TU Delft

Delft University of Technology

Document Version
Final published version

Licence
Dutch Copyright Act (Article 25fa)

Citation (APA)
Marchese, L., Davies, B., & Wehner, S. (2026). A Time/Utility Function for Scheduling Quantum Applications. In *Proceedings - 2026 International Conference on Quantum Communications, Networking, and Computing, QCNC 2026* (pp. 271-277). IEEE. https://doi.org/10.1109/QCNC69040.2026.00046

Important note
To cite this publication, please use the final published version (if applicable).
Please check the document version above.

Copyright
In case the licence states “Dutch Copyright Act (Article 25fa)”, this publication was made available Green Open Access via the TU Delft Institutional Repository pursuant to Dutch Copyright Act (Article 25fa, the Taverne amendment). This provision does not affect copyright ownership.
Unless copyright is transferred by contract or statute, it remains with the copyright holder.

Sharing and reuse
Other than for strictly personal use, it is not permitted to download, forward or distribute the text or part of it, without the consent of the author(s) and/or copyright holder(s), unless the work is under an open content license such as Creative Commons.

Takedown policy
Please contact us and provide details if you believe this document breaches copyrights.
We will remove access to the work immediately and investigate your claim.

This work is downloaded from Delft University of Technology.

2026 International Conference on Quantum Communications, Networking, and Computing (QCNC) | 979-8-3315-6110-9/26/$31.00 ©2026 IEEE | DOI: 10.1109/QCNC69040.2026.00046

A time/utility function for scheduling quantum applications

Luca Marchese*, Bethany Davies† and Stephanie Wehner‡
QuTech; Kavli Institute of Nanoscience; Quantum Computer Science, Department of Software Technologies, EEMCS
Delft University of Technology
Delft, Netherlands
*l.marchese@tudelft.nl, †b.j.davies@tudelft.nl, ‡s.d.c.wehner@tudelft.nl

***Abstract*—We introduce the concept of quantum time/utility functions (TUF) to enable effective scheduling decisions when executing quantum applications. The quantum TUF captures the relationship between the projected execution quality of an application and its execution time, enabling real-time, noise-aware scheduling. We consider different strategies for computing the TUF while addressing practical constraints, and introduce the Haar TUF, a simple, application-agnostic formula for random circuits. We implement the Haar TUF for a delegated quantum computing application, where we showcase both its ability to capture meaningful behaviour and its limitations in describing application-specific instances.**

***Index Terms*—quantum networks, decoherence, node scheduling**

Introduction

Quantum network applications, such as blind quantum computing (BQC) [1] and quantum key distribution [2], use unique quantum properties, like entanglement, to achieve capabilities that are classically impossible. A quantum network (QN) capable of supporting the concurrent execution of multiple quantum applications between any node of the network is currently being developed [3], [4]. When executing multi-node applications in this system, a node needs to frequently stop while waiting for responses from other nodes before resuming execution. The outcome is idling periods in which resources are wasted. One significant avenue to address this issue and improve the utilisation of available resources is to allow nodes to *multitask* [5], [6]: the capability of executing multiple applications concurrently on the same node.

Recently, QNodeOS, an operating system for quantum networks with multitasking capabilities [5], was developed. In the QNodeOS system, applications are divided into node-specific *programs*. At runtime each program is represented by a set of *tasks*, which are executed according to a *node scheduler* that enables multitasking (Fig. 1). Simple scheduling strategies such as first-in, first-out and earliest deadline first have been tested [7]. In simulations, it was found that multitasking enabled a lower makespan [8] (the total execution time for all programs), at the cost of execution quality. As can be seen in Fig. 2, multitasking can lead to longer waiting times on the execution of individual programs. Due to decoherence, the quality of stored quantum information degrades over time, leading to output errors.

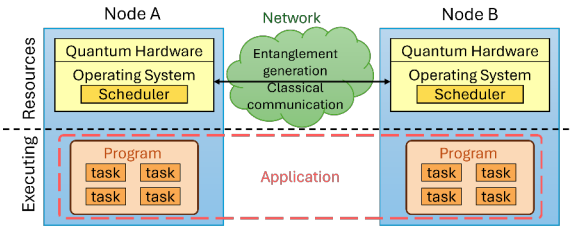


Fig. 1. **System overview** [5]. A quantum application is run on multiple nodes of a quantum network. The network provides entanglement generation and classical communication between the nodes. Each node is equipped with quantum hardware and an operating system that contains a scheduler. The scheduler is used to control the execution of programs, which are node-specific parts of a quantum application

In order to develop better scheduling strategies it is instrumental to create a tool capable of characterising the interplay between timing and expected execution quality of quantum applications. A scheduler equipped with this tool would have the opportunity to optimise the trade-off between makespan and execution quality. In this work we address this challenge. Inspired by time/utility functions (TUFs) used in classical scheduling (see Related Work below), we introduce the quantum TUF: a tool for characterising the effects of noise on the execution of quantum applications and make the necessary first step towards noise-aware scheduling. With this objective we provide:

- A **quantum TUF definition** aimed at supporting scheduling decisions in QN and quantum computing applications. Our definition makes explicit how one can capture the complex dependence of execution quality on noise in a format usable by a scheduler.
- Four **design criteria** for quantum TUFs. These criteria formalise the practical considerations that must be met for effective use of the TUF for quantum application scheduling.
- The **Haar TUF**, a candidate TUF for generic circuits that predicts execution quality with a simplified model of the application.
- An **example** of how the Haar TUF can be implemented with a delegated quantum computing application, while highlighting challenges and limitations.

979-8-3315-6110-9/26/$31.00 ©2026 IEEE
DOI 10.1109/QCNC69040.2026.00046

Authorized licensed use limited to: TU Delft Library. Downloaded on August 17,2026 at 12:35:37 UTC from IEEE Xplore. Restrictions apply.

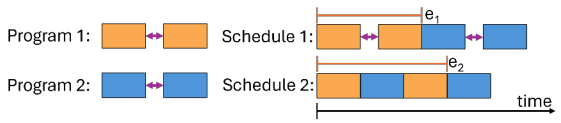


Fig. 2. **Quantum application scheduling.** Two possible scheduling solutions for two input programs that have a mandatory idling period (in purple) during their execution. In schedule 2 the makespan is smaller, but Programs 1 and 2 take longer to fully execute. The extra time required for the execution is a delay during which the state deteriorates due to decoherence

A. Related work

Given the limited lifetimes of quantum memories, it is natural to require that tasks be completed within a certain time window. This directly connects the scheduling of quantum applications to real-time scheduling [8], in which the requirement to complete execution by a certain time is specified by a *deadline*. Under strict deadlines, the utility of completing a task is assumed to be zero once the deadline has passed. A more general framework has been developed through TUFs [9], [10], which replace strict deadlines with functions that express the value of completing a task as a function of its completion time. Quantum real-time systems fit naturally within this framework, since the utility of completing tasks decays over time due to decoherence. TUFs have been proven effective in a variety of real-time scheduling settings [11]–[13]. However, they have not yet been applied to the scheduling of quantum applications.

A key step in defining a TUF for quantum applications is characterising its dependence on the system parameters that determine execution quality. The effects of noise on quantum algorithms are well studied: when parameters are accurately modelled, simulations [14] can predict performance, while analytical error-propagation methods [15], [16] provide bounds on execution accuracy. In this work, we focus on how such estimators can be applied to scheduling in quantum networks, where the main noise sources arise from long delays and entanglement generation, and where applications execute concurrently across multiple nodes.

This work is connected to QNodeOS [5], where multitasking on a nitrogen-vacancy (NV) centre quantum processor was demonstrated, and on Qoala [7], a QNodeOS extension in which simple scheduling strategies were tested. In the latter, the necessity of a tool to correctly balance the trade-off between makespan and execution quality was identified. This work provides a necessary step towards addressing these concerns. At the same time, our TUF framework is derived from general principles and is intended to remain applicable across a range of quantum architectures, requiring only minor architecture-specific adjustments.

I. Models

In order to provide noise information to the scheduler, we consider two models: a task model necessary for interfacing with the scheduler, and a noisy circuit layer model for describing quantum information quantities.

A. Task model

To represent a QN application at runtime we use an abstracted version of the framework described in [7], [17]. QN applications are divided into node-specific programs. At runtime, each program is instantiated (possibly multiple times). Each *program instance* is composed of a set of tasks $\{T_i\}$, which are units of execution. *Tasks* must be executed in a specific order, giving rise to dependency relations that can be described by a graph. Each task can be of four different types. *Classical local* tasks are for flow control and calculating parameters for quantum instructions. *Classical communication* tasks are for classical message exchanges. *Quantum local* tasks are for qubit instructions such as gates, measurements and initialisations. *Quantum communication* tasks are for generating entanglement with other nodes.

The tasks belonging to different applications form the input to the node scheduler. The job of the node scheduler is to dynamically decide which task to run, according to dependency constraints, resource availability, and performance optimisation.

B. Noisy circuit layer model

Quantum applications can be described with a *quantum circuit model* [18], which specifies the sequence of operations applied to qubits, including state initialisations, gates, and measurements. In the case of QN applications, the quantum circuit model describes the full application across multiple nodes. The quantum circuit formalism can be extended to describe noise [19], which includes both static errors and time-dependent decoherence. Quantum circuits can be formatted in

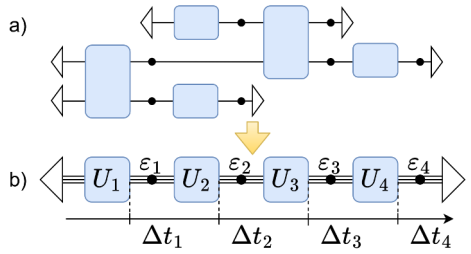


Fig. 3. a) A noisy quantum circuit. Gates are represented by boxes, noise by black dots. Initialisation and measurement are shown as triangles pointing left and right respectively. b) The corresponding noisy circuit layer model (NCLM), allowing each layer to be easily mapped to its corresponding delay.

a layered structure: alternating unitary operations and quantum channels describing noise. Any channel $\mathcal{E}$ can be written in the process matrix formalism [20] as

$$\mathcal{E}(\rho) = \sum_{\vec{\alpha}\vec{\beta}} \chi_{\vec{\alpha}\vec{\beta}} \sigma_{\vec{\alpha}} \rho \sigma_{\vec{\beta}}, \tag{1}$$

where $\sigma_{\vec{\alpha}} = \bigotimes_{n=1}^{N} \sigma_{\alpha_n}$ is the generalised Pauli basis, with σ_{α_n} the Pauli 4-vector $\{\mathbb{1}, \sigma_x, \sigma_y, \sigma_z\}$ for $\alpha_n = 0, 1, 2, 3$.

In the layer model, initialisation and measurements can be moved to the start or end of the circuit respectively, while unitaries and noise can be extended to act on N qubits via a

Authorized licensed use limited to: TU Delft Library. Downloaded on August 17,2026 at 12:35:37 UTC from IEEE Xplore. Restrictions apply.

tensor product with the identity channel (Fig. 3). We call this format the *noisy circuit layer model* (NCLM), whose size is given by the number of qubits N and the depth L (number of layers).

The NCLM structure is used in the literature for analysing how noise affects quantum computations [14]. We use it to describe when each operation occurs, by assigning a delay Δt_l between layers l and $l+1$. These delays can be used to compute decoherence effects described by time-dependent noise channels.

II. A TUF FOR QUANTUM APPLICATIONS

For quantum applications, a TUF should account for noise effects, which can be modelled using the NCLM. At the same time, TUFs must be assigned to individual tasks so they can inform scheduling decisions. This requires integrating the two models, which can be accomplished by mapping each task in the task model to a corresponding layer in the NCLM. A defining characteristic of TUFs is the explicit dependence on the completion time of a task. With our conventions, the completion time of the tasks assigned to layer l in the corresponding NCLM can be characterised by the delay Δt_{l-1} between layers $l-1$ and l.

Another challenge for a TUF aimed at informing execution quality is that the utility cannot be determined uniquely by the delay Δt_{l-1}, since many parameters can influence performance. Therefore, we introduce a set P of parameters necessary for the calculation. These parameters can include for example memory lifetimes, or timing of already executed tasks. The reason to distinguish between delay Δt_{l-1} and parameters P lies in how the scheduler uses them: the value of the TUF must be quickly obtainable for different values of Δt_{l-1} to inform scheduling decisions. Within this context we have the following definition.

Definition 1 (TUF for quantum application): Given a task T_i, assigned to layer l in a NCLM, the TUF is a function u_i assigned to task T_i that takes value

$$u_i(\Delta t_{l-1}; P), \tag{2}$$

where Δt_{l-1} is the delay between the layer $l-1$ and layer l, while P is a set of contextually relevant parameters required for its calculation. The output of the function represents the utility of completing the task T_i at time Δt_{l-1}.

Assigning tasks to the same layer is useful for entanglement generation, in which the corresponding task on each node must be synchronised. A note must be made about classical tasks, which cannot be easily assigned to a layer of the NCLM. Adding sublayers for classical tasks solves this issue, as we will show in Section V. However, to introduce the following concepts it is convenient to consider only delays between layers, as these affect quality of execution.

III. COMPUTING TUFS

Defining an instance of the TUF encounters multiple challenges, each of which can be addressed in different ways, leading to TUFs with different properties. Before introducing a candidate TUF, it is useful to establish a set of *design criteria*. These criteria highlight the properties of the different formulas and shed light on the practical challenges of defining a suitable TUF for the scheduler. The design criteria are as follows:

1) **Computability.** A single node must be able to evaluate the TUF using only the information that is locally available to it when the calculation is required. Although some parameters of remote nodes can be shared in advance [21], delay values are only available at runtime. Some nodes may also need to hide their operations for security reasons. For example, in BQC the server must be kept blind to the state prepared by the client.
2) **Interoperability.** Since nodes may be required to execute a wide range of applications, the TUF must be easily adaptable to various tasks and hardware configurations.
3) **Accuracy.** The previous criteria set limits on the design of the TUF, in terms of available inputs. Therefore, the TUF output will not necessarily be able to estimate exactly the true execution quality. Another important criterion for the TUF design is therefore the accuracy with which it approximates the true execution quality of the system.

Another optional criterion is **efficiency**: the capability to quickly retrieve TUF values to avoid overhead in scheduling decisions. Efficiency can be achieved by either precomputing and tabulating all relevant values of the TUF, or by designing a TUF whose computationally expensive components can be preprocessed in advance, leaving only lightweight operations for runtime.

A. Fidelity and probability of success TUF

An example of a TUF can be built from the fidelity. The fidelity for a quantum computation is defined as [18]

$$F(\mathcal{C}, \Delta t_{1:L}, |\psi\rangle) = \langle \psi_o | \rho_N | \psi_o \rangle, \tag{3}$$

where ρ_N is the density matrix [18] of the real noisy output state, $|\psi\rangle$ is the input state, $|\psi_o\rangle$ is the ideal output state, $\Delta t_{1:L}$ are the delays between layers, and $\mathcal{C}$ is the circuit in the NCLM representation of the application, which specifies all layer noises $\mathcal{E}_{1:L}$ and unitary operations $U_{1:L}$.[1]

Computing the fidelity directly is problematic because it depends on all delays. When the TUF is used by the scheduler, some layers are yet to be executed, and their delay is not known. In practice, while precise knowledge of future delays is not known, approximate estimates are in most cases available. For example, from network analysis [22], or from previous runs in the case of repeated executions. To represent this knowledge we use *delay distributions* $\mu_l(\Delta t_l)$. With these we can remove explicit dependence on future delays. For any function f, we have

$$\langle f(\Delta t_{1:L}) \rangle_{\mu_l} = \int f(\Delta t_{1:L}) \mu_l(\Delta t_l) d\Delta t_l, \tag{4}$$

[1]Using the notation $x_{a:b}$ to denote the set $\{x_l \text{ for } l = a, a+1, ..., b\}$

Authorized licensed use limited to: TU Delft Library. Downloaded on August 17,2026 at 12:35:37 UTC from IEEE Xplore. Restrictions apply.

where $\langle\cdot\rangle$ denotes temporal average.

With this tool we can define the fidelity TUF.

Definition 2 (Fidelity TUF): Given a task T_i belonging to layer l in a QN application, its fidelity TUF is defined as:

$$u_i^{(F)}(\Delta t_{l-1}; P_F) = \langle F(\mathcal{C}, \Delta t_{1:L}, |\psi\rangle)\rangle_{\mu_{l:L}}, \tag{5}$$

where we use the fidelity F from (3), and the contextual parameters P_F (2) are the (available) delays $\Delta t_{1:l-2}$, the delay distributions $\mu_{l:L}$, the NCLM $\mathcal{C}$, and the input state $|\psi\rangle$.

One limitation of the fidelity TUF is that we compare the final state to its ideal counterpart before measurement. In some applications, a specific pattern of the measurement outcome can be interpreted as a failure or success result. A metric such as success probability can define a *success probability TUF* that takes into account how the final measurement is interpreted. While this TUF could provide more information, it is difficult to define an explicit expression, as it varies for different applications.

Both the fidelity TUF and the probability of success TUF require full circuit information to be computed, so they can't be used in security-sensitive applications. Moreover, they require full tabulation for all combinations of delays to be used, since the dependency on the delays can be complex, limiting scalability. In the next section, we define the Haar TUF, to address these shortcomings.

IV. The Haar TUF

The Haar TUF aims to capture meaningful noise effects while remaining computationally efficient and relying on fewer contextual parameters (2). It is derived from an NCLM in which each layer unitary is replaced with a random unitary from the Haar measure [23], while leaving the noise channel unchanged. For this modified model we calculate expectation values of the fidelity. The use of the Haar measure is well established across various domains within quantum information [19], [24]–[27]. In this work we never sample directly from the Haar measure, as sampling from the Haar measure is inefficient [28], but use analytical results [29] to compute relevant expectation values for meaningful quantities, such as the fidelity. The Haar TUF, obtained from these analytical expressions, is simple to calculate, at the cost of being imprecise for specific circuit instances due to the loss of detailed state–noise interactions.

The Haar TUF is based on a quantity we call the *Haar expected fidelity*. For any application described by an NCLM and a pure state $\rho = |\psi\rangle\langle\psi|$ as input, the fidelity of the circuit can be written as

$$F = \langle\psi|\mathcal{I}^{-1}(\mathcal{N}(\rho))|\psi\rangle, \tag{6}$$

where $\mathcal{N}(\rho)$ is the quantum channel which describes the NCLM, while $\mathcal{I}(\rho)$ is the quantum channel of the circuit in an ideal, noiseless case. The two channels can be expanded as

$$\mathcal{N}(\rho) = \mathcal{E}_L \circ \mathcal{U}_L \circ ... \circ \mathcal{E}_1 \circ \mathcal{U}_1(\rho), \tag{7}$$

$$\mathcal{I}(\rho) = \mathcal{U}_L \circ ... \circ \mathcal{U}_1(\rho), \tag{8}$$

where $\mathcal{U}_l(\rho) = U_l \rho U_l^\dagger$, while U_l and $\mathcal{E}_l$ are the unitaries and noise channels (Section I-B).

We define F_H as the fidelity calculated with random unitaries instead of application-specific unitaries. The random variable F_H depends on known noise channels $\mathcal{E}_l$ and random unitaries U_l. We define the *Haar expected fidelity* as the expected value of the random variable F_H, obtained by independently averaging all N-qubit unitaries over the Haar measure μ_H:

$$\mathbb{E}[F_H] = \mathbb{E}_{U_1\sim\mu_H}[....\mathbb{E}_{U_L\sim\mu_H}[F_H]...]. \tag{9}$$

The Haar expected fidelity $\mathbb{E}[F_H]$ depends only on the noise channels $\mathcal{E}_l$, and can be calculated exactly.

Theorem 1 (Haar expected fidelity): Consider a NCLM with N qubits and L layers, each composed of a unitary U_l randomly sampled from the Haar measure and a noise channel $\mathcal{E}_l$. The Haar expected fidelity (9) of the circuit is given by

$$\mathbb{E}[F_H] = \overline{w}\left(1 - \frac{1}{d}\right) + \frac{1}{d}, \tag{10}$$

$$\text{where } \overline{w} = \prod_{l=1}^{L} w_l \quad \text{and} \quad w_l = \frac{d^2 F_e^{(l)} - 1}{d^2 - 1}, \tag{11}$$

where $F_e^{(l)} = \chi_{00}^{(l)}$ (1) is the entanglement fidelity of each layer [18], [30], while the dimension $d = 2^N$ is the size of the Hilbert space describing the system [18].

Proof: This result is a generalisation of Lemma 1 in [26] for heterogeneous layers. The proof is available in the full version on arXiv [31]. ∎

Time-dependent noise channels $\mathcal{E}_l(\Delta t_l)$ correspond to time-dependent entanglement fidelities $F_e^{(l)}(\Delta t_l)$. The *Haar expected fidelity* is therefore a quantity that depends on all delays Δt_l between layers. We can then use temporal averages (4) to calculate its corresponding TUF.

Definition 3 (Haar TUF): Given a task T_i belonging to layer l in a NCLM, its Haar TUF is defined as:

$$u_i^H(\Delta t_{l-1}; P_H) = \prod_{m=1}^{l-1} w_m(\Delta t_m) \prod_{m=l}^{L} \langle w_m\rangle_{\mu_m} \left(1 - \frac{1}{d}\right) + \frac{1}{d}, \tag{12}$$

where the contextual parameters P_H include the (available) delays $\Delta t_{1:l-2}$, the delay distributions $\mu_{l:L}$, and the parameters $w_m(\Delta t_m)$, which are defined in (11).

A. Properties of the Haar TUF

In this section we evaluate the Haar TUF (12) using the criteria defined in Section III: computability, interoperability, accuracy, and efficiency.

The Haar TUF is *computable*, since the contextual parameters are all available when the evaluation of the Haar TUF is required. The delays $\Delta t_{1:l-2}$ are between layers that have already been executed, and the dependency on future delays is removed through temporal averages (4). Additionally, the Haar TUF is *secure*, since it does not require information regarding which specific unitary is used in the application.

Authorized licensed use limited to: TU Delft Library. Downloaded on August 17,2026 at 12:35:37 UTC from IEEE Xplore. Restrictions apply.

Regarding *interoperability*, the Haar TUF can be used for many different applications and hardware configurations. In particular, the Haar TUF depends only on N and L for describing an application, since it does not need to know the specific unitaries executed. Additionally, the dependence of the Haar TUF on hardware parameters is isolated to a single parameter for each layer, $w_l(\Delta t_l)$ (11), together with the delays Δt_l, yielding a simplified interface with the hardware.

In terms of accuracy, the Haar TUF cannot provide precise application-specific values, since Haar averaging removes circuit-specific interactions. To gain insight into how far a specific circuit's fidelity may deviate from the Haar TUF, we examine the standard deviation (SD) of the random variable F_H (9), computed as follows.

Theorem 2: Consider an N-qubit NCLM with Haar random unitaries U_l and noise channels $\mathcal{E}_l$. The standard deviation of the fidelity F_H can be calculated for Pauli noise [18] as

$$\sigma_{F_H} = \overline{w}\frac{1}{d}\sqrt{\left(\sum_{l=1}^{L}\left(\frac{z_l}{w_l^2}-1\right)+\frac{\overline{z}}{\overline{w}^2}-1\right)}+O\left(\frac{1}{d^2}\right), \tag{13}$$

and for general noise as

$$\sigma_{F_H} = \overline{w}\sqrt{\frac{2}{d}\left(\sum_{l=1}^{L}\left(\frac{Re(y_l^{(a)})}{w_l^2}-1\right)+\frac{\overline{y^{(b)}}}{\overline{w}^2}-1\right)}+O\left(\frac{1}{d}\right), \tag{14}$$

where $\overline{x}=\prod_{l=1}^{L} x_l$ for any layer-specific variable, and

$$w_l = F_e^{(l)} + O(d^{-2}), \qquad z_l = \sum_{\vec{\alpha}} (\chi^{(l)}_{\vec{\alpha}\vec{\alpha}})^2, \tag{15}$$

$$y_l^{(a)} = \sum_{\vec{\alpha}} (\chi^{(l)}_{\vec{\alpha}\vec{0}})^2, \qquad y_l^{(b)} = \sum_{\vec{\alpha}} |\chi^{(l)}_{\vec{\alpha}\vec{0}}|^2, \tag{16}$$

where $\chi^{(l)}_{\vec{\alpha}\vec{\beta}}$ is the representation of the noise channel $\mathcal{E}_l$ in the process matrix formalism (1), and $O(d^{-n})$ are the order terms in a Taylor expansion for large d.

Proof: The proof is available in the full version on arXiv [31]. This result is a generalisation of the results in [32], which gives the single-layer case. ∎

This result can be used to calculate the right hand side in Chebyshev's concentration inequality, used to calculate the probability that the fidelity of a circuit chosen at random has a difference from the average fidelity greater than an error e:

$$\mathcal{P}(|F - \mathbb{E}[F_H]| > e) \leq \frac{\sigma^2_{F_H}}{e^2}. \tag{17}$$

Since $\sigma^2_{F_H}$ is $O(d^{-2})$ ($O(d^{-1})$) for Pauli (general) noise, as N increases the fidelity of any circuit drawn uniformly at random has an exponentially small probability of lying outside this region.

With respect to *efficiency*, the Haar TUF (12) is an example of a modular TUF, in which computationally expensive components can be precomputed, leaving only lightweight operations at runtime. In particular, the layer parameters $w_l(\Delta t_l)$ factor across layers and can therefore be evaluated independently. Moreover, the temporal averages $\langle w_l \rangle_{\mu_l}$, which may be expensive to obtain, can be precomputed and stored, ensuring that these costs do not affect scheduler performance. These properties make the Haar TUF lightweight at runtime and scalable for larger systems.

V. Example: delegated quantum computing

In this section, we demonstrate how the Haar TUF can be implemented for a delegated quantum computing (DQC) application. We show a realisation of the mapping between the NCLM and the task model, and highlight the discrepancy between the Haar TUF predictions and the actual fidelity of a specific application. DQC is a two-node application in which a client delegates a computation (rotation and measurement) to a server, while keeping the server blind to the computation. DQC is the fundamental building block of BQC [33].

We choose DQC for this implementation because it has been executed on an NV-centre platform [5] with an operating system equipped with scheduling capabilities. The experiment was used to provide the data used in this section.

A. NCLM and task model mapping

The mapping of the DQC application to the task model and the NCLM (Section I) is shown in Fig. 4. The quantum instructions can be grouped into three layers: one for entanglement generation, one for qubit preparation from the client, and one for the server correction. The first two layers are associated with delays Δt_1 and Δt_2, which can be used to parametrise how long the qubits are left in memory and affected by decoherence.

Additionally, two sublayers can be added between the second and third layers to be assigned to classical tasks: calculating the δ value on the client side, and receiving δ and preparing the required gate for correction on the server side. These two sublayers divide the interval Δt_2 into three subintervals: $\Delta t_{2,s}$ for $s = 1, 2, 3$.

In this simplified model we use three server tasks (S1 to S3) and three client tasks (C1 to C3) to run the application. In a real implementation, additional classical tasks may be present, which can be treated in the same fashion as the two classical tasks in our simplified model.

B. Haar TUF evaluation

The Haar TUF requires two types of inputs: the noise channels for each layer and the delay information.

Regarding the noise channels, we model the first layer as $\mathcal{D}_2 \circ \mathcal{P}_z^{\otimes 2}$ [18]: a two-qubit depolarizing channel, representing the imperfect entanglement generation, followed by two dephasing channels applied in parallel to the qubits. We model the second layer as a dephasing channel $\mathcal{P}_z$ on the only qubit in memory during the delay [5]. For the third layer we choose the trivial identity channel as the noise channel, assuming that the impact of gate and measurement errors is small compared to the other source of noise.

The delay distribution can be obtained from experimental data. Using the data provided in [5], we obtain the time interval

Authorized licensed use limited to: TU Delft Library. Downloaded on August 17,2026 at 12:35:37 UTC from IEEE Xplore. Restrictions apply.

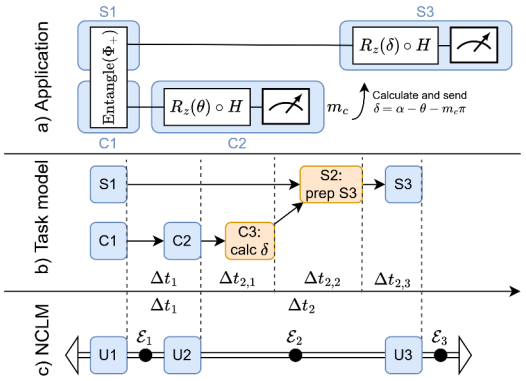


Fig. 4. a) The circuit for a DQC application. Client and server create a shared entanglement pair in state $|\Phi_+\rangle = (|00\rangle + |11\rangle)/\sqrt{2}$. The client performs gates and a measurement, resulting in outcome bit m_c. The client computes δ from m_c and DQC parameters $\alpha \in [0, 2\pi)$ (desired computation phase) and $\theta \in [0, 2\pi)$ (masking phase) and sends δ. Meanwhile, the server keeps its qubit in memory. On receiving δ, the server applies gates depending on δ, resulting in a single-qubit state, which depends only on α. b) Task model for the DQC application, divided by node: server on top and client on the bottom. In blue are the quantum tasks, which can be directly mapped to the layers of the NCLM and in orange are the classical tasks which can be assigned to additional sublayers. c) NCLM for the DQC application, with 2 qubits and 3 layers. Each layer has one or more quantum tasks assigned to it.

between the end of tasks S1, C2, C3, S2, and S3. From this data it is possible to evaluate the TUF. For example for task $C2$, belonging to layer $(2, 1)$, we have:

$$u^H_{C2}(\Delta t_1; \mu_2, w_{1:3}) = w_1(\Delta t_1)\langle w_2\rangle_{\mu_2}\left(1 - \frac{1}{d}\right) + \frac{1}{d}, \quad (18)$$

where the value of $\langle w_2\rangle_{\mu_2}$ is the sample mean over the measured delays.

C. Performance analysis

A simulation using NetSquid [34] is used as a baseline for comparison with the Haar TUF. The simulation is set up to run the application from Fig. 4, and to calculate the fidelity of the server qubit F_{m_c} conditioned on the measurement outcome m_c, and the probability p_{m_c} that each outcome can occur. The fidelity of the DQC application F_{DQC} is defined as:

$$F_{DQC} = \sum_{m_c} p_{m_c} F_{m_c}, \quad (19)$$

which can be evaluated for all combinations of delays $\{\Delta t_1, \Delta t_2\}$, and choice of application parameters α, θ. A sample mean over the delay distributions can be used to calculate the fidelity TUF $u^F(\Delta t_l)$ used for the comparison. The results are shown in Fig. 5. The comparison shows that the Haar TUF predictions differ from the simulation results. There are two effects that explain this behaviour.

(1) Single-qubit versus two-qubit fidelity. The F_{DQC} is computed only on the server qubit, whereas the Haar TUF estimates the fidelity of the full two-qubit channel involving both the client and the server. The resulting discrepancy is captured by

$$F_{\text{gap}} = F_{DQC} - F_{CS}, \quad (20)$$

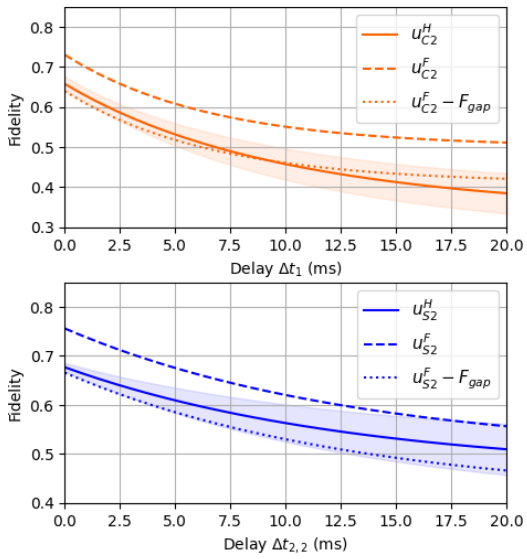


Fig. 5. **Comparison between Haar TUF and fidelity TUF.** The top (bottom) plot shows the results for task C2 (S2), according to the task model in Fig. 4b. In both plots solid lines represent the Haar TUF, the shaded area is ± 1 SD, and dashed lines represent the fidelity TUF. The main contribution to the gap between these two TUFs is F_{gap}, caused by calculating the fidelity with the server qubit, while the Haar TUF estimates the fidelity of both qubits. Subtracting F_{gap} from the fidelity TUF highlights other contributions to the mismatch, such as the state-dependent noise sensitivity.

where F_{CS} is the two-qubit fidelity obtained by simulating the same noise layers and unitaries but omitting the client measurement. In our simple model, $F_{\text{gap}} = p_d/4$ due to the specific structure of the circuit, where p_d is the parameter of the depolarizing noise in the first layer. This value is the same for all combinations of delays and application parameters.

(2) State-dependent noise sensitivity. Haar averaging removes information about how a noise channel interacts with specific quantum states. For instance, in layer 1 the dephasing noise $\mathcal{P}_z^{\otimes 2}$ frequently produces a $Z \otimes Z$ error for large Δt_1, but this error leaves the Bell state $|\Phi_+\rangle = (|00\rangle + |11\rangle)/\sqrt{2}$ invariant. As a result, the actual circuit fidelity is higher than the Haar-averaged value, an effect visible in task C2 at large Δt_1. Similarly, in the second layer the ideal state prepared on the server while waiting for the classical message is $|\pm_\theta\rangle = (|0\rangle \pm e^{-i\theta}|1\rangle)/\sqrt{2}$, which is maximally sensitive to dephasing. This yields a circuit-specific fidelity lower than the average case, as seen for task S2. If we remove effect (1) by shifting the simulation output by F_{gap}, we can see that the effects of state-dependent noise sensitivity are of the order of 1 SD, validating its use in learning about the difference between Haar-averaged values and specific circuits.

The Haar TUF captures overall application performance trends but lacks precision for individual circuits, as averaging over unitaries removes state-dependent effects. It also ignores classically controlled operations, assuming all qubits contribute equally to the fidelity. Refining the method could address these issues, but this is beyond our scope. Nevertheless, for coarse comparisons between tasks, its estimates may

Authorized licensed use limited to: TU Delft Library. Downloaded on August 17,2026 at 12:35:37 UTC from IEEE Xplore. Restrictions apply.

be sufficient to serve as a noise-aware scheduling tool.

VI. Conclusion

We have defined the first TUF for quantum applications, providing a framework for capturing the complex dependence of execution quality on noise in a format usable by the scheduler. This definition connects quantum noise modelling with classical scheduling frameworks. Our TUF definition provides the starting point for multiple ways to compute TUFs, which can trade off accuracy for faster computation.

In particular, we introduced the Haar TUF as an example of how one such formula can be defined, and showcased its qualities (lightweight, secure, simple) and drawbacks (accuracy). Through an implementation of the Haar TUF for a delegated quantum computing application, we demonstrated its use while highlighting practical limitations.

The main avenues for future work will be the development of new TUFs for accurately providing execution quality estimates, and the implementation of scheduling algorithms based on the TUF paradigm, to provide new opportunities for improving application execution on quantum systems.

Acknowledgment

The authors thank Bart van der Vecht and Mariagrazia Iuliano for helpful discussions. This project acknowledges financial support from NWO through a VICI grant.

References

[1] A. Broadbent *et al.*, "Measurement-Based and Universal Blind Quantum Computation," in *Formal Methods for Quantitative Aspects of Programming Languages*. Springer, 2010, pp. 43–86. [Online]. Available: https://doi.org/10.1007/978-3-642-13678-8_2

[2] P. W. Shor and J. Preskill, "Simple proof of security of the bb84 quantum key distribution protocol," *Physical review letters*, vol. 85, no. 2, p. 441, 2000.

[3] S. Wehner, D. Elkouss, and R. Hanson, "Quantum internet: A vision for the road ahead," *Science*, vol. 362, no. 6412, p. eaam9288, 2018.

[4] A. Dahlberg *et al.*, "A link layer protocol for quantum networks," in *Proceedings of the ACM Special Interest Group on Data Communication*, ser. SIGCOMM '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 159–173. [Online]. Available: https://doi.org/10.1145/3341302.3342070

[5] C. Delle Donne *et al.*, "An operating system for executing applications on quantum network nodes," *Nature*, vol. 639, no. 8054, pp. 321–328, 2025.

[6] J. P. Erickson and J. H. Anderson, *Soft Real-Time Scheduling*. Singapore: Springer Nature Singapore, 2022, pp. 233–267. [Online]. Available: https://doi.org/10.1007/978-981-287-251-7_4

[7] B. van der Vecht, A. T. Yücel, H. Jirovská, and S. Wehner, "Qoala: an application execution environment for quantum internet nodes," 2025. [Online]. Available: https://arxiv.org/abs/2502.17296

[8] Y.-C. Tian and D. C. Levy, *Handbook of real-time computing*. Springer Nature, 2022.

[9] E. D. Jensen, C. D. Locke, and H. Tokuda, "A time-driven scheduling model for real-time operating systems." in *Rtss*, vol. 85, 1985, pp. 112–122.

[10] C. D. Locke, "Best-effort decision-making for real-time scheduling," Ph.D. dissertation, Carnegie Mellon University, USA, 1986, aAI8702895.

[11] J. Wang and B. Ravindran, "Time-utility function-driven switched ethernet: Packet scheduling algorithm, implementation, and feasibility analysis," *IEEE Transactions on Parallel and Distributed Systems*, vol. 15, no. 2, pp. 119–133, 2004.

[12] A. AuYoung, L. Grit, J. Wiener, and J. Wilkes, "Service contracts and aggregate utility functions," in *2006 15th IEEE International Conference on High Performance Distributed Computing*. IEEE, 2006, pp. 119–131.

[13] T. Tidwell, R. Glaubius, C. D. Gill, and W. D. Smart, "Optimizing expected time utility in cyber-physical systems schedulers," in *2010 31st IEEE Real-Time Systems Symposium*. IEEE, 2010, pp. 193–201.

[14] T. Schuster, C. Yin, X. Gao, and N. Y. Yao, "A polynomial-time classical algorithm for noisy quantum circuits," *Physical Review X*, vol. 15, no. 4, p. 041018, 2025.

[15] Z. Yu and Y. Li, "Analysis of error propagation in quantum computers," *arXiv preprint arXiv:2209.01699*, 2022.

[16] A. Deshpande, P. Niroula, O. Shtanko, A. V. Gorshkov, B. Fefferman, and M. J. Gullans, "Tight bounds on the convergence of noisy random circuits to the uniform distribution," *PRX Quantum*, vol. 3, no. 4, p. 040329, 2022.

[17] H. Jirovská *et al.*, "Evaluating an rcpsp implementation of quantum program scheduling," 2023. [Online]. Available: https://resolver.tudelft.nl/uuid:6abe7dc4-0308-490e-b156-eed07426e129

[18] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information*. Cambridge University Press, 2000.

[19] A. Dalzell, N. Hunter-Jones, and F. Brandão, "Random quantum circuits transform local noise into global white noise," *Communications in Mathematical Physics*, vol. 405, 03 2024.

[20] C. J. Wood, J. D. Biamonte, and D. G. Cory, "Tensor networks and graphical calculus for open quantum systems," May 2015, arXiv:1111.6950 [quant-ph]. [Online]. Available: http://arxiv.org/abs/1111.6950

[21] T. R. Beauchamp, H. Jirovská, S. Gauthier, and S. Wehner, "A modular quantum network architecture for integrating network scheduling with local program execution," *IEEE Transactions on Quantum Engineering*, pp. 1–31, 2025.

[22] C. Bovy, H. Mertodimedjo, G. Hooghiemstra, H. Uijterwaal, and P. Van Mieghem, "Analysis of end-to-end delay measurements in internet," in *Proc. of the Passive and Active Measurement Workshop-PAM*, vol. 2002, 2002.

[23] A. Haar, "Der massbegriff in der theorie der kontinuierlichen gruppen," *Annals of Mathematics*, vol. 34, no. 1, pp. 147–169, 1933. [Online]. Available: http://www.jstor.org/stable/1968346

[24] R. O'Donnell and J. Wright, "Efficient quantum tomography," in *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing*, 2016, pp. 899–912.

[25] J. Helsen, I. Roth, E. Onorati, A. Werner, and J. Eisert, "General framework for randomized benchmarking," *PRX Quantum*, vol. 3, no. 2, Jun. 2022. [Online]. Available: http://dx.doi.org/10.1103/PRXQuantum.3.020357

[26] J. Emerson, R. Alicki, and K. Życzkowski, "Scalable noise estimation with random unitary operators," *Journal of Optics B: Quantum and Semiclassical Optics*, vol. 7, no. 10, p. S347–S352, Sep. 2005. [Online]. Available: http://dx.doi.org/10.1088/1464-4266/7/10/021

[27] M. Horodecki, P. Horodecki, and R. Horodecki, "General teleportation channel, singlet fraction, and quasidistillation," *Physical Review A*, vol. 60, no. 3, p. 1888, 1999.

[28] C. Dankert, R. Cleve, J. Emerson, and E. Livine, "Exact and approximate unitary 2-designs and their application to fidelity estimation," *Physical Review A—Atomic, Molecular, and Optical Physics*, vol. 80, no. 1, p. 012304, 2009.

[29] A. A. Mele, "Introduction to haar measure tools in quantum information: A beginner's tutorial," 2024.

[30] B. Schumacher, "Sending entanglement through noisy quantum channels," *Phys. Rev. A*, vol. 54, pp. 2614–2628, Oct 1996. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevA.54.2614

[31] L. Marchese, B. Davies, and S. Wehner, "A time/utility function for scheduling quantum applications (in preparation)," 2026.

[32] E. Magesan, R. Blume-Kohout, and J. Emerson, "Gate fidelity fluctuations and quantum process invariants," *Physical Review A—Atomic, Molecular, and Optical Physics*, vol. 84, no. 1, p. 012309, 2011.

[33] A. Broadbent, J. Fitzsimons, and E. Kashefi, "Universal blind quantum computation," in *2009 50th Annual IEEE Symposium on Foundations of Computer Science*, 2009, pp. 517–526.

[34] T. Coopmans *et al.*, "Netsquid, a network simulator for quantum information using discrete events," *Communications Physics*, vol. 4, no. 1, p. 164, 2021.

Authorized licensed use limited to: TU Delft Library. Downloaded on August 17,2026 at 12:35:37 UTC from IEEE Xplore. Restrictions apply.