IEEE INFOCOM 2025 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS) | 979-8-3315-4370-9/25/$31.00 ©2025 IEEE | DOI: 10.1109/INFOCOMWKSHPS65812.2025.11152936

Extended Abstract: A Modular Quantum Network Architecture for Integrating Network Scheduling with Local Program Execution.

Thomas R. Beauchamp, Hana Jirovská, Scarlett Gauthier, Stephanie Wehner
QuTech, Delft University of Technology, Delft, The Netherlands
Kavli Institute of Nanoscience, Delft University of Technology, Delft, The Netherlands
Quantum Computer Science, EEMCS, Delft University of Technology, Delft, The Netherlands
Correspondence email: *t.r.beauchamp@tudelft.nl*

***Abstract*—We propose an architecture for scheduling network operations enabling the end-to-end generation of entanglement according to user demand. The main challenge solved by this architecture is to allow for the integration of a network schedule with the execution of quantum programs running on processing end nodes in order to realise quantum network applications. A key element of this architecture is the definition of an entanglement packet to meet application requirements on near-term quantum networks where the lifetimes of the qubits stored at the end nodes are limited. Our architecture is fully modular and hardware agnostic, and defines a framework for further research on specific components that can now be developed independently of each other. In order to evaluate our architecture, we realise a proof of concept implementation on a simulated 6-node network in a star topology. We show our architecture facilitates the execution of quantum network applications, and that robust admission control is required to maintain quality of service.**

I. Introduction

The realisation of a quantum internet will enable the use of new networked applications beyond what is possible with the current classical internet. Such applications include the ability to perform verifiably secure secret sharing [1], [2], secure remote computation [3]–[5] and securely electing a leader [6], amongst many others [7]. The aim of any quantum network architecture therefore should be to ensure that these applications can be successfully executed.

To execute a quantum network application, it is necessary to establish *entangled links* between the *end nodes* — the quantum devices accessible to users. However, such links are difficult to produce and doing so requires the use of a limited quantity of resources in the quantum network [8]–[11]. Furthermore, at present each link has a limited usable lifetime as quantum memories experience decoherence over time, reducing the quality of stored entangled links [12]. Therefore, there exist two interacting scheduling problems which must be solved: firstly how should the limited network resources be assigned to pairs of users to allow them to generate entangled links (the *network scheduling problem*), and secondly how to schedule the execution of quantum applications to efficiently use any entangled links which are generated (the *local scheduling problem*).

The first execution environment for end nodes, QNodeOS [13], allows the second of these problems to be solved. *Qoala* [14], an upgrade to QNodeOS, offers an improved end node execution environment. It breaks an application into a program at each node, which is then further subdivided into blocks of instructions that can be scheduled for execution at runtime. Furthermore, Qoala comes equipped with a compiler which can provide advice about the quantity and quality of entangled links which are required. Although these application execution environments successfully address the local scheduling problem, they require the existence of a network schedule computed by a third party to supply allocations of time during which entanglement generation can take place.

However, there presently does not exist a network architecture which can produce schedules which are compatible with such execution environments. Without such an architecture, the network scheduling problem cannot be solved in a manner which still allows the local scheduling problem to be solved effectively. In this work we therefore propose such an architecture to unify the approach to the network and local scheduling problems.

A. Network Model

There are many different models for how a quantum network should operate. In the literature, one common example is what we will call a *pre-loaded* network, where there is a high probability that entangled links are immediately available to an application (e.g. [15], [16]). Such networks rely on continuously generating and buffering entanglement between each pair of network components. However, restrictions on achievable entanglement generation rates, buffer lifetimes and buffer capacities limit the possibility of near-term implementations of such networks. For example, experiments on leading hardware [17] have realised a three-node network, on which

This work was supported by the Quantum Internet Alliance (QIA). QIA has received funding from the European Union's Horizon Europe research and innovation programme under grant agreement No. 101102140. SW also acknowledges funding from the NWO VICI grant.

they reported memory lifetimes of 11ms and generation rates of one end-to-end link every 40s. Therefore, we instead consider a *generate-when-requested* network. In such networks, we do not assume that end-to-end entangled links can be stored between any two scheduled periods of time. This type of network is implementable with the technological maturity of current devices and those that will exist in the coming years.

1) Network Components: We assume that a quantum network is constructed from the following basic components: *end nodes*, *metropolitan hubs*, *repeater chains* and *junction nodes*. *End nodes* are devices controlled by the users of the network. They could either be a processing node with memory capabilities, such as in [9], [18], or a device which is capable of preparing/measuring single photons, such as in [19], [20]. The remaining components form the internals of the network. *Metropolitan hubs* are devices which allow pairs of nodes located close together to generate entangled links. Examples of such devices include [21], [22]. A metropolitan hub will typically have many end nodes connected to it. The final two devices are used for generating long-distance entanglement. Repeater chains are a linear chain of *repeater nodes* (e.g. [11], [23]). We treat repeater chains as a single component. Finally, junction nodes provide an interface between end nodes and repeater chains, as well as between multiple repeater chains. This removes the need to have a repeater chain between every pair of metropolitan hubs. A junction node may be as simple as a *border node*, which mediates entanglement between heterogeneous network links, possibly taking advantage of a quantum memory, or a junction node may be as advanced as a full quantum router [24], [25].

2) Software Defined Network: We assume that the quantum network follows the general architecture of a *software defined network* (SDN) (e.g. [26]). In particular, we assume that the network is overseen by a (logically) central controller, similar to for example [9], [27]–[29]. Such a controller has the authority to compute and enforce *network schedules*, which dictate when entangled links can be generated for particular pairs of end nodes. We also assume that each internal component of the network has a control API which can be used for installing network schedules.

II. Related Work

There are several other authors who have proposed architectures for quantum networks, however they each have a different scope for the architecture than ourselves. In particular, none of them explicitly consider the influence of requirements of executing local applications on the network.

For instance, Skrzypczyk *et al.* in [29] propose an architecture around using TDMA schedules to generate good quality entanglement. Whilst we build on their ideas about scheduling, they do not consider how their scheduling will impact the ability of end nodes in the network to execute the applications requiring this entanglement, nor how the demands are generated from the applications.

Cicconetti *et al.* in [30] and Gu *et al.* in [15] consider the problem of scheduling requests for entanglement generation in a quantum network. However in both cases they consider a continuous distribution network rather than a generate-when-requested network. Furthermore, they do not consider how the end nodes use the entanglement which is generated, and also consider a system where every edge in the network is in constant communication with the central controller, whereas we only require sporadic communications. A disadvantage of this compared to our approach is that due to latency when communicating between the edges and the central controller, there is an inherent loss in the quality of entangled links which can be created. Furthermore, they only consider requests for single entangled links, whereas we consider requests for packets of entangled links.

Van Meter *et al.* in [31] propose an architecture which focuses on routing of entanglement across many smaller networks, and the protocols required to do so. This again does not account for the local nodes, nor does it consider the interaction of scheduling entanglement generation on the network with executing the applications on end nodes. Furthermore, the scope of our work is to provide an architecture for a single quantum network which can be centrally controlled, rather than for a quantum internetwork.

Our architecture also has similarities to a software-defined network [26], [32], [33]. In particular, we see our work as a method of facilitating the implementation of a quantum SDN. There have been several examples of prior work on defining a quantum SDN. For example in [28], Kozlowski, Kuipers and Wehner give an implementation of a quantum SDN using the P4 language. However, this implementation only focuses on the network aspects, and does not consider the execution of applications on the end nodes. There has also been a quantum SDN proposed and demonstrated by Yang and Cui in [34]. However again this work does not explicitly consider scheduling at the end nodes in the network, and they focus on demands being registered via a web-interface. In contrast, we create a fully-autonomous architecture, where the end nodes, rather than the users themselves, submit the demands in response to users wishing to run specific applications.

III. Architecture

A visual overview of the architecture is given by Figures 1, 2, 3. The flow of information through the stages of the architecture is illustrated in Figure 1. Example timings for computing, distributing and executing network schedules are illustrated in Figure 2. An interaction diagram covering all interactions between end nodes and the central controller is illustrated in Figure 3.

A. Application Sessions

When executing a quantum network application, it will often be required for many instances of that application to be executed, from which a reliable outcome can be extracted. We capture this in an *application session*

$$S = (\mathcal{N}, \texttt{App}, N^{inst}, t^{\texttt{expiry}}), \tag{1}$$

Fig. 1: Flow of information through the stages of the architecture. The processes of 'Network Capability Update' and 'Capability Negotiation' allow the nodes to gather enough information to be able to submit a unified demand. These demands are then used to construct a central network schedule, which in turn is used when constructing the local node schedules.

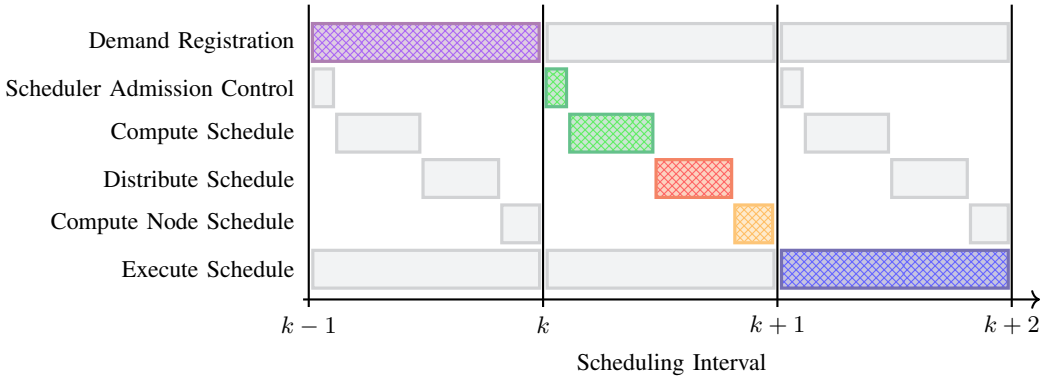


Fig. 2: Example of the timings for computing, distributing and executing the kth network schedule (in colour and hatched). In grey and unhatched are the corresponding operations for preceding and subsequent network schedules.

where $\mathcal{N}$ is the set of nodes in the application and N^{inst} is the number of instances of the application `App` they wish to execute, by time $t^{\texttt{expiry}}$. The *minimal service* which the nodes $\mathcal{N}$ need to obtain from the network is then that they generate sufficient entanglement to execute N^{inst} instances of `App`.

B. Demand Format

In order for the network to be able to provide minimal service, the central controller must be informed about what entanglement needs to be generated. Therefore, the nodes need to submit a suitable demand to the network which accurately captures the requirements on the generated entangled links.

When executing a quantum network application of the create-and-keep type [35], there needs to be a number s of entangled pairs co-existing in the quantum memory, each with some minimum fidelity. One could achieve this by strictly controlling the jitter between each successful end-to-end entangled link generation.

To overcome the difficulty of implementing strict jitter control, however, we instead introduce a notion of *packets of entanglement*. These are a tuple $p = (w, s, F)$, where w is a cut-off or *window*, s is the number of pairs required to co-exist and F is the initial fidelity at which links are generated. By correctly setting w based on local system parameters, given that new links are generated at fidelity F, if no link is older than age w, all s links will be of sufficiently high fidelity to be used. Therefore, the generation of a packet of entanglement directly enables the execution of one instance of the application `App`.

End nodes will submit demands to the network controller for the generation of these packets of entanglement. These demands have the following form:

$$\mathcal{D} = \left(\{(w, s, F, R)\}_{p\in\mathcal{P}}; t^{\texttt{minsep}}, t^{\texttt{expiry}}; N^{\texttt{inst}}\right). \quad (2)$$

For each packet $p = (w, s, F) \in \mathcal{P}$, where $\mathcal{P}$ is a list of suitable packets, we associate a minimum requested rate of generation R_p. Based on local requirements, end nodes specify a minimum separation between attempts to generate packets, $t^{\texttt{minsep}}$. This parameter ensures there is sufficient time for end nodes to perform local gates and measurements once the packet is generated. The expiry time and number of instances from the application session are carried forward into the demand.

C. Network Capabilities Update and Capabilities Negotiation

For each application session, one demand should be submitted to the central controller. However, it is not possible for only one node to construct the demand by itself, as it has insufficient information. For example, to calculate w it may be required to know about the capabilities of the other node. Furthermore, the end nodes are unaware of the internals of the network connecting them, so cannot know what fidelity and rate of end-to-end entanglement it is possible to achieve.

The first two phases of our architecture (see Figure 1 and **A**, **B** in Figure 3) facilitate the exchange and delivery of this necessary information. In the network capabilities update, the end nodes query the central controller to obtain essential information about the network. During the capabilities negotiation, the end nodes of an application session exchange information to formulate a unified demand.

Once these phases are concluded, the end nodes submit the demand to the network controller for registration (**C** in Figure 3).

D. Packet Generation Tasks

When a demand is received from end nodes, it passes through an initial registration phase. This is to screen out malicious or obviously infeasible demands.

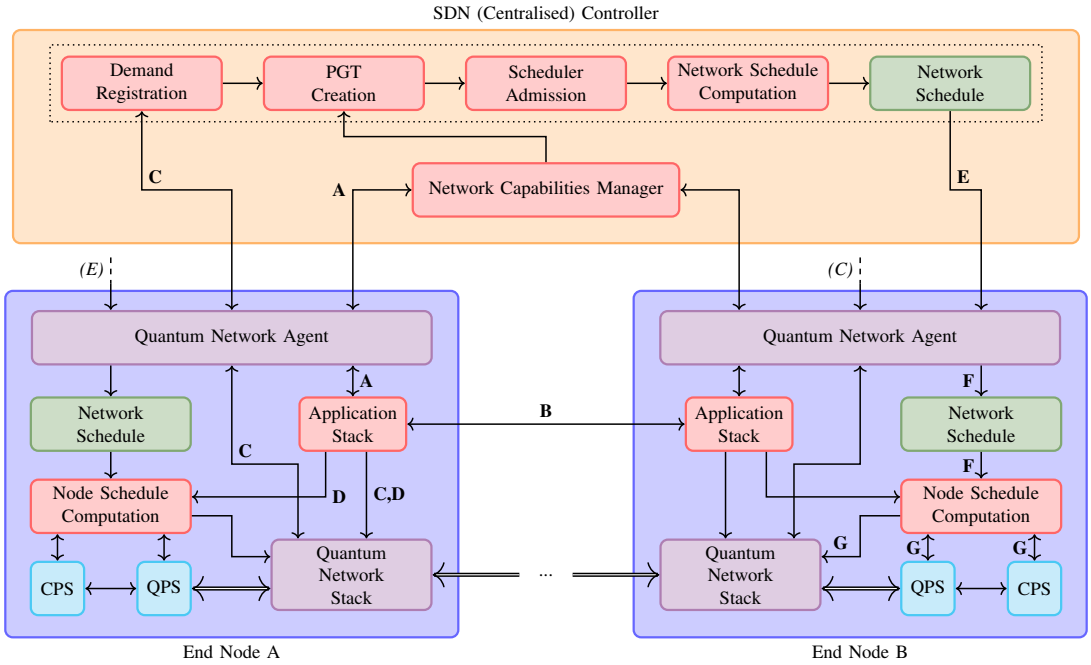


Fig. 3: Interaction Diagram for our proposed quantum network architecture. Single stroke arrows represent purely classical interactions and double-stroke arrows represent quantum interactions. CPS = Classical Processing System. QPS = Quantum Processing System. The labelled interactions are as follows: **A**: Network Capability Update; **B**: Capability Negotiation; **C**: Demand Registration; **D**: Session Initialisation; **E**: Network Schedule Distribution **F**: Input of network schedule into the local schedule; **G**: Execution of the schedule(s).

Once a demand has passed this check it is converted into the scheduler's internal representation, called *packet generation tasks* (PGTs) and added to the *demand queue*. These PGTs are the sources of jobs, called *packet generation attempts* (PGAs), which the network scheduler will be scheduling. PGTs have the form

$$\tau = (E, R^{\texttt{attempt}}, \rho, t^{\texttt{minsep}}, t^{\texttt{expiry}}) \tag{3}$$

where E is the duration of each PGA, $R^{\texttt{attempt}}$ is the rate at which PGAs are scheduled, ρ is the set of required network resources and both $t^{\texttt{minsep}}$ and $t^{\texttt{expiry}}$ are as in the demand. During this process, the central controller decides which packet p^* of the submitted packets $\mathcal{P}_d$ will be generated, and along which path in the network.

1) Packet Generation Attempts: During a scheduled PGA, the end nodes involved will make many attempts to generate entanglement between themselves, possibly using resources at internal network components. Each one of these attempts to generate end-to-end entanglement requires implementing some intermediary protocols, in particular performing many physical attempts to generate entanglement between neighbouring network components. Each of these physical attempts will only succeed with some very low probability, resulting in some average rate of successfully generating end-to-end entangled links. When a success occurs, the generated entangled link will be stored for up to time w before being discarded in order to mitigate decoherence due to noisy memories. The probability that an individual PGA succeeds and the maximum storage time w are not simply modelling parameters, they are inherited from the physical capabilities of the network components.

As we require s co-existing end-to-end entangled links to obtain a packet of entanglement, one cannot guarantee that a packet will be generated in a given PGA without setting $E = \infty$. Therefore, the network sets the value of E such that there is a known probability $p^{\texttt{packet}}$ of a packet being generated in a given PGA. The value of $p^{\texttt{packet}}$ may be a static value or can be determined by the central controller for each demand individually.

Once the value of $p^{\texttt{packet}}$ is known, then $R^{\texttt{attempt}}$ can be set to $R_{p^*}/p^{\texttt{packet}}$ if $R_{p^*} > 0$. We allow nodes to submit demands with $R_{p^*} = 0$ to indicate that they can accept any rate of packet generation. In this case, the central controller sets $R^{\texttt{attempt}}$ such that before the expiry time elapses, sufficiently many PGAs will be scheduled to ensure that $N^{\texttt{inst}}$ packets are generated with high probability.

E. Network Schedule Computation

The process of computing the network schedule is broken into three sub-phases, *admission control*, *schedule computation* and *schedule distribution*. Possible timings for these phases are shown in Figure 2.

1) Admission Control: Before the network schedule can be computed, first the central controller must determine which PGTs are to be scheduled. Any PGTs from the previous network schedule that have neither expired nor been terminated are automatically carried forward. Then, any PGTs which

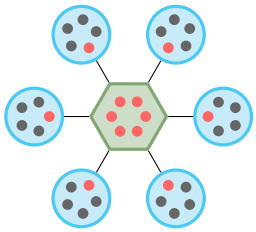


Fig. 4: Example of a star-topology network with 6 nodes. The outer circles represent the end nodes and the central hexagon a central junction node.

satisfy conditions for acceptance are admitted. Such a rule could be that no resource is over-utilised.

The precise management of the demand queue will depend on the specific implementation of this architecture.

2) Schedule Computation: Once the PGTs to be scheduled are determined, then the network schedule can be computed. This process entails assigning a series of start times to each PGT, from each of which a PGA is executed without pre-emption. The network schedule computed may be specified in either continuous or discrete time. In the latter case, the central controller determines the time-slots, which are required to be the same for all resources in the network schedule.

3) Schedule Distribution: Once the schedule has been computed it is distributed to the network components (**E** in Figure 3). Each network component only receives the portion of the schedule relevant to it. The transmitted network schedule specifies the start and end times of each scheduled PGA as well as an identifier for the demand from which the corresponding PGT was derived. The identifier ensures components of the network stack and local application runtime environments on end nodes can correctly interpret the network schedule.

It is critical that the network schedule is computed 'on time', so that it can be distributed to the network components before coming into effect. However, it is the role of an implementation to determine how to handle the case where a schedule is not computed on time.

IV. Implementation and Evaluation

A. Implementation

In order to perform an initial evaluation of our architecture, we created a simple implementation of our architecture on a 6-end node star-topology network, as illustrated in Figure 4. We nominate one of the end nodes to be a 'server' and the other five are 'client' nodes. Each client node submits new demands following a Poisson process with rate parameter λ, which is the same for all client end nodes. In our evaluation we refer to λ as the *session renewal rate*.

The demands submitted by client end nodes all have the same format:

$$\mathcal{D} = ((5ms, 2, 0.95, R); 300\mu s, t^{\texttt{submit}} + 4 \text{ days}; 100), \quad (4)$$

where R is the requested rate of packet generation and $t^{\texttt{submit}}$ is the time at which the demand is submitted. The ranges of λ and R we simulated were chosen to demonstrate a wide range of behaviours without requiring excessive simulation time.

The PGTs these demands are converted into by the central controller are given by

$$\tau = (81.4s, 5R, \rho, 300\mu s, t^{\texttt{submit}} + 4 \text{ days}), \quad (5)$$

where ρ is a list containing the paths between each end node involved and the central junction node. Each of these PGTs therefore has a resource utilisation of $E_\tau R_\tau^{\texttt{attempt}} \approx 400R$.

To implement admission control we implement two rules, the first of which depends upon a measure of utilisation of the network resources. Specifically, we define the *utilisation of a resource r by all tasks $\{\tau\}$* to be

$$\mathcal{U}_r := \sum_{\tau : r \in \rho_\tau} E_\tau R_\tau^{\texttt{attempt}}. \quad (6)$$

A resource r is said to be over-utilised if $\mathcal{U}_r > 1$. In our implementation, the rules of admission control ensure that no resource is over-utilised and that the estimated time to compute the schedule is less than half a scheduling interval.

To implement a network scheduler, we adapt (in Algorithm 1) an *earliest deadline first* (EDF) scheduler from real-time systems [36]. Based on the value of $R_\tau^{\texttt{attempt}}$ for each PGT, a series of release times and deadlines can be defined such that if a PGA is scheduled between each release time and the subsequent deadline, PGAs are scheduled at rate $R_\tau^{\texttt{attempt}}$. The scheduler then constructs the schedule by choosing at each possible time to schedule the PGA with the next deadline, as in Algorithm 1.

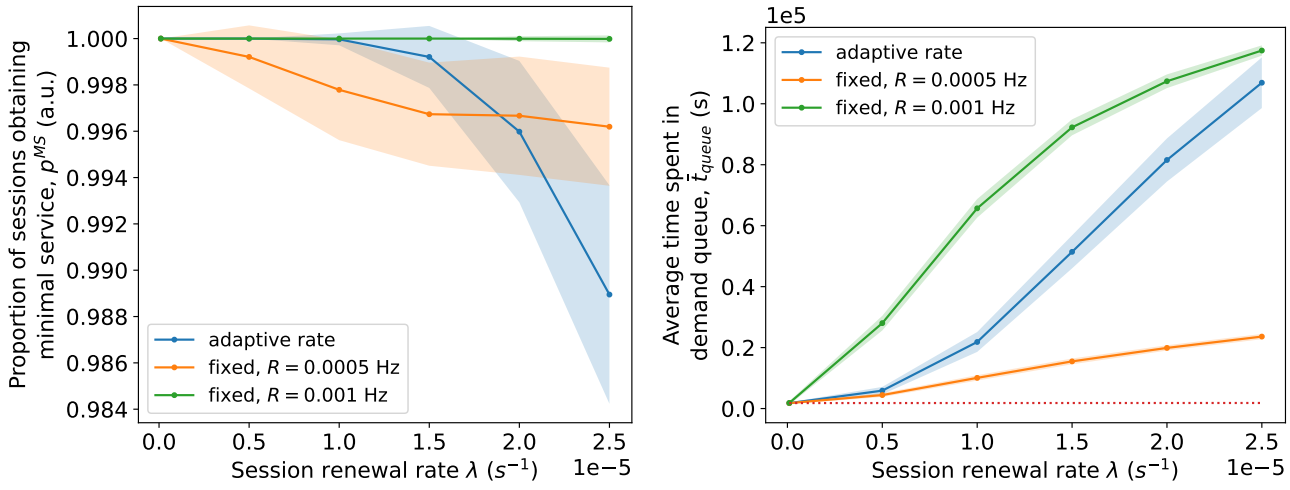
Algorithm 1: EDF Scheduling Algorithm. A PGA is said to be *eligible* if it has been released and the required resources are available.

Input : Desired schedule length, PGTs to be scheduled
Output: Scheduled PGAs

```
Set the initial decision time t* to the time at the start of the schedule;
while t* is less than the end time of the schedule do
    while there are eligible PGAs do
        Schedule the eligible PGA with the earliest deadline;
    end
    Update t* to the next time either a PGA is released, a PGA completes execution, or the end of the scheduling interval, whichever is earlier;
end
```

B. Evaluation

1) Metrics: When multiple PGTs require the same network resource, there exists contention for access to that resource. As the session renewal rate, λ, defines how frequently PGTs for an application session are added to the demand queue, it affects the amount of contention for network resources. Moreover, the



(a) (b)

Fig. 5: Results from simulations of a test create-and-keep type application with one server and 5 possible clients on a six-node star network. (a) Proportion of demands which obtain minimal service. (b) Average time a demand spends in the demand queue.

rate associated with a PGT impacts the amount of contention. The PGT rate determines how frequently PGAs must occupy contended resources in order to provide the PGT with minimal service.

We evaluate two metrics to assess the effect of increasing resource contention. The first is the proportion of submitted demands which obtain minimal service from the network, p^{MS}. End nodes can use such a metric to estimate the likelihood of a submitted demand obtaining minimal service, and a central controller can use this to monitor whether the amount of resource contention inhibits the probability of application sessions succeeding. The second metric we evaluate is the average time a demand spends in the demand queue, $\bar{t}_{queue}$. This corresponds to the latency between submitting a demand and being accepted for service.

2) Results: The results of our simulations can be seen in Figure 5. We focus on the effects of changing λ and R on the considered metrics.

The most important criteria that a network architecture needs to meet is that demands are capable of obtaining minimal service. From the results in Figure 5a, we can see that over all parameter regimes simulated over 98% of submitted demands obtained minimal service.

From Figure 5a it is clear that the session renewal rate λ and the requested rate of packet generation R have a large impact on the proportion of sessions which obtain minimal service. In particular, as λ increases, the proportion of sessions which obtain minimal service may only decrease. This is the expected behaviour, as an increase in λ directly translates into increased contention for network resources. We do note that in the case where $R = 0.001$Hz in Figure 5, $p^{MS} = 1$ for all λ, with only a small non-zero standard deviation for $\lambda > 2 \times 10^{-5}$. This indicates that there are parameter regimes where it is possible for all nodes to obtain minimal service, regardless of the session renewal rate. Figure 5b confirms that the average time a demand spends in the queue, $\bar{t}_{queue}$ always increases as λ increases. An extension of the duration of time that demands spend queued is direct evidence of increased contention for network resources.

The effect of R is more subtle, as we consider two values of a fixed rate as well as an adaptive rate. For the `CKA` test application considered operated in client-server mode, the lower fixed rate results in lower p^{MS}. However, for other applications we have observed that lower fixed rates correspond to a higher value of p^{MS}. In all cases, we observed that a lower fixed rate results in lower $\bar{t}_{queue}$. This is expected because a lower fixed rate again translates to a lower amount of contention for network resources.

References

[1] A. K. Ekert, "Quantum cryptography based on Bell's theorem," *Physical Review Letters*, vol. 67, no. 6, pp. 661–663, Aug. 1991. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevLett.67.661

[2] C. H. Bennett and G. Brassard, "Quantum cryptography: Public key distribution and coin tossing," *Theoretical Computer Science*, vol. 560, pp. 7–11, Dec. 2014. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0304397514004241

[3] P. Arright and L. Salvail, "Blind quantum computation," *International Journal of Quantum Information*, vol. 04, no. 05, pp. 883–898, 2006.

[4] A. Broadbent, J. Fitzsimons, and E. Kashefi, "Universal Blind Quantum Computation," in *2009 50th Annual IEEE Symposium on Foundations of Computer Science*. Atlanta, Georgia, USA: IEEE, Oct. 2009, pp. 517–526.

[5] A. M. Childs, "Secure assisted quantum computation," *Quantum Information and Computation*, vol. 5, no. 6, 2005, arXiv:quant-ph/0111046. [Online]. Available: http://arxiv.org/abs/quant-ph/0111046

[6] S. Tani, H. Kobayashi, and K. Matsumoto, "Exact Quantum Algorithms for the Leader Election Problem," *ACM Transactions on Computation Theory*, vol. 4, no. 1, pp. 1–24, Mar. 2012. [Online]. Available: https://dl.acm.org/doi/10.1145/2141938.2141939

[7] S. Wehner, D. Elkouss, and R. Hanson, "Quantum internet: A vision for the road ahead," *Science*, Oct. 2018, publisher: American Association for the Advancement of Science. [Online]. Available: https://www.science.org/doi/10.1126/science.aam9288

[8] W. Dür, H.-J. Briegel, J. I. Cirac, and P. Zoller, "Quantum repeaters based on entanglement purification," *Physical Review A*, vol. 59, no. 1, pp. 169–181, Jan. 1999, publisher: American Physical Society. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevA.59.169

[9] M. Pompili, S. L. N. Hermans, S. Baier, H. K. C. Beukers, P. C. Humphreys, R. N. Schouten, R. F. L. Vermeulen, M. J. Tiggelman, L. dos Santos Martins, B. Dirkse, S. Wehner, and R. Hanson, "Realization of a multinode quantum network of remote solid-state qubits," *Science*, vol. 372, no. 6539, pp. 259–264, Apr. 2021, publisher: American Association for the Advancement of Science. [Online]. Available: https://www.science.org/doi/10.1126/science.abg1919

[10] Á. G. Iñesta, G. Vardoyan, L. Scavuzzo, and S. Wehner, "Optimal entanglement distribution policies in homogeneous repeater chains with cutoffs," *npj Quantum Information*, vol. 9, no. 1, pp. 1–7, May 2023, number: 1 Publisher: Nature Publishing Group. [Online]. Available: https://www.nature.com/articles/s41534-023-00713-9

[11] C. Cabrillo, J. I. Cirac, P. García-Fernández, and P. Zoller, "Creation of entangled states of distant atoms by interference," *Physical Review A*, vol. 59, no. 2, pp. 1025–1033, Feb. 1999. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevA.59.1025

[12] C. E. Bradley, S. W. de Bone, P. F. W. Moller, S. Baier, M. J. Degen, S. J. H. Loenen, H. P. Bartling, M. Markham, D. J. Twitchen, R. Hanson, D. Elkouss, and T. H. Taminiau, "Robust quantum-network memory based on spin qubits in isotopically engineered diamond," Nov. 2021, arXiv:2111.09772 [cond-mat, physics:quant-ph]. [Online]. Available: http://arxiv.org/abs/2111.09772

[13] C. D. Donne, M. Iuliano, B. van der Vecht, G. M. Ferreira, H. Jirovská, T. van der Steenhoven, A. Dahlberg, M. Skrzypczyk, D. Fioretto, M. Teller, P. Filippov, A. R.-P. Montblanch, J. Fischer, B. van Ommen, N. Demetriou, D. Leichtle, L. Music, H. Ollivier, I. te Raa, W. Kozlowski, T. Taminiau, P. Pawełczak, T. Northup, R. Hanson, and S. Wehner, "Design and demonstration of an operating system for executing applications on quantum network nodes," 2024. [Online]. Available: https://arxiv.org/abs/2407.18306

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

[15] H. Gu, R. Yu, Z. Li, X. Wang, and F. Zhou, "Esdi: Entanglement scheduling and distribution in the quantum internet," *2023 32nd International Conference on Computer Communications and Networks (ICCCN)*, pp. 1–10, 2023.

[16] A. Pirker and W. Dür, "A quantum network stack and protocols for reliable entanglement-based networks," *New Journal of Physics*, vol. 21, no. 3, p. 033003, 2019, publisher: IOP Publishing. [Online]. Available: https://dx.doi.org/10.1088/1367-2630/ab05f7

[17] M. Pompili, C. Delle Donne, I. te Raa, B. van der Vecht, M. Skrzypczyk, G. Ferreira, L. de Kluijver, A. J. Stolk, S. L. N. Hermans, P. Pawełczak, W. Kozlowski, R. Hanson, and S. Wehner, "Experimental demonstration of entanglement delivery using a quantum network stack," *npj Quantum Information*, vol. 8, no. 1, pp. 1–10, Oct. 2022, number: 1 Publisher: Nature Publishing Group. [Online]. Available: https://www.nature.com/articles/s41534-022-00631-2

[18] V. Krutyanskiy, M. Galli, V. Krcmarsky, S. Baier, D. A. Fioretto, Y. Pu, A. Mazloom, P. Sekatski, M. Canteri, M. Teller, J. Schupp, J. Bate, M. Meraner, N. Sangouard, B. P. Lanyon, and T. E. Northup, "Entanglement of trapped-ion qubits separated by 230 meters," *Phys. Rev. Lett.*, vol. 130, p. 050803, Feb. 2023. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevLett.130.050803

[19] P. Drmota, D. Nadlinger, D. Main, B. Nichol, E. Ainley, D. Leichtle, A. Mantri, E. Kashefi, R. Srinivas, G. Araneda, C. Ballance, and D. Lucas, "Verifiable blind quantum computing with trapped ions and single photons," *Physical Review Letters*, vol. 132, no. 15, Apr. 2024. [Online]. Available: http://dx.doi.org/10.1103/PhysRevLett.132.150604

[20] R. C. Berrevoets, T. Middelburg, R. F. L. Vermeulen, L. D. Chiesa, F. Broggi, S. Piciaccia, R. Pluis, P. Umesh, J. F. Marques, W. Tittel, and J. A. Slater, "Deployed measurement-device independent quantum key distribution and bell-state measurements coexisting with standard internet data and networking equipment," *Communications Physics*, vol. 5, no. 1, p. 186, 2022. [Online]. Available: https://doi.org/10.1038/s42005-022-00964-6

[21] G. Vardoyan, S. Guha, P. Nain, and D. Towsley, "On the Stochastic Analysis of a Quantum Entanglement Distribution Switch," *IEEE Transactions on Quantum Engineering*, vol. 2, pp. 1–16, 2021.

[22] S. Gauthier, G. Vardoyan, and S. Wehner, "An architecture for control of entanglement generation switches in quantum networks," *IEEE Transactions on Quantum Engineering*, vol. 4, pp. 1–17, 2023.

[23] C. Simon, H. de Riedmatten, M. Afzelius, N. Sangouard, H. Zbinden, and N. Gisin, "Quantum Repeaters with Photon Pair Sources and Multimode Memories," *Phys. Rev. Lett.*, vol. 98, p. 190503, May 2007.

[24] Y. Lee, E. Bersin, A. Dahlberg, S. Wehner, and D. Englund, "A quantum router architecture for high-fidelity entanglement flows in quantum networks," *npj Quantum Information*, vol. 8, no. 1, pp. 1–8, 2022, publisher: Nature Publishing Group. [Online]. Available: https://www.nature.com/articles/s41534-022-00582-8

[25] G. Vardoyan, S. Guha, P. Nain, and D. Towsley, "On the exact analysis of an idealized quantum switch," *Performance Evaluation*, vol. 144, p. 102141, 2020.

[26] D. Kreutz, F. M. V. Ramos, P. E. Veríssimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-Defined Networking: A Comprehensive Survey," *Proceedings of the IEEE*, vol. 103, no. 1, pp. 14–76, Jan. 2015, conference Name: Proceedings of the IEEE.

[27] V. Martin, J. P. Brito, L. Ortiz, R. B. Mendez, J. S. Buruaga, R. J. Vicente, A. Sebastián-Lombraña, D. Rincon, F. Perez, C. Sanchez, M. Peev, H. H. Brunner, F. Fung, A. Poppe, F. Fröwis, A. J. Shields, R. I. Woodward, H. Griesser, S. Roehrich, F. De La Iglesia, C. Abellan, M. Hentschel, J. M. Rivas-Moscoso, A. Pastor, J. Folgueira, and D. R. Lopez, "MadQCI: a heterogeneous and scalable SDN QKD network deployed in production facilities." [Online]. Available: http://arxiv.org/abs/2311.12791

[28] W. Kozlowski, F. Kuipers, and S. Wehner, "A p4 data plane for the quantum internet," in *Proceedings of the 3rd P4 Workshop in Europe*. ACM, 2020, pp. 49–51. [Online]. Available: https://dl.acm.org/doi/10.1145/3426744.3431321

[29] M. Skrzypczyk and S. Wehner, "An Architecture for Meeting Quality-of-Service Requirements in Multi-User Quantum Networks," Nov. 2021, arXiv:2111.13124 [quant-ph]. [Online]. Available: http://arxiv.org/abs/2111.13124

[30] C. Cicconetti, M. Conti, and A. Passarella, "Request Scheduling in Quantum Networks," *IEEE Transactions on Quantum Engineering*, vol. 2, pp. 2–17, 2021. [Online]. Available: https://ieeexplore.ieee.org/document/9461156/

[31] R. Van Meter, R. Satoh, N. Benchasattabuse, T. Matsuo, M. Hajdušek, T. Satoh, S. Nagayama, and S. Suzuki, "A Quantum Internet Architecture," in *2022 IEEE International Conference on Quantum Computing and Engineering (QCE)*, Sep. 2022, pp. 341–352, arXiv:2112.07092 [quant-ph]. [Online]. Available: http://arxiv.org/abs/2112.07092

[32] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: enabling innovation in campus networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, pp. 69–74, Mar. 2008. [Online]. Available: https://dl.acm.org/doi/10.1145/1355734.1355746

[33] J. M. Halpern, R. HAAS, a. doria, L. Dong, W. Wang, H. M. Khosravi, J. H. Salim, and R. Gopal, "Forwarding and Control Element Separation (ForCES) Protocol Specification," Internet Engineering Task Force, Request for Comments RFC 5810, Mar. 2010, num Pages: 124. [Online]. Available: https://datatracker.ietf.org/doc/rfc5810

[34] Z. Yang and C. Cui, "Reconfigurable Quantum Internet Service Provider," May 2023, arXiv:2305.09048 [quant-ph]. [Online]. Available: http://arxiv.org/abs/2305.09048

[35] A. Dahlberg, M. Skrzypczyk, T. Coopmans, L. Wubben, F. Rozpędek, M. Pompili, A. Stolk, P. Pawełczak, R. Knegjens, J. d. O. Filho, R. Hanson, and S. Wehner, "A Link Layer Protocol for Quantum Networks," in *Proceedings of the ACM Special Interest Group on Data Communication*, Aug. 2019, pp. 159–173, arXiv:1903.09778 [quant-ph]. [Online]. Available: http://arxiv.org/abs/1903.09778

[36] C. L. Liu and J. W. Layland, "Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment," *Journal of the ACM*, vol. 20, no. 1, pp. 46–61, Jan. 1973. [Online]. Available: https://dl.acm.org/doi/10.1145/321738.321743