IEEE Transactions on Quantum Engineering

Received January 20, 2022; revised April 1, 2022; accepted April 4, 2022; date of publication April 19, 2022; date of current version May 13, 2022.

Digital Object Identifier 10.1109/TQE.2022.3168784

Efficient Quantum Network Communication Using Optimized Entanglement Swapping Trees

MOHAMMAD GHADERIBANEH, CAITAO ZHAN, HIMANSHU GUPTA, AND C. R. RAMAKRISHNAN

Department of Computer Science, Stony Brook University, Stony Brook, NY 11790 USA

Corresponding author: Mohammad Ghaderibaneh (e-mail: mghaderibane@cs.stonybrook.edu).

This work was supported in part by the National Science Foundation under Award FET-2106447 and Award CNS-2128187 and in part by a Cisco industry grant.

ABSTRACT Quantum network communication is challenging, as the no-cloning theorem in the quantum regime makes many classical techniques inapplicable; in particular, the direct transmission of qubit states over long distances is infeasible due to unrecoverable errors. For the long-distance communication of unknown quantum states, the only viable communication approach (assuming local operations and classical communications) is the teleportation of quantum states, which requires a prior distribution of the entangled pairs (EPs) of qubits. The establishment of EPs across remote nodes can incur significant latency due to the low probability of success of the underlying physical processes. The focus of our work is to develop efficient techniques that minimize EP generation latency. Prior works have focused on selecting entanglement *paths*; in contrast, we select *entanglement swapping trees*—a more accurate representation of the entanglement generation structure. We develop a dynamic programming algorithm to select an optimal swapping tree for a single pair of nodes, under the given capacity and fidelity constraints. For the general setting, we develop an efficient iterative algorithm to compute a set of swapping trees. We present simulation results, which show that our solutions outperform the prior approaches by an order of magnitude and are viable for long-distance entanglement generation.

INDEX TERMS Quantum communications, quantum networks (QNs).

I. INTRODUCTION

Fundamental advances in physical sciences and engineering have led to the realization of working quantum computers (QCs) [1], [2]. However, there are significant limitations to the capacity of individual QC [3]. Quantum networks (QNs) enable the construction of large, robust, and more capable quantum computing platforms by connecting smaller QCs. QNs [4] also enable various important applications [5]–[9]. However, QN communication is challenging—e.g., physical transmission of quantum states across nodes can incur irreparable communication errors, as the no-cloning theorem [10] proscribes making independent copies of arbitrary qubits. At the same time, certain aspects unique to the quantum regime, such as entangled states, enable novel mechanisms for communication. In particular, teleportation [11] transfers quantum states with just classical communication but requires an *a priori* establishment of entangled pairs (EPs). This article presents techniques for the efficient establishment of EPs in a network.

The establishment of EPs over long distances is challenging. Coordinated entanglement swapping (ES) (e.g., DLCZ protocol [12]) using quantum repeaters can be used to establish long-distance entanglements from short-distance entanglements. However, owing to the low probability of success of the underlying physical processes (short-distance entanglements and swappings), EP generation can incur significant latency—of the order of tens to hundreds of seconds between nodes hundreds of kilometers away [13]. Thus, we need to develop techniques that can facilitate the fast generation of long-distance EPs. We employ two strategies to minimize generation latencies: 1) select optimal swapping *trees* (not just paths as in prior works [14]–[17]) with a protocol that retains unused EPs and 2) use multiple trees for each given node pair; this reduces effective latency by using

all the available network resources. In the above context, we address the following problems.

1) *Quantum network routing (QNR) single-path* QNR-SP *problem:* Given a single (s, d) pair, select a minimum-latency swapping tree under given constraints.
2) QNR *problem:* Given a set of source–destination (s, d) pairs, select a set of swapping trees for each pair with the maximum aggregate EP generation rate, under fidelity and resource constraints.

To the best of our knowledge, no prior work has addressed the problem of selecting an efficient swapping tree for entanglement routing; they all consider selecting routing *paths* (Caleffi [18] selects a path using a metric based on balanced trees; see Section III-B). Almost all the prior works have considered the "waitless" model, wherein all the underlying physical processes much succeed *near simultaneously* for an EP to be generated; this model incurs minimal decoherence but yields very low EP generation rates. In contrast, we consider the "waiting" protocol, wherein, at each swap operation, the earlier arriving EP waits for a limited time for the other EP to be generated. Such an approach with efficient swapping trees yields high entanglement rates; the potential decoherence risk can be handled by discarding qubits that "age" beyond a certain threshold.

Our contributions: We formulate the entanglement routing problem (see Section III) in QNs in terms of selecting optimal swapping *trees* in the "waiting" protocol, under fidelity constraints. In this context, we make the following contributions.

1) For the QNR-SP problem, we design an optimal algorithm with fidelity and resource constraints (see Section IV).
2) Though polynomial time, the above optimal algorithm has high time complexity; we, thus, also design a near-linear time heuristic for the QNR-SP problem based on an appropriate metric, which essentially restricts the solutions to balanced swapping trees (see Section V).
3) For the general QNR problem, we design an efficient iterative augmenting-tree algorithm (see Section VI) and show its effectiveness with respect to an optimal linear programming (LP) solution based on hypergraph flows.
4) We conduct extensive evaluations (see Section VII) using NetSquid simulator and show that our solutions outperform the prior approaches by an order of magnitude, while incurring little fidelity degradation. We also show that our schemes can generate high-fidelity EPs over nodes 500–1000 km away.

II. QC BACKGROUND

Qubit states: Quantum computation manipulates *qubits* analogous to how classical computation manipulates *bits*. At any given time, a bit may be in one of two states, traditionally represented by 0 and 1. A quantum state represented by a *qubit* is a *superposition* of classical states and is usually written as

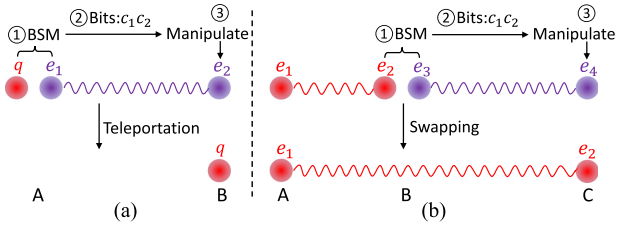

FIGURE 1. (a) Teleportation of $|q\rangle$ from *A* to *B*, while consuming an EP (e_1, e_2). (b) ES over the triplet of nodes (*A*, *B*, *C*), which results in *A*'s qubit entangled with *C*'s qubit. This can be viewed as a teleportation of e_2 from node *B* to *C*.

$\alpha_0|0\rangle + \alpha_1|1\rangle$, where α_0 and α_1 are *amplitudes* represented by complex numbers and such that $|\alpha_0|^2 + |\alpha_1|^2 = 1$. Here, $|0\rangle$ and $|1\rangle$ are the standard (orthonormal) *basis* states; concretely, they may represent physical properties, such as spin (down/up), polarization, charge direction, etc. When a qubit such as above is *measured*, it collapses to a $|0\rangle$ state with a probability of $|\alpha_0|^2$ and to a $|1\rangle$ state with a probability of $|\alpha_1|^2$. In general, a state of an n-qubit system can be represented as $\Sigma_{i=0}^{2^n-1}\alpha_i|i\rangle$, where "$i$" in $|i\rangle$ is i's bit representation.

Entanglement: Entangled pure[1] states are multiqubit states that cannot be "factorized" into independent single-qubit states. For example, the two-qubit state $\frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle$; this particular system is a *maximally entangled* state. We refer to maximally entangled pairs of qubits as EPs. The surprising aspect of entangled states is that the combined system continues to stay entangled, even when the individual qubits are physically separated by large distances. This facilitates many applications, e.g., the teleportation of qubit states by local operations and classical information exchange, as described next.

Teleportation: The direct transmission of quantum data is subject to unrecoverable errors, as classical procedures, such as amplified signals or retransmission, cannot be applied due to quantum no-cloning [10], [20].[2] An alternative mechanism for quantum communication is *teleportation* [see Fig. 1(a)], where a qubit q from a node A is recreated in another node B (*while "destroying" the original qubit* q) using only classical communication. However, this process requires that an EP already established over the nodes A and B. Teleportation can, thus, be used to reliably transfer quantum information. At a high level, the process of teleporting an arbitrary qubit, say qubit q, from node A to node B can be summarized as follows.

1) An EP pair (e_1, e_2) is generated over A and B, with e_1 stored at A and e_2 stored at B.

[1]In this article, we largely deal with only pure qubit states. Entanglement of general mixed states is defined in terms of separation of density matrices [19].

[2]Quantum error correction mechanisms [21], [22] can be used to mitigate the transmission errors, but their implementation is very challenging and is not expected to be used until later generations of QNs.

2) At A, a *Bell-state measurement* (BSM) operation over e_1 and q is performed, and the two-classical bit measurement output ($c_1 c_2$) is sent to B through the classical communication channel; at this point, the qubits q and e_1 at A are destroyed.
3) Manipulating the EP-pair qubit e_2 at B based on received (c_1, c_2) changes its state to q's initial state.

Depending on the physical realization of qubits and the BSM operation, it may not always be possible to successfully generate the two classical bits, as the BSM operation is stochastic.

Entanglement swapping: ES is an application of teleportation to generate EPs over remote nodes [see Fig. 1(b)]. If A and B share an EP and B teleports its qubit to C, then A and C end up sharing an EP. More elaborately, let us assume that A and B share an EP, and B and C share a separate EP. Now, B performs a BSM on its two qubits and communicates the result to C (teleporting its qubit that is entangled with A to C). When C finishes the protocol, it has a qubit that is entangled with A's qubit. Thus, an ES operation can be looked up as being performed over a triplet of nodes (A, B, C) with an EP available at the two pairs of adjacent nodes (A, B) and (B, C); it results in an EP over the pair of nodes (A, C).

Fidelity: decoherence and operation driven: Fidelity is a measure of how close a realized state is to the ideal. The fidelity of qubit decreases with time, due to interaction with the environment, as well as gate operations (e.g., in ES). Time-driven fidelity degradation is called *decoherence*. To bound decoherence, we limit the aggregate time a qubit spends in a quantum memory before being consumed. With regard to operation-driven fidelity degradation, Briegel *et al.* [23] give an expression that relates the fidelity of an EP generated by ES to the fidelities of the operands, in terms of the noise introduced by swap operations and the number of link EPs used. The order of the swap operations (i.e., the structure of the swapping tree) does not affect the fidelity. Thus, the operation-driven fidelity degradation of the final EP generated by a swapping tree T can be controlled by limiting the number of leaves of T, assuming that the link EPs have uniform fidelity (as in [15]).

Entanglement purification [23] and quantum error correction [24] have been widely used to combat fidelity degradation. Our work focuses on optimally scheduling ES operations with constraints on fidelity degradation, without purification or error correction.

Quantum memories: Multiple quantum memories have been recently proposed to bring QNs into realization. Types of quantum memories that support BSMs and gate unitary operations and probably have a long decoherence time can be used in quantum communications. Most of them are matter based, which have photonic interface to produce matter–matter entanglement over two neighboring nodes (see below). At a high level, there are three different quantum memory platforms: quantum dots, trapped atoms or ions, and color centers in diamond. Each has its own physical characteristics and applications. While quantum dots have the ability to process quantum information very fast, they exhibit a very low decoherence time among others [25], [26]. To overcome the low efficiency of single atom–photon coupling process, atomic ensemble schemes have been proposed [12], where along with dynamic decoupling and cooling techniques, decoherence times of a few seconds have been achieved [27]–[29]. For trapped ion memories, decoherence times from several minutes to few hours have been demonstrated [30], [31]. To further increase the entanglement generation rate, Bhaskar [32] proposes a way to use a single silicon–vacancy color center in diamond to perform asynchronous photonic BSM at the node located in the middle of two adjacent quantum nodes.

A. GENERATING EPS

As described above, teleportation, which is the only viable means of transferring quantum states over long distances, requires an *a priori* distribution of EPs. Thus, we need efficient mechanisms to establish EPs across remote QN nodes; this is the goal of our work. In the following, we start with describing how EPs are generated between adjacent (i.e., one-hop away) nodes and, then, discuss how EPs across a pair of remote nodes can be established via ESs.

1) GENERATING EP OVER ADJACENT NODES

The physical realization of qubits determines the technique used for sharing EPs between adjacent nodes. The *heralded entanglement* process [14], [18] to generate an atom–atom EP between adjacent nodes A and B is as follows.

1) Generate an EP of atom and a telecom-wavelength photon at node A and B. Qubits at each node are generally realized in an atomic form for longer term storage, while photonic qubits are used for transmission.
2) Once an atom–photon entanglement is locally generated at each node (at the same time), the telecom photons are then transmitted over an optical fiber to a photon–photon/optical BSM device C located in the middle of A and B so that the photons arrive at C at the same time.
3) The device C performs a BSM over the photons and transmits the classical result to A or B to complete ES.

Other entanglement generation processes have been proposed [21]; our techniques themselves are independent of how the link EP are generated.

2) GENERATING EP BETWEEN REMOTE NODES

Now, EP between nonadjacent nodes connected by a path in the network can be established by performing a sequence of ESs at intermediate nodes; this requires an *a priori* EP over each of the adjacent pairs of nodes in the path. For example, consider a path of nodes $x_0, x_1, x_2, x_3, x_4, x_5$, with an EP between every pair of adjacent nodes (x_i, x_{i+1}). Thus, each node x_i $(1 \leq i \leq 4)$ has two qubits, one of which is entangled

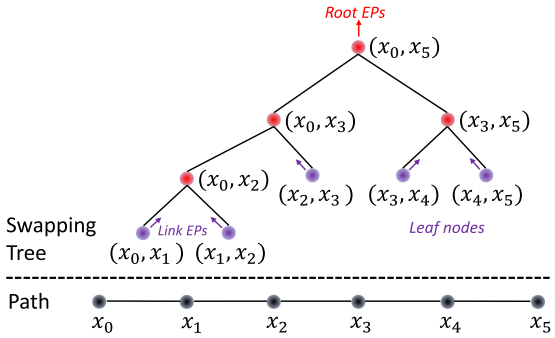

FIGURE 2. Swapping tree over a path. The leaves of the tree are the path links, which generate link EPs continuously.

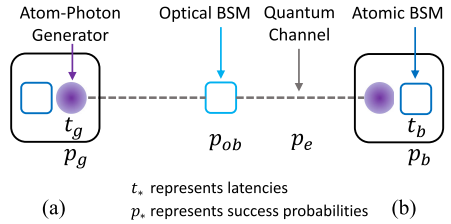

FIGURE 3. Key notations used.

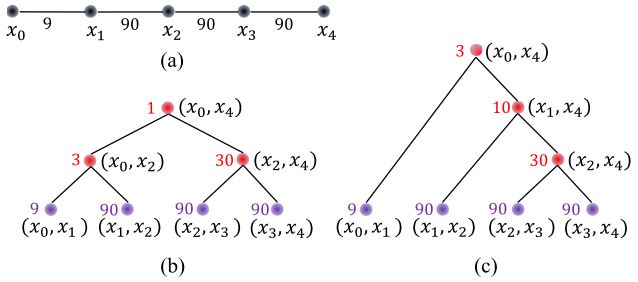

FIGURE 4. Consider the path in (a). The imbalanced tree of (b) has a higher EP generation rate than that of the balanced tree of (c). Here, the numbers represent the EP generation rates over adjacent links or node pairs.

with x_{i-1} and the other with x_{i+1}. Nodes x_0 and x_5 have only one qubit each. To establish an EP between x_0 and x_5, we can perform a sequence of ESs, as shown in Fig. 2. Similarly, the sequence of ES over the following triplets would also work: (x_2, x_3, x_4), (x_2, x_4, x_5), (x_0, x_1, x_2), (x_0, x_2, x_5).

3) SWAPPING TREES

In general, given a path $P = s \rightsquigarrow d$ from s to d, any complete binary tree (called a *swapping tree*) over the ordered links in P gives a way to generate an EP over (s, d). Each vertex in the tree corresponds to a pair of network nodes in P, with each leaf representing a link in P. Every pair of siblings (A, B) and (B, C) performs an ES over (A, B, C) to yield an EP over (A, C)—their parent; see Fig. 2. *Note that subtrees of a swapping tree execute in parallel.* Different swapping trees over the same path P can have different performance characteristics, as discussed later (see Fig. 4).

Expected generation latency/rate of EPs: In general, our goal is to continuously generate EPs at some rate using a swapping tree, using continuously generated EPs at the leaves. The stochastic nature of ES operations means that an EP at the tree's root will be successfully generated only after many failed attempts and, hence, significant latency. We refer to this latency as the *generation latency* of the EP at the root, and in short, just the generation latency of the tree. EP generation rate is the inverse of its generation latency. Whenever we refer to generation latency/rate, we implicitly mean *expected* generation latency/rate.

4) TWO GENERATION PROTOCOLS: `WaitLess` AND `Waiting`

When a swapping tree is used to (continuously) generate EPs, there are two fundamentally different generation protocols [13], [33].

1) `WaitLess` *protocol:* In this model, all the underlying processes, including link EP generations and atomic BSMs, are synchronized. If all of them succeed, then the end-to-end EP is generated. If *any* of the underlying processes fail, then all the generated EPs are discarded and the whole process starts again from scratch (from generation of EP at links). In the `WaitLess` protocol, all swapping trees over a given path P incur the same generation latency—thus, here, the goal is to select an optimal path P (as in [14] and [15]).
2) `Waiting` *protocol:* In the `Waiting` protocol, a qubit of an EP may wait (in a quantum memory) for its counterpart to become available so that an ES operation can be performed. Using such storage, we preclude discarding successfully generated EPs and, thus, reduce the overall latency in generation of a root-level EP. For example, let (A, B) and (B, C) be two siblings in a swapping tree and EP for (A, B) is generated first. Then, EP (A, B) may wait for the EP (B, C) to be successfully generated. Once the EP (B, C) is generated, the ES operation is done over the triplet (A, B, C) to generate the EP (A, C). If the EP (A, B) waits beyond a certain threshold, then it may decohere.

Hardware requirement differences: `WaitLess` protocols can generate EPs without quantum memories in a relay fashion if the EP generation among adjacent nodes can be tightly synchronized. In contrast, `Waiting` protocols benefit from memories with good coherence times (see Section VII).

5) WHY `Waiting`'S ENTANGLEMENT GENERATION RATE IS NEVER WORSE

The focus of the `WaitLess` protocol is to avoid qubit decoherence due to storage. However, it results in very low generation rates due to a very low probability of *all* the underlying processes succeeding at the same time. However, since qubit coherence times are typically higher than the link

generation latencies,[3] an appropriately designed `Waiting` protocol will always yield better generation rates *without significantly compromising the fidelity* (see Theorem 1). The key is to bound the waiting time to limit decoherence as desired, e.g., in our protocol, we restrict to trees with high expected fidelities (see Section III) and discard qubits that "age" beyond a threshold (see Section IV-B). Both the protocols use the same number of quantum memories (2 per node), though the `Waiting` protocols will benefit from low-decoherence memories; other hardware requirements also remain the same.

Theorem 1: Consider a QN, a path P, a swapping tree T over P, a `WaitLess` protocol X, and a `Waiting` protocol Y. Protocol Y discards qubits that age (stay in memory) beyond a certain threshold τ (presumably, equal to the coherence time). We claim that Y's EP generation rate will at least be that of X, irrespective of τ and T (as long as it is over P), while ensuring that EPs generated by Y are formed by noncohered qubits and the operation-driven fidelity degradation of Y EPs is same as X.

The above theorem suggests that the `Waiting` approach is always a better performing approach, irrespective of the decoherence time/limitations. See the proof in Appendix B.

III. MODEL, PROBLEM, AND RELATED WORKS

In this section, we discuss our network model, formulate the problem addressed, and discuss related work.

Network model: We denote a QN with a graph $G = (V, E)$, with $V = \{v_1, v_2, \ldots, v_n\}$ and $E = \{(v_i, v_j)\}$ denoting the set of nodes and links, respectively. Pairs of nodes connected by a link are defined as *adjacent* nodes. We follow the network model in [18] closely. Thus, each node has an atom–photon EP generator with generation latency (t_g) and probability of success (p_g). Generation latency is the time between successive attempts by the node to excite the atom to generate an atom–photon EP; this implicitly includes the times for photon transmission, optical BSM latency, and classical acknowledgement. *For clarity of presentation* and without loss of generality, we assume homogeneous network nodes with same parameter values. The generation rate is the inverse of generation latency, as before. A node's atom–photon generation capacity/rate is its aggregate capacity and may be split across its incident links (i.e., in generation of EPs over its incident links/nodes). Each node is also equipped with a certain number of atomic memories to store the qubits of the atom–atom EPs. A network link is a quantum channel (e.g., using an optical fiber or a free-space link) and, in our context, is used only for the establishment of link EP. In particular, a link $e = (A, B)$ is used to transmit telecom photons from A and B to the photon–photon BSM device in the middle of e. Thus, each link is composed of two half-links with a probability of transmission success (p_e) that decreases exponentially with the link distance (see Section VII). The optical BSM operation has a certain probability of success (p_{ob}). To facilitate atom–atom ES operations, each network node is also equipped with an atomic BSM device with an operation latency (t_b) and probability of success (p_b). See Fig. 3 for a pictorial view of the above notations. Finally, there is an independent classical network with a transmission latency (t_c); we assume that classical transmission always succeeds.

Single versus multiple links between nodes: For our techniques, multiple links between a pair of adjacent nodes can be replaced by a single link of aggregated rate/capacity. Hence, we assume only a single link between every pair of nodes. However, distinct multiple links between nodes have been used creatively in [14] (which refers to them as multiple channels); thus, we will discuss multiple links further in Section VII when we evaluate various techniques. We note that the all-photonic protocol in [39] is essentially a more sophisticated version of the multilink `WaitLess` protocol in [14] to further minimize memory requirements, but it uses multipartite cluster states, which are challenging to create. In either case, in terms of the selection of paths/trees, the path selection techniques from [14] should also apply to the all-photonic protocol with certain modifications to account for how the cluster states are generated.

EP generation latency of a swapping tree: Given a swapping tree and EP generation rates at the leaves (network links), we wish to estimate the generation latency of the EPs over the remote pair corresponding to the tree's root with the `Waiting` protocol. In the following, we develop a recursive equation. Consider a node (A, C) in the tree, with (A, B) and (B, C) as its two children. Let T_{AB}, T_{BC}, and T_{AC} be the corresponding (expected) generation latencies of the EPs over the three pairs of nodes. In the following, we derive an expression for T_{AC} in terms of T_{AB} and T_{BC}; this expression will be sufficient to determine the expected latency of the overall swapping tree by applying the expression iteratively. We start with an observation.

Observation 1: If two EP arrival processes X_1 and X_2 are exponentially distributed with a mean interarrival latency of λ each, then the expected interarrival latency of $\max(X, Y)$ is $(3/2)\lambda$.

From above, if assume T_{AB} and T_{BC} to be exponentially distributed with the same expected generation latency of T, then the expected latency of both EPs arriving is $(3/2)T$. Thus, we have

$$T_{AC} = \left(\frac{3}{2}T + t_b + t_c\right)/p_b. \tag{1}$$

Remarks: We make the following remarks regarding the above expression. First, when $T_{AB} \neq T_{BC}$, we are able to only derive an upper bound on T_{AC}, which is given by the above equation but with T replaced by $\max(T_{AB}, T_{BC})$.[4] However,

[3] Link generation latencies for 5–100 km links range from about 3–350 ms for typical network parameters [18], while coherence times of few seconds are very realistic (coherence times of several seconds [34], [35] have been shown long ago, and more recently, even coherence times of several minutes [36], [37] to a few hours [31], [38] have been demonstrated).

[4] The 3-over-2 formula as an upper bound has also been corroborated in a recent work [40], which derives analytical bounds on EP latency times in more general contexts.

in our methods, the above assumption of $T_{AB} = T_{BC}$ will hold as we would only be considering "throttled" trees to save on underlying network resources (see Section IV). Second, our motivation for the exponential distribution assumption stems from the fact that the EP generation latency at the *link level* is certainly exponentially distributed if we assume the underlying probabilistic events to have a Poisson distribution. Third, note that the resulting distribution is not exponential. Despite this, we apply the above equation recursively to compute the tree's generation latency. However, in our evaluations, we observe the validity of this approximation since our analysis matches closely with the simulation results. Finally, (1) is conservative in the sense that each round of an EP generation of any subtree's root starts from scratch (i.e., with no link EPs from prior round) and ends with either an EP generation at the *whole swapping tree*'s root or an atomic BSM failure at the subtree's root. We do not "pipeline" any operations across rounds within a subtree, which may lower latency; this is beyond the scope of this article.

A. PROBLEM FORMULATION

We now formulate the central problem of selecting *multiple* swapping trees for each given source–destination pair. The selection of multiple routes is a well-established strategy [14]–[16] to maximize entanglement rates.

*Quantum network routing (*QNR*) problem:* Given a QN and a set of source–destination pairs $\{(s_i, d_i)\}$, the QNR problem is to determine a set $\mathcal{T}_i$ of swapping trees for each pair (s_i, d_i) such that the sum of the EP rates of all the trees in $\bigcup_i \mathcal{T}_i$ is maximized under the following constraints.

1) *Node constraints:* For each node, the aggregate resources used by $\bigcup_i \mathcal{T}_i$ are less than the available resources; we formulate this formally in the following.
2) *Fidelity constraints:* Each swapping tree in $\bigcup_i \mathcal{T}_i$ satisfies the following: a) the number of leaves is less than a given threshold τ_l; this is to limit fidelity degradation due to gate operations; and b) total memory storage time of any qubit is less[5] than a given *decoherence threshold* τ_d.

Informally, the swapping trees may also satisfy some fairness constraint across the given source–destination pairs. A special case of the above QNR problem is to select a single tree for a source–destination pair; we address this in the next section.

Formulating node constraints: Consider a swapping tree $\mathcal{T} \in \bigcup_i \mathcal{T}_i$ over a path P. For each link $e \in P$, let $R(e, \mathcal{T})$ be the EP rate being used by $\mathcal{T}$ over the link e in P. Let us define $R_e = \sum_{\mathcal{T}} R(e, \mathcal{T})$, and let $E(i)$ be the set of edges incident on i. Then, the node capacity constraint is formulated as follows:

$$1/t_g \quad \geq \quad \sum_{e \in E(i)} R_e/({p_g}^2 {p_e}^2 p_{ob}) \quad \forall i \in V. \tag{2}$$

The above comes from the fact that to generate a single link EP over e, each end node of e needs to generate $1/({p_g}^2 {p_e}^2 p_{ob})$ photons successfully, since each photon (from each end node) has a generation success of p_g and a transmission success rate of p_e, and the optical BSM's success probability over the two successfully arriving photons is p_{ob}. Note that $1/t_g$ is a node's total generation capacity. In addition, the memory constraint is that for any node i, the memory available in i should be more than $2x + y$, where x is the number of swapping trees that use i as an intermediate node and y is the number of trees that use i as an end node.

B. RELATED WORKS

There have been a few works in the recent years that have addressed generating long-distance EPs efficiently. All of these works have focused on selecting an efficient routing path for the swapping process (Caleffi [18] also selects a path, but using a metric based on balanced trees). In addition, all except [18] have looked at the `WaitLess` protocol of generating the EPs. Recall that in the `WaitLess` model, the selection of paths suffices, while in the `Waiting` model, one needs to consider the selection of efficient swapping trees with high fidelity. The selection of optimal swapping trees is a fundamentally more challenging problem than the selection of paths—and has not been addressed before, to the best of our knowledge. We start with discussing how the `WaitLess` model works.

1) `WaitLess` APPROACHES

The most recent works to address the above problem are [14] and [15], both of which consider the `WaitLess` model. In particular, Shi and Qian [14] design a Dijkstra-like algorithm to construct an optimal path between a pair of nodes, when there are multiple links (channels) between adjacent nodes. Then, they use the algorithm iteratively to select multiple paths over multiple pairs of nodes. Chakraborty *et al.* [15] design a multicommodity-flow-like LP formulation to select routing paths for a set of source–destination pairs. They map the operation-based fidelity constraint to the path length (as in [23]) and use node copies to model the constraint in the LP. However, they explicitly assume that the link EP generation is deterministic—i.e., always succeeds. Among earlier relevant works, Pant *et al.* [16] propose a greedy solution for grid networks, and Chakraborty *et al.* [17] propose virtual-path-based routing in ring/grid networks.

2) `Waiting` APPROACH

Owing to photon loss, establishing long-distance entanglement between remote nodes at L distance by *direct* transmission yields EP rates that decay exponentially with L. The

[5] We note that, in our context, the storage time and the memory coherence time are statistical quantities due to the underlying statistical mechanisms. However, for the purposes of *selecting* a swapping tree, we use a fixed decoherence threshold τ_d value equal to the mean of the distribution of the coherence time (recent work [41] computes optimal cutoffs/thresholds, and their techniques can be used to pick τ_d). When simulating a selected tree for generation of EPs, we can implement coherence time as a statistical measure.

DLCZ protocol [12], [13] broke this exponential barrier using 2^k equidistant intermediate nodes to perform ES operations, implicitly over a balanced binary tree, with a `Waiting` protocol; this makes the EP generation rate decay only polynomially in L. More recently, Caleffi [18] formulated the entanglement generation rate on a given path between two nodes, under the more realistic condition where the intermediate nodes in the path may not all be equidistant, but still considered only balanced trees. Their path-based metric was then used to select the optimal path by enumerating over the exponentially many paths in the network.

Our approach (versus [18]): Though Caleffi [18] considers only balanced trees, its brute-force algorithm is literally impossible to run for networks more than a few tens of nodes (see Section VII). In our work, we observe that a path has many swapping trees, and, in general, imbalanced trees may even be better (see Fig. 4). Thus, we design a polynomial-time dynamic programming (DP) algorithm that delivers an *optimal* high-fidelity swapping tree; our DP approach effectively considers all possible swapping trees, not just balanced ones (note that, even over a single path, there are exponentially many trees). The incorporation of fidelity (including decoherence) in our DP approach requires nontrivial observation and analysis (see Section IV-B). Our `Balanced-Tree` heuristic (see Section V) is closer to [18]'s work, in that both consider only balanced trees; however, we use a heuristic metric that facilitates a polynomial-time Dijkstra-like heuristic to select the optimal path, while their recursive metric[6] (albeit more accurate than ours) is not amenable to an efficient (polynomial-time) search algorithm.

3) OTHER WORKS

Jiang *et al.* [42] address a related problem; given a path with uniform link lengths, they give an algorithm for selecting an optimal sequence of swapping and purification operations to produce an EP with fidelity constraints. In other recent works, Dahlberg *et al.* [43] design physical and link layer protocols of a QN stack, and Kozlowski *et al.* [44] propose a data plane protocol to generate EPs within decoherence thresholds along a *given* routing path. More recently, Bugalho *et al.* [45] propose an algorithm to efficiently distribute multipartite entanglement across over than two nodes.

IV. OPTIMAL ALGORITHM FOR SINGLE TREE

In this section, we consider a special case of the `QNR` problem, viz., the case wherein there is a single source–destination (s, d) pair and the goal is to select a *single* swapping tree for the (s, d) pair. For this special case, we design an optimal algorithm based on DP. This optimal algorithm can be used iteratively to develop an efficient heuristic for the general `QNR` problem, as in Section VI.

[6]We note that their formula (see [18, eq. (10)]) is incorrect as it either ignores the 3/2 factor or assumes the EP generations to be synchronized *across all* links. In addition, their expression for "qubit age" ignores the "waiting for ES" time completely.

`QNR` *single-path problem:* Given a QN and a source–destination pair (s, d), the `QNR-SP` problem is to determine a single swapping tree that maximizes the expected generation rate (i.e., minimizes the expected generation latency) of EPs over (s, d), under the capacity and fidelity constraints.

For homogeneous nodes and link parameters, it is easy to see that the best swapping tree is the balanced or almost-balanced tree over the shortest path. We note that `QNR-SP` is not a special case of `QNR` in the formal sense, e.g., the LP algorithm (see Appendix A) for `QNR` cannot be used for the `QNR-SP` problem, due to the single-tree requirement (LP may produce multiple trees). As described in Section III-B, the `QNR-SP` problem has been addressed before in [14] and [18] under different models.

A. DP FORMULATION

First, we note that a Dijkstra-like shortest path approach that builds a shortest-path tree greedily does not work for the `QNR-SP` problem—mainly because the task is to find an optimal *tree* rather than an optimal path. As noted before, a routing path can have exponentially many swapping trees over it, with different generation latencies. The recursive expression for computing the generation latency given in Section III suggests that a DP approach, similar to the Bellman–Ford or Floyd–Warshall's classical algorithms for shortest paths, may be applicable for the `QNR-SP` problem. However, we need to "combine" trees rather than paths in the recursive step of a DP approach. Consequently, we were unable to design a DP approach based on the Floyd–Warshall's approach but are able to extend the Bellman–Ford approach for the `QNR-SP` problem after addressing a few challenges discussed in the following.

DP formulation: We start with designing a DP algorithm without worrying about the decoherence constraint; we incorporate the decoherence constraint in the next subsection. Given a network, let $T[i, j, h]$ be the optimal expected latency of generating EP pairs over (i, j) using a swapping tree of height at most h. Note that $T[i, j, 0]$ for adjacent nodes (i, j) can be given by $\frac{t_g}{{p_g}^2 {p_e}^2 p_{ob}}$. Now, based on (1), we start with the following equation for computing $T[i, j, h]$ in terms of smaller height swapping trees.

$$T[i, j, h] = \min\left(T[i, j, h-1], \left(\frac{3}{2}B + t_c + t_b\right)/p_b\right)$$

where

$$B = \min_{k \in V} \ \max\left(T[i, k, h-1], T[k, j, h-1]\right). \quad (3)$$

However, there are three issues that need to be addressed before the above formulation can be turned into a viable algorithm. We address these in the following three paragraphs.

(1) The 3/2 factor; throttled trees: As mentioned in Section III, the 3/2 factor is an accurate estimate if the corresponding Ts are equal. However, in the above equation,

$T[i, k, h-1]$ and $T[k, j, h-1]$ may not be equal. In our overall methodology, to conserve node and link resources, we postprocess or "throttle" the swapping tree obtained from the DP algorithm by increasing the generation latencies of some of the *nonroot* nodes such that 1) the latencies of siblings are equalized and 2) the parents' latency is related to the children's latency by (1). We refer to this postprocessing as *throttling* and a tree that satisfies the above conditions as a *throttled* tree. Note that throttling does not alter the generation latency of the root and, thus, the overall tree; we prove the optimality of the overall algorithm formally in Theorem 2. In the following, we motivate throttling and describe how it is achieved.

Justification: In a given swapping tree, consider a pair of siblings x and y that have unequal generation latencies/rates. Let x be the one with a lower latency (higher rate). Then, x will likely have to discard many EPs while waiting for an EP from y. To minimize this discarding of EPs from x and to conserve underlying network resources so that they can be used in other swapping trees (in a general QNR solution), we "throttle," increase (decrease) the generation latency (rate) of, the sibling x to match that of y.

Throttling process: Consider a pair of siblings x and y in the tree; let their parent be z. Let T_x, T_y, and T_z be their current generation latencies, such that $T_z = (\frac{3}{2}\max(T_x, T_y) + t_c + t_b)/p_b$. There are two potential steps: 1) if the parent's latency is to be kept unchanged, but $T_x < T_y$, then T_x is increased to T_y, which, thus, makes the above equation valid; and 2) if the parent's latency T_z is increased to T (by the above first step, with z as a sibling), then we increase the latencies of both x and y to $2/3(Tp_b - t_c - t_b)$. It is easy to see that applying the two steps iteratively from the root to the leaves yields a throttled tree, as defined above.

(2) Capacity violation at node k: Note that the middle/common node k in (3) may violate (node) capacity constraints in the merged tree corresponding to $T[i, k, h]$, as it may use its full capacity in the trees corresponding to $T[i, k, h-1]$ and $T[k, j, h-1]$. We address the above by adding two additional parameters to the subproblem function T, corresponding to "usage percentage" of the end nodes. In particular, we define $T[i, j, h, u_i, u_j]$ as the optimal latency of a swapping tree of height at most h, under the constraint that the end nodes i and j use at most u_i and u_j percentage of the respective node generation capacities; here, u_i and u_j can be positive integers between 1 and 100. The base case $T[i, j, 0, u_i, u_j]$ for adjacent nodes (i, j) is given by $\frac{t_g \min(u_i, u_j)}{p_g^2 p_e^2 p_{ob}}$. Equation (3) is modified as follows to accommodate the additional usage parameters:

$$T[i, j, h, u_i, u_j] = \min\left(T[i, j, h-1, u_i, u_j], \left(\frac{3}{2}B + t_c + t_b\right)/p_b\right) \quad (4)$$

where

$$B = \min_{k,\ u+u'=100} \max\left(T[i, k, h-1, u_i, u], T[k, j, h-1, u', u_j]\right).$$

(3) Ensuring disjoint subtrees: Note that (3) implicitly assumes that the swapping trees corresponding to the latency values $T[i, k, h-1]$ and $T[k, j, h-1]$ are over disjoint paths, i.e., there is no node v such that both the paths contain v. If there is a common node v, then the combined tree corresponding to $[i, j, h]$ may violate the node capacity constraints at v. This issue also arises in the classical Bellman–Ford's or Floyd–Warshall's algorithms for shortest weighted paths, but is harmless with the assumption of positive-weighted cycles. We resolve the issue similarly here via the following lemma (see Appendix C for the proof).

Lemma 1: Consider two swapping trees $\mathcal{T}_{ik}$ and $\mathcal{T}_{kj}$ each of height at most $h-1$ over paths $P_1 : i \rightsquigarrow v \rightsquigarrow k$ and $P_2 : k \rightsquigarrow v \rightsquigarrow j$, each of which contains a common node $v \neq k$. Then, there exists two swapping trees $\mathcal{T}_{iv}$ and $\mathcal{T}_{vj}$ each of height at most $h-1$ over paths $P_1' : i \rightsquigarrow v$ and $P_2' : v \rightsquigarrow j$ such that: 1) P_1' is a subset of P_1, and P_2' is a subset of P_2, and 2) generation latency of $\mathcal{T}_{iv}$ is no greater than that of $\mathcal{T}_{ik}$, and generation latency of $\mathcal{T}_{vj}$ is no greater than that of $\mathcal{T}_{kj}$. ■

Lemma 1 implies that if the swapping trees $\mathcal{T}_{ik}$ and $\mathcal{T}_{kj}$ corresponding to the latency values $T[i, k, h-1]$ and $T[k, j, h-1]$ have a common node, then there exist swapping trees of equal or better latency without any common nodes and these trees can be used to build a lower latency tree over (i, j).

Overall DP algorithm and optimality: Our DP-based algorithm for the QNR-SP problem for a given (s, d) pair is as follows. We use a DP formulation based on (4) and the corresponding base case values to compute optimal generation latency $T[s, d, h, 100, 100]$ and the corresponding swapping tree $\mathcal{T}$. Then, we throttle the tree $\mathcal{T}$ as described in paragraph (1) above. The following theorem (see Appendix D for proof) states that the throttled tree thus obtained has the optimal (minimum) expected generation latency among all throttled trees.

Theorem 2: The above-described DP-based algorithm returns a throttled swapping tree over (s, d) with minimum expected generation latency (maximum expected generation rate) among all throttled trees over the given (s, d) pair. ■

B. INCORPORATING FIDELITY CONSTRAINTS

Till now, we have ignored the fidelity constraints. We incorporate them in this section by extending our DP formulation from the previous section. Limiting the decoherence, i.e., the qubit storage time, is challenging and is addressed first in the following. Limiting the number of leaves of a swapping tree is relatively easier and is discussed next. We start with a definition.

Definition 1 (Qubit/tree age): Given a swapping tree, the total time spent by a qubit in a swapping tree is the time spent from its "birth" via an atom–photon EP generation at a node

IEEE Transactions on Quantum Engineering

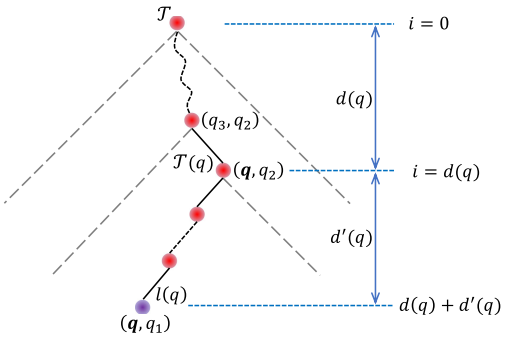

FIGURE 5. Qubit parameters in a swapping tree used to compute the *age* of a qubit q at a leaf node $l(q)$. Here, $l(q)$ is the left-most leaf of the subtree $\mathcal{T}(q)$.

till its consumption in a swapping operation or in generation of the tree's root EP. We refer to this as a qubit's *age*. The maximum age over all qubits in a swapping tree is called the tree's (expected) *age*. □

Estimating qubit age in a swapping tree: Consider a throttled swapping tree $\mathcal{T}$, with a generation latency of T. Consider two siblings (A, B) and (B, C) at a depth[7] of i ($i > 0$) from $\mathcal{T}$'s root. If we ignore t_c and t_b terms in (1), then the expected generation latency $T(i)$ of both (A, B) and (B, C) being at depth i is given by: $T(i) = \frac{T}{2}(\frac{2}{3}p_b)^i$. In addition, note that only *one* of the EPs (A, B) or (B, C) waits for $T(i)$ time on average. Thus, the expected waiting times for each of the four[8] qubits is $T(i)/2$.

Based on the above, we can now easily estimate the total waiting by a qubit q (referred to as q's *age*) before it is destroyed in a swapping operation. Let $l(q)$ be the leaf, i.e., the link EP, of $\mathcal{T}$ that contains the qubit q. Let $\mathcal{T}(q)$ be the maximal subtree in $\mathcal{T}$ such that $l(q)$ is either its right-most or left-most leaf. Note that $\mathcal{T}(q)$ is well defined for a tree $\mathcal{T}$ and a qubit q. Let $d(q)$ be the depth of the root of $\mathcal{T}(q)$ in $\mathcal{T}$, and let $d'(q)$ be the depth of $l(q)$ in the subtree $\mathcal{T}(q)$ (see Fig. 5). The expected age $A(q)$ of q can be estimated as follows. Note that *age* of q is the total waiting by q at each of $l(q)$'s ancestors in $\mathcal{T}(q)$; also note that at $\mathcal{T}(q)$'s root, the qubit q is destroyed, and hence, q does not age at any ancestor of $T(q)$'s root. It is easy to see that the expected age $A(q)$ is

$$A(q) = \left(\sum_{i=d(q)}^{d(q)+d'(q)} T(i)/2 \right) + (t_{ob} + t_p).$$

In the above equation, the last term is the time spent by q waiting for its link EP to be established and is given by sum of optical BSM (t_{ob}) and photon transmission latency (t_p). Note that the actual age of a qubit q is some distribution with the above mean. We observe the following.

Observation 2: Given a swapping tree $\mathcal{T}$, let $\mathcal{T}_l$ and $\mathcal{T}_r$ be its left and right children, respectively. If the atomic BSM probability p_b is $\leq 75\%$, then the expected age of the right-most or left-most descendant of either $\mathcal{T}_l$ or $\mathcal{T}_r$ is greater than the expected age of any other qubit in the tree. □

DP formulation with decoherence/age constraint: If we assume the atomic BSM probability $p_b \leq 75\%$, then we can design a DP algorithm for the QNR-SP problem with the decoherence constraint, as follows. Let $T[i, j, h, h_{ll}, h_{lr}, h_{rl}, h_{rr}, u_i, u_j]$ be the optimal latency from a swapping tree of height at most h, whose root's left (right) child's left-most and right-most descendants are at depths of (*exactly*) h_{ll} and h_{lr} (h_{rl} and h_{rr}), each of which is upper bounded by h. Here, u_i and u_j parameters are as before. Note that $T[i, j, 1, 0, 0, 0, 0, u_i, u_j] = \frac{t_g \min(u_i, u_j)}{p_g^2 p_e^2 p_{ob}}$. We have

$$\begin{aligned} &T[i, j, h, h_{ll}, h_{lr}, h_{rl}, h_{rr}, u_i, u_j] \\ &= \min \Big(T[i, j, h-1, h_{ll}, h_{lr}, h_{rl}, h_{rr}, u_i, u_j], \\ &\left(\frac{3}{2}B + t_c + t_b \right) / p_b \Big) \end{aligned} \tag{5}$$

where

$$\begin{aligned} B = \min_{k,\ g_i's, u+u'=100} \max\ (&T[i, k, h-1, h_{ll}-1, g_1, g_2 \\ &h_{lr}-1, u_i, u], \\ &T[k, j, h-1, h_{rl}-1, g_3, g_4 \\ &h_{rr}-1, u', u_j]) . \end{aligned}$$

The above formulation will give us the optimal latency swapping tree for each combination of $(h_{ll}, h_{lr}, h_{rl}, h_{rr})$. We remove the trees that violate the decoherence constraint and pick the minimum-latency tree from the remaining. This gives us a swapping tree with optimal latency under the decoherence constraint. The proof of optimality easily follows.

Constraint on the number of leaves: Limiting the number of leaves to τ_l can be easily done by adding another parameter for the number of leaves in the T array/function above. This adds another factor of $O(n^2)$ to the time complexity, as we need to check for *all* the combinations of the number of leaves in the two subtrees. To optimize, we can now replace the height parameter, but keeping the height parameters aids in parallelism, as described in the following.

Time complexity; DP-OPT *and* DP-Approx *algorithms:* Note that, in (6), we can precompute $\min_{g_1,g_2} T[\ldots, g_1, g_2, \ldots]$ and similarly $\min_{g_3,g_4} T[\ldots, g_3, g_4, \ldots]$ before computing B. With this, the time complexity of the DP formulation becomes $O(n^9)$, which can be further reduced $O(n^5(\log n)^4)$ if we assume height of a tree to be at most $(c \log n)$ for some constant c. For a real-time routing application, the above time complexity is still high—as the algorithm can take a few minutes on a single core. However, as the algorithm lends to obvious parallelism, it can be executed in as little as $O((\log n)^2)$ time with sufficiently many cores, using the height parameter sequentially. We can also reduce

[7]Defined as the distance of a node from the root; depth of the root is 0.
[8]Note that qubit B in (A, B) is different from that in (B, C).

the sequential time complexity to $O(n^5)$, by approximating the maximum qubit's age in a tree to the generation latency of the tree, which is at most $3/(2p_b)$ the actual value. Note that maximum age of a qubit is at least $2Tp_b/3$ and at most T, where T is the generation latency of the tree. Finally, we can make the algorithm more efficient by assuming the usage parameter values to be 50%.[9] We refer to the $O(n^5)$ algorithm with the above assumptions as `DP-Approx`, and the $O(n^5(\log n)^4)$ algorithm based on (6) as `DP-OPT`. Both the algorithms use throttling after the DP formulation.

V. `Balanced-Tree` HEURISTIC FOR `QNR-SP`

The DP-based algorithms presented in Section IV for the `QNR-SP` problem have high time complexity and, thus, may not be practical for real-time route finding in large networks. In this section, we develop an almost-linear time heuristic for the `QNR-SP` problem, based on the classic Dijkstra shortest path algorithm; the designed heuristic performs close to the DP-based algorithms in our empirical studies.

Basic idea: The main reason for the high complexity of our DP-based algorithms in Section IV is that the goal of the `QNR-SP` problem is to select an optimal swapping *tree* rather than a path. One way to circumvent this challenge efficiently while still selecting near-optimal swapping tree is to restrict ourselves to only "balanced" swapping trees. This restriction allows us to think in terms of the selection of paths—rather than trees—since each path has a unique[10] balanced swapping tree. We can then develop an appropriate path metric based on above and design a Dijkstra-like algorithm to select an (s, d) path that has the optimal metric value. We note that Caleffi [18] also proposed a path metric based on balanced swapping trees, but their metric, though accurate, only had a recursive formulation without a closed-form expression—and hence, was ultimately not useful in designing an efficient algorithm. In contrast, we develop an approximate metric with a closed-form expression, based on the "bottleneck" link, as follows.

Path metric M: Consider a path $P = (s, x_1, x_2, \ldots, x_n, d)$ from s to d, with links $(s, x1), (x1, x2), \ldots, (x_n, d)$ with given EP latencies. We define the path metric for path P, $M(P)$, as the EP generation latency of a balanced swapping over P, which can be estimated as follows. Let L be the link in P with maximum generation latency. If L's depth (distance from the root) is the maximum in a throttled swapping tree, then we can easily determine the accurate generation latency of the tree. However, in general, L may not have the maximum depth, in which case we can still estimate the tree's latency approximately, if the tree is balanced, as follows. In balanced swapping trees, *assuming* the maximum latency link L to be at the maximum depth gives us a constant-factor approximation of the tree's generation latency. Thus, let us assume L to be at the maximum depth of a balanced tree over P; this maximum depth is $d = \lceil(\log_2 |P|)\rceil$. Let the generation latency of L be T_L. If we ignore the $t_b + t_c$ term in (1), then the generation latency of a throttled swapping tree can be easily estimated to $T(\frac{3}{2p_b})^d$. The term $t_b + t_c$ can also be incorporated as follows. Let $T(i)$ denote the expected latency of the ancestor of L at a distance i from L. Then, we get the recursive equation: $T(i) = (\frac{3}{2}T(i-1) + t_b + t_c)/p_b$. Then, the path metric value $M(P)$ for path P is given by $T(d)$, the generation latency of the tree's root at a distance of d from L, and is equal to

$$M(P) = T(d) = \bar{p}^d T_L + [(\bar{p} - 1)/(\bar{p}^d - 1)](t_b + t_c)/p_b$$

where $\bar{p} = 3/(2p_b)$ and $d = \lceil(\log_2 |P|)\rceil$. The above is a (1+3/(2$p_b$))-factor approximation latency of a balanced and throttled swapping tree over P; this can be shown easily using analysis from Section IV-B.

Optimal balanced-tree selection: The above path metric $M()$ is a monotonically increasing function over paths, i.e., if a path P_1 is a subsequence of another path P_2, then $M(P_1) \leq M(P_2)$. Thus, we can tailor the classical Dijkstra's shortest path algorithm to select an (s, d) path with minimum $M(P)$ value, using the link's EP generation latencies as their weights. We refer to this algorithm as `Balanced-Tree`, and it can be implemented with a time complexity of $O(m + n \log n)$ using Fibonacci heaps, where m is the number of edges and n is the number nodes in the network.

Incorporating fidelity constraints: Fidelity constraints in our path-metric-based setting can be handled by essentially computing the optimal path for each path length (number of hops in the path) up to τ_l and then pick the best path among them that satisfies the fidelity constraints. This obviously limits the number of leaves to τ_l and addresses the operation-based fidelity degradation. The above also address the decoherence/age constraint, since it is easy to see (from analysis in Section IV-B) that the age of a balanced swapping tree can be very closely approximated in terms of the latency and the number of leaves. Now, to compute the optimal path for each path length, we can use a simple DP approach that run in $O(m\tau_l)$ time, where m is the number of edges and τ_l is the constraint on the number of leaves.

VI. `ITER`: ITERATIVE `QNR` HEURISTIC

The general `QNR` problem can be formulated in terms of hypergraph flows and solved using LP (see Appendix A). Although polynomial-time and provably optimal, the LP-based approach has a very high time complexity for it to be practically useful. Here, we develop an efficient heuristic for the `QNR` problem by iteratively using an `QNR-SP` algorithm.

`ITER` *heuristic:* To solve the `QNR` problem efficiently, we apply the efficient `DP-Approx` algorithm iteratively—finding an efficient swapping tree in each iteration for one of the (s_i, d_i) pairs. The proposed algorithm is similar to the classical Ford–Fulkerson augmenting path algorithm for the max network flow problem at a high level, with some low level and theoretical differences, as discussed in the

[9] This also enforces s and d to use only 50% capacity; this can be resolved by doubling s and d capacity *a priori*.

[10] In fact, there can be multiple balanced trees over a path whose length is not a power of 2, but, since they differ minimally in our context, we can pick a unique way of constructing a balanced tree over a path.

following. The iterative-DP-Approx algorithm for the QNR problem consists of the following steps.

1) Given a network, we compute maximum EP generation rates for each network link using (7). Use these as weights on the link.
2) For each (s_i, d_i) pair, use the DP-Approx algorithm to find the optimal path P_i, under the capacity and fidelity constraints. Consider a *throttled* and balanced swapping tree $\mathcal{T}_i$ over P_i. Let $\mathcal{T}^*$ be the swapping tree with highest generation rate; if this rate is below a certain threshold, then quit.
3) Construct a residual network graph by subtracting the resources used by $\mathcal{T}^*$, using (8).
4) Go to step 1.

Before we present the expressions required above, we would like to point out key differences of our context with the classic network flow setting. Even though we are augmenting our solution one *path* at a time, the network resources are fundamentally being used by swapping trees created over these paths. These path flows do not really have a direction of flow, but we can assign them a symbolic direction from source to the direction. Even with these symbolic directions, the flows in opposite directions over any edge k do not "cancel" each other as in the classical network flow. Moreover, flow conservation law does not hold in our context (e.g., even a path may not use same link rates on all links, due to them being at different depths of the tree), and thus, the max-flow min-cut theorem does not hold. Thus, ITER may not give an optimal solution, even for a single (s, d) pair.

Link EP generation rate/latency: Consider a pair of network node i and j with corresponding *current (residual)* values of node latencies as $t_g(i)$ and $t_g(j)$. Assuming p_g values to be same for both nodes, the minimum EP link rate for (i, j) is then given by

$$\min(1/t_g(i), 1/t_g(j)) {p_g}^2 {p_e}^2 p_{ob}. \tag{6}$$

Residual node capacities: Let P be a path added by ITER, at some earlier stage, and let $\mathcal{T}$ be the corresponding throttled swapping tree over P. As in Section III, let $R(e, \mathcal{T})$ be the EP generation rate being used by $\mathcal{T}$ over a link $e \in P$, $R_e = \sum_{\mathcal{T}} R(e, \mathcal{T})$, and $E(i)$ be the edges incident on i. Then, the residual node rates can be calculated similar to (2) as follows. In the following, $t_g{}'(i)$ is the *original* value

$$1/t_g(i) = 1/t_g{}'(i) - \sum_{e \in E(i)} R_e/({p_g}^2 {p_e}^2 p_{ob}) \quad \forall i \in V. \tag{7}$$

The residual memory capacity is easy to compute—each path/tree uses two memory units for each intermediate node and one memory unit for the end nodes.

VII. EVALUATIONS

The goal of our evaluations is to compare the EP generation rates, evaluate the fidelity of generated EPs, and validate our analytical models. We implement the various schemes over a discrete event simulator for QNs called NetSquid [46]. The NetSquid simulator accurately models various QN components/aspects, and in particular, we are able to define various QN components and simulate swapping trees protocols by implementing gate operations in ES.

A. SWAPPING TREE PROTOCOL

Our algorithms compute swapping tree(s), and we need a way to implement them on a network. We build our protocol on top of the link layer of [43], which is delegated with the task of continuously generating EPs on a link at a desired rate (as per the swapping tree specifications). Note that a link (a, b) may be in multiple swapping trees and, hence, may need to handle multiple link layer requests at the same time; we implement such link layer requests by creating independent atom–photon generators at a and b, with one pair of synchronized generators for each link layer request. As the links generate continuous EPs at desired rates, we need a protocol to swap the EPs. Omitting the tedious bookkeeping details, the key aspect of the protocol is that swap operation is done only when both the appropriate EP pairs have arrived. We implement all the gate operations (including atomic and optical BSMs) within NetSquid to keep track of the fidelity of the qubits. On BSM success, the swapping node transmits classical bits to the end node, which manipulates its qubit, and sends the final ack to the other end node. On BSM failure, a classical ack is send to all descendant link leaves, so that they can now start accepting new link EPs; note that in our protocol, a link l does not accept any more EPs, while its ancestor is waiting for its sibling's EP (see Fig. 6).

B. SIMULATION SETTING

We use a similar setting as in the recent work [14]. By default, we use a network spread over an area of 100˜km × 100˜km. We use the Waxman model [47], used to create Internet topologies, to randomly distribute the nodes and create links; we use the maximum link distance to be 10 km. We vary the number of nodes from 25 to 500, with 100 as the default value. We choose the two parameters in the Waxman model to maintain the number of links to 3% of the complete graph (to ensure an average degree of 3–15 nodes). For the QNR-SP problem, we pick (s, d) pairs within a certain range of distance, with the default being 30–40 km; for the QNR problem, we extend this range to 10–70 km.

1) PARAMETER VALUES

We use parameter values mostly similar to the ones used in [18] corresponding to a single-atom-based quantum memory platform and vary some of them. In particular, we use the atomic BSM probability of success (p_b) to be 0.4 and latency (t_b) to be 10 μs; in some plots, we vary p_b from 0.2 to 0.6. The optical BSM probability of success (p_{ob}) is half of p_b. We use atom–photon generation times (t_g) and probability of success (p_g) as 50 μs and 0.33, respectively. Finally, we use photon transmission success probability as $e^{-d/(2\tilde{}L)}$ [18], where L is the channel attenuation length (chosen as 20 km for an optical fiber) and d is the distance between the nodes.

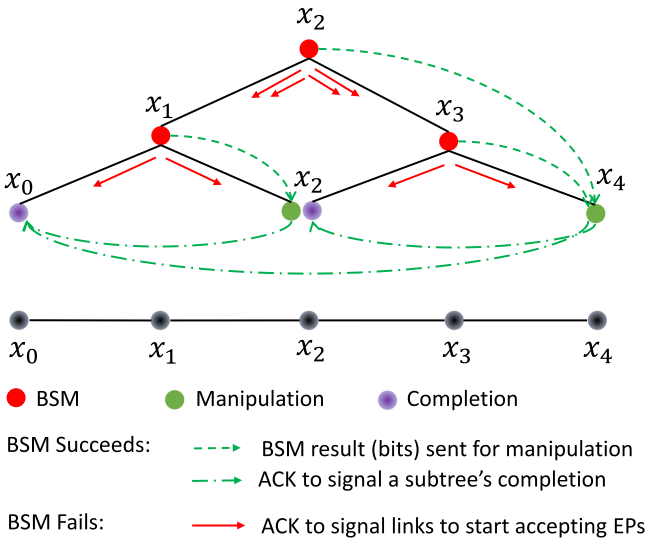

FIGURE 6. Illustration of the swapping tree protocol. The shown tree is not a swapping tree, but rather a certain hierarchy of nodes to illustrate the BSM operation in the swapping tree protocol. A link layer protocol continuously generates EPs over links (x_0, x_2) and (x_2, x_4). On receiving EP on links on either side, x_1 (x_3) attempts a BSM operation on the stored qubit atoms. If the BSM succeeds, x_1 (x_3) sends two classical bits (solid green arrows) to x_2 (x_4) for desired manipulation/correction after which x_2 (x_4) sends an ACK (dashed green arrows) to the other end node x_0 (x_2) to complete the EP generation. If BSM at x_1 and x_3 are both successful, then x_2 attempts the BSM as above. If a BSM at say x_1 fails, then x_1 failure signals (red arrows) to all the descendant nodes of the subtree rooted at x_1 so that they can start accepting new EPs from the link layer protocol. Note that, here, node x_2 plays multiple roles and, hence, appears at multiple places in the figure.

Each node's memory size is randomly chosen within a range of 15–20 units. Fidelity is modeled in NetSquid using two parameter values, viz., depolarization (for decoherence) and dephasing (for operation-driven) rates. We choose a decoherence time of 2 s based on achievable values with single-atom memory platforms [48]; note that decoherence times of even several minutes [36], [37] to hours [31], [38] has been demonstrated for other applicable memory platforms. Accordingly, we choose a depolarization rate of 0.01 such that the fidelity after a second is 90%. Similarly, we choose a dephasing rate of 1000, which corresponds to a link EP fidelity of 99.5% [15].

C. ALGORITHMS AND PERFORMANCE METRICS

To compare our techniques with prior approaches, we implement most recently proposed approaches, viz., 1) the `WaitLess`-based LP approach from [15] (called `Delft-LP` here) and 2) `Q-Cast` approach from [14] which is `WaitLess`-based but uses multiple links and requires memories. The `Waiting`-based algorithm by Caleffi [18] uses an exponential-time approach and is, thus, compared only for small networks. The approaches in [16] and [17] are not compared as they were found to be inferior to `Q-Cast`.

For all the algorithms except `Q-Cast`, we use only one link between adjacent nodes, since only `Q-Cast` takes advantage of multiple links in a creative way. In particular, for `Q-Cast`, we use $W = 1, 5$, or 10 sublinks (Shi and Qian[14] call them channels) on each link, with the node and link "capacity" divided equally among them. We note that in `Q-Cast`, each node requires $2W$ memories (two for each sublink) with sufficient coherence time to allow for the entire swapping operation over the path to be completed. The `Delft-LP` approach explicitly assumes that the generation of link EPs is deterministic, i.e., the value $p_g{}^2 p_e{}^2 p_{ob}$ is 1, and does not model node generation rates. We address these differences by extending their LP formulation: 1) we add a constraint on node generation rates and 2) add a $p_g{}^2 p_e(i, j)^2 p_{ob}$ factor to each link (i, j) in any path extracted from their LP solution.

Among our schemes, we use `DP-OPT`, `DP-Approx`, and `Balanced-Tree` (see Section IV-B) for the `QNR-SP` problem, and `LP` (see Appendix A) and `ITER` schemes for the `QNR` problem. For `ITER`, we use three schemes, i.e., `ITER-DPA`, `ITER-Bal`, and `ITER-SP`, which iterate over `DP-Approx`, `Balanced-Tree`, and `SP`, respectively. To be comprehensive, we also implement a simple `SP` algorithm, which picks a balanced swapping tree over the shortest path (minimum number of links). We compare the schemes largely in terms of EP generation rates; we also compare the execution times and EP fidelity.

D. COMPARISON WITH [18] FOR THE `QNR-SP` PROBLEM

Note that Caleffi [18] gives only the `QNR-SP` algorithm referred to as `Caleffi`; it takes exponential time making it infeasible to run for network sizes much larger than 15–20. In particular, for network sizes 17–20, it takes several hours, and our preliminary analysis suggests that it will take of the order of 10^{40} *years* on our 100-node network (see Appendix E). Thus, we use a small network of 15 nodes over a 25 km $\times$ 25 km area; we consider average node degrees of 3 or 6 (see Fig. 9). We see that `DP-OPT` outperforms `Caleffi` by 10% on average for the sparser graph and minimally for the denser graph. However, for some instances, `DP-OPT` outperformed `Caleffi` by as much as 300% (see Appendix F). We see that `DP-Approx` performs similar to `DP-OPT`, while `Balanced-Tree` is outperformed slightly by `Caleffi`; however, for this small network, since the `DP-OPT` and `DP-Approx` algorithms only take tens of hundreds of milliseconds (see Appendix E), `Balanced-Tree` need not be used in practice.

E. `QNR-SP` PROBLEM (SINGLE TREE) RESULTS

We start with comparing various schemes for the `QNR-SP` problem, in terms of EP generation rate. We compare `DP-Approx`, `DP-OPT`, `Balanced-Tree`, `SP`, and `Q-Cast`; note that the LP schemes cannot be used to select a *single* tree, as they turn into ILPs (see Fig. 7), where we plot the EP generation rate for various schemes for varying number of nodes, (s, d) distance, p_b, and network link density. We observe that `DP-Approx` and `DP-OPT` perform very closely, with the `Balanced-Tree` heuristic performing close to them; all these three schemes outperform the

IEEE Transactions on Quantum Engineering

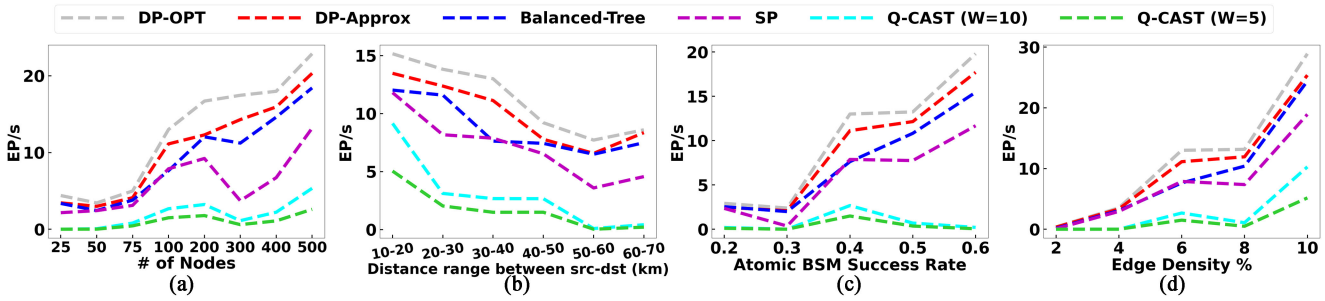

FIGURE 7. (a)–(d) QNR-SP problem: EP generation rates for varying parameters.

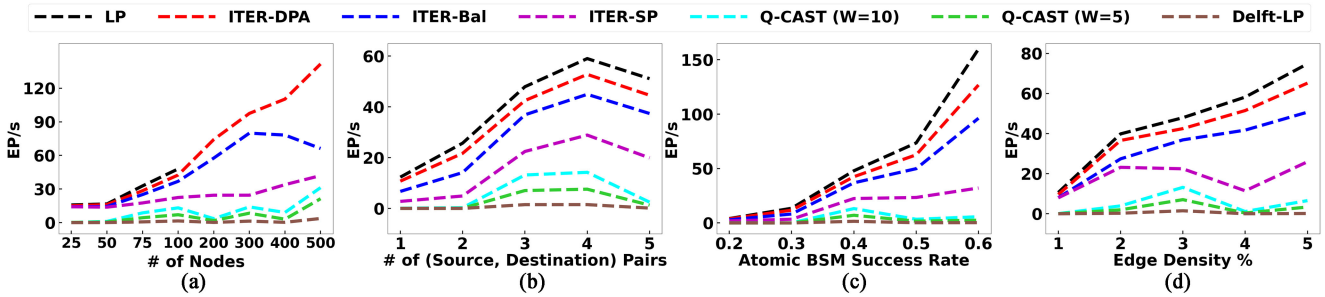

FIGURE 8. (a)–(d) QNR problem: EP generation rates for varying parameters.

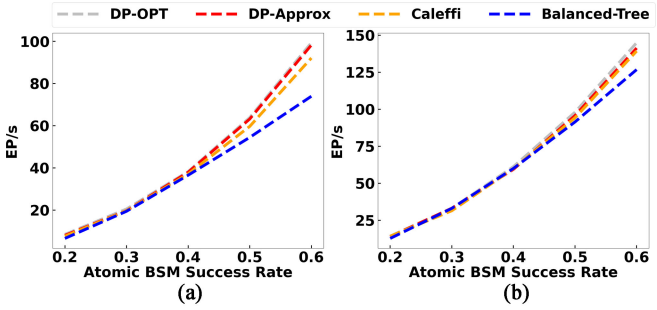

FIGURE 9. Compare the performance with Caleffi in a (a) low-density network and (b) high-density network.

Q-Cast schemes (for $W = 5, 10$ sublinks) by an order of magnitude. We do not plot Q-Cast for $W = 1$ sublinks, as it performs much worse (less than 10^{-3} EP/s). We note that Q-Cast's EP rates here are much lower than the ones published in [14], because Shi and Qian [14] uses link EP success probability of 0.1 or more, while in our more realistic model, the link EP success probability is $p_g{}^2 p_e{}^2 p_{ob} = 0.012$ for the default p_b value. We reiterate that our schemes require only two memory units per node, while the Q-Cast schemes requires $2W$ units. The main reason for poor performance of Q-Cast (in spite of higher memory and link synchronization) is that, in the WaitLess model, the EP generation over a path is a very low probability event—essentially p^l where p is the link-EP success probability and l is the path length, for the case of $W = 1$ (the analysis for higher W's is involved [14]). Finally, our proposed techniques also outperform the SP algorithm, especially when the number of possible paths (trees) between (s, d) pair increases. In addition, we see that performance increases with increase in p_b, number of nodes, or network link density, as expected due to the availability of better trees/paths; it also increases with the decrease in (s, d) distance as fewer hops are needed.

F. QNR PROBLEM RESULTS

We now present performance comparison of various schemes for the QNR problem. Here, we compare the following schemes: ITER-DPA, ITER-Bal, ITER-SP, Delft-LP, and Q-Cast with the optimal LP as the benchmark for comparison (LP was not feasible to run for more than 100 nodes); see Fig. 8. Our observations are similar to that for the QNR-SP problem results. We see that in all the plots, LP being optimal performs the best, but is closely matched by ITER-DPA and the efficient heuristic ITER-Bal. We observe that the performance gap between our proposed techniques and ITER-SP is higher than in the QNR-SP case, as SP picks paths based on just the number of links. Our schemes outperform both Delft-LP and Q-Cast by an order of magnitude, for the same reason as mentioned above.

G. FIDELITY AND LONG-DISTANCE ENTANGLEMENTS

We now investigate the fidelity of the EPs generated. First, we note that the Q-Cast and Delft-LP schemes will incur near-zero decoherence as they involve only transient storage. Decoherence for other schemes is also negligible as the EP generation latencies (tens of milliseconds) is much less than

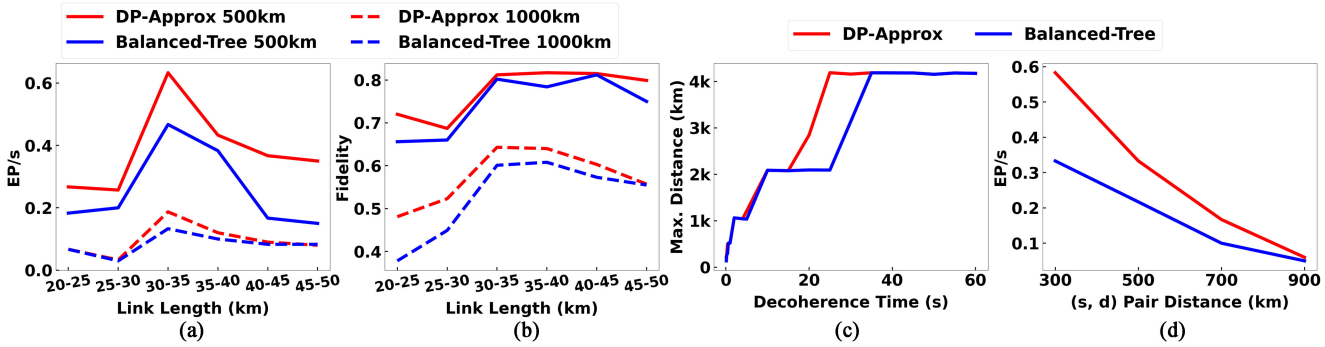

FIGURE 10. EP generation over linear paths. (a) EP rates and (b) fidelity over linear paths with varying link lengths. (c) Maximum reachable distance with links of 30–35 m lengths. (d) EP generation rates over linear paths with 10–50 km links to demonstrate the impact of varying link lengths.

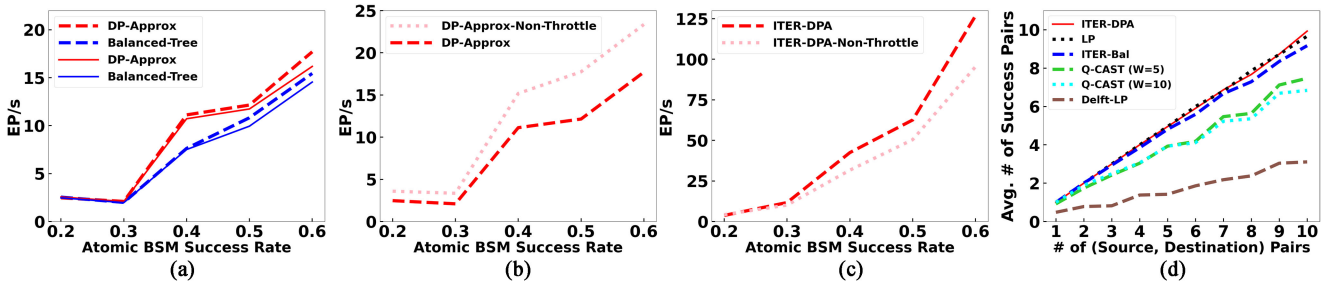

FIGURE 11. Comparing with analytical results. (a) Analytical versus simulation results. (b) Throttled versus nonthrottled trees (`QNR-SP`). (c) Throttled versus nonthrottled trees (`QNR`). (d) Fairness measure.

the coherence time. The operation-driven fidelity loss is expected to be similar for all schemes, as they all roughly use the same order of links. Overall, we observed fidelities of 94–97% across all schemes (not shown), with our schemes also performing better sometimes due to smaller number of leaves.

1) LONG PATH GRAPHS

To test the limits of the schemes in terms of decoherence and fidelity, we consider a long path network and estimate the fidelity of EPs generated by schemes for increasing distances and link lengths (link success probability decreases with increasing link length). Fig. 10(a) and (b) shows EP generation rates and fidelity for path lengths of 500 and 1000 km for varying link lengths, for the single-tree schemes `DP-Approx` and `Balanced-Tree`. `Q-Cast` and `Delft-LP` are not shown as their EP rate is near zero ($\leq 10^{-20}$) at these distances. We observe that our schemes yield EPs with qubit fidelities of 65–82% and 40–64% for 500 and 1000 km paths, respectively, with EP rates of 0.05–0.65 s^{-1}. These are viable results—since qubit copies with fidelities higher than 50% can be purified to smaller copies with arbitrarily higher fidelities [49], [50].

Now, in Fig. 10(c), we demonstrate the effect of decoherence time of quantum memories used in nodes. Here, we use 30–35 km links. We see that even with decoherence time of as low as 100 ms, `DP-Approx` is able to create EPs for up to 200 km, while `Balanced-Tree` can only create EP for paths up to 120 km; they perform similarly for larger decoherence times. As all the links are almost of the same length, the optimal swapping will be largely balanced trees, wherein the EP generation rate depends only on the tree height. Owing to this reason, the maximum achievable path length graph is close to a step function. We add that our schemes produce 0.008 EPs/s for distance of more than 4000 km.

Finally, in Fig. 10(d), we demonstrate the higher performance of nonbalanced trees when the links on a path may have much different lengths. In particular, we pick link lengths randomly in the range of 10–50 km. With this setting, we see that `DP-Approx` performs much better than `Balanced-Tree` and, in some cases, up to 100% better. Note that `Balanced-Tree` and `Caleffi` have similar performance over linear graphs, as there is no path selection scheme needed.

H. VALIDATING THE ANALYSIS: FAIRNESS

Fig. 11(a) compares the EP generation rates as measured by the analytical formulae and the actual simulations for the `QNR-SP` algorithms `DP-Approx` and `Balanced-Tree`. We observe that they match closely, validating our assumption of 3/2 factor in (1) and of exponential distributions at higher levels of the tree, and of the path metric $M()$ for `Balanced-Tree`. Fig. 11(b) and (c) plots the EP generation rates for throttled and nonthrottled trees. We see that the throttled tree underperforms the nonthrottled tree by only a

small margin for the single-tree case; however, for the multi-tree ITER-Bal algorithm, the throttled trees perform better as they are able to use the resources efficiently. Fig. 11(d) plots the average number of (s, d) pairs that get at least one tree/path for varying number of requests; we see that our schemes exhibit 90–99% fairness.

I. EXECUTION TIMES

We ran our simulations on an Intel i7-8700 CPU machine and observed that the WaitLess algorithms as well our Balanced-Tree and ITER-Bal heuristics run in fraction of a second even for a 500-node network; thus, they can be used in real time. Note that since our problems depend on real-time network state (residual capacities), the algorithms must run very fast. The other algorithms (viz., DP-OPT, DP-Approx, and ITER-DPA) can take minutes to hours on large networks and, hence, may be impractical on large network without significant optimization and/or parallelization. See Appendix G for the plot.

VIII. CONCLUSION

In this article, we designed techniques for an efficient generation of EP to facilitate QN communication, by selecting efficient swapping trees in a Waiting protocol. By extensive simulations, we demonstrated the effectiveness of our techniques and their viability in generating high-fidelity EP over long distances (500–1000 km). Our future work is focused on exploring more sophisticated generation structures, e.g., aggregated trees, taking advantage of pipelining across rounds, incorporating purification techniques, and extending our techniques to multimode memories [51], [52].

APPENDIX A
LP FORMULATION FOR THE QNR PROBLEM

In this appendix, we provide an optimal LP-based solution to the QNR problem. Although polynomial time, this solution has high complexity, so its main use is as a benchmark in evaluating the more efficient (but possibly suboptimal) algorithms for the problem.

Our approach follows from the observation that each swapping tree in a QN can be viewed as a special kind of path (called *B-hyperpath* [53]) over a hypergraph constructed from the network graph. We begin by describing the hypergraph construction for the single-pair case and ignoring fidelity constraints. We then extend traditional hypergraph flow algorithm to incorporate losses (e.g., due to BSM failures), stochasticity, and the interaction between memory constraints and stochasticity. Finally, we extend the formulation to multiple (s, d) pairs and incorporate fidelity constraints.

The optimal generation of long-distance entanglement was posed as an LP problem in [54], but differs from our more general formulation work in three main ways. First of all, Dai *et al.* [54] assume unbounded memory capacity at each swapping node to queue up incoming EPs. In contrast, our model has bounded memory capacity at each node, and consequently, our LP formulation deals with *expectations* over rates/latencies rather than scalar rate values. Second, our formulation accounts for node capacity constraints in addition to link constraints. Third, our formulation poses the problem in terms of hypergraph flows, which permits us to easily incorporate fidelity and decoherence constraints.

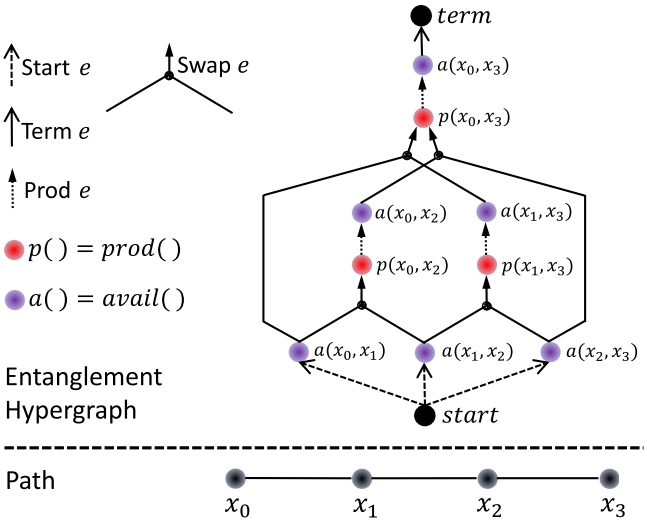

FIGURE 12. ST-hypergraph for a four-node linear network. Not all *prod* nodes are shown.

A. HYPERGRAPH-BASED REPRESENTATION OF ENTANGLEMENT GENERATION

We begin by recalling standard hypergraph notions [53], [55], [56].

Definition 2 (Hypergraph): A directed hypergraph $H = (V(H), E(H))$ has a set of vertices $V(H)$ and a set of (directed) *hyperarcs* $E(H)$, where each hyperarc e is a pair $(t(e), h(e))$ of nonempty disjoint subsets of $V(H)$. A *weighted* hypergraph is additionally equipped with a weight function $\omega : E(H) \to R^+$.

Sets $t(e)$ and $h(e)$ are called the *tail* and *head*, respectively, of hyperarc $(t(e), h(e))$. A hyperarc e is a *trivial* edge if both $t(e)$ and $h(e)$ are singleton and *nontrivial* otherwise. A hyperarc e, where $|h(e)| = 1$, i.e., whose head is singleton, is called a B-arc. A hypergraph consisting only of B-arcs is called a B-hypergraph.

Definition 3 (Connectivity and B-hyperpaths): A vertex t is *B-connected to* vertex s in hypergraph H if $s = t$ or there is a hyperarc $e \in E(H)$ such that $h(e) = \{t\}$ and every $v \in t(e)$ is B-connected to s in H. A B-hyperpath from s to t is a minimal B-hypergraph P such that $V(P) \subseteq V(H)$, $E(P) \subseteq E(H)$, and t is B-connected to s in P.

ST-hypergraph: Given a QN and single (s, d) pair, we first construct a hypergraph that represents the set of all possible swapping trees rooted at (s, d). Given a QN represented as an undirected graph $G = (V, E)$ and a single (s, d) pair, its *ST-hypergraph* is a hypergraph H constructed as follows (see Fig. 12). All pairs of the following vertices are unordered pairs.

1) $V(H)$ consists of:
   a) two distinguished vertices *start* and *term*;
   b) $prod(u, v)$ and $avail(u, v)$ for all distinct $u, v \in V$.
2) $E(H)$ consists of five types of hyperarcs.
   a) [Start] $e = (\{start\}, \{avail(u, v)\})\ \forall u, v \in V$.
   b) [Swap] $e = (\{avail(u, w), avail(w, v)\}, \{prod(u, v)\})$, for all *distinct* $u, v, w \in V$.
   c) [Prod] $e = (\{prod(u, v)\}, \{avail(u, v)\})\ \forall u, v \in V$.
   d) [Term] $e = (\{avail(s, d)\}, \{term\})$.

In an ST-hypergraph, vertices *start* and *term* represent source and sink nodes of a desired hypergraph flow (see the following). Other vertices represent EPs over a pair of nodes in G. Hyperarcs represent how the tail EPs contribute to that at the head. For ease of accounting, we categorize generated EPs using different types of vertices: *start* represents link-level EPs generated over links in G, *prod* represents EPs produced by atomic ES, and *avail* represents EPs generated from either of the above. "Start" and "Prod" arcs turn the *start* and *prod* EPs, respectively, into *avail* EPs and, thus, make them available for further swapping. "Swap" arcs represent swapping over the triplets of nodes (u, w, v). Note that an ST-hypergraph is a B-hypergraph, as "Swaps" are the only nontrivial hyperarcs, and their head is singleton.

Swapping trees as B-hyperpaths: Given a QNR problem with a single pair (s, d), it is easy to see that any swapping tree generating (s, t) EPs can be represented by a unique B-hyperpath from *start* to *term* in the above ST-hypergraph. Thus, it easily follows that a QNR problem of selection of (multiple) swapping trees is equivalent to finding an optimal hypergraph flow from *start* to *term* in H. Note that H has $O(|V|^2)$ vertices and $O(|V^3|)$ hyperarcs.

B. ENTANGLEMENT FLOW AS LP

We now develop an LP formulation to represent the QNR problem over (s, d) in G as a hypergraph flow problem in H. In contrast to the classic hypergraph flow formulation [53], we need to consider *lossy* flow, with loss arising from two sources: 1) ES operations have a given success probability, and 2) waiting for both qubits to arrive before performing ES leads to losses since the arrival of EPs follow independent probability distributions. For the latter, we make use of Observation 1. The proposed LP formulation is as follows.

1) *Variables*: z_a, for each hyperarc a in H, represents the EP generation rate over each of the (one or two) node pairs in a's tail. This enforces the condition that EP rates over the two node pairs in *prod* hyperarc's tail are equal. Thus, the LP solution will result in *throttled* swapping trees.
2) *Capacity constraints*: $z_a \in R^+$ for all hyperarcs a in H. We use (2) to add the following constraints due to nodes in G:

$$1/t_g \quad \geq \quad \sum_{x \in E(i)} z_{a(x)}/({p_g}^2 {p_e}^2 p_{ob}) \quad \forall i \in V.$$

Above $a(x)$ is the hyperarc in H of the form $(start, avail(x))$, where x is an edge in G.
3) *Flow constraints* that vary with vertex types. In the following, we use notations $out(v)$ and $in(v)$ to represent outgoing and incoming hyperarcs from v. Formally, $out(v)$ is $\{a \in E(H) : v \in t(a)\}$ and $in(v)$ is $\{a \in E(H) : v \in h(a)\}$.
   1) For each vertex v s.t. $v = avail(\cdot)$:

$$\sum_{a \in in(v)} z_a = \sum_{a' \in out(v)} z_{a'}.$$

   That is, there is no loss in making already generated entanglements available for further swapping.
   2) For each vertex v s.t. $v = prod(\cdot)$

$$\sum_{a \in in(v)} z_a p_b (2/3) = \sum_{a' \in out(v)} z_{a'}.$$

   The $(2/3)p_b$ factor follows from Observation 1 and accounts for loss due to swapping failures as well as due to waiting for the arrival of both EPs for swapping.
4) *Objective*: Maximize $\sum_{a \in in(term)} z_a$.

Multiple-pair multipath: The above LP formulation for the single-pair QNR problem can be readily extended to the multiple-pair case. Let $\{(s_1, d_1), (s_2, d_2), \ldots, (s_n, d_n)\}$ be a set of source–destination pairs. The *only change* is that the hypergraph H now has n arcs $(\{avail(s_i, d_i)\}, \{term\})$ for all i. The other arcs model the generation of EPRs independent of the pairs and, thus, are unchanged. It is interesting to note that the multipair problem, typically formulated as multicommodity flow in classical networks, is posed here as single-commodity flow over hypergraphs.

C. FIDELITY

Constraints on loss of fidelity due to noisy BSM operations and from decoherence due to the age of qubits can be added to the LP formulation, as follows. Recall that the constraint on operation-based fidelity loss is modeled by limiting the number leaves of the swapping tree, and in Section IV-B, we formulated the decoherence constraint by limiting the heights of the left-most and right-most descendants of the root's children. These *structural* constraints on swapping trees can be lifted to the LP formulation by 1) adding the leaf count and heights as parameters to *prod* and *avail* vertices and 2) swapping the EPs generated from only the compatible vertices.

In particular, we generalize the ST-hypergraph to a *fidelity-constrained* one called $H^{(F)}$, where the *prod* and *avail* vertices are parameterized by $u, v \in V$, and in addition by (n, h), where n is the number of leaves and

$h = (h_{ll}, h_{lr}, h_{rl}, h_{rr})$ represents the depths of left-most and right-most descendants of the root's children, of the trees rooted at (u, v) with those parameter values. In terms of edges, the most interesting difference $H^{(F)}$ and H is in "Swap" edges. In $H^{(F)}$, "Swap" edges are $(\{avail(u, w, n', h')avail(w, v, n'', h'')\}, \{prod(u, v, n, h)\})$ only if $n = n' + n''$, and h, h', h'' are such that $h_{ll} = h'_{ll} + 1$, $h_{lr} = h'_{rr} + 1$, $h_{rl} = h''_{ll} + 1$, and $h_{rr} = h''_{rr} + 1$. The above constraints ensure that only compatible subtrees are composed into bigger trees. The other changes are for bookkeeping: "Gen" are from $gen(u, v)$ to $avail(u, v, 1, (0, 0, 0, 0))$; "Prod" are from $prod(u, v, n, h)$ to $avail(u, v, n, h)$; and finally "Term" are from $avail(s, t, n, h)$ to $term$ for $n \leq \tau_l$ and h such that $f(h) \leq \tau_d$; here, $f(h)$ gives the tree's age based on h values (following Section IV-B) while using the link rates based on 50% node capacity usage.

APPENDIX B
PROOF OF THEOREM 1

Proof (sketch): We provide a main intuition behind the claim in Theorem 1. The key claim is that at any instant, the `Wait-Less` protocol generates an EP and the `Waiting` protocol will also be able to generate an EP. Consider an instant t in time when the `WaitLess` protocol X generates an EP, as a result of all the underlying processes succeeding at time t. Right before time t, consider the state of the EPs in the swapping tree T of the `Waiting` protocol Y: Essentially, some of the nodes in T have (generated) EPs that are waiting for their sibling EP to be generated; note that these generated EPs have not aged yet, else they would have been already discarded by Y. Now, at time t, during X's execution, all the underlying processes succeed instantly—it is easy to see that in the protocol Y too, all the ungenerated EP would now be generated instantly[11]—yielding a full EP at the root (using qubits that have not aged beyond the threshold). Finally, since the number of operations in T is the same as the number of BSM operations incurred by X to generate an EP, the fidelity degradation due to operations is the same in both the protocols.

APPENDIX C
PROOF OF LEMMA 1

Proof: We first prove the claim that given any swapping tree $\mathcal{T}_{xy}$ over a path $P : x \rightsquigarrow w \rightsquigarrow y$, there exists a swapping tree $\mathcal{T}_{xw}$ over a path $P' : x \rightsquigarrow w$ such that P' is a subset of P and generation latency of $\mathcal{T}_{xw}$ is less than that of $\mathcal{T}_{xy}$. This claim can be easily proved by induction as follows. Consider the following cases.

1) w is the root of $\mathcal{T}_{xy}$, in which case T_{xw} is the left child of the root.
2) i and w have a common ancestor a that is other than the root of $\mathcal{T}_{xy}$. In this case, $a = w$, and the subtree rooted at $a = w$ is the required $\mathcal{T}_{xw}$.
3) The only common ancestor of i and w is the root a of $\mathcal{T}_{xw}$, which is not w. In this case, we apply the inductive hypothesis on right subtree $\mathcal{T}_{ay}$ of $\mathcal{T}_{xy}$, to extract a subtree $\mathcal{T}_{aw}$, which along with the left right subtree $\mathcal{T}_{ia}$ of $\mathcal{T}_{xy}$ gives the required subtree $\mathcal{T}_{xw}$. This proves the above claim.

Now, to prove the lemma, let us consider the swapping trees $\mathcal{T}_{ik}$ and $\mathcal{T}_{kj}$ given to us. By the above claim, there are swapping trees $\mathcal{T}_{iv}$ and $\mathcal{T}_{vj}$, which will satisfy the requirements of the given lemma's claim.

[11]Here, we have implicitly assumed that if n BSM operations succeed in X protocol at some instant t, then at the same instant, n BSM operations anywhere in Y will also succeed.

TABLE I Execution Times of the `QNR-SP` Algorithm Over Small Networks

Algorithm	Number of nodes					
	10	13	15	16	18	20
`Balanced-Tree`	239 μs	360 μs	373 μs	492 μs	530 μs	552 μs
`DP-Approx`	4 ms	10 ms	14.7 ms	17.6 ms	28 ms	34 ms
`DP-OPT`	148 ms	363 ms	572 ms	706 ms	1 s	1.7 s
Caleffi [18]	92 ms	4.6 s	14 s	26 min	3.2 h	12.8 h

APPENDIX D
PROOF OF THEOREM 2

Proof: We show that $T[i, j, h, u_i, u_j]$ is indeed the optimal latency over the nodes (i, j) using a throttled swapping tree of height at most h and with u_j and u_j as the usage percentages at nodes i and j. We use proof by induction over h. The base case is obvious. The inductive hypothesis is that the above statement is true for all heights $\leq (h - 1)$. Now, let $\mathcal{T}$ be an optimal-latency swapping tree of height at most h between a pair of nodes (i, j), for some height greater than 1, and node usage percentages at i and j of u_i and u_j respectively. Let the expected latency of $\mathcal{T}$ be L. Let the two children subtrees of the root of $\mathcal{T}$ be $\mathcal{T}_1$ and $\mathcal{T}_2$, each of latency L_c; note that, as $\mathcal{T}$ is throttled, the expected latencies of $\mathcal{T}_1$ and $\mathcal{T}_2$ are equal. Thus, we have $L_c = (\frac{3}{2}L + t_c + t_b)/p_b)$ by (1). Note that $\mathcal{T}_1$ and $\mathcal{T}_2$ are of heights at most $h - 1$, and without loss of generality, we can assume $\mathcal{T}_1$ and $\mathcal{T}_2$ to be disjoint (as per Lemma 1). Let $\mathcal{T}_1$ and $\mathcal{T}_2$ be between the pairs of nodes (i, k) and (k, j) with end node usage percentages of (u_i, u_k) and (u'_k, u_j), respectively. Now, optimal throttled trees over (i, k) and (k, j) must have a latency of at most that of $\mathcal{T}_1$ and $\mathcal{T}_2$, i.e., L_c. Finally, by (4) and the inductive hypothesis, we have that $T[i, j, h, u_i, u_j]$ (and throttled) will be at most L.

APPENDIX E
EXECUTION TIMES OF CALEFFI [18] ALGORITHM

Here, we give the execution times of different algorithms especially `Caleffi`'s for small networks of 10–20 nodes (see Table I). We see that `Balanced-Tree` and `DP-Approx` take fractions of a second, while `DP-OPT` takes up to 2 s. However, as expected, `Caleffi`'s execution time increases

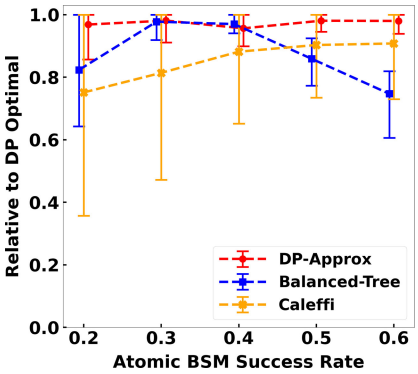

FIGURE 13. Compare the performance with `Caleffi` relative to `DP-OPT` (the closer to 1, the better).

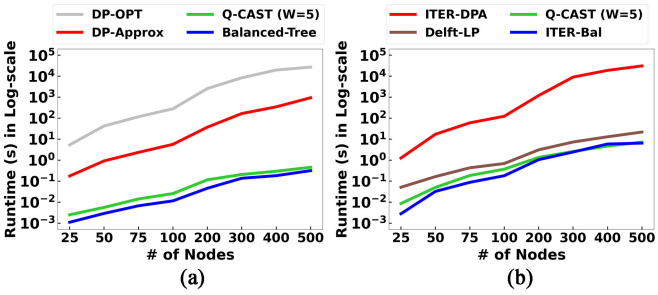

FIGURE 14. Execution time comparison of various algorithms for `QNR-SP` and `QNR` algorithms.

exponentially with the increase in the number of nodes—with 20-node network taking more than 10 h. In the following, we further estimate `Caleffi`'s execution time for larger graphs.

Rough estimate of `Caleffi`*'s execution time for large graphs.* Consider an n-node network with an average node degree of d. Consider a node pair (s, d). We try to estimate the number of paths from s to d—the goal here is merely to show that the number is astronomical for $n = 100$, and thus, our analysis is very approximate (more accurate analysis seems beyond the scope of this article). Let $P(l)$ be the number of simple paths from s to a node x in the graph of length at most l. For large graphs and large l, we can assume $P(l)$ to be roughly same for all x. We estimate that $P(l+1) = P(l) + P(l) * 6 * (1 - l/n)$. The first term is to count paths of length at most $l - 1$; in the second term, the factor 6 comes from the fact that the destination x has six neighbors and the factor $(1 - l/n)$ is the probability that a path counted in $P(l)$ does not contain x (to constrain the paths to be simple, i.e., without cycles). Using $P()$, the execution time of `Caleffi` can be roughly estimated to be at least $P(n-1) * 500/(5 * 10^9)$ s, where the factor 500 is a conservative estimate of the number of instructions used in computing the latency for a path and 5×10^9 is the number of instructions a 5-GHz machine can execute in a second. The above yields executions times of a few seconds for $n = 15$, about an hour for $n = 20$, about 350 hour for $n = 25$, 10^{16} hour for $n = 50$, and 10^{44} hour for $n = 100$. The above estimates for $n = 15$–20 are within an order of magnitude of our actual execution times and, thus, validate our estimation approach.

APPENDIX F
COMPARISON WITH `Caleffi`: MORE DETAILS

Fig. 9 shows that `DP-OPT` outperforms `Caleffi` by a margin of around 10% when averaging multiple experiments. However, when we look at one experiment at a time and compute the `Caleffi`'s performance relative to `DP-OPT` for each experiment, we see a larger difference between `DP-OPT` and `Caleffi`. Fig. 13 plots the error bar of the relative performance of three algorithms compared to `DP-OPT` at each experiment. The lower cap of `Caleffi` at 0.2 atomic BSM success rate is 0.35, which means that at an extreme sample, the `DP-OPT` is almost 300% better than `Caleffi`. In that extreme sample, the number of hops between the source and the destination is large (thus, the overall EP rate is small, which affects little when averaging with other experiments in Fig. 9). Moreover, we observe that the larger the number of hops between the source and the destination, the larger the gap of relative performance between `DP-OPT` and `Caleffi`. This observation aligns with what is shown in Fig. 7(b): our `DP-OPT` has an larger advantage in ratio when the source and destination are far away.

APPENDIX G
EXECUTION TIMES PLOT

We give here the plot for execution times of various schemes (see Fig. 14).

REFERENCES

[1] F. Arute *et al.*, "Quantum supremacy using a programmable superconducting processor," *Nature*, vol. 574, pp. 505–510, 2019, doi: 10.1038/s41586-019-1666-5.

[2] J. Gambetta, "IBM's roadmap for scaling quantum technology," 2020. [Online]. Available: https://www.ibm.com/blogs/research/2020/09/ibm-quantum-roadmap/

[3] M. Caleffi, A. S. Cacciapuoti, and G. Bianchi, "Quantum Internet: From communication to distributed computing!," in *Proc. 5th ACM Int. Conf. Nanoscale Comput. Commun.*, 2018, pp. 1–4, doi: 10.1145/3233188.3233224.

[4] C. Simon, "Towards a global quantum network," *Nature Photon.*, vol. 11, no. 11, pp. 678–680, 2017, doi: 10.1038/s41566-017-0032-0.

[5] Z. Eldredge, M. Foss-Feig, J. A. Gross, S. L. Rolston, and A. V. Gorshkov, "Optimal and secure measurement protocols for quantum sensor networks," *Phys. Rev. A*, vol. 97, no. 4, 2018, Art. no. 042337, doi: 10.1103/PhysRevA.97.042337.

[6] V. Scarani, H. Bechmann-Pasquinucci, N. J. Cerf, M. Dušek, N. Lütkenhaus, and M. Peev, "The security of practical quantum key distribution," *Rev. Modern Phys.*, vol. 81, no. 3, pp. 1301–1350, 2009, doi: 10.1103/RevModPhys.81.1301.

[7] P. Kómár *et al.*, "A quantum network of clocks," *Nature Phys.*, vol. 10, no. 8, pp. 582–587, 2014, doi: 10.1038/nphys3000.

[8] T.-Y. Chen *et al.*, "Field test of a practical secure communication network with decoy-state quantum cryptography," *Opt. Exp.*, vol. 17, no. 8, pp. 6540–6549, 2009, doi: 10.1364/OE.17.006540.

[9] M. Marcozzi and L. Mostarda, "Quantum consensus: An overview," 2021, *arXiv:2101.04192*, doi: 10.48550/arXiv.2101.04192.

[10] D. Dieks, "Communication by EPR devices," *Phys. Lett. A*, vol. 92, no. 6, pp. 271–272, Nov. 1982, doi: 10.1016/0375-9601(82)90084-6.

[11] C. H. Bennett, G. Brassard, C. Crépeau, R. Jozsa, A. Peres, and W. K. Wootters, "Teleporting an unknown quantum state via dual classical and Einstein-Podolsky-Rosen channels," *Phys. Rev. Lett.*, vol. 70, no. 13, pp. 1895–1899, 1993, doi: 10.1103/PhysRevLett.70.1895.

[12] L.-M. Duan, M. D. Lukin, J. I. Cirac, and P. Zoller, "Long-distance quantum communication with atomic ensembles and linear optics," *Nature*, vol. 414, no. 6862, pp. 413–418, 2001, doi: 10.1038/35106500.

[13] N. Sangouard, C. Simon, H. De Riedmatten, and N. Gisin, "Quantum repeaters based on atomic ensembles and linear optics," *Rev. Modern Phys.*, vol. 83, no. 1, pp. 33–80, 2011, doi: 10.1103/RevModPhys.83.33.

[14] S. Shi and C. Qian, "Concurrent entanglement routing for quantum networks: Model and designs," in *Proc. Annu. Conf. ACM Special Int. Group Data Commun. Appl., Technol., Archit., Protoc. Comput. Commun.*, 2020, pp. 62–75, doi: 10.1145/3387514.3405853.

[15] K. Chakraborty, D. Elkouss, B. Rijsman, and S. Wehner, "Entanglement distribution in a quantum network: A multicommodity flow-based approach," *IEEE Trans. Quantum Eng.*, vol. 1, 2020, Art. no. 4101321, doi: 10.1109/TQE.2020.3028172.

[16] M. Pant *et al.*, "Routing entanglement in the quantum internet," *NPJ Quantum Inf.*, vol. 5, no. 1, pp. 1–9, 2019, doi: 10.1038/s41534-019-0139-x.

[17] K. Chakraborty, F. Rozpedek, A. Dahlberg, and S. Wehner, "Distributed routing in a quantum internet," 2019, *arXiv:1907.11630*, doi: 10.48550/arXiv.1907.11630.

[18] M. Caleffi, "Optimal routing for quantum networks," *IEEE Access*, vol. 5, pp. 22 299–22 312, 2017, doi: 10.1109/ACCESS.2017.2763325.

[19] O. Gühne and G. Tóth, "Entanglement detection," *Phys. Rep.*, vol. 474, nos. 1–6, pp. 1–75, 2009, doi: 10.1016/j.physrep.2009.02.004.

[20] W. K. Wootters and W. H. Zurek, "A single quantum cannot be cloned," *Nature*, vol. 299, pp. 802–803, 1982, doi: 10.1038/299802a0.

[21] S. Muralidharan, L. Li, J. Kim, N. Lütkenhaus, M. D. Lukin, and L. Jiang, "Optimal architectures for long distance quantum communication," *Sci. Rep.*, vol. 6, no. 1, 2016, Art. no. 20463, doi: 10.1038/srep20463.

[22] S. J. Devitt, W. J. Munro, and K. Nemoto, "Quantum error correction for beginners," *Rep. Prog. Phys.*, vol. 76, no. 7, Jun. 2013, Art. no. 076001, doi: 10.1088/0034-4885/76/7/076001.

[23] H.-J. Briegel, W. Dür, J. I. Cirac, and P. Zoller, "Quantum repeaters: The role of imperfect local operations in quantum communication," *Phys. Rev. Lett.*, vol. 81, pp. 5932–5935, Dec. 1998, doi: 10.1103/PhysRevLett.81.5932.

[24] J. Roffe, "Quantum error correction: An introductory guide," *Contemporary Phys.*, vol. 60, no. 3, pp. 226–245, Jul. 2019, doi: 10.1080/00107514.2019.1667078.

[25] D. Press *et al.*, "Ultrafast optical spin echo in a single quantum dot," *Nature Photon.*, vol. 4, no. 6, pp. 367–370, 2010, doi: 10.1038/nphoton.2010.83.

[26] H. Wang *et al.*, "Towards optimal single-photon sources from polarized microcavities," *Nature Photon.*, vol. 13, no. 11, pp. 770–775, 2019, doi: 10.1038/s41566-019-0494-3.

[27] Y. Sagi, I. Almog, and N. Davidson, "Process tomography of dynamical decoupling in a dense cold atomic ensemble," *Phys. Rev. Lett.*, vol. 105, Jul. 2010, Art. no. 0 53201, doi: 10.1103/PhysRevLett.105.053201.

[28] E. Vetsch, D. Reitz, G. Sagué, R. Schmidt, S. T. Dawkins, and A. Rauschenbeutel, "Optical interface created by laser-cooled atoms trapped in the evanescent field surrounding an optical nanofiber," *Phys. Rev. Lett.*, vol. 104, May 2010, Art. no. 203603, doi: 10.1103/PhysRevLett.104.203603.

[29] C. Deutsch *et al.*, "Spin self-rephasing and very long coherence times in a trapped atomic ensemble," *Phys. Rev. Lett.*, vol. 105, Jul. 2010, Art. no. 020401, doi: 10.1103/PhysRevLett.105.020401.

[30] C. Langer *et al.*, "Long-lived qubit memory using atomic ions," *Phys. Rev. Lett.*, vol. 95, Aug. 2005, Art. no. 060502, doi: 10.1103/PhysRevLett.95.060502.

[31] P. Wang *et al.*, "Single ion qubit with estimated coherence time exceeding one hour," *Nature Commun.*, vol. 12, no. 1, 2021, Art. no. 233, doi: 10.1038/s41467-020-20330-w.

[32] M. K. Bhaskar *et al.*, "Experimental demonstration of memory-enhanced quantum communication," *Nature*, vol. 580, no. 7801, pp. 60–64, 2020, doi: 10.1038/s41586-020-2103-5.

[33] W. Tittel *et al.*, "Photon-echo quantum memory," 2008, *arXiv:0810.0172*, doi: 10.48550/arXiv.0810.0172.

[34] J. J. Longdell, E. Fraval, M. J. Sellars, and N. B. Manson, "Stopped light with storage times greater than one second using electromagnetically induced transparency in a solid," *Phys. Rev. Lett.*, vol. 95, no. 6, 2005, Art. no. 063601, doi: 10.1103/PhysRevLett.95.063601.

[35] E. Fraval, M. J. Sellars, and J. J. Longdell, "Dynamic decoherence control of a solid-state nuclear-quadrupole qubit," *Phys. Rev. Lett.*, vol. 95, no. 3, 2005, Art. no. 030506, doi: 10.1103/PhysRevLett.95.030506.

[36] M. Steger *et al.*, "Quantum information storage for over 180 s using donor spins in a ^{28}Si "semiconductor vacuum"," *Science*, vol. 336, no. 6086, pp. 1280–1283, 2012, doi: 10.1126/science.1217635.

[37] K. Saeedi *et al.*, "Room-temperature quantum bit storage exceeding 39 minutes using ionized donors in silicon-28," *Science*, vol. 342, no. 6160, pp. 830–833, 2013, doi: 10.1126/science.1239584.

[38] M. Zhong *et al.*, "Optically addressable nuclear spins in a solid with a six-hour coherence time," *Nature*, vol. 517, no. 7533, pp. 177–180, 2015, doi: 10.1038/nature14025.

[39] K. Azuma, K. Tamaki, and H.-K. Lo, "All-photonic quantum repeaters," *Nature Commun.*, vol. 6, no. 1, 2015, Art. no. 6787,, doi: 10.1038/ncomms7787.

[40] T. Coopmans, S. Brand, and D. Elkouss, "Improved analytical bounds on delivery times of long-distance entanglement," *Phys. Rev. A*, vol. 105, Jan. 2022, Art. no. 012608, doi: 10.1103/PhysRevA.105.012608.

[41] B. Li, T. Coopmans, and D. Elkouss, "Efficient optimization of cut-offs in quantum repeater chains," in *Proc. IEEE Int. Conf. Quantum Comput. Eng.*, 2020, pp. 158–168, doi: 10.1109/QCE49297.2020.00029.

[42] L. Jiang, J. M. Taylor, N. Khaneja, and M. D. Lukin, "Optimal approach to quantum communication using dynamic programming," *Proc. Nat. Acad. Sci., USA,* vol. 104, no. 44, pp. 17291–17296, 2007, doi: 10.1073/pnas.0703284104.

[43] A. Dahlberg *et al.*, "A link layer protocol for quantum networks," in *Proc. ACM Special Int. Group Data Commun.*, 2019, pp. 159–173, doi: 10.1145/3341302.3342070.

[44] W. Kozlowski, A. Dahlberg, and S. Wehner, "Designing a quantum network protocol," in *Proc. 16th Int. Conf. Emerg. Netw. Experiments Technol.*, 2020, pp. 1–16, doi: 10.1145/3386367.3431293.

[45] L. Bugalho, B. C. Coutinho, and Y. Omar, "Distributing multipartite entanglement over noisy quantum networks," 2021, *arXiv:2103.14759*.

[46] T. Coopmans *et al.*, "NetSquid, a discrete-event simulation platform for quantum networks," *Commun. Phys.*, vol. 4, 2021, Art. no. 164, doi: 10.1038/s42005-021-00647-8.

[47] B. M. Waxman, "Routing of multipoint connections," *IEEE J. Sel. Areas Commun.*, vol. 6, no. 9, pp. 1617–1622, Dec. 1988, doi: 10.1109/49.12889.

[48] P. van Loock *et al.*, "Extending quantum links: Modules for fiber- and memory-based quantum repeaters," *Adv. Quantum Technol.*, vol. 3, no. 11, 2020, Art. no. 1900141, doi: 10.1002/qute.201900141.

[49] C. H. Bennett, G. Brassard, S. Popescu, B. Schumacher, J. A. Smolin, and W. K. Wootters, "Purification of noisy entanglement and faithful teleportation via noisy channels," *Phys. Rev. Lett.*, vol. 76, no. 5, pp. 722–725, 1996, doi: 10.1103/PhysRevLett.76.722.

[50] C. H. Bennett, D. P. DiVincenzo, J. A. Smolin, and W. K. Wootters, "Mixed-state entanglement and quantum error correction," *Phys. Rev. A*, vol. 54, no. 4, pp. 3824–3851, 1996, doi: 10.1103/PhysRevA.54.3824.

[51] 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, May 2007, Art. no. 190503, doi: 10.1103/PhysRevLett.98.190503.

[52] O. A. Collins, S. D. Jenkins, A. Kuzmich, and T. A. B. Kennedy, "Multiplexed memory-insensitive quantum repeaters," *Phys. Rev. Lett.*, vol. 98, Feb. 2007, Art. no. 060502, doi: 10.1103/PhysRevLett.98.060502.

[53] I. Beckenbach, "Matchings and flows in hypergraphs," Ph.D. dissertation, Dept. Math. Informat., Freie Univ., Berlin, Germany, 2019.

[54] W. Dai, T. Peng, and M. Z. Win, "Optimal remote entanglement distribution," *IEEE J. Sel. Areas Commun.*, vol. 38, no. 3, pp. 540–556, Mar. 2020, doi: 10.1109/JSAC.2020.2969005.

[55] G. Gallo, G. Longo, S. Pallottino, and S. Nguyen, "Directed hypergraphs and applications," *Discrete Appl. Math.*, vol. 42, nos. 2/3, pp. 177–201, 1993, doi: 10.1016/0166-218X(93)90045-P.

[56] M. Thakur and R. Tripathi, "Linear connectivity problems in directed hypergraphs," *Theor. Comput. Sci.*, vol. 410, nos. 27–29, pp. 2592–2618, 2009, doi: 10.1016/j.tcs.2009.02.038.

MOHAMMAD GHADERIBANEH received the B.S. degree in electrical engineering from Islamic Azad University, Urmia, Iran, in 2008, and the M.S. degree in telecommunication engineering from Shahid Beheshti University, Tehran, Iran, in 2011. He is currently working toward the Ph.D. degree in computer science with Stony Brook University, Stony Brook, NY, USA.

From 2012 to 2018, he was a Software and Telecommunication Engineer and developed wireless communication networks such as Terrestrial Trunked Radio. In 2022, he was an intern with Google as a Software Engineer and designed a machine learning model to improve YouTube's review process. His research interests include quantum networks and machine learning application in wireless networks.

CAITAO ZHAN received the B.S. degree in computer science and technology from the China University of Geosciences, Wuhan, China, in 2017. He is currently working toward the Ph.D. degree with the Department of Computer Science, Stony Brook University, Stony Brook, NY, USA.

He does research in the broad area of computer networks. His research interests include the intersection of wireless networks and machine learning, quantum communication, and quantum sensing.

HIMANSHU GUPTA received the B.Tech. degree in computer science and engineering from the Indian Institute of Technology Bombay, Mumbai, India, in 1992, and the M.S. and Ph.D. degrees in computer science from Stanford University, Stanford, CA, USA, in 1999.

He is currently a Professor of computer science with Stony Brook University, Stony Brook, NY, USA, where he has been a faculty since 2002. His recent research interests have included wireless networks, with a focus on free-space optical communication networks and spectrum management. His current research interests include quantum networks and communication, and distributed quantum algorithms.

C. R. RAMAKRISHNAN received the M.Sc. (Tech.) degree from the Birla Institute of Technology and Science, Pilani, India, in 1987, and the Ph.D. degree from Stony Brook University, Stony Brook, NY, USA, in 1995, both in computer science.

He is currently a Professor of computer science with Stony Brook University, where he has been a faculty since 1997. His recent research interests have included logic programming and verification, with a focus on analyzing properties of probabilistic programming. His current research interests include quantum networks and communication, and distributed quantum algorithms.