Kausa: KPI-aware Scheduling Algorithm for Multi-flow in Multi-hop IoT Networks

Guillaume Gaillard, Dominique Barthel, Fabrice Theoleyre, Fabrice Valois

Kausa: KPI-aware Scheduling Algorithm for Multi-flow in Multi-hop IoT Networks

Guillaume Gaillard[3], Dominique Barthel[1], Fabrice Theoleyre[2], and Fabrice Valois[3]

[1]Orange Labs R&D, F-38240, Meylan, France. dominique.barthel@orange.com
[2]ICube, Université de Strasbourg / CNRS, F-67412 Illkirch, France. theoleyre@unistra.fr
[3]Univ Lyon, INSA Lyon, Inria, CITI, F-69621 Villeurbanne, France.
{fabrice.valois,guillaume.gaillard1}@insa-lyon.fr

April 25, 2016

Abstract

The telecommunication operators focus on the Internet of Things (IoT) and route the traffic of several clients on a multi-hop infrastructure. Operators need to offer Service Level Agreements (SLAs) to each client, guaranteeing a minimum reliability or a maximum delay for each application. The deterministic IETF 6TiSCH protocol stack is particularly appropriate to provide SLA guarantees, because it allocates dedicated time-frequency blocks for a given traffic. We propose *Kausa*, a scheduling algorithm to assign a route and allocate resources to each client flow. We optimize the network lifetime while respecting the flow-level requirements. Kausa efficiently deals with lossy links, by scheduling ad-hoc retransmission opportunities. It limits both the buffer occupation and the end-to-end delay. Our simulations mimic multiple scenarios on multi-hop topologies, highlighting the relevance of our approach.

Keywords: SLA, IoT, multi-hop, FTDMA, reliability, scheduling, resource allocation, delivery, delay, backtracking

This work has been accepted to be published in the ADHOCNOW 2016 Conference Proceedings. The final publication will be available at Springer via http://dx.doi.org/[insert DOI].

1 Introduction

With the Internet of Things (IoT), the density of wireless devices and data traffic grows in the cities, increasing the radio channel occupation and the interference. The new digital services such as telemetering or smart parking require reliability and timeliness. In this context, the delivery of the data in short time is harder.

Instead of having IoT applications compete against one another to access the radio channel, we adopt the point of view of an operator dedicated to the IoT. In order to scale with the increasing traffic demand, the operator shares a relay infrastructure for its clients over the city. The characteristics of each application are specified through specific Service Level Agreements (SLA) [2]. We assume that the operator globally allocates the resource for each client. Centralized scheduling allows a better spectrum usage and a larger network capacity.

The operator must satisfy a specific level of Quality of Service (QoS) for each client flow, as defined in the SLAs [2]. SLAs specify both the maximum intensity of the offered traffic and the expected Key Performance Indicators (KPIs): the end-to-end delay and the Packet Delivery Ratio (PDR). For instance, data collection applications such as gas metering, require

all the generated information: the effective delivery of each and every meter value is the main KPI. Applications such as pollution or light monitoring require a quick detection of a change: the main KPI is the delay.

The Packet Error Rate (PER) of the links depends on the radio environment. It has variable impact on the end-to-end delivery. Retransmissions allow to satisfy the PDR constraint, but they increase the delay and the traffic load. Indeed, the operator needs to over-provision resource to satisfy the QoS.

The IETF 6TiSCH Working Group brings IPv6 over the Time Slotted Channel Hopping mode of IEEE 802.15.4e. 6TiSCH is a good candidate technology because it benefits from the Frequency and Time Division Multiple Access (FTDMA) technology [11]. The operator is able to allocate resources to each flow, to quantify the remaining network capacity, and to adapt to interference.

One can build the FTDMA schedule with the Traffic-Aware Scheduling Algorithm (TASA) [6] and the routing graph provided by RPL [6]. Using the Expected Transmission Count (ETX) metric, the scheduler over-provisions enough resource to enable retransmissions and hence satisfies the reliability constraint. However, this approach reduces the set of used routes to the ones provided by RPL: the load is not correctly balanced over the nodes. This also increases the buffer occupation and reduces the network efficiency.

Furthermore, TASA does not consider long messages that are fragmented in order to be transmitted on various time-frequency blocks. In this case the end-to-end delay is important because the fragments are independently scheduled.

As far as we know, no resource allocation algorithm exists that both considers different flow-level KPIs, and balances the traffic load over the network topology.

Our contribution is three-fold:

1. we first assign a route to each flow, based on the traffic load and the reliability constraints. In particular, we adopt a multi-path approach: different flows from the same source may use different paths;
2. we propose Kausa, a KPI-aware scheduling algorithm supporting fragmentation. We provide a backtracking technique that enhances its performance;
3. we demonstrate with simulations the performance of our approach in terms of SLA satisfaction and allocation efficiency.

2 Related Work

2.1 Technical background: a general vision of 6TiSCH

FTDMA enhances the multi-channel technology, and reduces interference [10]. Several packets are multiplexed on independent channels and time slots. The nodes are synchronized so that they share an FTDMA schedule [8].

The IETF 6TiSCH Working Group runs an IP stack over IEEE802.15.4e TSCH networks [11]. In TSCH networks, the FTDMA schedules are build on periodical set of time slots, named *slotframes*. At each slot, every channel is mapped to a new channel offset. This *channel hopping* mechanism enhances the reliability of FTDMA by spreading the failure probability among the channels [8]. 6TiSCH considers both centralized and distributed schedule construction [6]. Each node has a *Scheduling Function* [11] to manage its allocations.

6TiSCH provides the possibility to dedicate resource to a multi-hop flow. A *track* is a set of time-frequency blocks along a route, that may only be used for the transmissions of determined frames [11]. Each node forwards the fragments according to their track IDs.

2.2 SLA-aware Scheduling

In order to satisfy the SLAs, the operator needs to manage and schedule the client traffic [2]. The schedule must consider the flow-level KPIs as constraints while maximizing the lifetime of the network.

Decentralized or distributed algorithms enable reactivity (e.g. in case of mobility, or node failure) and limit the control overhead by allowing local decisions.

In the D-SAR [12] and CFDS [5] algorithms, the source nodes request QoS whenever they have traffic to transmit. According to the required bandwidth and QoS of each source node, the allocation is built *en route* to the destination. The QoS demands are supposed dynamic, changing with each traffic burst. Neither D-SAR nor CFDS provide stable guarantees of QoS for a given flow, because they are designed for temporary reservations. Phung et al propose a multi-channel schedule based on distributed reinforcement learning and adaptation to network changes [7]. Being local to each node, the autonomous reinforcement learning does not provide end-to-end guarantees, necessary for satisfying the SLAs.

TASA centrally allocates resource for a given traffic load on each node [6]. Based on 2-hop conflicting sets, TASA gives priority to the most loaded sub-parts of the given routing tree. TASA yields optimal schedule length in this case [6]. Compact schedules optimize both the energy consumption and the delay. Neither flow-level KPIs in terms of delay and delivery nor load balancing are considered.

Industrial deployments also have strong constraints on their traffic: Pöttner et. al [9] propose a scheduling method to gather data under time-critical delivery in an oil refinery. Typical expected delay in the data delivery is 3s. They partition the network into small sets of nodes (no more than 24 nodes in practice). They provide both a schedule and the required transmission powers, for one given reliability constraint. In our contribution, we consider different KPIs and network-wide techniques that reduce the overall load.

Dobslaw et al propose SchedEx [1], a scheduler extension that modifies a schedule in order to guarantee a minimal network level end-to-end reliability. The authors calculate the number of necessary retransmissions for all the packets, at each link of the routing tree. This expected number of retransmissions is defined according to the load of a radio link and its reliability. In other words, Schedex does not guarantee flow isolation with differentiated PDR requirements.

Our solution provides flow-level guarantees and favors traffic balancing on the network while efficiently allocating the time-frequency blocks.

3 Network model and properties

In this section we specify the network characteristics and the parameters considered by Kausa. Table 1 summarizes the parameters and notations we use, as well as the default values we used in the performance evaluation.

3.1 Node model

We model the IoT network as a set of wireless nodes exchanging messages in a multi-hop way. We assume the topology is known and static. Each node has a rank value that represents its distance to a gateway. Each node has one half-duplex radio interface. It can store a limited number of frames in its buffer. We consider 3 distinct types of nodes (Fig. 1):

1. **the leaf nodes:** the sensor nodes that only generate the client traffic;
2. **the relays:** the intermediary routers that forward the client traffic;
3. **the gateways:** the final receivers that collect the client traffic.

Each leaf node runs one or several applications that generate messages of constant size (possibly fragmented into several frames). We only consider upward traffic, converging toward the gateways. Fig. 1 shows a portion of topology with 3 leaf nodes, 3 relays and one gateway.

3.2 Radio link model

We use the Packet Error Rate (PER) to characterize the link quality. We assume that the PER is time-invariant on the scale of the scheduling [8]. Because channel hopping is used, we also consider the PER independent from the channel offset [8].

The radio interference is heterogeneous: depending on the concentration of the nodes and the traffic density, the PER evolves [7]. We model an heterogeneous environment using a classical Path Loss model [4]. Noise and shadowing are normal random

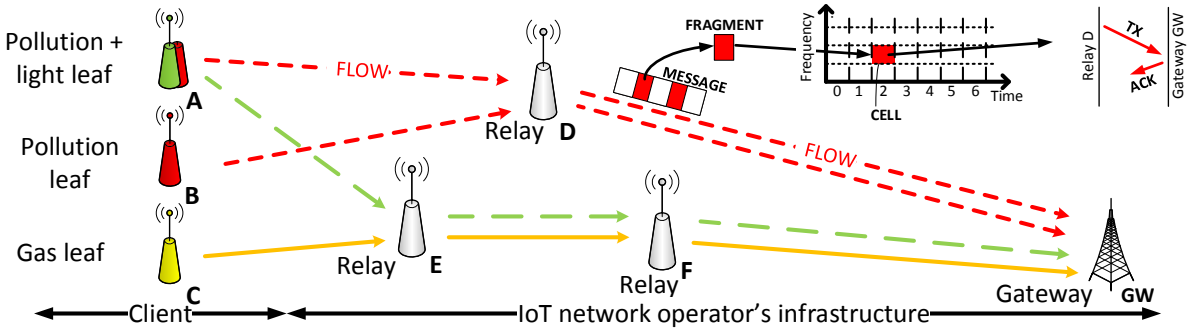


Figure 1: Network topology

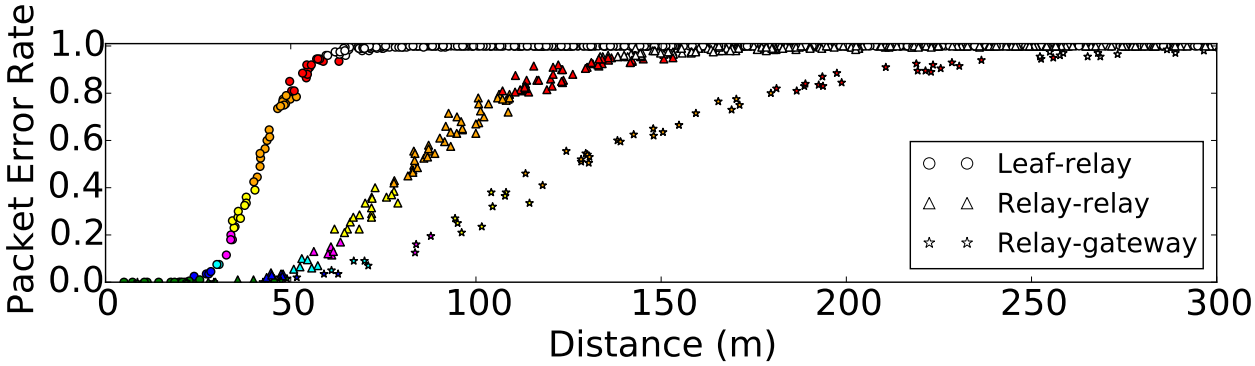


Figure 2: Distribution of Packet Error Rates (PER) by role type.

variables. Fig. 2 shows the variations of the PER according to the distance, calculated for a large set of nodes. We assume the attenuation depends on the type of link because of their location and transmission power. For instance, the operator places the gateways on highpoints whereas the leaf nodes suffer bad radio conditions. The routers are located outdoor and use high transmission power (e.g. 3 dBm). We set the path-loss exponents and the reference distances in the typical ranges (Eq. 2.53 in [4]).

Using FTDMA, the medium is divided into time slots and frequency channels. We assume that co-channel interference is only limited to a few hops (e.g. 1 or 2). The algorithm allocates time-frequency blocks, named *cells*, within the FTDMA schedule (Fig. 1). The schedule is divided into periodical slotframes (set of time slots) that repeat in time, indefinitely.

Typically, during one cell, 2 neighboring nodes can exchange a frame of 127 B and its corresponding acknowledgement (Fig. 1). If necessary, an applicative message is fragmented into frames that separately transit on a single cell each.

3.3 Flow model

For each application, a leaf node creates one *flow* of messages. The client traffic is converge-cast. The operator collects the traffic of each flow at a gateway (Fig. 1).

We create *tracks* assigning a given number of cells at each hop along a path. Each track is identified with a track ID. We associate each flow to a unique

track. This way, each flow is isolated.

The scheduler provisions enough cells for the end-to-end transmission of the fragments of each message. The scheduler allocates additional over-provisioning cells in order to consider hop-by-hop retransmissions [3].

The operator only provides guarantees on the flows that respect the SLA specification [2]. We address the allocation of resource for periodical traffic patterns. For each flow, the source leaf node generates a given number of fixed-size messages at each slotframe. The leaf node buffers the corresponding fragments until their first transmission slot. During one slotframe, all the fragments must reach a gateway.

4 An overview of Kausa

We consider a centralized scheduling algorithm that allocates FTDMA cells to a set of flows. Kausa takes as input parameters:

1. each flow's SLA parameters: the traffic profile and the expected KPIs;
2. the topology information: the PER of each link, and the rank of every node.
3. interference: the maximum hop distance after which we neglect interference.

In order to limit the final number of allocated cells, the algorithm considers, per hop, a maximum number of hop-by-hop retransmissions: either for each fragment ($n_{rtx}^{max(frag)}(f)$) or for each message ($n_{rtx}^{max(msg)}(f)$).

Fig. 3 shows the allocation process. Kausa first builds a path for each flow (step 1). Each path is chosen in order to balance the load over the network while satisfying a minimum reliability. Kausa computes the number of cells needed at each hop to satisfy the PDR, according to the PER of each link (step 2).

If the chosen path does not suit the PDR or delay KPIs, a link-level backtracking takes place (Sec. 5.6). New paths are considered by recursively eliminating links from the possibilities. We avoid both the most loaded and the least reliable portions of the network. The recursion stops when the source has no more possible links to a next hop. The link-level backtracking favors the respect of the KPIs at the expense of the load-balancing.

The set of cells associated to a message at a given hop is named a *range*. At each hop, one *range* contains enough cells for the transmissions of the fragments of each message, as well as enough over-provisioned cells for the anticipated retransmissions. Kausa sequentially allocates the cells within each range into the FTDMA schedule (step 3). In Fig. 3, Kausa allocates flow $k+1$ considering the cells already allocated for flow k and the previous flows.

In order to avoid buffer overflows, we limit the maximum amount of buffered fragments on each node, considering the least favorable scenario of retransmissions. Kausa accordingly anticipates or delays the allocations of a given message.

If no allocation is found for a given message, a flow-level backtracking takes place (Sec. 5.7). We successively remove the allocations of the previous flows, look for new paths, and try other allocations. The flow-level backtracking stops when it reaches the first allocated flow.

5 Detailing Kausa: an SLA-aware FTDMA scheduler

5.1 Ordering the flows

We order the flows in decreasing order of the load metric, so that Kausa first deals with the most greedy ones. We compute the load metric directly from the SLA parameters (Table 1). In Eq. (1), the load metric corresponds to the number of expected fragments during a slotframe, for a given flow f:

$$n_{msg}(f) \cdot n_{frag}(f) \cdot PDR_{msg}^{sla}(f) \qquad (1)$$

In case two flows have similar values for this metric (e.g. less than 1% difference), we consider the expected delay $DEL_{msg}^{sla}(f)$ as the second ordering criterion.

When several flows have the same KPIs (typically for one multi-source application, e.g. telemetering),

Table 1: Parameters of the network model and their values in the simulations

Variable	Explication	value	range
	Size of slotframe (slots)	1000	[100,1200]
	Number of channels	16	
	Number of leaves running each app.	100	
	Number of relays, sinks	24, 2	
$n_{msg}(f)$	Num. of messages per slotframe for a flow f	1	[1, 12]
$n_{frag}(f)$	Num. of fragments per message (app.1, app.2)	2, 3	
$PDR^{sla}_{msg}(f)$	Expected end-to-end PDR (app.1)	0.80	[0.80, 0.98]
	Expected end-to-end PDR (app.2)	0.97	[0.97, 0.997]
$DEL^{sla}_{msg}(f)$	Expected end-to-end delay (slots) (app.1)	60	[30, 140]
	Expected end-to-end delay (slots) (app.2)	90	[45, 210]
$pdr^{sla}_{frag}(f)$	Expected end-to-end PDR - fragments of f		
$per(i)$	Packet Error Rate (PER) on the i_{th} link of a path		
$n_{cell}(A)$	Number of allocated cells on a node A		
$n^{max(msg)}_{rtx}(f)$	Max. num. of retrans. per (hop, msg)	16	
$n^{max(frag)}_{rtx}(f)$	Max. num. of retrans. per (hop, frag)	8	
$n^{max}_{buffer}(f)$	Maximum buffer occupation	20	

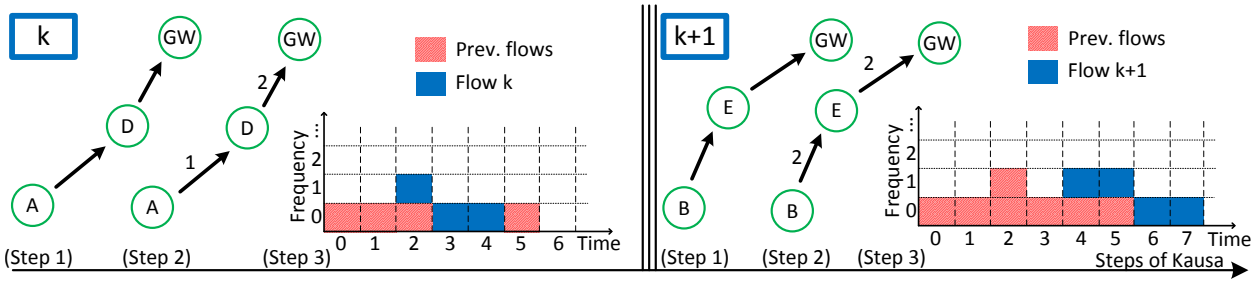


Figure 3: Step-by-step overview of Kausa, for two flows k and $k+1$.

they have the same load metric and delay. We use the source rank to decide between the flows (and in case of identical source rank the track ID). We give the furthest source in terms of rank the highest priority, since the route may be longer, and hence the load higher for the considered flow.

5.2 Building the paths

We provide a load-balancing and KPI-aware multi-path mechanism. Our objective is to minimize the impact of each flow on the network in terms of allocation load. We construct the path for one flow, taking into account the previous allocations. We need to find valid next-hop nodes for each router in the path, in terms of traffic load and reliability.

We update the routing topology, starting from the gateways and using a vector-distance (e.g. Bellman-Ford) algorithm. We choose for each node the next-hop forming the best route to a gateway, in terms of maximum node load, route load, and ETX. We consider the three criteria in lexicographical order:

$$< \max_{i\in[1,l_{gw}]} n_{cell}(A_i), \sum_{i\in[1,l_{gw}]} n_{cell}(A_i), \sum_{i\in[1,l_{gw}]} ETX(i) > \quad (2)$$

In Eq. (2), the set of links $i \in [1, l_{gw}]$ is the route from the considered node to a gateway. $ETX(i)$ is given

for each i as $1/(1-per(i))$, A_i is the transmitter of i. Note that the sum of the ETX allows us to estimate the necessary number of timeslots for the allocation over a given path.

The next-hop of a node is a neighboring relay with a lower rank to avoid loop. This allows us to build robust path. We verify that the obtained path is sufficiently reliable while considering independent fragment transmissions [3]. Kausa disregards any path that requires, at any given link and for one fragment, more retransmissions than the maximum $n_{rtx}^{max(frag)}(f)$: thus, we avoid paths with too low a reliability. Eq. (3) expresses the satisfaction of the fragment reliability constraint using the maximum number of retransmissions:

$$\prod_{l=1}^{l_{gw}} \left(1 - per(l)^{n_{rtx}^{max(frag)}(f)}\right) \geq pdr_{frag}^{sla}(f) \qquad (3)$$

If (3) is not verified, the chosen link is not reliable enough: the path is not valid and we need to backtrack.

5.3 Computing the hop-by-hop number of allocations

We evaluate for each hop the minimum number of over-provisioning cells needed to satisfy the message PDR constraint. We use an inverse greedy algorithm starting with the max. number of retransmissions for a message, $n_{rtx}^{max(msg)}(f)$ [3].

The resulting number of allocated cells must respect the delay in case the cells are consecutively allocated. If this condition does not hold, the path cannot be valid and we need to backtrack.

5.4 Allocating the messages

The path has been computed. We now have to allocate cells for each hop. The allocation is done message after message, for each ordered flow. We associate the message with one range for each hop in the path (Fig. 4). Each range is a set of cells dedicated to the message on a given link.

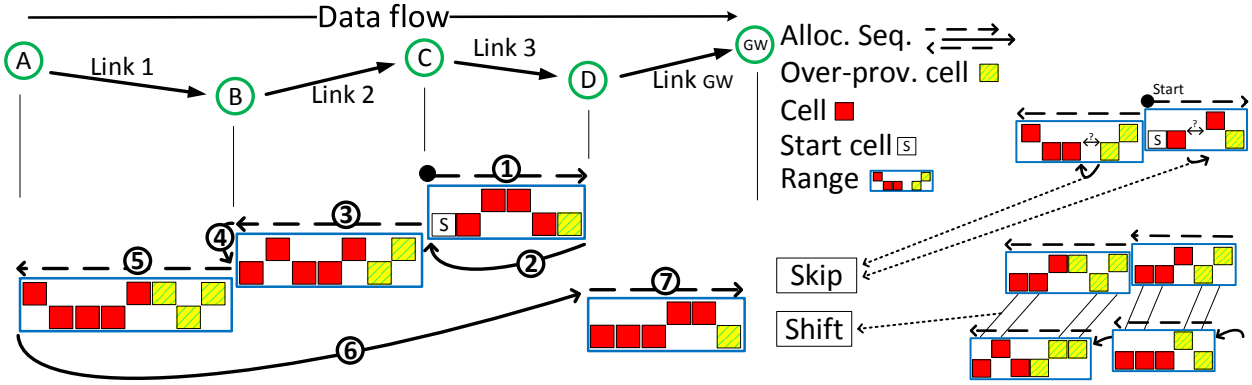
The ranges are allocated in sequence. In order to avoid scheduling the next transmission before the last reception, the ranges of the same message are contiguous but do not overlap. Indeed, a set of ranges is valid if it respects the delay constraint: the difference between the first slot of the first range and the last slot of the last range is less than the delay (in slots). Fig. 4a represents the allocation sequence for a given message, on a path from a leaf node A to a gateway. The numbered arrows show the allocation order. The sequence begins with the *starting range* at the link with the most allocated cells in the slotframe: e.g. at step 1, the range of Link 3 is the starting range.

1. **Allocating the cells within the starting range:** we first have to position the cells of the starting range. We iteratively try all the possible *start cells* by scanning the whole slotframe. We finally select the starting range which minimizes, for all the cells, the channel occupation. We terminate the scan early if we find a zero value for a range before finishing the slotframe.

2. **Allocating the cells for the previous hops:** we allocate the cells of the previous hops from last to first in upward direction. Next considered link is Link 2, step 2 and 3, in Fig. 4a. We choose to minimize the buffering delay, starting from the cell closest from the next *start cell*. We repeat the same process (step 4 and 5) until we reach the first hop AB.

3. **Allocating the cells within the following hops:** the ranges of cells for the next hops are allocated consecutively to the last cell of the starting range (step 6). If the cell conditions cannot be verified or the delay constraint is not satisfied, scanning all the slotframe, the range is not valid.

4. **Modifying the starting range:** if the allocation fails on a given hop, or if the delay constraint is not verified, the allocation starts again for the same message with the second best starting range. We resume the search if it was not finished, and choose the new start cell accordingly.



(a) Range allocation sequence.

(b) Skipping and Shifting.

Figure 4: Kausa range allocation.

If we cannot find any starting range, the allocation of the message fails and we need flow-level backtracking.

5.5 Conditions of a cell allocation

We allocate consecutive cells in a range, toward earlier or later time slots (to the source or the destination, step 5 or 7 in Fig. 4a). The cells within a range must respect the following conditions:

Half-duplex condition: the transmitter and receiver are not already using their radio interface;

Conflict-free condition: there is an available channel in the neighborhood;

Reception buffer condition: the receiving node has enough available buffer in case successful transmissions occur at the first cells of the range. This is the worst case scenario: the receiver's buffer is occupied early.

Transmission buffer condition: the sending node has enough available buffer for the not yet transmitted fragments in case successful transmissions occur at the end of the range. Among the cases that validate the SLA conditions, this is the worst case scenario: the transmitter's buffer is freed late.

If one of these conditions is not respected, we *skip* a cell (Fig. 4b) and try with the next ones. Note that the buffer conditions must be verified for the whole range, because as no fragment is transmitted in a skip, no memory is freed.

During the allocation of a given range, if the buffer conditions are no longer valid, we need to recursively *shift* the ranges to earlier in time until the problem is solved (Fig. 4b). For each range, we move the allocated cells and verify at each cell the conditions. Note that if we reach the slotframe size while shifting, the allocation is not valid and we need to modify the starting range.

5.6 The link-level backtracking procedure

Kausa builds an alternative path for a given flow (step 1 in Fig. 3):

1. when the routing algorithm does not find a suitable path for a given flow;
2. when the provision of cells fails on the path for a given flow.

One link in the path is blacklisted for the flow. Then, the new path is completed following the routing metric (Eq. 2) but avoiding the blacklisted links.

If the backtracking call is due to an impossibility to build a path or a non respect of the delay constraint, we eliminate the link with the worst PER, to build a more reliable path. In all other cases, we temporarily

blacklist the link with the highest allocated load in the neighboring nodes, to build a less loaded path. If no valid path is found at this step, we definitely eliminate the link with the worst PER, and remove from the blacklist the temporarily added links.

We do recursive backtracking calls until either we find a valid path respecting the delay (success) or all source neighbors are blacklisted (failure). When the link-level backtracking fails, our algorithm drops this flow altogether and proceeds with the next one. Thus, the link-level backtracking does not cover all the possible paths, but progressively eliminates the links with the worst quality.

5.7 The flow-level backtracking procedure

A flow-level backtracking takes place when the allocation fails for a given message (step 3 fails in Fig. 3). We first save the state of the allocation.

If the link-level procedure on the flow fails, we backtrack on the links of the previous flow (e.g. in Fig. 3, if it fails on flow $k+1$, we backtrack on step 1, flow k). When the backtracking procedure reaches the first allocated flow ($k = 0$) and fails, there is no solution under the assumptions we made. We resume Kausa with the previously saved allocation and proceed with next flow.

6 Performance Evaluation

6.1 Scenario and simulation setup

We run a Monte Carlo network simulator using Python. We compare Kausa with an extension of TASA, enabling the provision of cells for the retransmissions [3]. We run the algorithms on 16 randomly generated topologies and with 12 values of each of the 4 parameters (traffic intensity, slotframe size, expected PDR KPI, expected delay KPI). Table 1 summarizes the parameters and default values.

The simulations emphasize the impact of the variations of each parameter,the others kept at default. We use the same scenario as in [3] and our radio model (Sec. 3.2). We consider two applications with strong delay constraint (app.1) and with strong PDR constraint (app.2). For the two applications, the ranges of variations of the delay and PDR constraints are realized in proportion of the two default values (Table 1). They are expressed as a percentage in the figures.

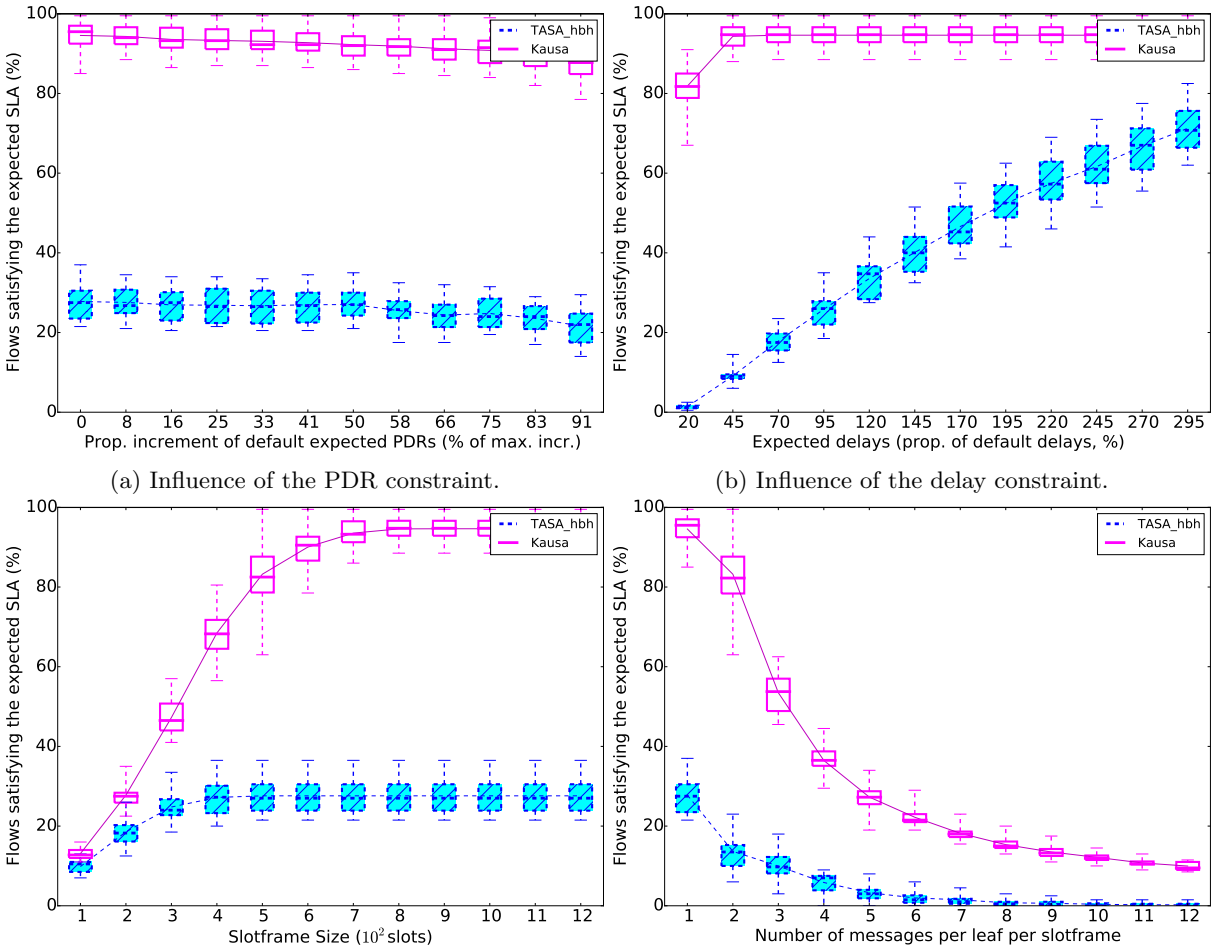
The leaf nodes are uniformly spread in a rectangle of 400×200 meters. The relays are placed on a triangle mesh (every approximately 70 meters). The 2 gateways are placed at positions (100,100) and (300,100).

6.2 Results

Fig. 5 shows that Kausa outperforms TASA with retransmissions, in terms of SLA satisfaction. With strong requirements (half the flows expect a PDR of 97%) around 90% of the flows validate their KPIs for Kausa (Fig. 5a, 5b), with a schedule length of 800 slots (Fig. 5c). The delay constraint is the main limiting factor in TASA: its performance increases almost linearly when relaxing the constraint (Fig. 5b). When the PDR constraint increases (Fig. 5a), the performance of Kausa slightly decreases: Kausa finds specific solutions depending on the topology. The two sinks are saturated when each leaf node generates more than 3 messages (Fig. 5d). Even when the intensity of traffic reaches the limit of capacity, Kausa allows some flows to be serviced (50% with 3 messages).

In Fig. 6 the total amount of allocations on the whole network is similar for the two protocols (around 2000 cells for default traffic). In Fig. 6a, for high PDRs ($\geq$ 90%, 50% of maximum increase), the allocations increase in TASA while they remain stable in Kausa. Indeed, the backtracking process leaves flows unallocated in Kausa, whereas TASA serves all the traffic indistinctly. Fig. 6b and Fig. 6c show that Kausa uses less allocations, whichever parameter is considered, because it finds paths with a better tradeoff between length and link quality. Fig. 6d shows that Kausa does not provision cells in case of saturation, and does not waste bandwidth.

Finally, Fig. 7 shows that with default traffic, the maximum buffer occupation in Kausa remains around 10 fragments (half the maximum), versus



(a) Influence of the PDR constraint.

(b) Influence of the delay constraint.

(c) Influence of the slotframe size.

(d) Influence of the traffic load.

Figure 5: Evaluation of the performance in terms of PDR satisfaction.

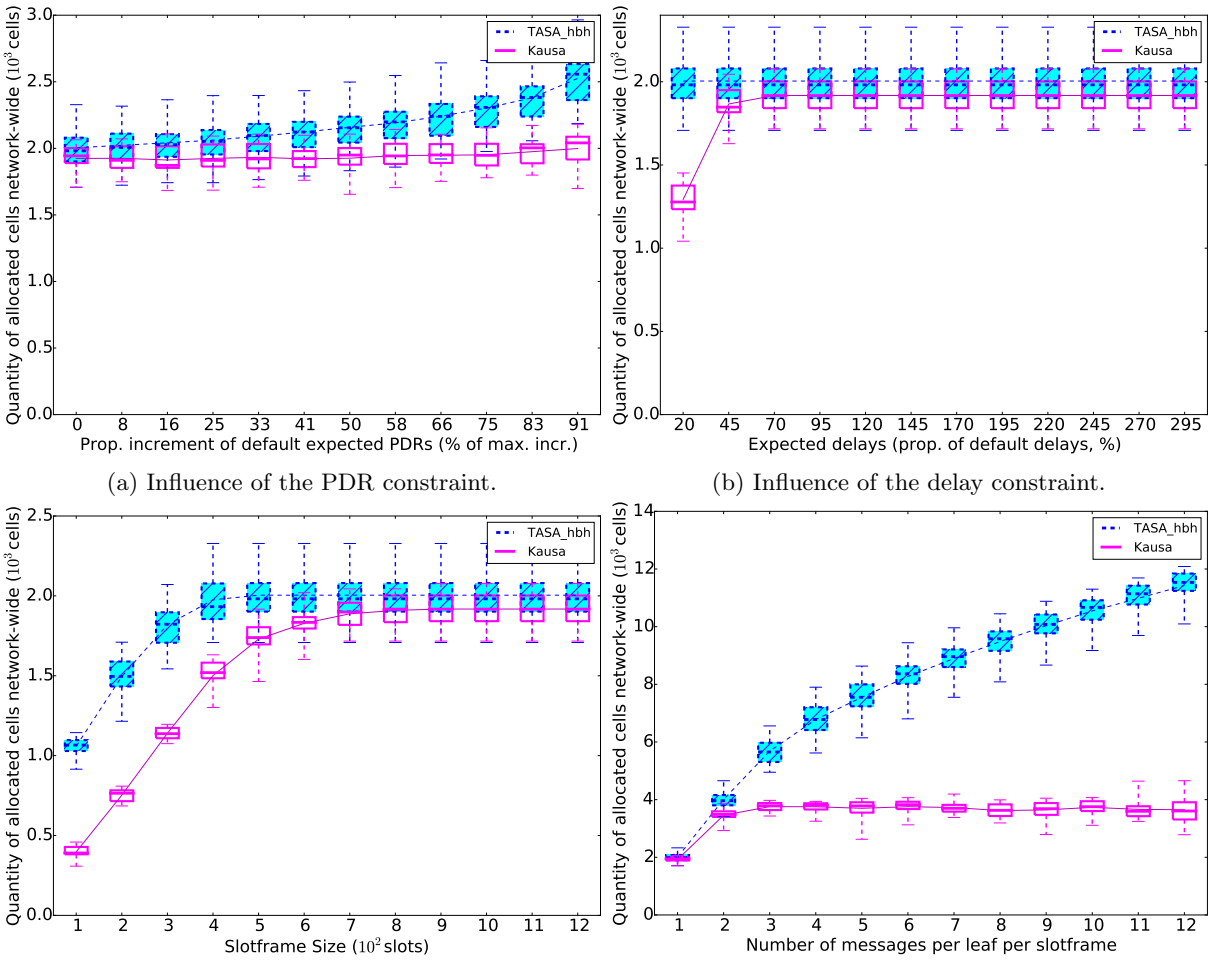
about 40 fragments for TASA (Fig. 7a, 7b, and 7c). Fig. 7d shows that with high traffic Kausa preserves the buffers better than TASA.

7 Conclusion and Future Work

The number of devices that form the IoT grows, along with the requirements of quality of service for a diversity of applications. This makes the resource allocation for IoT networks more challenging in the smart cities.

In this work, we propose a solution that centrally allocates FTDMA resources under flow-level QoS constraints. Our algorithm balances the traffic load, to prolong the network lifetime, while satisfying delivery and delay constraints for a multi-flow scenario. A controlled backtracking algorithm allows reaching a satisfying solution in a reasonable time.

We simulate an urban environment, with a set of typical applications, their characteristics and requirements. We model a FTDMA technology, where the nodes have limited buffer capacities, and where the links have different qualities. In this context, Kausa



(a) Influence of the PDR constraint.

(b) Influence of the delay constraint.

(c) Influence of the slotframe size.

(d) Influence of the traffic load.

Figure 6: Evaluation of the performance in terms of network resource usage.

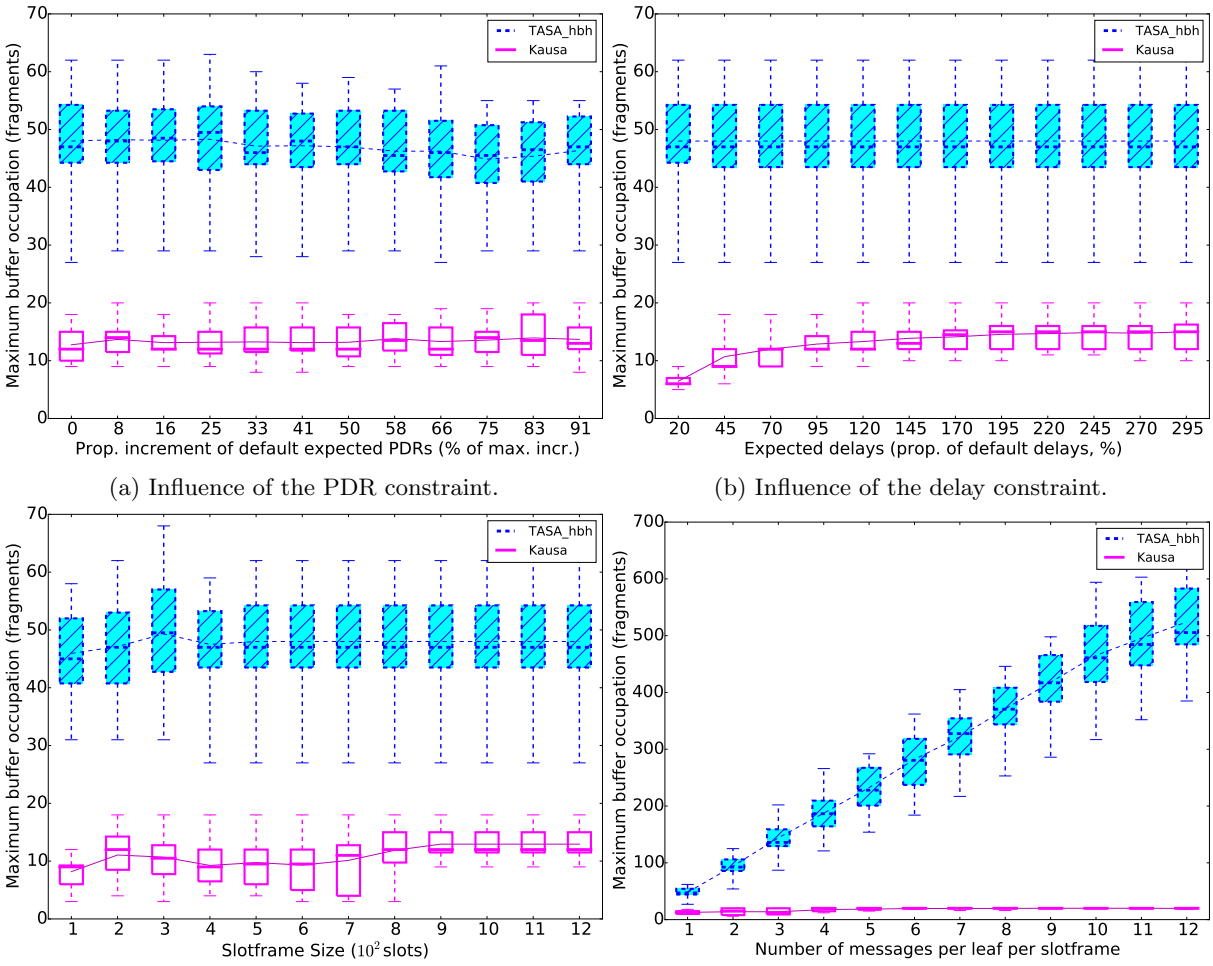
performs better than TASA in terms of SLA satisfaction. We ensure flow-level QoS without creating buffer overflows.

In a future work we will study how to enhance Kausa by taking into account different queue management strategies on each router. We envision an adaptation of Kausa for dynamic or bursty traffics.

References

[1] Dobslaw, F., Zhang, T., Gidlund, M.: End-to-end reliability-aware scheduling for Wireless sensor networks. In: IEEE Trans. on Industrial Informatics. PP(99), (2014)

[2] Gaillard, G., Barthel, D., Theoleyre, F., Valois, F.: Service Level Agreement Architecture for Wireless Sensor Networks: a WSN Operator's Point of View. In: IEEE/IFIP Network Op-



(a) Influence of the PDR constraint.

(b) Influence of the delay constraint.

(c) Influence of the slotframe size.

(d) Influence of the traffic load.

Figure 7: Evaluation of the performance in terms of node buffer occupation.

erations and Management Symposium, Krakow (2014)

[3] Gaillard, G., Barthel, D., Theoleyre, F., Valois, F.: Enabling Flow-level Reliability on FT-DMA Schedules with efficient Hop-by-hop Overprovisioning. [Research Report] RR-8866, INRIA Grenoble - Rhône-Alpes (2016)

[4] Andrea Goldsmith: Wireless communications. Cambridge university press (2005)

[5] Morell, A., Vilajosana, X., Vicario, J.L., Watteyne, T.: Label switching over IEEE802.15.4e networks. Trans. on Emerging Telecom. Technologies. (2013)

[6] Palattella, M.R., Accettura, N., Grieco, L.A., Boggia, G., Dohler, M., Engel T.: On Optimal Scheduling in Duty-Cycled Industrial IoT Applications Using IEEE802.15.4e TSCH. IEEE Sensors Journal. 13(10), 3655–3666, (2013)

[7] Phung, K., Lemmens, B., Goossens, M., Nowe,

A., Tran, L., Steenhaut, K.: Schedule-based multi-channel communication in wireless sensor networks: A complete design and performance evaluation. Ad Hoc Networks. 26, 88–102 (2015)

[8] Pister, K., Doherty, L.: TSMP: Time synchronized mesh protocol. IASTED Distributed Sensor Networks. 391–398 (2008)

[9] Pöttner, W.B., Seidel, H., Brown, J., Roedig, U., Wolf, L.: Constructing schedules for time-critical data delivery in wireless sensor networks. ACM Transactions on Sensor Networks (TOSN). 10(3),44 (2014)

[10] Soua, R. and Minet, P.: Multichannel assignment protocols in wireless sensor networks: A comprehensive survey. Pervasive and Mobile Computing. 16, 2–21 (2015)

[11] Thubert, P., Watteyne, T., Palattella, M.R., Vilajosana, X., Wang, Q.: IETF 6tsch: Combining ipv6 connectivity with industrial performance. IMIS. 541–546, (2013)

[12] Zand, P., Chatterjea, S., Ketema, J., Havinga, P.: A distributed scheduling algorithm for real-time (d-sar) industrial wireless sensor and actuator networks. In: 17th IEEE Conference on Emerging Technologies & Factory Automation (ETFA). pp. 1–4 (2012)