPL's keep-alive messages, to improve the channel quality estimation. Since such lower-layer messages are not critical, they could be scheduled such that all channels would be periodically sampled in a round-robin fashion, helping the channel quality estimation to obtain a better picture of the environment.

The development of MABO-TSCH is, to the best of our knowledge, the first work to employ online learning in TSCH networks.

8. CONCLUSIONS AND FUTURE WORK

This article introduces MABO-TSCH, a solution for distributed blacklisting that is optimized for multi-hop networks and compliant with the IEEE802.15.4 TSCH standard. The solution comprises of three main algorithms. The *first* algorithm assigns multiple channel offsets for each time slot, so that each link has a set of frequencies to choose from, all of them orthogonal to other scheduled links, completely avoiding interference between nodes in a multi-hop network. The *second* algorithm provides a pair-wise blacklist negotiation mechanism with little overhead. The *third* algorithm offers a channel quality estimation based on multi-armed bandit problem that is able to achieve near-optimal results without any type of learning phase or hardware requirement.

Considering the main scenario studied, with a 40-node indoor network, MABO-TSCH outperforms the default blind frequency hopping with a 43% higher throughput in the simulation, and 23% higher throughput in the real experimentation. MABO-TSCH selects the best frequency approximately 75% of the times. It improves network performance even when just a limited number of channels offsets are available to be employed in each time slot.

Creating optimal TSCH schedules limits the set of frequencies that the blacklisting mechanism can use. There is a trade off between the optimality of the schedule (its length), and the usefulness of blacklisting. An open question is how the joint optimization of schedules and blacklisting mechanism can improve even further the performance of networks in highly dynamic environments.

Even stochastic online algorithms such as MAB-based require the use of parameters that may impact the reactivity of the learning process in dynamic scenarios. In this work we employed ϵ-greedy strategy and were able to obtain great improvement with fixed single ϵ value. However, it is clear that dynamically adjusting ϵ would improve even further the results. This is another open question to be explored in future works.

We also intend to integrate MABO-TSCH with the RPL routing protocol and use distributed scheduling based on the 6top protocol being standardized by the IETF 6TiSCH working group.

ACKNOWLEDGMENTS

This work was partially supported by NSF grants CCF-1423624 and AST-1248017, and by Inria through the DIVERSITY associate team and the European Commission through the H2020 F-Interop and ARMOUR projects.

REFERENCES

1. Accenture. Driving Unconventional Growth through the Industrial Internet of Things 2014.
2. 802.15.4-2015 - IEEE Standard for Low-Rate Wireless Personal Area Networks (WPANs) 2015.

3. Srinivasan K, Kazandjieva Ma, Agarwal S, Levis P. The β -factor : Measuring Wireless Link Burstiness. *Computer* 2008; :29–41.
4. Sexton D, Mahony M, Lapinski M, Werb J. Radio Channel Quality in Industrial Wireless Sensor Networks. *2005 Sensors for Industry Conference*, 2005; 88–94.
5. Watteyne T, Mehta A, Pister K. Reliability Through Frequency Diversity: Why Channel Hopping Makes Sense. *ACM Symposium on Performance Evaluation of Wireless Ad Hoc, Sensor, and Ubiquitous Networks (PE-WASUN)*, ACM: Tenerife, Canary Islands, Spain, 2009.
6. Kohvakka M, Kuorilehto M. Performance Analysis of IEEE 802.15.4 and ZigBee for Large-scale Wireless Sensor Network Applications. *ACM Symposium on Performance Evaluation of Wireless Ad Hoc, Sensor, and Ubiquitous Networks (PE-WASUN)*, ACM, 2006; 48–57.
7. Anastasi G, Conti M, Di Francesco M. A Comprehensive Analysis of the MAC Unreliability Problem in IEEE 802.15.4 Wireless Sensor Networks. *IEEE Transactions on Industrial Informatics* 2011; **7**(1):52–65.
8. Baccour N, Koubâa A, Mottola L, Zúñiga MA, Youssef H, Boano CA, Alves M. Radio Link Quality Estimation in Wireless Sensor Networks. *ACM Transactions on Sensor Networks* 2012; **8**(4):1–33.
9. Kuleshov V, Precup D. Algorithms for Multi-armed Bandit Problems. *CoRR* 2014; **abs/1402.6028**.
10. Du P, Roussos G. Adaptive Time Slotted Channel Hopping for Wireless Sensor Networks. *Computer Science and Electronic Engineering Conference (CEEC)*, IEEE, 2012; 29–34.
11. Tavakoli R, Nabi M, Basten T, Goossens K. Enhanced Time-Slotted Channel Hopping in WSNs using Non-Intrusive Channel-Quality Estimation. *International Conference on Mobile Ad Hoc and Sensor Systems (MASS)*, IEEE, 2015; 217–225.
12. Gai Y, Krishnamachari B. Distributed Stochastic Online Learning Policies for Opportunistic Spectrum Access. *IEEE Transactions on Signal Processing* 2014; **62**(23):6184–6193.
13. Anandkumar A, Michael N, Tang A. Opportunistic Spectrum Access with Multiple Users: Learning under Competition. *IEEE International Conference on Computer Communications (INFOCOM)*, IEEE, 2010.
14. Liu K, Zhao Q. Distributed Learning in Multi-Armed Bandit With Multiple Players. *IEEE Transactions on Signal Processing* 2010; **58**(11):5667–5681.
15. Welsh DJ, Powell MB. An Upper Bound for the Chromatic Number of a Graph and its Application to Timetabling Problems. *The Computer Journal* 1967; **10**(1):85–86.
16. Vermorel J, Mohri M. *European Conference on Machine Learning (ECML)*, chap. Multi-armed Bandit Algorithms and Empirical Evaluation. Springer, 2005; 437–448.
17. Chen Y, Gomes PH, Krishnamachari B. Multichannel Data Collection for Throughput Maximization in Wireless Sensor Networks. *IEEE International Conference on Mobile Ad hoc and Sensor Systems (MASS)*, IEEE: Philadelphia, PA, USA, 2014; 443 – 451.
18. Watteyne T, Vilajosana X, Kerkez B, Chraim F, Weekly K, Wang Q, Glaser S, Pister K. OpenWSN: a Standards-based Low-power Wireless Development Environment. *Transactions on Emerging Telecommunications Technologies* 2012; **23**(5):480–493.