# TAWNS – A Terrestrial Acoustic and Wireless Network Simulation Framework

Leonhard Brüggemann[1(✉)], Bertram Schütz[2], and Nils Aschenbruck[1]

[1] Institute of Computer Science, Osnabrück University, 49074 Osnabrück, Germany
{brueggemann,aschenbruck}@uni-osnabrueck.de
[2] Fraunhofer Institute for Communication, Information Processing and Ergonomics, 53177 Bonn, Germany
bertram.schuetz@fkie.fraunhofer.de

**Abstract.** Wireless Sensor Networks that address audio-related applications have been researched for over a decade. But reproducibility and customizability are a challenge when focusing on research questions in outdoor environments (e.g., localization of wildlife species). It usually involved the design of customized prototypes and time-consuming and often error-prone deployments to generate a ground truth. In this paper, we propose TAWNS, an Open Source model framework for Omnet++/INET which supports researchers conducting simulations on wireless *acoustic* sensor networks - either as the main focus of study or as a component. It inherits all the networking and wireless communication elements supported by the library INET and provides a novel, easy-to-use spatial audio simulator for 3-D sound environments. For the latter, we adapted and extended the library Scaper by adding a customized model for attenuating and delaying the acoustic signals. Our model parameters are derived from multiple measurements with different hardware. We share them in this paper, serving as a basis for other researchers. To give an impression of the hardware requirements, we also evaluate the system resources of our simulator.

**Keywords:** TAWNS · WASN · simulator · Omnet++ · INET · wireless acoustic sensor networks · spatial sound generation · bioacoustic research

## 1 Introduction

Wireless Acoustic Sensor Networks (WASN) in terrestrial environments are, among others, applied in the military context, surveillance, or bioacoustic applications (e.g., [15]). Such networks consist of multiple wireless communicating devices, each equipped with at least one microphone, a processing unit, and at least one communication interface. Such networks must meet various requirements due to their constrained resources and the vast amount of recorded data that must be processed and transferred, especially in outdoor environments when exposed to weather and various structures.


E. Hyytiä and V. Kavitha (Eds.): VALUETOOLS 2022, LNICST 482, pp. 292–306, 2023.
https://doi.org/10.1007/978-3-031-31234-2_18

We designed and deployed our low-cost, customized, wireless acoustic sensor network to acquire a ground truth for our localization idea. Although the localization of sound sources is a well-researched problem in the military context (e.g., [18]), it is far from being solved when it comes to locating wildlife, especially birds and similar. That is mainly the case because acoustic signals of, e.g., bird species are much more sophisticated and, thus, harder to distinguish, detect, or identify, aggravating the localization of wildlife significantly. When we wanted to compare our results to other datasets, it was quite difficult because no public available ground truth data exists. Instead, it is the current state of the art that research in WASN requires building custom-designed sensor nodes and performing some deployments to acquire a dataset that is used for evaluation (e.g., [16]). We see three major challenges with this current state of the art:

1. Comparability: The custom-designed prototype and the deployment site affect the performance of the system and the acquired data, so it becomes difficult to compare the results.

2. Reproducibility: The measured data is seldom publicly available and, thus, cannot be used by other researchers. Furthermore, it is impossible to reproduce other measurements because the deployments of different research groups extend throughout the world with different species and conditions at the deployment sites.

3. Costs and Time: The hardware used in the deployments might cost some hundred euros per device. That makes it pretty expensive if large deployments should be conducted.

Furthermore, conducting field measurements is quite time-consuming and error-prone due to their vulnerability to weather and high power consumption, which requires a periodic change of batteries. Of course, these challenges are already known and have been stated by [11,21]. They propose developing a framework for providing ground-truth datasets and various other features in demand for bioacoustic recognition and localization systems. In this paper, we do not aim to provide a whole bioacoustic framework but focus on the lack of ground-truth data. Instead of simply providing a platform that can be used to share measured datasets, we provide a simulator extension for Omnet++, which everyone can use to generate ground-truth data themselves. The contributions of this paper are the following:

- We will describe our simulator's design and system architecture as well as the requirements that guided its design.
- We share multiple sets of parameters based on our measurements for the sound simulation in outdoor environments that everyone can use.
- We share our results for two scenarios to give an impression on the simulator's hardware requirements when the simulation time and the number of entities increase.
- We publish our simulator as Open Source under this link[1].

The paper is structured as follows: Sect. 2 summarizes the related work on WASN simulation. Section 3 states a short requirements analysis that guided our design decisions. Then, Sect. 4 describes the system architecture and our

[1] https://sys.cs.uos.de/tawns/index.shtml.

**Table 1.** Survey of frameworks that support sound generation and fit to 3-D outdoor environments

| Framework | Mixing Audios | Sound Attenuation | Sound Delay | 3-D Environment | Application Context |
|---|---|---|---|---|---|
| Blender | ✓ | ✓ | – | ✓ | gaming |
| Panda 3D | ✓ | ✓ | – | ✓ | gaming |
| Audaspace | ✓ | ✓ | – | ✓ | audio engine gaming |
| OpenAl-Soft | ✓ | ✓ | – | ✓ | audio API gaming |
| FMOD | ✓ | ✓ | – | ✓ | audio API gaming |
| Pyroomacoustics | ✓ | ✓ | ✓ | ✓ | room acoustic |
| SoundScape Renderer | ✓ | ✓ | ✓ | – | real-time spatial audio reproduction |
| Scaper | ✓ | ✓ | ✓ | ✓ | sound synthesis and augmentatiosn |

modifications and extension for WASN simulations. After that, we share our evaluation of the system requirements in Sect. 5. In Sect. 6, we summarize our contributions and identify the immediate steps in the ongoing development of our simulator.

## 2 Related Work

While simulators for wireless networks have been part of research for decades, there are only a few simulators for generating sounds. Both kinds of simulators abstract reality, make various assumptions, and have helped answer various research questions. However, a single simulation for one domain lacks features for the second domain. In the past, this had already been the case when, e.g., running traffic or vehicular network simulations, which resulted in the Omnet++ frameworks VEINS [25] and SUMO [20]. Now, it is a similar case for research on wireless acoustic sensor networks. To the best of our knowledge, no simulator similar to the here proposed framework has been implemented yet. We will survey related work of wireless network simulators and frameworks for spatial sound simulation in the following subsections.

### 2.1 Wireless Network Simulators

Various wireless network simulators have been part of the research community for years and are still constantly extended. For this paper, we focused on two state-of-the-art simulators supporting many networking and wireless communication elements. NS-3 [22] is a well-known Open Source framework and the successor of ns-2 [4], which was quite a popular simulator during the early 2000s. NS-3 targets primarily research and educational use, evolves around building a testbed environment, and is still used in research (e.g., [9]). Like NS-3, the popular Open Source simulator Omnet++ [5] also aims at a testbed environment. It does so by developing modules that interact together to build complex systems. The INET framework [3] for Omnet++ provides many elements of networking and wireless communication that are required for simulating network devices and various protocols.

Although both simulators provide many general features required when simulating wireless sensor networks, they do not include any audio-related entities or applications. For example, it is impossible to simulate an acoustic environment whose sounds are recorded by simulated network devices and then transmitted.

### 2.2 Generating Spatial Audio Mixtures

While there are many audio-related frameworks, APIs, or libraries, only a few target the generation of spatial audio mixtures with various sources, so-called soundscapes. Of those few, we are unaware of any framework that already supports the generation of spatial sound mixtures for 3-D *outdoor* environments. We looked closely at multiple frameworks and checked whether they might be adapted or extended to use cases concerning species identification or individual census or localization. For that, we set four mandatory criteria such as (1) mixing multiple sounds into new audio files, (2) applying a sound attenuation to the acoustic signals, (3) delaying the audio signal according to the distance between the receiver and sender, and (4) adapting to a 3-D environment. The latter implies that the frameworks should not require indoor obstacles such as walls or ceilings to calculate the propagation of acoustic signals. If those four features are supported, it is possible to extend such a library without fundamental modifications to its frameworks. We summarized our analysis in Table 1.

Computer games are prominent applications in which the replayed sounds generate an environmental feeling. Well known Open Source game engines are Blender [17], or Panda 3D [7] that use audio frameworks such as Audaspace [1], FMOD [2] or OpenAl-Soft [6]. Although they support the mixing and attenuating of audios, the audio signals are not delayed, probably because it is insignificant in computer games. However, it is a crucial feature for time-related evaluations in WASNs, e.g., for the localization of acoustic sound sources.

The Python library Pyroomacoustic [24] assists acoustic research in indoor environments. They provide many utilities that assist in many acoustic use cases. Sadly, the propagation of the acoustic signal requires an indoor environment consisting of walls and similar because the propagation of acoustic signals is based on ray-tracing. This library needs to be extended for outdoor scenarios.

The SoundScape Renderer [10] is a framework for real-time spatial audio rendering. It supports three sound rendering modules, but only one of its renderers supports attenuating and delaying acoustic signals according to the distance between the source and listener. That renderer does not only attenuate a signal according to the source-listener distance but also to their angle. In addition, SoundScape Renderer always requires 2-D environments and must be fundamentally extended to adapt to 3-D outdoor environments.

Scaper [23] is a library focusing on sound synthesis and augmentation. This framework always requires an audio library that has to be provided by the user. It distinguishes between background and foreground audios mixed according to customized parameters. The foreground audios are added according to a set distribution or might be added manually. It supports the configuration of various other parameters, including an attenuation parameter and a delay which can be set for each foreground audio.

In our opinion, most existing frameworks do not fit our stated criteria and are probably relatively complex to extend for simulating soundscapes in 3D outdoor environments. The Python library Scaper appears to be the best candidate for extending its functionality to generate spatial soundscapes.

## 3 Requirement Analysis

There are some specific challenges in designing a simulator for terrestrial wireless acoustic sensor networks, so an initial requirement analysis is conducted. Due to page limitations, we do not provide a complete, formal analysis here but state the key functionalities that guide the simulator's design.

**Support of various Network Technologies:** The simulator should support various network types and communication technologies that are relevant when deploying a WASN in reality.

**Generation of spatial Soundscapes:** Each sound source emits acoustic signals that propagate through the air and, thus, are delayed and attenuated by its environment. The mixture of multiple sounds is called soundscapes.

**Simulation of Movement:** The wireless nodes, as well as the sound sources, should be able to stay in one place or move in all directions.

**Setup in 3-D Environment:** As most research in terrestrial environments concerns 3-D environments, the simulator should support it as well.

**Extendability and Customization:** The simulator's entities should be extendable and customizable, e.g., it should be possible to customize the network and its devices or apply various kinds of sound sources. Thus, Open Source software is preferred.

## 4 System Architecture

The TAWNS works as a stand-alone framework and consists of three parts: (1) the wireless acoustic sensor network consisting of other network devices and the sound sources, (2) an audio library for acoustic data of the sound sources, and (3) our extension for the Python library Scaper to generate the spatial soundscapes. Although those three components exist separately, they are glued together in TAWNS by adding modules with new functionalities or developing modules that wrap up specific information. Figure 1 illustrates the three components.

(1) To support various elements of networking and wireless communication, the already stated simulators, NS-3 and Omnet++/INET, are appropriate candidates. Due to its modular design and excellent documentation, we finally decided to use Omnet++ combined with the framework INET. Furthermore, we think its design with the configuration files is more user-friendly, especially for non-computer scientists. Our network components of TAWNS are based on the INET library. Thus, they inherit its network stack and all its corresponding functionality. We extended some of INET's modules by audio-related functions, e.g., the

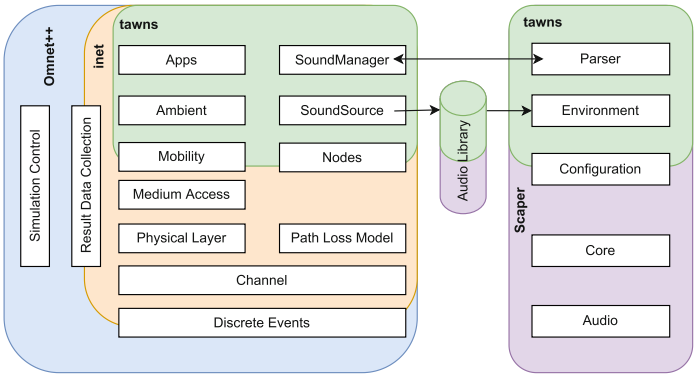



**Fig. 1.** Illustration of TAWNS' system architecture

network nodes can now be configured to have a customized number of microphones or transmit actual audio data instead of dummy traffic. We added new modules that hold the information required for generating the soundscapes, such as the module SoundSource, which handles all movement and acoustic emissions of a sound entity during a simulation or the module Ambient to set background acoustic. Both modules refer to the audio library that provides the acoustic files. Of special importance is the module SoundManager. It collects all information from the modules that are necessary for the generation of spatial soundscapes and initiates the audio generation for each sensor node. Our implementation is written mainly in C++, Omnet++'s configuration files, and Python. We deliberately decided not to use any libraries that Omnet++/INET and Scaper do not already use to keep the number of dependencies at a minimum.

(2) The audio library contains all audios used in generating soundscapes and is already required by the library Scaper. It contains the back- and foreground audios. Preparing such a library allows extended customization of the sound environment in the simulation. Furthermore, there are various databases for sounds. For wildlife in general, there is the Tierstimmenarchiv Berlin[2], or if focusing on birds, there is the database at xeno-canto[3].

(3) While the wireless sensor network's core functionality could be inherited from existing work, generating spatial soundscapes for a simulated 3-D outdoor environment is more complicated. Scaper distinguishes between fore- and background audios. Many parameters can be set during its configuration, including the parameters delay and signal-to-noise. Our new spatial soundscape extensions evolve around those two parameters by adapting them to the 3-D environment used in Omnet++. By calculating the distance between a sound source and the sensor node and assuming a particular speed of sound, it is straightforward to calculate the delay by dividing the distance by the speed of sound. However, fitting Scaper's signal-to-noise parameter (snr) to an existing sound model is tricky

[2] www.tierstimmenarchiv.de (accessed: 16.06.2022).
[3] www.xeno-canto.org (accessed: 16.06.2022).

because common sound attenuation models do not distinguish between back- and foreground audios. We discuss an appropriate sound attenuation model for Scaper in the following section.

### 4.1 Sound Attenuation Model of Scaper-Extension

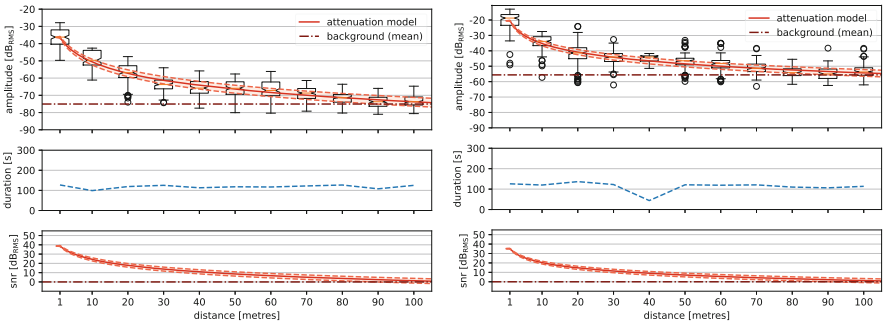
A sound attenuation model predicts the sound at a specific distance. The physical background is quite complex and is the content of many books (e.g., [1]). Generally, a more realistic attenuation model requires knowledge about the sound source (such as the behavior, height, or emission angle) and its sound characteristics (such as the amplitude, frequencies, and direction), and its surroundings (consisting of obstacles, vegetation, trees, wind, temperature, etc.). Especially the sound characteristics become complicated for some kinds of wildlife such as birds, cicadas, or bats. For example, bird sounds can cover a wide range of frequencies that propagate differently, or birds might constantly change their location or direction, affecting, among others, the amplitude and propagation of their sound, resulting eventually in complex sound characteristics. Besides, there is the 3-D outdoor environment, where various sounds combine and interfere. Thus, applying more realistic sound models will not only result in much more detailed modeling, but will also not fit Scaper's approach of mixing back- and foreground audios.

We think the following simple model based on the inverse square law provides a sufficient abstraction for our use case:

$$L_{p_2} = L_{p_1} - 20 \cdot \log_{10}(\frac{R_2}{R_1}) \tag{1}$$

where $L_{p_x}$ is the sound pressure level measured in decibel ($dB_{SPL}$) at location x and $R_x$ is the distance to location x. Note that the model refers to the sound pressure level measured in dB and requires some information about the distances. Sadly, such a model will also not fit with Scaper's back- and foreground audio concept. Neither the $dB_{SPL}$ values nor the distance information are (usually) available for wildlife recordings. The samples within the recorded audios are typically measured in dB relative to their bit-depth, which is referred to as $dB_{FS}$ (dB relative to full-scale). Without detailed specifications on the recording system (e.g., microphone's sensitivity, applied gain, or digitizer's input clip voltage), it is not possible to convert the values from $dB_{FS}$ to $dB_{SPL}$. But even then, it is seldom the case that the distance information on acoustic recordings is known.

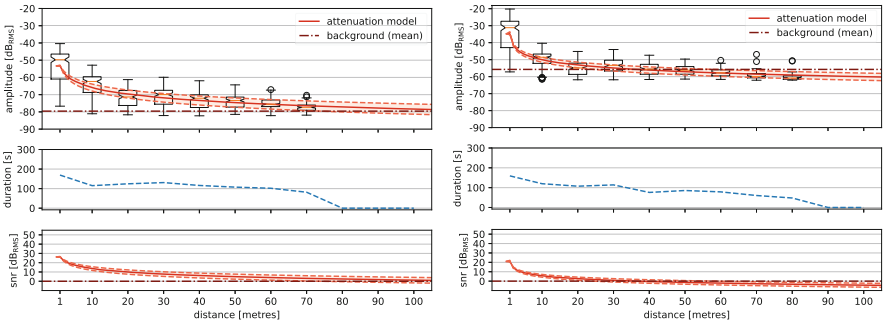
We finally decided to design a model based on measurements we conducted in the past. Our model's idea is as follows: When we record audio, we basically convert the sound pressure and its changes at the microphone to an electronic signal that might be amplified and eventually digitized and saved on the device. So, we have an indirect relation to the sound pressure that, by theory, propagates according to the inverse square law. Therefore, we can assume a logarithmic model but due to many technical, environmental and biologic factors (e.g., birds




(a) SOLO-based sensor node (Fringilla coelebs)




(b) Recorded by AudioMoth (Fringilla coelebs)




(c) SOLO-based sensor node (Luscinia megarhynchos)




(d) Recorded by AudioMoth (Luscinia megarhynchos)

**Fig. 2.** Exemplary illustration of the sound attenuation for two species on two kinds of sensor nodes

are no acoustic point sources or the environment affects the acoustic propagation and recording systems), we do not know the parameters.

In the following, we assume a logarithmic attenuation of sounds, such as

$$A(x) = a \cdot \log_{10}(\max\{1, x + b\}) + c, \quad \text{where } a, b, c \in \mathbb{R} \tag{2}$$

with A(x) referring to the attenuation relative to the background noise (= environment sounds), x referring to the distance between source and recorder, and a,b,c being constant parameters we will derive next. We will set an upper bound due to the high values that are returned by a logarithmic function for values below 1. Such a model is close to being adequate for extending Scaper. According to the specification of the signal-to-noise parameter, it is measured in $dB_{LUFS}$, which involves the perceived loudness of audio for the human ear [8]. In our understanding, using a human-perceived unit when modeling physical phenomena is not reasonable, even in a simple model like the one proposed here. Hence,

we modified Scaper to use $dB_{RMS}$ instead, which calculates to root mean square of an audio file.

In order to estimate the parameters in Eq. 2, experiments are necessary. We deliberately make use from measurements taken in winter (december 2019) in North Rhine-Westphalia in Germany because we want to reduce the acoustic interference caused by other birds and similar. During the measurements, we replayed audios of five bird species and recorded the acoustic signal at the distance of 1 m, 10 m, and every 10 m up to 100 m. Note that we did not record all distances simultaneously but one after the other. We recorded the bird voices by a Raspberry Pi Zero W-based recording device similar to [26], which is used in bioacoustic research. As some hardware was not available anymore, we decided on Rohde VideoMic[4], Sabrent USB soundcard[5], and set a fixed gain of 12dB for the microphone. We also recorded the data with a popular low-budget recording device called AudioMoth v1 [19] that is also applied in bioacoustic research (Audiomoth' gain level 4). We replayed four species in their more or less natural amplitudes according to [13,14]. The measurements were made on 16th, 17th, 18th and 19th December 2019. In order to reduce the interference within the acquired data, e.g., caused by wind, cars, or the microphone's self-noise, we applied a bandpass filter (lower bound: 2000Hz, upper bound: 8000Hz) on the frequencies outside the power spectrum of our replayed acoustic signal. After that, we extracted the bird voices and calculated the amplitude (in $dB_{RMS}$). For bird voices with more than one-second duration, we split the audio into one-second intervals with an overlap of 500 milliseconds. The main reason for this is that our replayed bird voice does not hold one amplitude constantly but might change rapidly. We visualized our results for two species in Fig. 2. Although the confidence intervals are relatively small, the boxplots do not constantly decline over the distance and show some deviation and sometimes a high number of outliers. We identify three primary reasons for that:

1. The bird songs on the audio are not always performed in the same amplitude and have different power spectrums, resulting in different attenuations.

2. The audio recordings lasted at least six minutes in which some acoustic interference was measured, e.g., a loud car on a distant street or similar. We excluded such intervals when the interference was dominant.

3. The used microphone has a relatively high sensitivity (e.g., for our custom-based device, it is -33.0dB re 1 V/Pascal) which means that the same sound pressure at the microphone can cause different audio samples.

It is important to note that the visualized sample sizes and audio durations of the boxplots vary over the distances because some features became less and less clear to identify during the manual extraction of signals. That was especially the case for the species Regulus regulus or Luscinia megarhynchos, in which no songs could be identified at a 70m distance or more.

We used the extracted data to fit 10000 attenuation models with different initial guesses of the start parameters and selected the model with the lowest

[4] www.rode.com/de/microphones/on-camera/videomicro (accessed: 18.06.2022).
[5] www.sabrent.com/products/au-mmsa (accessed: 18.06.2022).

**Table 2.** Derived parameters for Eq. 2 including standard deviation and rmse

| | | SOLO-based recording device | | | | AudioMoth | | | |
|---|---|---|---|---|---|---|---|---|---|
| Bird Species | $db_A$ | a | b | c | rmse | a | b | c | rmse |
| Turdus philomelos | 100 | $-9.2 \pm 0.3$ | $1.4 \pm 0.3$ | $-23.5 \pm 1.1$ | 6.1 | $-7.1 \pm 0.2$ | $-0.2 \pm 0.1$ | $-14.9 \pm 0.7$ | 4.7 |
| Fringilla coelebs | 92 | $-10.8 \pm 0.3$ | $2.3 \pm 0.3$ | $-23.5 \pm 1.2$ | 4.7 | $-8.6 \pm 0.3$ | $0.9 \pm 0.2$ | $-14.8 \pm 1.0$ | 4.9 |
| Luscinia megarhynchos | 86 | $-7.3 \pm 0.4$ | $1.8 \pm 0.6$ | $-40.3 \pm 1.8$ | 8.4 | $-5.7 \pm 0.3$ | $-0.5 \pm 0.1$ | $-29.3 \pm 1.1$ | 7.2 |
| Regulus regulus | 75 | $-5.6 \pm 0.2$ | $-0.7 \pm 0.0$ | $-55.2 \pm 0.7$ | 4.8 | $-3.5 \pm 0.2$ | $-0.9 \pm 0.0$ | $-46.4 \pm 0.7$ | 4.2 |
| Luscinia megarhynchos | 74 | $-5.4 \pm 0.4$ | $-0.0 \pm 0.3$ | $-53.5 \pm 1.3$ | 6 | $-4.2 \pm 0.3$ | $-0.8 \pm 0.1$ | $-40.7 \pm 0.9$ | 5.4 |

rmse. The red lines in Fig. 2 refer to that best-fit model. The dashed line refers to the model's lower and upper bound when considering the estimated parameters' standard deviation. By computing the difference between the fitted model and the loudness of background (in Scaper referred to as the parameter $ref\_db$), we can eventually calculate the attenuation model for each species and each recording device. We share our acoustic sensors' model parameters in Table 2 including their standard deviation and rmse. Note that the birds' amplitudes cover their typical ranges and might be used for other bird species. To derive a more precise model for other custom-designed acoustic sensor, we recommend to repeat above's procedure. With these parameters TAWNS can be configured now allowing for realistic performance calculations.

## 5 Simulation Performance

Running simulations can quickly become time-consuming, also requiring a lot of memory and disk space. In order to get an idea of the simulation time and system resources, we provide a performance evaluation of our simulator, address critical parameters, and identify bottlenecks that will be dealt with in the future.

We focus on two research scenarios: (1) In one scenario, we assume wireless acoustic sensor nodes and sound sources that do not move, e.g. to check primarily on localization methods. The sensor nodes transmit their acoustic data to a sink. In other words, during the simulation, the wireless sensor nodes and the sound sources do not move but stay in one place. (2) In the second scenario, both, sensor nodes and sound sources, will move according to traces previously generated by the random walk model of BonnMotion [12].

To compare the measurements, we have to fix some configuration parameters in order to assure comparability between the runs:

**Constrained Area:** We will assume a constrained area of a width and length of 1000 m, and a height of 100 m. No entity can leave the simulation.

**Adhoc Network:** We will assume an ad-hoc network with distance-sequenced vector routing.

**Sensors:** Each sensor is equipped with one microphone (sampling frequency: 44.1kHZ, bit-depth: 16 bit). They will record acoustic data continuously and, if

possible, transfer it to the nearby base station. They are configured identically, except for their initial location.

**Basestation:** We assume a base station in the center of the constrained area that receives the acoustic data of all sensors.

**Ambient:** We provided an ambient sound with some wind rustling and little other interference.

**Soundsource:** During the simulation, we allow only one kind of sound source, a bird that emits the acoustic signal every time. We assume that each sound source will emit at most 300 times their sound during the simulation. All sound sources are configured identically, except for their starting location and mobility model.

**Audio Data:** The audio files are generated in WAVE format by Scaper and will be transferred as in the lossless audio codec FLAC.

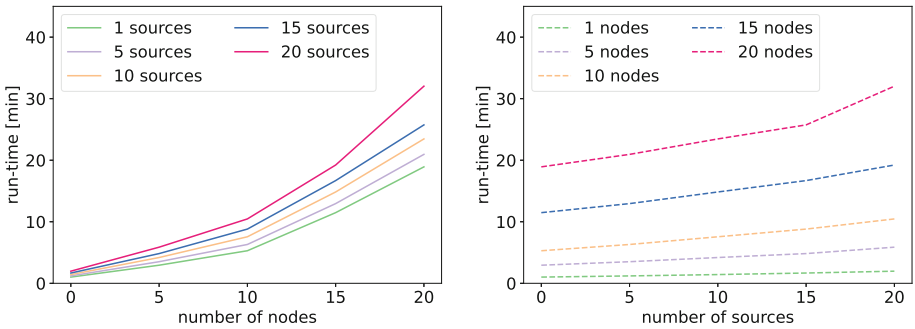
**Mobility:** All generated mobility traces have the same fixed length during all runs.

**Power Consumption:** We neglect any power consumption constraints during the simulation.

**Seed-based Evaluation:** To make the simulation runs comparable with each other, a set of static seeds is used for the random number generator, thus, assuring identical placements of the sensors and sound sources.

All simulations run on an x86-64bit computer running the operating system Ubuntu 5.15 and being equipped with 64 CPUs (2xAMD EPYC 7452 32-Core Processor, 2,35 GHz), 251 Gbyts DIMM DDR4 (3200 MHz), and 500 GByte disk capacity. For each scenario, we will increase the number of wireless sensor nodes (1, 5, 10, 15, 20), the number of sound sources (1, 5, 10, 15, 20), and run different simulation times (450 s, 900 s, 1800 s, 3600 s). Each simulation is repeated ten times. During the measurements, we monitor the run-time, the residual set size (RSS), and the required disk capacity. In the following evaluation, we will focus on the worst-case measurements to identify the upper bound of system resources.

Before discussing our results, it is essential to note that multiple processes are spawned during a simulation. Their sequential and parallel execution as well as the invocation order affect the run-time, consumed memory, and required disk capacity. Some spawned processes belong to the Omnet++ framework and function as a wrapper, e.g., to run the compiled simulation file. They show little CPU usage and consume a constant amount of memory. While all 2000 measured simulation runs, they have never exceeded 30 MB. Thus, we decided to neglect them in the evaluation. When running a simulation with TAWNS, two kinds of processes require many system resources. The first process is of the TAWNS executable that initializes all modules required for the scenario, simulates the network and wireless communication part, and spawns subprocesses to generate the audios. Those (sub-)processes are the second kind and generate the acoustic data for each sensor node. To reduce the risk of memory overruns during




(a) Constant number of nodes (b) Constant number of sources

**Fig. 3.** Exemplary visualization of mean maximal run-times for a simulation time of 1800 s in scenario II

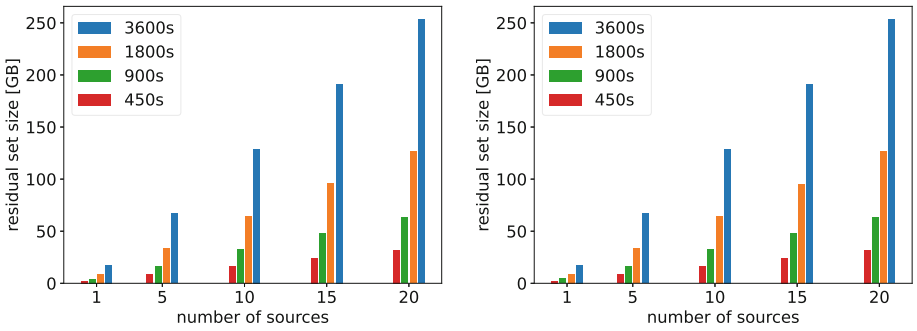
data generation, the (sub-)processes are spawned sequentially due to their high memory consumption.

### 5.1 Run-Time

Due to page limitations, we focus on the run-time for scenario II and a simulation time of 1800 s and 20 sources (see Fig. 3). The run-time increases by the number of nodes from about 2.0 min to about 5.9 min, then 10.5 min and 19.2 min until it reaches 32 min. Meanwhile, the increasing number of sources seems to have only little impact. While the run-time increases exponentially with the number of nodes, it grows more or less linearly with the number of sources. These trends are profound for low simulation times in both scenarios. For the sake of clarity, we did not visualize the relatively high standard deviation that proved to be caused by the different seeds. Our analysis has shown that the standard deviation stays below one minute when using the same seed multiple times. Note that the simulation time affects the run-time linearly.

### 5.2 Memory

We focus on the residual set size (RSS), the process's actual physical memory. In order to give a worst-case estimate, we focus on the mean over the maximal RSS measured for each run. For the sake of clarity we dropped the standard deviation that always stayed below 0.5 GB. In Fig. 4, we visualize the impact of the number of sources which is the dominant factor besides the simulation time. According to our data, the number of nodes does not affect the RSS significantly. When focusing on one simulation time, e.g., 3600 s in both scenarios, it becomes apparent that the RSS increases with the number of sources from about 17 GB to 67 GB, then 129 GB and 190 GB until it reaches about 250 GB. When the simulation time declines, so does the RSS. These observations can be made for




(a) Scenario I (b) Scenario II

**Fig. 4.** Visualization of RSS for both scenarios when number of sound sources increases

both scenarios. We can summarize that memory consumption is proportional to the number of sources and simulation time. For a simulation time of 3600 s and 20 sources, it requires 250 GB of memory. Our analysis shows that the processes to generate the audio mixtures cause the high RSS. More specifically, it increases the moment when the Scaper library starts to generate the audio mixtures, so we think that this is an issue within the Scaper library. Still, we are unsatisfied with the high memory consumption and are currently working on a solution for the next version of TAWNS.

### 5.3 Disk Capacity

At least two kinds of information are stored on the disk permanently when running a simulation. For network and wireless communication, that is the network statistics, such as the routing tables and the wireless traffic. As such data depends on the scenario, we neglect that for our evaluation and focus on the mandatory disk capacity when generating the audio files. The audio mixtures are created using Scaper, whose audio mixtures are always stored in the wave format. The mixtures can be encoded by other audio codecs (supported by our simulator), but the audio generation itself requires storing the data in wave format first. With that in mind, we can calculate the upper bound that is met for the last simulated node by summarizing the consumed disk capacity for all previous audios (with the optional applied audio codec) and adding the consumed capacity of the current node. All interdependences mentioned above can be summarized in the following formula:

$$C = \sum_{j \in \{n_0, \ldots, n_{i-1}\}} A_j(f_j \cdot b_j \cdot s_{n_i} + o_j) + f_{n_i} \cdot b_{n_i} \cdot s_{n_i} + o_{n_i} \tag{3}$$

with $n_i$ referring to the node i, $A_j$ referring to the applied audio codec of node j, $f_j$ referring to the sampling frequency of node j, $b_j$ referring to the bit-depth of

node j, $s_j$ referring to the recording time of node j in seconds, and $o_j$ referring to the header information of the WAVE format. As the sampling frequency and bit-depth are constants and the disk capacity saved by the applied audio codecs might also be regarded as a scaling factor, the required disk capacity scales with the number of simulated nodes and the recorded time. During our above-specified scenarios, which use FLAC audio codec, we measured a maximum disk capacity of at most 2,823.13 MB for 3600 s simulation time, which is about half when generating audios in WAVE format. It declines almost linear to the simulation time.

## 6 Conclusion

In this paper, we presented a solution for the challenge of comparing, reproducing, and conducting research in wireless acoustic sensor networks. Our simulator does not only provide the option to compare and reproduce research results in WASNs, but it also allows everyone to evaluate research questions without building and deploying expensive wireless acoustic sensor devices in the field. In order to achieve that, we used the Omnet++ simulator and combined and extended the libraries INET and Scaper. Our simulator can be equally used when focusing on network communication in WASN or generating acoustic sounds in a 3-D environment. As a starting pitch, we shared ten sets of model parameters derived from real measurements that can be used for generating sound mixtures. We also evaluated the performance of our simulator. For now, we recommend running the simulator on clusters (e.g., in a docker image), especially when performing long simulation runs or using many sound sources. In the future, there will always be some technical improvements continuously added to TAWNS, extending its features and making it easier to use. For example, we plan to support other sound simulation frameworks, such as Pyroomacoustic, to fit indoor acoustics better.

## References

1. Audaspace. https://audaspace.github.io/. Accessed 18 Aug 2022
2. FMOD. https://www.fmod.com/. Accessed 18 Aug 2022
3. INET Framework. https://inet.omnetpp.org/. Accessed 18 Aug 2022
4. The Network Simulator - ns-2. https://www.isi.edu/nsnam/ns/. Accessed 18 Aug 2022
5. OMNeT++ Discrete Event Simulator. https://omnetpp.org/. Accessed 18 Aug 2022
6. OpenAL Soft. https://openal-soft.org/. Accessed 18 Aug 2022
7. Panda3D. https://www.panda3d.org/. Accessed 18 Aug 2022
8. Loudness normalisation and permitted maximum level of audio signals. EBU European Broadcasting Union (2020)
9. Abdel-Malek, M.A., Akkaya, K., Bhuyan, A., Ibrahim, A.S.: A proxy signature-based swarm drone authentication with leader selection in 5G networks. IEEE Access **10**, 57485–57498 (2022)

10. Ahrens, J., Geier, M., Spors, S.: The SoundScape renderer: a unified spatial audio reproduction framework for arbitrary rendering methods. In: 124th Convention of the Audio Engineering Society. Audio Engineering Society (2008)
11. Blumstein, D.T., et al.: Acoustic monitoring in terrestrial environments using microphone arrays: applications, technological considerations and prospectus. J. Appl. Ecol. **48**(3), 758–767 (2011)
12. Bothe, A., Aschenbruck, N.: BonnMotion 4 – taking mobility generation to the next level. In: 2020 IEEE 39th International Performance Computing and Communications Conference (IPCCC), pp. 1–8 (2020)
13. Brackenbury, J.H.: Power capabilities of the avian sound-producing system. J. Exp. Biol. **78**(1), 163–166 (1979)
14. Brumm, H.: Song amplitude and body size in birds. Behav. Ecol. Sociobiol. **63**(8), 1157 (2009)
15. Cobos, M., Antonacci, F., Alexandridis, A., Mouchtaris, A., Lee, B.: A survey of sound source localization methods in wireless acoustic sensor networks. Wirel. Commun. Mob. Comput. **2017** (2017)
16. Collier, T.C., Kirschel, A.N.G., Taylor, C.E.: Acoustic localization of antbirds in a Mexican rainforest using a wireless sensor network. J. Acoust. Soc. Am. **128**(1), 182–189 (2010)
17. Community, B.O.: Blender - a 3D modelling and rendering package (2018)
18. Damarla, T., Kaplan, L.M., Whipps, G.T.: Sniper localization using acoustic asynchronous sensors. IEEE Sens. J. **10**(9), 1469–1478 (2010)
19. Hill, A.P., Prince, P., Piña Covarrubias, E., Doncaster, C.P., Snaddon, J.L., Rogers, A.: AudioMoth: evaluation of a smart open acoustic device for monitoring biodiversity and the environment. Methods Ecol. Evol. **9**(5), 1199–1211 (2018)
20. Lopez, P.A., et al.: Microscopic traffic simulation using SUMO. In: 2018 21st International Conference on Intelligent Transportation Systems (ITSC), pp. 2575–2582 (2018)
21. Rhinehart, T.A., Chronister, L.M., Devlin, T., Kitzes, J.: Acoustic localization of terrestrial wildlife: current practices and future opportunities. Ecol. Evol. **10**(13), 6794–6818 (2020)
22. Riley, G.F., Henderson, T.R.: The ns-3 network simulator. In: Wehrle, K., Güneş, M., Gross, J. (eds.) Modeling and Tools for Network Simulation, pp. 15–34. Springer, Berlin, Heidelberg (2010). https://doi.org/10.1007/978-3-642-12331-3_2
23. Salamon, J., MacConnell, D., Cartwright, M., Li, P., Bello, J.P.: Scaper: a library for soundscape synthesis and augmentation. In: 2017 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA), pp. 344–348 (2017)
24. Scheibler, R., Bezzam, E., Dokmanić, I.: Pyroomacoustics: a python package for audio room simulation and array processing algorithms. In: 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 351–355 (2018)
25. Sommer, C., et al.: Veins: the open source vehicular network simulation framework. In: Virdis, A., Kirsche, M. (eds.) Recent Advances in Network Simulation. EICC, pp. 215–252. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-12842-5_6
26. Whytock, R.C., Christie, J.: Solo: an open source, customizable and inexpensive audio recorder for bioacoustic research. Methods Ecol. Evol. **8**(3), 308–312 (2017)