HAL open science

Modeling Wireless Sensor Networks Using Finite-Source Retrial Queues with Unreliable Orbit

Patrick Wüchner, János Sztrik, Hermann De Meer

To cite this version:

Patrick Wüchner, János Sztrik, Hermann De Meer. Modeling Wireless Sensor Networks Using Finite-Source Retrial Queues with Unreliable Orbit. Performance Evaluation of Computer and Communication Systems (PERFORM), Oct 2010, Vienna, Austria. pp.73-86, 10.1007/978-3-642-25575-5_7 . hal-01586895

HAL Id: hal-01586895

https://inria.hal.science/hal-01586895

Submitted on 13 Sep 2017

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution 4.0 International License

Modeling Wireless Sensor Networks Using Finite-Source Retrial Queues with Unreliable Orbit*

Patrick Wüchner[1], János Sztrik[2], and Hermann de Meer[1]

[1] Faculty of Informatics and Mathematics,
University of Passau, Innstraße 43,
94032 Passau, Germany,
patrick.wuechner@uni-passau.de

[2] Faculty of Informatics, University of Debrecen,
Egyetem tér 1. Po.Box 12, 4010 Debrecen, Hungary,
jsztrik@inf.unideb.hu

Abstract. Motivated by the need for performance models suitable for modeling and evaluation of wireless sensor networks, we introduce a retrial queueing system with a finite number of homogeneous sources, unreliable servers, orbital search, and unreliable orbit. All random variables involved in model construction are assumed to be independent and exponentially distributed. Providing a generalized stochastic Petri net model of the system, steady-state analysis of the underlying continuous-time Markov chain is performed and steady-state performance measures are computed by the help of the MOSEL-2 tool. The main novelty of this investigation is the introduction of an unreliable orbit and its application to wireless sensor networks. Numerical examples are derived to show the influence of sleep/awake time ratio, message dropping, and message blocking on the senor nodes' performance.

Keywords: performance evaluation, unreliable finite-source retrial queue, wireless sensor network, energy efficiency, self-organization

1 Introduction

Wireless sensor networks (WSNs, [1, 12]) are communication networks with harsh resource constraints. They need lightweight, energy-efficient, and self-organizing communication protocols.

In this paper, we propose a model that allows to discuss the trade-off between the energy efficiency and performance of WSNs by showing the positive and

* This research is partially supported by the German-Hungarian Intergovernmental Scientific Cooperation (HAS&DFG, 436 UNG 113/197/0-1), by the New Hungary Development Plan (TÁMOP 4.2.1./B-09/1/KONV-2010-0007), by the AutoI project (STREP, FP7 Call 1, ICT-2007-1-216404), by the ResumeNet project (STREP, FP7 Call 2, ICT-2007-2-224619), by the SOCIONICAL project (IP, FP7 Call 3, ICT-2007-3-231288), and by the EuroNF Network of Excellence (FP7, IST 216366).

negative effects of message dropping and blocking for variable sleep/awake time ratios. The model is based on retrial queueing systems [7] in which arriving jobs that find all servers unavailable do not line up in a queue, but join a so-called orbit. An orbit is a buffer from where the jobs retry to get service until they are successfully served. In contrast to ordinary queueing systems, the server(s) might be idle even if the buffer contains jobs.

Due to their broad practical applicability, e.g., in the field of communication networks, and due to their non-triviality, retrial queues have been receiving wide interest in the scientific community. The interested reader is referred to [7] for a recent introduction and summary of main methods, results, and applications.

Here, we focus on finite-source retrial queues. The arrival process is then non-Poisson and depends on the number of customers already staying at the system (see [7, p. 32]). While several variants of finite-source retrial queues have been studied in related work (e.g., in [2–4, 6, 17–19]), the authors are not aware of any discussion of the orbit's unreliability, not even in the infinite-source case.

Our main contribution is presenting and discussing a generalized model of finite-source retrial queues with *unreliable orbit* that also takes unreliable servers (cf. [5, 15, 3, 17]) and orbital search (cf. [8, 11, 13, 18, 19]) into account.

By the help of this model, we discuss the influence of the sleep/awake time ratio, message dropping, and message blocking on the senor nodes' mean response time, serving probability, and blocking probability.

The paper is organized as follows. We introduce the investigated WSN scenario in Sect. 2. In Sect. 3, we present the full model description in form of a generalized stochastic Petri net (GSPN), discuss the underlying continuous-time Markov chain (CTMC), and present the main performance measures. Numerical results, conveniently derived using the MOSEL-2 tool, are presented and their implications on WSN design are discussed in Sect. 4. We conclude by summarizing the paper and giving directions for future work in Sect. 5.

2 Use Case: Wireless Sensor Network

An example WSN scenario is shown in Fig. 1. Immobile sensor nodes (circles) are deployed in a two-dimensional (x, y) area. The nodes are labeled with their distance to the sink measured by the number of communication hops. The sink node (solid black circle) is located at coordinate $(7, 5)$.[1]

Due to harsh resource constraints and the resulting limited transmission range, each node i (located at (x_i, y_i)) is only able to communicate directly with its immediate neighbors, i.e., with all nodes j where $|x_j - x_i| \leq 1$ and $|y_j - y_i| \leq 2$. For example, the node located at $(6, 4)$ is able to exchange messages directly with the nodes located at $(6, 6)$, $(5, 5)$, $(5, 3)$, $(6, 2)$, $(7, 3)$, and also with the sink. We further assume that each node is aware of its own distance to the sink measured in the number of hops.

[1] The presented concepts also hold, mutatis mutandis, for mobile sensor nodes. The nodes' labels then need to be updated regularly. In this paper, however, we limit ourselves to immobile nodes for the sake of conciseness.

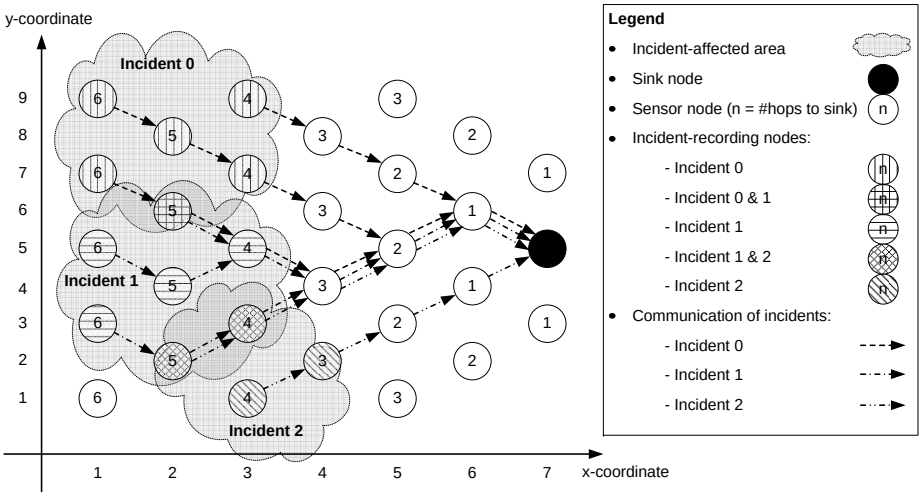

Fig. 1. WSN example scenario.

The purpose of the given sensor network is to monitor the covered area, record incidents, and communicate the appearance of incidents to the sink in a multi-hop fashion. In a real system, such incidents could be, e.g., the recognition of an intrusion, temperatures or humidity exceeding or under-running predefined thresholds, or the detection of fire, gas, vibration, movement, noise, etc. We assume that each node may be equipped with several sensors. Hence, each node may detect several but a finite number of distinct incidents.

In the example scenario sketched in Fig. 1, three incidents (Incidents 0, 1, and 2) can be detected by the sensor network. For example, node $(2, 6)$ detects Incidents 0 and 1, immediately generates a message for each incident, and tries to send both messages toward the sink. Node $(2, 6)$ has six neighbors of which two are closer to the sink: node $(3, 7)$ and node $(3, 5)$.

In principle, all neighbors can be aware of the transmission due to the broadcast character of the wireless air interface. Hence, each neighbor may serve as next-hop node. However, nodes are self-organizing and, for saving energy, may decline to receive new messages, store messages, or (re-)send messages depending on the node's energy status, the sender's distance to the sink, etc.

By sending an acknowledgment, a receiver agrees to accept the message and to take care of forwarding it further. If none of the neighbors accepts the message, node $(2, 6)$ stores it locally and *retries* to forward it later. If in the meanwhile node $(2, 6)$ receives further messages reporting the same incident, it merges the messages. Hence, each incident is only stored once at the node.

As soon as the message has successfully been transferred to the next hop, the latter takes care of forwarding the message toward the sink. Messages that reach the sink leave the sensor network.

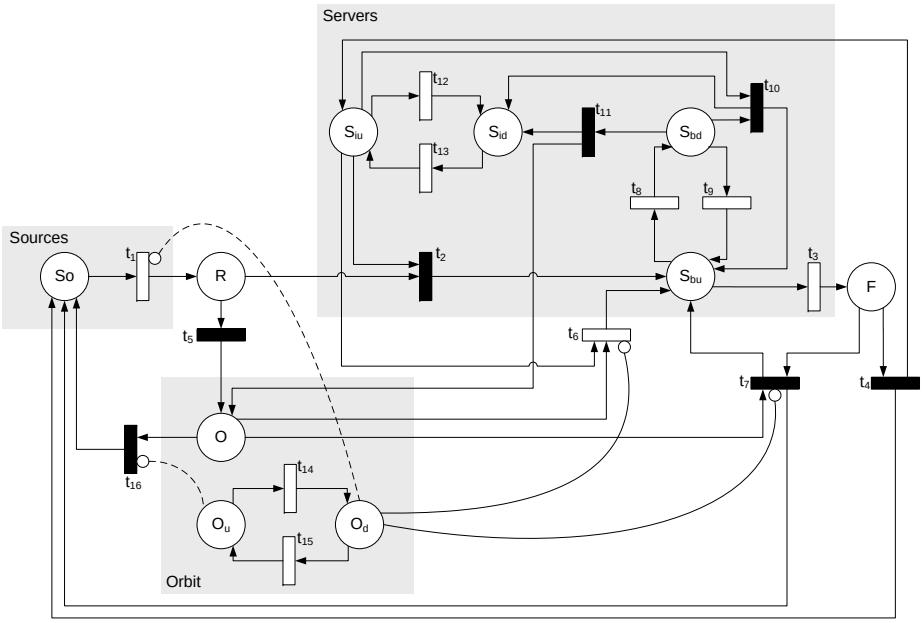

Fig. 2. GSPN of finite-source retrial queue with unreliable servers, unreliable orbit, and orbital search.

3 Model and Performance Measures

As a first step toward assessing the influence of system parameters on the performance and energy efficiency of the WSN scenario introduced in Sect. 2, we focus on a single node and its neighbors in this paper.

In this section, we present a model representing such a group of nodes. After constructing the model in the form of a GSPN (cf. [10, p. 64]) and discussing the underlying CTMC (cf. [10, p. 96]), performance measures are derived based on the model's steady-state probabilities.

3.1 Generalized Stochastic Petri Net Model

In Figure 2, the model is graphically represented in form of a GSPN.The model is a generalization of finite-source retrial queues with unreliable servers (cf. [3, 17]) and finite-source retrial queues with orbital search (cf. [18, 19]) by introducing an unreliable orbit, server hopping, and a variable number of server repairmen.

The main model parameters, the places' descriptions, and the transitions' functions are summarized in Tabs. 1, 2, and 3, respectively. Tab. 4 maps the use case's parameters to the model parameters and defines default values used as a basis for numerical evaluations in Sect. 4.

Note that not all model properties can be mapped to the graphical GSPN representation conveniently. For example, the dashed inhibitor arcs are subject

Table 1. Main model parameters.

Parameter	Symbol	Range
Number of sources	K	$\mathbb{N}$
Number of servers	S	$\mathbb{N}$
Number of repairmen	S_R	$\mathbb{N}$
Arrival rate	λ	$\mathbb{R}^+$
Service rate	μ	$\mathbb{R}^+$
Retrial rate	ν	$\mathbb{R}^+$
Search probability	p	$(0, 1)$
Busy server breakdown rate	δ_{Sb}	$\mathbb{R}^+$
Busy server repair rate	τ_{Sb}	$\mathbb{R}^+$
Idle server breakdown rate	δ_{Si}	$\mathbb{R}^+$
Idle server repair rate	τ_{Si}	$\mathbb{R}^+$
Orbit breakdown rate	δ_O	$\mathbb{R}^+$
Orbit repair rate	τ_O	$\mathbb{R}^+$
Sources blocked (servers down)[2]	β	$0, 1$
Sources blocked (orbit down)[2]	γ	$0, 1$
Server hopping[2]	σ	$0, 1$
Server flushing[2]	ϕ	$0, 1$
Orbit flushing[2]	ω	$0, 1$

Table 2. Places used in Fig. 2 with capacity (column "Cap.") and initial marking (column "i.M.").

ID	Description	Cap.	i.M.
So	Active sources	K	K
R	Incoming requests	1	0
S_{bu}	Servers, busy & up	S	0
S_{iu}	Servers, idle & up	S	S
S_{bd}	Servers, busy & down	S	0
S_{id}	Servers, idle & down	S	0
O	Orbit	K	0
O_u	Orbit up	1	1
O_d	Orbit down	1	0
F	Finished requests	1	0

to additional guard functions. Please refer to Tab. 3 (`IF` statements in column *Value*) for a comprehensive list of all guard functions that have to be considered.

The model consists of three main parts (gray boxes in Fig. 2): a finite set of sources, a finite set of servers, and the orbit component.

In our finite-source model, there are K sources represented by K Petri net tokens initially residing in place So. All tokens located in place So represent incidents that are currently not sensed by, reported to, or accepted by the node under investigation. New incidents arrive to the node with arrival rate λ per unreported incident (transition t_1). Remember that incoming reports of incidents that are already currently processed by the node do not imply new tasks which motivates the application of a finite-source model.

Arriving tokens enter place R from where they immediately try to enter the group of S servers. The node immediately tries to forward new messages (represented by tokens in place R) to one of the next hops, represented by the group of S servers. Each server might be idle and up (tokens in place S_{iu}), busy and up (S_{bu}), idle and down (S_{id}), or busy and down (S_{bd}). Busy servers represent next-hop nodes that are currently not able to receive incident messages since they are processing former messages (t_3) with rate μ. A server is considered down when the corresponding next-hop node is sleeping, i.e., in power-saving mode. Only servers that are up and idle are ready to receive tokens. If none of the servers (next hops) is idle and up (awake), arriving tokens (incident messages) move to the orbit O (investigated node's local storage of messages) via t_5.

[2] The parameter value 0 refers to *false/off/disabled*, the parameter value 1 refers to *true/on/enabled*.

[3] Type of transition: I: immediate transition ⇒ column *Value* denotes priority (PRIO) or weight (WEIGHT); E: timed transition ⇒ column *Value* denotes firing rate.

[4] Unless otherwise stated, these values are used as a basis for numerical results presented in Sect. 4.

Table 3. Transitions used in Fig. 2.

ID	Type[3]	Description	Value
t_1	E	Request generation	`IF` $(\beta = 0$ `OR` $(S_{id} + S_{bd} < S))$ `AND` $(\gamma = 0$ `OR` $O_u = 1)$: λSo
t_2	I	Incoming request to server	`PRIO` 1
t_3	E	Service	μS_{bu}
t_4	I	Served and no orbital search	`WEIGHT` $1 - p$
t_5	I	Incoming request to orbit	`PRIO` 0
t_6	E	Retrial	`IF` $O_u = 1$: νO
t_7	I	Served and orbital search	`IF` $O_u = 1$: `WEIGHT` p
t_8	E	Busy server breakdown	$\delta_{Sb} S_{bu}$
t_9	E	Busy server repair	$\tau_{Sb} \min(S_{bd}, S_R)$
t_{10}	I	Server hopping on failure	`IF` σ: `PRIO` 1
t_{11}	I	Server flushing on failure	`IF` ϕ: `PRIO` 0
t_{12}	E	Idle server breakdown	$\delta_{Si} S_{iu}$
t_{13}	E	Idle server repair	$\tau_{Si} \min(S_{bi}, \max(0, S_R - S_{bd}))$
t_{14}	E	Orbit breakdown	δ_O
t_{15}	E	Orbit repair	τ_O
t_{16}	I	Orbit flushing on failure	`IF` ω `AND` $O_u = 0$

Table 4. Default parameters based on use case.

Parameter	Symbol	Default[4]
Number of incident types (sources)	K	10
Mean inter-arrival time per unreported incident (1/arrival rate)	λ^{-1}	1 min
Mean retrial time per stored incident message (1/retrial rate)	ν^{-1}	5 ms
Number of potential next hops (servers)	S	5
Mean processing time at next hop (1/service rate)	μ^{-1}	20 ms
Probability that node is aware of next hop's service completion (orbital search)	p	0.1
Ratio of sleep/awake time	α	10
All nodes' mean awake time (1/failure rate)	δ^{-1}	50 ms
All nodes' mean sleeping time (1/repair rate)	τ^{-1}	$\alpha\delta^{-1}$
Message transfer from sleeping to operational next hop (server hopping)	σ	0 (off)
Drop message on falling asleep (server/orbit flushing)	ϕ, ω	0 (off)
Block incoming messages if next hops are down or re-forwarding is disabled (source blocking on server/orbit failure)	β, γ	0 (off)

Idle servers fail (t_{12}), i.e., idle next hops fall asleep, with rate δ_{Si}. Each failed idle server gets repaired (t_{13}), i.e., idle next hop wakes up, with a rate of τ_{Si} (if $S_R \geq S$). Similarly, busy servers fail (t_8) with rate δ_{Sb} and get repaired (t_9) with τ_{Sb} each (if $S_R \geq S$). If $\delta_{Si} = \delta_{Sb} > 0$, we call the breakdowns *independent*, and if $\delta_{Si} = 0 < \delta_{Sb}$, we call them *active* (cf. [17]). In the following, we assume that all breakdowns and repairs are independent with rates $\delta := \delta_{Si} = \delta_{Sb}$ and $\tau := \tau_{Si} = \tau_{Sb}$, respectively. We call δ^{-1} and τ^{-1} the next hops' *mean awake and sleeping time*, respectively.

For keeping the model general and being able to compare the model to related work, we also allow the number of server repairmen S_R to be smaller than the number of servers. If $0 < S_R < S$, failed busy servers are repaired with a higher priority than idle servers. Hence, S_R is the maximum number of repairmen available for failed busy servers and $\max(0, S_R - S_{bu})$ is the remaining maximum number of repairmen available for failed idle servers (see column *Value* in Tab. 3 for t_9 and t_{13}). In the following, we assume that $S_R = S$.

The model allows server *hopping* (t_{10}) and *flushing* (t_{11}) on server failure if the parameters σ and ϕ are set to *true* (i.e., 1), respectively. Server hopping

enables tokens to be directly transferred from a failing server to any operational idle server. If server flushing is active, tokens are moved from a failing server to the orbit. If both options are enabled, server hopping has higher priority. Workload at failing servers is resumed after repair if both options are disabled. In the following, we assume that hopping is always disabled ($\sigma = 0$) and server flushing is referred to as *next-hop dropping* since incident messages are dropped from failing next hops if $\phi = 1$.

Tokens located in the orbit (O) represent incident messages stored in the node under investigation for retransmission. Each token located in the orbit (O) is retrying (t_6) to enter the group of servers after an exponentially distributed retrial time with mean $1/\nu$.

Representing the node's potential to refrain from storing incoming messages and from sending stored messages for power-saving reasons, the orbit is subject to failure (t_{14}) with rate δ_O. A failed orbit gets repaired (t_{15}) with rate τ_O. Depending on parameter ω, the failing orbit may discard all stored tokens (*node dropping*, $\omega = 1$) via t_{16} or keep them for later resumption. In the following, we assume $\delta_O = \delta$ and $\tau_O = \tau$.

Via parameters β and γ, *blocked sources* (cf., [17]) can be modeled (see guard function of t_1). In the blocked case, sources do not generate new calls if all servers ($\beta = 1$) and/or the orbit ($\gamma = 1$) is down. In the unblocked case ($\beta = \gamma = 0$), sources are aware neither of server nor orbit failures. Blocked sources represent the node's ability to decline incoming messages since no next hop is available (*next-hop blocking*, $\beta = 1$) or since the node itself is in power-saving mode (*node blocking*, $\gamma = 1$). Here, power-saving of the investigated node refers to the case when it still might forward messages to next hops but refrains from storing and re-sending new incident messages that cannot be sent immediately.

With probability p, a busy server on service completion (token at place F) informs the (operational) orbit of its upcoming idleness and, if available, directly receives (t_7) the next token from the orbit (*orbital search*). With a probability of $1 - p$, orbital search is not performed (t_4). For $p \approx 1$, the performance of a reliable retrial queue resembles the performance of a classical first-come-first-served queue. In the following, we assume that the investigated node gets aware of a next hop's idleness with a probability of 10%, i.e., $p = 0.1$.

3.2 Underlying Markov Chain

The described GSPN can be mapped to the five-dimensional stochastic process $X(t) = (S_{bu}(t), S_{bd}(t), S_{iu}(t), O(t), O_u(t))$, where $0 \leq S_{bu}(t) \leq \min(S, K)$, $0 \leq S_{bd}(t) \leq \min(S, K)$, $0 \leq S_{iu}(t) \leq S$, $0 \leq O(t) \leq K$, and $O_u(t) \in \{0, 1\}$ are the number of tokens in places $S_{bu}, S_{bd}, S_{iu}, O$, and O_u, respectively, at time $t \geq 0$. Note that $S_{id}(t) = S - (S_{bu}(t) + S_{bd}(t) + S_{iu}(t))$, $So(t) = K - (O(t) + S_{bu}(t) + S_{bd}(t))$, and $O_d(t) = 1 - O_u(t)$. Places R and F do not have to be considered because all states where $R(t) > 0$ or $F(t) > 0$ are vanishing due to enabled immediate transitions.

Since all involved random variables are exponentially distributed, $X(t)$ constitutes a CTMC. We denote the state space of $X(t)$ with $\mathbf{X}$. Since $\mathbf{X}$ is both finite and irreducible, the CTMC is ergodic for all positive values of the arrival

rate λ. From now on, the system is assumed to be in steady state, i.e., $t \to \infty$. Due to space limitations and the high complexity of the underlying CTMC, we refrain from visualizing it here. We also refrain from giving equations for the size of $\mathbf{X}$ based on K and S which is a tedious combinatorial problem. Note that for a less complex model, this was achieved in [19].

3.3 Main Performance Measures

The stationary probabilities of the CTMC discussed in Sect. 3.2 are

$$\begin{aligned} &P(s_{bu}, s_{bd}, s_{iu}, o, o_u) \\ &\quad = \lim_{t\to\infty} P(S_{bu}(t) = s_{bu}, S_{bd}(t) = s_{bd}, S_{iu}(t) = s_{iu}, O(t) = o, O_u(t) = o_u)\,. \end{aligned}$$

The stationary probabilities can conveniently be derived using the software tool MOSEL-2 (see Sect. 4). Knowing the stationary probabilities, the main performance measures of the finite-source retrial queue can be obtained as follows:

- *Mean number of tokens at operational servers* $\overline{S_{bu}}$*:*

$$\overline{S_{bu}} = \sum_{\forall(s_{bu},s_{bd},s_{iu},o,o_u)\in\mathbf{X}} s_{bu} P(s_{bu}, s_{bd}, s_{iu}, o, o_u)\,.$$

- *Mean number of tokens at failed servers* $\overline{S_{bd}}$*:*

$$\overline{S_{bd}} = \sum_{\forall(s_{bu},s_{bd},s_{iu},o,o_u)\in\mathbf{X}} s_{bd} P(s_{bu}, s_{bd}, s_{iu}, o, o_u)\,.$$

- *Mean number of tokens at the orbit* $\overline{O}$*:*

$$\overline{O} = \sum_{\forall(s_{bu},s_{bd},s_{iu},o,o_u)\in\mathbf{X}} o P(s_{bu}, s_{bd}, s_{iu}, o, o_u)\,.$$

- *Probability that all servers are down* P_{S_d}*:*

$$P_{S_d} = \sum_{\forall(0,s_{bd},0,o,o_u)\in\mathbf{X}} P(0, s_{bd}, 0, o, o_u)\,.$$

- *Probability that orbit is down* P_{O_d}*:*

$$P_{O_d} = \sum_{\forall(s_{bu},s_{bd},s_{iu},o,0)\in\mathbf{X}} P(s_{bu}, s_{bd}, s_{iu}, o, 0)\,.$$

- *Mean number of blocked active sources* $\overline{So_b}$*:*

$$\overline{So_b} = \sum_{\substack{\forall(0,s_{bd},0,o,o_u)\in\mathbf{X}\ |\ \beta=\mathbf{1};\\ \forall(s_{bu},s_{bd},s_{iu},o,0)\in\mathbf{X}\ |\ \gamma=\mathbf{1};}} (K - (s_{bu} + s_{bd}))P(s_{bu}, s_{bd}, s_{iu}, o, o_u)\,.$$

- *Utilization of servers:* $\rho = \frac{\overline{S_{bu}}}{S}$.
- *Mean number of busy servers:* $\overline{S_b} = \overline{S_{bu}} + \overline{S_{bd}}$.
- *Mean number of tokens flushed on orbit failure:* $\overline{O_f} = \omega\overline{O}$.
- *Mean number of tokens at service or orbit:* $\overline{M} = \overline{S_b} + \overline{O}$.
- *Mean number of active sources:* $\overline{So} = K - \overline{M}$.
- *Probability that a specific source is blocked:* $P_b = \frac{\overline{So_b}}{\overline{So}}$.
- *Mean number of unblocked active sources:* $\overline{So_a} = \overline{So} - \overline{So_b}$.
- *Overall arrival rate:* $\lambda_{in} = \lambda\overline{So_a}$.
- *Departure rate of served tokens:* $\lambda_s = \mu\overline{S_{bu}}$.
- *Departure rate of unserved tokens:* $\lambda_u = \lambda_{in} - \lambda_s$.
- *Probability of an incoming token getting served:* $P_s = \frac{\lambda_s}{\lambda_{in}}$.
- *Mean waiting time:* $\overline{W} = \frac{\overline{O}}{\lambda_{in}}$.
- *Mean response time:* $\overline{T} = \frac{\overline{M}}{\lambda_{in}}$.

4 Numerical Results

Instead of deriving the underlying CTMC and solving the system of global balance equations manually, we directly formulate the GSPN shown in Fig. 2 using the Modeling, Specification and Evaluation Language (MOSEL-2). MOSEL-2's evaluation environment then derives the performance measures by automatic generation and numerical evaluation of the underlying CTMC. Due to space limitations, we refer the interested reader to [9, 16] for details on MOSEL-2. The corresponding MOSEL-2 model is available on request. The scalability of MOSEL-2 in the context of finite-source retrial queues is discussed in [18].

The validity of the proposed model can be shown by comparing its results in the case of a reliable orbit to numerical results obtained in [3] and [17] (both partly based on the Pascal program provided in [14, p. 272–274]), and [18]. For all comparable parameter settings, our results perfectly match the reference results.

Unless stated otherwise, the model parameters are chosen according to Tab. 4 in the following.

On the x-axis of all result graphs, the parameter α (*sleep/awake ratio*) is given. As already indicated in Tab. 4, $\alpha = \frac{\delta}{\tau} = \frac{\tau^{-1}}{\delta^{-1}}$, where $\tau^{-1} := \tau_O^{-1} = \tau_{Sb}^{-1} = \tau_{Si}^{-1}$ is the mean time the nodes are in power-saving mode, and $\delta^{-1} := \delta_O^{-1} = \delta_{Sb}^{-1} = \delta_{Si}^{-1}$ is the mean time the nodes (investigated node and its idle or busy next hops) are awake. Hence, α is the proportion of time the investigated nodes are in power-saving mode. The higher α, the longer nodes sleep in comparison to the time they are awake and the less energy they use in the course of time.[5]

On the y-axis of the presented result graphs, we focus on the mean response time $\overline{T}$, source blocking probability P_b, and probability of getting served P_s.

[5] Note that in this paper we do not yet consider energy used by the process of switching between awake and sleep states.

4.1 Influence of Sleep/Awake Ratio and Mean Awake Time

Fig. 3 shows the effect of modifying the sleep/awake ratio α (x-axis) and the mean awake time δ^{-1} (curves) on the mean response time $\overline{T}$ (y-axis).

Increasing the energy efficiency (sleep/awake ratio) clearly results in a higher mean response time. Increasing the mean awake time while keeping the mean sleep/awake ratio constant also increases the mean response time, because the sleeping time is a multiple of the awake time. Unfortunately, decreasing the mean awake time is not arbitrarily possible, since switching state often comes with additional overhead not (yet) handled by the model.

An important result is that accepting higher response times, i.e., increased delay tolerance, considerably eases the design of energy-efficient WSNs.

For sleep/awake ratio $\alpha \approx 0$, the probability that a node sleeps is very small. For all values of the mean awake time δ^{-1}, the mean response time $\overline{T}$ is then close to the processing time $\mu^{-1} \approx 20ms$ of the next hop. Since the inter-arrival time $\lambda^{-1} = 1min$ is very large in comparison to the processing time, the probability that at arrival of a new incident all next hops are busy is close to zero. Consequently, the probability for having to wait (and to retry) is also close to zero which results in a negligible waiting time.

In the following, we set the mean awake time to $\delta^{-1} = 50ms$ and discuss whether the mean response time can be decreased by dropping messaged from sleeping nodes.

4.2 Influence of Sleep/Awake Ratio and Message Dropping

Fig. 4 shows of the effect of the sleep/awake ratio α (x-axis) and incident message dropping by sleeping nodes $\phi = \omega$ (curves) on the mean response time $\overline{T}$ (y-axis, top) and on the serving probability P_s (y-axis, bottom).

It can be seen at the top of Fig. 4 that dropping stored incident messages from sleeping nodes significantly decreases the mean response time, even below the reliable case. This is because messages that are dropped may spend less time in the system than messages that are successfully served.

Therefore, we also need to have a look at the probability P_s that an accepted incident message can be successfully processed. This is shown at the bottom of Fig. 4. Dropping stored messages clearly implies smaller serving probabilities. Therefore, dropping should be used for outdated messages only. However, the age of tokens cannot be considered by the presented model.

Note that even for $\alpha \approx 0$, i.e., the probability of the nodes being awake is close to 1, the serving probability is not necessarily close to 1. The serving probability also highly depends on the absolute value of the awake time since after each awake time, stored incident messages are dropped (regardless how short the sleep time is afterwards).

In Sect. 4.3, we aim at increasing the serving probability by blocking incoming messages when nodes are sleeping.

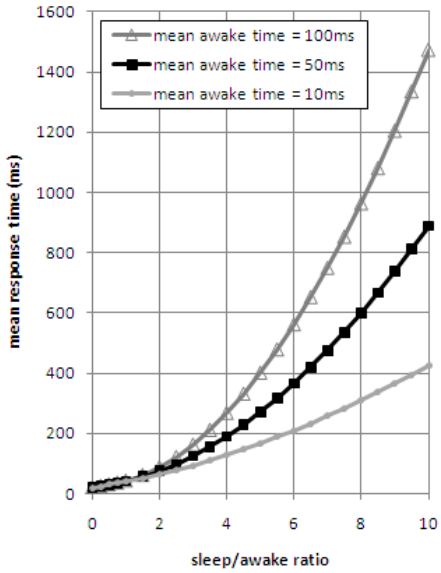

Fig. 3. Effect of sleep/awake ratio α (x-axis) and mean awake time δ^{-1} (curves) on mean response time $\overline{T}$ (y-axis).

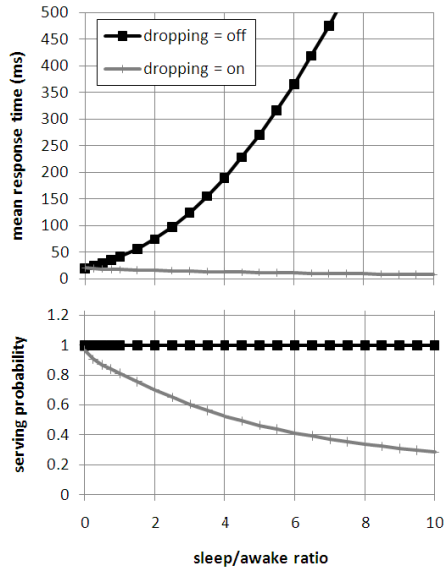

Fig. 4. Effect of sleep/awake ratio α (x-axis) and message dropping $\phi = \omega$ (curves) on mean response time $\overline{T}$ (y-axis, top) and on serving probability P_s (y-axis, bottom).

4.3 Influence of Sleep/Awake Ratio and Message Blocking

Fig. 5 shows the effects of the sleep/awake ratio α (x-axis) and of the message blocking $\beta = \gamma$ (curves) on the mean response time $\overline{T}$ (y-axis, top), the serving probability P_s (y-axis, middle), and the blocking probability P_b (y-axis, bottom).

First, we discuss the serving probability shown in the middle of Fig. 5. Blocking incoming messages has a positive effect on the serving probability. For sleep/awake ratio $\alpha > 3$, the effect of next-hop blocking (—+—) has significantly more effect than node blocking (—◆—).

In comparison to the non-blocking case (—+—), the following improvements can be observed:

- Some improvement (—◆—): Node blocking prevents arriving messages from being dropped immediately when the node is in power-saving mode and immediate forwarding is not possible due to busy or sleeping next hops at the same time.
- Good improvement (—+—): If next-hop blocking is enabled, messages do not join the node when no next hop is accepting messages. This reduction of the number of messages stored at the node increases the serving probability, since all stored messages are dropped later with a high probability.
- Best improvement (—◆—): The combination of both blocking mechanism results in the best improvement of the serving probability.

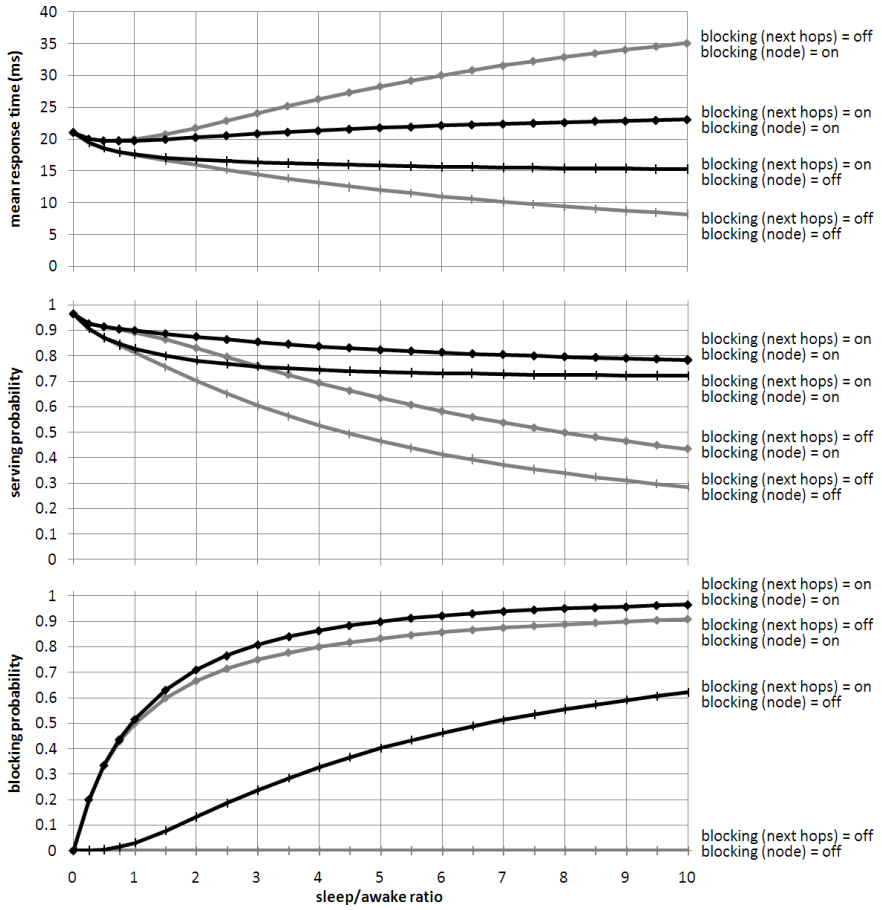

Fig. 5. Effect of sleep/awake ratio α (x-axis) and message blocking β & γ (curves) on mean response time $\overline{T}$ (y-axis, top), serving probability P_s (y-axis, middle), and blocking probability P_b (y-axis, bottom).

Focusing on the two best cases with respect to the serving probability (next-hop blocking is enabled, curves ◆ and ┼), we now discuss which is the better alternative regarding the mean response time given at the top of Fig. 5.

Curve ┼ refers to the case when node blocking is disabled. As seen before, the serving probability drops significantly when the sleep/awake ratio is increased, because it is more likely that a message needs to wait in the node and hence may be dropped if storage gets disabled before an operational and free next hop is found. Since dropped messages tend to stay less time in the system than served messages, the mean response time decreases.

In the scenario of curve ┼, the probability that an incoming message gets immediately dropped is higher than in the scenario of curve ◆, where this situation is not possible due to node blocking.

Curve ◆ follows the same behavior as + for small sleep/awake ratios ($\alpha < 0.5$) but is then dominated by another effect that can be explained as follows. Messages are only accepted when they can at least be stored locally. Hence, if they cannot be forwarded immediately on arrival, they will (in contrast to scenario +) always experience waiting time. Additionally, since the next-hop nodes follow the same sleep/awake ratio as the investigated node, stored messages have to wait longer before being successfully transferred to an available next hop with increasing sleep/awake ratio.

Hence, while ◆ is more attractive than + with respect to the serving probability, it is less attractive regarding the mean response time. Moreover, it can be seen at the bottom of Fig. 5 that + has a significantly lower probability of blocking incoming incident messages than ◆.

We can therefore conclude that next-hop blocking is more attractive than node blocking. It should be noted, however, that next-hop blocking is relying on local knowledge about the next hops' state. How this knowledge can be obtained in a self-organized manner and which overhead might come with this is not yet covered by this paper.

5 Conclusions

A generalization of finite-source retrial queueing systems is studied. In addition to unreliable servers, the model can be used to evaluate the effects of an unreliable orbit. The model is used to discuss the trade-off between energy efficiency and performance in a wireless sensor network use case. The model evaluation is carried out using the MOSEL-2 tool. The numerical results are discussed in detail and show the positive and negative effects of message dropping and message blocking.

Our future work aims at finding self-organized mechanisms to provide the sensor nodes with the necessary local knowledge. We also need to discuss in more detail how single-node results can be aggregated suitably to evaluate multiple-hop scenarios and larger network topologies. Validation of model results by comparison with test-bed results is considered. Finally, the presented model should assist in developing energy-efficient and self-organizing lightweight communication protocols for wireless sensor networks.

References

1. I. F. Akyildiz and M. C. Vuran. *Wireless Sensor Networks*. John Wiley & Sons, July 2010.
2. B. Almasi, G. Bolch, and J. Sztrik. Heterogeneous finite-source retrial queues. *Journal of Mathematical Sciences*, 121(5):2590–2596, June 2004.
3. B. Almasi, J. Roszik, and J. Sztrik. Homogeneous finite-source retrial queues with server subject to breakdowns and repairs. *Mathematical and Computer Modelling*, 42:673–682, 2005.

4. J. Amador. On the distribution of the successful and blocked events in retrial queues with finite number of sources. In *Proc. of the 5th Int'l Conf. on Queueing Theory and Network Applications*, pages 15–22, 2010.
5. J. R. Artalejo. New results in retrial queueing systems with breakdown of the servers. *Statistica Neerlandica*, 48:23–36, 1994.
6. J. R. Artalejo. Retrial queues with a finite number of sources. *J. Korean Math. Soc.*, 35:503–525, 1998.
7. J. R. Artalejo and A. Gómez-Corral. *Retrial Queueing Systems: A Computational Approach*. Springer Verlag, 2008.
8. J. R. Artalejo, V. C. Joshua, and A. Krishnamoorthy. An M/G/1 retrial queue with orbital search by the server. In J. R. Artalejo and A. Krishnamoorthy, editors, *Advances in Stochastic Modelling*, pages 41–54. Notable Publications Inc., NJ, 2002.
9. K. Begain, G. Bolch, and H. Herold. *Practical Performance Modeling – Application of the MOSEL Language*. Kluwer Academic Publishers, 2001.
10. G. Bolch, S. Greiner, H. de Meer, and K. Trivedi. *Queueing Networks and Markov Chains*. John Wiley & Sons, New York, 2nd edition, 2006.
11. S. R. Chakravarthy, A. Krishnamoorthy, and V. Joshua. Analysis of a multi-server retrial queue with search of customers from the orbit. *Performance Evaluation*, 63(8):776–798, 2006.
12. F. Dressler. *Self-Organization in Sensor and Actor Networks*. John Wiley & Sons, 2007.
13. A. N. Dudin, A. Krishnamoorthy, V. Joshua, and G. V. Tsarenkov. Analysis of the BMAP/G/1 retrial system with search of customers from the orbit. *Eur. J. Operational Research*, 157(1):169–179, 2004.
14. G. Falin and J. Templeton. *Retrial Queues*. Chapman & Hall, 1997.
15. J. Wang, J. Cao, and Q. Li. Reliability analysis of the retrial queue with server breakdowns and repairs. *Queueing Systems*, 38:363–380, 2001.
16. P. Wüchner, H. de Meer, J. Barner, and G. Bolch. A brief introduction to MOSEL-2. In R. German and A. Heindl, editors, *Proc. of 13th GI/ITG Conference on Measurement, Modelling and Evaluation of Computer and Communication Systems (MMB 2006)*. GI/ITG/MMB, University of Erlangen, VDE Verlag, 2006.
17. P. Wüchner, H. de Meer, G. Bolch, J. Roszik, and J. Sztrik. Modeling finite-source retrial queueing systems with unreliable heterogeneous servers and different service policies using MOSEL. In K. Al-Begain, A. Heindl, and M. Telek, editors, *ASMTA 2007 Conference*, pages 75–80, Prague, Czech Republic, June 2007.
18. P. Wüchner, J. Sztrik, and H. de Meer. Homogeneous finite-source retrial queues with search of customers from the orbit. In *Proc. of 14th GI/ITG Conference on Measurement, Modelling and Evaluation of Computer and Communication Systems (MMB 2008)*, Dortmund, Germany, March 2008.
19. P. Wüchner, J. Sztrik, and H. de Meer. Finite-source M/M/S retrial queue with search for balking and impatient customers from the orbit. *Computer Networks*, 53(8):1264–1273, June 2009.