Boosting Sensor Network Calculus by Thoroughly Bounding Cross-Traffic

Steffen Bondorf and Jens B. Schmitt
Distributed Computer Systems (DISCO) Lab, University of Kaiserslautern, Germany

***Abstract*—Sensor Network Calculus (SensorNC) provides a framework for worst-case analysis of wireless sensor networks. The analysis proceeds in two steps: For a given flow, (1) the network is reduced to a tandem of nodes by computing the arrival bounds of cross-traffic; (2) the flow is separated from the cross-traffic by subtracting cross-flows and concatenating nodes on its path. While the second step has seen much treatment, the first step has not at all. This is in sharp contrast to the fact that arrival bounding takes roughly 80% of the total analysis time and is equally crucial for the tightness of the bounds. Therefore, we turn our attention to this first SensorNC analysis step with the goal to boost the performance and applicability of the overall framework. The main technical contribution is a generalized version of the concatenation theorem within the SensorNC setting. This generalization is instrumental in simplifying and streamlining the cross-traffic arrival bound computations such that run times can be reduced by more than a factor of 5. Even more important, it enables a localization of the information necessary to execute the calculations at the node level, thus enabling a distribution of the SensorNC analysis within a self-modeling WSN.**

I. Introduction

Sensor Network Calculus (SensorNC) is a framework for worst-case analysis of wireless sensor networks and as such allows to derive tight upper bounds on end-to-end delay for sensor-to-sink data flows as well as on maximum buffer requirements for sensor nodes. SensorNC is strongly based on network calculus [1], but partially slims it down (specific arrival and service curves are used) and partially extends it (feedforward network analysis). Since the initial proposal in [2], it has been extended in several aspects: Multiple sinks [3], in-network processing [4], improved delay analysis [5], [4], and [6], to name a few. Furthermore, it was applied for diverse purposes, e.g., to model and analyze cluster-tree based IEEE 802.15.4 networks [7], [6], to evaluate traffic splitting in meshed WSNs [8], or to plan the trajectories of multiple mobile sinks in a large-scale, time sensitive WSN [9].

From a high-level conceptual point of view, the SensorNC analysis is divided into two steps:

1) **Cross-Traffic Arrival Bounding**: For a specific flow of interest, the interference of its cross-flows is captured in terms of arrival curves at every node it traverses. Thus, the focus is set on the flow of interest's path. The result is a tandem of nodes with each one potentially contributing an interfering flow aggregate, i.e., a cross-traffic arrival bound. This is illustrated in Figure 1(a).
2) **End-to-end Service Curve Calculation**: Given the resulting tandem from Step 1, the end-to-end service curve for the flow of interest is calculated by performing a sequence of left-over service curve calculations (subtraction of interfering flows) and concatenation of nodes (using the min-plus convolution).

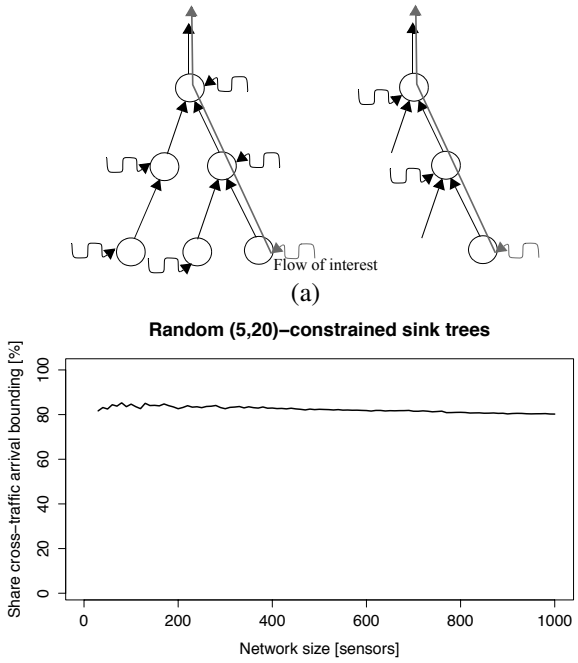

Figure 1. Cross-Traffic Arrival Bounding: (a) Tandem abstraction; (b) observed share of total calculation time.

Several works have delved into the details of step 2, in particular addressing the order in which to perform the operations. In [10], it is shown that it is generally favorable to perform the concatenation before the subtraction as much as possible to achieve the so-called Pay Multiplexing Only Once (PMOO) property. Yet, in [11] it is demonstrated that this is not always optimal and instead an optimization-based approach is proposed to deliver tight delay bounds. [12] further details the optimization-based approach and establishes the NP-completeness of finding tight delay bounds in general feedforward networks. However, in [4] it is argued that for single sink, homogeneous WSNs the PMOO analysis actually coincides with the optimal solution.

Hence, much effort has been devoted to tune step 2 of the analysis. In contrast, to the best of our knowledge, apart from some preliminary investigations in [13] there is no work trying to optimize step 1, the cross-traffic arrival bound calculation. In the literature, it is either assumed that it has been performed

already or, as implied by the original SensorNC paper [2], it is performed in a straightforward recursive manner, which can actually yield sub-optimal bounds [13]. Moreover, step 1 actually consumes a high fraction of the computation time when performing SensorNC analyses. To illustrate this, we show the fraction of time invested for arrival bound computations to the total run time for the analysis of random sink trees with a given number of nodes in Figure 1(b). As can be observed, arrival bound computations consume roughly 80% of the overall run time of the analysis – independent of the network size. Thus, improvements in the arrival bound computation are a very promising candidate to tune the performance of SensorNC.

Therefore, in this paper, we focus on step 1 of the analysis and improve on the cross-traffic arrival bound calculation in several aspects. To that end, we generalize the classical concatenation theorem as it applies to tandems of nodes towards a sink-tree structure. Equipped with this new generalized concatenation theorem, we derive a new iterative algorithm to perform the arrival bound calculation, which we implemented with the DISCO Deterministic Network Calculator [14]. Using this implementation, we demonstrate in Section V that run times for SensorNC analyses can generally be decreased by a factor of five without compromising arrival bound tightness.

However, we believe that the even larger impact of this improved arrival bound calculation algorithm is in its superior structure from the perspective of a distributed application of the SensorNC, for example, in an in-network sensing task admission control system as it would be desirable for large-scale WSNs. In particular, using the conventional recursive arrival bounding method for a distributed SensorNC would require detailed knowledge at each node about large parts of the network topology including paths of flows as well as their merging points in order to trace flow transformations, quantify cross-traffic and carry out an analysis. Moreover, even small changes in the network's defining parameters would lead to recalculations of all bounds, although the change is only local.

In contrast, using the new iterative arrival bounding method allows to virtually separate cross-flows from the aggregate they are merged into and separately bound their impact on it. The separation virtually shifts the aggregation of cross-flows to the server where they influence another flow's performance characteristics. Thus, our analysis establishes locality in a way such that we do not need access to topological information on a cross-flow's path, i.e., every flow can collect all the parameters defining its effect – we call this *flow-locality*. Furthermore, this flow-locality makes SensorNC much more resilient to recalculations as the impact of parameter variations does not propagate to flows not directly depending on this parameter.

Outline: The remainder of the paper is structured as follows: Section II presents background on (sensor) network calculus. In Section III, we present and prove the generalized concatenation theorem for improved cross-traffic arrival bounding. In Section IV, we discuss the major shortcoming of current SensorNC analyses and show how to overcome it by applying our generalized concatenation theorem. Section V evaluates the impact of our improvement and Section VI concludes the paper.

II. Background on Sensor Network Calculus

Here, we present the basics of (sensor) network calculus.

A. Modeling of Flows and Performance Characteristics

In network calculus, flows are modeled as cumulative functions, i.e., they are non-negative and wide-sense increasing:

$$\mathcal{F}_0 = \left\{ f : \mathbb{R}^+ \to \mathbb{R}^+ \mid f(0) = 0,\ \forall s \le t\ :\ f(t) \ge f(s) \right\}.$$

In particular, we are interested in the functions $A(t)$ and $A'(t)$ counting a flow's data put into a system $\mathcal{S}$ up until time t and put out from $\mathcal{S}$ until t. We further demand systems and flows to preserve causality by fulfilling the flow constraint, i.e., $\forall t \in \mathbb{R}^+ :\ A(t) \ge A'(t)$.

These definitions allow us to define performance characteristics of flows.

Definition 1. *(Backlog and Delay) Assume a flow with input function A traverses a system $\mathcal{S}$ and results in the output function A'. The* backlog *of the flow at time t is defined as*

$$B(t) = A(t) - A'(t).$$

The (virtual) delay *for a data unit arriving at $\mathcal{S}$ at time t is defined as*

$$D(t) = \inf \left\{ \tau \ge 0 \mid A(t) \le A'(t+\tau) \right\}.$$

B. Sensor Network Calculus Performance Analysis

In network calculus, we assume to only know bounding functions on the actual flows. These functions are not defined over the time of the observation t but over the duration of an observation d. In particular, the input flow in (sensor) network calculus is bounded by so-called arrival curves.

Definition 2. *(Arrival Curve) Given a flow with input function A, a function $\alpha \in \mathcal{F}_0$ is an arrival curve for A iff*

$$\forall\, 0 \le d \le t\ :\ A(t) - A(t-d) \le \alpha(d)$$

Sensor network calculus further restricts the set of arrival curves to token-bucket shaped traffic

$$\mathcal{F}_{\text{TB}} = \left\{ \gamma_{r,b} : \mathbb{R}^+ \to \mathbb{R}^+ \mid \gamma_{r,b}(0) = 0,\ \underset{d>0}{\forall} \gamma_{r,b}(d) = b + r \cdot d \right\}$$
$$\subseteq \mathcal{F}_0,\ r, b \ge 0.$$

Curves of $\mathcal{F}_{\text{TB}}$ are well suited to bound periodically generated and reported measurement data. The burst parameter b corresponds to the maximum size of a measurement and $r = \frac{b}{p}$ bounds the long term rate of data generation, with p being the sensing period.

Network calculus can be cast into a $(\wedge, +)$-algebraic framework over $\mathcal{F}_0$. A detailed treatment of $(\wedge, +)$-algebra and network calculus can be found in [15] and [1], [16], respectively. Here, we only present the most important operations.

Definition 3. *($(\wedge, +)$-Operations) The $(\wedge, +)$-algebraic aggregation, convolution and deconvolution of two functions $f, g \in \mathcal{F}_0$ are defined as*

$$\begin{aligned}
\textit{aggregation:}\quad & (f+g)(d) &&= f(d) + g(d), \\
\textit{convolution:}\quad & (f \otimes g)(d) &&= \inf_{0 \le s \le d} \{ f(d-s) + g(s) \}, \\
\textit{deconvolution:}\quad & (f \oslash g)(d) &&= \sup_{u \ge 0} \{ f(d+u) - g(u) \}.
\end{aligned}$$

Note that deconvolution is not exactly dual to convolution [1].

Aggregating arrival curves is crucial in SensorNC and can simply be achieved by

Corollary 4. *(Arrival Curve Aggregation in SensorNC) For the aggregation of n arrival curves of $\mathcal{F}_{\text{TB}}$ it holds that*

$$\sum_{i=1}^{n} \gamma_{r_i,b_i} = \gamma_{\sum_{i=1}^{n} r_i, \sum_{i=1}^{n} b_i}.$$

Defining curves characterizing the transformation a flow experiences when traversing a system $\mathcal{S}$, we need to use convolution and deconvolution.

Definition 5. *(Service Curve) If the service provided by a system $\mathcal{S}$ for a given input function A results in an output function A' we say that $\mathcal{S}$ offers a service curve β iff*

$$A' \geq A \otimes \beta.$$

In SensorNC, service is given by rate-latency functions from

$$\mathcal{F}_{\text{RL}} = \left\{\beta_{R,T} : \mathbb{R}^+ \to \mathbb{R}^+ \,|\, \beta_{R,T}(d) = \max\{0, R \cdot (d-T)\}\right\} \subseteq \mathcal{F}_0,\ T \geq 0,\ R > 0.$$

For example, TDMA channel access [5] as well as duty cycling nodes [17] can be modeled with these functions.

A number of systems fulfill, however, a stricter definition of service curves which permits certain derivations that are not permissible under the general service curve model [1].

Definition 6. *(Strict Service Curve) Let $\beta \in \mathcal{F}_0$. System $\mathcal{S}$ offers a* strict *service curve β to a flow if, during any backlogged period of duration d the output of the flow is at least equal to $\beta(d)$.*

Most notably, the logical separation of a flow form an aggregate requires a strict service curve.

Bounding a flow after traversing a system, i.e., deriving an arrival curve that bounds $A'(t)$, from a service curve and an arrival curve for $A(t)$ is done as follows:

Theorem 7. (Output Arrival Curve) *Assume a flow f has an arrival curve α and consider f traversing the system $\mathcal{S}$ offering a service curve β. After being transformed by $\mathcal{S}$, i.e., at the system's output, f is bounded by the arrival curve*

$$\alpha'(d) = \left(\alpha \dot{\oslash} \beta\right)(d). = \begin{cases} 0 & \text{if } d = 0 \\ (\alpha \oslash \beta)(d) & \text{otherwise} \end{cases}.$$

The deconvolution does not guarantee $(\alpha \oslash \beta)(0) = 0$ and is thus not closed in $\mathcal{F}_0$. It has to be slightly augmented to fulfill the arrival curve definition. This, however, does not impact the performance bounds derivation.

Theorem 8. *(Performance Bounds) Consider a system $\mathcal{S}$ that offers a service curve β. Assume a flow f with arrival curve α traverses the system. Then we obtain the following performance bounds for f:*

backlog: $\forall t \in \mathbb{R}^+ :\ B(t) \leq (\alpha \oslash \beta)(0)$

delay: $\forall t \in \mathbb{R}^+ :\ D(t) \leq \inf\{d \geq 0 \mid (\alpha \oslash \beta)(-d) \leq 0\}$

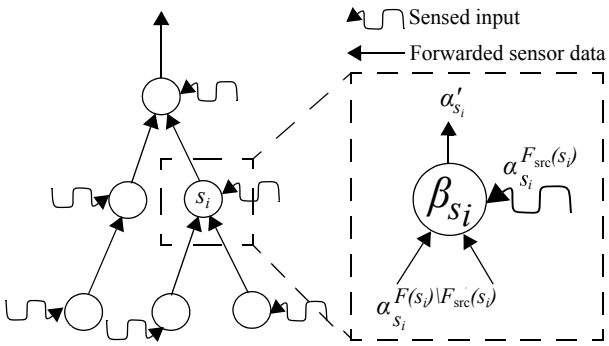

Figure 2. Sensor Network Model [2].

One of the strongest results of network calculus is the concatenation theorem that enables us to investigate tandems of systems as if they were single systems:

Theorem 9. *(Concatenation Theorem for Tandem Systems) Consider a flow that traverses a tandem of systems $\mathcal{S}_i$, $i = 1, \ldots, n$. Each $\mathcal{S}_i$ offers a service curve $\beta_{\mathcal{S}_i}$ to the flow. Then the concatenation of the n systems offers a service curve $\bigotimes_{i=1}^{n} \beta_{\mathcal{S}_i}$ to the flow.*

Using the concatenation theorem for an end-to-end analysis, it is possible to derive *tight* performance bounds on backlog and delay; in contrast, a node-by-node application of Theorem 7 and Theorem 8 does, in general, not guarantee tightness.

C. Sensor Network Calculus System Model

Besides the basic assumptions from the previous subsection, SensorNC also typically assumes a restriction on the topology space: Network topologies are limited to sink trees with a single sink (see Figure 2). While multiple sinks have been addressed in [3], such topologies can be transformed into a set of sink trees with one sink each without much loss of precision. Anyway, apart from this restriction, we aim for the greatest amount of generality possible. Therefore our considerations neither requires sensors to be homogeneous nor impose any restriction on the outdegree or maximum distance to the sink as found in [2], [18], [19], [5]. We use the term node synonymously with server and sensor because we assume that every node provides both functionalities, sensing data and relaying incoming flows.

III. The Generalized Concatenation Theorem

The concatenation theorem is only applicable to a limited set of network configurations. The topology must be a tandem of servers and the flows need to traverse it entirely from end-to-end. Otherwise the binary $(\wedge, +)$-convolution $\otimes$ cannot be used within the analysis. Naturally, networks are more complex than such simple $1:1$ input/output systems without any internal addition of cross-flows. In this paper, we provide a generalized version of Theorem 9 for arbitrary sink-tree networks in SensorNC – exactly accounting for flow entanglement in these $n:1$ input/output systems while preserving tightness and achieving a high reduction of complexity.

To that end, we use the following two corollaries (from Theorem 7 and 8, respectively).

Corollary 10. *(Output Arrival Curve in SensorNC) Assume a flow f with arrival curve $\alpha = \gamma_{r,b} \in \mathcal{F}_{\text{TB}}$ traverses a system $\mathcal{S}$ offering a service curve $\beta = \beta_{R,T} \in \mathcal{F}_{\text{RL}}$. In SensorNC, f is bounded at the output of $\mathcal{S}$ by the arrival curve*

$$\gamma_{r',b'}(d) = \left(\gamma_{r,b} \dot{\oslash} \beta_{R,T}\right)(d) = \begin{cases} 0 & \text{if } d = 0 \\ \gamma_{r,\, b+r\cdot T}(d) & \text{otherwise.} \end{cases}$$

We can apply Corollary 10 to adapt Theorem 8.

Corollary 11. *(Performance Bounds in SensorNC) Consider a system $\mathcal{S}$ that offers a service curve $\beta = \beta_{R,T} \in \mathcal{F}_{\text{RL}}$. Assume a flow f with arrival curve $\alpha = \gamma_{r,b} \in \mathcal{F}_{\text{TB}}$ traverses $\mathcal{S}$. Then we obtain the following SensorNC performance bounds:*

$$\textit{backlog: } \forall t \in \mathbb{R}^+ : \; B(t) = b + r \cdot T,$$
$$\textit{delay: } \forall t \in \mathbb{R}^+ : \; D(t) = T + \frac{b}{R}.$$

Next, we prove that the above output arrival curve derivation is distributive over flow aggregation in SensorNC.

Lemma 12. *(Distributivity of $\dot{\oslash}$ with respect to $+$) For any $\alpha^{f_1}, \alpha^{f_2} \in \mathcal{F}_{\text{TB}}$ and $\beta \in \mathcal{F}_{\text{RL}}$ it holds that*

$$\left(\alpha^{f_1} + \alpha^{f_2}\right) \dot{\oslash} \beta \;=\; \alpha^{f_1} \dot{\oslash} \beta + \alpha^{f_2} \dot{\oslash} \beta.$$

Proof: Let $\alpha^{f_1} = \gamma_{r_1,b_1}$, $\alpha^{f_2} = \gamma_{r_2,b_2}$ and $\beta = \beta_{R,T}$. From Corollary 10 it follows that

$$\begin{aligned} \left(\alpha^{f_1} + \alpha^{f_2}\right) \dot{\oslash} \beta &= \left(\left(\gamma_{r_1,b_1} + \gamma_{r_2,b_2}\right) \dot{\oslash} \beta_{R,T}\right)(d) \\ &= \left(\gamma_{r_1+r_2,\, b_1+b_2} \dot{\oslash} \beta_{R,T}\right)(d). \end{aligned}$$

If $d = 0$ we have $\alpha^{f_1} \dot{\oslash} \alpha^{f_2}(d) = 0$ and for $d > 0$ we get

$$\begin{aligned} &\left(\gamma_{r_1+r_2,\, b_1+b_2} \dot{\oslash} \beta_{R,T}\right)(d) \\ &= \left(\gamma_{r_1+r_2,\, (b_1+b_2)+(r_1+r_2)\cdot T}\right)(d) \\ &= \left(\gamma_{r_1,\, b_1+r_1\cdot T} + \gamma_{r_2,\, b_2+r_2\cdot T}\right)(d) \\ &= \left(\gamma_{r_1,\, b_1+r_1\cdot T}\right)(d) + \left(\gamma_{r_2,\, b_2+r_2\cdot T}\right)(d) \\ &= \left(\gamma_{r_1,b_1} \dot{\oslash} \beta_{R,T}\right)(d) + \left(\gamma_{r_2,b_2} \dot{\oslash} \beta_{R,T}\right)(d) \\ &= \left(\alpha^{f_1} \dot{\oslash} \beta\right)(d) + \left(\alpha^{f_2} \dot{\oslash} \beta\right)(d). \end{aligned}$$

■

The composition rule of $\dot{\oslash}$ follows from $f \oslash g \oslash h = f \oslash (g \otimes h)$ [1] by an argumentation similar to Lemma 12.

Lemma 13. *(Composition of $\dot{\oslash}$) For $f, g, h \in \mathcal{F}_0$ it holds that*

$$f \dot{\oslash} g \dot{\oslash} h \;=\; f \dot{\oslash} (g \otimes h).$$

SensorNC operates on curves shaped as required by the distributivity. This allows to generalize the concatenation theorem to SensorNC's sink-tree networks. Table I summarizes the notation required to precisely quantify all parameters involved.

Theorem 14. *(SensorNC Concatenation Theorem) Consider a set of flows F, $|F| = n$, with arrival curves $\alpha^{f_1}, \ldots, \alpha^{f_n} \in \mathcal{F}_{\text{TB}}$ that originate in a sink tree. For the purpose of their aggregate output bound calculation, the share of service offered to each flow $f \in F$ from its source to the* sink *within this aggregate is the concatenation of the service on its path. Then, the entire flow aggregate's output is bounded by*

$$\alpha'_{\text{sink}} \;=\; \sum_{f \in F(\text{sink})} \left(\alpha^f \dot{\oslash} \bigotimes_{i=0}^{L(f,\text{sink})} \beta_{P(f,i)} \right).$$

Quantifier	Definition
foi	Flow of interest
sink	Sink node of a (sub)tree
α^f	Arrival curve of flow f
$F(s)$	Set of flows at server s
$F_{\text{src}}(s)$	Set of flows originating in s
α^f_s, α^F_s	Arrival bound of flow f, set of flows F at server s
α_s	Abbreviation for $\alpha_s^{F(s)}$
$x(f)$, $x(F)$	All cross-traffic of flow f, set of flows F
$\bar{x}(f,s)$	Newly merging cross-traffic of f at server s
$P(f)$	Path of flow f
$P(f,i)$	Server at location (index) i on f's path
$L(f,s)$	Location (index) of server s on f's path $P(f)$
β_s	Service curve of server s
$\beta^{\text{l.o.}f}$	Left-over service curve for flow f

Table I
SENSOR NETWORK CALCULUS NOTATION.

Applying Corollary 11, we can rephrase the equation to

$$\alpha'_{\text{sink}} \;=\; \gamma_{r'_{\text{sink}},\, b'_{\text{sink}}}$$

with

$$r'_{\text{sink}} \;=\; \sum_{f \in F(\text{sink})} r^f$$

$$b'_{\text{sink}} \;=\; \sum_{f \in F(\text{sink})} \left(b^f + r^f \cdot \sum_{i=1}^{L(f,\text{sink})} T_{P(f,i)} \right).$$

Proof: First, we apply Corollary 10 to derive the tree's output from its sink's input flows.

$$\alpha'_{\text{sink}} \;=\; \sum_{f \in F(\text{sink})} \left(\alpha^f_{\text{sink}} \dot{\oslash} \beta_{\text{sink}} \right)$$

Next, we virtually separate all flows from each other and establish a tandem topology in their point of view. We recursively apply Corollary 10, Lemma 12 and Lemma 13 to the subtree defining the involved output arrival bounds. Note, that every server sees all flows crossing the subtree above it due to the lack of demultiplexing in sink trees.

We start with the separation of a single flow f:
Without loss of generality assume $L(f, \text{sink}) \geq 3$. To navigate through the sink tree, we use the function $up(F,s,i) = \bigcup_{f \in F} P(f, L(f,s) - i)$, $i \in \mathbb{N}^+$, that returns the set of servers that are i hops upstream from server s, i.e., further away from the sink, and are traversed by flows in F. Also let $s_i^f = up(\{f\}, s, i)$ and sink $= s_0$.

$$\begin{aligned} \alpha'_{\text{sink}} &= \alpha'_{s_0} \\ &= \sum_{f \in F(s_0)} \left(\alpha^f_{s_0} \dot{\oslash} \beta_{s_0}\right) \\ &= \alpha_{s_1^f} \dot{\oslash} \beta_{s_1} \dot{\oslash} \beta_{s_0} \\ &\quad + \sum_{s_1 \in up(F(s_0),s_0,1) \setminus s_1^f} \left(\alpha_{s_1} \dot{\oslash} \beta_{s_1}\right) \dot{\oslash} \beta_{s_0} + \alpha^{F_{\text{src}}(s_0)} \dot{\oslash} \beta_{s_0} \\ &= \left(\alpha^f_{s_1^f} + \alpha^{F(s_1^f)\setminus\{f\}}_{s_1^f}\right) \dot{\oslash} \beta_{s_1} \dot{\oslash} \beta_{s_0} \\ &\quad + \left(\sum_{s_1 \in up(F(s_0),s_0,1) \setminus s_1^f} \left(\alpha_{s_1} \dot{\oslash} \beta_{s_1}\right) + \alpha^{F_{\text{src}}(s_0)} \right) \dot{\oslash} \beta_{s_0} \end{aligned}$$

$$
\begin{aligned}
&= \alpha^f_{s_1^f} \dot{\oslash} \beta_{s_1} \dot{\oslash} \beta_{s_0} + \alpha^{F(s_1^f)\setminus\{f\}}_{s_1^f} \dot{\oslash} \beta_{s_1} \dot{\oslash} \beta_{s_0} \\
&\quad + \left(\sum_{s_1 \in up(F(s_0),s_0,1)\setminus s_1^f} \left(\alpha_{s_1} \dot{\oslash} \beta_{s_1}\right) + \alpha^{F_{\text{src}}(s_0)} \right) \dot{\oslash} \beta_{s_0} \\
&= \alpha^f_{s_1^f} \dot{\oslash} \beta_{s_1} \dot{\oslash} \beta_{s_0} + \alpha^{F(s_0)\setminus\{f\}}_{s_0} \dot{\oslash} \beta_{s_0} \\
&= \alpha^f_{s_1^f} \dot{\oslash} \left(\beta_{s_1} \otimes \beta_{s_0}\right) + \alpha^{F(s_0)\setminus\{f\}}_{s_0} \dot{\oslash} \beta_{s_0} \\
&\dots \\
&= \alpha^f_{s_2^f} \dot{\oslash} \bigotimes_{i=0}^{2} \beta_{s_i^f} + \alpha^{F(s_0)\setminus\{f\}}_{s_0} \dot{\oslash} \beta_{s_0} \\
&\dots \\
&= \alpha^f \dot{\oslash} \bigotimes_{i=0}^{L(f,s_0)} \beta_{s_i^f} + \alpha^{F(s_0)\setminus\{f\}}_{s_0} \dot{\oslash} \beta_{s_0}
\end{aligned}
$$

Next, we repeat the separation for the remaining flows in $F(\text{sink})$ and get

$$
\alpha'_{\text{sink}} = \sum_{f \in F(\text{sink})} \left(\alpha^f \dot{\oslash} \bigotimes_{i=0}^{L(f,\text{sink})} \beta_{P(f,i)} \right).
$$

The SensorNC specific calculation follows from Corollaries 4, 10 and 11:

$$
\begin{aligned}
\alpha'_{\text{sink}} &= \gamma_{r'_{\text{sink}}, b'_{\text{sink}}} \\
&= \sum_{f \in F} \gamma'_{r^f_{\text{sink}}, b^f_{\text{sink}}} \\
&= \gamma_{\sum_{f\in F}\left(r^f_{\text{sink}}\right)', \sum_{f\in F}\left(b^f_{\text{sink}}\right)'}
\end{aligned}
$$

with

$$
\sum_{f \in F(\text{sink})} \left(b^f_{\text{sink}}\right)' = \sum_{f \in F(\text{sink})} \left(b^f + r^f \cdot \sum_{i=0}^{L(f,\text{sink})} T_{P(f,i)} \right).
$$

■

The simple tandem topology lacking any nesting of flows is a special sink-tree network where the $n : 1$ input/output relation is instantiated with $n = 1$. In this case, the generalized concatenation theorem for SensorNC reduces to

$$
\alpha'_{\text{sink}} = \alpha^f \dot{\oslash} \bigotimes_{i=0}^{L(f,\text{sink})} \beta_{P(f,i)},
$$

i.e., the output bound of Theorem 7 with the concatenated service curve for tandems from Theorem 9.

The virtual separation establishes what we call *flow-locality* within the derivation (see Figure 3; transition from (a) to (b)). Compared to the previous computation, as shown in Figure 1(a), it is not required to bound a flow's cross-traffic arrivals recursively. This novel property allows for a simple two-tier iteration over the cross-flows and their respective paths in order to calculate a sink tree's output arrival curve.

IV. Improving Sensor Network Calculus Analysis

In this section, we demonstrate the use of the new theorem in the state-of-the-art network analysis for SensorNC, the Pay Multiplexing Only Once (PMOO).

PMOO is a complex composition of the operations given in Section II with the aim to provide an end-to-end left-over service curve for the flow of interest of the analysis. Theorem 8 is then used to derive the flow's performance bounds. Note, that the PMOO analysis is designed for servers that preserve the order of data within a single flow but arbitrarily multiplex the data of different incoming flows.

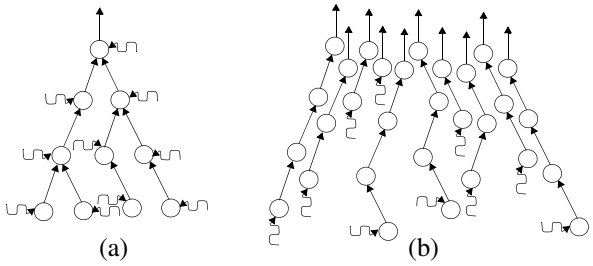

Figure 3. Sink-tree network with flow aggregation (a) converted to a set of flow-local views by Theorem 14 (b).

Theorem 15. *(PMOO $\beta^{\text{l.o. foi}}$ in SensorNC [11]) The PMOO end-to-end left-over service curve for a specific flow of interest, $\beta^{\text{l.o. foi}}$, is given by*

$$
\beta^{\text{l.o. foi}} = \beta_{R^{\text{l.o. foi}},\, T^{\text{l.o. foi}}}
$$

with

$$
\begin{aligned}
R^{\text{l.o. foi}} &= \bigwedge_{s \in P(\text{foi})} \left(R_s - r_s^{x(\text{foi})} \right) \\
T^{\text{l.o. foi}} &= \sum_{s \in P(\text{foi})} \left(T_s + \frac{b^{\bar{x}(\text{foi},s)} + r_s^{x(\text{foi})} \cdot T_s}{R^{\text{l.o. foi}}} \right)
\end{aligned}
$$

where all service curves on $P(\text{foi})$ are required to be strict.

As already shown in Figure 1(b), the most expensive part of a SensorNC analysis is bounding arrivals of all cross-traffic flows, $\alpha_s^{x(\text{foi})}$, and the newly merging cross-traffic, $\alpha^{\bar{x}(\text{foi},s)}$, along the flow of interest's path. To derive a tight bound it is necessary to correctly account for all flow transformations within the subnetwork abstracted by this analysis step. Therefore, knowledge about the topology, nodal service, flow transformations and flow merging locations is required. For that reason, cross-traffic arrival bounding has so far been done by an expensive recursion over the topology – from the server on the flow of interest's path towards each cross-flow's sink.

Lemma 16. *(Recursive Aggregate Arrival Bounding) Without loss of generality we assume that server s has at least one subtree with at least two levels. To navigate through this sink tree, we use the function $up(F, s, i) = \bigcup_{f\in F} P(f, L(f, s) - i)$, $i \in \mathbb{N}^+$, that returns the set of servers that are i hops upstream from s, i.e, further away from the* sink*, and are traversed by the flows in F. Also let $s_i^f = up(\{f\}, s, i)$, $s_0 = s$ and $n = \max\{f \in F \mid L(f, s)\}$. Then the recursive aggregate arrival bound is derived as*

$$
\begin{aligned}
\alpha_s &= \sum_{s_1 \in up(F(s),s,1)} \left(\alpha_{s_1} \dot{\oslash} \beta_{s_1}\right) + \alpha^{F_{\text{src}}(s)} \\
&= \sum_{s_1 \in up(F(s),s,1)} \Bigg(\\
&\qquad \sum_{s_2 \in up(F(s_1),s_1,1)} \left(\alpha_{s_2} \dot{\oslash} \beta_{s_2}\right) + \alpha^{F_{\text{src}}(s_1)} \\
&\qquad \Bigg) + \alpha^{F_{\text{src}}(s)}
\end{aligned}
$$

$$
\begin{aligned}
&\dots \\
&= \sum_{s_1 \in up(F(s),s,1)} (\dots \\
&\quad \left(\sum_{s_n \in up(F(s_{n-1}),s_{n-1},1)} \left(\alpha_{s_n} \dot{\oslash} \beta_{s_n}\right) + \alpha^{F_{\text{src}}(s_{n-1})} \right) \\
&\quad \dots \dot{\oslash} \beta_{s_1}) + \alpha^{F_{\text{src}}(s)}
\end{aligned}
$$

When the topology is traversed, intermediate arrival bounds have to be computed at every server where multiple flows merge. This needs to be done recursively, again.

In contrast to this recursive cross-traffic arrival bounding, we can exploit the generalized concatenation theorem for SensorNC. Yet, we first need to extend its capabilities to derive bounds for subsets of flows in order to fulfill PMOO's need to separate the flow of interest from other flows.

Corollary 17. *We can apply Theorem 14 to a subset of flows* $F \subseteq F\,(\text{sink})$ *with*

$$
\left(\alpha^{F}{}_{\text{sink}}\right)' = \sum_{f \in F} \left(\alpha^f \dot{\oslash} \bigotimes_{i=0}^{L(f,\text{sink})} \beta_{P(f,i)} \right)
$$

if the remaining flows in $F(\text{sink}) \setminus F$ *have a lower priority than those in* F.

For the PMOO analysis, the flow of interest is always considered to have the lowest priority among all flows. This assumption preserves the worst-case semantic of (sensor) network calculus. In order to account for it, we use Corollary 17 to separate cross-traffic from the flow of interest. Its bounds can then be derived with the left-over service of Theorem 15 remaining after the cross-flows have been served.

We can now use the generalized concatenation theorem for cross-traffic arrival bounding:

Theorem 18. *(SensorNC Arrival Bounding) Given the flow of interest, we can derive its cross-traffic arrival bound at any server as follows:*

$$
\alpha_s^{x(\text{foi})} = \sum_{f \in x(\text{foi}) \setminus F_{\text{src}}(s)} \left(\alpha^f \dot{\oslash} \bigotimes_{i=1}^{L(f,s)} \beta_{P(f,i)} \right) + \alpha^{F_{\text{src}}(s) \cap x(\text{foi})}
$$

Applying Theorem 8, we can rephrase the equation to

$$
\alpha_s^{x(\text{foi})} = \gamma_{r_s^{x(\text{foi})}, b_s^{x(\text{foi})}}
$$

with

$$
r_s^{x(\text{foi})} = \sum_{f \in x(\text{foi})} r^f
$$

$$
b_s^{x(\text{foi})} = \sum_{f \in x(\text{foi}) \setminus F_{\text{src}}(s)} \left(b^f + r^f \cdot \sum_{i=1}^{L(f,s)} T_{P(f,i)} \right) + b^{F_{\text{src}}(s) \cap x(\text{foi})}.
$$

Proof: The cross-traffic aggregate at , $\alpha_s^{x(\text{foi})}$, consists of the sum of all cross-flows arriving from the subtrees upstream from it as well as the cross-flows originating at it.

$$
\begin{aligned}
\alpha_s^{x(\text{foi})} &= \sum_{s_1 \in up(F(s),s,1)} \left(\sum_{f \in F(s_1) \cap x(\text{foi})} \left(\alpha^f \dot{\oslash} \bigotimes_{i=0}^{L(f,s_1)} \beta_{P(f,i)} \right) \right) \\
&\quad + \alpha^{F_{\text{src}}(s) \cap x(\text{foi})} \\
&= \sum_{f \in x(\text{foi}) \setminus F_{\text{src}}(s)} \left(\alpha^f \dot{\oslash} \bigotimes_{i=1}^{L(f,s)} \beta_{P(f,i)} \right) + \alpha^{F_{\text{src}}(s) \cap x(\text{foi})}
\end{aligned}
$$

■

Note that we can derive the newly merging cross-traffic $\alpha^{\bar{x}(\text{foi},s)}$ in the same way, yet, without the need of having Corollary 17. By definition, the flows in $\alpha^{\bar{x}(\text{foi},s)}$ do not share a previous server with the flow of interest. Thus, there is no need to separate it from its cross-traffic as they never interfered upstream of s.

Using the new bounding method from Theorem 18 instead of the conventional one from Lemma 16 has several practical advantages when applied in cross-traffic arrival bounding. We conclude this section by discussing the most important advantages before evaluating their impact in Section V:

Faster Computation. The arrival bounds for the entire cross-traffic and for newly merging cross-flows, $\alpha_s^{x(\text{foi})}$ and $\alpha^{\bar{x}(\text{foi},s)}$, both depend on the flow of interest. Thus, in a self-modeling WSN a sensor node needs to compute both bounds for each flow to keep track of their state. With Theorem 18 we reduce the computational effort of arrival bounding by establishing flow-locality. The aggregation of individual cross-flow arrival bounds is *virtually* shifted from locations in a subtree to the server the bound is required. Thus, only flow-local results need to be aggregated – this allows for reuse in the derivation of a flow's cross-traffic arrival bounds. In contrast, due to the interweavement of the recursive Lemma 16 with the topology, it was generally not possible to share any results between the derivations of $\alpha_s^{x(\text{foi})}$ and $\alpha^{\bar{x}(\text{foi},s)}$ for different flows of interest.

Lower Communication Overhead. Flow-locality also enables to overcome the need for an additional protocol such as Deluge [20] distributing the information required to derive $\alpha_s^{x(\text{foi})}$ and $\alpha^{\bar{x}(\text{foi},s)}$. If virtually separated, a flow's arrival at a server can be calculated hop-by-hop without compromising tightness (Lemma 13). A flow can carry information about its current arrival bound as payload, pushing the information to all sensors concerned. Thus, the state is updated on demand, i.e., independent of a polling interval, resulting in much less communication.

Quick Reaction to Changes. The flow-locality also affects the recomputation effort in case of parameter modifications. Using the conventional method based on Lemma 16, locality of a modified parameter did not matter much due to the complex setting of flow aggregation locations; a change to a single parameter always invalidated a large amount of the derivation's intermediate results and triggered expensive recomputations, usually of the entire subtree. Theorem 18 prevents such an invalidation from spreading to flows not directly affected by a change, e.g., flows not crossing a sensor that adapted its rate, and thus enables a quick reaction to changes.

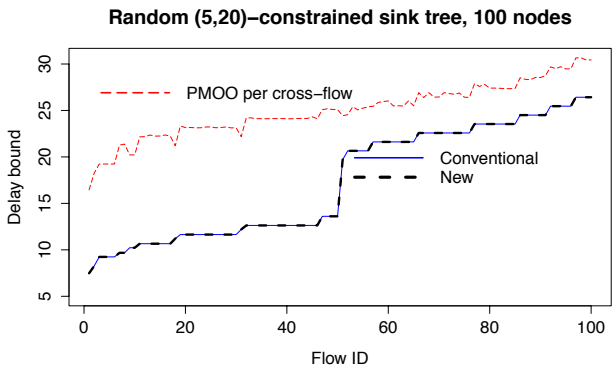

Figure 4. Delay bounds using different cross-traffic arrival bounding methods.

V. Evaluation

We have extended the DISCO Deterministic Network Calculator (DiscoDNC) [14] with the generalized concatenation theorem in order to benchmark our results against existing analyses. For that, we randomly created (o, d)-constrained sink trees [19] with varying size, *maximum outdegree* o and *maximum depth* d.

A. Tightness of Bounds

First, we investigate the impact of the generalized concatenation theorem on the tightness of bounds. We use the delay bound as measure and benchmark the PMOO analysis using Theorem 18 ("New") against those using either recursive arrival bounding ("Conventional") or executing a separate PMOO analysis for each cross-flow. The latter was chosen because it also results in bounds with (restricted) flow-locality – only the reuse of per-flow arrival bounds is permitted, nothing more. Figure 4 shows the resulting delay bounds for a random $(5, 20)$-constrained homogeneous sink tree with 100 servers, $\alpha = \gamma_{1,1}$ and $\beta = \beta_{75,1}$.

Despite being flow-local, the generalized concatenation-based arrival bounding preserves the tightness of the recursive version whereas per-cross-flow PMOO bounding results in a significant increase of delay bounds. The first observation is a consequence of Lemma 12 and Theorem 14. The latter one, PMOO per-cross-flows bounding's inferiority, is due to the following: There is no prioritization among flows in a cross-traffic aggregate. Therefore, they need to be considered to interfere with each other in the worst possible way when applying PMOO naively to arrival bounding. That leads to overly pessimistic cross-traffic arrivals and thus loose bounds. The following corollary confirms this reasoning.

Corollary 19. *For any partition* $F_1(s), \ldots, F_n(s)$, $n \in \mathbb{N}^+$, *of flows crossing a server* s, *SensorNC possesses the property*

$$\sum_{i=1}^{n} \left(\alpha_s^{F_i(s)} \dot{\oslash} \beta_s^{\text{l.o. } F_i(s)} \right) \geq \left(\sum_{i=1}^{n} \alpha_s^{F_i(s)} \right) \dot{\oslash} \beta_s.$$

Proof: Without loss of generality assume a single server s with service curve β_s that is traversed by two flows f_1 and f_2 with arrival curves $\alpha_s^{f_1}$ and $\alpha_s^{f_2}$, respectively. We know that the left-over service cannot exceed the original service, i.e., $\forall f, s \; \beta^{\text{l.o. } f} \leq \bigotimes_{s \in P(f)} \beta_s$, and from Lemma 12 it follows that

$$\sum_{i=1}^{2} \left(\alpha_s^{f_i} \dot{\oslash} \beta_s^{\text{l.o. } f_i} \right) \geq \sum_{i=1}^{2} \left(\alpha_s^{f_i} \dot{\oslash} \beta_s \right) = \left(\sum_{i=1}^{2} \alpha_s^{f_i} \right) \dot{\oslash} \beta_s.$$

■

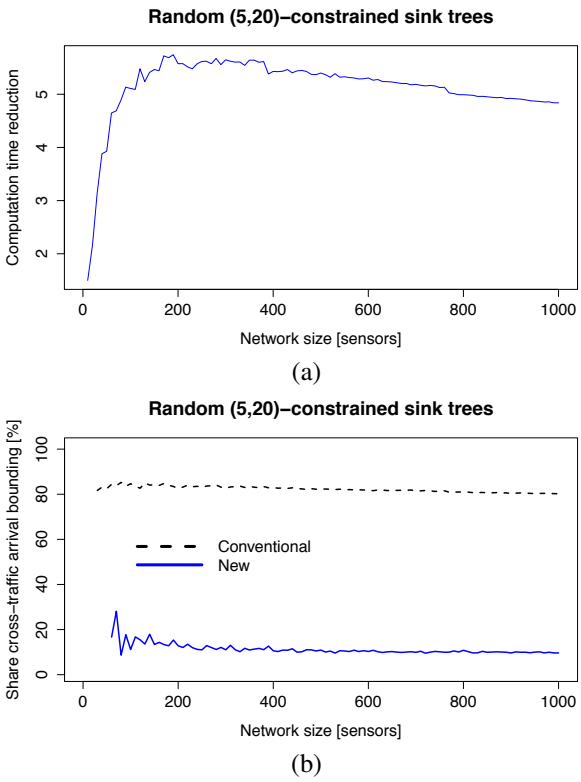

Figure 5. (a) Computation time reduction and (b) share of the arrival bounding on the overall computation time.

B. Computational Effort

SensorNC analyses have usually been carried out by a central entity due to the knowledge about and control over the information needed to derive bounds. The most prominent example is a design space exploration evaluating the effect of different system configurations on the performance. We investigate the impact of Theorem 18 on the execution time of an end-to-end analysis of all flows in the network. All servers offer $\beta_{1000,1}$ and generate a flow shaped with $\alpha_{0.001,0.1}$ in order to guarantee for stability. We have created 40 random (o, d)-constrained sink trees per network size. The results constitute the average over the respective simulations.

Relative Improvements: Figure 5(a) shows the factor by which computation times are reduced. Starting with small network sizes, the effort needed to bound cross-traffic arrivals naturally increases with the number of servers and therefore the gain by our new method is getting bigger, too. However, after this initial phase it converges to a value of ≈ 5.7. Figure 5(b) illustrates the according reduction of the arrival bounding's share of the overall analysis: From 80 to less than 15%. For small random sink trees the reduction highly depends on the actual shape of the topology which explains the small oscillation in the graph.

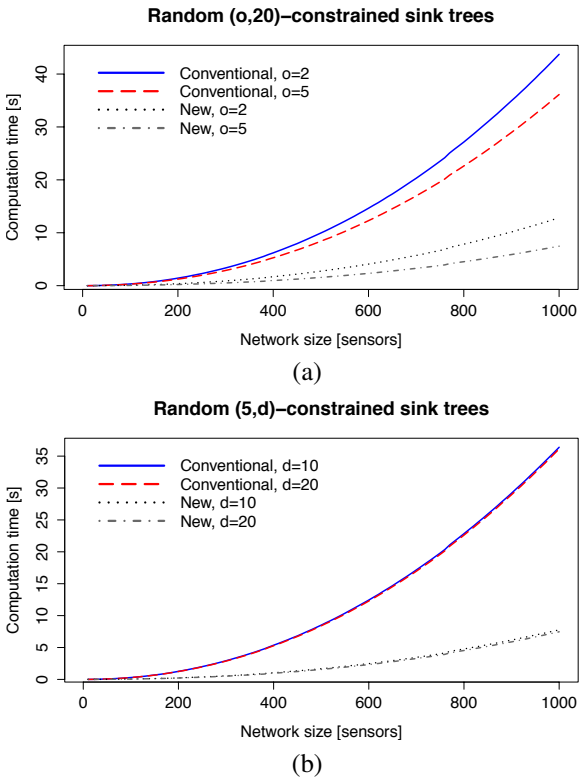

Figure 6. Scaling behavior when altering (a) the maximum out degree o and (b) the maximum depth d.

Absolute Improvements: Figure 6 compares the scaling behavior of the superior PMOO analyses. Both react similar to changes of o (Figure 6(a)) and d (Figure 6(b)). While our new method's run time still scales proportionally to the network size, it does so much slower. This is probably the best one could hope for when improving SensorNC analysis without compromising the tightness of the results.

Note that Figure 6 shows the computation times for the analysis of a single network configuration averaged over 40 repetitions. When varying the configuration it is not only sensible to experiment with different values for o and d or test multiple deployments complying with the resulting (o, d)-constraint, but also to test heterogeneous network configurations by varying the two defining parameters of flows and sensors independently. Thus, a design space exploration can easily consist of tens of thousands of distinct analyses to be executed. The overall execution time clearly has to scale in this number.

C. SensorNC as a Distributed Service

A PMOO analysis with recursive bounding of cross-traffic arrivals practically does not allow to distribute the execution over the sensor network as it would essentially be equal to the centralized execution. It thus imposes the need to gather all the required information at each sensor; a characteristic we conceptually evaluate in this section. The concatenation-based arrival bounding method, however, can prevent this overhead while simultaneously benefiting from the reduced computational effort. The flow-locality allows for a straightforward distribution of the required information:

- Flows carry their current arrival bound as payload such that it eventually reaches all the servers the flow traverses.
- Servers store each incoming flow's arrival bound and then adapt it according to the provided service curve.

To provide the information required in the left-over service curve derivation of Theorem 15 and the performance bound derivation of Corollary 11, some additional information in the payload of a flow suffices:

- The residual service rate $R^{\text{l.o. foi}}$ on its previous path; crossed servers adapt it if necessary and store the value.
- The sum of latencies T experienced thus far; crossed servers simply add their latency and store the value.
- The sum of merging cross-traffic bursts $b^{\bar{x}(\text{foi},s)}$ up to the current server; servers add the local value.
- Every flow needs to carry its original arrival curve in order to derive its performance bounds.

This way every server is provided with all information.

Next, we evaluate the impact of the above scheme. We consider a self-modeling WSN providing a task admission control scheme based on delay bounds. As soon as a new task is supposed to be added to the network, the WSN checks if it can schedule the task's data flow without compromising any other flow's delay constraint. In order to do so, it is necessary to derive each flow's delay bound under the hypothetical new configuration. For simplicity of the comparison between the recursive and the concatenation-based PMOO analysis, we assume a fully occupied (o, d)-constrained sink tree when evaluating the addition of a task at its root.

In such a sink tree, there are $N := \frac{o^{d+1}-1}{o-1} - 1$ nodes above the root, all of which hold information necessary to derive delay bounds. Whereas the above scheme distributes this information during normal operation, the previous, recursive scheme requires two preceding phases to acquire it. In the request phase, there is communication to N servers to query their parameter settings and in the reporting phase N flows answer the query (Figure 7). Note that these $2 \cdot N$ temporary flows interfere with the existing traffic. They therefore need to be scheduled at a lower priority so that they do not force existing flows to violate their deadlines. Thus, termination of the two phases is not guaranteed. In the new concatenation-based scheme there is no such *communication overhead*.

Second, we compare the *storage demand* at a node. In the recursive PMOO analysis, it was required to store all N service curves of a subtree's nodes, N arrival curves of the flows originating in it and the N output arrival curves of flow aggregates at each sensor node. In order to derive the latter values, it is also necessary to have extensive knowledge about the topology: Where do flows originate? Which paths do flows take? Where do which flows aggregate? (see Figure 3(a)) In contrast, the new analysis requires the arrival curves of all N flows, their arrival bounds at the server as well as their $R^{\text{l.o.}}$ and $\sum_{s \in P(f)} T_s$ (which equals the recursive PMOO's storage demand for service curves). Additionally, a server has to store the sum of each flow f's $b^{\bar{x}(f)}$ along its path to execute

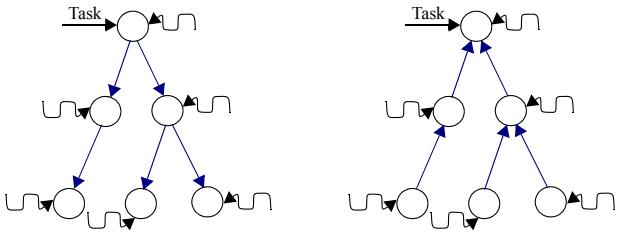

Figure 7. Task admission control in a $(2, 3)$-constrained sink tree: (a) Request phase and (b) reporting phase.

Theorem 15. Further, the recursive analysis needs to store the arrival bounds $\alpha_s^{x(f)}$ and $\alpha^{\bar{x}(f,s)}$, for the entire cross-traffic and for newly merging cross-flows, respectively, for each flow f at server s. In the new analysis, this is not necessary as both values are simply derived by adding up already stored values. Thus, the new scheme considerably reduces the storage demand to execute a PMOO analysis.

Regarding the *computational effort*, a sensor node previously had to bound cross-traffic arrivals at all non-leaf nodes above it, i.e., execute $N - o \cdot d$ operations, and compute the output arrival curve at all upstream nodes [2], [18], i.e., execute another N operations. In the new scheme, a node only computes the output arrival curve of flows crossing it, i.e., it just executes N operations. In the second analysis step, deriving PMOO's arrival bounds, the computational effort is reduced from a repetitive execution of the operations for each flow, considering the unique aggregation on its path that defines $\alpha_s^{x(f)}$ and $\alpha^{\bar{x}(f,s)}$, to adding up the per-flow bounds that were derived only once. This reduction in complexity of the analysis leads to faster computation times as discussed in Section V-B. The effort to execute Theorem 15 stays the same.

With the generalized concatenation theorem we can keep the communication overhead low, decrease storage demands and reduce computational effort. All of this is achieved with a simple scheme that allows to distribute the analysis over the entire network.

VI. Conclusion

Of the two steps involved in network calculus performance analysis, bounding a certain flow of interest's cross-traffic arrivals and deriving its delay and backlog bound, we focus on the former one in this paper. Although much effort has so far been invested in the latter one, we showed that typically it only consumes around 20% of the computation time. The high effort of cross-traffic arrival bounding is caused by the degree it is interwoven with the topology it is applied to. In order to break this dependency, we contribute a novel theorem that generalizes the classical concatenation theorem – which is only applicable to tandems of servers – to enable for sink tree analysis. The generalized concatenation theorem for sensor network calculus allows to bound arrivals of individual cross-flows virtually separated from each other and from the topology. Without compromising the quality of the bounds, the separate results can be combined to the whole cross-traffic aggregate's arrival bound. This characteristic introduces a so-called flow-locality that leads to considerably reduced complexity and thus resource demand of the entire analysis and also boosts resilience against recalculations caused by parameter changes. We could decrease the share of arrival bounding from 80% to less than 15% and thereby reduce the overall time to completion of the analysis by more than a factor of 5. The new flow-locality further provides the interesting opportunity to distribute a network calculus analysis across a sensor network in order to execute performance control and monitoring tasks such as distributed admission control for large-scale WSNs inside the network itself.

References

[1] J.-Y. Le Boudec and P. Thiran, *Network Calculus: A Theory of Deterministic Queuing Systems for the Internet*. Springer, 2001.

[2] J. B. Schmitt and U. Roedig, "Sensor Network Calculus – A Framework for Worst Case Analysis," in *Proc. IEEE DCOSS*, June 2005, pp. 141–154.

[3] J. B. Schmitt, F. A. Zdarsky, and U. Roedig, "Sensor Network Calculus with Multiple Sinks," in *Proc. of the Performance Control in Wireless Sensor Networks Workshop at the IFIP NETWORKING*, May 2006, pp. 6–13.

[4] J. B. Schmitt, F. A. Zdarsky, and L. Thiele, "A Comprehensive Worst-Case Calculus for Wireless Sensor Networks with In-Network Processing," in *Proc. IEEE RTSS*, December 2007, pp. 193–202.

[5] N. Gollan and J. B. Schmitt, "Energy-Efficient TDMA Design Under Real-Time Constraints in Wireless Sensor Networks," in *Proc. IEEE/ACM MASCOTS*, October 2007, pp. 80–87.

[6] A. Koubâa, M. Alves, and E. Tovar, "Modeling and Worst-Case Dimensioning of Cluster-Tree Wireless Sensor Networks," in *Proc. IEEE RTSS*, December 2006, pp. 412–421.

[7] P. Jurcik, A. Koubâa, R. Severino, M. Alves, and E. Tovar, "Dimensioning and Worst-Case Analysis of Cluster-Tree Sensor Networks," *ACM Transactions on Sensor Networks*, pp. 14:1–14:47, September 2010.

[8] H. She, Z. Lu, A. Jantsch, D. Zhou, and L.-R. Zheng, "Performance Analysis of Flow-Based Traffic Splitting Strategy on Cluster-Mesh Sensor Networks," *International Journal of Distributed Sensor Networks*, 2012.

[9] W. Y. Poe, M. A. Beck, and J. B. Schmitt, "Achieving High Lifetime and Low Delay in Very Large Sensor Networks using Mobile Sinks," in *Proc. IEEE DCOSS*, May 2012, pp. 17–24.

[10] J. B. Schmitt, F. A. Zdarsky, and I. Martinovic, "Improving Performance Bounds in Feed-Forward Networks by Paying Multiplexing Only Once," in *Proc. GI/ITG MMB*, March 2008, pp. 1–15.

[11] J. B. Schmitt, F. A. Zdarsky, and M. Fidler, "Delay Bounds under Arbitrary Multiplexing: When Network Calculus Leaves You in the Lurch ..." in *Proc. IEEE INFOCOM*, April 2008, pp. 1669–1677.

[12] A. Bouillard, L. Jouhet, and E. Thierry, "Tight Performance Bounds in the Worst-Case Analysis of Feed-Forward Networks," in *Proc. IEEE INFOCOM*, March 2010, pp. 1–9.

[13] J. B. Schmitt and F. A. Zdarsky, "The DISCO Network Calculator – A Toolbox for Worst Case Analysis," in *Proc. ValueTools*, October 2006.

[14] S. Bondorf and J. B. Schmitt, "The DiscoDNC v2 – A Comprehensive Tool for Deterministic Network Calculus," in *Proc. ValueTools*, December 2014.

[15] F. Baccelli, G. Cohen, G. J. Olsder, and J.-P. Quadrat, *Synchronization and Linearity: An Algebra for Discrete Event Systems*. Wiley, 1992.

[16] C.-S. Chang, *Performance Guarantees in Communication Networks*. Springer, 2000.

[17] S. Bondorf and J. B. Schmitt, "Statistical Response Time Bounds in Randomly Deployed Wireless Sensor Networks," in *Proc. IEEE LCN*, October 2010, pp. 340–343.

[18] U. Roedig, N. Gollan, and J. B. Schmitt, "Validating the Sensor Network Calculus by Simulations," in *Proc. Performance Control in Wireless Sensor Networks Workshop at WICON*, October 2007.

[19] J. B. Schmitt and U. Roedig, "Worst Case Dimensioning of Wireless Sensor Networks under Uncertain Topologies," in *Proc. Workshop on Resource Allocation in Wireless Networks at IEEE WiOpt*, April 2005.

[20] J. W. Hui and D. Culler, "The dynamic behavior of a data dissemination protocol for network programming at scale," in *ACM SenSys*, November 2004, pp. 81–94.