A Comprehensive Worst-Case Calculus for Wireless Sensor Networks with In-Network Processing

Jens B. Schmitt[1], Frank A. Zdarsky[1], Lothar Thiele[2]

[1]Distributed Computer Systems Lab, University of Kaiserslautern, Germany
[2]Computer Engineering and Networks Laboratory, Swiss Federal Institute of Technology Zurich

Abstract

Today's wireless sensor networks (WSN) focus on energy-efficiency as the main metric to optimize. However, an increasing number of scenarios where sensor networks are considered for time-critical purposes in application scenarios like intrusion detection, industrial monitoring, or health care systems demands for an explicit support of performance guarantees in WSNs and, thus, in turn for a respective mathematical framework. In [1], a sensor network calculus was introduced in order to accommodate a worst-case analysis of WSNs. This sensor network calculus focused on the communication aspect in WSNs, but had not yet a possibility to treat in-network processing in WSNs. In this work, we now incorporate in-network processing features as they are typical for WSNs by taking into account computational resources on the sensor nodes. Furthermore, we propose a simple, yet effective priority queue management discipline which achieves a good balance of response times across sensor nodes in the field.

1. Introduction

Wireless sensor networks have requirements and characteristics that are considerably different from those of common computer networks. Issues include low energy reserves, limited processing power, uncontrolled environments and other adverse factors. The main attention has been laid on meeting these restrictions, so there has been much research on minimizing energy usage by introducing sleep times, reducing the amount of transmitted data, etc. While these topics remain important, other aspects have been somewhat neglected. Long sleep times go together with long delays, which can grow rapidly depending on the network topology. This becomes a problem for a growing number of WSN applications that have real-time requirements in addition to energy-efficiency targets such as intrusion detection, industrial monitoring, or health care systems, see also [2]. In order to accommodate these applications a mathematical framework taking into account the peculiarities of WSNs is desirable. We base this framework on network calculus, a promising theory for the analysis of worst-case models of data flow-oriented systems, which provides a good match with WSNs. However, what network calculus lacks is the consideration of transformations of the data flow inside the network as it is typical for WSNs in the form of in-network processing. In particular, in-network processing is an important ingredient to achieve energy-efficiency by reducing the amount of data that has to be communicated, under the typical assumption that computation is by several orders of magnitude cheaper than communication. Consequently, our motivation is to provide a comprehensive network calculus-based worst-case analysis methodology for WSNs taking in-network processing into account.

In the following we make a set of assumptions: a sink-tree based WSN topology is supposed; we assume that data sources' behavior can be upper-bounded (by arrival curves), as well as the communication and computational resources in WSNs can be lower-bounded (with the aid of service curves); control traffic from sink to sources or nodes is abstracted from (by modeling it into, e.g., latency terms in the nodal service curves, thus modeling a strict priority for control traffic). As we discuss below these should not be very restrictive assumptions and are satisfied in many interesting WSN scenarios.

Related Work. While there has been a growing body of work on real-time aspects in WSNs (see, e.g., [2] for a review and [3] for an interesting large-scale tracking system with real-time requirements and references therein), holistic system models on the interplay between performance characteristics and energy-efficiency targets have not been addressed much. One notable exception is the work by Chiasserini and Garetto [4], which proposes a performance model based on Markov chains that relates performance characteristics as, e.g., data delivery delay and energy management parameters as the duty cycle of nodes. However, for applications with real-time requirements such an average case analysis methodology is of limited use.

What is needed for real-time WSNs is a worst-case analysis methodology. In [1], a framework for the worst-case analysis of WSNs has been proposed based on network calculus [5]. It allows to relate again energy management parameters with performance characteristics, but now under worst-case assumptions. In [6], a methodology to analyze 802.15.4 cluster-tree WSNs has been proposed, again based on network calculus. Both of these approaches focus on the communication aspects within WSNs and do not integrate the processing resources on the sensor nodes. As discussed above the integration of communication and computation in a worst-case methodology for WSNs is the goal of this work.

Outline. In the following section, a brief overview of the most important concepts and results from network respectively real-time calculus as required for this paper is provided. The customization of these concepts to a WSN setting as well as their extension to the sink-tree case is presented in Section 3. Here, the way to calculate the bounds is different from [1] and [6] and constitutes a first contribution of this work. In Section 4, we advance the sensor network calculus by also including computational resources on the sensor nodes, which constitutes the major contribution of this paper. This is accomplished by integrating the real-time calculus [7] concept of workload transformations [8] that has also been coined data scaling in [9]. In Section 5, a queue management scheme that attempts to achieve a balancing of the response times across all sensor nodes in the field is proposed and integrated into the analytical framework as a further contribution. Numerical experiments that investigate some basic characteristics and trade-offs of the proposed methodology are discussed in Section 6.

2. Network Calculus Background

Network calculus is a min-plus system theory for deterministic queuing systems. A detailed treatment of min-plus algebra and of network calculus can be found in [10] and [11], [5], respectively. As network calculus is built around the notion of cumulative functions for input and output flows of data, the set of real-valued, non-negative, and wide-sense increasing functions passing through the origin plays a major role:

$$\mathcal{F} = \left\{ f : \mathbb{R}^+ \to \mathbb{R}^+ : \forall t \geq s : \; f(t) \geq f(s),\; f(0) = 0 \right\}.$$

In particular, the input function $R(t)$ and the output function $R^*(t)$, which cumulatively count the number of bits that are input to respectively output from a system $\mathcal{S}$, are in $\mathcal{F}$. Throughout the paper, we assume in- and output functions to be continuous in time and space. This is not a major restriction as there are transformations from discrete to continuous models [5].

Definition 1: (Min-plus Convolution and Deconvolution) The min-plus convolution respectively deconvolution of two functions f and g are defined to be

$$(f \otimes g)(t) = \inf_{0 \leq s \leq t} \{ f(t-s) + g(s) \},$$

$$(f \oslash g)(t) = \sup_{u \geq 0} \{ f(t+u) - g(u) \}.$$

Let us turn now to the performance characteristics of flows that can be bounded by network calculus means:

Definition 2: (Backlog and Delay) Assume a flow with input function R that traverses a system $\mathcal{S}$ resulting in the output function R^*. The backlog of the flow at time t is defined as

$$b(t) = R(t) - R^*(t).$$

Assuming first-in-first-out delivery, the delay for an input at time t is defined as

$$d(t) = \inf \{ \tau \geq 0 : R(t) \leq R^*(t+\tau) \}.$$

The arrival and server processes specified by input and output functions are bounded based on the central network calculus concepts of arrival and service curves:

Definition 3: (Arrival Curve) Given a flow with input function R, a function $\alpha \in \mathcal{F}$ is an arrival curve for R iff

$$\forall t, s \geq 0, s \leq t : R(t) - R(t-s) \leq \alpha(s)$$

$$\Leftrightarrow R \leq R \otimes \alpha \Leftrightarrow \alpha \geq R \oslash R.$$

Definition 4: (Service Curve) Let us suppose that a server process is able to process $C(t)$ bits of input data until time t. Then, $\beta \in \mathcal{F}$ is a minimum service curve for $C \in \mathcal{F}$ iff

$$\forall t,\; 0 \leq s \leq t : \; C(t) - C(t-s) \geq \beta(s).$$

Assuming the server is presented an input function R that it outputs as R^* this implies

$$R^* \geq R \otimes \beta.$$

Theorem 1: (Performance Bounds) Consider a system $\mathcal{S}$ that offers a service curve β and that stores input data in a FIFO-ordered queue. Assume a flow R traversing the system that has an arrival curve α. Then we obtain the following performance bounds for the backlog b, delay d and output arrival curve α^* for R^*:

$$\begin{aligned} b(t) &\leq (\alpha \oslash \beta)(0) =: v(\alpha, \beta), \\ d(t) &\leq \inf \{ t \geq 0 : (\alpha \oslash \beta)(-t) \leq 0 \} =: h(\alpha, \beta), \\ \alpha^* &\leq \alpha \oslash \beta. \end{aligned}$$

One of the strongest results of network calculus (albeit being a simple consequence of the associativity of $\otimes$) is the concatenation theorem that enables us to investigate tandems of systems as a single system:

Theorem 2: (Concatenation Theorem for Tandem Systems) Consider a flow that traverses a tandem of systems $\mathcal{S}_i$, $i = 1, ..., n$. Assume that $\mathcal{S}_i$ offers a service curve β_i, $i = 1, ..., n$ to the flow. Then the concatenation of the two systems offers a service curve $\bigotimes_{i=1}^{n} \beta_i$ to the flow.

So far we have only covered the tandem network case, the next result factors in the existence of other interfering flows. In particular, it states the *left-over* minimum service curve available to a flow at a single node under cross-traffic from other flows at that node. Let us model this scenario with a fixed-priority resource sharing discipline where some flow has a lower priority than all other flows. Then we can state the following theorem, see [7]:

Theorem 3: (Left-Over Service Curve) Consider a greedy system that offers a service curve β and that serves an input flow characterized by α. Then the left-over service available to other flows with lower priority is for all $t \geq 0$:

$$\beta^{l.o.}(t) = \sup_{0 \leq s \leq t} \{\beta(s) - \alpha(s)\}.$$

As a shorthand notation, we define

$$(\beta \ominus \alpha)(t) = \sup_{0 \leq s \leq t} \{\beta(s) - \alpha(s)\}.$$

Consider a node with service curve β arbitrarily multiplexes two flows 1 and 2, with arrival curves α_i, $i = 1, 2$. Then we can conclude from the last theorem that the least service available to flow 1 can be given as

$$\beta^1 = \beta \ominus \alpha_2$$

3. Enhancing Sensor Network Calculus

The sensor network calculus framework described in [1] has been a first step to enable a concise worst-case analysis of WSNs. It provides a way to derive bounds on performance measures as, e.g., the maximum delay experienced by any data flow in a WSN based on analyzing single servers in isolation and adding up their respective bounds. From conventional network calculus it was clear already then that this additive bounding method is more pessimistic than a method that would allow to take advantage of the concatenation result and take a holistic view on the WSN, yet the tree structure of the network does not allow for a direct application of the concatenation theorem. In this section, we now present a generalization of the concatenation result for the sink-tree network case, which results in better bounds than in [1].

3.1. Sensor Network System Model

We assume the common class of single base station oriented operation models, with a WSN being abstractly modeled as shown in Fig. 1. It is also assumed that the routing protocol being used forms a sink tree in the sensor network.

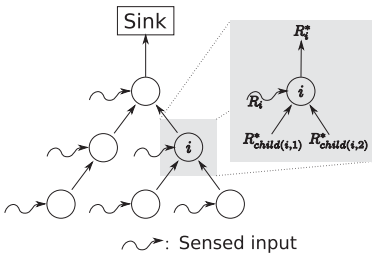

Figure 1. Sensor network model.

Each sensor node i senses its environment and thus is exposed to an input function R_i corresponding to its sensed input traffic. If sensor node i is not a leaf node of the tree then it will also receive sensed data from all of its child nodes $child(i, 1), \ldots, child(i, n_i)$, where n_i is the number of child nodes of sensor node i. Sensor node i forwards/processes its input, which results in an output function R_i^* from node i towards its parent node.

3.2. Incorporation of Network Calculus Components

Now the basic network calculus components, arrival and service curve, have to be incorporated. First, the arrival curve $\bar{\alpha}_i$ of each sensor node in the field has to be derived. The input of each sensor node in the field, taking into account its sensed input and its children's input, is given by:

$$\bar{R}_i = R_i + \sum_{j=1}^{n_i} R^*_{child(i,j)}.$$

Thus, the arrival curve for the total input function for sensor node i is given by:

$$\bar{\alpha}_i = \alpha_i + \sum_{j=1}^{n_i} \alpha^*_{child(i,j)}.$$

As an example, we could use simple token-bucket functions to model inputs. These are defined as $\gamma_{r,b}(t) = b + rt, t > 0$ and $\gamma_{r,b}(0) = 0$.

Second, the service curve has to be specified. The service curve depends on the way packets are scheduled in a sensor node, which, from the communication perspective, mainly depends on link layer characteristics. More specifically, the service curve depends on how the duty cycle and therefore the energy-efficiency goals are set. Again as an example, assume a service curve modeling the periodic availability of the full medium capacity C after an initial delay T. This closely captures the characteristics of a TDMA

medium access. The form of this curve for a node receiving s time units of service in a frame of duration f is shown in Fig. 2. The structure is similar to the one proposed in [6] for 802.15.4 networks. This curve can be approximated by a so-called rate-latency curve $\beta_{R,T}(t) = \max(R(t-T), 0)$ with $R = \frac{s}{f}C$ and $T = f - s$. That curve is shown labeled as β in the figure and can be considered the fluid version of the TDMA service curve.

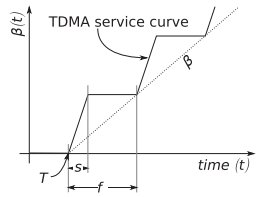

Figure 2. TDMA service curve.

3.3. Calculation of Internal Traffic Flow

The output of sensor node i, i.e., the traffic that it forwards to its parent in the tree, is constrained by the following arrival curve:

$$\alpha_i^* = \bar{\alpha}_i \oslash \beta_i. \quad (1)$$

Before performance characteristics like the maximum delay from a given node to the sink or local buffer requirements, especially at the most challenged sensor nodes just below the sink (which are called 1-hop nodes in the following), can be calculated, an iterative procedure to calculate the network-internal flows is required:

1. Let us assume that arrival curves for the sensed input α_i and service curves β_i for sensor node i, $i = 1, \ldots, n$, are given.

2. For all leaf nodes the output bound α_i^* can be calculated according to Theorem 1. Each leaf node is now marked as "calculated".

3. For all nodes only having children that are marked "calculated" the output bound α_i^* can be calculated according to Equation (1) and they can again be marked "calculated".

4. If all 1-hop nodes are marked "calculated" the algorithm terminates, otherwise go to step 3.

3.4. Calculation of Performance Bounds

After the network-internal flows have been computed, the local worst-case buffer requirements B_i and *per-node* delay bounds D_i for each sensor node i can be calculated according to Theorem 1:

$$B_i = v(\bar{\alpha}_i, \beta_i) = \sup_{s \geq 0}\{\bar{\alpha}_i(s) - \beta_i(s)\},$$

$$D_i = h(\bar{\alpha}_i, \beta_i) = \sup_{s \geq 0}\{\inf\{\tau \geq 0 : \bar{\alpha}_i(s) \leq \beta_i(s+\tau)\}\}.$$

One way to calculate the maximum delay from a given node till its data reaches the sink is now to just add up the per-node maximum delays on the path to the sink. Let us call this method total flow analysis (TFA) as flows are added up as on their way to the sink. This is what has been done in [1] and [6], the latter also proposed a further method that first computes the left-over service curve for a flow of interest at each node along its path and then concatenates all of these. Let us call this method separated flow analysis (SFA), as the flow of interest is separated from other flows by the application of Theorem 3, before its performance bounds are computed. As shown in [6] this can give an improvement over TFA. However, we can further improve this by making more careful use of the concatenation theorem, effectively *paying the costs of multiplexing only once*, which is why the method is called pay multiplexing only once analysis (PMOOA). The idea is to apply the concatenation as much as possible *before* the application of Theorem 3.

To illustrate the PMOOA method and its benefits, a very simple example illustrated in Fig. 3 shall be discussed. Here

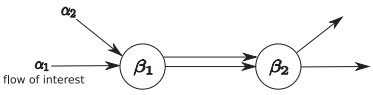

Figure 3. Simple example to illustrate bounding methods.

we can first concatenate and then apply Theorem 3 for the PMOOA, or we can apply TFA or SFA. The delay bounds for each of the three methods can be calculated as follows

$$\begin{aligned} d_{TFA} &= h\left(\alpha_1 + \alpha_2, \beta_1\right) + h\left(\left(\alpha_1 + \alpha_2\right) \oslash \beta_1, \beta_2\right) \\ d_{SFA} &= h\left(\alpha_1, [\beta_1 \ominus \alpha_2] \otimes [\beta_2 \ominus (\alpha_2 \oslash \beta_1)]\right) \\ d_{PMOOA} &= h\left(\alpha_1, [(\beta_1 \otimes \beta_2) \ominus \alpha_2]\right) \end{aligned}$$

If, for example, we assume specific rate-latency service curves $\beta_1 = \beta_2 = \beta_{3,0}$ and token-bucket arrival curves $\alpha_1 = \alpha_2 = \gamma_{1,1}$, this results in the following maximum delay calculations: $d_{TFA} = \frac{4}{3}$, $d_{SFA} = \frac{3}{2}$, $d_{PMOOA} = 1$. The PMOOA consequently gives the tightest bound. Note that the SFA in this example even gets worse than the TFA, although in larger scenarios it usually performs better than TFA as experienced by [6].

In [12] and [13], the PMOO analysis for general feed-forward networks as well as its relation to the other methods is described accurately and discussed in very much detail. Yet, due to the sink-tree structure of WSNs, all flows that join a flow of interest will remain multiplexed up to the sink, so that it is possible to simplify the analysis to Algorithm 1. Here, it is assumed that the service curves are located on the edges of the graph to be analyzed. In [14], a similar procedure specific for FIFO multiplexing is presented, yet as is also experienced in [15] releasing the FIFO assumption can make the treatment of performance bounds tricky (in that work it is shown that the famous pay bursts only once phenomenon is not true any more, similar to our findings in [13], but for the single flow tandem case instead of aggregate multiplexing).

Algorithm 1 Sink-tree PMOO Analysis.

1. Let $M = \{e_1, \ldots, e_k\}$ be the set of edges the flow of interest is traversing on the way from its source to the sink. Each edge e_i has an incoming node n_{i-1} and an outgoing node n_i.
2. Let $\beta_{k+1}^{l.o.} = \delta_0$ with
$$\delta_d(t) = \begin{cases} 0 & \text{if } t \leq d \\ \infty & \text{otherwise} \end{cases}$$
δ_0 is the neutral element of the min-plus convolution.
3. For all $e_{k \geq i \geq 1} \in M$, add up all upper output bounds of interfering flows from incoming nodes $n \neq n_{i-1}$ (for $i - 1 = 0$ this is the sum of all incoming flows except the flow of interest) and update the left-over service curve:
$$\beta_i^{l.o.} = \left(\beta_{i+1}^{l.o.} \otimes \beta_{e_i}\right) \ominus \sum_{n \neq n_{i-1}} \alpha_n^*$$
with α_n^* calculated according to the procedure specified in Section 3.3.
4. $\beta_1^{l.o.}$ is the left-over service curve for the flow of interest.

4. Augmenting Sensor Network Calculus by Modeling Computation

In the previous section, we reduced the sensor nodes to pure communication resources. This is of course a very coarse-grained abstraction of the functionality of a sensor node. In fact, a typical wireless sensor network commonly applies some in-network processing to the data that is delivered towards the sink, often in order to save energy by data aggregation. It is thus very interesting to factor computational resources on the nodes, i.e., the usage of the processing unit, into the sensor network calculus model. Since the workload units are however usually different between communication and computation, we also need to introduce elements into the model that translate between the usage of communication and computational resources. These elements have been coined as workload transformations or scaling elements in [8] resp. [9]. While they are conceptually simple components translating for example a certain number of bytes received from another sensor node into a worst-case sequence of events for the processing unit on the sensor node, they aggravate an end-to-end analysis as they build up "walls" between the different resources that inhibit the direct application of the concatenation result. However, we demonstrate how an end-to-end-analysis is still feasible by moving the scaling elements to the borders of the topology and taking into account the effect of this moving such that the application of the PMOO concatenation result is possible again.

4.1. Background on Data Scaling

In this subsection, we provide the necessary definitions and results for introducing scaling elements into network calculus models as presented in [9].

Definition 5: (Scaling Function) A scaling function $S \in \mathcal{F}$ assigns an amount of scaled data $S(a)$ to an amount of data a.

As can be seen from the definition of scaling functions, they are a very general concept for taking into account transformations in a network calculus model. Note however that they do not model any queueing effects–scaling is assumed to be done infinitely fast. Queueing related effects must still be modeled in the service curve element of the respective component.

Definition 6: (Scaling Curves) Given a scaling function S, two functions $\underline{S}$, $\overline{S} \in \mathcal{F}$ are minimum and maximum scaling curves of S iff $\forall b \geq 0$ it applies that
$$\begin{aligned} \underline{S}(b) &\leq \inf_{a \geq 0} \{S(b+a) - S(a)\} \\ \overline{S}(b) &\geq \sup_{a \geq 0} \{S(b+a) - S(a)\} \end{aligned}$$

Theorem 4: (Alternative Systems) Consider the two systems in Fig. 4 and let $R(t)$ be the input function. System (a) consists of a server with minimum service curve β whose output is scaled with scaling function S and system (b) consists of a scaling function whose output is input to a server with minimum service curve β_S. Given system (a) the lower bound of the output function of system (b), that is $S(R) \otimes \beta_S$ is also a valid lower bound for the output function of system (a) if
$$\beta_S = \underline{S}(\beta)$$

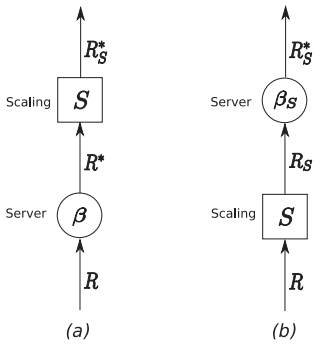

Figure 4. Alternative Systems.

This means in effect that performance bounds for system (b) under this assumption are also valid bounds for system (a), i.e., we can effectively move a scaling function in front of a service curve element as long as we transform the respective service curve using the minimum scaling curve of the scaling element. In [9], it is also shown that bounds computed in the alternative system, i.e., after shifting the scaling element downstream, remain tight.

The following corollary states the effect scaling has on arrival constraints of a traffic flow.

Corollary 1: (Arrival Constraints under Scaling) Let R be an input function with arrival curve α that is fed into a scaling function with maximum scaling curve $\overline{S}$. An arrival curve for the scaled output from the scaling element is given by

$$\alpha_S = \overline{S}(\alpha)$$

4.2. Data Scaling in Sink Trees

Using the scaling element we can now provide a much finer modeling of the WSN integrating the local processing resources into the overall model. This is illustrated in Fig. 5.

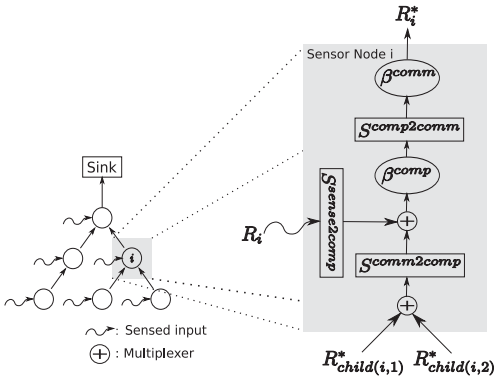

Figure 5. Advanced sensor network model.

This model is still based on the WSN model from Fig. 1, yet the modeling of the sensor nodes is refined by the integration of computational resources accommodated by the use of scaling elements. Besides different scaling elements we also introduce a multiplexing element, which allows to explicitly model how the superposition of a number of data flows is performed, e.g., arbitrary, FIFO or some specific order that may be based on strict priorities. Thus, the data that arrives from different predecessors in the sink tree is at first multiplexed and afterwards a scaling element models the transformation of the received data into a certain computational demand. The local sensing data are also transformed using a specific scaling element before they are multiplexed with the flows from upstream. This aggregate is then served by the processing unit of the sensor node providing a service curve β^{comp}. How much data is forwarded to the next node downstream depends on how the processing is for example able to compress the data flow by, e.g., data aggregation. This is captured by another scaling element, before the resulting flow is presented to the communication subsystem of the sensor node that is represented by another service curve element β^{comm}.

The problem with this new model now is that the scaling elements hinder an end-to-end analysis based on the PMOO principle. However, as we have seen in the previous subsection for a simple setting, it is possible to shift the scaling elements across a service curve element. So the idea is to shift all the scaling elements downstream in order to be able to do a true end-to-end analysis again. What remains to be discussed is how to *shift* the scaling elements *across multiplexers* where they conceptually have to be split onto several sub-flows. In the following theorem we provide a sufficient condition in order to be able to do this shifting without compromising the worst-case analysis.

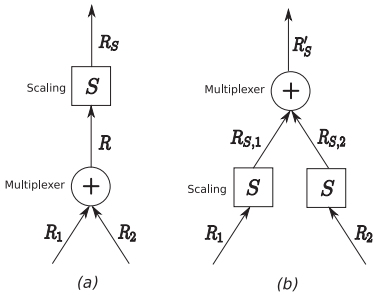

Figure 6. Scaling Element with Multiplexer.

Theorem 5: (Shift of Scaling Element across Multiplexer) Assume a situation as depicted in Fig. 6, i.e., two flows are multiplexed and then fed into a scaling function S with maximum scaling curve $\overline{S}$ and minimum scaling curve $\underline{S}$. Provided that the minimum scaling curve is super-additive and the maximum scaling curve is sub-additive we

can transform system (a) into system (b) without improving the worst-case scenario in system (b) over the one in system (a).

Proof: In Fig. 6, the multiplexer just adds the two input flows

$$R = R_1 + R_2$$

and the output is scaled by S

$$R_S = S(R) = S(R_1 + R_2).$$

Using the maximum and minimum scaling curve, we obtain

$$\begin{aligned} R_S &\leq \overline{S}(R_1 + R_2), \\ R_S &\geq \underline{S}(R_1 + R_2). \end{aligned}$$

On the other hand for system (b) in Fig. 6 we obtain the following relationships

$$R_S^{'} = R_{S,1} + R_{S,2} = S(R_1) + S(R_2) \leq \overline{S}(R_1) + \overline{S}(R_2),$$

$$R_S^{'} = R_{S,1} + R_{S,2} = S(R_1) + S(R_2) \geq \underline{S}(R_1) + \underline{S}(R_2).$$

As we want system (b) not to be excluding scenarios that are possible in system (a), we effectively require the upper resp. lower bound of R_S to be smaller resp. larger than those for $R_S^{'}$, i.e., it is required that

$$\begin{aligned} \overline{S}(R_1 + R_2) &\leq \overline{S}(R_1) + \overline{S}(R_2), \\ \underline{S}(R_1 + R_2) &\geq \underline{S}(R_1) + \underline{S}(R_2). \end{aligned} \tag{2}$$

These inequalities are just equivalent to $\overline{S}$ resp. $\underline{S}$ being sub-additive and super-additive, which is what has been assumed. □

Hence, there is no more hindrance to shift all the scaling elements to the sources of traffic. This is done by always shifting the remaining scaling functions closest to the leaves of the tree, thus accumulating scaling elements at the traffic flows where they enter the system. In effect, all the input to the system is scaled to the same resource units before entering a system of "homogenized" servers, where an end-to-end concatenation based on the techniques presented in Section 3 can be applied again.

Discussion. The alert reader may have noticed that while by shifting the scaling elements across a multiplexer we have not improved on the worst-case scenario, we may have actually worsened the situation. In fact, only when the scaling curves are linear we can be sure not to overestimate the worst case (because then we would have equality in the inequations (2) and systems (a) and (b) from Fig. 4 would be equivalent systems). Another source of overestimating the worst-case scenario when scaling elements are shifted behind service curve elements can be observed. Recall Fig. 4, the output bound for system (b) is calculated as $\overline{S}(\alpha) \oslash \underline{S}(\beta)$, which however cannot happen in any sample path of the system as maximum and minimum scaling curve cannot be attained at the same time unless they are equal. This means the more maximum and minimum scaling curves differ the more the worst-case scenario is overestimated. Here a potential cure to this situation could be suggested by moving the scaling elements upwards to the sink instead of downwards. However, a moment's consideration shows that this is only possible in a tandem network case and not in our case of a sink-tree network because shifting different scaling elements across a multiplexer upwards is not possible. Both of these effects counter the positive effect of a holistic end-to-end analysis and are quantitatively discussed in Section 6.

5. Response Time Balancing

In a WSN with real-time constraints one often faces the situation that the response time for *any* sensor in the field should not exceed a certain value. This puts nodes that are topologically distant from the sink at an obvious disadvantage. Therefore, it would be better to provide them with a preferential treatment at forwarding nodes than data from nodes with shorter paths to the sink. A simple strategy that always gives strict priority to flows that have had a longer path should strike a good compromise between investing computational effort and state management for flows and a good balance of response times across the sensor field. Specifically, we can carry the state information that is required in each packet, because it is simply the hop count. We call this queue management scheme of always giving priority to the data that has traversed the highest number of hops so far: *Longest Flow First (LFF)*.

While LFF is a very intuitive countermeasure against the topologically unfortunate placement of some nodes, the interesting question now is how to accommodate this queue management strategy in the worst-case analysis. This is actually very simple within our framework (see also Algorithm 2): We step through the tree level by level, starting from the lowest leaf level. While analyzing the flows originating at level i, we only consider flows of equal or higher priority, i.e., we temporarily ignore all flows that cannot interfere due to their strictly lower priority (those originating at levels $< i$).

Algorithm 2 LFF Network Analysis.

Given: sink tree of service curve elements with d levels
Initialization: deactivate all traffic sources
FOR i from d down to 1
    Activate all sources at level i
    Perform PMOOA for all flows originating at level i

6. Numerical Experiments

In this section, we investigate numerically under realistic parameter settings how advantageous it is to perform a holistic analysis of a WSN based on the PMOO result and the shifting of scaling elements from Section 4 instead of a component-wise analysis based on the total flow analysis (TFA) from Section 3. Furthermore, we investigate how this relation is affected by a deviation between minimum and maximum scaling curve. Last but not least, it is also examined how well the LFF queue management performs in comparison to arbitrary multiplexing with respect to balancing response times more equally among sensor nodes in the field.

6.1. Experimental Design

First of all we generate a set of random WSN topologies by placing n sensor nodes on a square based on a uniform random distribution. Each sensor has a transmission range of $20m$ such that, for suitable sensor field sizes, connectivity is ensured with high likelihood (not fully connected topologies are discarded). The sink is placed in the center of the field and the forwarding tree towards the sink is constructed as shortest path tree. For each experiment we generate 10 topologies in order to avoid random effects due to topological peculiarities. In most of the experiments we used $n = 100$ nodes and a square sensor field of $100m^2$. In the experiments concerning the performance of the LFF queue management we varied the number of nodes and adapted the area size accordingly to be proportional to $\sqrt{n}$, so that connectivity characteristics are conserved.

The instantiation of the sensor node related parameters is based on MICAz motes from Crossbow Technology Inc. Thus, a physical data rate of $250kbps$ is assumed, for the MAC we suppose a TDMA-based scheme that allows for a duty cyle of 1% with a TDMA frame length of $100ms$. That means we obtain the service curve for communication as $\beta^{comm} = \beta_{2.5[kbps],0.099[s]}$. For the packet length $36B$ TinyOS packets are assumed. Regarding computational resources, the MICAz is equipped with an Atmel ATmega 128L microcontroller that operates at 8 Mhz. Hence, supposing on average 4 cycles per instruction, it has a raw capacity of 2 MIPS. Assuming the processing unit also has to perform other tasks, for example related to network control operations, and it also applies a power management scheme with certain sleep periods in low power modes, we set its availability for data path related processing (including sensing) to 10% of its full capacity with a potential latency of 1 ms. Thus we obtain as service curve for computation $\beta^{comp} = \beta_{0.2[MIPS],0.001[s]}$.

With respect to the scaling elements we use token-bucket functions for the maximum scaling curves and rate-latency functions for the minimum scaling curves. The maximum scaling curve for communication into computation resource demand transformation is assumed to be $\overline{S}^{comm2comp} = \gamma_{5000[Instr./p],b[Instr.]}$, i.e., in the long-term a received packet consumes 5000 instructions, deviations from this are modeled by the bucket depth parameter b that will be varied in the experiments. The minimum scaling curve is supposed to be given as $\underline{S}^{comm2comp} = \beta_{5000[Instr./p],T[p]}$, where the latency parameter T will again be varied in the experiments. Similarly maximum and minimum scaling curves for the transformation of sensing into computation resource demand are given as $\overline{S}^{sense2comp} = \gamma_{6000[Instr./p],b[Instr.]}$and $\underline{S}^{sense2comp} = \beta_{6000[Instr./p],T[p]}$, where a somewhat higher demand for sensing is supposed since raw sensing values have to be processed first. The scaling curves for the transformation of computation into communication resource demand is set such that it is roughly inverse to the other scaling elements and additionally models some compression due to, e.g., data aggregation. The compression factor varies depending on the number of children of a node. In particular, it decays exponentially with the number of child nodes in a given interval from 0.6 to 0.2, i.e., the downstream flow of a node can be scaled down to one fifth the size of the aggregate input flow, roughly speaking.

The fresh arrivals from the sensing unit are modeled by a token bucket $\gamma_{0.1[p/s],1[p]}$, i.e., it is assumed that a packet is created every 10 seconds with the obvious possibility of an instantaneous packet arrival in an arbitrary short interval.

While it is clear that many of these values are a somewhat arbitrary and would need to be tailored to a specific scenario, they should nevertheless be in a reasonably realistic region.

In the baseline comparison between holistic and component-wise analysis we assume the multiplexers to be arbitrary, later on we investigate LFF at the multiplexing elements. All of this is performed with the aid of the DISCO Network Calculator [16][1], in which we implemented all of the above presented concepts and methods.

6.2. Benefit of End-to-End Analysis

Baseline Comparison. At first we want to find out how the holistic analysis based on PMOOA performs in comparison to the component-wise analysis based on TFA. Here, we assume identical minimum and maximum scaling curves going through the origin. All other parameters are chosen as described in the previous subsection. In Fig. 7, the results of both methods for 10 different random topologies with 100 nodes each are shown.

[1]The DISCO Network Calculator is publicly available under http://disco.informatik.uni-kl.de/content/Downloads.

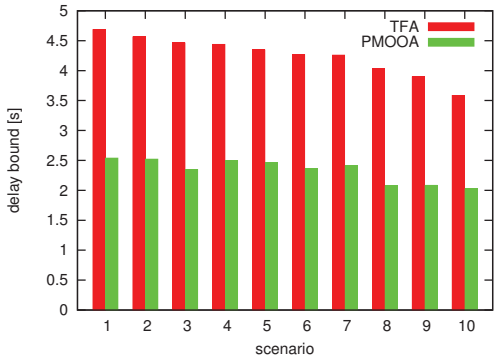

Figure 7. Baseline comparison between PMOOA and TFA for different scenarios.

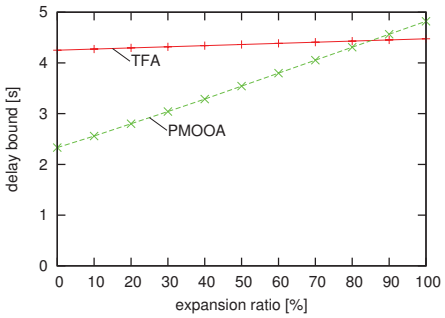

Figure 8. Investigation of PMOOA vs. TFA under the drifting apart of maximum and minimum scaling curves.

For each scenario the maximum delay observed across all nodes in the field is reported. As can be perceived easily the PMOOA is a clear winner with the TFA delivering delay bounds up to 1.9 times higher than those for the PMOOA.

Effect of Uncertainty about Scaling. Often scaling cannot be captured as deterministically as in the previous section, i.e., with minimum and maximum scaling curve being identical. Yet, usually it can still be upper- and lower-bounded, which means we obtain differing minimum and maximum scaling curves. As discussed in Section 4, this may have an adverse effect on the performance of the holistic analysis method. To investigate this issue, we now vary the minimum and maximum scaling curves to be no more linear. Effectively we shift the maximum scaling curve upwards along the y-axis and shift the minimum scaling curves along the x-axis to the right. This drifting apart of maximum and minimum scaling curve is achieved by setting the bucket depth parameters of the maximum scaling curves respectively the latency parameters of the minimum scaling curves incrementally higher until we reach certain limits. These limits are as follows: for $\underline{S}^{comm2comp}$ the maximum latency is 0.006; for $\overline{S}^{comm2comp}$ the maximum bucket depth is 240; for $\underline{S}^{sense2comp}$ the maximum latency is 0.006; for $\overline{S}^{sense2comp}$ the maximum bucket depth is 300; for $\underline{S}^{comp2comm}$ the maximum latency is 240; for $\overline{S}$ the maximum bucket depth is 0.006.

In Fig. 8, the shift along the axes is illustrated by an expansion factor giving the percentage of how much these limits have been utilized. The delay bound is averaged over 10 random topologies for each method.

It can be observed that the PMOOA still performs better than the TFA up to a certain point of the drifting apart of the maximum and minimum scaling curve. After this point it is actually better to use the TFA because the uncertainty about the scaling becomes too large to justify the application of the PMOOA. Where this point is located depends a great deal on the respective scenario. So, as a general wisdom, it is good to perform both analyses to be on the safe side.

6.3. Benefit of LFF Queue Management

In the last experiment, we investigate how much balancing of response times across the field is accommodated by assuming LFF as a queue management discipline for multiplexing flows. We do this for different sizes of the WSN ranging from 100 nodes up to 1000 nodes with a corresponding deeper sink-tree structure using the basic set of parameter values as presented in 6.1 with identical scaling curves (we only perform the PMOOA anyway). The results of this experiment are shown in Fig. 9.

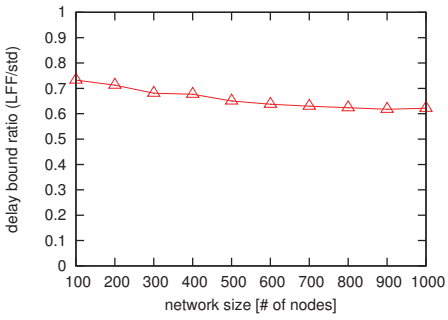

Figure 9. Use of LFF vs. arbitrary multiplexing.

Here, the ratio of the delay bound for each size of WSN (averaged over 10 random topologies) between PMOOA with LFF and PMOOA with arbitrary multiplexing is given. As can be perceived the gain by using LFF instead of arbitrary multiplexing is substantial at about a 30-40 % reduction of the delay bound. For the improvement due to LFF, a slight dependency on an increasing network size is visible.

7. Conclusions

We believe to have taken a significant step on the way to a comprehensive sensor network calculus, an analytical framework that shall enable a tight performance dimensioning and control of WSNs, which besides energy-efficiency goals also have timing requirements. In particular, we presented a new method to calculate end-to-end delays in sink-tree networks under arbitrary multiplexing that outperforms existing methods. Most importantly, in our opinion, we extended our analytical framework to incorporate computational resources besides the communication aspects of WSNs. This is considered very important for WSNs as it allows to model their typical in-network processing. Since delay bounds in larger WSNs may differ substantially based on their topological position we also proposed a simple queue management that alleviates this problem and demonstrated how it is accommodated within our analytical framework. In numerical experiments, we demonstrated the quantitative behavior of the proposed concepts and methods.

References

[1] J. Schmitt and U. Roedig. Sensor network calculus - a framework for worst case analysis. In *Proc. Distributed Computing on Sensor Systems (DCOSS)*, pages 141–154, June 2005.

[2] J.A. Stankovic, T. Abdelzaher, C. Lu, L. Sha, and J.Hou. Real-time communication and coordination in embedded sensor networks. *Proceedings of the IEEE*, 91:1002–1022, 2003.

[3] T. He, P. Vicaire, T. Yan, L. Luo, L. Gu, G. Zhou, R. Stoleru, Q. Cao, J.A. Stankovic, and T. Abdelzaher. Achieving real-time target tracking using wireless sensor networks. In *Proc. 12th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, April 2006.

[4] C.-F. Chiasserini and M. Garetto. Modeling the performance of wireless sensor networks. In *Proc. IEEE INFOCOM*, March 2004.

[5] J.-Y. Le Boudec and P. Thiran. *Network Calculus A Theory of Deterministic Queuing Systems for the Internet*. Number 2050 in Lecture Notes in Computer Science. Springer-Verlag, Berlin, Germany, 2001.

[6] A. Koubaa, M. Alves, and E. Tovar. Modeling and worst-case dimensioning of cluster-tree wireless sensor networks. In *Proc. 27th IEEE International Real-Time Systems Symposium (RTSS'06)*, pages 412–421, Rio de Janeiro, Brazil, 2006. IEEE Computer Society.

[7] L. Thiele, S. Chakraborty, and M. Naedele. Real-time calculus for scheduling hard real-time systems. In *Proc. IEEE International Symposium on Circuits and Systems (ISCAS)*, volume 4, pages 101–104, 2000.

[8] E. Wandeler, A. Maxiaguine, and L. Thiele. Quantitative Characterization of Event Streams in Analysis of Hard Real-Time Applications. In *10th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, pages 450–459, May 2004.

[9] M. Fidler and J. Schmitt. On the way to a distributed systems calculus: An end-to-end network calculus with data scaling. In *ACM SIGMETRICS/Performance 2006 (SIGMETRICS'06)*, pages 287–298, St. Malo, France, June 2006. ACM.

[10] F. Baccelli, G. Cohen, G. J. Olsder, and J.-P. Quadrat. *Synchronization and Linearity: An Algebra for Discrete Event Systems*. Probability and Mathematical Statistics. John Wiley & Sons Ltd., West Sussex, Great Britain, 1992.

[11] C.-S. Chang. *Performance Guarantees in Communication Networks*. Telecommunication Networks and Computer Systems. Springer-Verlag, London, Great Britain, 2000.

[12] J. Schmitt, F. Zdarsky, and I. Martinovic. Performance Bounds in Feed-Forward Networks under Blind Multiplexing. Technical Report 349/06, University of Kaiserslautern, Germany, April 2006.

[13] J. Schmitt, F. Zdarsky, and M. Fidler. Delay Bounds under Arbitrary Multiplexing. Technical Report 360/07, University of Kaiserslautern, Germany, July 2007.

[14] L. Lenzini, L. Martorini, E. Mingozzi, and G. Stea. Tight end-to-end per-flow delay bounds in fifo multiplexing sink-tree networks. *Perform. Eval.*, 63(9-10):956–987, 2006.

[15] J.-Y. Le Boudec and G. Rizzo. Pay bursts only once does not hold for non-fifo guaranteed rate nodes. *Perform. Eval.*, 62(1-4):366–381, 2005.

[16] J. Schmitt and F. Zdarsky. The DISCO Network Calculator - A Toolbox for Worst Case Analysis. In *Proceedings of the First International Conference on Performance Evaluation Methodologies and Tools (VALUETOOLS'06), Pisa, Italy*. ACM, November 2006.