![TU Delft logo]

Delft University of Technology

NetSquid, a NETwork Simulator for QUantum Information using Discrete events

Coopmans, Tim; Knegjens, Robert; Dahlberg, Axel; Maier, David; Nijsten, Loek; de Oliveira Filho, Julio; Papendrecht, Martijn; Rabbie, Julian; Rozpedek, F.D.; Skrzypczyk, Matthew

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

NetSquid, a NETwork Simulator for QUantum Information using Discrete events

Tim Coopmans [1,2,5], Robert Knegjens [1,5], Axel Dahlberg[1,2], David Maier [1,2], Loek Nijsten[1],
Julio de Oliveira Filho[1], Martijn Papendrecht[1,2], Julian Rabbie[1,2], Filip Rozpędek [1,2,3], Matthew Skrzypczyk [1,2],
Leon Wubben[1,2], Walter de Jong [4], Damian Podareanu[4], Ariana Torres-Knoop[4], David Elkouss[1,6 ✉] &
Stephanie Wehner[1,2,6 ✉]

In order to bring quantum networks into the real world, we would like to determine the requirements of quantum network protocols including the underlying quantum hardware. Because detailed architecture proposals are generally too complex for mathematical analysis, it is natural to employ numerical simulation. Here we introduce NetSquid, the NETwork Simulator for QUantum Information using Discrete events, a discrete-event based platform for simulating all aspects of quantum networks and modular quantum computing systems, ranging from the physical layer and its control plane up to the application level. We study several use cases to showcase NetSquid's power, including detailed physical layer simulations of repeater chains based on nitrogen vacancy centres in diamond as well as atomic ensembles. We also study the control plane of a quantum switch beyond its analytically known regime, and showcase NetSquid's ability to investigate large networks by simulating entanglement distribution over a chain of up to one thousand nodes.

[1] QuTech, Delft University of Technology and TNO, Delft, The Netherlands. [2] Kavli Institute of Nanoscience, Delft, The Netherlands. [3] Pritzker School of Molecular Engineering, University of Chicago, Chicago, IL, USA. [4] SURF, Amsterdam, The Netherlands. [5]These authors contributed equally: Tim Coopmans, Robert Knegjens. [6]These authors jointly supervised this work: David Elkouss, Stephanie Wehner. ✉email: d.elkousscoronas@tudelft.nl; s.d.c.wehner@tudelft.nl

Quantum communication can be used to connect distant quantum devices into a quantum network. At short distances, networking quantum devices provides a path towards a scalable distributed quantum computer[1]. At larger distances, interconnected quantum networks allow for communication tasks between distant users on a quantum internet. Both types of quantum networks have the potential for large societal impact. First, analogous to classical computers, it is likely that any approach for scaling up a quantum computer so that it can solve real world problems impractical to treat on a classical computer, will require the interconnection of different modules[2–4]. Furthermore, quantum communication networks enable a host of tasks that are impossible using classical communication[5].

For both types of networks, many challenges must be overcome before they can fulfil their promise. The exact extent of these challenges remains unknown, and precise requirements to guide the construction of large-scale quantum networks are missing. At the physical layer, many proposals exist for quantum repeaters that can carry qubits over long distances (see e.g.[6–8] for an overview). Using analytical methods[9–30] and ad-hoc simulations[31–38] rough estimates for the requirements of such hardware proposals have been obtained. Yet, while greatly valuable to set minimal requirements, these studies still provide limited detail. Even for a small-scale quantum network, the intricate interplay between many communicating devices, and the resulting timing dependencies makes a precise analysis challenging. To go beyond, we would like a tool that can incorporate not only a detailed physical modelling, but also account for the implications of time-dependent behaviour.

Quantum networks cannot be built from quantum hardware alone; in order to scale they need a tightly integrated classical control plane, i.e. protocols responsible for orchestrating quantum network devices to bring entanglement to users. Fundamental differences between quantum and classical information demand an entirely new network stack in order to create entanglement, and run useful applications on future quantum networks[39–44]. The design of such a control stack is furthermore made challenging by numerous technological limitations of quantum devices. A good example is given by the limited lifetimes of quantum memories, due to which delays in the exchange of classical control messages have a direct impact on the performance of the network. To succeed, we hence need to understand how possible classical control strategies do perform on specific quantum hardware. Finally, to guide overall development, we need to understand the requirements of quantum network applications themselves. Apart from quantum key distribution (QKD)[45–49] and a few select applications[50–53], little is known about the requirements of quantum applications[5] on imperfect hardware.

Analytical tools offer only a limited solution for our needs. Statistical tools (see e.g.[54–57]) have been used to make predictions about certain aspects of large regular networks using simplified models, but are of limited use for more detailed studies. Information theory[58] can be used to benchmark implementations against the ideal performance. However, it gives no information about how well a specific proposal will perform. As a consequence, numerical methods are of great use to go beyond what is feasible using an analytical study. Ad-hoc simulations of quantum networks have indeed been used to provide further insights on specific aspects of quantum networks (see e.g.[31–38,59–61]). However, while greatly informative, setting up ad-hoc simulations for each possible networking scenario to a level of detail that might allow the determination of more precise requirements is cumbersome, and does not straightforwardly lend itself to extensive explorations of new possibilities.

We would hence like a simulation platform that satisfies at least the following three features: First, accuracy: the tool should allow modelling the relevant physics. This includes the ability to model time-dependent noise and network behaviour. Second, modularity: it should allow stacking protocols and models together in order to construct complicated network simulations out of simple components. This includes the ability to investigate not only the physical layer hardware, but the entirety of the quantum network system including how different control protocols behave on a given hardware setup. Third, scalability: it should allow us to investigate large networks.

Evaluating the performance of large classical network systems, including their time-dependent behaviour is the essence of classical network analysis. Yet, even for classical networks, the predictive power of analytical methods is limited due to complex emergent behaviour arising from the interplay between many network devices. Consequently, a crucial tool in the design of such networks are network simulators, which form a tool to test new ideas, and many such simulators exist for purely classical networks[62–64]. However, such simulators do not allow the simulation of quantum behaviour.

In the quantum domain, many simulators are known for the simulation of quantum computers (see e.g.[65]). However, the task of simulating a quantum network differs greatly from simulating the execution of one monolithic quantum system. In the network, many devices are communicating with each other both quantumly and classically, leading to complex stochastic behaviour, and asynchronous and time-dependent events. From the perspective of building a simulation engine, there is also an important difference that allows for gains in the efficiency of the simulation. A simulator for a quantum computation is optimised to track large entangled states. In contrast, in a quantum network the state space grows and shrinks dynamically as qubits get measured or entangled, but for many protocols, at any moment in time the state space describing the quantum state of the network is small. We would thus like a simulator capable of exploiting this advantage.

In this paper we introduce the quantum network simulator NetSquid: the NETwork Simulator for QUantum Information using Discrete events. NetSquid is a software tool (available as a package for Python and previously made freely available online[66]) for accurately simulating quantum networking and modular computing systems that are subject to physical non-idealities. It achieves this by integrating several key technologies: a discrete-event simulation engine, a specialised quantum computing library, a modular framework for modelling quantum hardware devices, and an asynchronous programming framework for describing quantum protocols. We showcase the utility of this tool for a range of applications by studying several use cases: the analysis of a control plane protocol beyond its analytically accessible regime, predicting the performance of protocols on realistic near-term hardware, and benchmarking different quantum devices. These use cases, in combination with a scalability analysis, demonstrate that NetSquid achieves all three features set forth above. Furthermore, they show its potential as a general and versatile design tool for quantum networks, as well as for modular quantum computing architectures.

Results and discussion

NetSquid in a nutshell. Simulating a quantum network with NetSquid is generally performed in three steps. Firstly, the network is modelled using a modular framework of components and physical models. Next, protocols are assigned to network nodes to describe the intended behaviour. Finally, the simulation is executed for a typically large number of independent runs to collect statistics with which to determine the performance of the network. To explain these steps and the features involved further, we
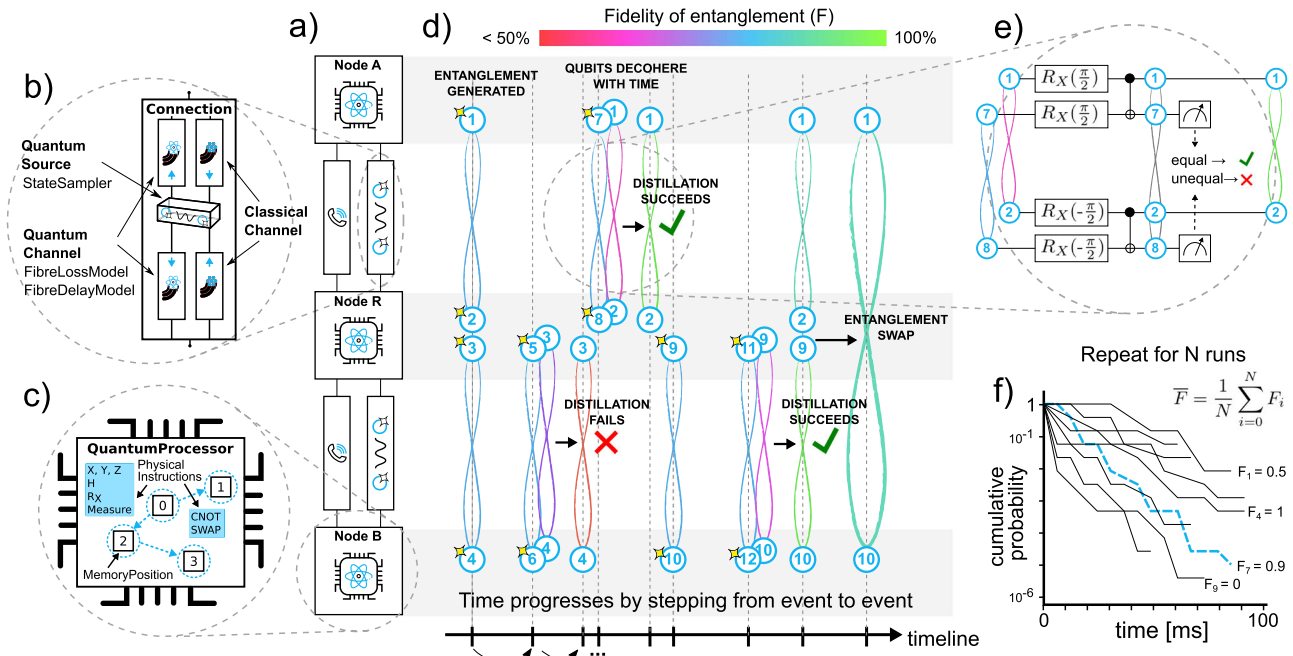
Fig. 1 Illustrative example of a NetSquid use case. Each sub-figure explains part of the modelling and simulation process. For greater clarity the figures are not based on real simulation data. The scenario shown is a quantum repeater utilising entanglement distillation (see main text). **a** The setup of a quantum network using node and connection components. **b** A zoom in showing the subcomponents of the entangling connection component. The quantum channels are characterised using fibre delay and loss models. The quantum source samples from an entangled bipartite state sampler when externally triggered by the classical channel. **c** A zoom in of the quantum memory positions within a quantum processor illustrating their physical gate topology. The physical single-qubit instructions possible on each memory in this example are the Pauli (X, Y, Z), Hadamard (H), and X-rotation (R_X) gates, and measurement. The blue-dashed arrows show the positions and control direction (where applicable) for which the two-qubit instructions controlled-X (CNOT) and swap are possible. Noise and error models for the memories and gates are also assigned. **d** Illustration of a single simulation run. Time progresses by discretely stepping from event to event, with new events generated as the simulation proceeds. Qubits are represented by circles, which are numbered according to the order they were generated. A star shows the moment of generation. The curved lines between qubits denote their entanglement with the colour indicating fidelity. The state of each qubit is updated as it is accessed during the simulation, for instance to apply time-dependent noise from waiting in memory. **e** A zoom in of the distillation protocol. The shared quantum states of the qubits are combined in an entangling step, which then shrinks as two of the qubits are measured. The output is randomly sampled, causing the simulation to choose one of two paths by announcing success or failure. **f** A plot illustrating the stochastic paths followed by multiple independent simulation runs over time, labelled by their final end-to-end fidelity F_i. The blue dashed line corresponds to the run shown in (**d**). The runs are typically executed in parallel. Their results are statistically analysed to produce performance metrics such as the average outcome fidelity and run duration.

consider a simple use case for illustration. For a more detailed presentation of the available functionality and design of the NetSquid framework see section 'Design and functionality of NetSquid' of the Methods.

The scenario we will consider is the analysis of an entanglement distribution protocol over a quantum repeater chain with three nodes. The goal of the analysis is to estimate the average output fidelity of the distributed entangled pairs. The entanglement distribution protocol is depicted in Fig. 1d, e. It works as follows. First, the intermediate node generates two entangled pairs with each of its adjacent neighbours. Entanglement generation is modelled as a stochastic process that succeeds with a certain probability at every attempt. When two pairs are ready at one of the links, the DEJMPS entanglement distillation scheme[67] is run to improve the quality of the entanglement. If it fails, the two links are discarded and the executing nodes restart entanglement generation. Once both distilled states are ready, the intermediate node swaps the entanglement to achieve end-to-end entanglement. We remark that already this simple protocol is rather involved to analyse.

We begin by modelling the network. The basic element of NetSquid's modular framework is the 'component'. It is capable of describing the physical model composition, quantum and classical communication ports, and, recursively, any

subcomponents. All hardware elements, including the network itself, are represented by components. For this example we require three remote nodes linked by two quantum and two classical connections, the setup of which is shown in Fig. 1a. In Fig. 1b, c the nested structure of these components is highlighted. A selection of physical models is used to describe the loss and delay of the fibre optic channels, the decoherence of the quantum memories, and the errors of quantum gates.

Quantum information in NetSquid is represented at the level of qubits, which are treated as objects that dynamically share their quantum states. These internally shared states will automatically merge or 'split'—a term we use to mean the separation of a tensor product state into two separately shared sub-states—as qubits entangle or are measured, as illustrated by the distillation protocol in Fig. 1e. The states are tracked internally, i.e. hidden from users, for two reasons: to encourage a node-centric approach to programming network protocols, and to allow a seamless switching between different quantum state representations. The representations offered by NetSquid are ket vectors, density matrices, stabiliser tableaus and graph states with local Cliffords, each with trade-offs in modelling versatility, computation speed and network (memory) scalability (see the subsection 'Fast and scalable quantum network simulation' below and Supplementary Note 1).
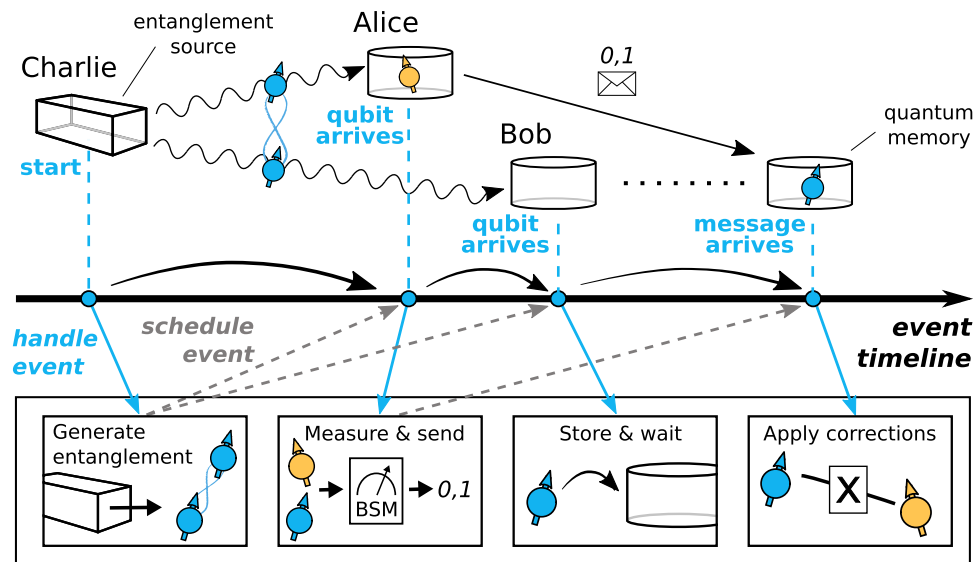
Fig. 2 Abstract example of simulating a quantum protocol with discrete events. When setting up the simulation, protocol actions are defined to occur when a specific event occurs, as in the setup of a real system. Instead of performing a continuous time evolution, the simulation advances to the next event, and then automatically executes the actions that should occur when the event takes place. Any action may again define future events to be triggered after a certain (stochastic) amount of time has elapsed. For concreteness a simplified quantum teleportation example is shown where a qubit, shown as an orange circle with arrow, is teleported between the quantum memories of Alice and Bob. Here, entanglement is produced using an abstract source sending two qubits, shown as blue circles with arrows, to Alice and Bob. Once the qubit has traversed the fibre and reaches Alice's lab, an event is triggered that invokes the simulation of Alice's Bell state measurement (BSM) apparatus. The simulation engine steps from event to events defined by the next action, which generally occur at irregular intervals. This approach allows time-dependent physical non-idealities, such as quantum decoherence, to be accurately tracked.

Discrete-event simulation, an established method for simulating classical network systems[68], is a modelling paradigm that progresses time by stepping through a sequence of events—see Fig. 2 for a visual explanation. This allows the simulation engine to efficiently handle the control processes and feedback loops characteristic of quantum networking systems, while tracking quantum state decoherence based on the elapsed time between events. A novel requirement for its application to quantum networks is the need to accurately evolve the state of the quantum information present in a network with time. This can be achieved by retroactively updating quantum states when the associated qubits are accessed during an event. While it is possible to efficiently track a density matrix, quantum operations requiring a singular outcome for classical decision making, for instance a quantum measurement, must be probabilistically sampled. A single simulation run thus consists of a sequence of random choices that forms one of many possible paths. In Fig. 1d we show such a run for the repeater protocol example, which demonstrates the power of the discrete-event approach for tracking qubit decoherence and handling feedback loops.

The performance metrics of a simulation are determined statistically from many runs. Due to the independence of each run, simulations can be massively parallelised and thereby efficiently executed on computing clusters. For the example at hand we choose as metrics the output fidelity and run duration. In Fig. 1f the sampled run from Fig. 1d, which resulted in perfect fidelity, is plotted in terms of its likelihood and duration together with several other samples, some less successful. By statistically averaging all of the sampled runs the output fidelity and duration can be estimated.

In the following sections, we will outline three use cases of NetSquid: first, a quantum switch, followed by simulations of quantum repeaters based on nitrogen-vacancy technology or atomic-ensemble memories. We will also benchmark NetSquid's scalability in both quantum state size and number of quantum network nodes. Although the use cases each provide relevant insights into the performance of the studied hardware and protocols, we emphasise that one can use NetSquid to simulate arbitrary network topologies.

Simulating a quantum network switch beyond its analytically known regime. As a first use case showcasing the power of NetSquid, we study the control plane of a recently introduced quantum switch beyond the regime for which analytical results have been obtained, including its performance under time-dependent memory noise.

The switch is a node which is directly connected to each of k users by an optical link. The communications task is distributing Bell pairs and n-partite Greenberger–Horne–Zeilinger (GHZ) states[69] between $n \leq k$ users. The switch achieves this by connecting Bell pairs which are generated at random intervals on each link (See Fig. 3).

Intuitively, the switch can be regarded as a generalisation of a simple repeater performing entanglement swapping with added logic to choose which parties to link. Even with a streamlined physical model, it is already rather challenging to analytically characterise the switch use case[56].

In the following, we recover via simulation a selection of the results from Vardoyan et al.[56], who studied the switch as the central node in a star network, and extend them in two directions. First, we increase the range of parameters for which we can estimate entanglement rates using the same model as used in the work of Vardoyan et al. Second, simulation enables us to investigate more sophisticated models than the exponentially distributed erasure process from their work, in particular we analyse the behaviour of a switch in the presence of memory dephasing noise.

The protocol for generating the target n-partite GHZ states is simple. Entanglement generation is attempted in parallel across all k links. If successful they result in bipartite Bell states that are stored in quantum memories. The switch waits until n Bell pairs have been generated until performing an n-partite GHZ
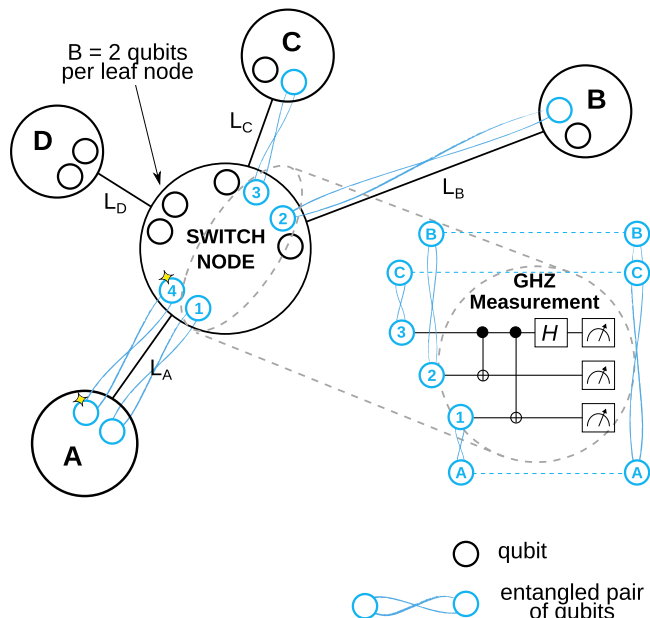
Fig. 3 A quantum switch in a star-shaped network topology as studied by Vardoyan et al.[56]. The switch (central node) is connected to a set of users (leaf nodes) via an optical fibre link that distributes perfect Bell pairs at random times, following an exponential distribution with mean rate $\mu \propto e^{-\beta L}$, where L denotes the distance of the link and β the attenuation coefficient. Associated with each link there is a buffer that can store B qubits at each side of the link. As soon as n Bell pairs with different leaves are available, the switch performs a measurement in the n-partite Greenberger–Horne–Zeilinger (GHZ) basis, which results in an n-partite GHZ state shared by the leaves. The GHZ-basis measurement consists of: first, controlled-X gates with the same qubit as control; next, a Hadamard (H) gate on the control qubit; finally, measurement of all qubits individually. The figure shows four leaf nodes, GHZ size $n = 3$ and a buffer size $B = 2$.

measurement, which converts the pairs into a state locally equivalent to a GHZ state. An additional constraint is that the switch has a finite buffer B of number of memories dedicated for each user (see Fig. 3). If the number of pairs stored in a link is B and a new pair is generated, the old one is dropped and the new one is stored.

The protocol can be translated to a Markov chain. The state space is represented by a k-length vector where each entry is associated with a link and its value denotes the number of stored links. The switch's mean capacity, i.e. the number of states produced per second, can be derived from the steady-state of the Markov chain[56].

Using NetSquid, it is straightforward to fully reproduce the previous model and study the behaviour of the network without constructing the Markov Chain (details can be found in Supplementary Note 3). In Fig. 4a, we use NetSquid to study the capacity of a switch network serving nine users. The figure shows the capacity (number of produced GHZ-states per second), which we investigate for three use cases. First we consider a switch network distributing bipartite entanglement. Second, we consider also a switch-network serving bipartite entanglement but with link generation rates that do not satisfy the stability condition for the Markov Chain if the buffer B is infinitely large, i.e. a regime so far intractable. Third, we consider a switch-network distributing four-partite entanglement where the link generation rates μ differ per user, a regime not studied so far, and compute the capacity.

Beyond rate, it is important to understand the quality of the states produced. Answering this question with Markov chain models seems challenging. In order to analyse entanglement

quality, we introduce a more sophisticated decoherence model where the memories suffer from decay over time. In particular, we model decoherence as exponential T_2 noise, which impacts the quality of the state, as expressed in its fidelity with the ideal state. In Fig. 4b, we show the effect of the time-dependent memory noise on the average fidelity.

Sensitivity analysis for the physical modelling of a long range repeater chain. The next use case is the distribution of long-distance entanglement via a chain of quantum repeater nodes[6,9] based on nitrogen-vacancy (NV) centres in diamond[70,71]. This example consists of a more detailed physical model and more complicated control plane logic than the quantum switch or the distillation example presented at the start of this section. It is also an example of how NetSquid's modularity supports setting up simulations involving many nodes; in this case the node model and the protocol (which runs locally at a node) only need to be specified once, and can then be assigned to each node in the chain. Furthermore, the use of a discrete-event engine allows the actions of the individual protocols to be simulated asynchronously, in contrast to the typically sequential execution of quantum computing simulators.

The NV-based quantum processor includes the following three features. First, the nodes have a single communication qubit, i.e. a qubit acting as the optical interface that can be entangled with a remote qubit via photon interference. This seemingly small restriction has important consequences for the communications protocol. In particular, entanglement can not proceed in parallel with both adjacent nodes. As a consequence, operations need to be scheduled in sequence and the state of the communication qubit transferred onto a storage qubit. Second, the qubits in a node are connected with a star topology with the communication qubit located in the centre. Two-qubit gates are only possible between the communication qubit and a storage qubit. Third, communication and storage qubits have unequal coherence times. Furthermore, the storage qubits suffer additional decoherence when the node attempts to generate entanglement. Previous repeater-chain analyses, e.g.[22,23,43], did not take all three into account simultaneously.

Together with the node model, we consider two protocols: SWAP-ASAP and NESTED-WITH-DISTILL. In SWAP-ASAP, as soon as adjacent links are generated the entanglement is swapped. NESTED-WITH-DISTILL is a nested protocol[9] with entanglement distillation at every nesting level. For a description of the simulation, including the node model and protocols, see Methods, section 'Implementing a processing-node repeater chain in NetSquid'.

The first question that we investigate is the distance that can be covered by a repeater chain. For this we choose two sets of hardware parameters that we dub near-term and 10× improved (see Supplementary Note 4) and choose two configurations: one without intermediate repeaters and one with three of them. We observe, see Fig. 5a, that the repeater chain performs worse in fidelity than the repeaterless configuration with near-term hardware. For improved hardware, we see two regimes, for short distances the use of repeaters increases rate but lowers fidelity while from 750 km until 1500 km the repeater chain outperforms the no-repeater setup.

The second question that we address is which protocol performs best for a given distance. We consider seven protocols: no repeater, and repeater chains implementing SWAP-ASAP or NESTED-WITH-DISTILL over 1, 3 or 7 repeaters. The latter is motivated by the fact that the NESTED-WITH-DISTILL protocol is defined for $2^n - 1$ repeaters ($n \geq 1$), and thus 1, 3 and 7 are the first three possible configurations. In Fig. 5b, we sweep over the hardware parameter space for two distances, where we improve
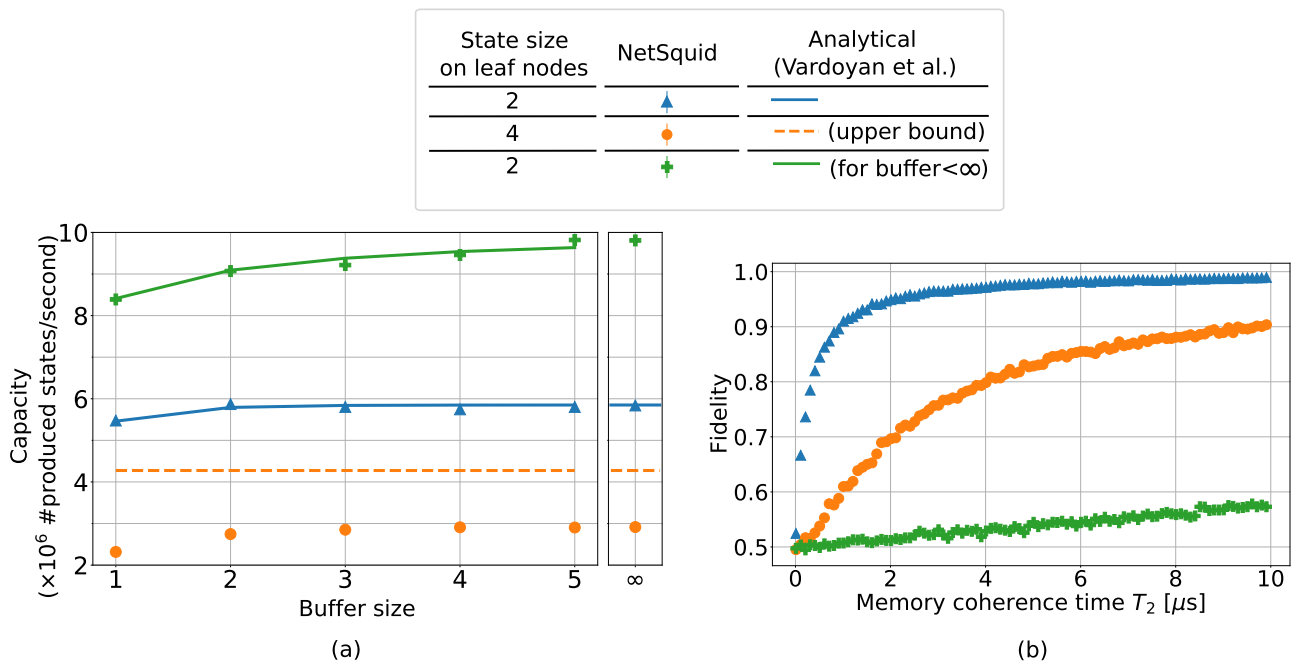
Fig. 4 Performance analysis of the quantum switch with nine users using NetSquid. a Capacity as a function of the buffer size (number of quantum memories that the switch has available per user) for either 2−or 4− qubit Greenberger–Horne–Zeilinger (GHZ)-states. For each scenario, the generation rate μ of pairs varies per user. For the blue scenario (2-partite entanglement, $\mu = [1.9, 1.9, 1.9, 1, 1, 1, 1, 1]$ MHz), the capacity was determined analytically by Vardoyan et al. using Markov Chain methods[56]. Here we extend this to 4-partite entanglement (orange scenario, same μs), for which Vardoyan et al. have found an upper bound (by assuming unbounded buffer and each μ = maximum of original rates = 1.9 MHz) but no exact analytical expression. The green scenario ($\mu = [15, 1.9, 1.9, 1, 1, 1, 1, 1]$ MHz) does not satisfy the stability condition for the Markov chain for unbounded buffer size (each leaf's rate < half of sum of all rates) so in that case steady-state capacity is not well-defined. We note that regardless of buffer size, the switch has a single link to each user, which is the reason why the capacity does not scale linearly with buffer size. **b** Average fidelity of the produced entanglement on the user nodes (no analytical results known) with unbounded buffer size. The fact that the green curve has lower fidelity than the blue one, while the former has higher rates, can be explained from the fact that the protocol prioritises entanglement which has the longest storage time (see Supplementary Note 3). Each data point represents the average of 40 runs (each 0.1 ms in simulation). Standard deviation is smaller than dot size.
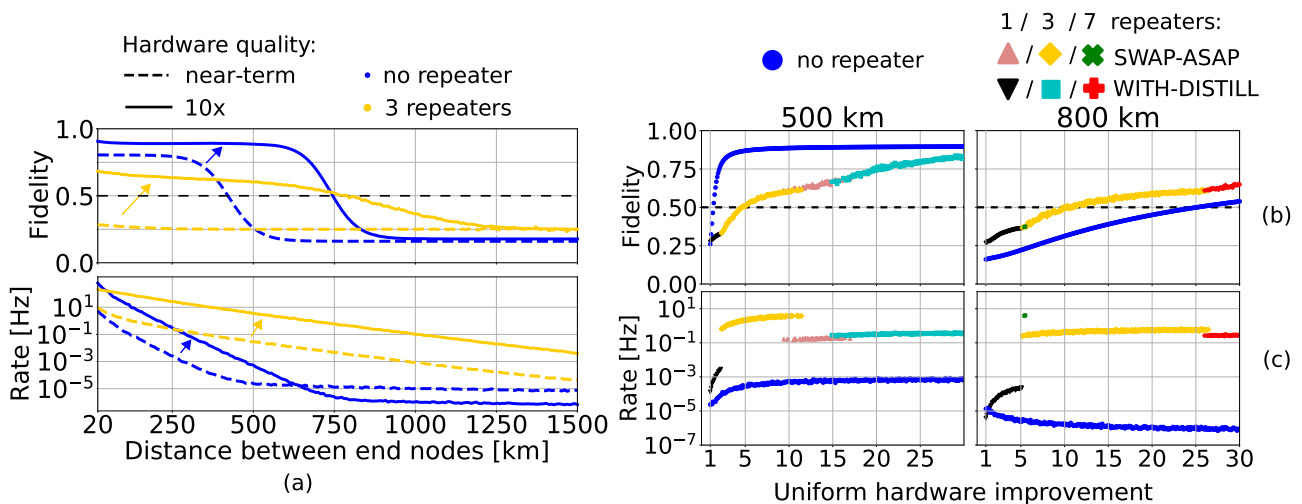
Fig. 5 Performance of repeaters based on nitrogen-vacancy (NV) centres in diamond. a Fidelity and entanglement distribution rate achieved with near-term and 10× improved hardware (Supplementary Note 4) with the SWAP-ASAP protocol. Dashed line represents classical fidelity threshold of 0.5. We observe that for near-term hardware, the use of three repeaters yields worse performance in terms of fidelity than the no-repeater setup. For improved hardware we observe (i) that for ~0–750 km, repeaters improve upon rate by orders of magnitude while still producing entanglement (fidelity > 0.5), while (ii) for ~750–1500 km, repeaters outperform in both rate and fidelity. **b, c** Fidelity and rate achieved without and with repeaters (1, 3 or 7 repeaters) as function of a hardware improvement factor (Methods, section 'How we choose improved hardware parameters') for two typical distances from both distance regime (i) and (ii), for two protocols SWAP-ASAP and NESTED-WITH-DISTILL. For the repeater case, only the best-performing number-of-repeaters and protocol in terms of achieved fidelity is shown in (**b**), accompanied by its rate in (**c**). Each data point represents the average over (**a**) 200 and (**b**) 100 runs. Standard deviation is smaller than dot size.
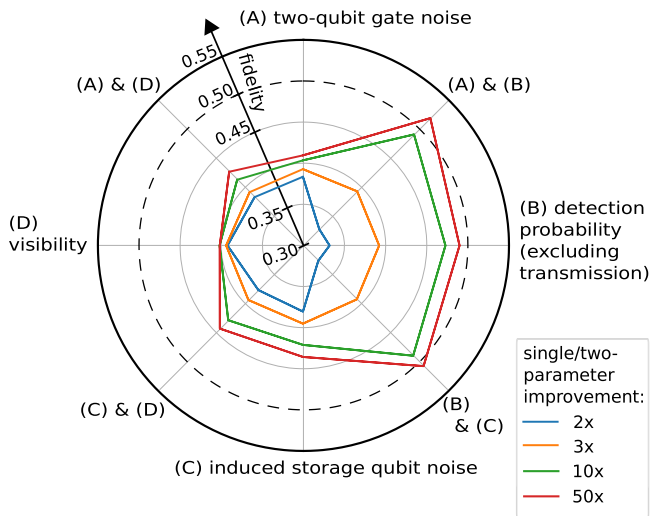
Fig. 6 Sensitivity of fidelity in various hardware parameters for nitrogen-vacancy (NV) repeater chains. The NV hardware model consists of ~15 parameters and from those we focus on four parameters in this figure: (A) two-qubit gate fidelity, (B) detection probability, (C) induced storage qubit noise and (D) visibility. We start by improving all ~15 parameters, including the four designated ones, using an improvement factor of 3 (Methods, section 'How we choose improved hardware parameters'). Then, for each of the four parameters only, we individually decrease their improvement factor to 2, or increase it to 10 or 50. The figure shows the resulting fidelity (horizontal and vertical grid lines; dashed line indicates maximal fidelity which can be attained classically). Note that at an improvement factor of 3 (orange line), all ~15 parameters are improved by three times, resulting in a fidelity of 0.39. In addition, we vary the improvement factor for combinations of two of the four parameters (diagonal lines). The 3× improved parameter values can be found in Supplementary Table II. The other values (at 2/10/50×) are approximately: two-qubit gate fidelity F_{EC} (0.985/0.997/0.9994), detection probability $p_{det}^{nofibre}$ (6.8%/58%/90%), induced storage qubit noise $N_{1/e}$ (2800/14,000/70,000), visibility V (95%/99%/99.8%). The fidelities shown are obtained by simulation of the SWAP-ASAP protocol (three repeaters) with a total spanned distance of 500 km. Each data point represents the average of 1000 runs (standard deviation on fidelity < 0.002).

all hardware parameters simultaneously and the improvement is quantified by a number we refer to as 'improvement factor' (see section 'How we choose improved hardware parameters' of the Methods). For 500 km, we observe that the no-repeater configuration achieves larger or equal fidelity for the entire range studied. However, repeater schemes boost the rate for all parameter values. If we increase the distance to 800 km, then we see that the use of repeaters increases both rate and fidelity for the same range of parameters. If we focus on the repeater scheme, we observe for both distances that for high hardware quality, the NESTED-WITH-DISTILL scheme, which includes distillation, is optimal. In contrast, for lower hardware quality, the best-performing scheme that achieves fidelities larger than the classical bound 0.5 is the SWAP-ASAP protocol.

We note that beyond 700 km the entanglement rate decreases when the hardware is improved. This is due to the presence of dark counts, i.e. false signals that a photon has been detected. At large distances most photons dissipate in the fibre, whereby the majority of detector clicks are dark counts. Because a dark count is mistakenly counted as a successful entanglement generation attempt, improving (i.e. decreasing) the dark count rate in fact results in a lower number of observed detector clicks, from which the (perceived) entanglement rate plotted in Fig. 5a is calculated.

Lastly, in Fig. 6, we investigate the sensitivity of the entanglement fidelity for the different hardware parameters. We take as the figure of merit the best fidelity achieved with a SWAP-ASAP protocol. The uniform improvement factor is set to 3, while the following four hardware parameters are varied: a two-qubit gate noise parameter, photon detection probability (excluding transmission), induced storage qubit noise and visibility. We observe that improving the detection probability yields the largest fidelity increase from 2× to 50× improvement, while this increase is smallest for visibility. We also see that improving two-qubit gate noise or induced storage qubit noise on top of an increase in detection probability yields only a small additional fidelity improvement, which however boosts fidelity beyond the classical threshold of 0.5. These observations indicate that detection probability is the most important parameter for realising remote-entanglement generation with the SWAP-ASAP scheme, followed by two-qubit gate noise and induced storage qubit noise.

Performance comparison between two atomic-ensemble memory types through NetSquid's modular design. Finally, we showcase that NetSquid's modular design greatly reduces the effort of assessing possible hardware development scenarios. We demonstrate the power of this modularity by simulating point-to-point remote-entanglement generation based on either of two types of atomic-ensemble based quantum memories: atomic frequency combs (AFC)[72] and electronically induced transparency (EIT)[73,74] memories. Both simulations are identical except for the choice of a different quantum memory component.

The two types of memories are a promising building block for high-rate remote entanglement generation through quantum repeaters because of their high efficiency (EIT) or their ability for multiplexing (AFC), i.e. to perform many attempts at entanglement generation simultaneously without network components having to wait for the arrival of classical messages that herald successful generation. The first type of memories, AFCs, are based on a photon-echo process, where an absorbed photon is re-emitted after an engineered duration. In contrast, the second type, EITs, emit the photon after an on-demand interval, due to optical control. In principle the AFC protocol can be extended to offer such on-demand retrieval as well. At this point both technologies are promising candidates and it is not yet clear which outperforms the other and under what circumstances.

Atomic-ensemble based repeaters have been analytically and numerically studied before with streamlined physical models[19,75]. NetSquid's discrete-event based paradigm allows us to go beyond that by concurrently introducing several non-ideal characteristics. In particular, we include the emission of more than one photon pair, photon distinguishability and time-dependent memory efficiency. Efficiency in this context is the probability that the absorbed photon will be re-emitted. All these characteristics have a significant impact on the performance of the repeater protocol.

In order to compare the two memory types, we simulate many rounds of the BB84 quantum key distribution protocol[76] between two remote nodes, using a single repeater positioned precisely in between them. Entanglement generation is attempted in synchronised rounds over both segments in parallel. At the end of each round, the two end nodes measure in the X- or Z-basis, chosen uniformly at random, and the repeater performs a probabilistic linear-optical Bell-state measurement. Upon a successful outcome, we expect correlation between the measurement outcomes if they were performed in the same basis. As a figure of merit we choose the asymptotic BB84 secret-key rate.

The results of our simulations are shown in Fig. 7, where the rate at which secret key between the two nodes can be generated
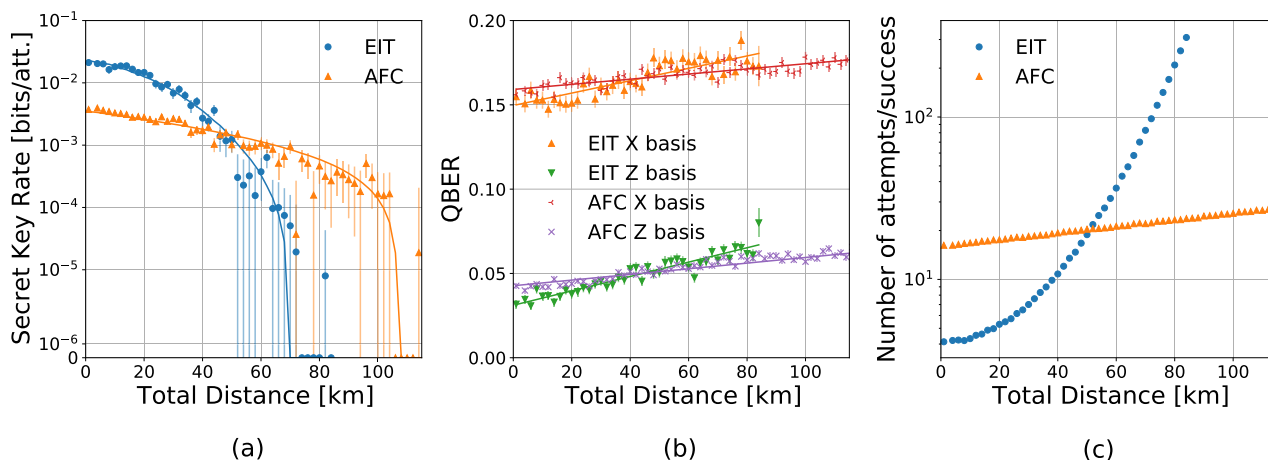
Fig. 7 Performance comparison of a single quantum repeater with atomic frequency comb (AFC) or electronically induced transparency (EIT) quantum memories. Shown are: (**a**) the secret key rate in secret bits per entanglement generation attempt, (**b**) the quantum bit error rate (QBER) in the X and Z bases (**c**) the average number of attempts necessary for one successful end-to-end entanglement generation. Each data point is obtained using 10.000 (EIT) or 30.000 (AFC) successful attempts at generating entanglement between the end nodes. Solid lines are fits. Note that for the secret key plot we use logarithmic scale with added 0 at the origin of the axes. Error bars denote standard deviation and are symmetrical.
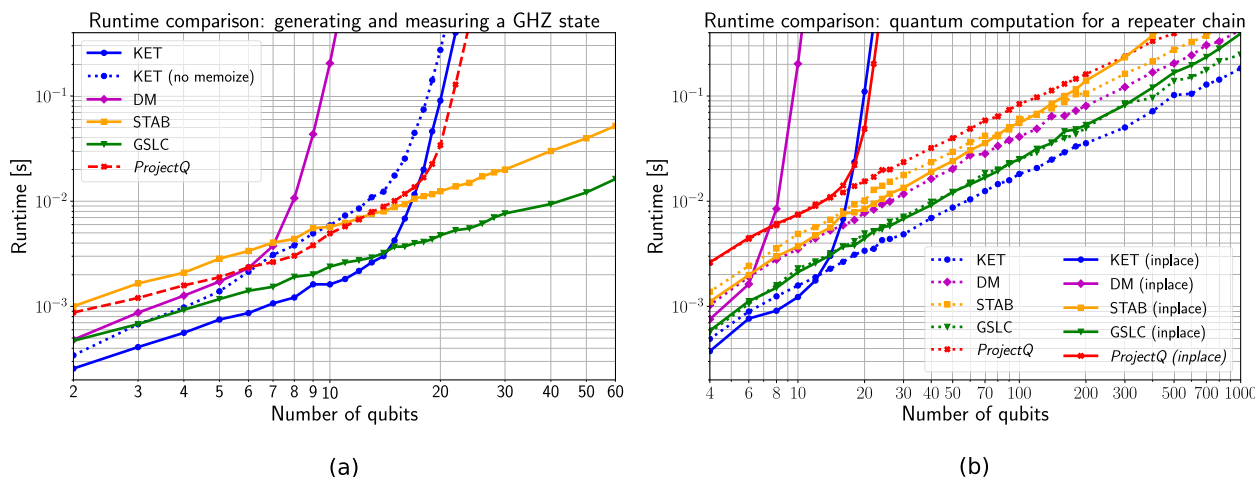
Fig. 8 Runtime comparison of NetSquid's quantum state formalisms. Runtime comparisons of the available quantum state formalisms in NetSquid as well as ProjectQ ket vector for two benchmark use cases. The KET, DM, STAB and GSLC formalisms refer to the use of ket vectors, density matrices, stabiliser tableaus and graph states with local Cliffords, respectively. **a** Generating a Greenberger-Horne-Zeilinger (GHZ) state. Qubits are *split* off from the shared quantum state after a measurement. For the KET formalism the effect of turning off memoization (dotted line) is also shown. **b** Quantum computation involved in a repeater chain. Each formalism is shown with qubits split (dotted lines) versus being kept *in-place* (solid lines) after measurement.

is obtained as a function of the distance between the nodes. For the parameters considered (see Supplementary Note 7), we observe that EIT memories outperform AFC memories at short distances. The crossover performance point is reached at ~50 km, beyond which AFC memories outperform EIT memories.

In the use case above, we showcased NetSquid's modularity by only replacing the memory component. We emphasise that this modularity also applies to different parts of the simulation. For example, if the quantum switch should produce a different type of multipartite state than GHZ states, then one only needs to change the circuit at the switch node. A different example is the NV repeater chain, where one could replace the protocol module (currently either SWAP-ASAP or NESTED-WITH-DISTILL).

Fast and scalable quantum network simulation. NetSquid has been designed and optimised to meet several key performance criteria: to be capable of accurate physical modelling, to be

scalable to large networks, and to be sufficiently fast to support multi-variate design analyses with adequate statistics. While it is not always possible to jointly satisfy all the criteria for all use cases, NetSquid's design allows the user to prioritise them. We proceed to benchmark NetSquid to demonstrate its capabilities and unique strengths for quantum network simulation.

Benchmarking of quantum computation. To accurately model physical non-idealities, it is necessary to choose a representation for quantum states that allows a characterisation of general processes such as amplitude damping, general measurements, or arbitrary rotations. NetSquid provides two representations, or 'formalisms', that are capable of universal quantum computation: ket state vectors (KET) and density matrices, both stored using dense arrays. The resource requirements for storage in memory and the computation time associated with applying quantum operations both scale exponentially with the number of qubits. While the density matrix scales less favourably, 2^{2n} versus 2^n for n

qubits, its ability to represent mixed states makes it more versatile for specific applications. Given the exponential scaling, these formalisms are most suitable for simulations in which a typical qubit lifetime involves only a limited number of (entangling) interactions.

When scaling to large network simulations it can happen that hundreds of qubits share the same entangled quantum state. For such use cases, we need a quantum state representation that scales sub-exponentially in time and space. NetSquid provides two such representations based on the stabiliser state formalism: 'stabiliser tableaus' (STAB) and 'graph states with local Cliffords' (GSLC)[77,78] that the user can select. Stabiliser states are a subset of quantum states that are closed under the application of Clifford unitaries and single-qubit measurement in the computational basis. In the context of simulations for quantum networks stabiliser states are particularly interesting because many network protocols consist of only Clifford operations and noise can be well approximated by stochastic application of Pauli gates. For a theoretical comparison of the STAB and GSLC formalisms see Supplementary Note 1.

The repetitive nature of simulation runs due to the collection of statistics via random sampling allows NetSquid to take advantage of 'memoization' for expensive quantum operations, which is a form of caching that stores the outcome of expensive operations and returns them when the same input combinations reoccur to save computation time. Specifically, the action of a quantum operator onto a quantum state for a specific set of qubit indices and other discrete parameters can be efficiently stored, for instance as a sparse matrix. Future matching operator actions can then be reduced to a fast lookup and application, avoiding several expensive computational steps—see the Methods, section 'Qubits and quantum computation' for more details.

In the following we benchmark the performance of the available quantum state formalisms. For this, we first consider the generation of an n qubit entangled GHZ state followed by a measurement of each qubit (see section 'Benchmarking' of the Methods). For a baseline comparison with classical quantum computing simulators we also include the ProjectQ[79] package for Python, which uses a quantum state representation equivalent to our ket vector. We show the average computation time for a single run versus the number of qubits for the different quantum computation libraries in Fig. 8a. The exponential scaling of the universal formalisms in contrast to the stabiliser formalisms is clearly visible, with the density matrix formalism performing noticeably worse. For the ket formalism we also show the effect of memoization, which gives a speed-up roughly between two and five.

Let us next consider a more involved benchmarking use case: the quantum computation involved in simulating a repeater chain i.e. only the manipulation of qubits, postponing all other simulation aspects, such as event processing and component modelling, to the next section. This benchmark involves the following steps: first the $N-1$ pairs of qubits along an N node repeater chain are entangled, then each qubit experiences depolarising noise, and finally adjacent qubits on all but the end-nodes do an entanglement swap via a Bell state measurement (BSM). If the measured qubits are split from their shared quantum states after the BSM, then the size of any state is limited to four qubits.

The average computation time for a single run versus the number of qubits in the chain are shown for the different quantum computation libraries in Fig. 8b, where we have again included ProjectQ. We observe that for the NetSquid formalisms (but not for ProjectQ) keeping qubits 'in-place' after each measurement is more performant than 'splitting' them below a certain threshold due to the extra overhead of doing the latter.
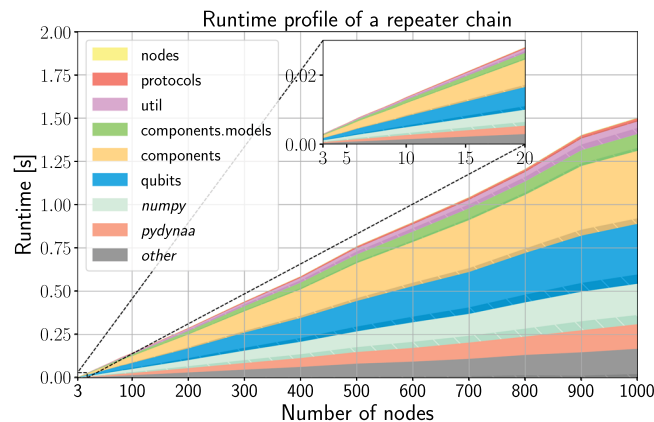
**Fig. 9 Runtime profile of a repeater chain simulation using Netsquid.** Runtime profile for a repeater chain simulation with a varying number of nodes in the chain. The maximum quantum state size is four qubits. The total time spent in the functions of each NetSquid subpackage and its main package dependencies (in italics) is shown. The dark hatched bands show the largest contribution from a single function in each NetSquid sub-package, as well as in NumPy and uncategorised (*other*) functions. The sub-packages are stacked in the same order as they are listed in the legend.

The ket vector formalism is seen to be the most efficient for this benchmarking use case if states are split after measurement. When the measurement operations are performed in-place the GSLC formalism performs the best beyond 15 qubits.

Benchmarking of event-driven simulations. As explained in the results section, a typical NetSquid simulation involves repeatedly sampling many independent runs. As such NetSquid is 'embarrassingly parallelisable': the reduction in runtime scales linearly with the number of processing cores available, assuming there is sufficient memory available. Nonetheless, given the computational requirements associated with collecting sufficient statistics and analysing large parameter spaces it remains crucial to optimise the runtime performance per core.

Depending on the size of the network, the detail of the physical modelling, and the duration of the protocols under consideration, the number of events processed for a single simulation run can range anywhere from a few thousand to millions. To efficiently process the dynamic scheduling and handling of events NetSquid uses the discrete-event simulation engine PyDynAA[80] (see section 'Discrete event simulation' of the Methods). NetSquid aims to schedule events as economically as possible, for instance by streamlining the flow of signals and messages between components using inter-connecting ports.

To benchmark the performance of an event-driven simulation run in NetSquid we consider a simple network that extends the single repeater (without distillation) shown in Fig. 1 into an N node chain—see Supplementary Note 2 for further details on the simulation setup. For the quantum computation we will use the ket vector formalism based on the benchmarking results from the previous section, and split qubits from their quantum states after measurement to avoid an exponential scaling with the number of nodes. In Fig. 9 we show the average computation time for deterministically generating end-to-end entanglement versus the number of nodes in the chain. Also shown is a relative breakdown in terms of the time spent in the NetSquid sub-packages involved, as well as the PyDynAA and NumPy packages. We observe that the biggest contribution to the simulation runtime is the components sub-package, which accounts for 30% of the total at 1000 nodes. The relative time spent in each of the NetSquid sub-packages, as well as NumPy and PyDynAA, is seen to remain
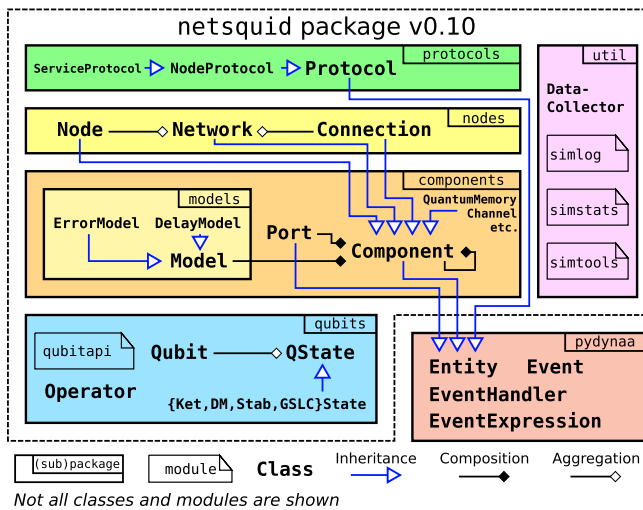
Fig. 10 Overview of NetSquid's software architecture. The sub-packages that make up the NetSquid package are shown stacked in relation to each other and the PyDynAA package dependency. The main classes in each (sub-)package are highlighted, and their relationships in terms of inheritance, composition and aggregation are shown. Also shown are the key modules users interact with, which are described in the main text. In this paper NetSquid version 0.10 is described.

constant with the number of nodes. The total runtime of each of the NetSquid sub-packages is the sum of many small contributions, with the costliest function for the components sub-package for a 1000 node chain, for example, contributing only 7% to the total.

Extending this benchmark simulation with more detailed physical modelling may shift the relative runtime distribution and impact the overall performance. For example, more time may be spent in calls to the 'components' and 'components.models' sub-packages, additional complexity can increase the volume of events processed by the 'pydynaa' engine, and extra quantum characteristics can lead to larger quantum states. In case of the latter, however, the effective splitting of quantum states can still allow such networks to scale if independence among physical elements can be preserved.

Comparison with other quantum network simulators. Let us compare NetSquid to other existing quantum network simulators. First, SimulaQron[81] and QuNetSim[82] are two simulators that do not aim at realistic physical models of channels and devices, or timing control. Instead, SimulaQron's main purpose is application development. It is meant to be run in a distributed fashion on physically-distinct classical computers. QuNetSim focuses on simplifying the development and implementation of quantum network protocols.

In contrast with SimulaQron and QuNetSim, the simulator SQUANCH[83] allows for quantum network simulation with configurable error models at the physical layer. However, SQUANCH, similar to SimulaQron and QuNetSim, does not use a simulation engine that can accurately track time. Accurate tracking is crucial for e.g. studying time-dependent noise such as memory decoherence.

Other than NetSquid, there now exist three discrete-event quantum simulators: the QuISP[84], qkdX[85] and SeQUeNCe[86] simulators. With these simulators it is possible to accurately characterise complex timing behaviour, however they differ in goals and scope. Similarly to NetSquid, QuISP aims to support the investigation of large networks that consist of too many entangled qubits for full quantum-state tracking. In contrast to

NetSquid, which achieves this by managing the size of the state space, and providing the stabiliser representation as one of its quantum state formalisms, QuISP's approach is to track an error model of the qubits in a network instead of their quantum state. qkdX, on the other hand, captures the physics more closely through models of the quantum devices but is restricted to the simulation of quantum key distribution protocols. Lastly, SeQUeNCe, similar to NetSquid, aims at simulation at the level of hardware, control plane or application. It has a fixed control layer consisting of reprogrammable modules. In contrast, NetSquid's modularity is not tied to a particular network stack design. Furthermore, it is unclear to us how performant SeQUeNCe's quantum simulation engine is: currently, at most a 9-node network has been simulated, whereas NetSquid's flexibility to choose a quantum state representation enables scalability to simulation of networks of up to 1000 nodes.

Conclusions. In this work we have presented our design of a modular software framework for simulating scalable quantum networks and accurately modelling the non-idealities of real world physical hardware, providing us with a design tool for future quantum networks. We have showcased its power and also its limitations via example use cases. Let us recap NetSquid's main features.

First, NetSquid allows the modelling of any physical device in the network that can be mapped to qubits. To demonstrate this we studied two use cases involving nitrogen-vacancy centres in diamond as well as atomic-ensemble based memories.

Second, NetSquid is entirely modular, allowing users to set up large scale simulations of complicated networks and to explore variations in the network design; for example, by comparing how different hardware platforms perform in an otherwise identical network layout. Moreover, this modularity makes it possible to explore different control plane protocols for quantum networks in a way that is essentially identical to how such protocols would be executed in the real world. Control programmes can be run on any simulated network node, exchanging classical and quantum communication with other nodes as dictated by the protocol. That allows users to investigate the intricate interplay between control plane protocols and the physical devices dictating the performance of the combined quantum network system. As an example, we studied the control plane of a quantum network switch. NetSquid has also already found use in exploring the interplay between the control plane and the physical layer in[39,87,88].

Finally, to allow large scale simulations, the quantum computation library used by NetSquid has been designed to manage the dynamic lifetimes of many qubits across a network. It offers a seamless choice of quantum state representations to support different modelling use cases, allowing both a fully detailed simulation in terms of wave functions or density matrices, or simplified ones using certain stabiliser formalisms. As an example use case, we explored the simulation run-time of a repeater chain with up to one thousand nodes.

In light of the results we have presented, we see a clear application for NetSquid in the broad context of communication networks. It can be used to predict performance with accurate models, to study the stability of large networks, to validate protocol designs, to guide experiment, etc. While we have only touched upon it in our discussion of performance benchmarks, NetSquid would also lend itself well to the study of modular quantum computing architectures, where the timing of control plays a crucial role in studying their scalability. For instance, it might be used to validate the microarchitecture of distributed

quantum computers or more generally to simulate different components in modular architectures.

Methods

Design and functionality of NetSquid. The NetSquid simulator is available as a software package for the Python 3 programming language. It consists of the sub-packages 'qubits', 'components', 'models', 'nodes', 'protocols' and 'util', which are shown stacked in Fig. 10. NetSquid depends on the PyDynAA software library to provide its discrete-event simulation engine[80]. Under the hood speed critical routines and classes are written in Cython[89] to give C-like performance, including its interfaces to both PyDynAA and the scientific computation packages NumPy and SciPy. In the following subsections we highlight some of the main design features and functionality of NetSquid; for a more detailed presentation see Supplementary Note 1.

Discrete event simulation. The PyDynAA package provides a fast, powerful, and lightweight discrete-event simulation engine. It is a C++ port of the core engine layer from the DynAA simulation framework[80], with bindings added for the Python and Cython languages. DynAA defines a concise set of classes and concepts for modelling event-driven simulations. The simulation engine manages a timeline of 'events', which can only be manipulated by objects that are sub-classes of the 'entity' base class. Simulation entities can dynamically schedule events on the timeline and react to events by registering an 'event handler' object to wait for event(s) with a specified type, source entity, or identifier to be triggered.

To deal with the timing complexities encountered in NetSquid simulations, an 'event expression' class was introduced to PyDynAA to allow entities to also wait on logical combinations of events to occur. Atomic event expressions, which describe regular wait conditions for standard events, can be combined to form composite expressions using logical 'and' and 'or' operators to any depth. This feature has been used extensively in NetSquid to model both the internal behaviour of hardware components, as well as for programming network protocols.

Qubits and quantum computation. The qubits sub-package of NetSquid defines the 'qubit' object that is used to track the flow of quantum information. Qubits internally share quantum state ('QState') objects, which grow and shrink in size as qubits interact or are measured. The 'QState' class is an interface that is implemented by a range of different formalisms, as presented in section 'Benchmarking of quantum computation' of the Results and Discussion. Via the qubit-centric API, which provides functions to directly manipulate qubits without knowledge of their shared quantum states, users can programme simulations in a formalism agnostic way. Functionality is also provided to automatically convert between quantum states that use different formalisms, and to sample from a distribution of states, which is useful for instance for pure state formalisms.

The ket and density matrix formalisms use dense arrays (vectors or matrices, respectively) to represent quantum states. Applying a k qubit operator to an n qubit ket vector state generally involves the computationally expensive task of performing 2^{n-k} matrix multiplications on 2^k temporary sub-vectors and aggregating the result (only in special cases can this be done in-place)[90,91]. The analogous application of an operator to a density matrix is more expensive due to the extra dimension involved. However, as discussed in section 'Fast and scalable quantum network simulation' of the Results and Discussion, the repetitive nature of NetSquid simulations allows us to take advantage of operators frequently being applied to the same qubit indices for states of a given size. For these operators, we compute a $2^n \times 2^n$ dimensional sparse matrix representation of the k qubit operator via tensor products with the identity and memoize this result for the specific indices and size. When the memoization is applicable the computational cost of applying a quantum operator can then be reduced to just sparse matrix multiplication onto a dense vector or matrix. Memoization is similarly applicable to general Clifford operators in the stabiliser tableau formalism. To use memoization on operators that depend on a continuous parameter, such as arbitrary rotations, the parameter can be discretised i.e. rounded to some limited precision.

Physical modelling of network components. All physical devices in a quantum network are modelled by a 'component' object, and are thereby also all simulation entities, as shown in Fig. 10. Components can be composed of subcomponents, which makes setting up networks in NetSquid modular. The network itself, for instance, can be modelled as a composite component containing 'node' and 'connection' components; these composite components can in turn contain components such as quantum memories, quantum and classical channels, quantum sources, etc., as illustrated in Fig. 1. The physical behaviour of a component is described by composing it of 'models', which can specify physical characteristics such as transmission delays or noise such as photon loss or decoherence. Communication between components is facilitated by their 'ports', which can be connected together to automatically pass on messages.

NetSquid also allows precise modelling of quantum computation capable devices. For this it provides the 'quantum processor' component, a subclass of the quantum memory. This component is capable of executing 'quantum programmes' i.e. sequences of 'instructions' that describe operations such as quantum gates and measurements or physical processes such as photon emission. Quantum

programmes fully support conditional and iterative statements, as well as parallelisation if the modelled device supports it. When a programme is executed its instructions are mapped to the physical instructions on the processor, which model the physical duration and errors associated to carrying out the operation. A physical instruction can be assigned to all memory positions or only to a specific position, as well as directionally between specific memory positions in the case of multi-qubit instructions.

Asynchronous framework for programming protocols. NetSquid provides a 'protocol' class to describe the network protocols and classical control plane logic running on a quantum network. Similarly to the component class, a protocol is a simulation entity and can thereby directly interact with the event timeline. Protocols can be nested inside other protocols and may describe both local or remote behaviour across a network. The 'node protocol' subclass is specifically restricted to only operating locally on a single node. Inter-protocol communication is possible via a signalling mechanism and a request and response interface defined by the 'service protocol' class. Protocols can be programmed using both the standard callback functionality of PyDynAA and a tailored asynchronous framework that allows the suspension of a routine conditioned on an 'event expression'; for example, to wait for input to arrive on a port, a quantum programme to finish, or to pause for a fixed duration.

The 'util' sub-package shown in Fig. 10 provides a range of utilities for running, recording and interacting with simulations. Functions to control the simulation are defined in the 'simtools' module, including functions for inspecting and diagnosing the timeline. A 'data collector' class supports the event-driven collection of data during a simulation, which has priority over other event handlers to react to events. The 'simstats' module is responsible for collecting a range of statistics during a simulation run, such as the number of events and callbacks processed, the maximum and average size of manipulated quantum states, and a count of all the quantum operations performed. Finally, the 'simlog' module allows fine grained logging of the various modules for debugging purposes.

Benchmarking. To perform the benchmarking described in section 'Fast and scalable quantum network simulation' of the Results and Discussion we used computing nodes with two 2.6 GHz Intel Xeon E5-2690 v3 (Haswell) 12 core processors and 64 GB of memory. Because each process only requires a single core, care was taken to ensure sufficient cores and memory were available when running jobs in parallel. The computation time of a process is the arithmetic average of a number of successive iterations; to avoid fluctuations due to interfering CPU processes the reported time is a minimum of five such repeated averages. To perform the simulation profiling the Cython extension modules of both NetSquid and PyDynAA were compiled with profiling on, which adds some runtime overhead. Version 0.10.0 and 0.3.5 of NetSquid and PyDynAA were benchmarked. We benchmarked against ProjectQ version 0.4.2 using its 'MainEngine' backend. See Supplementary Note 2 for further details.

Using the same machine, simulations for Fig. 5b, c were run, which took almost 260 core hours wallclock time in total, while simulations for Fig. 7 took roughly 625 core hours. For Fig. 4 (≈10 h in total), Fig. 5(a) (≈90 min) and Fig. 6 (≈30 min), a single core Intel Xeon Gold 6230 processor (3.9 GHz) with 192 GB RAM was used.

Implementing a processing-node repeater chain in NetSquid. Here, we explain the details of the most complex of our three use cases, namely the repeater chain of Nitrogen-Vacancy-based processing nodes from section 'Sensitivity analysis for the physical modelling of a long range repeater chain' of the Results and Discussion (see Supplementary Notes 3 and 7 for details on the other two use cases). We first describe how we modelled the NV hardware, followed by the repeater protocols used. With regard to the physical modelling, let us emphasise that this is well established (see e.g.[92]); the main goal here is to explain how we used this model in a NetSquid implementation.

In our simulations the following NetSquid components model the physical repeater chain: 'nodes', each holding a single 'quantum processor' modelling the NV centre, and 'classical channels' that connect adjacent nodes and are modelled as fibres with a constant transmission time. We choose equal spacing between the nodes. If we were to simulate individual attempts at entanglement generation, we would also need components for transmitting and detecting qubits such as was used in previous NetSquid simulations of NV centres[39]. However, in order to speed up simulations we insert the entangled state between remote NVs using a model. We designed two types of protocols to run on each node of this network that differ in whether they implement a scheme with or without distillation.

In the remainder of this section, we describe the components modelling. More detailed descriptions of the hardware parameters and their values used in our simulation can be found in Supplementary Note 4.

Modelling a nitrogen-vacancy centre in diamond. In NetSquid, the NV centre is modelled by a quantum processor component, which holds a single communication qubit (electronic spin-1 system) and multiple storage qubits (^{13}C nuclear spins). The decay of the state held by a communication qubit or storage qubit is implemented using a noise model, which is based on the relaxation time T_1 and the dephasing time T_2. If a spin is acted upon after having been idle for time Δt, then to

its state ρ we first apply a quantum channel

$$\rho \mapsto E_0 \rho E_0^\dagger + E_1 \rho E_1^\dagger$$

where

$$E_0 = |0\rangle\langle 0| + \sqrt{1-p}|1\rangle\langle 1|, E_1 = \sqrt{p}|0\rangle\langle 1|$$

and $p = 1 - e^{-\Delta t/T_1}$. Subsequently, we apply a dephasing channel

$$\mathcal{N}_p^{\mathrm{deph}} : \rho \mapsto (1-p)\rho + pZ\rho Z \qquad (1)$$

where $Z = |0\rangle\langle 0| - |1\rangle\langle 1|$ and the dephasing probability equals

$$p = \frac{1}{2}\left(1 - e^{-\Delta t/T_2} \cdot e^{\Delta t/(2T_1)}\right).$$

The electron and nuclear spins have different T_1 and T_2 times.

We allow the quantum processor to perform the following operations on the electron spin: initialisation (setting the state to $|0\rangle$), readout (measurement in the $\{|0\rangle, |1\rangle\}$ basis) and arbitrary single-qubit rotation. In particular, the latter includes Pauli rotations

$$R_P(\theta) = \cos(\theta/2)\mathbb{1}_2 - i\sin(\theta/2)P \qquad (2)$$

where θ is the rotation angle, $P \in \{X, Y, Z\}$ and $\mathbb{1}_2 = |0\rangle\langle 0| + |1\rangle\langle 1|$, $X = |0\rangle\langle 1| + |1\rangle\langle 0|$, $Y = -i|0\rangle\langle 1| + i|1\rangle\langle 0|$ and $Z = |0\rangle\langle 0| - |1\rangle\langle 1|$ are the single-qubit Pauli operators.

For the nuclear spin, we have only initialisation and rotations $R_Z(\theta)$ for arbitrary rotation angle θ. In addition, we allow the two-qubit controlled-$R_X(\pm\theta)$ gate between an electron (e) and a nuclear (n) spin:

$$|0\rangle\langle 0|_e \otimes R_X(\theta)_n + |1\rangle\langle 1|_e \otimes R_X(-\theta)_n.$$

We model each noisy operation O_{noisy} as the perfect operation O_{perfect} followed by a noise channel $\mathcal{N}$:

$$O_{\mathrm{noisy}} = \mathcal{N} \circ O_{\mathrm{perfect}}.$$

If O is a single-qubit rotation, then $\mathcal{N}$ is the depolarising channel:

$$\mathcal{N}_p^{\mathrm{depol}} : \rho \mapsto \left(1 - \frac{3p}{4}\right)\rho + \frac{p}{4}\left(X\rho X + Y\rho Y + Z\rho Z\right) \qquad (3)$$

with parameter $p = 4(1-F)/3$ with F the fidelity of the operation.

If O is single-qubit initialisation, $\mathcal{N} = \mathcal{N}_p^{\mathrm{depol}}$ with parameter $p = 2(1-F)$. The noise map of the controlled-R_X gate is an identical single-qubit depolarising channel on both involved qubits, i.e. $\mathcal{N} = \mathcal{N}_p^{\mathrm{depol}} \otimes \mathcal{N}_p^{\mathrm{depol}}$.

Finally, we model electron spin readout by a POVM measurement with the Kraus operators

$$M_0 = \begin{pmatrix} \sqrt{f_0} & 0 \\ 0 & \sqrt{1-f_1} \end{pmatrix}, \; M_1 = \begin{pmatrix} \sqrt{1-f_0} & 0 \\ 0 & \sqrt{f_1} \end{pmatrix} \qquad (4)$$

where $1 - f_0$ ($1 - f_1$) is the probability that a measurement outcome 0 (1) is flipped to 1 (0).

Simulation speedup via state insertion. For generating entanglement between the electron spins of two remote NVs, we simulate a scheme based on single-photon detection, following its experimental implementation in[93]. NetSquid was used previously to simulate each generation attempt of this scheme, which includes the emission of a single photon by each NV, the transmission of the photons to the midpoint through a noisy and lossy channel, the application of imperfect measurement operators at the midpoint, and the transmission of the measurement outcome back to the two involved nodes[39]. For larger internode distances, simulating each attempt requires unfeasibly long simulation times due to the exponential decrease in attempt success rate. To speed up our simulations in the examples studied here, we generate the produced state between adjacent nodes from a model which has shown good agreement with experimental results[93]. This procedure includes a random duration and noise induced on the storage qubits, as we describe below.

Let us define

$$
\begin{aligned}
p_{00} =\; & \alpha^2\big[2p_{\mathrm{det}}(1-p_{\mathrm{det}})(1-p_{\mathrm{dc}}) \\
& + 2p_{\mathrm{dc}}(1-p_{\mathrm{dc}})(1-p_{\mathrm{det}})^2 \\
& + p_{\mathrm{det}}^2(1-p_{\mathrm{dc}}) \cdot \frac{1}{2}(1+V)\big] \\
p_{10} =\; & \alpha(1-\alpha) \cdot \big[(1-p_{\mathrm{dc}}) \cdot p_{\mathrm{det}} \\
& + 2p_{\mathrm{dc}}(1-p_{\mathrm{dc}})(1-p_{\mathrm{det}})\big] \\
p_{01} =\; & p_{10} \\
p_{11} =\; & (1-\alpha)^2 \cdot p_{\mathrm{dc}}
\end{aligned}
$$

where p_{det} is the detection probability, p_{dc} the dark count probability, V denotes photon indistinguishability and α is the bright-state parameter (see Supplementary Note 4 for parameter descriptions). We follow the model of the produced

entangled state from the experimental work of [93], whose setup consists of a beam splitter with two detectors located between the two adjacent nodes. In their model, the unnormalised state is given by

$$\rho = \begin{pmatrix} p_{00} & 0 & 0 & 0 \\ 0 & p_{01} & \pm\sqrt{V p_{01} p_{10}} & 0 \\ 0 & \pm\sqrt{V p_{01} p_{10}} & p_{10} & 0 \\ 0 & 0 & 0 & p_{11} \end{pmatrix}$$

where $\pm$ denotes which of the two detectors detected a photon (each occurring with probability $\frac{1}{2}$). We also follow the model of[93] for double-excitation noise and optical phase uncertainty, by applying a dephasing channel to both qubits with parameter $p = p_{\mathrm{dexc}}/2$, followed by a dephasing channel of one of the qubits, respectively.

The success probability of a single attempt is

$$p_{\mathrm{succ}} = p_{00} + p_{01} + p_{10} + p_{11}.$$

The time elapsed until the fresh state is put on the electron spins is $(k-1)\cdot\Delta t$ with $\Delta t := (t_{\mathrm{emission}} + L/c)$, where t_{emission} is the delay until the NV centre emits a photon, L the internode distance and c the speed of light in fibre. Here, k is the number of attempts up to and including successful entanglement generation and is computed by drawing a random sample from the geometric distribution $\mathrm{Pr}(k) = p_{\mathrm{succ}} \cdot (1 - p_{\mathrm{succ}})^{k-1}$. After the successful generation, we wait for another time Δt to mimic the photon travel delay and midpoint heralding message delay.

Every entanglement generation attempt induces dephasing noise on the storage qubits in the same NV system. We apply the dephasing channel (eq. (1)) at the end of the successful entanglement generation, where the accumulated dephasing probability is

$$\frac{1 - (1 - 2p_{\mathrm{single}})^k}{2} \qquad (5)$$

where p_{single} is the single-attempt dephasing probability (see eq. 46 in Supplementary Note 4).

How we choose improved hardware parameters. Here, we explain how we choose 'improved' hardware parameters. Let us emphasise that this choice is independent of the setup of our NetSquid simulations and only serves the purpose of showcasing that NetSquid can assess the performance of hardware with a given quality.

By 'near-term' hardware, we mean values for the above defined parameters as expected to be achieved in the near future by NV hardware. If we say that an error probability is improved by an improvement factor k, we mean that its corresponding no-error probability equals $\sqrt[k]{p_{\mathrm{ne}}}$, where p_{ne} is the no-error probability of the near-term hardware. For example, visibility V is improved as $\sqrt[k]{V}$ while the probability of dephasing p of a gate is improved as $1 - \sqrt[k]{1-p}$. A factor $k = 1$ thus corresponds to 'near-term' hardware. By 'uniform hardware improvement by k', we mean that all hardware parameters are improved by a factor k. We do not improve the duration of local operations or the fibre attenuation. The near-term parameter values as well as the individual improvement functions for each parameter can be found in Supplementary Note 4.

NV repeater chain protocols. For the NV repeater chain, we simulated two protocols: SWAP-ASAP and NESTED-WITH-DISTILL. Both protocols are composed of five building blocks: ENTGEN, STORE, RETRIEVE, DISTILL and SWAP. By ENTGEN, we denote the simulation of the entanglement generation protocol based on the description in the previous subsection: two nodes wait until a classical message signals that their respective electron spins hold an entangled pair. In reality, such functionality could be achieved by a link layer protocol[39]. STORE is the mapping of the electron spin state onto a free nuclear spin, and RETRIEVE is the reverse operation. The DISTILL block implements entanglement distillation between two remote NVs for probabilistically improving the quality of entanglement between two nuclear spins (one at each NV), at the cost of reading out entanglement between the two electron spins. It consists of local operations followed by classical communication to determine whether distillation succeeded. The entanglement swap (SWAP) converts two short-distance entangled qubit pairs $A - M$ and $M - B$ into a single long-distance one $A - B$, where A, B and M are nodes. It consists of local operations at M, including spin readout, and communicating the measurement outcomes to A and B, followed by A and B updating their knowledge of the precise state $A - B$ they hold in the perfect case. We opt for such tracking as opposed to applying a correction operator to bring $A - B$ back to a canonical state since the correction operator generally cannot be applied to the nuclear spins directly. Details of the tracking are given in Supplementary Note 6. The circuit implementations for the building blocks, 'quantum programmes' in NetSquid, are given in Supplementary Note 5.

Let us explain the SWAP-ASAP and NESTED-WITH-DISTILL protocols in spirit; the exact protocols run asynchronously on each node and can be found in Supplementary Note 5. In the SWAP-ASAP protocol, a repeater node performs ENTGEN with both its neighbours, followed by SWAP as soon as it holds the two entangled pairs. Next, NESTED-WITH-DISTILL is a nested protocol on $2^n + 1$ nodes (integer $n \geq 0$) with distillation at each nesting level which is based on the

BDCZ protocol[9]. For nesting level $n = 0$, there are no repeaters and the two nodes only perform ENTGEN once. For nesting level $n > 0$, the chain is divided into a left part and a right part of $2^{n-1} + 1$ nodes, and the middle node (included in both parts) in the chain generates twice an entangled pair with the left end node following the $(n - 1)$-level protocol; STORE is applied in between to free the electron spin. Subsequently, DISTILL is performed with the two pairs as input (restart if distillation fails), after which the same procedure is performed on the right. Once the right part has finished, the middle node performs SWAP to connect the end nodes. If needed, STORE and RETRIEVE are applied prior to DISTILL and SWAP in order achieve the desired configuration of qubits in the quantum processor, e.g. for DISTILL to ensure that the two involved NVs hold an electron–electron and nuclear–nuclear pair of qubits, instead of electron-nuclear for both entangled pairs.

Data availability

Code availability

References

1. Van Meter, R. & Devitt, S. J. The path to scalable distributed quantum computing. *Computer* **49**, 31–42 (2016).
2. Lekitsch, B. et al. Blueprint for a microwave trapped ion quantum computer. *Sci. Adv.* **3**, e1601540 (2017).
3. Monroe, C. et al. Large-scale modular quantum-computer architecture with atomic memory and photonic interconnects. *Phys. Rev. A* **89**, 022317 (2014).
4. Stephens, A. M. et al. Deterministic optical quantum computer using photonic modules. *Phys. Rev. A* **78**, 032318 (2008).
5. Wehner, S., Elkouss, D. & Hanson, R. Quantum internet: a vision for the road ahead. *Science* 362 https://science.sciencemag.org/content/362/6412/eaam9288. https://science.sciencemag.org/content/362/6412/eaam9288.full.pdf (2018).
6. Munro, W. J., Azuma, K., Tamaki, K. & Nemoto, K. Inside quantum repeaters. *IEEE J. Selec. Top. Quantum Electron.* **21**, 78–90 (2015).
7. Muralidharan, S. et al. Optimal architectures for long distance quantum communication. *Sci. Rep.* **6**, 20463 https://doi.org/10.1038/srep20463 EP— (2016).
8. Gisin, N. & Thew, R. Quantum communication. *Nat. Photon.* **1**, 165 https://doi.org/10.1038/nphoton.2007.22. EP—(2007).
9. Briegel, H.-J., Dür, W., Cirac, J. I. & Zoller, P. Quantum repeaters: the role of imperfect local operations in quantum communication. *Phys. Rev. Lett.* **81**, 5932–5935 (1998).
10. Dür, W., Briegel, H.-J., Cirac, J. I. & Zoller, P. Quantum repeaters based on entanglement purification. *Phys. Rev. A* **59**, 169–181 (1999).
11. Duan, L.-M., Lukin, M. D., Cirac, J. I. & Zoller, P. Long-distance quantum communication with atomic ensembles and linear optics. Nature **414**, 413 https://doi.org/10.1038/35106500 EP – (2001).
12. Amirloo, J., Razavi, M. & Majedi, A. H. Quantum key distribution over probabilistic quantum repeaters. *Phys. Rev. A* **82**, 032304 (2010).
13. Kimiaee Asadi, F. et al. Quantum repeaters with individual rare-earth ions at telecommunication wavelengths. *Quantum* **2**, 93 (2018).
14. Bernardes, N. K., Praxmeyer, L. & van Loock, P. Rate analysis for a hybrid quantum repeater. *Phys. Rev. A* **83**, 012323 (2011).
15. Borregaard, J., Kómár, P., Kessler, E. M., Sørensen, A. S. & Lukin, M. D. Heralded quantum gates with integrated error detection in optical cavities. *Phys. Rev. Lett.* **114**, 110502 (2015).
16. Bruschi, D. E., Barlow, T. M., Razavi, M. & Beige, A. Repeat-until-success quantum repeaters. *Phys. Rev. A* **90**, 032306 (2014).
17. Chen, Z.-B., Zhao, B., Chen, Y.-A., Schmiedmayer, J. & Pan, J.-W. Fault-tolerant quantum repeater with atomic ensembles and linear optics. *Phys. Rev. A* **76**, 022329 (2007).
18. Collins, O. A., Jenkins, S. D., Kuzmich, A. & Kennedy, T. A. B. Multiplexed memory-insensitive quantum repeaters. *Phys. Rev. Lett.* **98**, 060502 (2007).
19. Guha, S. et al. Rate-loss analysis of an efficient quantum repeater architecture. *Phys. Rev. A* **92**, 022357 (2015).
20. Hartmann, L., Kraus, B., Briegel, H.-J. & Dür, W. Role of memory errors in quantum repeaters. *Phys. Rev. A* **75**, 032310 (2007).
21. Jiang, L. et al. Quantum repeater with encoding. *Phys. Rev. A* **79**, 032325 (2009).
22. Nemoto, K. et al. Photonic quantum networks formed from NV-centers. Sci. Rep. **6**, 26284 https://doi.org/10.1038/srep26284 EP – (2016).
23. Razavi, M., Piani, M. & Lütkenhaus, N. Quantum repeaters with imperfect memories: cost and scalability. *Phys. Rev. A* **80**, 032301 (2009).
24. Razavi, M. & Shapiro, J. H. Long-distance quantum communication with neutral atoms. *Phys. Rev. A* **73**, 042303 (2006).
25. Simon, C. et al. Quantum repeaters with photon pair sources and multimode memories. *Phys. Rev. Lett.* **98**, 190503 (2007).
26. Vinay, S. E. & Kok, P. Practical repeaters for ultralong-distance quantum communication. *Phys. Rev. A* **95**, 052336 (2017).
27. Wu, Y., Liu, J. & Simon, C. Near-term performance of quantum repeaters with imperfect ensemble-based quantum memories. *Phys. Rev. A* **101**, 042301 (2020).
28. Sangouard, N. et al. Long-distance entanglement distribution with single-photon sources. *Phys. Rev. A* **76**, 050301 (2007).
29. Sangouard, N. et al. Robust and efficient quantum repeaters with atomic ensembles and linear optics. *Phys. Rev. A* **77**, 062301 (2008).
30. Sangouard, N., Dubessy, R. & Simon, C. Quantum repeaters based on single trapped ions. *Phys. Rev. A* **79**, 042340 (2009).
31. Abruzzo, S. et al. Quantum repeaters and quantum key distribution: analysis of secret-key rates. *Phys. Rev. A* **87**, 052315 (2013).
32. Brask, J. B. & Sørensen, A. S. Memory imperfections in atomic-ensemble-based quantum repeaters. *Phys. Rev. A* **78**, 012350 (2008).
33. Muralidharan, S., Kim, J., Lütkenhaus, N., Lukin, M. D. & Jiang, L. Ultrafast and fault-tolerant quantum communication across long distances. *Phys. Rev. Lett.* **112**, 250501 (2014).
34. Pant, M., Krovi, H., Englund, D. & Guha, S. Rate-distance tradeoff and resource costs for all-optical quantum repeaters. *Phys. Rev. A* **95**, 012304 (2017).
35. Ladd, T. D., van Loock, P., Nemoto, K., Munro, W. J. & Yamamoto, Y. Hybrid quantum repeater based on dispersive CQED interactions between matter qubits and bright coherent light. *New J. Phys.* **8**, 184–184 (2006).
36. van Loock, P. et al. Hybrid quantum repeater using bright coherent light. *Phys. Rev. Lett.* **96**, 240501 (2006).
37. Zwerger, M. et al. Quantum repeaters based on trapped ions with decoherence-free subspace encoding. *Quantum Sci. Technol.* **2**, 044001 (2017).
38. Jiang, L., Taylor, J. M. & Lukin, M. D. Fast and robust approach to long-distance quantum communication with atomic ensembles. *Phys. Rev. A* **76**, 012301 (2007).
39. Dahlberg, A. et al. A link layer protocol for quantum networks. In *Proceedings of the ACM Special Interest Group on Data Communication*, SIGCOMM '19, 159–173 https://doi.org/10.1145/3341302.3342070 (Association for Computing Machinery, New York, NY, USA, 2019).
40. Meter, R. V. Quantum networking and internetworking. *IEEE Netw.* **26**, 59–64 (2012).
41. Aparicio, L., Van Meter, R. & Esaki, H. Protocol design for quantum repeater networks. In *Proceedings of the 7th Asian Internet Engineering Conference*, AINTEC '11, 73–80 https://doi.org/10.1145/2089016.2089029 (Association for Computing Machinery, New York, NY, USA, 2011).
42. Meter, R. V. & Touch, J. Designing quantum repeater networks. *IEEE Commun. Mag.* **51**, 64–71 (2013).
43. Meter, R. V., Ladd, T. D., Munro, W. J. & Nemoto, K. System design for a long-line quantum repeater. *IEEE/ACM Trans. Netw.* **17**, 1002–1013 (2009).
44. Pirker, A. & Dür, W. A quantum network stack and protocols for reliable entanglement-based networks. *New J. Phys.* **21**, 033003 (2019).
45. Acín, A. et al. Device-independent security of quantum cryptography against collective attacks. *Phys. Rev. Lett.* **98**, 230501 (2007).
46. Branciard, C., Cavalcanti, E. G., Walborn, S. P., Scarani, V. & Wiseman, H. M. One-sided device-independent quantum key distribution: security, feasibility, and the connection with steering. *Phys. Rev. A* **85**, 010301 (2012).
47. Scarani, V. et al. The security of practical quantum key distribution. *Rev. Mod. Phys.* **81**, 1301–1350 (2009).
48. Xu, F., Ma, X., Zhang, Q., Lo, H.-K. & Pan, J.-W. Secure quantum key distribution with realistic devices. *Rev. Mod. Phys.* **92**, 025002 (2020).
49. Pirandola, S. et al. Advances in quantum cryptography. *Adv. Optics Photon.* **12**, 1012–1236 (2020).
50. Barz, S. et al. Demonstration of blind quantum computing. *Science* **335**, 303–308 (2012).
51. Nickerson, N. H., Fitzsimons, J. F. & Benjamin, S. C. Freely scalable quantum technologies using cells of 5-to-50 qubits with very lossy and noisy photonic links. *Phys. Rev. X* **4**, 041041 (2014).
52. Lipinska, V., Murta, G. & Wehner, S. Anonymous transmission in a noisy quantum network using the *w* state. *Phys. Rev. A* **98**, 052320 (2018).
53. Khabiboulline, E. T., Borregaard, J., De Greve, K. & Lukin, M. D. Optical interferometry with quantum networks. *Phys. Rev. Lett.* **123**, 070504 (2019).

54. Shchukin, E., Schmidt, F. & van Loock, P. Waiting time in quantum repeaters with probabilistic entanglement swapping. *Phys. Rev. A* **100**, 032322 (2019).

55. Vinay, S. E. & Kok, P. Statistical analysis of quantum-entangled-network generation. *Phys. Rev. A* **99**, 042313 (2019).

56. Vardoyan, G., Guha, S., Nain, P. & Towsley, D. On the stochastic analysis of a quantum entanglement switch. *SIGMETRICS Perform. Eval. Rev.* **47**, 27–29 (2019).

57. Razavi, M., Thompson, K., Farmanbar, H., Piani, M. & Lütkenhaus, N. Physical and architectural considerations in quantum repeaters. In Arakawa, Y., Sasaki, M. & Sotobayashi, H. (eds.) https://doi.org/10.1117/12.811880 *Quantum Communications Realized II*, vol. 7236, 18–30. International Society for Optics and Photonics (SPIE, 2009).

58. Wilde, M. M. Quantum Iinformation Theory. (Cambridge University Press, 2013).

59. Pant, M. et al. Routing entanglement in the quantum internet. *npj Quantum Inform.* **5**, 25 (2019).

60. Kuzmin, V., Vasilyev, D., Sangouard, N., Dür, W. & Muschik, C. Scalable repeater architectures for multi-party states. *npj Quantum Inform.* **5**, 1–6 (2019).

61. Khatri, S., Matyas, C. T., Siddiqui, A. U. & Dowling, J. P. Practical figures of merit and thresholds for entanglement distribution in quantum networks. *Phys. Rev. Res.* **1**, 023032 (2019).

62. Varga, A. The OMNeT++ discrete event simulation system. In *Proceedings of the European Simulation Multiconference (ESM'2001)* (2001).

63. Riley, G. F. & Henderson, T. R.The ns-3 Network Simulator, 15–34 https://doi.org/10.1007/978-3-642-12331-3_2 (Springer Berlin Heidelberg, Berlin, Heidelberg, 2010).

64. Lantz, B., Heller, B. & McKeown, N. A network in a laptop: rapid prototyping for software-defined networks. In *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*, 1–6 (2010).

65. Fingerhuth, M., Babej, T. & Wittek, P. Open source software in quantum computing. *PLOS One* **13**, e0208561 (2018).

66. Netsquid website and online documentation. https://netsquid.org. Access to documentation requires registration.

67. Deutsch, D. et al. Quantum privacy amplification and the security of quantum cryptography over noisy channels. *Phys. Rev. Lett.* **77**, 2818 (1996).

68. Wehrle, K., Günes, M. & Gross, J. Modeling and Tools for Network Simulation (Springer Science & Business Media, 2010).

69. Greenberger, D. M., Horne, M. A. & Zeilinger, A. Going beyond Bell's theorem. In *Bell's Theorem, Quantum Theory and Conceptions of the Universe*, 69-72 (Springer, 1989).

70. Awschalom, D. D., Hanson, R., Wrachtrup, J. & Zhou, B. B. Quantum technologies with optically interfaced solid-state spins. *Nat. Photon.* **12**, 516–527 (2018).

71. Doherty, M. W. et al. The nitrogen-vacancy colour centre in diamond. *Phys. Rep.* **528**, 1–45 (2013).

72. Afzelius, M., Simon, C., De Riedmatten, H. & Gisin, N. Multimode quantum memory based on atomic frequency combs. *Phys. Rev. A* **79**, 052329 (2009).

73. Fleischhauer, M., Imamoglu, A. & Marangos, J. P. Electromagnetically induced transparency: optics in coherent media. *Rev. Mod. Phys.* **77**, 633 (2005).

74. Lukin, M. Colloquium: trapping and manipulating photon states in atomic ensembles. *Rev. Mod. Phys.* **75**, 457 (2003).

75. Krovi, H. et al. Practical quantum repeaters with parametric down-conversion sources. *Appl. Phys. B* **122**, 52 (2016).

76. Bennett, C. H. & Brassard, G. Quantum cryptography: public key distribution and coin tossing. *Theor. Comput. Sci.* **560**, 7–11 (2014).

77. Aaronson, S. & Gottesman, D. Improved simulation of stabilizer circuits. *Phys. Rev. A* **70**, 052328 (2004).

78. Anders, S. & Briegel, H. J. Fast simulation of stabilizer circuits using a graph-state representation. *Phys. Rev. A* **73**, 022334 (2006).

79. Steiger, D. S., Häner, T. & Troyer, M. ProjectQ: an open source software framework for quantum computing. *Quantum* **2**, 49 (2018).

80. de Oliveira Filho, J., Papp, Z., Djapic, R. & Oosteveen, J. Model-based design of self-adapting networked signal processing systems. In *Proceedings of IEEE 7th International Conference on Self-Adaptive and Self-Organizing Systems*, 41-50 (IEEE, 2013).

81. Dahlberg, A. & Wehner, S. SimulaQron—a simulator for developing quantum internet software. *Quantum Sci. Technol.* **4**, 015001 (2018).

82. Diadamo, S., Notzel, J., Zanger B., Bese, M. M., QuNetSim: A Software Framework for Quantum Networks, IEEE Transactions on Quantum Engineering https://doi.org/10.1109/TQE.2021.3092395 (2021).

83. Bartlett, B. A distributed simulation framework for quantum networks and channels. arXiv:quant-ph/1808.07047 (2018).

84. Matsuo, T. Simulation of a dynamic, RuleSet-based quantum network. arXiv:1908.10758 (2020).

85. Mailloux, L. O. et al. A modeling framework for studying quantum key distribution system implementation nonidealities. *IEEE Access* **3**, 110–130 (2015).

86. Wu, X. et al. SeQUeNCe: a customizable discrete-event simulator of quantum networks. arXiv:2009.12000 (2020).

87. Lee, Y., Bersin, E., Dahlberg, A., Wehner, S. & Englund, D. A quantum router architecture for high-fidelity entanglement flows in multi-user quantum networks. arXiv:2005.01852 (2020).

88. Kozlowski, W., Dahlberg, A. & Wehner, S. Designing a quantum network protocol. In *Proceedings of the 16th International Conference on Emerging Networking EXperiments and Technologies (CoNEXT '20)*, 16 (ACM, 2020).

89. Behnel, S. et al. Cython: the best of both worlds. *Comput. Sci. Eng.* **13**, 31–39 (2011).

90. De Raedt, K. et al. Massively parallel quantum computer simulator. *Comput. Phys. Commun.* **176**, 121–136 (2007).

91. Häner, T. & Steiger, D. S. 0.5 petabyte simulation of a 45-qubit quantum circuit. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, SC '17* https://doi.org/10.1145/3126908.3126947 (Association for Computing Machinery, New York, NY, USA, 2017).

92. Rozpędek, F. et al. Near-term quantum repeater experiments with NV centers: overcoming the limitations of direct transmission. *Phys. Rev. A* **99**, 052330 (2019).

93. Humphreys, P. C. et al. Deterministic delivery of remote entanglement on a quantum network. *Nature* **558**, 268–273 (2018).

94. Coopmans, T. et al. Replication Data for: NetSquid, a discrete-event simulation platform for quantum networks https://doi.org/10.34894/URV169 (2021).

95. Coopmans, T. et al. Simulation Code for: NetSquid, a discrete-event simulation platform for quantum networks https://doi.org/10.34894/DU3FTS (2021).

Author contributions

T.C. realised the NV repeater chain and the quantum switch simulations. R.K., L.W. realised the benchmarking simulations. D.M., J.R. realised the atomic ensembles simulations. R.K. and J.O. designed NetSquid's software architecture and R.K. led its software development. T.C, A.D, R.K., D.M., L.N., J.O., M.P., F.R, J.R., M.S., A.T., L.W., and S.W designed use case driven architectures, and contributed to the development of NetSquid and the modelling libraries used in the simulations. W.J., D.P., A.T. contributed to the optimal execution of simulations on computing clusters. T.C., D.E., R.K., D.M., and S.W. wrote the manuscript. All authors revised the manuscript. D.E. and S.W. conceived and supervised the project.

Competing interests

The authors declare no competing interests.

Additional information

Supplementary information The online version contains supplementary material available at https://doi.org/10.1038/s42005-021-00647-8.

Correspondence and requests for materials should be addressed to D.E. or S.W.

Peer review information *Communications Physics* thanks the anonymous reviewers for their contribution to the peer review of this work. Peer reviewer reports are available.

Reprints and permission information is available at http://www.nature.com/reprints

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.