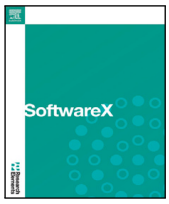
SoftwareX 30 (2025) 102086

Contents lists available at ScienceDirect

SoftwareX

journal homepage: www.elsevier.com/locate/softx



Original software publication

Agenet: Age of Information evaluation in wireless networks

Chathuranga M. Wijerathna Basnayaka [a,*], Nuno Fachada [a,b]

[a] COPELABS, Universidade Lusófona, Campo Grande 376, 1749-024 Lisboa, Portugal
[b] Center of Technology and Systems (UNINOVA-CTS) and Associated Lab of Intelligent Systems (LASI), 2829-516 Caparica, Portugal

ARTICLE INFO

Keywords:
Age of Information
Short-packet transmission
Wireless networks

ABSTRACT

The Age of Information (AoI) has emerged as a critical performance metric for evaluating time-sensitive wireless communications systems, where maintaining freshness of information and transmission reliability is crucial. In modern ultra-reliable low-latency communication networks, short-packet transmissions are essential for energy efficiency and low latency. This paper introduces Agenet, an open-source Python package designed to estimate AoI in cooperative wireless networks. It implements a system model over Rayleigh fading channels, combining finite blocklength information theory and AoI analysis. The package offers tools to calculate signal-to-noise ratio, block error rate, and both theoretical and simulated AoI. By enabling analysis of AoI performance under various network configurations, Agenet supports research and development of efficient wireless systems for time-critical applications.

Code metadata

Current code version	v1.0.0
Permanent link to code/repository used for this code version	https://github.com/ElsevierSoftwareX/SOFTX-D-24-00576
Permanent link to Reproducible Capsule	Not applicable
Legal Code License	MIT License
Code versioning system used	Git
Software code languages, tools, and services used	Python
Compilation requirements, operating environments & dependencies	Python ≥ 3.9, Matplotlib ≥ 3.8, NumPy ≥ 1.26, Pandas ≥ 2.3, Rich ≥ 13.8, Rich-argparse ≥ 1.5, Rich-tools ≥ 0.5.1, SciPy ≥ 1.20
If available Link to developer documentation/manual	https://cahthuranag.github.io/agenet/
Support email for questions	basnayakac8@gmail.com

1. Introduction

1.1. Motivation and significance

The rapid advancement of wireless communication technologies has triggered the era of the Internet of Things (IoT), cyber–physical systems, and robotic networks, offering unprecedented opportunities to enhance efficiency across various aspects of our daily lives . These technologies enable the collection and transmission of massive quantities of data through wireless networks, facilitating communication amongst numerous nodes, including sensors, actuators, machines, autonomous vehicles, Unmanned Aerial Vehicles (UAVs) and a wide array of smart devices. In this context, timely delivery of information has become crucial, particularly for mission critical applications where outdated data can lead to severe consequences. However, traditional performance metrics such as latency and delay have proven inadequate in capturing the essence of information freshness. To address this limitation, researchers have introduced a novel performance metric called the age of information (AoI) [1]. AoI measures the time elapsed since the generation of the most recent update successfully received at the destination, providing a comprehensive understanding of information freshness by considering both the generation time and the successful delivery of updates.

The AoI approach, first introduced in the 1990s for analysing time consistency in real-time databases [2], has recently attracted increasing interest within the domain of wireless communication research. This

* Corresponding author.
E-mail address: basnayakac8@gmail.com (Chathuranga M. Wijerathna Basnayaka).

https://doi.org/10.1016/j.softx.2025.102086
Received 28 October 2024; Received in revised form 1 February 2025; Accepted 4 February 2025
Available online 20 February 2025

Table 1
Overview of agenet.

Software Purpose	Python package for estimating Age of Information (AoI) in cooperative wireless networks
Research Fields	Wireless Communications, Networking, Information Theory
Key Features	Average AoI estimation, SNR and block error rate calculations, Parameter sweep analysis with visualisation tools
Target Users	Wireless communications researchers, Network system designers, Academic staff and graduate students
Primary Applications	Time-critical cooperative wireless systems, Industrial IoT networks, UAV communications, Disaster monitoring systems

renewed attention is largely due to its relevance in the evaluation of status update systems and mission-critical applications. Unlike traditional timeliness measures, such as packet delay or round-trip latency, the AoI metric provides a broader and more precise measure of information freshness at the receiver. It shifts the focus from transmission delay alone to the overall freshness of the received updates, making it an alternative to conventional performance metrics [3]. AoI is particularly effective in scenarios where latency-based metrics fall short. For instance, even when data transmission occurs with minimal delay, the information might still be outdated if the rate of packet generation is low. Conversely, high-frequency updates may still lack freshness when significant queuing delays arise during transmission. As a result, AoI is recognised as a critical metric for assessing performance in systems requiring timely updates, particularly those supporting mission-critical functions. The significance of AoI is amplified in applications where real-time decisions rely on the most current data. For example, in autonomous driving systems, even slight outdated sensor data could result in catastrophic consequences, such as collisions. Similarly, in IoT networks, ensuring that the receiver has up-to-date information is essential since outdated status data can lead to system failures or serious repercussions. Applications such as UAV networks, real-time monitoring systems, and autonomous vehicles depend heavily on timeliness metrics to ensure accurate and up-to-date information delivery [4]. Guaranteeing data freshness in these contexts is crucial, as outdated information can compromise safety and performance.

Several types of theoretical studies have analysed AoI in different wireless networks [5–7]. However, as per the authors' knowledge, there is no software available that is based on AoI analysis. This indicates a significant gap in a field where practical implementation tools are needed to validate and implement AoI-based solutions in real-world wireless network scenarios.

This research software, Agenet, was motivated by the need to analyse AoI under constraints such as short packet transmissions. Its purpose is to help researchers explore the impact of system parameters, such as block length and transmission power, on information freshness. By addressing these critical aspects of AoI in wireless communication systems, this software package can provide valuable insights for the design and optimisation of future wireless networks that prioritise information freshness. Table 1 provides a summary of the software's purpose, features, target domains and primary applications.

1.2. Problem formulation

The problem addressed by the proposed software consists in estimating the AoI in a short packet decode-and-forward (DF) cooperative relaying system, as illustrated in Fig. 1. In this system, a source node (S) transmits newly generated updates to a destination node (D) with the assistance of a relay node (R). The communication process employs a Time Division Multiple Access (TDMA) medium access control protocol structured into two distinct time slots within each transmission block. During the initial slot, S transmits data to R. Subsequently, in the second slot, R processes the received data through decoding and forwards it to D. Due to the challenging propagation environment, it is assumed that there is no direct communication link between S and D [7,8]. This assumption is based on unfavourable line-of-sight (LoS) conditions, which prevent reliable direct transmission. This assumption reflects practical scenarios where physical constraints in the environment prevent direct communication between the source and destination, such as dense urban environments with high-rise buildings, industrial settings with heavy machinery and metallic obstacles, and underground-to-surface communications. In these environments, relay assistance becomes essential for maintaining reliable communication links.

1.2.1. Signal reception model

The signal reception at each node can be expressed as:

$$Y_1 = \sqrt{P_S} H_{SR} X_1 + W_{SR}, \tag{1}$$

$$Y_2 = \sqrt{P_R} H_{RD} X_2 + W_{RD}, \tag{2}$$

where Y_1 and Y_2 represent the received signals at the relay and destination nodes, respectively. In these equations, H_{ij} denotes the channel coefficient between nodes i and j, with $i \in \{S, R\}$ and $j \in \{R, D\}$. The term P_i denotes the transmission power at node i, while X_1 and X_2 are the transmitted signals. W_{ij} represents the Additive White Gaussian Noise (AWGN) in the channel, characterised by zero mean and variance σ_{ij}^2. The channel coefficient H_{ij} follows a complex normal distribution for Rayleigh fading channels, denoted as $H_{ij} \sim \mathcal{CN}(0, 1)$.[1]

1.2.2. Channel gain modelling

The probability density function (PDF)[2] of the small-scale channel gain g_{ij} can be expressed as:

$$f_{g_{ij}}(z) = e^{-z}, z \geq 0. \tag{3}$$

The large-scale channel gain α_{ij} between the communication nodes is formulated as:

$$-10\log(\alpha_{ij}) = 20\log(d_{ij}) + 20\log\left(\frac{4\pi f_c}{c}\right), \tag{4}$$

where f_c and d_{ij} represent the carrier frequency and distance between nodes i and j, respectively. Variable c denotes the speed of light in free space. Taking into account both small-scale and large-scale channel gains, the channel coefficient between the communication nodes is expressed as:

$$H_{ij} = \sqrt{\alpha_{ij} g_{ij}}. \tag{5}$$

1.2.3. SNR calculation

The instantaneous received signal-to-noise ratio (SNR) γ_j at the receiving node j is given by:

$$\gamma_j = \frac{\alpha_{ij} g_{ij} P_i}{\sigma_{ij}^2}. \tag{6}$$

[1] $\mathcal{CN}(0, 1)$ represents the complex normal distribution with zero mean and unit variance.

[2] $f_X(x)$ denotes the probability density function of an arbitrary random variable X.

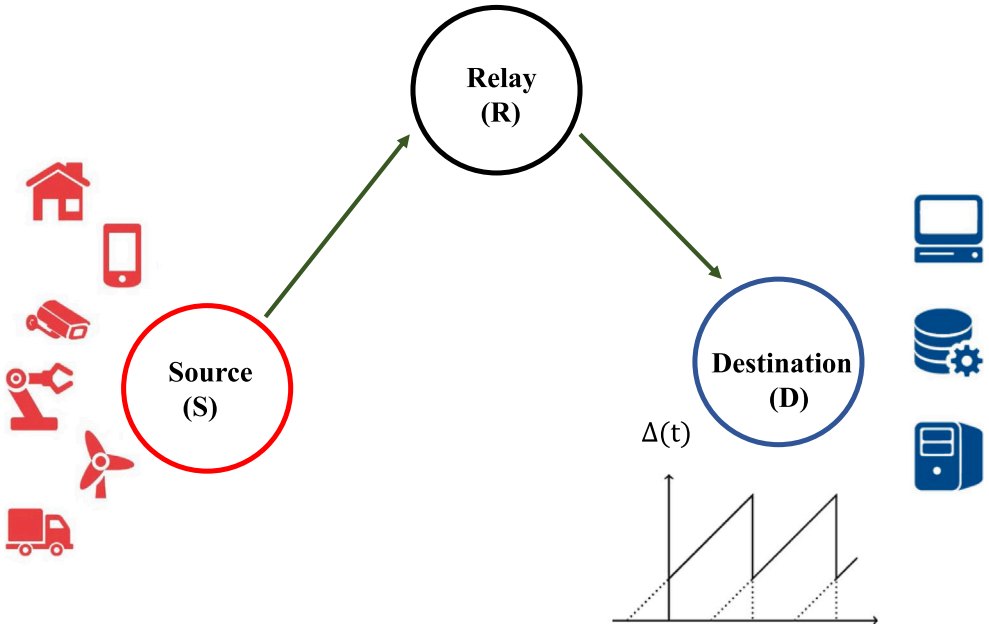


Fig. 1. System model for short packet transmission based wireless relay system.

The average SNR can be calculated as:

$$\bar{\gamma_j} = \frac{\alpha_{ij} P_i}{\sigma_{ij}^2}. \tag{7}$$

1.2.4. Block error probability

In the context of the DF relay protocol, the overall decoding error probability is expressed as:

$$\varepsilon = \varepsilon_R + \varepsilon_D(1 - \varepsilon_R), \tag{8}$$

where ε_R and ε_D represent the block error probabilities at R and D, respectively. It is assumed that the channel remains static throughout each transmission block, meaning the fading coefficients remain constant for this duration. Building upon Polyanskiy's work on short packet communication [9] and assuming perfect channel state information at the receiver, the expected block error probability for a given block length is formulated as:

$$\varepsilon_j = \mathbb{E}\left[Q\left(\frac{n_{ij} C(\gamma_j) - k}{\sqrt{n_{ij} V(\gamma_j)}} \right) \right], \tag{9}$$

where $\mathbb{E}[.]$ denotes the expectation operator, n_{ij} denotes the block length allocation for the transmission in channel uses (cu), and $Q(x) = \frac{1}{\sqrt{2\pi}} \int_x^\infty e^{-\frac{t^2}{2}} dt$. The term $V(\gamma_j)$ represents the channel dispersion, defined as $V(\gamma_j) = \frac{\log_2{}^2 e}{2}(1 - \frac{1}{(1+\gamma_j)^2})$ in (bits/cu)^2. Additionally, $C(\gamma_j) = \log_2(1+\gamma_j)$ represents the channel capacity of a complex AWGN channel in bits/cu, and k denotes the number of information bits per block. In this analysis, one channel use corresponds to one transmitted symbol (1 cu/symbol) and it is assumed that one symbol transmits one bit. Thus, for simplicity of implementation, we use bits as the unit of measurement for block length in the software package. For Rayleigh fading channel conditions, ε_j can be formulated as:

$$\varepsilon_j = \int_0^\infty f_{\gamma_j}(z) Q\left(\frac{n_{ij} C(\gamma_j) - k}{\sqrt{n_j V(\gamma_j)}} \right) dz, \tag{10}$$

where $f_{\gamma_j}(z)$ denotes the PDF of the received SNR γ_j at the communication node j. Due to the complexity of the Q-function, deriving a closed-form expression for the overall decoding error probability is challenging. Therefore, an approximation technique is employed, as proposed in [10,11], to simplify (10):

$$\varepsilon_j \approx \int_0^\infty f_{\gamma_j}(z) \Theta_j(z), \tag{11}$$

where $\Theta_j(z)$ represents a linear approximation of $Q\left(\frac{n_{ij} C(\gamma_j) - k}{\sqrt{n_{ij} V(\gamma_j)}} \right)$, which can be expressed as :

$$\Theta_j(z) = \begin{cases} 1, & \gamma_j \le \phi_j, \\ \frac{1}{2} - \beta_j \sqrt{n_{ij}}(\gamma_j - \psi_j), & \phi_j < \gamma_j < \delta_j, \\ 0, & \gamma_j \ge \delta_j, \end{cases} \tag{12}$$

where $\beta_j = \frac{1}{2\pi\sqrt{2^{\frac{2k}{n_{ij}}} - 1}}$, $\psi_j = 2^{\frac{k}{n_{ij}}} - 1$, $\phi_j = \psi_j - \frac{1}{2\beta_j\sqrt{n_{ij}}}$, and $\delta_j = \psi_j + \frac{1}{2\beta_j\sqrt{n_{ij}}}$. Utilising (11) and (12), a closed-form expression for the block error probability is derived:

$$\varepsilon_j \approx 1 - \left(\frac{\beta_j \sqrt{n_{ij}} P_i \alpha_{ij}}{\sigma_{ij}^2} \right) \left(e^{-\frac{\phi_i \sigma_{ij}^2}{\alpha_{ij} P_i}} - e^{-\frac{\delta_j \sigma_{ij}^2}{\alpha_{ij} P_i}} \right). \tag{13}$$

Finally, by substituting (13) into (8), a closed-form expression for the overall error probability is obtained:

$$\varepsilon \approx 1 - \left[\left(\frac{\beta_R \beta_D \sqrt{n_{SR} n_{RD}} \times P_S P_R \alpha_{SR} \alpha_{RD}}{\sigma_{SR}^2 \sigma_{RD}^2} \right) \left(e^{-\frac{\phi_R \sigma_{SR}^2}{\alpha_{SR} P_S}} - e^{-\frac{\delta_R \sigma_{SR}^2}{\alpha_{SR} P_S}} \right) \left(e^{-\frac{\phi_D \sigma_{RD}^2}{\alpha_{RD} P_R}} - e^{-\frac{\delta_D \sigma_{RD}^2}{\alpha_{RD} P_R}} \right) \right]. \tag{14}$$

1.2.5. Age of information analysis

The AoI analysis in this wireless relay system considers the S generates new status updates every transmission circle to keep the information at the D as fresh as possible. Then, generated updates are transmitted to its destination using the relay. If the generation time of the freshest update received at destination time stamp t is $g(t)$, then the AoI can be defined as a random process as :

$$\Delta(t) = t - g(t). \tag{15}$$

As illustrated in Fig. 2, it is assumed that at $t = 0$ the measurements of the AoI starts and the AoI at the destination is set to $\Delta(0) = \Delta_0$. The

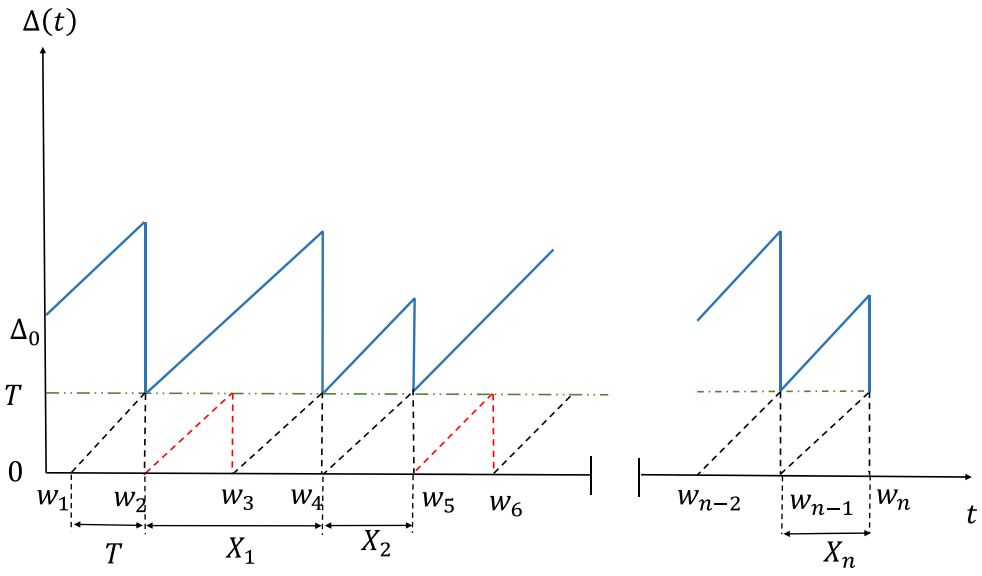


Fig. 2. Evolution of AoI $\Delta(t)$ at destination with the time : The source generate updates at time stamps $w_1, w_2, \ldots, w_{n-1}$ and the destination receive these updates at time stamps $w_2, w_3, \ldots, w_n$.

source produces updates at time stamps $w_1, w_2, \ldots, w_{n-1}$. Updates can be generated at the beginning of any time cycle following the generate-at-will update generation model. The destination receives these updates at time stamps $w_2, w_3, \ldots, w_n$. As depicted in Fig. 2, data update i is transmitted from the source at time stamp $t = g_i$, and it is successfully delivered to its destination at time stamp $w_{i+1} = g_i + T$, where $T = (n_{SR} + n_{RD})T_s$ is the total time allocated for a transmission cycle, with T_s representing the symbol duration. Therefore, if the update packet delivered successfully at time w_{i+1}, the AoI at the destination can be calculated as

$$\Delta(w_{i+1}) = T. \tag{16}$$

The AoI increases linearly until the next update is successfully delivered to the destination. As the example in Fig. 2 shows, since one packet fails to be decoded at time w_3, $\Delta(t)$ continues to increase linearly. Here, X is the inter-departure time between two consecutive successfully received status updates at the destination; it is a geometric random variable with mean $E[X] = \frac{T}{1-\epsilon}$ and second moment $E\left[X^2\right] = \frac{T^2(1+\epsilon)}{(1-\epsilon)^2}$. It assumes that the end-to-end delay of each successfully received update is always a constant given by $E[Y] = T$. Then, by applying graphical methods to the saw-tooth age waveform in Fig. 2, the average AoI (AAoI) can be calculated [4] as follows:

$$\Delta^{AAoI} = \frac{E[X^2]}{2E[X]} + T. \tag{17}$$

Then, by substituting $E[X] = \frac{T}{1-\epsilon}$ and $E[X^2] = \frac{T^2(1+\epsilon)}{(1-\epsilon)^2}$ into (17), the expression for the AAoI at the destination in the relay network can be obtained:

$$\Delta^{AAoI} = \frac{T}{2} + \frac{T}{1-\epsilon}. \tag{18}$$

In the Agenet package, (18) is employed for the theoretical AAoI estimation.

2. Software description

The theoretical foundations of AoI analysis in wireless networks form the basis for their practical implementation in the Agenet software package. Agenet converts the mathematical models and concepts presented in the previous section into a user-friendly Python framework that enables researchers to analyse AoI under various network configurations. Standing on a modular architecture, the package implements all key components of the theoretical model, such as SNR calculations, block error probability estimation, and AoI analysis. The software offers a front-end script and functions to simulate the AAoI in wireless networks. As depicted in Fig. 3, Agenet's architecture comprises five modules: *cli*, *simulation*, *snratio*, *blkerr*, and *aaoi*, each responsible for specific aspects of the AAoI estimation process. Table 2 provides an overview of the main functions, their corresponding modules, as well as descriptions of their roles in the software.

The script, `agenet`, provides a command-line user interface, accepting model parameters, running the simulation, and optionally displaying and/or exporting results, allowing the package to be used by researchers unfamiliar with Python. All functionality related with this command line interface is defined in the *cli* module. In particular, the module's `_main()` function serves as the entry point for the script, handling arguments, initiating simulations, and managing output generation, which can include plots and/or tables, either displayed in the screen or exported to files.

The `simulation` module serves as the core of the software, managing the overall simulation process. Its low-level `_sim()` function performs a single simulation, estimating the AAoI of the system for a given number of events. However, this function is not intended for direct use by clients. For the purpose of performing a single simulation and obtaining the AAoI, client code should invoke the higher-level `sim()` function, which performs parameter validation and returns additional metrics such as the SNR and block error rates. The `ev_sim()` has a similar purpose, but performs multiple runs, providing statistically significant results. Both `sim()` and `ev_sim()` leverage the lower-level `_sim()` function. The `multi_param _ev_sim()` function extends this further by iterating over various parameter combinations, allowing for comprehensive system analysis under different conditions. It is the function invoked by `_main()` via the `agenet` command line script.

During a simulation run, `_sim()` first utilises functions from the *snratio* module to compute the SNR. It calls `snr()` to calculate instantaneous SNR values for both the source node and the relay. For average SNR calculations, it uses `snr_avg()`. These SNR values are crucial for subsequent error rate calculations. Next, `_sim()` interacts with the *blkerr* module to determine block error rates. It uses `block_error()` for instantaneous error rates and `block_error_th()` for theoretical error rates. These calculations are performed for both the source

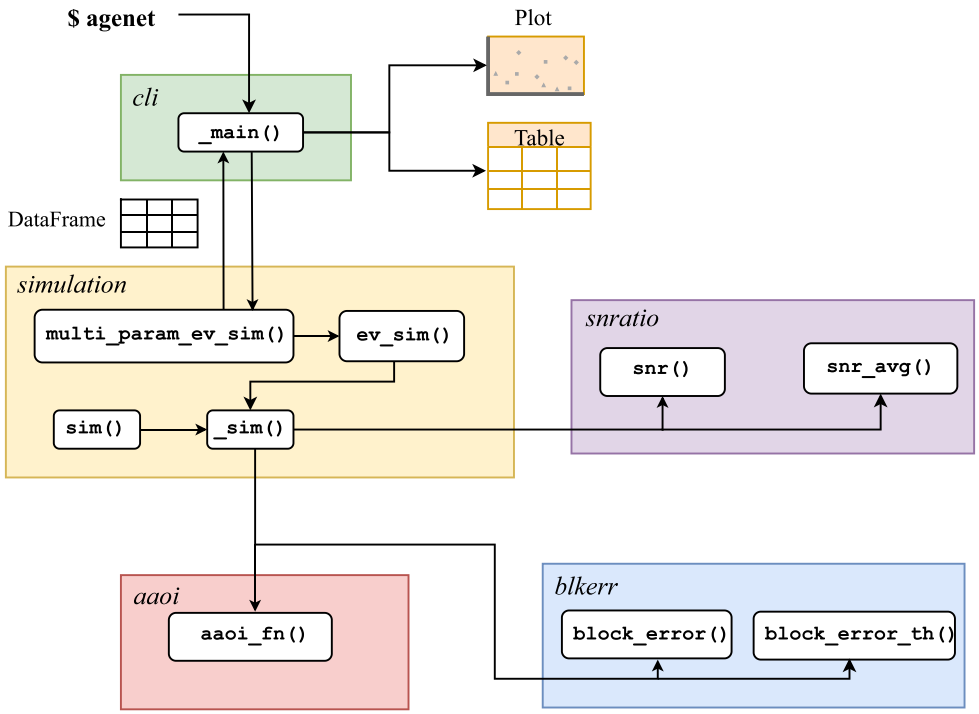


Fig. 3. Architecture of the Agenet package. Coloured boxes represent Python modules, while rounded boxes highlight the main functions in each module. Arrows illustrate typical user and data workflows when using the package. Users will typically use the package through the `agenet` command line application (shown top right), or via one of the functions in the *simulation* module. Note, however, that functions starting with an underscore should not be invoked directly by users in common usage scenarios.

Table 2
Agenet modules, their respective functions, and the descriptions of each function.

Module	Function	Description
simulation	`_sim()`	Low-level function for simulating a communication system and obtaining the AAoI. It performs a single simulation run, estimating the AAoI of the system for a given number of events, while calculating the theoretical AAoI using Eq. (18).
	`sim()`	High-level function that wraps `_sim()`, providing parameter validation and returning additional metrics such as SNR and block error rates.
	`ev_sim()`	Executes multiple simulation runs to calculate expected values, enhancing statistical reliability.
	`multi_param_ev_sim()`	Allows for simulations across multiple parameter combinations, facilitating comprehensive system analysis.
snratio	`snr()`	Computes the instantaneous SNR based on Eq. (6).
	`snr_avg()`	Calculates the average SNR as defined in Eq. (7).
blkerr	`block_error()`	Computes the instantaneous block error rate for the short-packet communication based on Eq. (9).
	`block_error_th()`	Calculates the theoretical block error rate as per Eq. (14).
aaoi	`aaoi_fn()`	Implements the AAoI calculation based on the model illustrated in Fig. 2 and formalised in Eq. (17).
cli	`_main()`	Entry point for the `agenet` command, handling parameter input, simulation execution, and result output such as tables and plots.

node and the relay. The AAoI calculations are performed by calling `aaoi_fn()` from the *aaoi* module. This function takes the generation and receiving time of the delivered packets during the simulation process, returning the AAoI.

The architecture of Agenet is designed to closely emulate the theoretical model presented in Section 1. By structuring the code to directly implement equations such as (14) and (17), Agenet provides a practical tool for users to explore and validate theoretical predictions about AoI in wireless relay networks. In other words, this close alignment between theory and implementation facilitates the translation of mathematical models into practical simulations, enabling researchers to gain insights into the behaviour of AoI in complex network scenarios.

3. Illustrative examples

This section demonstrates the usage of Agenet through practical examples, showcasing its capabilities in analysing the AoI in wireless networks.

3.1. Installation

Agenet can be installed via PyPI or directly from GitHub, ensuring all necessary dependencies are automatically installed. For users preferring a stable release, installation via PyPI is recommended:

```
$ pip install agenet
```

```
$ git clone https://github.com/cahthuranag/agenet.git
$ cd agenet
$ python -m venv env
$ source env/bin/activate  # On Windows use 'env\Scripts\
    activate'
$ pip install -e .[dev]
$ pre-commit install
```

Fig. 4. Development setup commands for Agenet, showing the sequence of steps to clone the repository, create a virtual environment, install the package in editable mode with development dependencies, and configure pre-commit hooks.

```
frequency   num_events   num_bits   info_bits   power   distance   N0
50000000    100.0        400.0      350.0       0.001   500.0      1e-13
50000000    100.0        400.0      350.0       0.005   500.0      1e-13
50000000    100.0        400.0      350.0       0.01    500.0      1e-13
50000000    100.0        400.0      350.0       0.05    500.0      1e-13
50000000    100.0        400.0      350.0       0.1     500.0      1e-13
50000000    100.0        400.0      350.0       0.5     500.0      1e-13
num_bits_2   info_bits_2   power_2   distance_2   N0_2
400.0        350.0         0.001     500.0        1e-13
400.0        350.0         0.005     500.0        1e-13
400.0        350.0         0.01      500.0        1e-13
400.0        350.0         0.05      500.0        1e-13
400.0        350.0         0.1       500.0        1e-13
400.0        350.0         0.5       500.0        1e-13
aaoi_theo   aaoi_sim    snr1_avg     snr2_avg     blkerr1_th   blkerr2_th
0.31609     0.325381    0.9118906    0.911890     0.5946200    0.594620
0.0931378   0.091128    4.5594532    4.559453     0.1667744    0.166774
0.0816209   0.082119    9.1189065    9.118906     0.0872947    0.087294
0.0737880   0.073318    45.594532    45.59453     0.0181209    0.018120
0.0728859   0.073080    91.189065    91.18906     0.0091030    0.009103
0.0721759   0.072462    455.94532    455.9453     0.0018274    0.001827
```

Fig. 5. Table displayed in the terminal by the `agenet` command due to the use of the `-t` parameter. The table shows simulation results obtained by sweeping over transmission power levels.

For those seeking the latest development version, installation directly from GitHub is possible:

```
$ pip install git+https://github.com/cahthuranag/
agenet.git#egg=agenet
```

Developers wishing to contribute to Agenet or modify the code should follow steps in Fig. 4 for a development setup.

This development setup clones the repository, creates a virtual environment, installs Agenet in editable mode with development dependencies, and sets up pre-commit hooks for code quality checks. This configuration allows for easier testing and development of the Agenet package.

3.2. Command-line interface

Agenet provides a comprehensive command-line interface, `agenet`, that enables users to configure and execute simulations with a high degree of customisation. This interface allows researchers to easily adjust network parameters, simulation settings, and output options to estimate AoI under various conditions. When run without any options, the `agenet` command displays the help information, providing an overview of available parameters and their default values. This ensures that users are aware of the simulation options before running an actual simulation. To execute a simulation, at least one parameter must be specified. Table 3 lists all available parameters and their respective default values.

This comprehensive set of command-line parameters allows users to configure various aspects of the simulation, including general simulation settings, source and relay node characteristics, and output options. By offering such detailed control, `agenet` enables users to model and analyse a wide range of wireless network scenarios, facilitating in-depth studies of AoI in different system configurations.

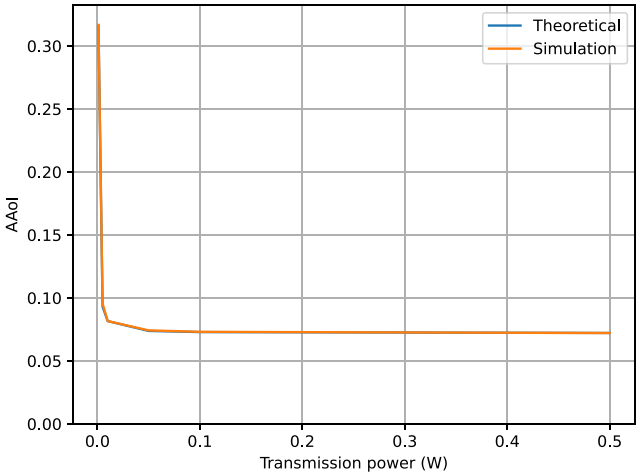


Fig. 6. Plot generated by the `agenet` command via the `-p` parameter. The plots shows simulation results obtained by sweeping over transmission power levels.

To execute a basic simulation, at least one parameter must be specified. For instance, the following command executes 10 simulation runs and presents the results in a tabular format:

```
$ agenet --num-runs 10 -t
```

For more specific scenarios, users can customise parameters. For example, the following command simulates a network with a 6 GHz

Table 3
Command line arguments for the `agenet` script.

Category	Parameter	Description	Default
General	`-f, --frequency`	Signal frequency (Hz)	5e9
	`-e, --num-events`	Events per simulation	100
	`-r, --num-runs`	Number of simulation runs	10
	`-s, --seed`	Random number generator seed	Random
Source node	`--num-bits`	Total number of bits	400
	`--info-bits`	Number of information bits	350
	`--power`	Transmission power (W)	5e−3
	`--distance`	Node distance (m)	500
	`--N0`	Noise power (W)	1e−13
Relay node	`--num-bits-2`	Total bits in relay	`--num-bits`
	`--info-bits-2`	Info bits in relay	`--info-bits`
	`--power-2`	Relay transmission power (W)	`--power`
	`--distance-2`	Relay–destination distance (m)	`--distance`
	`--N0-2`	Relay noise power (W)	`--N0`
Output	`-t, --show-table`	Show results table	False
	`-o, --save-csv`	Save results to CSV	None
	`-p, --show-plot`	Show plot	False
	`--save-plot`	Save plot to file	None
	`--debug`	Debugging level	0

```
$ agenet --num-bits 500 --info-bits 400 --power 1e-2  \
   --distance 600  --num-bits-2 600 --info-bits-2 350 \
   --power-2 8e-3  --distance-2 400 500 600 \
   --save-plot aoi_vs_distance_2.png --save-csv results.csv
```

Fig. 7. Command line example demonstrating the use of `agenet` with asymmetric network parameters, performing a distance sweep analysis for the relay-to-destination link while saving both plot and numerical results to files.

```
$ python
>>> from agenet import sim, ev_sim, multi_param_ev_sim
>>> import numpy as np
>>> import pandas as pd
>>> frequency = 5e9
>>> num_events = 100
>>> num_bits = 400
>>> info_bits = 350
>>> power = 5e-3
>>> distance = 500
>>> N0 = 1e-13
>>> num_bits_2 = 600
>>> info_bits_2 = 500
>>> power_2 = 10e-3
>>> distance_2 = 300
>>> N0_2 = 2e-13
>>> num_runs = 100
>>> seed = 42
>>> powers = [1e-3, 1e-2, 1]
>>> distances = [300, 400, 500]
```

Fig. 8. The initialisation of simulation parameters, including system parameters (frequency, bits, power, distance, noise), relay node parameters, and parameter sweep arrays for transmission power and distance analysis.

frequency, 200 events per simulation run, 10 mW transmission power, and 600 m distance:

```
$ agenet -f 6e9 -e 200 --power 1e-2 --distance 600 -t
```

The `agenet` command also supports parameter sweeps, allowing researchers to analyse system behaviour across a range of values. The following command performs a sweep over transmission power levels, specifying a pseudo-random number generator seed, allowing for these results to be reproduced exactly:

```
$ agenet --power 1e-3 5e-3 1e-2 5e-2 1e-1 5e-1 -t -p
--seed 123
```

This command simulates six different power levels, displaying results in both tabular and graphical formats as illustrated in Figs. 5 and 6, respectively.

This simple example shows that the AAoI decreases with increasing transmission power, thus improving information freshness. At around 0.1 W of transmission power, the AAoI stabilises and reaches a minimum of approximately 0.72 s for the considered parameters. This

```
>>> aaoi_th, aaoi_sim, snr1, snr2, blkerr1, blkerr2 =
sim(frequency, num_events, num_bits, info_bits, power,
    distance, N0)
>>> print(f"Theoretical AAoI: {aaoi_th:.3f}")
Theoretical AAoI: 0.093
>>> print(f"Simulated AAoI: {aaoi_sim:.3f}")
Simulated AAoI: 0.096
>>> print(f"Relay SNR: {snr1:.2f} dB")
Relay SNR: 4.56 dB
>>> print(f"Destination SNR: {snr2:.2f} dB")
Destination SNR: 4.56 dB
>>> print(f"Block Error Rate at Relay: {blkerr1:.3f}")
Block Error Rate at Relay: 0.167
>>> print(f"Block Error Rate at Destination: {blkerr2:.3f}")
Block Error Rate at Destination: 0.0167
```

Fig. 9. Basic single simulation example using `sim()` function, showing the calculation of AAoI metrics (theoretical and simulated), along with SNR and block error rate measurements for both relay and destination nodes in a symmetric network configuration.

```
>>> results_as = sim(frequency, num_events, num_bits,
    info_bits,
power, distance, N0, num_bits_2, info_bits_2,
power_2, distance_2, N0_2)
>>> aaoi_th, aaoi_sim, snr1, snr2, blkerr1, blkerr2 =
    results_as
>>> print(f"Theoretical AAoI: {aaoi_th:.3f}")
Theoretical AAoI: 0. 107
>>> print(f"Simulated AAoI: {aaoi_sim:.3f}")
Simulated AAoI: 0.109
>>> print(f"Relay SNR: {snr1:.2f} dB,
Block Error Rate: {blkerr1:.3f}")
Realy SNR: 4.56 dB, Block Error Rate: 0.167
>>> print(f"Destination SNR: {snr2:.2f} dB,
Block Error Rate: {blkerr2:.3f}")
Destination SNR: 12.67 dB, Block Error Rate: 0.060
```

Fig. 10. Simulation example for asymmetric network configuration using `sim()` function, showing AAoI calculations along with SNR and block error rate measurements for both relay and destination nodes.

can be attributed to the minimal transmission errors at this power level. This example highlights that transmission power is a crucial factor in designing for optimal information freshness in a relay-based wireless communication system. For more complex network topologies, Agenet – and its `agenet` command – can simulate relay networks with distinct parameters for source and relay nodes, as in Fig. 7. This command models a two-hop network with different configurations for each node, saving the plot and the simulation results to the `aoi_vs_distance_2.png` and `results.csv` files, respectively. Note that it is only possible to generate plots when sweeping over one parameter. If more than one parameter is sweeped (or, conversely, if all parameters are simulated with only one value), the `agenet` command will not create a plot if requested, notifying the user of the root cause.

3.3. Using agenet as a python library

While the command-line interface provides a convenient way to run simulations, Agenet can also be used directly as a Python library. To do so, the required functions must be imported and simulation parameters set, as in Fig. 8.

3.3.1. Basic single simulation

The `sim()` function performs a single simulation run, calculating both theoretical and simulated AAoI values, along with SNR and block error rates for both relay and destination nodes, as in Fig. 9. For networks with different source and relay node configurations, `sim()` can evaluate asymmetric system performance by accepting distinct parameters for each node, as in Fig. 10.

3.3.2. Multiple simulation runs

Multiple simulation iterations can be performed using `ev_sim()` to generate statistically significant results for reliable performance analysis, as in Fig. 11. The output presents AAoI values from multiple iterations for improved accuracy.

3.3.3. Parameter sweep analysis

Parameter sweep analysis using `multi_param_ev_sim()` examines system behaviour across different parameter combinations to understand performance variations, as in Fig. 12. The results demonstrate how AAoI varies with changes in system parameters, as in Fig. 13.

```
>>> results=ev_sim(num_runs, frequency, num_events, num_bits,
info_bits, power, distance, N0, seed=42)
>>> aaoi_th, aaoi_sim, snr1, snr2, blkerr1, blkerr2 = results
>>> print(f"Average Theoretical AAoI: {aaoi_th:.3f}")
Average Theoretical AAoI: 0.093
>>> print(f"Average Simulated AAoI: {aaoi_sim:.3f}")
Average Simulated AAoI: 0.093
```

Fig. 11. Example code showing multiple simulation runs using `ev_sim()` to calculate average theoretical and simulated AAoI values with improved statistical significance.

```
>>> results_df, errors = multi_param_ev_sim(num_runs, [
    frequency],
[num_events], [num_bits], [info_bits], powers,
distances, [N0], seed=42)
>>> pd.set_option("display.max_columns", None)
>>> pd.set_option("display.width", None)
>>> print(results_df)
```

Fig. 12. Parameter sweep analysis using `multi_param_ev_sim()` to examine system behaviour across multiple parameter combinations.

```
>>> print(results_df)
   frequency  num_events  num_bits  info_bits  power  distance  N0
0  5.000000e+09         100       400        350  0.001       300  1.000000e-13
1  5.000000e+09         100       400        350  0.001       400  1.000000e-13
2  5.000000e+09         100       400        350  0.001       500  1.000000e-13
3  5.000000e+09         100       400        350  0.010       300  1.000000e-13
4  5.000000e+09         100       400        350  0.010       400  1.000000e-13
5  5.000000e+09         100       400        350  0.010       500  1.000000e-13
6  5.000000e+09         100       400        350  1.000       300  1.000000e-13
7  5.000000e+09         100       400        350  1.000       400  1.000000e-13
8  5.000000e+09         100       400        350  1.000       500  1.000000e-13

   num_bits_2  info_bits_2  power_2  distance_2  N0_2          aaoi_theory  aaoi_sim
0         400          350    0.001         300  1.000000e-13     0.116451  0.119679
1         400          350    0.001         400  1.000000e-13     0.177287  0.183603
2         400          350    0.001         500  1.000000e-13     0.316090  0.339969
3         400          350    0.010         300  1.000000e-13     0.075266  0.075528
4         400          350    0.010         400  1.000000e-13     0.077956  0.078123
5         400          350    0.010         500  1.000000e-13     0.081621  0.081887
6         400          350    1.000         300  1.000000e-13     0.072032  0.072106
7         400          350    1.000         400  1.000000e-13     0.072056  0.072097
8         400          350    1.000         500  1.000000e-13     0.072088  0.072121

     snr1_avg     snr2_avg     blkerr1_th  blkerr2_th
0    2.533030     2.533030     0.279449    0.279449
1    1.424829     1.424829     0.440413    0.440413
2    0.911891     0.911891     0.594620    0.594620
3   25.330296    25.330296     0.032375    0.032375
4   14.248291    14.248291     0.056809    0.056809
5    9.118907     9.118907     0.087295    0.087295
6 2533.029591  2533.029591     0.000329    0.000329
7 1424.829145  1424.829145     0.000585    0.000585
8  911.890653   911.890653     0.000914    0.000914
```

Fig. 13. Table displayed in the terminal by the Python library showing simulation results obtained by sweeping over transmission power levels and distances. Results include AAoI, SNR, and block error rate calculations.

4. Impact

Agenet provides a systematic and user-friendly tool for estimating AoI, contributing to the study of information freshness in wireless systems. Researchers without experience in Python should face no major difficulties using it. Furthermore, due to Agenet's modular structure, which implements specific steps of the AoI analysis process in separate modules, experienced users can insert their own data and code into intermediate steps of the simulation pipeline. This flexibility allows researchers to start their analyses from any point in the sequence without necessarily running the entire pipeline from scratch.

One of the key contributions of Agenet is its implementation of finite block length analysis based on block error rate calculation for short packet transmission. By incorporating finite block length information theory into AoI estimation, Agenet enables researchers to model and analyse performance under realistic conditions for mission-critical wireless communication systems, quantifying trade-offs between transmission reliability and information freshness. Consequently,

more efficient protocol designs and resource allocation strategies can be developed where radio resources are limited, prioritising user requirements, whether transmission reliability or information freshness, potentially improving time-critical application performance.

While various software toolboxes are available for analysing finite block length performance, including *Spectre: Short Packet Communication Toolbox* [12], to the best of the authors' knowledge, Agenet represents the first software package to integrate finite block length theory into AoI estimation.

By offering a standardised tool for AoI analysis, Agenet can enhance collaboration among researchers in the field of wireless communications. In particular, it has the potential to stimulate novel research questions concerning information freshness and transmission reliability across various wireless network scenarios. This, in turn, enables new investigations and developments in this rapidly evolving field, providing valuable insights into areas such as contemporary IoT applications and mission-critical wireless systems. Practical applications of the package include the design and optimisation of real-world wireless communication systems where the freshness of information and timely data delivery are critical. Examples include, but are not limited to, autonomous vehicles, industrial automation, smart cities, health monitoring applications, traffic management systems, and environmental monitoring. Furthermore, the package may prove valuable for disaster management and can be incorporated into financial trading platforms.

Finally, the package can also serve as a practical teaching resource in university courses, enabling students and researchers to engage directly with AoI metrics and computations.

5. Conclusions

Agenet is an open-source software package designed for systematic estimation and analysis of AoI in cooperative wireless networks. It implements a system model that combines finite block length information theory-based block error analysis with AoI analysis over Rayleigh fading channels, providing users and researchers with an important tool for studying short packet-based communication networks. Its modular architecture allows for flexible estimation of AoI under different types of network configurations and channel conditions, making it suitable for both academic research and practical applications in mission-critical wireless systems. Additionally, Agenet provides a significant contribution to the simulation-based study of trade-offs between transmission reliability and freshness of the information. This software package has the potential to accelerate research and development in wireless communication systems where timely delivery of information is crucial.

CRediT authorship contribution statement

Chathuranga M. Wijerathna Basnayaka: Writing – review & editing, Writing – original draft, Software, Methodology, Conceptualization. **Nuno Fachada:** Writing – review & editing, Writing – original draft, Supervision, Software, Methodology, Funding acquisition, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

This research was partially funded by the Fundação para a Ciência e a Tecnologia (FCT, https://ror.org/00snfqn58) under Grants UIDB/04111/2020, UIDB/00066/2020, and CEECINST/00002/2021/CP2788/CT0001, as well as by the Instituto Lusófono de Investigação e Desenvolvimento (ILIND), Portugal under Projects COFAC/ILIND/COPELABS/1/2022 and COFAC/ILIND/COPELABS/1/2024.

References

[1] Kosta A, Pappas N, Angelakis V. Age of information: A new concept, metric, and tool. Found Trends Netw 2017;12(3):162–259.
[2] Kaul S, Gruteser M, Rai V, Kenney J. Minimizing age of information in vehicular networks. In: Proc. IEEE int. conf. sensor, mesh ad hoc commun. netw.. 2011, p. 350–8. http://dx.doi.org/10.1109/SAHCN.2011.5984917.
[3] Basnayaka CMW, Jayakody DNK, Chang Z. Age of information based URLLC-enabled UAV wireless communications system. IEEE Internet Things J 2021.
[4] Yates RD, Sun Y, Brown DR, Kaul SK, Modiano E, Ulukus S. Age of information: An introduction and survey. IEEE J Sel Areas Commun 2021;39(5):1183–210. http://dx.doi.org/10.1109/JSAC.2021.3065072.
[5] Moradian M, Dadlani A. Age of information in scheduled wireless relay networks. In: 2020 IEEE wireless communications and networking conference. WCNC, 2020, p. 1–6. http://dx.doi.org/10.1109/WCNC45663.2020.9120608.
[6] Song J, Gündüz D, Choi W. Optimal scheduling policy for minimizing age of information with a relay. IEEE Internet Things J 2024;11(4):5623–37. http://dx.doi.org/10.1109/JIOT.2023.3308113.
[7] Pan H, Chan T-T, Leung VCM, Li J. Age of information in physical-layer network coding enabled two-way relay networks. IEEE Trans Mob Comput 2023;22(8):4485–99. http://dx.doi.org/10.1109/TMC.2022.3166155.
[8] Gu Y, Chen H, Li Y, Vucetic B. Ultra-reliable short-packet communications: Half-duplex or full-duplex relaying? IEEE Wirel Commun Lett 2018;7(3):348–51. http://dx.doi.org/10.1109/LWC.2017.2777857.
[9] Polyanskiy Y, Poor HV, Verdu S. Channel coding rate in the finite blocklength regime. IEEE Trans Inform Theory 2010;56(5):2307–59. http://dx.doi.org/10.1109/TIT.2010.2043769.
[10] Makki B, Svensson T, Zorzi M. Finite block-length analysis of the incremental redundancy HARQ. IEEE Wirel Commun Le. 2014;3(5):529–32.
[11] Gu Y, Chen H, Li Y, Vucetic B. Ultra-reliable short-packet communications: Half-duplex or full-duplex relaying? IEEE Wirel Commun 2017;7(3):348–51.
[12] Polyanskiy Y. Spectre: Short packet communication toolbox. 2024, https://github.com/yp-mit/spectre. [Accessed 18 December 2024].