Real-Time Communications over Cluster-Tree Sensor Networks with Mobile Sink Behaviour

Petr Jurčík[1], Ricardo Severino[1], Anis Koubâa[1,2], Mário Alves[1], Eduardo Tovar[1]
[1]*CISTER/IPP-HURRAY Research Unit, Polytechnic Institute of Porto (ISEP/IPP), Porto, Portugal*
[2]*Al-Imam Muhammad Ibn Saud University, CCIS Department, Riyadh, Saudi Arabia*
{petr, rars, aska, mjf, emt}@isep.ipp.pt

Abstract

Modelling the fundamental performance limits of Wireless Sensor Networks (WSNs) is of paramount importance to understand the behaviour of WSN under worst-case conditions and to make the appropriate design choices. In that direction, this paper contributes with a methodology for modelling cluster-tree WSNs with a mobile sink. We propose closed-form recurrent expressions for computing the worst-case end-to-end delays, buffering and bandwidth requirements across any source-destination path in the cluster-tree assuming error free channel. We show how to apply our theoretical results to the specific case of IEEE 802.15.4/ZigBee WSNs. Finally, we demonstrate the validity and analyze the accuracy of our methodology through a comprehensive experimental study, therefore validating the theoretical results through experimentation.

1. Introduction

Wireless Sensor Networks (WSNs) emerge as enabling infrastructures for large-scale distributed embedded systems. Timeliness is an important requirement to be fulfilled in these systems. However, issues such as large scale and communication/computing and energy limitations pose important difficulties in guaranteeing a correct behaviour of these systems.

Evaluating the performance limits of WSNs is therefore a crucial task, particularly when the network is expected to operate under worst-case conditions [1]. For achieving real-time communications over sensor networks, it is mandatory to rely on deterministic routing and MAC (Medium Access Control) protocols. Usually, these networks use hierarchical logical topologies such as cluster-tree or hexagonal (e.g. [2-4]). Issues such as the use of contention-free MAC protocols (e.g. time division or token passing) and the possibility of performing end-to-end resource reservation contrast with what can be achieved in mesh-like topologies, where contention-based MACs and probabilistic routing protocols are used.

This work was partially funded by FCT under the CISTER Research Unit (FCT UI 608), PLURALITY and CMU-PT projects, ARTIST2/ARTISTDesign NoEs., and by the Czech Republic MPO project (61 03001).

In a previous work [5], the authors have provided a methodology and closed-form expressions to dimension the network resources in a cluster-tree WSN with a static sink. The sink – a central point that collects all sensory data – was assumed to be statically attached to the root. That work aimed at evaluating the worst-case network performance assuming a cluster-tree topology of balanced height and load. This symmetry property was explored to derive per-hop and end-to-end resource requirements in addition to the worst-case delays of upstream flows (i.e. from child nodes to the root).

However, while the static sink behaviour is adequate for root-centric WSN applications (e.g. a surveillance system delivering alarms to a central station), other applications may impose or benefit from collecting data at different network locations (e.g. a doctor with a hand-held computer collecting patients' status). Therefore, in this paper we investigate the worst-case resource dimensioning and analysis of cluster-tree WSNs with mobile sink behaviour assuming an error free channel. In practice, this assumption does not hold, but theoretical results are almost likely to be valid in interference free and "clean" environments (e.g. outdoor and open space areas).

We consider the sink as an autonomous entity that does not make part of the static cluster-tree WSN, but can be associated to any of its routers through any (wired or wireless) communication means. Thus, the sink mobility does not impact the WSN topology, but affects the data flow destination (any router in the WSN). Contrarily to [5], in this paper we address not only upstream flows (sink in the root) but also downstream flows (sink not in the root), thus achieve a more complete analysis, yet more complex.

For the sake of simplicity and space, this paper address neither mobility management issues (namely how routes must be updated upon mobility of the sink) nor the impact of this procedure on the worst-case analysis (potential network inaccessibility times). The paper proposes and describes a system model, an analytical methodology and a software tool that permit the worst-case dimensioning and analysis of cluster-tree WSNs. In this way, it is possible to guarantee the routers' buffers size to avoid overflows and to minimize each cluster's duty cycle

(maximizing nodes' lifetime) still satisfying that messages' deadlines are met.

Importantly, we show how to instantiate our generic methodology to IEEE 802.15.4/ZigBee [6, 7] protocols, which are very promising technologies for WSNs (In fact, in 2007, 7 million IEEE 802.15.4-enabled chips were sold, an increase of 1400% from 2004 [7]). Finally, we assess the validity of our theoretical model by comparing worst-case results (buffer requirements and message end-to-end delays) with the maximum and average values measured through an experimental test-bed based on Commercial-Off-The-Shelf technologies.

Contributions of this paper

(1) We provide a generic system model, encompassing the cluster-tree topology model and the data-flow model; we also identify the worst-case cluster scheduling for any location of the sink (Section 3).
(2) We present a methodology, based on Network Calculus, to characterize input and output flows in each router in the cluster-tree WSN (Section 4) and to derive upper bounds on buffer requirements and per-hop and end-to-end delays (Section 5).
(3) We show how to apply our methodology to dimension IEEE 802.15.4/ZigBee cluster-tree WSNs (Section 6).
(4) We demonstrate the validity of our methodology through an experimental test-bed (Section 7).

Other related work

The evaluation of fundamental performance limits of WSNs has been addressed in several research works [1-3]. In [1], the authors evaluated the asymptotic behaviour of operational lifetime and energy-constrained capacity of sensor networks. In [2], the authors have evaluated the real-time capacity of multi-hop WSNs, identifying how much real-time data the network can transfer by their deadlines. A capacity bound has been derived for (ideal) MAC protocols with fixed priority packet scheduling mechanisms. In [3], the authors have analyzed the fundamental limits for acceptable loads, utilization, and delays in multi-hop sensor networks with linear and grid topologies, in case of all sensor nodes contribute equally to the network load.

The worst-case analysis and resource dimensioning of WSNs using Network Calculus has been pursued by Schmitt *et al*. ([10-12]), who proposed the Sensor Network Calculus methodology. In [10], Sensor Network Calculus was introduced and basic components such as arrival and service curves were defined. The system model assumes generic tree-based topologies with nodes transmitting sensor data towards the sink that is associated with the root. The authors have also proposed a general iterative procedure to compute the network internal flows and, subsequently, resource requirements and delay bounds. On the contrary, our work provides recurrent equations to avoid iterative complex and time consuming computations, which are not suitable for large-scale WSNs. In [11], the previous Sensor Network Calculus framework was extended to incorporate in-network processing features (e.g. data aggregation) to reduce the amount of data that has to be transmitted. In our work, we abstract from the computational resources in the network nodes and from data aggregation. In [12], the authors have specified the worst-case topology (i.e. the topology that exhibits the worst-case behaviour in terms of buffer requirements, delay bounds and network lifetime) in networks with random nodes deployment. Finding the general worst-case topology is a complex task, thus their methodology explores the worst-case tree constrained on maximum depth and number of child routers that maximizes the arrival curve of the root. As compared to [10-12], our system model is more accurate for the specific case of cluster-tree topologies and the sink can be associated with any router in WSNs.

On the other hand, several research works have dealt with sink mobility in order to minimize energy consumption in the network [13, 14]. The proposed approaches use random, predictable or controlled mobility of one or more sinks [13]. Four strategies (random, geographically, intelligent and genetic algorithm-based strategies) focusing on optimal sink placement for minimizing the worst-case delay as well as maximizing the lifetime of a WSN have been introduced in [14]. Conversely, in our work we compute the worst-case delays and resource requirements for given sink positions.

2. Background on Network Calculus

Network Calculus [9] is a mathematical methodology based on min-plus algebra that applies to the deterministic analysis of queuing/flows in the networks. This section briefly introduces the aspects that are most significant to this paper. For additional details please refer to [9, 17].

A basic system model S in Network Calculus consists of a buffered FIFO node with the corresponding transmission link. For a given data flow, the *input function* $R(t)$ is a cumulative number of bits that have arrived to system S in the time interval $(0, t)$. The *output function* $R^*(t)$ is the number of bits that have left S in the same interval $(0, t)$. An *arrival curve* $\alpha(t)$ upper bounds the input function of a system S such that for $\forall s$, $0 \le s \le t$, $R(t) - R(s) \le \alpha(t - s)$. A *service curve* $\beta(t)$ represents a lower bound on the transmitted cumulated flow, thus for $\forall t$ there exists $t_0 \le t$ such that $R^*(t) - R^*(t_0) \ge \beta(t - t_0)$. The knowledge of the arrival and service curves enables us to determine performance bounds, namely the *delay bound* D_{max} given by the maximum horizontal distance between $\alpha(t)$ and $\beta(t)$, which represents the worst-case delay of the message traversing system S, and the *backlog bound* Q_{max} given by the maximum vertical distance between $\alpha(t)$ and $\beta(t)$, which represents the minimum buffer size requires inside S. These concepts are shown in Figure 9.

So far, we have handled a system S as a single buffered node. However, system S might also be a sequence of

nodes or even a complete network. In this case, the *concatenation theorem* enables us to investigate serial nodes in sequence as a single node.

Concatenation Theorem. *Assume a flow with input function R(t) traverses system S_1 and S_2 in sequence, where S_1 offers service curve $\beta_1(t)$ and S_2 offers $\beta_2(t)$. Then the concatenation of these two systems offers the following single service curve $\beta(t)$ to the traversing flow:*

$$\beta(t) = (\beta_1 \otimes \beta_2)(t) \quad (1)$$

where $\otimes$ *is the min-plus convolution defined for f, g* $\in \boldsymbol{F}$, *where* $\boldsymbol{F}$ *is the set of wide-sense increasing functions, as:*

$$(f \otimes g)(t) = \inf_{0 \le s \le t} \{f(t-s) + g(s)\}; \quad for\ \forall t \ge 0$$

The accuracy of the worst-case bounds depends on how tightly the selected arrival and service curves follow the real network behaviour. Different types of arrival and service curves have been proposed in Network Calculus (e.g., [9, 10]). However, the (*b*, *r*) arrival curve and rate-latency service curve are the most used in such network models. The (*b, r*) *arrival curve* is defined as $\alpha(t) = b + r \cdot t$ for $\forall t > 0$, where *b* is called burst tolerance, and *r* is the average data rate. The *rate-latency service curve* is defined as $\beta_{R,T}(t) = R \cdot (t-T)^+$, where $R \ge r$ is the guaranteed link bandwidth, *T* is the maximum latency of the service, and $(x)^+ = max(0, x)$. These curves lead to a fair trade-off between computing complexity and approximation accuracy of the real system behaviour.

Hereafter, we consider a data flow constrained by the (*b*, *r*) arrival curve $\alpha(t)$ and traversing system *S* with a rate-latency service curve $\beta_{R,T}(t)$. Then, the guaranteed performance bounds D_{max} and Q_{max} (see Figure 9 for additional intuition) are easily computed as:

$$D_{max} = \frac{b}{R} + T \qquad Q_{max} = b + r \cdot T \quad (2)$$

With Network Calculus, it is also possible to express an upper bound of the outgoing flow with output function $R^*(t)$, called *output bound*, as (the proof in [16]):

$$\alpha^*(t) = \alpha(t) \oslash \beta_{R,T}(t) = \alpha(t) + r \cdot T \ge \alpha(t) \quad (3)$$

where $\oslash$ *is the min-plus deconvolution defined as:*

$$(f \oslash g)(t) = \sup_{s \ge 0} \{f(t+s) - g(s)\}; \quad for\ \forall t \in R\ and\ f, g \in \boldsymbol{F}$$

Due to the accumulation of the data flows in the direction of the sink, the nodes offer a service curve $\beta(t)$ to this aggregated data flow. Thus, the delay and backlog bounds can be computed for the entire aggregate data flow at each node. Using the *aggregate scheduling* theorem, tighter bounds can be computed for individual flows traversing the network. In this paper, we use both approaches to compare the results.

Aggregate Scheduling. *Consider a node multiplexing two data flows, 1 and 2, in FIFO order. Assume that flow 2 is constrained by the (b, r) arrival curve $\alpha_2(t)$ and the node guarantees a service curve $\beta_{R,T}(t)$ to the aggregate of these two flows. Define the family of functions as:*

$$\beta_1(t, \theta) = (R - r_2)\left[t - \left(\frac{b_2 + r_2(T-\theta)}{R - r_2} + T\right)\right]^+ . 1_{\{t > \theta\}} \quad (4)$$

Then, for any $\theta \ge 0$, $\beta_1(t, \theta)$ is a service curve guaranteed for flow 1.

3. System model

This section defines the cluster-tree topology and data-flow models that will be considered in the analysis. It also elaborates on the worst-case cluster scheduling; that is, the time sequence of clusters' active periods leading to the worst-case end-to-end delay for a message to be routed to the sink.

3.1 Cluster-tree topology model

Cluster-tree WSNs feature a tree-based logical topology, where nodes are organized in different groups, called *clusters*. Each node is connected to one node at lower depth, called *parent node*, and can be connected to multiple nodes at upper depth, called *child nodes*.

Consider Figure 1. The cluster-tree topology contains two main types of nodes. First, the nodes that can associate with previously associated nodes and can participate in the multi-hop routing are referred to as *routers* (R_{ij}, i.e router *j* at depth *i*). Second, the leaf nodes that do not allow association of other nodes and do not participate in routing are referred to as *end-nodes* (*N*). The router that has no parent is called *root* (it might hold special functions such as identification, formation and control of the entire topology). Routers and end-nodes can both have sensing capabilities. Therefore they are generally referred to as *sensor nodes*. Each router forms its cluster and is referred to as *cluster-head* of this cluster.

In this paper we aim at specifying the worst-case cluster-tree topology, i.e. the network topology configuration that leads to the worst-case performance. This means that a dynamically changing cluster-tree WSN can assume different configurations, but it can never exceed the worst-case topology, in terms of maximum depth and number of child routers/end-nodes. Thus, the worst-case cluster-tree topology is graphically represented by a rooted balanced directed tree [15] defined by the following three parameters:

- $\boldsymbol{H}$**:** Height of the tree, i.e. the maximum number of logical hops from the deepest router to the root. A tree with only a root has a height of zero.
- $\boldsymbol{N_{end\_node}^{MAX}}$: Maximum number of end-nodes that can be associated to a router.
- $\boldsymbol{N_{router}^{MAX}}$: Maximum number of child routers that can be associated to a parent router.

The *depth* of a node is defined as the number of logical hops from that node to the root. The root is at depth zero, and the maximum depth of an end-node is *H*+1.

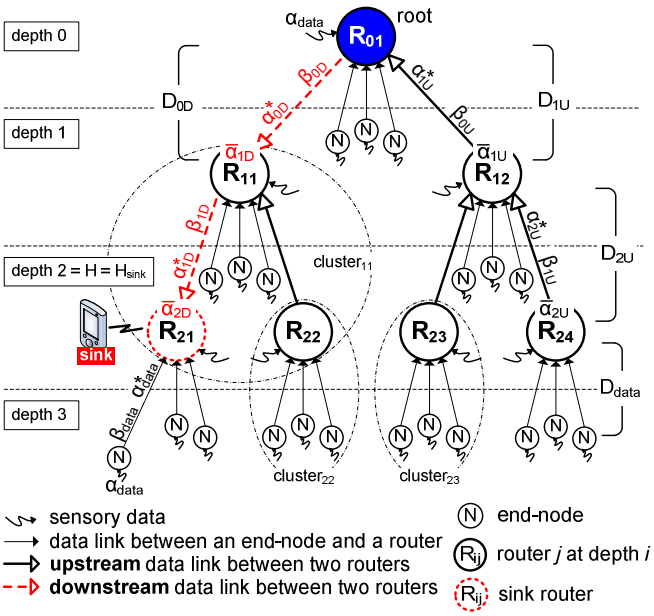

Figure 1. The cluster-tree topology and data-flow models.

Note that the *sink* is a special type of node that gathers the sensory data from all sensor nodes inside the network. Unlike previous work, we relax the assumption that the sink is only associated with the root and consider the sink to be an autonomous and topology-independent mobile node. The mobile behaviour means that a sink moves arbitrarily within a static cluster-tree WSN and can be associated with any router within communication range. The router, to which the sink is in a given moment associated, is referred to as *sink router*. There can be more than one mobile sink in a WSN, but we assume that only one is active (i.e. gathers the sensory data) at a given time. We specify another parameter, $H_{sink} \in (0, H)$, to represent the depth at a given moment of the sink router in a cluster-tree topology. Note that if the sink is associated with the root, i.e. $H_{sink} = 0$, the network contains only upstream flows. This case has already been analysed in [5]. In this paper, we analyze the case where $H_{sink} > 0$.

Our terminology and conventions are as illustrated in Figure 1, corresponding to a configuration where $H = 2$, $N_{end\_node}^{MAX} = 3$, $N_{router}^{MAX} = 2$, and $H_{sink} = 2$. Note that a cluster-tree WSN may contain additional nodes per router than those defined by N_{router}^{MAX} and $N_{end\_node}^{MAX}$ parameters. However, these additional nodes cannot be granted guaranteed resources.

3.2 Data-flow model

In this paper, we assume that all sensory data is exclusively sent to the sink. All sensor nodes are assumed to sense and transmit data upper bounded by the arrival curve $\alpha_{data}(t) = b_{data} + r_{data} \cdot t$. In case of different data flows, $\alpha_{data}(t)$ is considered to represent the upper bound of the highest flow in a network. This may introduce some pessimism to the analysis if the variance between data flows is significant.

Each end-node is granted a service guarantee from its parent router corresponding to the rate-latency service curve $\beta_{data}(t) = R_{data} \cdot (t - T_{data})^+$. By applying Eq. (3) to a flow constrained by the arrival curve $\alpha_{data}(t)$ and that is granted a service curve $\beta_{data}(t)$, we obtain the output arrival curve $\alpha_{data}^*(t)$, which upper bounds the outgoing data flow from any end-node:

$$\alpha_{data}^*(t) = \alpha_{data}(t) + r_{data} \cdot T_{data} \tag{5}$$

On the other hand, the amount of bandwidth allocated by each router depends on the cumulative amount of data at its inputs, which increases towards the sink. Thus, the total input function R of each router depends on the depth, and consists of the sum of the output functions R^* of its end-nodes and child routers. Additionally, the router itself can be equipped with sensing capability producing a traffic bounded by $\alpha_{data}(t)$. Thus, the arrival curve constraining the total input function R of a router at general depth i is expressed as:

$$\bar{\alpha}_i = \alpha_{data} + N_{end\_node}^{MAX} \cdot \alpha_{data}^* + \sum_{j=1}^{N_{router}^{MAX}} \alpha_{router(i+1,j)}^* \tag{6}$$

This result can then be used in Eq. (3). The outgoing flow of a router at depth i is upper bounded by the output arrival curve as follows:

$$\alpha_i^* = \bar{\alpha}_i \oslash \beta_{i-1} \tag{7}$$

Hence, the data-flow analysis consists in the computation of the arrival curves $\bar{\alpha}_i$ and α_i^*, using iteratively Eqs. (6) and (7), from the deepest routers until reaching the sink. After that, the resource requirements of each router, in terms of buffer requirement Q_i and bandwidth requirement R_i, and the worst-case end-to-end delay bound of WSN are computed.

In cluster-tree WSNs where the sink can be associated with a router other than the root, data flows may then be redirected in the downstream directions. Data flows over upstream links (called *upstream flows*) have already been analysed in [5]. Data flows over downstream links (called *downstream flows*), where data is sent from a parent router to its child router, are analysed in this paper. In what follows, the upstream and downstream flows are marked by the subscripts **U** and **D**, respectively (e.g. α_{iU}^*, α_{iD}^*). We also assume two types of service curves (i.e. β_{iU} for upstream flows and β_{iD} downstream flows) provided by each parent router at depth i to its child routers at depth i+1, and expressed as:

$$\beta_{iU}(t) = R_{iU} \cdot (t - T_{iU})^+ \qquad \beta_{iD}(t) = R_{iD} \cdot (t - T_{iD})^+ \tag{8}$$

To ensure the symmetry properties of the worst-case cluster-tree topology assumed in our methodology, the same downstream or upstream service curves must be guaranteed to all downstream or upstream flows at a given depth, respectively.

3.3 Time division cluster scheduling

In general, the radio channel is a shared communication medium where more than one node can transmit at the same time. In cluster-tree WSNs, messages are forwarded from cluster to cluster until reaching the sink. The time window of each cluster is periodically divided into an *active period* (AP), during which the cluster-head enables data transmissions inside its cluster, and a subsequent *inactive period*, during which all cluster nodes may enter low-power mode to save energy resources. To avoid collisions between multiple clusters, it is mandatory to schedule active periods of different clusters in an ordered sequence, called *Time Division Cluster Schedule* (TDCS). In other words, TDCS is equivalent to a permutation of active periods of all clusters in a WSN such that no inter-cluster interference occurs. In case of one collision domain (i.e. all nodes hear each other), the TDCS must be non-overlapping, i.e. only one cluster can be active at any time. On the contrary, in a network with multiple collision domains, the clusters from different non-overlapping collision domains may be active at the same time.

Due to the cumulative flow effect, the amount of traffic increases in the direction of the sink such that the maximum flow is reached in the cluster to which the sink is associated (e.g. cluster$_{11}$ in Figure 1). Hence, the duty cycles of the clusters closer to the sink should be higher than the ones of the clusters that are farther from the sink, to ensure efficient bandwidth utilization [18].

The TDCS significantly affects the resource requirements and delay bounds in cluster-tree WSNs. The number of feasible TDCSs in a network with n routers inside one collision domain is equal to the number of permutations, given by n factorial ($n!$). Note that for each data flow originated in a given node, there is a corresponding best-case/worst-case TDCS that minimizes/maximizes the end-to-end delay of that flow, respectively. Thus, it is impossible to determine a general best-case or worst-case TDCS meeting the requirements of all data flows. On one hand, the best-case TDCS of a data flow originated in node R_{24} (Figure 1), for example, comprises the consecutive sequence of active periods corresponding to the ordered sequence of the clusters traversed along the routing path from R_{24} to the sink. On the other hand, the worst-case TDCS comprises the same ordered sequence of active periods, but in the reverse order, which means starting from the sink backward to R_{24}. The active periods of other clusters, which are not on the routing path, are appended to the previously formed sequence in arbitrary order such that a complete TDCS is produced (see example in [17]). Using our methodology based on the symmetry properties of the cluster-tree model, the network resources of a WSN are dimensioned for the worst-case TDCS of a data flow originated in the end-node that is farthest from the sink (i.e., a flow along the longest path in a WSN).

To reduce the resource requirements of the routers, we introduce the following *priority rule*: "*When a router handles the links in different directions (e.g. R_{01} and R_{11} in Figure 1), the incoming flows via upstream data links are served before the outgoing flow via downstream data link.*" Using this rule, the end-to-end delay of an incoming data flow can be reduced to at most one TDCS cycle duration.

4. Input and output data flows analysis

In our model, we assume that the end-nodes have sensing capabilities, but the sensing capability of routers is optional. For an improved analysis, we introduce a binary variable S whose value is equal to 1 if routers have sensing capabilities; otherwise S is equal to 0.

The total input data flow of each router as shown in Eq. (6) comprises, among other terms, the sum of the output flows of its end-nodes and, optionally, its own sensory data flow constrained by $\alpha_{data}(t)$. This part of the total input flow is the same for upstream and downstream flows, hence we introduce the substitution:

$$\bar{\alpha}_H(t) = S \cdot \alpha_{data}(t) + N_{end\_node}^{MAX} \cdot \alpha_{data}^{*}(t)$$

Thus, using Eq. (5) we get:

$$\bar{\alpha}_H(t) = \left(N_{end\_node}^{MAX} + S\right) \cdot \alpha_{data}(t) + N_{end\_node}^{MAX} \cdot r_{data} \cdot T_{data} \quad (9)$$

where, $\bar{r}_H = \left(N_{end\_node}^{MAX} + S\right) \cdot r_{data}$ is the resulting *aggregate rate* of $\left(N_{end\_node}^{MAX} + S\right)$ input data flows, and $\bar{b}_H = \left(N_{end\_node}^{MAX} + S\right) \cdot b_{data} + N_{end\_node}^{MAX} \cdot r_{data} \cdot T_{data}$ is the *burst tolerance*. Note that $\bar{\alpha}_H(t)$ is also equal to the total input upstream flow of the deepest routers (at depth H).

4.1 Upstream data flows

In [5], the output and input upstream flows were analyzed and derived in detail. Thus, here we only summarize the final general recurrent expressions. The arrival curve, constraining the total input upstream flow of each router at depth i, is expressed as follows:

$$\bar{\alpha}_{iU} = \left(\sum_{j=0}^{H-i} (N_{router}^{MAX})^j\right) \cdot \bar{\alpha}_H + \sum_{j=1}^{H-i} \left((N_{router}^{MAX})^j \cdot \sigma_{i+j-1}\right) \quad (10)$$

for $\forall i$, $0 \le i \le H$, where $\sigma_n = \left(\sum_{k=0}^{H-(n+1)} (N_{router}^{MAX})^k\right) \cdot \bar{r}_H \cdot T_{nU}$

The output bound for the upstream data flow from each child router at depth i, receiving a service curve $\beta_{i-1}(t)$ from a parent router at depth i-1, is then expressed as:

$$\alpha_{iU}^{*} = \bar{\alpha}_{iU} + \sigma_{i-1} = \left(\sum_{j=0}^{H-i} (N_{router}^{MAX})^j\right) \cdot \bar{\alpha}_H + \sum_{j=0}^{H-i} \left((N_{router}^{MAX})^j \cdot \sigma_{i+j-1}\right) \quad (11)$$

for $\forall i$, $0 < i \le H$

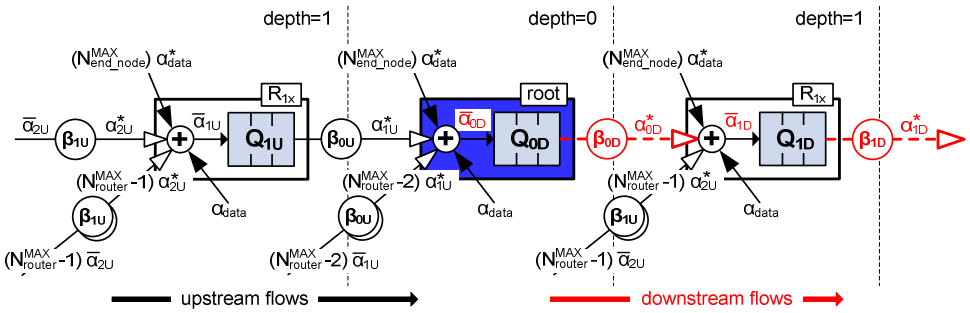

Figure 2. The queuing system model for upstream and downstream flows.

4.2 Downstream data flows

In [17], we derive the arrival curve of the total input downstream flow $\bar{\alpha}_{iD}$ and the upper bound of the output downstream flow α^*_{iD} depth by depth, using the Network Calculus methodology, starting from depth 0 (i.e. the root). In our analysis, we consider the queuing model in Figure 2. For the sake of space, here we only summarize the final general recurrent expressions.

The arrival curve constraining the total input downstream flow of a router at a given depth i, for $i = 0, \cdots\cdots, (H_{sink}\text{-}1)$, is expressed as:

$$\bar{\alpha}_{iD} = \left(\sum_{j=0}^{i}(N_{router}^{MAX})^{H-j}\right)\cdot\bar{\alpha}_H + (N_{router}^{MAX}-1)\cdot\sum_{j=0}^{i}\delta_j + \sum_{j=0}^{i-1}\tau_j \quad (12)$$

for $\forall i,\ 0 \le i < H_{sink}$, where

$$\delta_n = \sum_{k=0}^{H-(n+1)}((N_{router}^{MAX})^k \cdot \sigma_{k+n})$$

$$\sigma_n = \left(\sum_{k=0}^{H-(n+1)}(N_{router}^{MAX})^k\right)\cdot\bar{r}_H\cdot T_{nU}$$

$$\tau_n = \left(\sum_{k=0}^{n}(N_{router}^{MAX})^{H-k}\right)\cdot\bar{r}_H\cdot T_{nD}$$

The upper bound of the output downstream flow from a parent router at depth i, providing a service curve $\beta_{iD}(t)$, towards its child router at depth i+1 is expressed as:

$$\alpha^*_{iD} = \bar{\alpha}_{iD} + \tau_i = \quad (13)$$

$$\left(\sum_{j=0}^{i}(N_{router}^{MAX})^{H-j}\right)\cdot\bar{\alpha}_H + (N_{router}^{MAX}-1)\cdot\sum_{j=0}^{i}\delta_j + \sum_{j=0}^{i}\tau_j$$

for $\forall i,\ 0 \le i < \mathrm{H_{sink}}$.

Note that the sink can be associated to the router at a depth lower than the height of the cluster-tree, i.e. $H_{sink} < H$ (Figure 3.a) or equal to the height of the cluster-tree, i.e. $H_{sink} = H$ (Figure 3.b).

In the case of $H_{sink} < H$, the arrival curve constraining the total input downstream flow is expressed as:

$$\bar{\alpha}_{(H_{sink})D} = \bar{\alpha}_H + N_{router}^{MAX}\cdot\alpha^*_{(H_{sink}+1)U} + \alpha^*_{(H_{sink}-1)D} \quad (14)$$

If $H_{sink} = H$, the arrival curve constraining the total input downstream flow is expressed as:

$$\bar{\alpha}_{(H_{sink})D} = \bar{\alpha}_H + \alpha^*_{(H_{sink}-1)D} \quad (15)$$

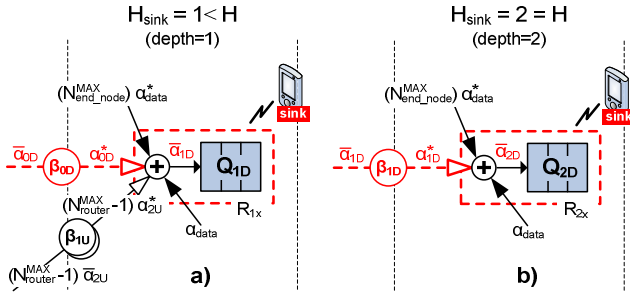

Figure 3. Possible locations of a sink router and correspondent data flows.

5. Worst-case network dimensioning

Supporting time-sensitive WSN applications implies to predict and guarantee maximum end-to-end communication delays. To ensure bounded end-to-end delays and to avoid buffer overflow, network resources must be known in advance, and dimensioned along the path from a source to a sink.

5.1 Per-router resources analysis

We aim at specifying the minimum bandwidth of each downstream data links and the minimum buffer size at each downstream router needed to store the bulk of data incoming through the router's inputs.

Bandwidth requirements

Consider a parent router at depth i providing a service curve $\beta_{iD}(t)$ to its child router at depth i+1 (see Figure 1). The total input downstream flow of the parent router is constrained by the arrival curve $\bar{\alpha}_{iD}(t)$ and dispatched through a downstream link to its child router. Thus, to ensure a bounded delay, the guaranteed amount of bandwidth R_{iD} must be greater than or equal to the arrival rate of total input downstream flow $\bar{r}_{iD}$. As a result, by applying Eqs. (12) and (9) we obtain:

$$R_{iD} \geq \bar{r}_{iD} = r^*_{iD} = \left(\sum_{j=0}^{i} (N^{MAX}_{router})^{H-j}\right) \cdot \bar{r}_H = \left(\sum_{j=0}^{i} (N^{MAX}_{router})^{H-j}\right) \cdot \left(N^{MAX}_{end\_node} + S\right) \cdot r_{data} \quad (16)$$

for $\forall i, 0 \leq i < H_{sink}$.

Note that it is possible to determine the *total number of routers* in a network using Eq. (16) by having $i = H$ and $\bar{r}_H = 1$, which is expressed as:

$$\Sigma(N^{MAX}_{router}, H) = \sum_{j=0}^{H} (N^{MAX}_{router})^{H-j} \quad (17)$$

Buffer requirements

To avoid buffer overflow, the buffer of a downstream router at depth i must be able to store all incoming data, constrained by the arrival curve $\bar{\alpha}_{iD}(t)$, until it is dispatched through the downstream link to a child router at depth i+1. The required buffer size Q_{iD} of the downstream router at depth i must be at least equal to the burst tolerance b^*_{iD} of the output bound $\alpha^*_{iD}(t)$ (see Figure 9). Hence, according to Eq. (13) we get:

$$Q_{iD} = b^*_{iD} = b^{*\ BURST}_{iD} + b^{*\ UP\_LAT}_{iD} + b^{*\ DOWN\_LAT}_{iD} = \left(\sum_{j=0}^{i} (N^{MAX}_{router})^{H-j}\right) \cdot \bar{b}_H + (N^{MAX}_{router} - 1) \cdot \sum_{j=0}^{i} \delta_j + \sum_{j=0}^{i} \tau_j \quad (18)$$

for $\forall i, 0 \leq i < H_{sink}$.

Observe that the required buffer size is the sum of three terms. The first term is the sum of burst tolerances b_{data} of the sensory data flows of all sensor nodes inside all sub-trees of a given router. The second and third terms represent the cumulative effect of the service latency at each depth for upstream and downstream flows, respectively.

In case of a sink router at depth H_{sink}, the buffer requirement must be greater than or equal to the burst tolerance $\bar{b}_{(H_{sink})D}$ of total input flow $\bar{\alpha}_{(H_{sink})D}$ given by Eq. (14) or Eq. (15).

5.2 End-to-end delay analysis

The *worst-case end-to-end delay* is the delay bound of a data flow transmitted along the longest path in the network. There are two approaches to compute this queuing delay.

Per-hop end-to-end delay

The first approach consists in computing the per-hop delay bounds of the aggregate input flows, and then deducing the end-to-end delay bound as the sum of per-hop delays. According to Eq. (2), the delay bound between a parent router at depth i, which offers service curve $\beta_{iD}(t)$ to its total input downstream flow constrained by arrival curve $\bar{\alpha}_{iD}(t)$, and its child router at depth i+1 is expressed as: $D_{iD} = \bar{b}_{iD}/R_{iD} + T_{iD}$. In case of the upstream flow, the delay bound between a child router at depth i and its parent router at depth i-1 offering service curve $\beta_{(i-1)U}(t)$ has been derived in [5], and is expressed as: $D_{iU} = \bar{b}_{iU}/R_{(i-1)U} + T_{(i-1)U}$.

Hence, the maximum end-to-end delay is the sum of all per-hop delay bounds as follows:

$$D^{MAX}_{e2e} = D_{data} + \sum_{i=1}^{H} D_{iU} + \sum_{i=0}^{H_{sink}-1} D_{iD} \quad (19)$$

where $D_{data} = b_{data}/R_{data} + T_{data}$ is the delay bound between an end-node and its parent router.

This approach is a bit pessimistic, since the delay bound at each router is computed for the aggregation of input flows. Tighter end-to-end delay bounds can be computed for individual flows, as follows.

Per-flow end-to-end delay

The idea of this approach is to derive the service curves offered to an individual flow F by the routers along the path, using the aggregate scheduling theorem in Eq. (4), and then deduce the network-wide service curve for flow F based on the concatenation theorem. Finally, according to Eq. (2), the end-to-end delay bound of a given flow F will be computed using the network-wide service curve applied to the arrival curve of the input flow. The maximum end-to-end delay is equal to the delay bound of a data flow along the longest path in the network. The complete algorithm has been presented in [5], and it is valid for upstream as well as for downstream flows.

6. Application to IEEE 802.15.4/ZigBee

So far, we have presented the general methodology for providing timeliness guarantees in cluster-tree WSNs with mobile sink behaviour independently of any specific protocol. In this section, we show how to apply the aforementioned methodology to the specific case of IEEE 802.15.4/ZigBee cluster-tree WSNs.

6.1 IEEE 802.15.4/ZigBee protocols features

The IEEE 802.15.4/ZigBee [6, 7] protocols have several appealing properties for WSNs. The MAC layer supports the beacon-enabled or non beacon-enabled modes. We only consider the beacon-enabled mode, since it has ability to provide timeliness guarantees by using the *Guaranteed Time Slot* (GTS) mechanism.

In beacon-enabled mode, beacon frames are periodically sent by a central node, called PAN coordinator, to synchronize nodes that are associated with it and to describe the structure of the superframe (Figure 4). The superframe, corresponding to the *Beacon Interval* (*BI*), is defined by the time between two consecutive beacons, and includes an active period and, optionally, a following inactive period. The active period, corresponding to the *Superframe Duration* (*SD*), is divided into 16 equally-sized time slots. Each active period can be

further divided into a *Contention Access Period* (CAP) and an optional *Contention Free Period* (CFP). Within the CFP, Guaranteed Time Slots (GTSs) can be allocated to a set of child nodes. The CFP supports up to 7 GTSs and each GTS may contain multiple time slots. Each GTS can transfer data either in *transmit direction*, i.e. from child to parent (upstream flow), or *receive direction*, i.e. from parent to child (downstream flow).

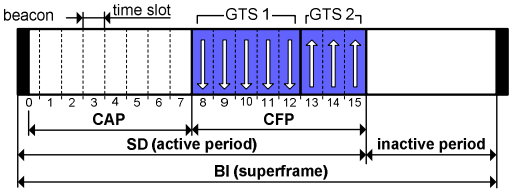

Figure 4. IEEE 802.15.4 superframe structure.

The structure of the superframe is defined by two parameters, the *Beacon Order* (*BO*) and the *Superframe Order* (*SO*), as follows:

$$BI = aBaseSuperframeDuration \cdot 2^{BO}$$
$$SD = aBaseSuperframeDuration \cdot 2^{SO}$$

where *aBaseSuperframeDuration* = 15.36 ms assuming the 2.4 GHz ISM frequency band with 250 kbps data rate, and $0 \le SO \le BO \le 14$.

While IEEE 802.15.4 in beacon-enabled mode supports only star-based topologies, ZigBee standard has proposed its extension to cluster-tree based topologies. Note that each cluster is active during its *SD*. To avoid the collisions between multiple superframe durations, the appropriate scheduling of SDs must be used (Section 3.3). For the sake of simplicity, we assume that all clusters have the same duty cycle, and whole WSN is inside one collision domain. Hence, the TDCS is given by the non-overlapping sequence of equally-sized SDs (Figure 5), and the duration of a TDCS cycle is equal to *BI*.

6.2 Guaranteed bandwidth of a GTS time slot

The whole data transmission in a GTS, including the frame, inter-frame spacing (IFS) and potential acknowledgment, must be completed before the end of the GTS. The maximum time required for the whole transmission of a MAC frame, called MPDU (MAC Protocol Data Unit) is then expressed as:

$$T_{MPDU} = MPDU_{max}/C + IFS + AckWaitDuration \cdot \Omega$$

where $MPDU_{max}$ is the user defined maximum size of the frame, C is the data rate (we assume 250 kbps), and $\Omega = 1$ for an acknowledged transmission or $\Omega = 0$ for an unacknowledged transmission. The maximum number of MAC frames that can be transmitted during one time slot is expressed as:

$$N_{MPDU} = \left\lfloor \frac{TS}{T_{MPDU}} \right\rfloor$$

where *TS* is the duration of a time slot and is equal to SD/16. In the remaining time, a frame smaller than $MPDU_{max}$ can be transmitted if the whole transmission can be completed before the end of the GTS. The transmission time of last frame is then expressed as:

$$T_{last} = TS - N_{MPDU} \cdot T_{MPDU} - IFS - AckWaitDuration \cdot \Omega$$

Finally, assuming a full duty cycle (i.e. *SO* = *BO*) the guaranteed bandwidth of one GTS time slot is expressed as:

$$R_{TS}^{100\%} = \frac{N_{MPDU} \cdot MPDU_{max} + max\,(T_{last}, 0) \cdot C}{SD} \qquad (20)$$

For more details, interested readers are referred to [17].

6.3 Characterization of the service curve

Each parent router must reserve a GTS with enough time slots for each of its child nodes (requiring guaranteed service). For downstream data link, a parent router at depth i must reserve a GTS with N_{iD}^{TS} time slots in receive direction to its child router at depth i+1 such that the resulting link bandwidth is greater than or equal to its total input arrival rate $\bar{r}_{iD}$. It results that:

$$N_{iD}^{TS} = \left\lceil \frac{\bar{r}_{iD}}{R_{TS}} \right\rceil \qquad (21)$$

Hence, a GTS with N_{iD}^{TS} time slots provides rate-latency service $\beta_{R_i T_i}(t)$, where $R_i = N_{iD}^{TS} \cdot R_{TS}$ is the guaranteed bandwidth and T_i is the service latency.

The service latencies depend on the TDCS such that their worst-case values are achieved for the worst-case TDCS of a data flow along the longest path in a WSN. Let us consider the example in Figure 1, where an end-node of router R_{24} sends sensory data to the sink associated with the router R_{21} (i.e. a flow along the longest routing path). Thus, the corresponding worst-case TDCS may be given by the following sequence of superframe durations: SD_{11}, SD_{01}, SD_{12}, SD_{24}, SD_{23}, SD_{21}, SD_{22}. The worst-case service latencies at each depth, except depth 0, are given by the distance between the active periods of consecutive clusters on the longest routing path to the sink.

According to Figure 5, the worst-case service latency guaranteed to a flow over *downstream data link* at given depth is expressed as:

- the service latency guaranteed by a router at depth 0 to the child router at depth 1 (priority rule, Section 3.3):

$$T_{0D} = (N_{router}^{MAX} - 1) \cdot N_{0U}^{TS} \cdot TS$$

- the service latency guaranteed by a router at depth i to the child router at depth i+1, for $\forall i$, $0 < i < H_{sink}$:

$$T_{iD} = BI - SD - \left(N_{iD}^{TS} - N_{(i-1)D}^{TS}\right) \cdot TS$$

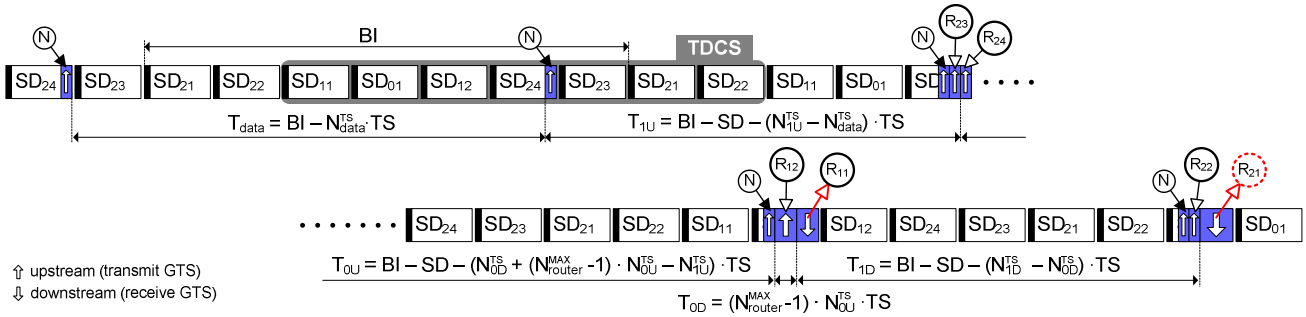

Figure 5. The worst-case service latencies for a flow along the longest path in the WSN related to the example in Figure 1.

6.4 IEEE 802.15.4/ZigBee WSN setup

For our experimental scenario, we consider a simple cluster-tree WSN corresponding to the configuration where $H = 2$, $N_{end\_node}^{MAX} = 1$, $N_{router}^{MAX} = 2$. For the sake of simplicity, only end-nodes are equipped with sensing capability (i.e. $S = 0$) and generate data flows bounded by the arrival curve $\alpha_{data}(t)$. We assume a minimum possible value of SO (e.g. $SO = 4$), imposed by some technological limitations [25], namely due to the non-preemptive behaviour of the TinyOS [21] operating system. According to Eq. (17), the total number of routers is equal to 7. Hence, BO must be set such that at least 7 SDs with $SO = 4$ can fit inside the BI without overlapping. In general, we obtain:

$$BI \geq \Sigma(N_{router}^{MAX}, H) \cdot SD \Leftrightarrow$$

$$BO_{min} = \lceil log_2(\Sigma(N_{router}^{MAX}, H) \cdot 2^{SO}) \rceil$$

As a result for $SO = 4$, the minimum BO is equal to 7, such that a maximum of $2^7/2^4 = 8$ SDs can fit in one BI. The maximum duty cycle of each cluster is then equal to $(1/8) = 12.5$ %. Note that to maximize the lifetime of a WSN, the lowest duty cycles must be chosen. On the other hand, low duty cycles enlarge end-to-end delays. Hence, long lifetime is in contrast to the fast timing response of a WSN, so a trade-off must be found.

According to [6], the minimum CAP is equal to 7.04 ms, assuming the 2.4 GHz ISM band, which corresponds to 1 time slot with $SO = 4$. The remaining slots can be allocated for GTSs. Hence, the maximum CFP length is equal to $L_{CFP} = 15$ time slots. A router cannot reserve more than L_{CFP} time slots for 7 GTSs maximum, i.e. for its $N_{end\_node}^{MAX}$ end-nodes and N_{router}^{MAX} child routers. Assuming that each end-node requires allocation of a GTS with N_{data}^{TS} time slots (i.e. $r_{data} \leq N_{data}^{TS} \cdot R_{TS}$) from its parent router, then each child router can allocate a GTS with the maximum number of time slots equal to:

$$\lfloor (L_{CFP} - N_{data}^{TS} \cdot N_{end\_node}^{MAX}) / N_{router}^{MAX} \rfloor$$

According to Eq. (16), the arrival rate r_{data} must not exceed the maximum bandwidth that a parent router can reserve. Obviously, due to the cumulative flow effect, the maximum bandwidth will be required by the sink router. Hence, the corresponding link bandwidth guaranteed by the parent router at depth $H_{sink} - 1$ to the sink router at depth H_{sink} is equal to:

$$R_{(H_{sink}-1)} = \left\lfloor \frac{L_{CFP} - N_{data}^{TS} \cdot N_{end\_node}^{MAX}}{N_{router}^{MAX}} \right\rfloor \cdot R_{TS} \qquad (22)$$

As a result applying Eq. (16), we obtain the maximum arrival rate of the sensory data flow as:

$$r_{data}^{MAX} = \left(\frac{\left\lfloor \frac{L_{CFP} - N_{data}^{TS} \cdot N_{end\_node}^{MAX}}{N_{router}^{MAX}} \right\rfloor \cdot R_{TS}}{\left(\sum_{j=0}^{(H_{sink}-1)} (N_{router}^{MAX})^{H-j} \right) \cdot \left(N_{end\_node}^{MAX} + S_{ij} \right)} \right) \qquad (23)$$

Note that the aforementioned expressions are valid for $\forall H_{sink}, 1 \leq H_{sink} \leq H$. The expressions for $H_{sink} = 0$ have been already derived in [5]. The value of burst tolerance b_{data} is selected according to the burstiness of sensory data.

7. Experimental evaluation

In this section, we compare the worst-case theoretical results based on Network Calculus with the experimental results obtained through a test-bed based on IEEE 802.15.4/ZigBee technologies. The analytical results are computed using a MATLAB model [19], and the experimental results are obtained using a real test-bed based on the TelosB motes [20].

7.1 Experimental setup

The experimental test-bed (illustrated in Figure 6) consists of 14 TelosB motes running the TinyOS 1.x [21] operating system with our open source implementation of the IEEE 802.15.4/ZigBee protocol stack [22]. For debugging purposes, we have used the Chipcon CC2420 packet sniffer [23] that provides a raw list of the transmitted packets, and the Daintree Sensor Network Analyzer (SNA) [24] that provides additional functionalities, such as the graphical topology of the network.

We configured the application running on the sensor nodes to generate 3 bytes at the data payload. Hence, the maximum size of the MAC frame is equal to $MPDU_{max} = 192$ bits (for details see [17]).

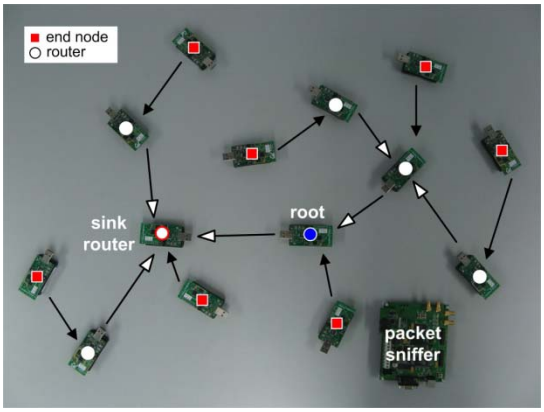

Figure 6. The test-bed deployment for H_{sink} =1.

TinyOS 1.x flushes the reception buffer of the radio transceiver after processing the first arriving frame. Thus, the frames that arrive during the processing time of the first frame are discarded. This problem has been already reported and fixed in TinyOS 2.x. Since our implementation of IEEE 802.15.4/ZigBee protocol stack was built over TinyOS 1.x, we overcame the aforementioned problem by setting the inter-frame spacing (IFS) time (i.e. time between two consecutive frames) such that no frame arrives during the frame processing times. The experimental value of IFS equal to 3.07 ms was measured.

According to Eq. (20), the bandwidth guaranteed by one time slot for $SO = 4$ is equal to 3.125 kbps with 100% duty cycle. Hence, in our experimental scenario with a 12.5 % duty cycle (i.e. $BO = BO_{min} = 7$), the guaranteed bandwidth of one time slot is equal to $R_{TS} = 3.125 \cdot 0.125 = 0.3906$ kbps.

Let us assume $N_{data}^{TS} = 1$. Then according to Eq. (23), we obtain the maximum arrival rates of the sensory data flow as follows:

- $r_{data}^{MAX} = 456$ bps for $H_{sink} = 2$
- $r_{data}^{MAX} = 684$ bps for $H_{sink} = 1$
- $r_{data}^{MAX} = 911$ bps for $H_{sink} = 0$ (root)

As a result of $r_{data} \leq \min(r_{data}^{MAX})$ and $r_{data} \leq R_{TS}$, we consider an average arrival rate equal to $r_{data} = 390$ bps, which corresponds to 4 frames (192 bits each) generated during one Beacon Interval ($BI = 1.96608$ sec). We assume that the burst tolerance is equal to $b_{data} = 576$ bits, which corresponds to 3 frames generated at once. Hence, each sensory data flow is bounded by arrival curve $\alpha_{data}(t) = 576 + 390 \cdot t$. The frames can be generated as constant bitrate (CBR) or variable bitrate (VBR) traffic upper bounded by the arrival curve $\alpha_{data}(t)$ (Figure 7).

Finally, let us summarize the complete network setting:

- $N_{router}^{MAX} = 2$
- $N_{end\_node}^{MAX} = 1$
- $H = 2$
- $SO = 4$ ($SD = 245.76$ ms)
- $BO = 7$ ($BI = 1966.08$ ms)
- Duty Cycle = 12.5 %
- $MPDU_{max} = 192$ bits
- $r_{data} = 390$ bits
- $b_{data} = 576$ bits
- IFS = 3.07 ms
- $L_{CFP} = 15$
- $S = 0$

We assume the worst-case TDCS of a flow along the longest routing path from router R_{24} to the sink (Figure 1) given by the following sequence of superframe durations: SD_{11}, SD_{01}, SD_{12}, SD_{24}, SD_{23}, SD_{21}, SD_{22}. Note that we consider only unacknowledged transmissions.

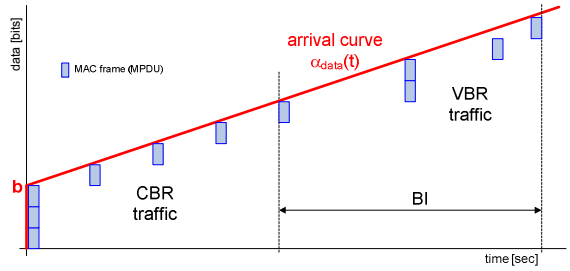

Figure 7. The sensory traffic generation.

7.2 Experimental vs. theoretical results

Buffer requirements

Figure 8 shows the theoretical worst-case buffer requirements as compared to the maximum values obtained through real experimentation, for $H_{sink} = 2$.

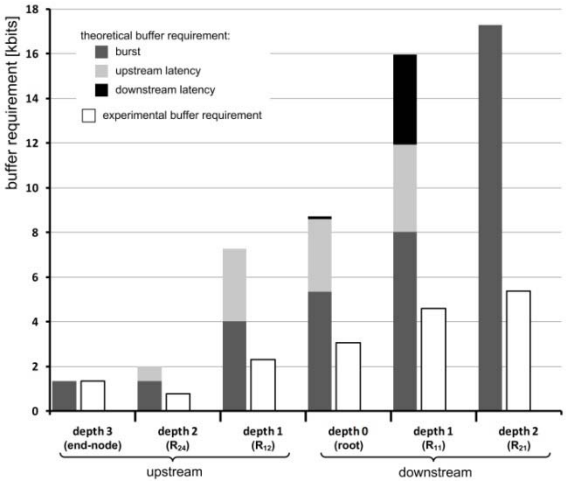

Figure 8. Buffer requirements.

First, the theoretical buffer requirements are divided into three parcels according to their origin, as already shown in Section 4.2. Observe that the cumulative effect of the burst is more important than the cumulative effect of the service latencies. The effect of the service latencies may be more important for other setting of b_{data} and r_{data}. So, the different settings of the sensory arrival curve affect the buffer requirements. The minor effect of the upstream service latency at depth 0 is given by the priority rules (Section 3.3), such that the data arriving during the transmit GTS (i.e. upstream flow) are stored in the root until the receive GTS (i.e. downstream flow), at the end of the same SD, is active and data is dispatched (Figure 5).

The next observation confirms that the theoretical values upper bound the experimental values. The pessimism of the theoretical bounds is justified by the fact that the Network Calculus analytical model is based on a continuous approach (arrival and service curves are

continuous) in contrast to the real stepwise behaviour of flows and services in the test-bed. In practice, the data is actually transmitted only during its GTS, while in the analytical model we consider a continuous data flow during the whole *BI*, since it represents the average rate and not the instantaneous rate. Figure 9 illustrates the problem and shows the arrival and service curves of a data flow sent by an end-node to its parent router. The burst of the outgoing data flow b^*_{data} (Eq. (5)) is equal to Q^{TH}_{max}, in case of the analytical model, or Q^{EXP}_{max}, in the experimental case. Due to the cumulative flow effect, the discrepancy between theoretical (Q^{TH}_{max}) and experimental (Q^{EXP}_{max}) values of buffer requirement increases with depth. The rate-latency service curve used in our analysis results from a trade-off between computing complexity and pessimism.

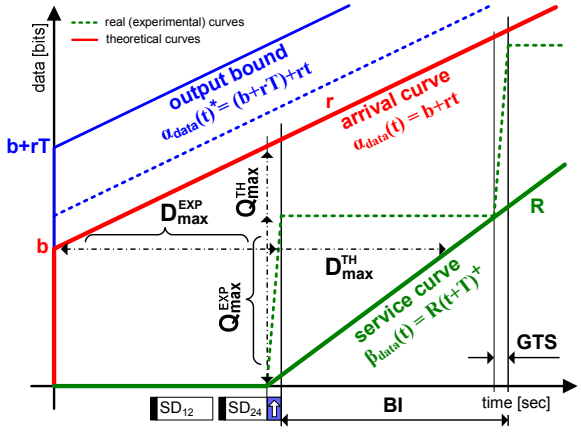

Figure 9. Theoretical vs. experimental data traffic.

The numerical values of theoretical worst-case as well as experimental maximum buffer requirements are summarized in Table 1. The bandwidth requirements given by Eq. (16) and the corresponding number of time slots are also presented. In Tables 1 and 2, *U* means upstream router at depth *i* or upstream link to a router at depth *i*, and *D* means downstream router or downstream link from a router at depth *i*.

Table 1
Buffer Requirements: Theoretical vs. Experimental Results

	depth		theoretical results (worst-case values)			experimental results (maximum values)
			R_i [kbps]	N_i^{TS}	Q_i [kbits]	Q_i [kbits]
$H_{sink} = 0$ (root)	0	U	1.7	3	**15.995**	5.376
	1	U	0.39	1	7.329	2.304
	2	U	—	—	2.008	0.768
$H_{sink} = 1$	0	D	1.56	4	8.667	3.072
		U	1.17	3	—	—
	1	D	—	—	14.02	5.376
		U	0.39	1	7.257	2.304
	2	U	—	—	2.008	0.768
$H_{sink} = 2$	0	D	1.56	4	8.667	3.072
		U	1.17	3	—	—
	1	D	2.34	6	**15.966**	4.608
		U	0.39	1	7.257	2.304
	2	D	—	—	**17.3**	5.376
		U	—	—	2.008	0.768
end-node			0.39	1	1.337	1.344

Observe in Table 1 that end-nodes have the smallest buffer requirement as they are the leaves of the tree, and that the buffer requirement grows in direction of the sink router. Since the sink can be associate with any router and in order to avoid buffer overflow, all routers at depth *i* should allocate a buffer of capacity greater or equal to the worst-case buffer requirement at given depth *i* (e.g. all router at depth 0 allocate a buffer of capacity equal to 15.995 kbits), which effectively demonstrates how these analytical results can be used by a system designer.

Delay bounds

In Figure 10, we compare the worst-case, maximum and average values of per-hop delay bounds in each router, and the end-to-end delay bounds for $H_{sink} = 2$. A first observation confirms that theoretical values upper bound the experimental values. The difference between theoretical worst-case (D^{TH}_{max}) and experimental maximum (D^{EXP}_{max}) delays (Figure 9) is due to the aforementioned continuous and stepwise behaviours of the analytical model and test-bed, respectively. The experimental delays comprise mainly the service latencies (Figure 9) decreasing in the direction of the sink (Figure 5). Hence, the maximum per-hop delays also decrease in the direction of the sink, as can be observed in Figure 10. The low downstream delay at depth 0 results from the priority rule (Section 3.3). The end-to-end delays bounds are quite high, even though the b_{data} and r_{data} are low. This is mainly due to high value of $SO = 4$ (i.e. $BI = 1.966$ sec). Hence, the end-to-end delay bounds can be reduced using lower values of *SO* or higher bandwidth guarantees, using lower IFS, for example.

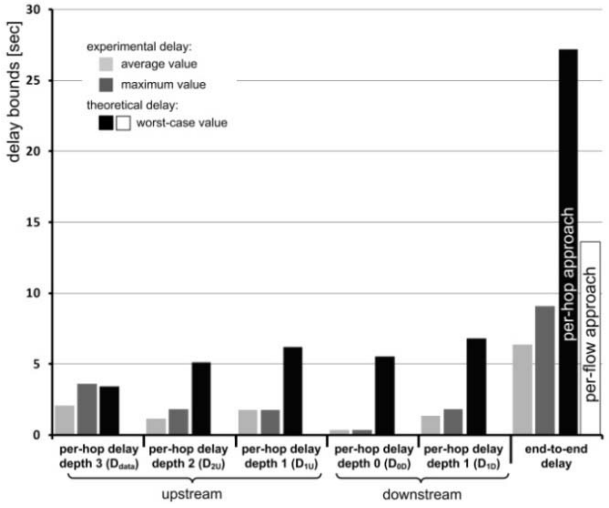

Figure 10. Delay bounds.

Observe also that the worst-case end-to-end delay obtained by the per-flow approach offers less pessimism than the delay from the per-hop approach.

Table 2 presents the worst-case, maximum and average numerical values of per-hop and per-flow delay bounds, and the end-to-end delays for given sink positions. Note that the average values were computed from the set of 15

measurements, involving 1155 frames each. The theoretical worst-case end-to-end delays are obtained as the sum of per-hop delays using Eq. (19) (first term), or by per-flow approach (Section 5.2), which results in the family of service curves as a function of $\theta \geq 0$. In our analysis we assume $\theta = T + (b_2/R)$ as a trade-off between computation complexity and optimality. The determination of the optimal service curve, leading to the lowest worst-case delay, will be addressed in future work.

Table 2
Delay Bounds: Theoretical vs. Experimental Results

	depth	theoretical results (worst-case values)	experimental results	
			maximum	average
		D_i [sec]	D_i [sec]	D_i [sec]
$H_{sink} = 0$ (root)	1 U	6.257	1.764	1.308
	2 U	5.143	1.812	1.602
	D_{e2e}	14.82/**9.69**	**7.154**	4.952
$H_{sink} = 1$	0 D	5.547	0.104	0.099
	1 U	6.195	1.76	1.728
	2 U	5.143	1.809	1.602
	D_{e2e}	20.31/**10.53**	**7.251**	5.471
$H_{sink} = 2$	0 D	5.547	0.104	0.099
	1 D	6.814	1.812	1.321
	1 U	6.195	1.766	1.728
	2 U	5.143	1.814	1.135
	D_{e2e}	27.13/**13.65**	**9.074**	6.325
end-node (D_{data})		3.425	3.578	2.042

8. Conclusions and future work

In this paper, we tackled the worst-case dimensioning of cluster-tree wireless sensor networks (WSN) assuming that the data sink can be mobile, i.e. can be associated to any router in the sensor network. We provided a system model, an analytical methodology and a software tool that enables system designers to dimension and analyze these networks assuming an error-free channel. In this way, it is possible to guarantee the routers' buffer size to avoid buffer overflows and to minimize each cluster's duty cycle (maximizing nodes' lifetime) still satisfying that messages' deadlines are met.

Importantly, we showed how it is possible to instantiate our generic methodology in IEEE 802.15.4/ZigBee, which are promising technologies for WSN applications. We also developed a 7 clusters test-bed based on Commercial-Off-The-Shelf technologies, namely TelosB motes [20] running our open-ZB protocol stack [22] over TinyOS [21]. This test-bed enabled us to assess the pessimism of our worst-case theoretical results (buffer requirements and message end-to-end delays), by comparing these to the maximum and average values measured in the experiments.

Ongoing and future works include improving the current methodology to encompass clusters operating at different duty-cycles and to provide a model that enables real-time control actions, i.e. the sink assuming the role of controlling sensor/actuator nodes.

References

[1] Z. Hu and B. Li, "Fundamental Performance Limits of Wireless Sensor Networks," *Ad Hoc and Sensor Networks*, Nova Science Publishers, pp. 81-101, ISBN 1-59454-396-8, Hardcover, 2005.

[2] T. F. Abdelzaher, S. Prabh, R. Kiran, "On real-time capacity limits of multihop wireless sensor network," In IEEE Real-Time Systems Symposium (RTSS'04), Portugal, Dec. 2004.

[3] J. Gibson, G. G. Xie, Y. Xiao, "Performance Limits of Fair-Access in Sensor Networks with Linear and Selected Grid Topologies, " In GLOBECOM Ad Hoc and Sensor Networking Symposium, Washington DC, Nov. 2007

[4] S. Prabh, T. F. Abdelzaher, "On Scheduling and Real-Time Capacity of Hexagonal Wireless Sensor Networks, "In Euromicro Conference on Real-Time Systems (ECRTS'07), Italy, July 2007.

[5] A. Koubaa, M. Alves, E. Tovar, "Modeling and Worst-Case Dimensioning of Cluster-Tree Wireless Sensor Networks," In Real Time Systems Symposium (RTSS'06), Brazil, Dec. 2006.

[6] IEEE 802.15.4 Standard-2003, "Part 15.4: Wireless Medium Access Control and Physical Layer Specifications for Low Rate Wireless Personal Area Networks," IEEE SA Standards Board.

[7] Zigbee Alliance, "ZigBee Specification," v. 1.0, April 2005.

[8] A. Koubaa, M. Alves, E. Tovar, "IEEE 802.15.4: a Federating Communication Protocol for Time-Sensitive Wireless Sensor Networks," Chapter of the book *Sensor Networks and Configurations: Fundamentals, Techniques, Platforms, and Experiments*, Springer-Verlag, Germany, pp. 19-49, Jan. 2007.

[9] J-Y. Le Boudec and P. Thiran, "A Theory of Deterministic Queuing Systems for the Internet," In Lecture Notes in Computer Science (LNCS), Vol. 2050, May 2004.

[10] J. B. Schmitt and U. Roedig, "Sensor Network Calculus - A Framework for Worst Case Analysis," In IEEE/ACM Conference on Distributed Computing in Sensor Systems (DCOSS'05), USA, June 2005.

[11] J. B. Schmitt, F. Zdarsky, L. Thiele, "A Comprehensive Worst-Case Calculus for Wireless Sensor Networks with In-Network Processing," In IEEE Real-Time Systems Symposium (RTSS'07), USA, Dec. 2007.

[12] J. B. Schmitt and U. Roedig, "Worst Case Dimensioning of Wireless Sensor Networks under Uncertain Topologies," In Workshop on Resource Allocation in Wireless NETworks (RAWNET'05), Italy, April 2005.

[13] S. R. Gandham, M. Dawande et al. "Energy efficient schemes for wireless sensor networks with multiple mobile base stations," In IEEE GLOBECOM, USA, Dec. 2003.

[14] W. Y. Poe and J. B. Schmitt, "Minimizing the Maximum Delay in Wireless Sensor Networks by Intelligent Sink Placement," Technical Report 362/07, U. Kaiserslautern, Germany, July 2007.

[15] R. Diestel, "Graph Theory", Springer-Verlag, ISBN 0-387-95014-1, Hardcover, 2000.

[16] A. Koubaa, M. Alves, E. Tovar, "Modeling and Worst-Case Dimensioning of Cluster-Tree Wireless Sensor Networks: proofs and computation details," IPP-HURRAY!, TR-060601, June 2006.

[17] P. Jurcik, R. Severino, A. Koubaa, M. Alves, E. Tovar, "Worst-case Dimensioning of Cluster-Tree Sensor Networks with Mobile Sink Behaviour," Technical Report IPP-HURRAY!,TR-080401, available online http://www.hurray.isep.ipp.pt:8080, May 2008.

[18] A. Koubaa, A. Cunha, M. Alves, "A Time Division Beacon Scheduling Mechanism for IEEE 802.15.4/ZigBee Cluster-Tree Wireless Sensor Networks," In Euromicro Conference on Real-Time Systems (ECRTS'07), Italy, July 2007.

[19] MATLAB tool, http://www.open-zb.net/downloads.php, 2008.

[20] TelosB Datasheet, Crossbow Inc., http://www.xbow.com.

[21] TinyOS Community Forum, http://www.tinyos.net.

[22] A. Cunha, A. Koubaa, R. Severino, and M. Alves, "Open-ZB: an open-source implementation of the IEEE 802.15.4/ZigBee protocol stack on TinyOS," In IEEE Conference on Mobile Ad-hoc and Sensor Systems (MASS'07), Italy, Oct. 2007.

[23] CC2420DK Development Kit, http://www.ti.com.

[24] Daintree Sensor Network Analyzer (SNA), http://www.daintree.net.

[25] A. Cunha, R. Severino, N. Pereira, A. Kouba, M. Alves, "ZigBee over TinyOS: implementation and experimental challenges, " In the 8th Portuguese Conference on Automatic Control (CONTROLO 2008), Vila Real, Portugal, 2008.